Redbooks
ibm.com/redbooks

IBM Software Defined Environment

Dino Quintero
William M Genovese
KiWaon Kim
Ming Jun MJ Li
Fabio Martins
Ashish Nainwal
Dusan Smolej
Marcin Tabinowski
Ashu Tiwary



Cloud

Power Systems

IBM

Redbooks



International Technical Support Organization

IBM Software Defined Environment

August 2015

SG24-8238-00

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (August 2015)

This edition applies to IBM Cloud Manager version 4.1.0.1, IBM PowerVC version 1.2.1.1, IBM Platform Resource Scheduler version 2.2, IBM UrbanCode Deploy with Patterns version 6.1.0.1.545068, IBM UrbanCode Deploy version 6.1.0.1.532639, IBM SAN Volume Controller 7000 version 6.4.0.0, IBM Spectrum Scale version 4.1.0.0 (formerly IBM General Parallel File System, GPFS), RedHat Enterprise Linux Server release 6.5 (Santiago), IBM AIX 7000 version 6.4.0.0, IBM PowerKVM version 2.1.0 build 17 Gold (pkvm2_1), IBM Virtual I/O Servers versions 2.2.3.3 and 2.2.3.2, Brocade SAN Switch FOS version 7.1.1, IBM Hardware Management Console version 8 release 8.1.0.0.

© Copyright International Business Machines Corporation 2015. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices ix
Trademarks x

IBM Redbooks promotions xiii

Preface xv
Authors. xv
Now you can become a published author, too xvii
Comments welcome. xvii
Stay connected to IBM Redbooks xviii

Chapter 1. Introduction to the IBM Software Defined Environment 1
1.1 Characteristics of the Software Defined Environment. 3
1.2 The “software-defined everything” vision 3
1.3 Infrastructure matters 6
1.4 Purpose of this Redbooks publication. 7
1.5 Scope of the SDE solution 7
1.6 Supporting technologies 9
1.6.1 Cloud workload orchestration and management. 10
1.6.2 IBM pattern engine strategy 14
1.6.3 SDE service orchestration. 18
1.6.4 IBM UrbanCode Deploy 22
1.7 Software Defined Compute 33
1.7.1 PowerVM 33
1.7.2 PowerKVM 34
1.7.3 Kernel-based virtualization 34
1.8 IBM z/VM. 34
1.9 Hyper-V 34
1.10 ESX 35
1.11 Software-defined storage 35
1.11.1 IBM SAN Volume Controller 35
1.11.2 IBM Storwize V7000 36
1.11.3 IBM XIV 36
1.11.4 IBM Spectrum Scale (was IBM General Parallel File System, or GPFS) 37
1.11.5 IBM Software Defined Network for Virtual Environments 38
1.12 Scenarios for an example company 39
1.12.1 Enable line-of-business cloud users 39
1.12.2 Business expansion: Advanced orchestration 41
1.13 Deploy and managing clouds 42
1.13.1 Cloud provider advanced automation and orchestration. 42
1.14 Benefits and outcomes 43

Chapter 2. Reference architecture framework and methods 47
2.1 Various definitions of reference architecture. 48
2.2 The Enterprise Technology Framework 49
2.3 Approaches and methods 52
2.3.1 The core 52
2.3.2 The hybrid core 53
2.3.3 Benefits of reference architectures 55

© Copyright IBM Corp. 2015. All rights reserved.

Chapter 3. SDE reference architecture 57
3.1 A closer look at SDE reference architecture 58
3.1.1 Relationship with the IBM Cloud Enabled Data Center and cloud computing reference architecture 59
3.1.2 SDE reference architecture, logical view 61
3.1.3 SDE reference architecture with OpenStack integration 62
3.2 SDE with OpenStack reference security architecture 69
3.2.1 IBM dynamic cloud security portfolio 69
3.3 Servers: x86, Power Systems, System z 77
3.3.1 x3500 7839 series 77
3.3.2 x3950 Series 7233 Series 78
3.4 Hypervisors xKVM, PowerKVM, PowerVM, z/VM, VMware 102
3.4.1 xKVM 104
3.4.2 KVM cloud network topology 108
3.4.3 PowerKVM 110
3.4.4 PowerVM 116
3.4.5 Architecture and solution context 118
3.4.6 Virtual I/O Server in relation to the PowerVM hypervisor and virtualization 120
3.4.7 Power Systems network virtualization 121
3.4.8 System z virtual machine, z/VM 126
3.5 Storage GPFS, GPFS Storage Server, XIV, V7K, DS8K 132
3.5.1 IBM Spectrum Scale 132
3.5.2 IBM software-defined storage vision within SDE 132
3.5.3 Spectrum Scale use cases 140
3.5.4 IBM Spectrum Scale architecture 141
3.5.5 System x GPFS Storage Server 145
3.5.6 IBM XIV storage 148
3.5.7 IBM Storwize V7000 storage systems 162
3.5.8 IBM System Storage DS8000 series (DS8K) 171
3.6 Examples 180
3.6.1 Three-tier commerce transactional workload 180
3.6.2 Big data analytics workloads 184
3.6.3 The IBM business analytics and optimization capabilities 187

Chapter 4. Cloud provider installation guidelines and implementation scenarios . . 195
4.1 Structure of the IBM Cloud Manager installation 196
4.2 Cloud installation 198
4.3 Cloud setup for this publication 198
4.3.1 Hardware and software setup for this publication 199
4.3.2 Cloud topology for this publication 201
4.4 Deployment server installation 202
4.4.1 IBM Cloud Manager hardware prerequisites 202
4.4.2 Create the virtual machine to host the Cloud Manager deployment server 203
4.4.3 Install Red Hat Enterprise Linux 203
4.4.4 Customize Red Hat Enterprise Linux to host the deployment server 203
4.4.5 Install IBM Cloud Manager 4.1 on the deployment server 207
4.4.6 Update the Cloud Manager to the latest level 214
4.5 Deploy the Controller node 218
4.5.1 Hardware prerequisites for the Controller node 218
4.5.2 Create the logical partition to host the Controller node 218
4.5.3 Install Red Hat Enterprise Linux 219
4.5.4 Customize Red Hat Enterprise Linux to host the Controller node 219
4.5.5 Network considerations and requirements 219

4.5.6 Configure network interfaces in the Controller node 220
4.5.7 Deploying the Controller node 225
4.6 Add KVM Compute nodes to the cloud 236
4.6.1 Verify software prerequisites for KVM Compute nodes 237
4.6.2 Enable KVM support on your hardware 237
4.6.3 Install Red Hat Enterprise Linux on the physical host 238
4.6.4 Customize RHEL to meet the IBM Cloud Manager requirements 238
4.6.5 Configure management and data network interfaces 238
4.6.6 Set up a Yum repository 242
4.6.7 Install KVM packages on the physical host 243
4.6.8 Deploy the KVM Compute node using the deployment server 245
4.7 Add PowerKVM Compute nodes to the cloud 253
4.7.1 Prerequisites for PowerKVM Compute nodes 254
4.7.2 Verify that your hardware supports PowerKVM 254
4.7.3 Enable PowerKVM in Advanced System Management Interface (ASMI) 255
4.7.4 Install IBM PowerKVM 255
4.7.5 Customize PowerKVM to meet the Cloud Manager requirements 255
4.7.6 Configuring management and data network interfaces 261
4.7.7 Deploy the PowerKVM Compute node using the deployment server 266
4.7.8 Enable the Kimchi console in nova configuration file (optional) 273
4.8 Add PowerVM Compute nodes to the cloud 274
4.8.1 Prerequisites for PowerVM Compute nodes 274
4.8.2 Set up Virtual I/O Servers 275
4.8.3 Set up the IBM Power Virtualization Center 284
4.8.4 Add PowerVM Compute nodes to the OpenStack cloud 297
4.9 Enable the self-service portal 311
4.9.1 Prepare the environment (env.json) file 312
4.9.2 Prepare the topology (topo.json) file 313
4.9.3 Use the knife command to update your cloud topology 315
4.9.4 Connect the self-service portal to your OpenStack cloud 316
4.9.5 Install the IBM Cloud Manager license 325
4.10 Network configuration in IBM Cloud Manager 326
4.10.1 Open vSwitch 327
4.10.2 Shared Ethernet Adapters 329
4.10.3 Network configuration overview 330
4.10.4 Create networks in IBM Cloud Manager 339
4.10.5 Network setup in the cloud prepared for this publication 350
4.11 Storage configuration in IBM Cloud Manager 357
4.11.1 Integrate Cinder with Logical Volume Manager 357
4.11.2 Integrate Cinder with SAN Volume Controller or Storwize V7000 367
4.11.3 Integrate Cinder with Power Virtualization Center (PowerVC) 379
4.11.4 Integrate Cinder with IBM Spectrum Scale 387
4.11.5 Integrate Cinder with Network File System 402
4.12 Manage images in IBM Cloud Manager 412
4.12.1 Manage images in KVM and PowerKVM Compute nodes 413
4.12.2 Manage images in PowerVM Compute nodes 431
4.13 Manage instances in IBM Cloud Manager 443
4.13.1 Manage instances by using Cloud Management dashboard 443
4.13.2 Manage instances using self-service portal 460
4.13.3 Manage instances using the command-line interface 467
4.13.4 Identify an instance in the Compute node 472
4.14 Manage flavors in IBM Cloud Manager 480
4.14.1 Create flavors for KVM and PowerKVM Compute nodes 480

4.14.2 Create flavors for PowerVM Compute nodes 482
4.15 Use SSH key pairs in IBM Cloud Manager 485
4.15.1 Create key pairs 486
4.15.2 Inject key pairs in instances 488
4.16 Developer tool installation: UrbanCode Deploy with Patterns 492

Chapter 5. IBM big data and analytics capabilities 507
5.1 Logical architecture 508
5.2 IBM big data and analytics portfolio 509
5.2.1 Data sources (1) 509
5.2.2 Analytical sources (2) 511
5.2.3 IBM actionable insights (3) 518
5.2.4 New and enhanced applications (4) 521
5.2.5 On-premises or hybrid clouds and as-a-service capabilities (5) 521
5.3 Big data use cases 523
5.3.1 Big data exploration use case 523
5.3.2 Enhanced 360° View of the Customer use case 524
5.3.3 For more examples 526

Chapter 6. Advanced cloud provider scenarios 527
6.1 IBM Platform Resource Scheduler 528
6.1.1 IBM Platform Resource Scheduler overview 528
6.1.2 The Enterprise Grid Orchestrator 529
6.1.3 Platform Resource Scheduler installation 530
6.1.4 Using PRS 533
6.1.5 Platform Resource Scheduler use example 539
6.2 Manual compute and storage expansion 552
6.2.1 Manual compute expansion 552
6.2.2 Manual storage expansion 553

Chapter 7. Cloud user deployment and use scenarios 563
7.1 Heat engine and HOT templates 564
7.1.1 HOT template structure 564
7.1.2 Creating a sample HOT template 567
7.1.3 Create the Stack in Cloud Management dashboard 580
7.2 IBM UrbanCode Deploy with Patterns 583
7.2.1 Creating the versioner agent in IBM UrbanCode Deploy 584
7.2.2 Connect IBM UrbanCode Deploy with Patterns to the OpenStack cloud 591
7.2.3 Deploying environments in UrbanCode Deploy with Patterns 598
7.2.4 Creating components and applications in UrbanCode Deploy 609
7.2.5 Exporting and importing components from UrbanCode Deploy 628
7.2.6 Deploying environments in UrbanCode Deploy with Patterns using components created in UrbanCode Deploy 635
7.2.7 Importing blueprint definitions in IBM UrbanCode Deploy with Patterns 648
7.2.8 Required corrections in the cloud setup for this publication 653
7.3 IBM BigInsights big data analytics software 655
7.4 Acme Air scenario 668
7.4.1 DevOps Pro (Maureen) provisions an application to the hybrid cloud 669
7.4.2 Infrastructure Administrator (Michael) inspects and configures the hybrid cloud 674
7.4.3 Cloud users interact with a mobile application 680
7.4.4 Cloud user interactions with the reservations application 681
7.4.5 AcmeAirDataScale-PowerKVM.yaml file (HOT template) 684
7.4.6 Section summary 694

7.5 High availability for applications 695
7.5.1 Introduction to OpenStack High Availability 695
7.5.2 Stateless versus stateful services 696
7.5.3 Active/passive configuration 696
7.5.4 Active/active configuration 696

Chapter 8. Recently released architecture and future direction 699
8.1 Software-defined infrastructure: Next generation 700
8.1.1 Workload engines 701
8.1.2 Resource management 704
8.1.3 OpenStack: Juno, latest release information 707
8.1.4 Next-generation architecture for information and insights 709

Appendix A. IBM Software Defined Network for Virtual Environments 711
Software-defined networking overview and relationship to SDE 712
Limitations of current networking technologies 714
Vendor dependence 715
Benefits of software-defined networking 717
The relationship between SDE and SDN 720
SDN reference architecture and standards 723
SDN reference architecture 723
IBM Software Defined Network for Virtual Environments context 730
Introduction to IBM SDN VE 730
Product versions 731
Elements of the IBM SDN VE solution 732
IBM SDN scenarios and use cases 738
The enterprise 738
Carriers and service providers 738
SDN installation process of the SDN VE and DOVE components 741
Introduction 741
Install and configure the Dove Management Console 742

Appendix B. Cloud computing review 775
Types of clouds 776
Public cloud 776
Private cloud 777
Hybrid cloud 777
Dynamic clouds and how they differ from hybrid clouds 779
Types of cloud services 781
Software as a service (SaaS) 782
Platform as a Service (SaaS) 782
Infrastructure as a service (IaaS) 782
How technical computing fits into cloud deployment models 783
Business value: Technical computing and HPC clouds 784
Cloud workload adoption and deployment model choice 788
Six steps toward cloud computing 788
Create the IT strategy and roadmap 789
Assess and select workloads for the cloud 789
Determine the cloud delivery model 790
Determine the value 790
Establish the architecture 791
Implement the IT strategy, roadmap, and cloud services 791
The result: Dynamic cloud through Software Defined Environments 791

Related publications 795
IBM Redbooks 795
Other publications 795
Online resources 795
Help from IBM 796

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright IBM Corp. 2015. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Active Cloud Engine®
Active Memory™
AIX®
AppScan®
BigInsights™
Bluemix™
Cognos®
Concert™
CPLEX®
DataStage®
DB2®
developerWorks®
DS4000®
DS5000™
DS8000®
Easy Tier®
EnergyScale™
FICON®
FlashCopy®
Global Business Services®
Global Technology Services®
GPFS™
Guardium®
IBM®
IBM Concert™
IBM Elastic Storage™
IBM Flex System®
IBM Flex System Manager®
IBM MobileFirst™
IBM PureData™
IBM SmartCloud®
IBM Spectrum™
IBM Spectrum Storage™
IBM UrbanCode™
IBM Watson™
ILOG®
Information on Demand®
Informix®
InfoSphere®
Insight™
Insights™
Linear Tape File System™
LSF®
Micro-Partitioning®
Netcool®
OMEGAMON®
Orchestrate®
POWER®
Power Architecture®
POWER Hypervisor™
Power Systems™
POWER6®
POWER6+™
POWER7®
POWER7+™
POWER8™
PowerHA®
PowerLinux™
PowerPC®
PowerVM®
PureData®
PureFlex®
PureSystems®
QRadar®
Rational®
Real-time Compression™
Redbooks®
Redpaper™
Redbooks (logo) ®
SPSS®
Storwize®
Symphony®
System i®
System Storage®
System z®
System z10®
Tivoli®
TM1®
Trusteer®
VTAM®
WebSphere®
X-Force®
XIV®
z/Architecture®
z/OS®
z/VM®
z/VSE®
z10™
zEnterprise®

The following terms are trademarks of other companies:

Evolution, Inc., and Inc. device are trademarks or registered trademarks of Kenexa, an IBM Company.

SoftLayer, and SoftLayer device are trademarks or registered trademarks of SoftLayer, Inc., an IBM Company.

Trusteer Apex, and Trusteer logo are trademarks or registered trademarks of Trusteer, an IBM Company.

Intel, Intel Xeon, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

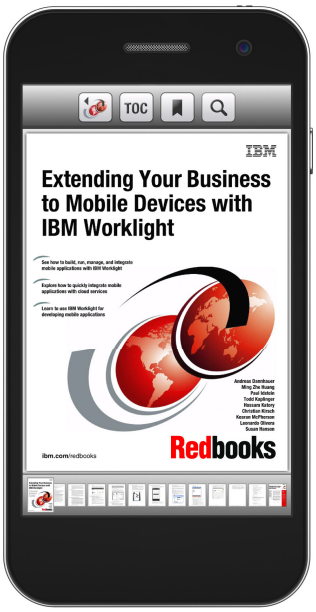
Other company, product, or service names may be trademarks or service marks of others.

Find and read thousands of IBM Redbooks publications

- Search, bookmark, save and organize favorites
- Get up-to-the-minute Redbooks news and announcements
- Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App

iOS **Download Now** Android



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks
About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

This IBM® Redbooks® publication introduces the IBM Software Defined Environment (SDE) solution, which helps to optimize the entire computing infrastructure—compute, storage, and network resources—so that it can adapt to the type of work required. In today's environment, resources are assigned manually to workloads, but that happens automatically in a SDE. In an SDE, workloads are dynamically assigned to IT resources based on application characteristics, best-available resources, and service level policies so that they deliver continuous, dynamic optimization and reconfiguration to address infrastructure issues. Underlying all of this are policy-based compliance checks and updates in a centrally managed environment.

Readers get a broad introduction to the new architecture. Think integration, automation, and optimization. Those are enablers of cloud delivery and analytics. SDE can accelerate business success by matching workloads and resources so that you have a responsive, adaptive environment. With the IBM Software Defined Environment, infrastructure is fully programmable to rapidly deploy workloads on optimal resources and to instantly respond to changing business demands.

This information is intended for IBM sales representatives, IBM software architects, IBM Systems Technology Group brand specialists, distributors, resellers, and anyone who is developing or implementing SDE.

Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Austin Center.

Dino Quintero is a Complex Solutions Project Leader and an IBM Senior Certified IT Specialist with the ITSO in Poughkeepsie, New York. His areas of knowledge include enterprise continuous availability, enterprise systems management, system virtualization, technical computing, and clustering solutions. He is an Open Group Distinguished IT Specialist. Dino holds a Master of Computing Information Systems degree and a Bachelor of Science degree in Computer Science from Marist College.

William M Genovese is an Executive Architect, Client Technical Leader and Advisor in IBM Sales and Distribution, Financial Services sector. Bill has been with IBM for more than 10 years (IBM Sales and Distribution, IBM Systems Technology Group, IBM Global Technology Services®, and IBM Global Business Services®), and has more than 21 years experience working with financial services clients. Bill is also an IBM Executive (Senior) Certified IT Architect, an IBM Certified Senior Technology Consultant, and IBM Certified Level 1 IT Specialist, with specializations in enterprise (technical) architecture, systems engineering, and application architecture.

KiWaon Kim is an IBM AIX® development support specialist with IBM Global Technical Services in South Korea. His areas of expertise are in AIX, networking, and IBM VTAM®. KiWaon is an IBM Certified Level 2 IT Specialist.

© Copyright IBM Corp. 2015. All rights reserved.

Ming Jun MJ Li is a senior Technical Architect and Cloud Technical Lead in the Systems Technology Group, in the Great China Group. He has 15 years of experience in a broad set of areas, including software development, solution design, and technical support. His areas of expertise include enterprise IT architecture, virtualization, cluster file systems, and OpenStack-based cloud solutions. He holds a Master's degree in software engineering from Tsinghua University.

Fabio Martins is a Senior Software Support Specialist with IBM Technical Support Services in Brazil. He has worked at IBM for 10 years. His areas of expertise include IBM AIX, IBM PowerVM®, IBM PowerKVM, IBM PowerHA®, PowerVC, IBM PureSystems®, IBM DS8000®, IBM DS5000™, IBM DS4000®, Linux, and Brocade SAN switches and directors. He is a Certified Product Services Professional in Software Support Specialization and a Certified Advanced Technical Expert on IBM POWER® Systems. He has worked extensively on IBM Power Systems™ for Brazilian customers, providing technical leadership and support, including how-to questions, problem determination, root cause analysis, performance concerns, and other general complex issues. He holds a Bachelor's degree in Computer Science from Universidade Paulista (UNIP).

Ashish Nainwal is a Managing Consultant for Power Systems in the Systems Technology Group Lab Services in the ASEAN region. With almost 10 years experience in Power Systems, he has expertise in the Power suite of software, including PowerVM, PowerHA, cloud products, and Power security. He is a authorized PowerCare consultant and has conducted performance, availability, and security engagements for IBM clients. He has co-written IBM Redbooks publications about PowerHA and articles for IBM developerWorks® and has been granted a patent for complex security architecture. He has a Bachelor's in Technology degree from the National Institute of Technology, India and an MBA from Symbiosis, India.

Dusan Smolej is a Senior IT Architect for Global Technology Services at IBM in the Czech Republic. He has more than 19 years of experience in the IT industry and 13 years with IBM. He works as an Integration Architect and was a Lead Architect for a successful complex engagements for local and international clients. His areas of expertise include service-oriented architecture, enterprise application integration, digital archiving, cloud computing, grid computing, performance analysis, and optimization. He holds an M.Sc. degree in Electrical Engineering from the Technical University of Kosice, Slovakia.

Marcin Tabinowski works as an IT Specialist in Systems Technology Group Lab Services in Poland. He has more than eight years of experience in designing and implementing IT solutions that are based on storage and IBM Power Systems. His main responsibilities are designing, consulting, implementing, and documenting projects, including storage systems, storage area networks, Power Systems, disaster recovery, virtualization, and data migration. Presales, post-sales, and training are also part of his everyday duties. Martin holds many certifications that span different IBM storage products and Power Systems. He also holds a Master of Science in Computer Science from Wroclaw University of Technology, in Poland.

Ashu Tiwary is a Senior Technical Staff Member with the IBM Systems Field Assist Support Team (FAST), focusing on issues involving multiple IBM and third-party products. He has more than 20 years of experience in a variety of IT areas, including systems administration, enterprise systems management, technical support, development, and architecture.

Thanks to the following people for their contributions to this project:

Rich Conway, David Bennin, and Donald Brennan
IBM International Technical Support Organization, Austin Center

Yu Zhong Cao, Zi Lian Ji, and Hong Jin Wei
IBM China

Rafael Folco and Pedro Principeza
IBM Brazil

Christopher Ansari, Jay Bryant, John Kasperski, Kevin Rudd, Johnny Shieh, David Starr, Richard Theis, Drew Thorstensen, and Dave Willoughby
IBM USA

Yong Feng
IBM Canada

Now you can become a published author, too

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time. Join an ITSO residency project and help write a book in your area of expertise, although honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form:

  ibm.com/redbooks

- Send your comments by email:

  redbooks@us.ibm.com

- Mail your comments:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction to the IBM Software Defined Environment

In a world awash in data and connectivity, new mobile and social engagement models, along with intensive workloads such as big data analytics, are becoming critical to business success. IT is moving out of the back office and becoming central to business strategy. Expectations are high. Today's workloads demand extreme scalability, real-time responsiveness, and continuous availability. In many cases, however, IT organizations are unable to keep up due to rapidly expanding business needs, growing infrastructure complexity, and resource constraints.

Organizations are under increasing pressure to more quickly create and deploy more engaging sets of applications to meet ever-growing business requirements. Lines of business can no longer wait for traditional IT deployment cycles, and they must use emerging, rapid deployment techniques, including continuous deployment in many cases. Application and infrastructure deployment and lifecycle management must become self-service and turn-key for rapid, error-free, continuous deployment and management.

This IBM Redbooks publication describes the IBM Software Defined Environment solution, which uses IBM products, an open design, and goes a step further to show a complete integration of these products that provides significant value to a wide range of IBM clients. It is best to use the full integration shown here for maximum value, but the solution is modular and open, so not all components are required and some components can be replaced with non-IBM equivalent components. Having OpenStack at its core enables integration of components from the OpenStack ecosystem.

This book explains the SDE technology solution, its architecture, and the solution context, and provides implementation details. This chapter includes the following sections:

- 1.1, "Characteristics of the Software Defined Environment" on page 3
- 1.2, "The "software-defined everything" vision" on page 3
- 1.3, "Infrastructure matters" on page 6
- 1.4, "Purpose of this Redbooks publication" on page 7
- 1.5, "Scope of the SDE solution" on page 7
- 1.6, "Supporting technologies" on page 9
- 1.7, "Software Defined Compute" on page 33

© Copyright IBM Corp. 2015. All rights reserved.

- 1.8, “IBM z/VM” on page 34
- 1.11, “Software-defined storage” on page 35
- 1.12, “Scenarios for an example company” on page 39
- 1.13, “Deploy and managing clouds” on page 42
- 1.14, “Benefits and outcomes” on page 43

1.1 Characteristics of the Software Defined Environment

This chapter provides an introduction to the IBM Software Defined Environment solution.

Investments in enterprise virtualization, centralized administration, and hardware with enhanced management and optimization functions have laid the groundwork for a new era in business responsiveness. The technical capabilities now exist, so the time is right to take the next step by using those technologies to enable a fully programmable IT infrastructure that is able to sense and respond to workload demands *automatically*. IBM calls this idea a *Software Defined Environment*. It is a new approach for holistic, simplified IT management in which software provisions and configures entire infrastructures based on real-time workload needs. Software Defined Environment (SDE) is a term coined by IBM for its "software-defined everything" vision. Software Defined Environments encompass more than just the infrastructure use model of cloud storage and services. They also affect how developers interact and use application-aware infrastructures to deploy their next generations of applications.

An SDE optimizes the entire computing infrastructure—compute, storage, and network resources—so that it can adapt to the type of work required. In today's environment, resources are assigned manually to workloads, but that happens automatically in a SDE. In an SDE, workloads are dynamically assigned to IT resources based on application characteristics, best-available resources, and service-level policies that deliver continuous, dynamic optimization and reconfiguration to address infrastructure issues. Underlying all of this are policy-based compliance checks and updates in a centrally managed environment.

By dynamically assigning workloads to IT resources based on a variety of factors, including the characteristics of specific applications, the best-available resources, and service-level policies, a Software Defined Environment can deliver continuous, dynamic optimization, and reconfiguration to address infrastructure issues.

1.2 The "software-defined everything" vision

Software-defined everything is a phrase that serves to group a variety of software-defined computing technologies into one over-arching framework and architecture. The umbrella of software-defined everything technologies includes, among other terms, software-defined networking (SDN), software-defined computing, software-defined data centers (SDDC), software-defined storage (SDS), and software-defined storage networks.

With software-defined everything, the computing infrastructure is virtualized and delivered as a service. In a software-defined everything environment, management and control of the networking, storage, and data center infrastructure is automated by intelligent software rather than by the hardware components of the infrastructure.

So, think *integration*, *automation*, and *optimization*. Those enable cloud delivery and analytics. SDE can accelerate business success by matching workloads and resources so you have a responsive, adaptive environment.

With the IBM Software Defined Environment, infrastructure is fully programmable to rapidly deploy workloads on optimal resources and to instantly respond to changing business demands:

Software Abstracted and virtualized IT infrastructure resources are managed by software.

Defined Applications automatically define infrastructure requirements and configuration.

Environment An IT infrastructure extends multiple environments to go beyond the data center.

As Figure 1-1 shows, SDE also represents the next step in the evolution of infrastructure as a service (IaaS), by providing a more dynamic and agile infrastructure, and more fine-grained automated policy-based controls for workload definition, optimization, and orchestration, in the cloud operating environment. SDE enables and supports more detailed definition and abstraction of services, and composition patterns for traditional workloads in the private cloud data center and enables more robust application programmatic interface (API) based integration services to the external ecosystem and (software as a service) SaaS environments or public clouds. This is important, as in these emerging hybrid (private and public) cloud deployment architectures, workload demand is dynamic, and SDE provides an underpinning architecture and elasticity to meet these new business demands.

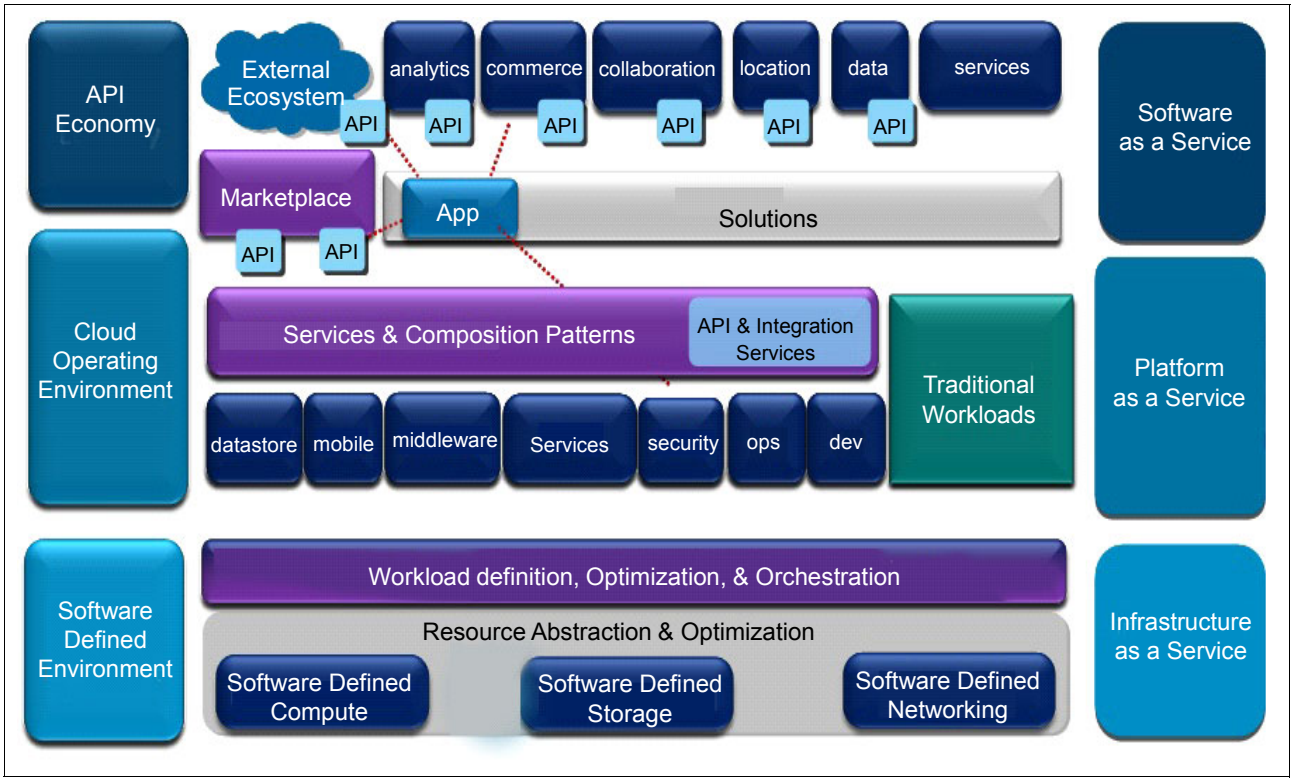


Figure 1-1 SDE Represents the Evolution® of infrastructure as a service (IaaS)

Based on the traditional view of IT Virtualization, infrastructure can typically be characterized through and by the following challenges:

- Hardware centric; bottoms up
- Server, storage, and network silos
- Homogeneous compute silos
- Managing large numbers of individual systems
- Managing virtual resource as you manage hardware
- Management with extensive manual process intervention

A software-defined view of IT virtualization, in contrast, has several advantages:

- Is more workload aware, and provides a top down view
- Uses server, storage, and network integration (SDI) for responsiveness
- Uses heterogeneous compute federation
- Manages pools of systems as a single system
- Truly uses virtualization to manage IT
- Is managed by advanced programmed automation

Through SDE, clients will maintain transactional integrity for the systems of record, while increasing agility and scale for the systems of engagement through flexible, efficient, programmable, infrastructure.

SDE provides the following value for clients:

- Responsiveness through flexible and elastic infrastructure and radically improved time to value
- Resiliency and stability providing repeatability, consistency, and improved availability through automation
- Cost reduction through higher use and efficiencies, and significantly reduced energy costs
- Security isolation supporting multitenancy

With SDE, preferred practices are captured with workload-aware orchestration and optimization “patterns” for software and infrastructure. Figure 1-2 on page 6 shows how patterns are used to describe both the software and infrastructure components, relationships, policies and service levels, used to provision and manage resources for a business service. Automation and optimization constitutes is a distributed and collaborative approach that is performed with the context of the workloads, and business services. Policies and service levels are derived from the business services, and provided in the context of the resources being managed by each component in the system (software, server, storage, network).

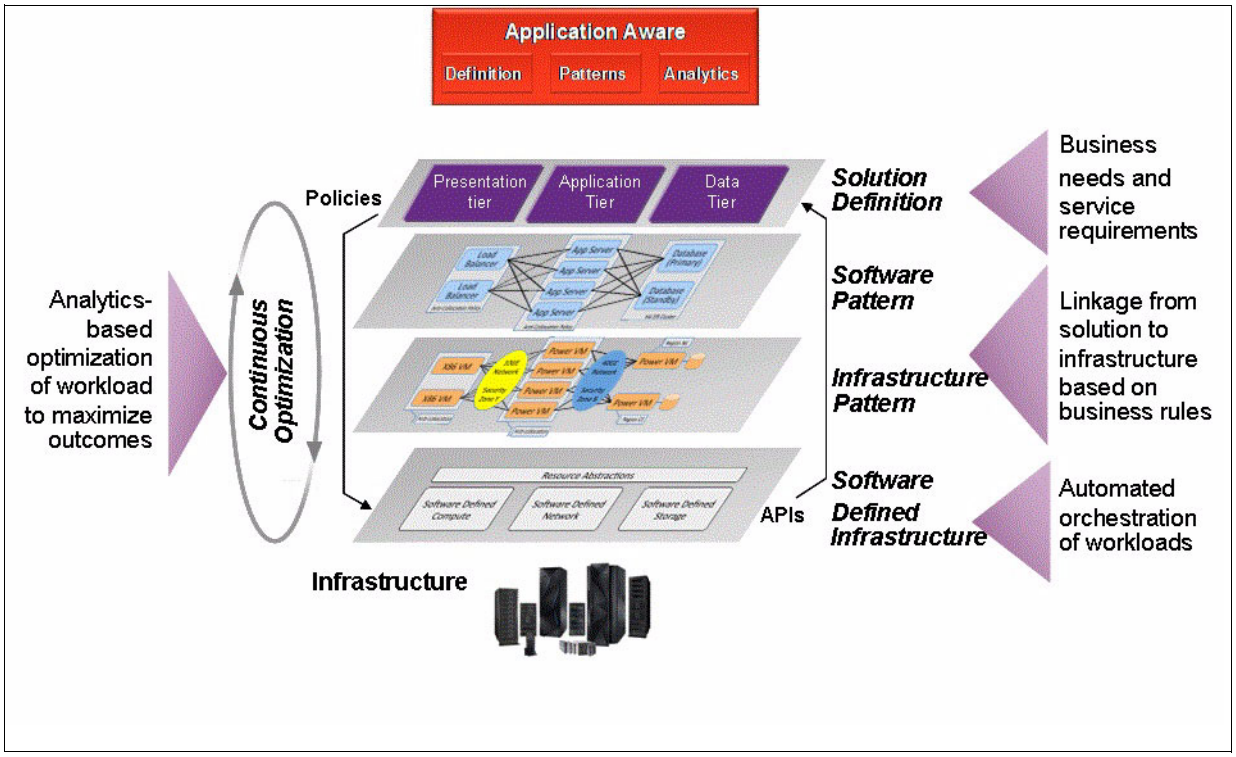


Figure 1-2 Workload optimization through orchestration -- application aware and resource smart

A Software Defined Environment changes the rules that govern the way technology is deployed because business needs are linked directly to infrastructure management. Business objectives are literally programmed in, spelled out in a *solution definition*—a description of what IT must deliver to the business. Everything from business policies to availability and compliance requirements need only be input once.

These business “ground rules” are used to identify the required service components and through them, the necessary infrastructure resources. It’s a combination of reusable, rapidly deployable patterns of expertise, or recommended configurations, that define how the business need will be met. The solution definition is thus dynamically tied to its optimal server, storage and network resources based on preferred practices and business rules.

With the stage set, the software-defined infrastructure automatically orchestrates all resources to meet workload requirements in near real time, using infrastructure analytics to optimize outcomes. Scaling happens in direct response to shifting demand as driven by business workloads. Appropriate server, storage and networking assets are assigned as needed to maintain service levels—all without manual intervention. The Software Defined Environment framework transforms static infrastructure into a dynamic, continuously optimized, workload-aware virtualized resource that enables line-of-business users to take advantage of IT as and when needed without delay, thus enabling far greater business agility.

1.3 Infrastructure matters

To enable a Software Defined Environment, the management tools that link rules, policies and definitions to control hardware are critical, of course. However, infrastructure technologies play a vitally important role in maximizing the value of the approach.

In a Software Defined Environment workloads do not depend on specific systems. That's a major strength of the approach and access point for many clients; everything runs on the most appropriate combination of hardware assets, whatever they may be, and in many cases using existing infrastructure investments and building blocks as a foundational starting point. Thanks to open standards, the enterprise decision-markers are free to choose the infrastructure. Investments are protected and there is no need to be locked in to any one vendor's technology. In addition, responsiveness to new demands is greatly enhanced because assets can be readily repurposed.

Note: The infrastructure adapts to business need, rather than the other way around. This is a key technology in this IBM solution.

In the next section, 1.6.1, "Cloud workload orchestration and management" on page 10, OpenStack is described as an underpinning and foundation open cloud technology that is used for Software Defined Environments.

Although it benefits from open systems, a Software Defined Environment also requires that the physical infrastructure become aware of, and responsive to, workload requirements. IBM is a leader in the advanced IT that makes this possible. For example, infrastructure analytics are essential to optimize workload execution. Technologies such as IBM zAware use machine learning to monitor and analyze hardware activity, logging and compiling usage patterns so that the environment can know how to allocate computing resources for optimal results. Other automation technologies such as IBM Easy Tier® use the same business logic principles that underlie the Software Defined Environment to operatively place the most-used data on the fastest storage resources.

Technologies like these highlight a key concept about the Software Defined Environment. It's a mosaic (very much like a data centers technology)—a flexible approach made up of many different solutions working in concert. Using the right technologies can make a real difference.

1.4 Purpose of this Redbooks publication

The purpose of the IBM Redbooks publications is to provide clients an initial detailed view into the IBM SDE solution in a delivered lab environment based on real world business use cases, from a cloud provider perspective, and a cloud business user perspective. It is useful for a wide variety of readers: Line of business management and users, enterprise architects, application architects,and developers, cloud solution architects and specialists, infrastructure architects, and systems and service management architects and specialists.

It provides detail from business requirements and use case scenarios through architectural design, to implementation details for the deployment and solution architecture.

1.5 Scope of the SDE solution

As stated earlier in 1.3, "Infrastructure matters" on page 6, the SDE approach is very flexible in terms of hardware and infrastructure platforms. In this book, the IBM SDE solution is described, which uses IBM products, an open design, and goes a step further to show a complete integration of these products that provides significant value to a wide range of clients.

It is best to use the full integration shown in the Redbooks publications for maximum value, but its true that the solution is modular and open, so that not all components are required, and some components can be replaced with non-IBM equivalent components. Having OpenStack at its core, enables integration of components from the OpenStack ecosystem.

The IBM SDE solution described in this solution brief is up and running in IBM's internal development lab, but is not available as a packaged offering.

All key elements described in this IBM SDE solution brief are available as stand-alone products. The IBM OpenStack distribution is available as part of the IBM Cloud Manager product and is also used in other IBM products, such as IBM Cloud Orchestrator and PowerVC. Some integration aspects of this solution have not yet been released in IBM products, but will be used to validate and gain feedback on these concepts with clients and partners.

The elements and key themes of the IBM SDE solution are focused, in two major scenarios:

- One focused on the cloud user, those who develop and manage line-of-business applications in the cloud
- One focused on the cloud provider, those that build and manage clouds

Also very important to note is that the scope of the solution and the deployment architecture options for both the software and infrastructure for this book are ideally suited for the specific scenarios that are demonstrated to support cloud providers and enable line of business cloud users. One or more deployment architectures are referred to as we go through the examples. The scenarios are explained in 1.12.1, "Enable line-of-business cloud users" on page 39, and in 1.13, "Deploy and managing clouds" on page 42. Figure 1-3 on page 9 illustrates the primary logical deployment architecture for the SDE solution deployed in Chapter 3 of this book.

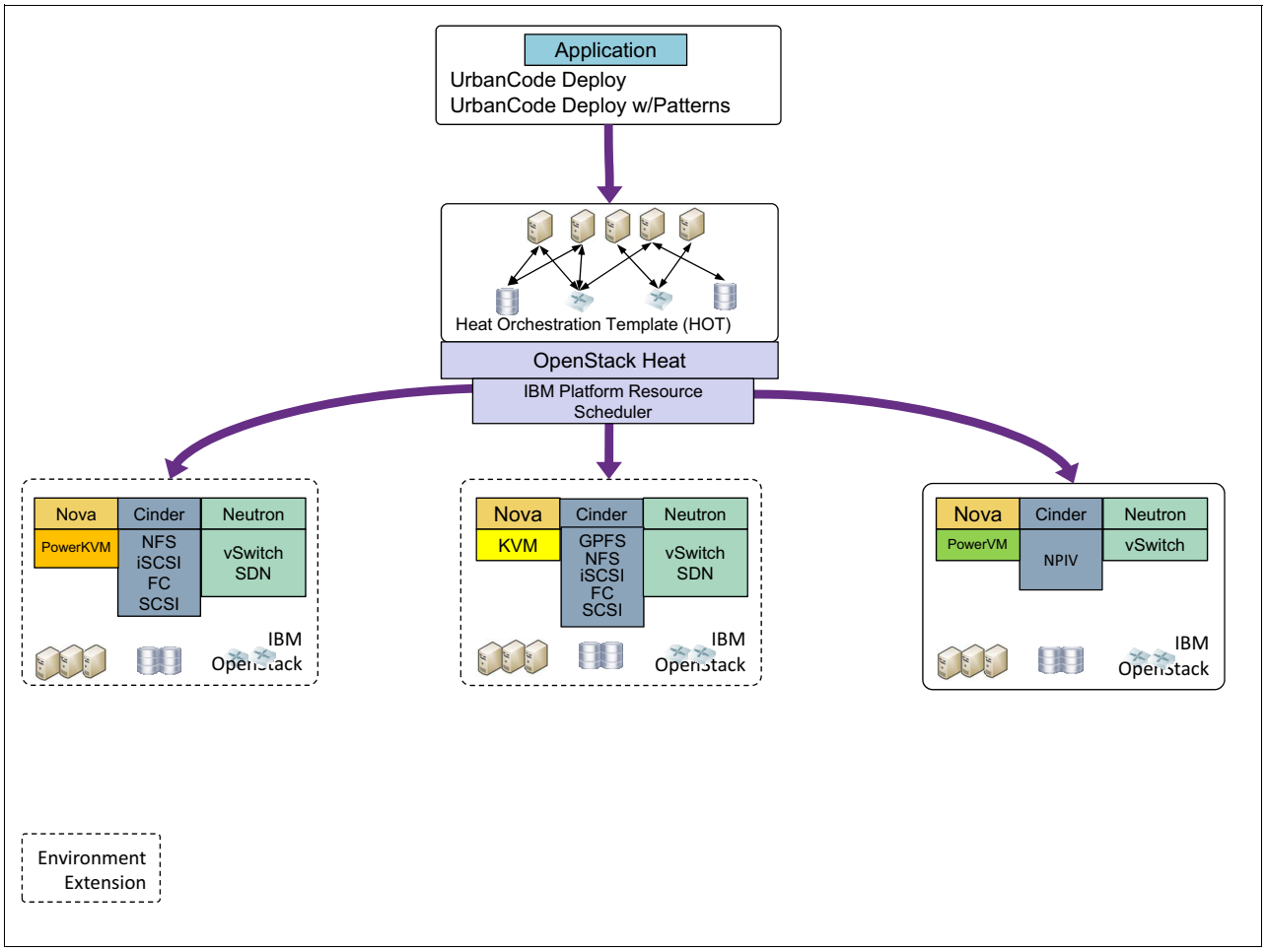


Figure 1-3 SDE solution deployment architecture

The supporting technologies for the deployment architecture and solution are explained in 1.6, “Supporting technologies” on page 9.

1.6 Supporting technologies

Software Defined Environment is a very broad term which can span multiple different products, technologies and solutions. Defining what needs to be achieved determines the approach and selection of correct components. To bring order to the Software Defined Environment, these components can be put in 4 main categories:

- Cloud workload orchestration and management
- Software-defined compute (SDC)
- Software defined storage (SDS)
- Software-defined networking (SDN)

In this section several SDE components of each category are described.

1.6.1 Cloud workload orchestration and management

Cloud workload orchestration and management is the foundation of Software Defined Environments. SDE with OpenStack provides a platform for cloud computing and enables creation, configuration and deployment of private, public or hybrid clouds, application workloads to each deployment topology, along with all other software-defined infrastructure (SDI) technologies, such as software-defined compute (SDC, also known as *virtualization*), software-defined storage (SDS), and software-defined networking (SDN). The main supporting technologies and products are described in the subsections that follow.

OpenStack

The OpenStack project started in 2011 as the main component in cloud management. It is an open source, cloud computing platform developed by the large number of companies and developers all around the world. It's based on the Apache 2.0 license and runs on variety of Linux operating systems. OpenStack is a global collaboration of developers and cloud computing technologists that seek to produce a ubiquitous infrastructure as a service (IaaS) open source computing platform for public and private clouds. The OpenStack ecosystem is growing and consists of more than 190 companies, 5,600 members, and 3,600 developers. There have been more than 200,000 downloads.

OpenStack is very flexible in terms of configuration options, and you can choose whether or not to implement several services offered by the software. Each component is easily configured through the application programming interface (API) that the tool provides.

Another reason for clients to adopt OpenStack is that it supports different hypervisors (Xen, VMware or kernel-based virtual machine [KVM] for instance) and several virtualization technologies (such as bare metal or high-performance computing). This is important, because hypervisors are becoming a commodity. We will continue to optimize the IBM cloud platform through flexibility and control, with added differentiating capabilities (optimized workloads, image management, DevOps, orchestration).

OpenStack architecture involves the set of services shown in Table 1-1.

Table 1-1 OpenStack services

Service	Code name	Description
Compute	Nova	Acts as the system controller; allows the provisioning and management of virtual machines (VMs) using the disk images available on OpenStack.
Object storage	Swift	Provides an object storage and reading service.
Block storage	Cinder	Provides permanent block storage to the running VMs; those volumes can easily be plugged and unplugged from the instances.
Network management	Neutron	Enables connectivity between the VMs through virtual networks.
Authentication	Keystone	Enables authentication of the OpenStack users to allow them to use certain services.

Service	Code name	Description
Image service	Glance	Allows disk image storage and retrieval. These images are used by OpenStack compute service when provisioning the VMs.
Billing service	Ceilometer	Monitors and measures the use of cloud resources (use of VMs network traffic) for billing or benchmarking.
Cloud template	Heat	Enables several cloud composite applications to be launched and managed, allowing the user to automate a cloud deployment from configuration text files.
Dashboard	Horizon	Provides a user interface to configure and deploy all the services previously described; from this interface, the user is able to create, run, and manage instances, their resources, and their connectivity.

Figure 1-4 illustrates the relationships of the OpenStack conceptual architecture and services.

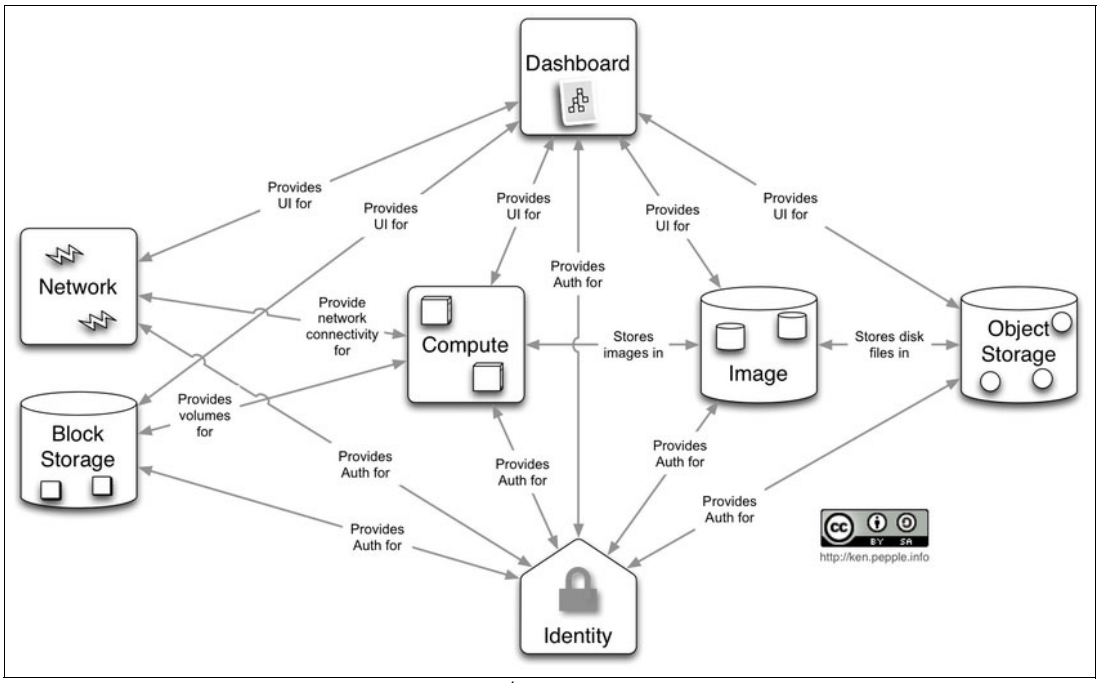


Figure 1-4 OpenStack conceptual architecture[1]

For more information, see the OpenStack website:

http://www.openstack.org

[1] Image courtesy of Ken Pepple, http://ken.pepple.info/

IBM OpenStack Heat orchestration

Heat is one of the newer core projects in the OpenStack community, started in March 2012. Taking into account this fairly short history, it became one of the most rapidly advancing OpenStack projects.

Heat is an orchestration engine that comes with OpenStack. It provides a way of using OpenStack services without a need for an in-depth knowledge of how to deploy individual components of the infrastructure, such as storage volumes or secured networks. It enables the deployment of complex infrastructure environments and multitier applications in a simple and repeatable way. It provides this simplicity by aggregating the APIs from the underlying OpenStack services into a single template text file that can have different versions. The template file contains an abstract configuration of all core services and can be treated like source code. That's why Heat templates are sometimes referred to as "infrastructure as code." The templates can be nested.

From the OpenStack architecture point of view, Heat is layered above Nova, Glance, Cinder, and Neutron services. It uses the APIs of these services to provision the infrastructure and applications described in the template file. Heat, in turn, provides a REST interface that can be used by cloud management layer to manage the infrastructure and the application lifecycles.

Originally, Heat used the Amazon Cloud Formation (CFN) template format. But recently, a native OpenStack specification named Heat Orchestration Template (HOT) was implemented. Over time, we expect that HOT will replace the proprietary CFN format.

HOT overview

HOT constitutes a critical piece of the IBM strategy to provide a single comprehensive and open specification for describing both infrastructure and workload patterns. Future HOT enhancements will enable users to define various deployment and runtime policies. In the IBM SDE solution, the HOT document is automatically generated by DevOps and orchestration tools. We briefly analyze the key components of that template.

A HOT template is written by using YAML and has three major sections. (YAML is a simple language for describing data objects by using human-friendly syntax.)

Note[a]**:** YAML (jæm?l, rhymes with camel) is a human-readable data serialization format that takes concepts from programming languages such as C, Perl, and Python, and ideas from XML and the data format of electronic mail (RFC 2822). YAML is a recursive acronym for "YAML Ain't Markup Language." Early in its development, YAML was said to mean "Yet Another Markup Language," but it was then reinterpreted to distinguish its purpose as data-oriented, rather than document markup.

a. Source: Wikipedia, http://en.wikipedia.org/wiki/YAML

Input parameters

The input parameters are used to customize the template at deployment time. For each parameter, you can define several attributes, such as data type, an optional default value, and an optional description. You can also restrict the values of input parameters that users can supply by adding a constraints section to the parameter definition.

Consider the following excerpt in Example 1-1 from the HOT template used to deploy a sample application.

Example 1-1 Template to define the volume size

```
volume_size:
   type: number
   description: Size of the volume to be created.
   default: 100
   constraints:
   - range: { min: 1, max: 1024 }
description: must be between 1 and 1024 Gb.
```

In this example, we define a parameter of type number called `volume_size` with the default size of 100 GB and the allowable size ranging between 1 and 1024 GB.

Resources

The Resources section provides an abstract description of the actual resources that are created by Heat when the template is deployed. This may include Nova servers (instances), networks, ports, routers, security groups, cinder storage volumes, or any other allowable resource type. See the code snippet in Example 1-2.

Example 1-2 Template to define the Cinder storage volume

```
bi_controller_cinder_volume:
   type: OS::Cinder::Volume
   properties:
   size: { get_param: volume_size }
   volume_type:{ get_param: volume_type }
   availability_zone: { get_param: availability_zone }
```

Output

This section defines parameters that are available to the caller via the API or command-line client after the template is deployed. This may include parameters such as the host name of deployed instances or URLs of web applications that are deployed as part of the template.

See example output in Example 1-3.

Example 1-3 Template to define parameters for the web applications

```
outputs:
   BIAdminURL:
   description: URL for bi_controller
   value:
   str_replace:
   template: http://host/bi_controller
   params:
host: { get_attr: [bi_controller, first_address] }
```

In this code snippet, a `BIAdminURL` parameter will be returned. The actual value of the URL is constructed by using the IP address of the first network interface assigned to the instance by OpenStack. In our case, it is the public interface of the bi_controller node.

OASIS-Topology and Orchestration Specification for Cloud Applications

The OASIS-TOSCA (Topology and Orchestration Specification for Cloud Applications) Technical Committee works to enhance the portability of cloud applications and services. TOSCA will enable the interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behavior of these services (for example, to deploy, patch, or shut down), independent of the supplier that creates the service and any particular cloud provider or hosting technology. TOSCA also makes it possible for higher-level operational behavior to be associated with cloud infrastructure management.

By increasing service and application portability in a vendor-neutral ecosystem, TOSCA will enable the following capabilities:

- Portable deployment to any compliant cloud
- Smoother migration of existing applications to the cloud
- Flexible bursting (user choice)
- Dynamic, multi-cloud provider applications

IBM is developing its orchestration and pattern engines to be OASIS TOSCA-compliant to further promote open cloud technologies for clients.

For more information about the charter, objectives, and goals, see OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) Technical Committee web page:

https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

There are also two videos that provide a very good “Understanding of OASIS TOSCA” and an “Overview of the Growing TOSCA Ecosystem”:

http://www.youtube.com/playlist?list=PLaYKtNo_BitZXdvyNDwBi29OIHxdi459v

1.6.2 IBM pattern engine strategy

In Software Defined Environments and open cloud technology computing orchestration and automation patterns, “content is king.” The vendor with the most content, best content, and, most importantly, most flexible content, will win. This is not a small market by any stretch of the imagination. Existing IBM patterns produced $50 million in revenue in 2013 (USD), while IBM Business Partners helped generate contributions of more than US $100 million for PureApplications revenue. As such, this is a key success factor and principle for SDE: The capability to add more orchestration and automation flexibility through openness to work with applications and infrastructure.

To broaden the library of content, IBM will continue to embrace the broader ecosystem to enable developers to reuse and contribute orchestration and automation content. Open source technologies are rapidly maturing and gaining adoption, driven by IBM community leadership. For example, looking at the number of Google hits for searches using the following terms, we see a tremendous growing interest in open cloud technologies:

Google search hits[2]:

- OpenStack: 2.6M
- OpenStack Heat: 267K
- OpenStack HOT: 637K
- OpenStack Heat Orchestration: 59K
- Ubuntu JuJu: 4.2M

[2] As of 27 March 2014

- Ubuntu Charms: 3.9M
- Ubuntu Juju Orchestration: 40.3K

The IBM pattern engine strategy provides the spectrum of capabilities for application deployment on clouds, including capabilities not yet delivered by OpenStack. IBM intends to fill these gaps in open source software patterns so that clients can orchestrate and automate their entire landscapes as they see fit in their data centers. As a fundamental part of this strategy, IBM will continue to embrace technologies from the open ecosystem as they mature and gain market acceptance.

Figure 1-5 illustrates the IBM strategy for open source for the pattern engines and tools:

Embrace Use open source technology in IBM products.

Extend Add value by using IBM extensions. As the capabilities increase to add extensions to the marketplace, the value will increase.

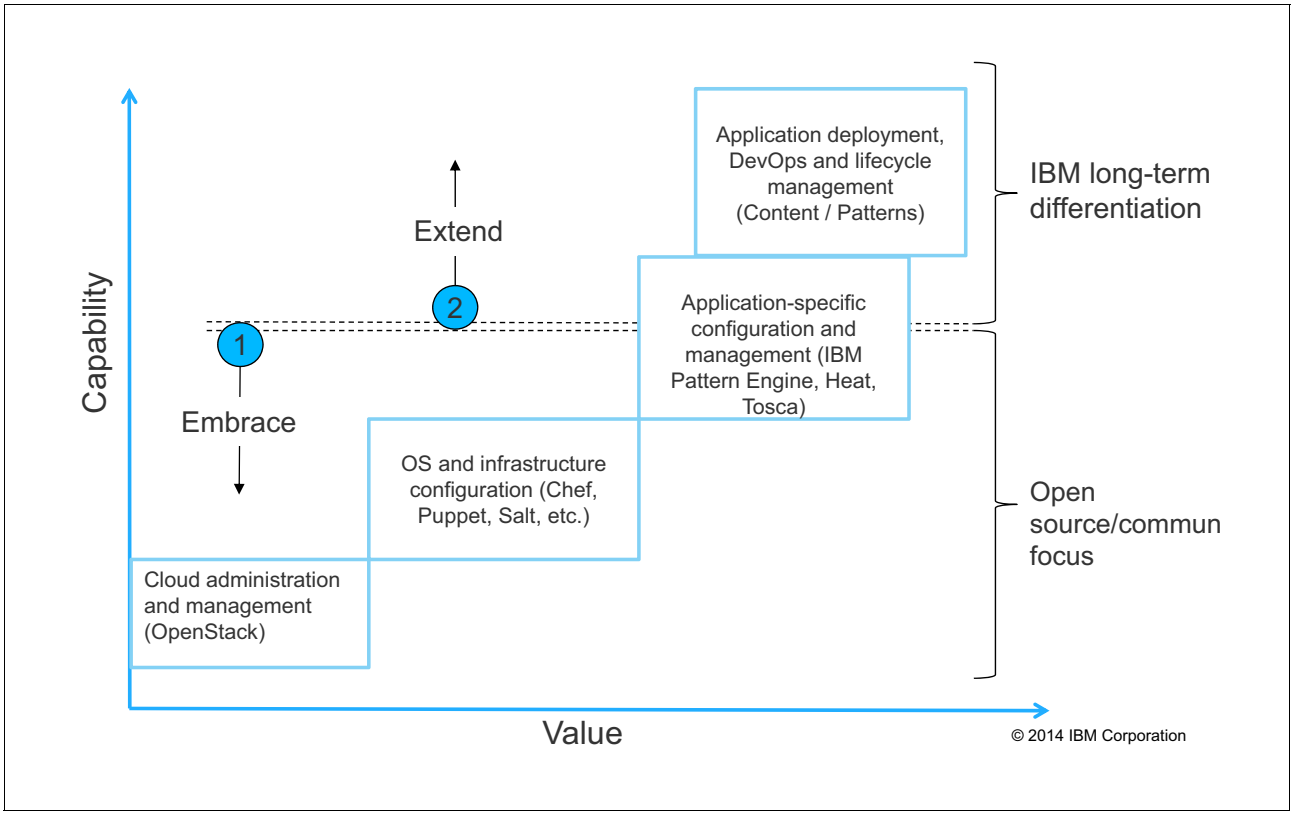


Figure 1-5 Pattern engine strategy

In a Software Defined Environment, an open solution is needed to integrate with the existing technology assets in a data center environment. This is especially relevant in today's trends in cloud Computing architectures, which are moving in the direction of Hybrid deployment models where an on-premises, private cloud based on an OpenStack based IaaS (to integrate the infrastructure architecture and technology) may be used to connect to a software as a service (SaaS) off-premises public cloud. The PaaS layer in a cloud architecture and platform provides the usable services and helps to integrate the systems of record hosted on IaaS in the private data center to the systems of engagement (mobile, tablets, and so on). An open pattern engine strategy and solution is important and critical here for enablement for all of the integrated layers for cloud computing.

Figure 1-6 illustrates the open source ecosystem and how IBM will offer differentiation through integration from IaaS to PaaS to enable faster and flexible application deployment, integrated DevOps (Development-Operations), and improved lifecycle management, all through clients being able to use their existing assets to create their own solutions.

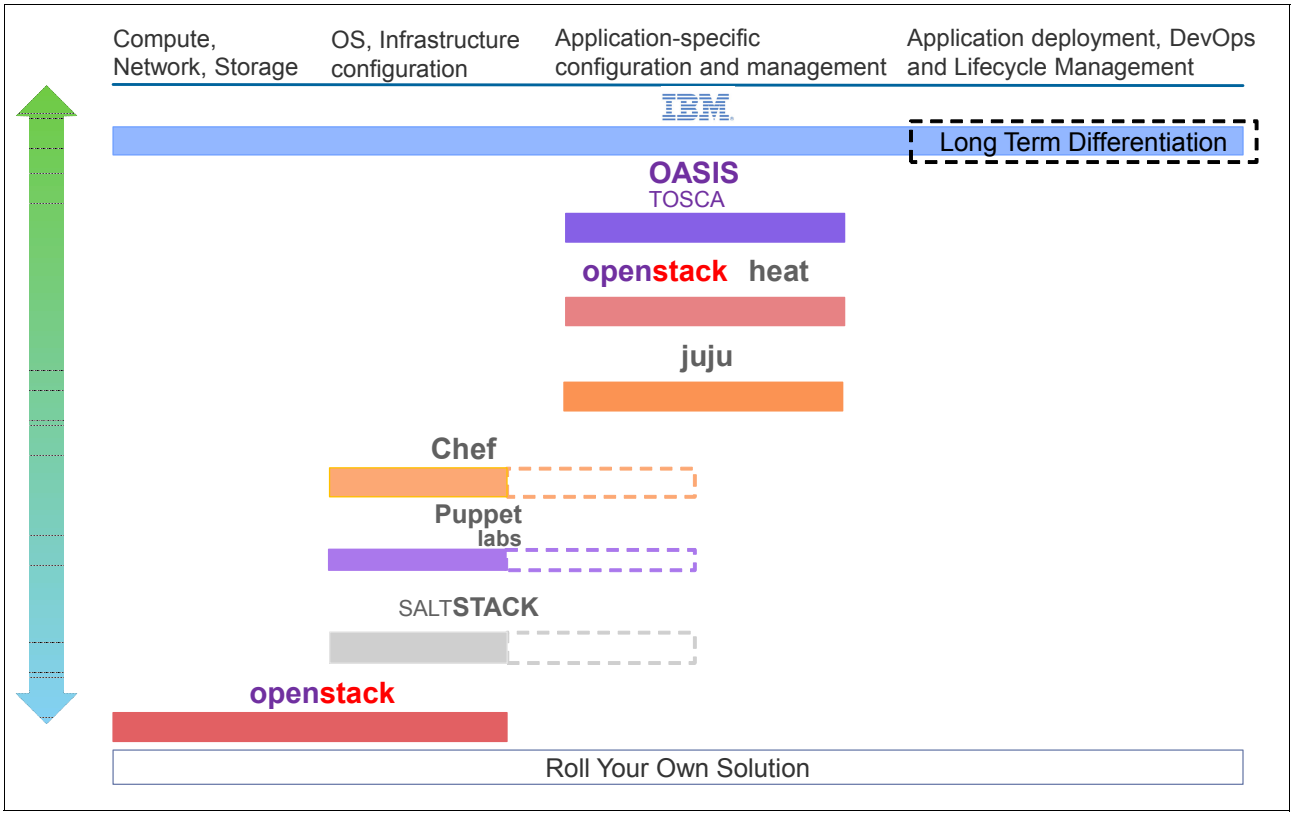


Figure 1-6 Open Source ecosystem and IBM differentiation

Looking at the Open Source ecosystem diagram, IBM is using OpenStack to build the foundation and will continue to use the following additional open standards and open source communities:

- IBM has adopted Heat and the HOT format for templates as the orchestration layer, because the Open Source community has rallied around this as the standard. (Heat and HOT were approved for integration in the OpenStack Havana release in 2013.)
- IBM, Rackspace, and others are gaining prominence as contributors and project leads within the Heat project.
- Heat and HOT work with IBM clients' existing investments in technologies, such as Chef and Puppet.
- As the diversity of the workload definition format continues to proliferate (Juju, for example), the OpenStack community has realized the need for TOSCA as an enabler for workload interoperability (HP, Fujitsu, and so on).
- TOSCA and the OpenStack TOSCA YAML profile will serve as a mechanism for growing the OpenStack Heat and HOT ecosystem beyond the existing OpenStack community.
- TOSCA and HOT are on a path for semantic convergence.
- The Landscaper tool includes the IBM HOT document or template, which also includes the added IBM extensions added through an editor. This provides the following advantages:
  - Design of patterns via simple graphics or a text editor

- User interaction of lifecycle management for patterns
- Composition from parts
- Separation of roles

- The IBM OpenStack Heat plus the IBM Extensions pattern engine further helps to show APIs to instantiate patterns and manage instances, retrieve and manipulate the state of instances, and enable for the request of delta updates more dynamically and quickly. This will provide faster enablement, coordination, and provisioning of policy-based management for cloud services, based on patterns.
- Infrastructure scheduling and optimization is provided by Platform Computing, Platform Resource Scheduler (PRS), Platform Cluster Manager Advanced Edition (PCMAE) and Spectrum Scale, which optimizes both heterogeneous IBM and non-IBM infrastructure for the right placement of workloads.

IBM products will focus on OpenStack Heat as the unified *IBM pattern engine* and on TOSCA and the HOT as the mechanism for interoperability across workload definitions. The use of Heat and HOT templates in the YAML format is demonstrated in Chapter 6, “Advanced cloud provider scenarios” on page 527.

Figure 1-7 shows the future state for the IBM pattern engine from a content, tool, engine, and platform perspective.

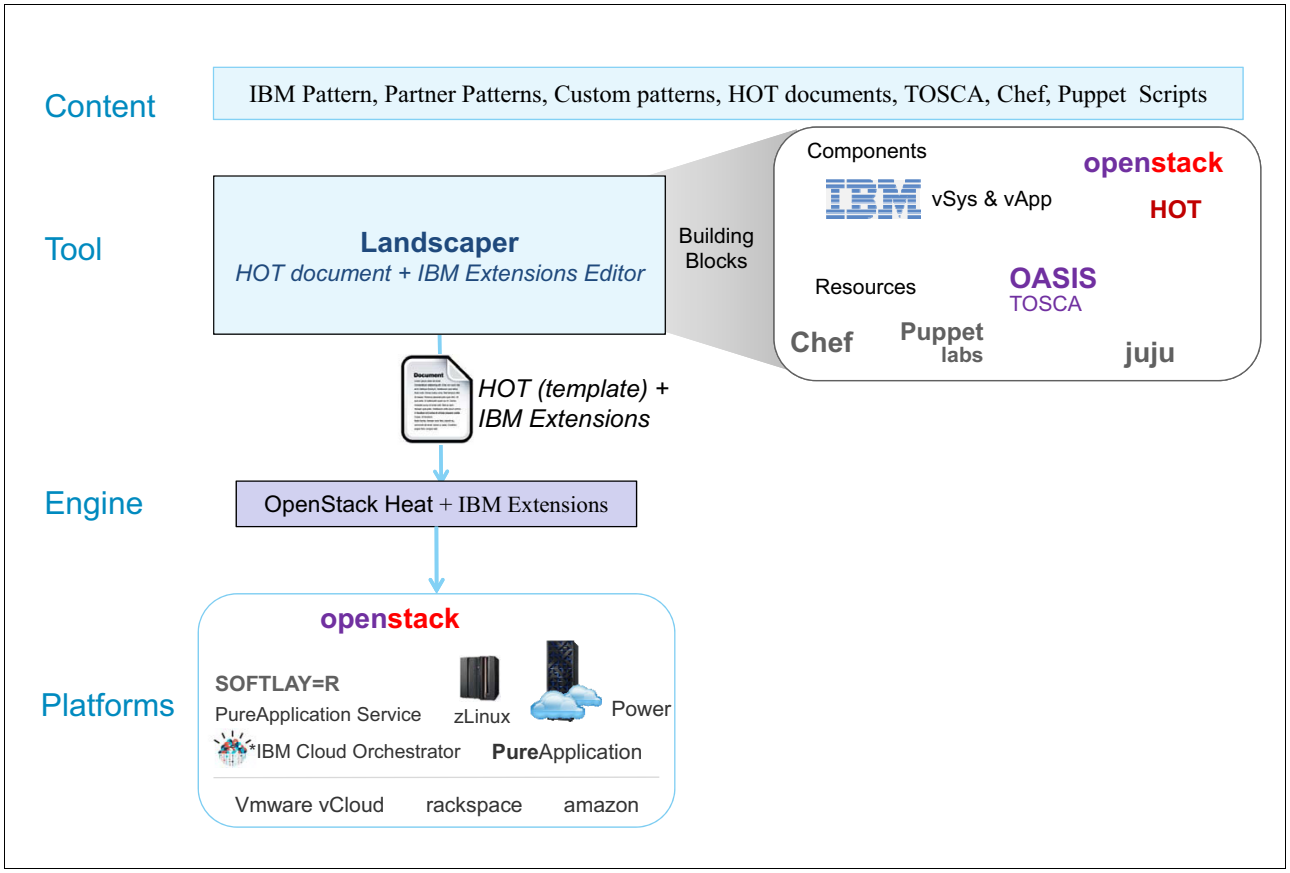


Figure 1-7 IBM pattern engine and tools, future state

The Software Defined Environment provides the next generation of cloud computing architecture through the enhanced automation and orchestration provided by the pattern engine and tools, as explained in this section. Figure 1-8 on page 18 provides a *blueprint* reference architecture that shows how it all comes together. It is the functional basis for our

solution in this book. Actual SDE hardware reference architectures aligned with specific workloads is explained in Chapter 2, “Reference architecture framework and methods” on page 47.

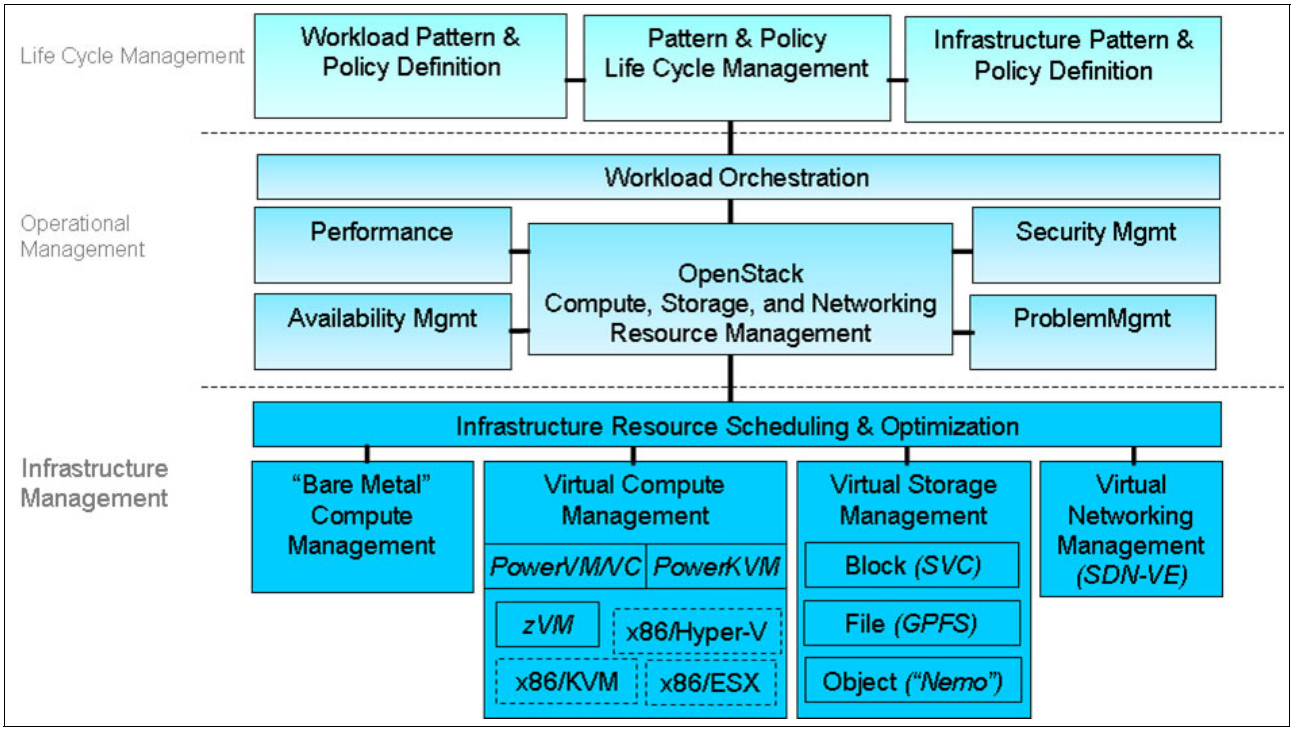


Figure 1-8 SDE next generation automation for cloud computing

1.6.3 SDE service orchestration

There is more than one IBM solution available to manage clouds and deploy orchestrated services for the client’s cloud environment. There are three SDE-based tools that will work with Heat or HOT and provide deployment. The appropriate selection of the tools depends on the business, IT, and application development organizations and where the responsibilities are defined between each of the organizations. The benefit is that IBM provides flexibility for cloud service orchestration, automation, and provisioning to tailor the fit all organizations, based on OpenStack open cloud architecture and technologies. Each of these options is described in this section.

IBM Cloud Manager

As described in the Cloud Management section, 1.6.1, “Cloud workload orchestration and management” on page 10, one of the products is IBM Cloud Manager with OpenStack (previously called SmartCloud Entry). Cloud Manager is based on OpenStack, easy to use and to deploy cloud management software, with additional IBM enhancements and enterprise class support. It supports many different hypervisors, storage, and network solutions.

Cloud Manager offers these key benefits:

- Enables rapid IT response to ever-changing demands of business via self-service provisioning of infrastructure services, because users can redeploy virtual servers with an easy-to-use interface.
- Yields improved virtualization operational efficiency and greater business effectiveness. Administrators capture and manage standard virtual machine (VM) images with support for common business processes.

- Provides the capability to track and correlate the cost of infrastructure to department use through basic usage metering. Organizations and managed service providers or cloud service providers (MSPs or CSPs) can then align services with expenses.
- Supports production-grade cloud operations and interoperability at scale through an enhanced foundation and full OpenStack API compatibility.
- Offers an open computing cloud alternative to proprietary vendors with world-class support from IBM.
- Provides hybrid capability to IBM SoftLayer® through an IBM Services offering.

See these web pages for more information:

- The IBM Cloud Manager with OpenStack introduction on IBM developerWorks

  http://www.ibm.com/developerworks/servicemanagement/cvm/sce/index.html
- The IBM Cloud Manager with OpenStack welcome page in the IBM Knowledge Center

  http://www.ibm.com/support/knowledgecenter/SST55W/welcome

IBM Cloud Orchestrator

An alternative solution to provide advanced SDE orchestration is driven by IBM Cloud Orchestrator (formerly called IBM SmartCloud® Orchestrator) to provide unification of business process, application workload, and infrastructure services.

Note: IBM Cloud Orchestrator is not installed nor part of the IBM SDE solution environment described in this book. IBM Cloud Manager is installed and implemented in the lab environment for this SDE solution, along with UrbanCode Deploy with Patterns.

Information for Cloud Orchestrator is provided here for comparison and explanatory purposes to illustrate additional advanced service orchestration capabilities as another deployment solution architecture option in a SDE environment.

Cloud Orchestrator is a comprehensive service delivery platform for cloud environments that can help lower costs and increase speed when delivering business services. As part of the IBM Cloud Foundation product suite, Cloud Orchestrator provides advanced cloud management capabilities for enterprises that need to integrate their business policies and systems in a customized cloud service through workflow coordination. Cloud Orchestrator gives you access to ready-to-use patterns and content packs, and that makes configuration, provisioning, and deployment quicker. It integrates metering, usage, accounting, monitoring, and capacity management of your cloud services.

Cloud Orchestrator supports a SDE infrastructure and workload orchestration with a graphical interface to compose and customize services. SDE workload orchestration supports pattern-based workload placement, optimization, and operations.

Cloud Orchestrator helps with the following tasks:

- Quickly deploy on-premises and off-premises cloud services.
- Provision and scale cloud resources.
- Automate manual IT processes to reduce IT costs and labor.
- Integrate with existing environments using application programming interfaces and tools (extensions).
- Deliver services with IBM SoftLayer, existing OpenStack platforms, PowerVM, IBM System z®, VMware or Amazon EC2.

For more information, see the IBM Cloud Orchestrator overview page:

http://www.ibm.com/software/products/en/ibm-cloud-orchestrator

Virtual pattern types in Cloud Orchestrator

Cloud Orchestrator supports three types of virtual patterns: virtual systems, virtual applications, and shared services.

- A *virtual system* instance is a collection of virtual machines. Each virtual machine in a virtual system instance represents a node in an application server environment.
- A *virtual application* is defined by a virtual application pattern. It is a complete set of platform resources that fulfill a business need, including web applications, databases, user registries, messaging services, and transaction processes.
- *Shared services* provide a predefined virtual application pattern that is deployed and shared by multiple application deployments in the cloud, including virtual applications, virtual systems, and virtual machine images.

The IBM SDE solution uses composable virtual system patterns (vSys) to help clients enable line-of-business (LOB) application providers and development operations (DevOps) to achieve rapid time to value in deploying applications and services in their cloud infrastructure. Virtual systems are deployed from virtual system patterns. Those patterns consist of parts that include virtual machines and scripts running on those virtual machines.

Cloud Orchestrator provides a vSys Toolkit with capabilities to deploy virtual system patterns through Cloud Orchestrator. The toolkit provides building blocks that can be used in other toolkits or process applications to build custom services and processes.

Cloud Orchestrator supports importing, deploying, and exporting service templates according to TOSCA. This enables the use of third-party content provided in a standardized format.

The IBM Software Defined Environment solution supports deploying a virtual system pattern easily in OpenStack with support for OpenStack *flavors* (virtual hardware templates) and OpenStack Heat Orchestration Templates (HOT). OpenStack versions include predefined Small, Medium, Large standard VM sizes (cores, memory, disk), and allow for user-defined combinations of cores, memory, and disk.

On-premises and off-premises orchestration with Cloud Orchestrator

Cloud Orchestrator provides connectivity to off-premises cloud resources by using an IaaS gateway. The gateway provides a routing mechanism that allows geographical deployments for multiple OpenStack instances that can be deployed in multiple sites and federated. The IaaS gateway provides cloud adapters that can be used to connect to off-premises resources, including EC2 today, with release plans to support IBM SoftLayer.

Figure 1-9 provides a SDE hybrid cloud (private, on-premises and public, or off-premises and pubic) reference solution architecture that uses Cloud Orchestrator with SoftLayer.

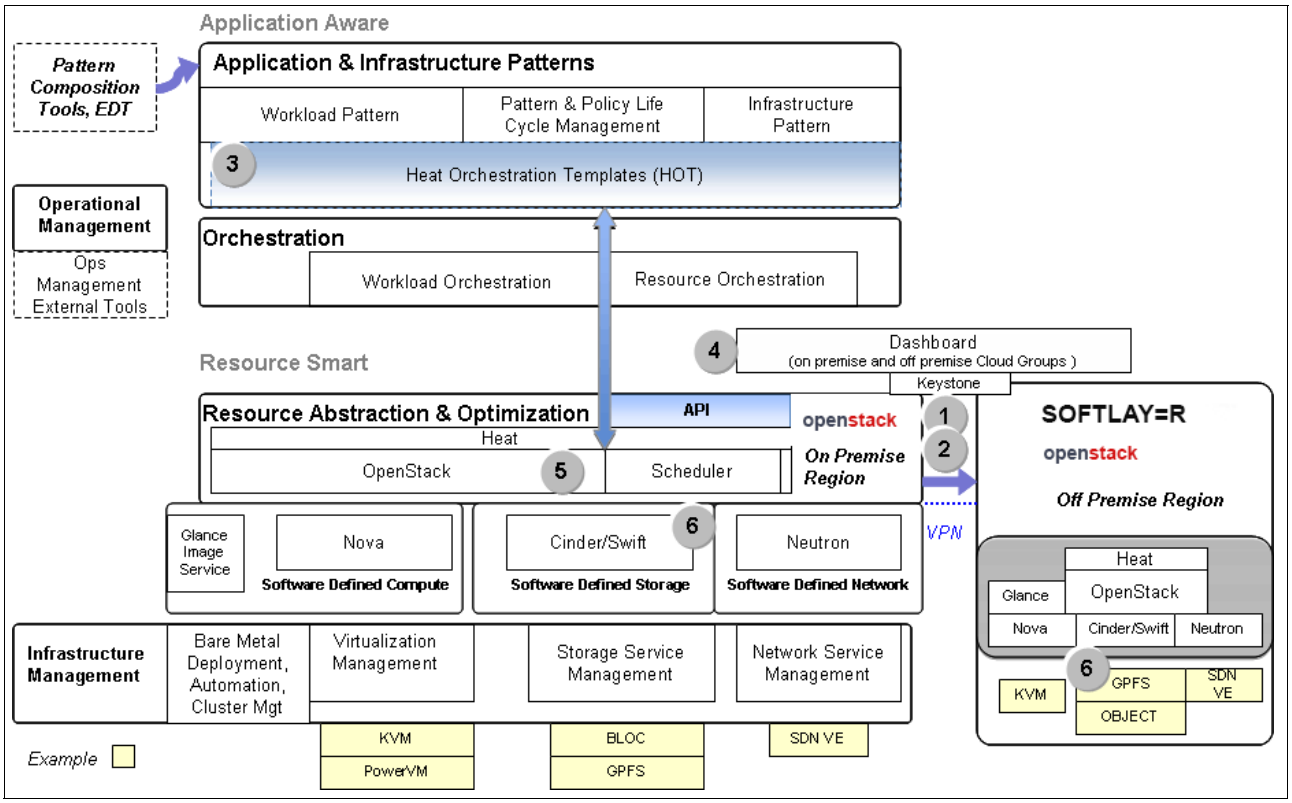


Figure 1-9 SDE hybrid cloud, OpenStack multiregion reference architecture

Key to numbers in Figure 1-9:

1. On-premises and off-premises OpenStack environments are deployed and configured to support a hybrid cloud based on client requirements.
2. The client is a multinational company that is operating in multiple data centers, so multiregion zones for on-premises and off-premises is configured to support the business and IT requirements.
3. HOT templates define workload and infrastructure patterns that are aligned with location requirements.
4. Deployment occurs via OpenStack API and Jumpgate, the dashboard, or Cloud Manager.
5. HEAT policy engine uses a HOT template, which enables policy parameters to be passed.
6. SoftLayer Resource services are based on policies (in this example, Storage Type).

Client value

Clients benefit from the following advantages:

- Cloud Orchestrator automates end-to-end: provisioning, configuration, and lifecycle management across on-premises and off-premises resources, with self-service seamless to the cloud tenant.
- Provides flexibility to choose the best optimized resource pool to deploy on-premises or off-premise.
- A single platform to show off-premises cloud use under the same visibility, control, and automation platform of the on-premises infrastructure.

1.6.4 IBM UrbanCode Deploy

IBM UrbanCode™ Deploy automates the deployment of applications, data, and configurations into development, test, and production environments. With an easy-to-use and intuitive interface, application automation specialists can model the topology of a multitier software solution and define the processes and configuration properties required to deploy and configure each tier. The sample solution described in this book uses UrbanCode Deploy to organize the software artifacts into logically grouped components and to map these components into the underlying resources managed by IBM open stack.

Note: IBM UrbanCode Deploy is not installed and part of the IBM SDE solution described in this book. UrbanCode Deploy with Patterns is described in the next section, along with the positioning in SDE for comparison purposes. It is installed and implemented in this solution.

Information for UrbanCode Deploy is provided here for comparison and explanatory purposes to illustrate capabilities in a SDE environment as another option.

In UrbanCode Deploy, components are the smallest functional unit of a software solution. Components are versioned, and each version can contain artifacts, such as a web application `.war` file or static files. In other words, components are the smallest unit which your application is composed. The components can originate from various sources, such as a Subversion repository, Git, file system, or (commonly) a build system. UrbanCode Deploy keeps track of versions of a given component. The versions of the component reference the artifacts that will be deployed as unit.

Components are composed into applications. An application is a model that defines the components, processes, target environments, and configuration values necessary for deployment.

Figure 1-10 on page 23 illustrates the relationship between UrbanCode Deploy, open cloud environments, a banking application, and components.

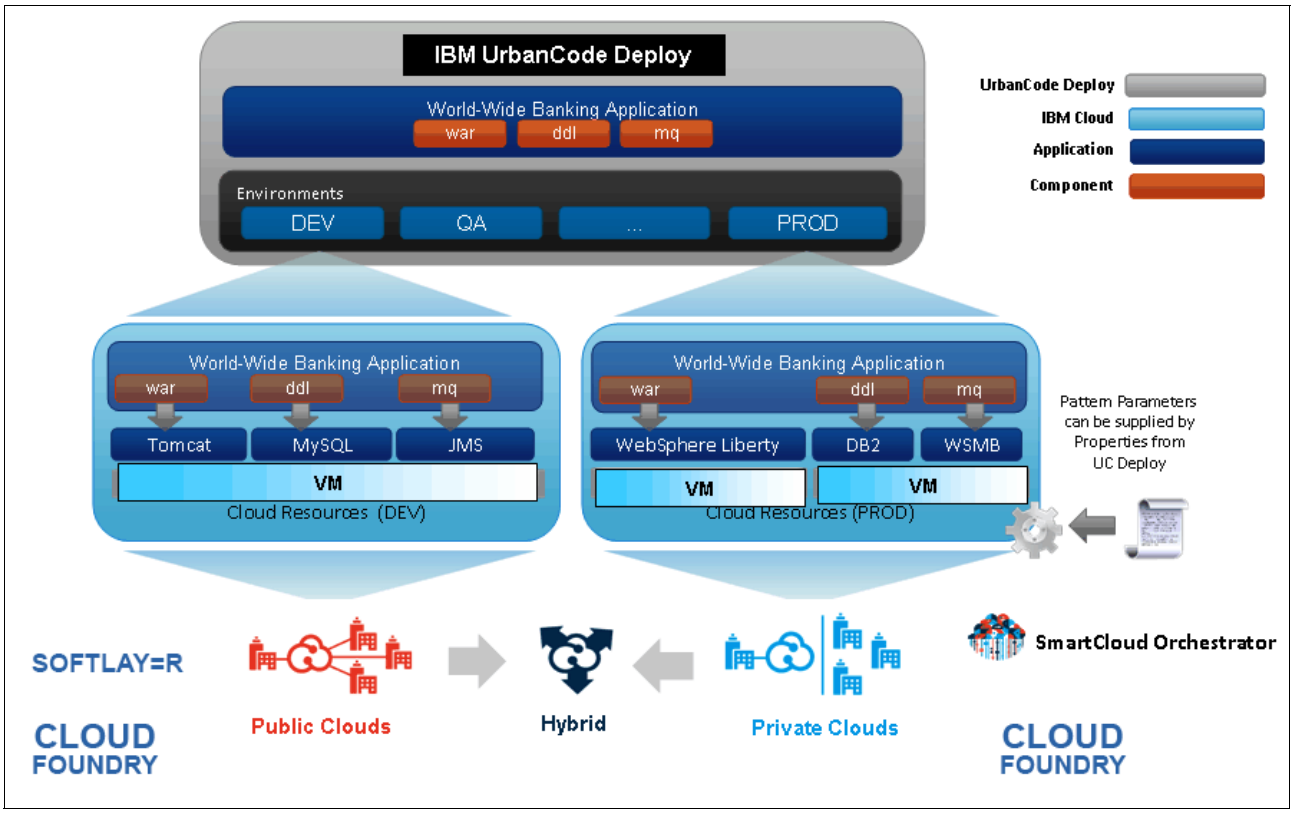


Figure 1-10 UrbanCode deploy into the cloud

The application-to-components relationship for a sample IBM BigInsights™ application is shown in Figure 1-11.

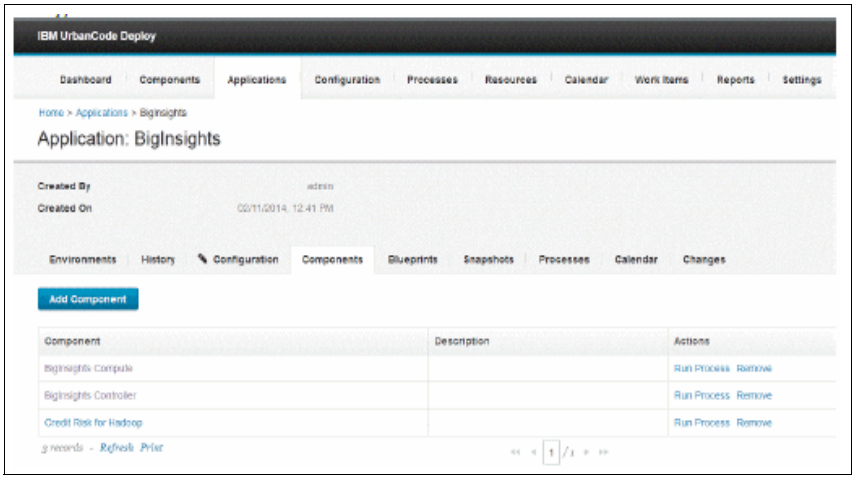


Figure 1-11 Sample application and components: BigInsights

Figure 1-11 represents a different node type in the multitier topology.

In UrbanCode Deploy, an *environment* captures the mapping of components to the target resources where they are to be deployed, as well as the configuration property values for the components. A *resource* represents a deployment target, such as a database, an Apache server, or operating system instance. An *agent* is used to enable a deployment to a resource. The agent is installed on a target system and communicates to the UrbanCode Deploy server via a messaging system, which makes it highly scalable and reliable. The agent is sent requests for work on behalf of a component within the context of an environment. The agent facilitates intelligent downloading of component artifacts and the execution of the tasks defined in the component process by using the configuration properties that are resolved by the UrbanCode Deploy server for the given environment.

After the resource is defined in the environment and it has an agent in the online state, you can begin making deployment requests. These concepts are illustrated in Figure 1-12.

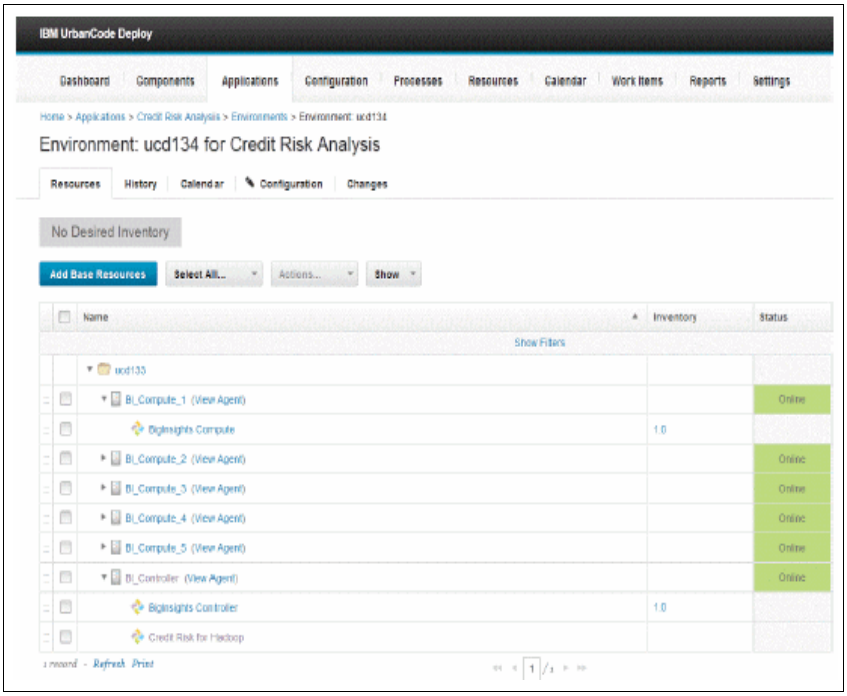


Figure 1-12 UrbanCode Deploy environment resources and components

In Figure 1-12, the `ucd134` environment points to six resources of two types: BI_Controller and BI_Compute. Each of the resources is associated with one or more components. At deployment time, an agent on each of the resources installs the corresponding components.

With that, you have the entire infrastructure set up for deploying to environments. There are two steps necessary to accomplish deployment in UrbanCode Deploy. They both are called *processes*:

- Component process
- Application process

Each component that is a member of an application has one or more processes that are used for deploying and configuring the contents of the component. A process implements a workflow. A process is visually designed and implemented in the Process Editor. In the editor, you can drag various steps to facilitate the deployment. The Editor pallet contains a rich collection of predefined steps that enable you to work with repositories (for example, to download artifacts), scripting (execute shell commands), file utilities (create a compressed file), and many others. Steps that are not directly available in the palette can be inserted into the workflow by using plug-ins. Plug-ins for many popular software packages, such as Tomcat, are available at no charge from the UrbanCode plug-in site:

https://developer.ibm.com/urbancode/plugins/

An imported plug-in shows as a set of new steps in the pallet.

Figure 1-13 shows the process definition for the BI_Controller component.

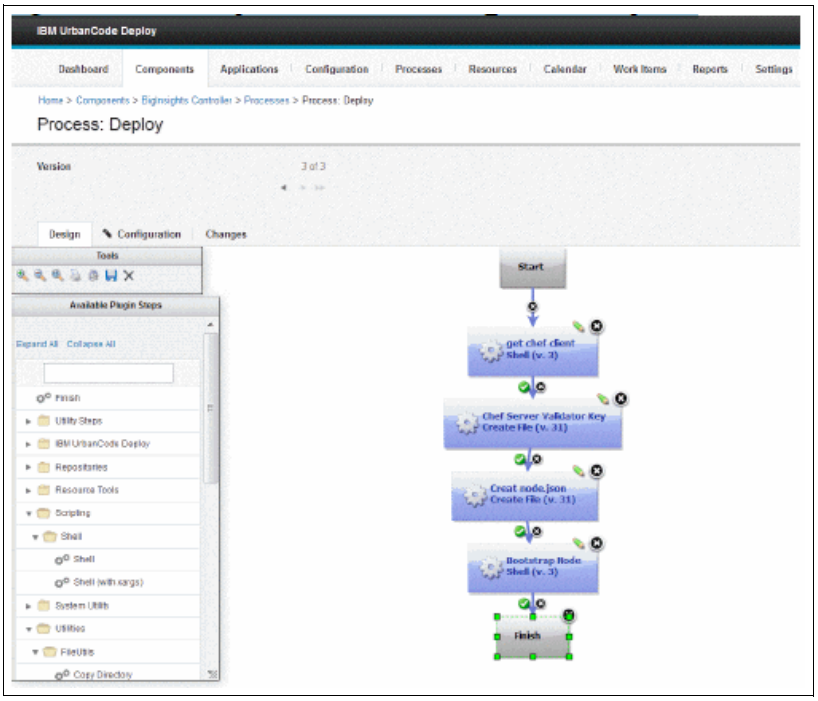


Figure 1-13 Component process definition for the BI component

The application process, in turn, manages the component processes. In the application Process editor, all of the steps refer to component processes. In the application process, you specify which component and which process for that component to use (for example, the Install process). A component may have multiple processes defined for it. To deploy the application, you select the application process, the component versions, and the target environment. At that time, the application process “knows” which component processes to run. UrbanCode Deploy has an inventory system that keeps track of which component version and configurations have been deployed on a resource.

When a deploy request is made, UrbanCode Deploy will check the inventory system to determine if the component process needs to be run or can be skipped, thus, optimizing the deployment process for faster deployments. There are many more optimizations within UrbanCode Deploy to handle the scale of enterprise applications such as tags to manage blue/green deployments.

UrbanCode Deploy with Patterns

IBM UrbanCode Deploy with Patterns is a full-stack environment management and deployment solution that enables users to design, deploy and update full-stack environments for multiple clouds. Extending the benefits of UrbanCode Deploy, this solution optimizes your continuous delivery throughput by automating infrastructure in the cloud, using open standards defined by OpenStack. With UrbanCode Deploy with Patterns, organizations can holistically manage and control their application workloads, and users can design, deploy, and reuse environments quickly with improved collaboration, lifecycle management, and cloud portability.

UrbanCode Deploy with Patterns provides the following capabilities:

- Design full-stack application environments using a pattern designer to develop and update open, full-stack application environments in a diagram or text editor.
- Deploy full-stack application environments for multiple clouds using a multi-cloud and hybrid strategy across your delivery pipeline.
- Manage the environment lifecycle by updating existing environments and executing infrastructure changes based on changes to the pattern artifacts.
- Support your end-to-end delivery pipeline by integrating with UrbanCode Deploy to support full-stack environment provisioning.

Note: The key difference with UrbanCode Deploy with Patterns is the capability to orchestrate and set up more of the application development aspects into the Heat and HOT orchestration templates and tie the automation and deployment to the infrastructure deployments and VMs to support continuous engineering and DevOps.

Figure 1-15 illustrates how UrbanCode Deploy automates application deployment to hybrid cloud resources.

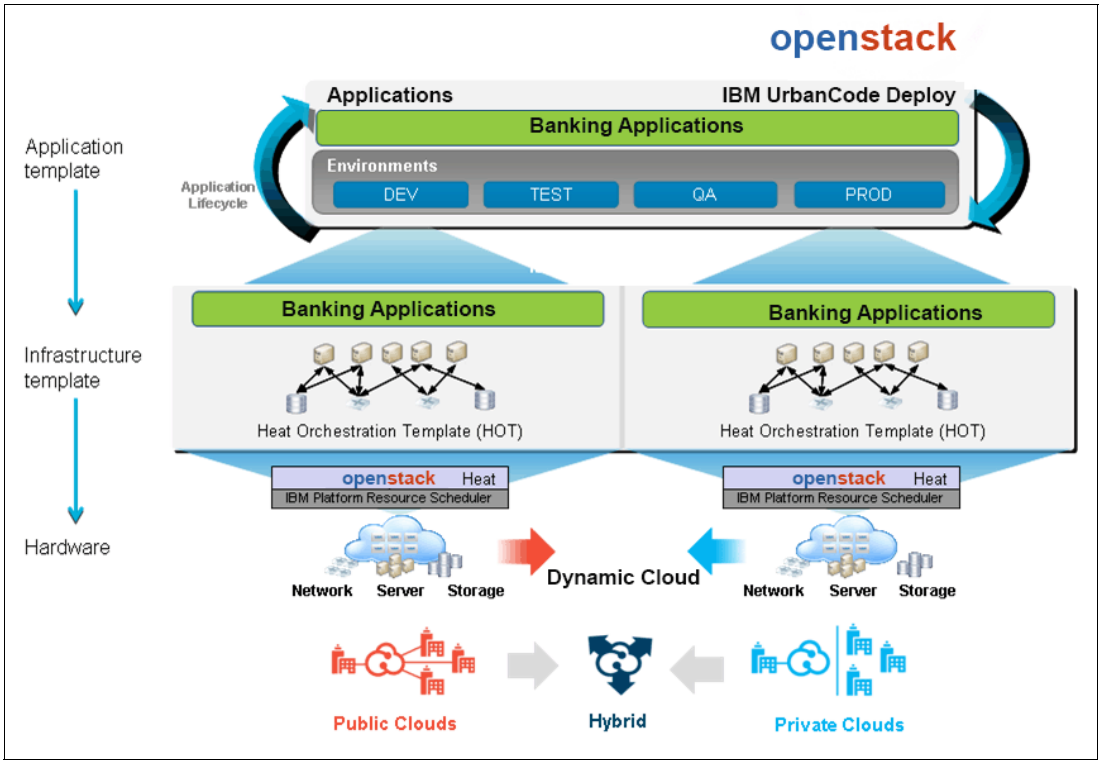


Figure 1-14 UrbanCode Deploy with Patterns

The key point to remember is that UrbanCode Deploy focuses on the application and middleware layer to accelerate the integration between development and operations. This can be seen in Figure 1-15.

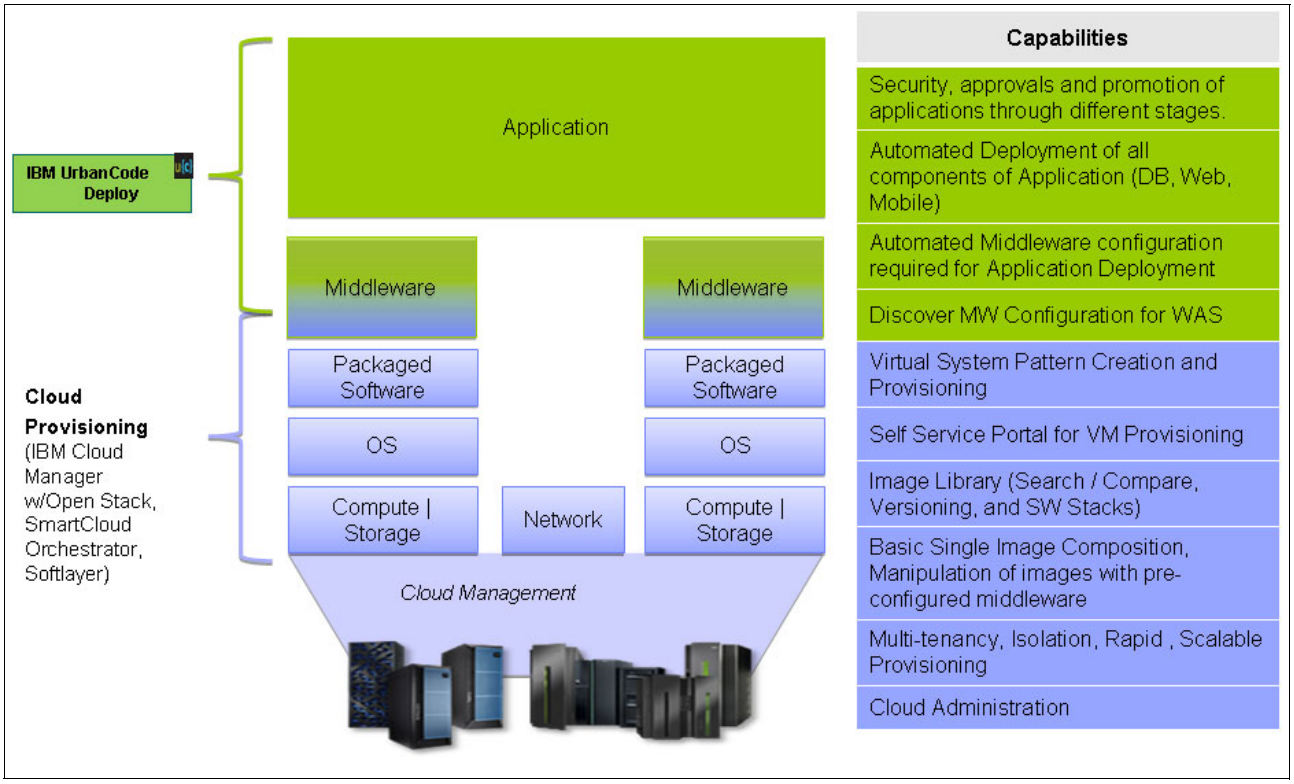


Figure 1-15 Application deployment via UrbanCode Deploy versus cloud provisioning

For more information about UrbanCode Deploy with Patterns see the following web pages:

- IBM UrbanCode Deploy with Patterns

  `http://www.ibm.com/software/products/en/ucdep-patterns`
- IBM UrbanCode Deploy with Patterns section in the IBM Knowledge Center

  `http://www.ibm.com/support/knowledgecenter/SSWS3W/welcome`
- IBM UrbanCode Deploy with Patterns in the Developer Center on IBM developerWorks

  `https://developer.ibm.com/urbancode/products/urbancode-deploy-with-patterns/`

Taking these capabilities together, it's important to see and map these, in context, to individual roles and responsibilities (or personas) between the application teams and infrastructure. Figure 1-16 on page 28 provides a representation of how that looks.

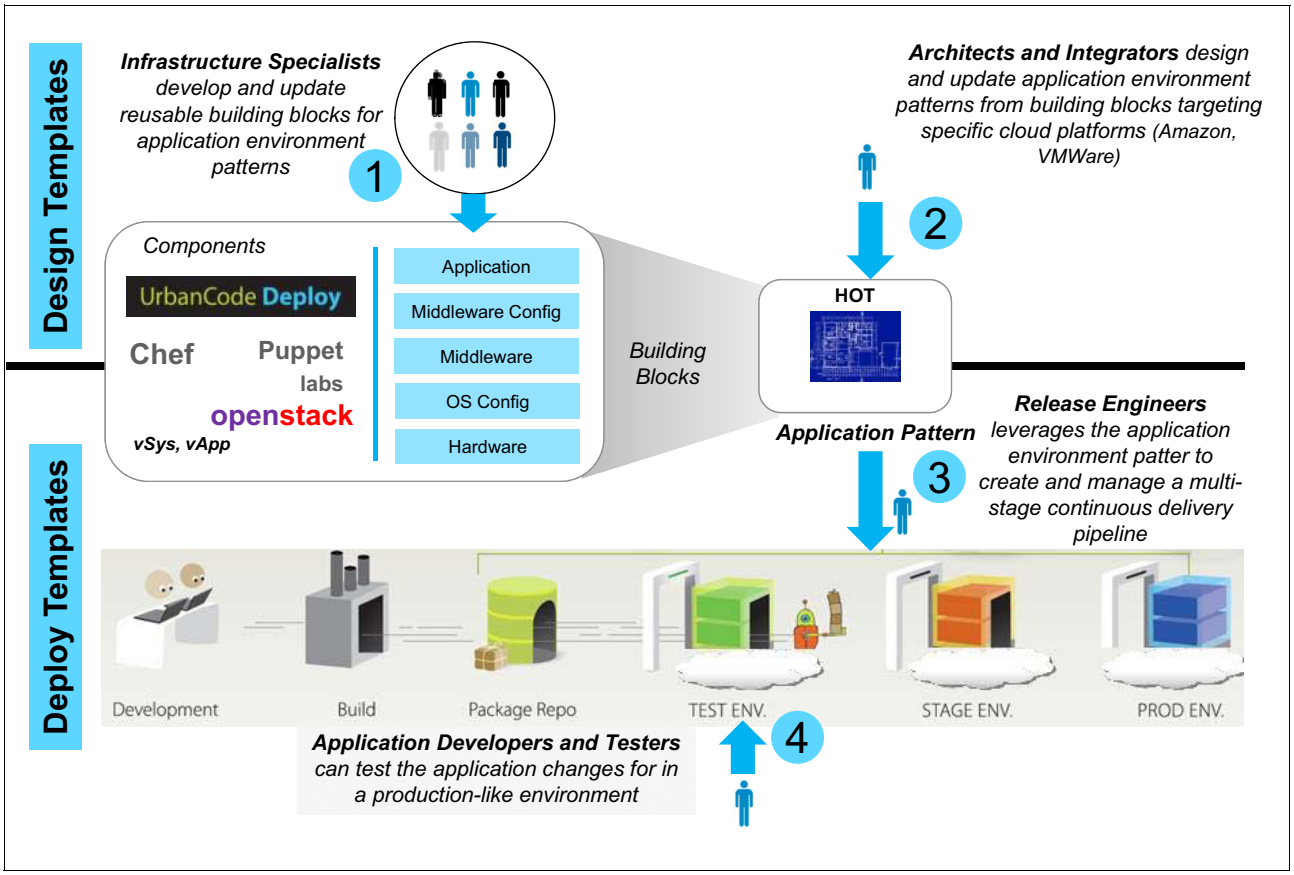


Figure 1-16 SDE and DevOps User Personas

Key to numbers referenced in Figure 1-16:

1. *Infrastructure specialists* develop and update reusable building blocks for application environment patterns, to work with OpenStack.
2. *Architects and integrators* design and update application environment patterns from building blocks targeting specific cloud platforms (Amazon, VMware, and so on).
3. *Release engineers* use the application environment pattern to create and manage a multi-stage continuous delivery pipeline.
4. *Application developers and testers* can test the application changes in a production-like environment before releasing to production.

In this solution, we illustrate SDE Service Orchestration through IBM Cloud Manager and also through UrbanCode Deploy with Patterns in Chapter 6, “Advanced cloud provider scenarios” on page 527.

Next, the individual supporting technologies and components for the specific SDE solution in this book in a real working lab environment are described. Figure 1-17 on page 29, provides a high-level illustration of these technologies.

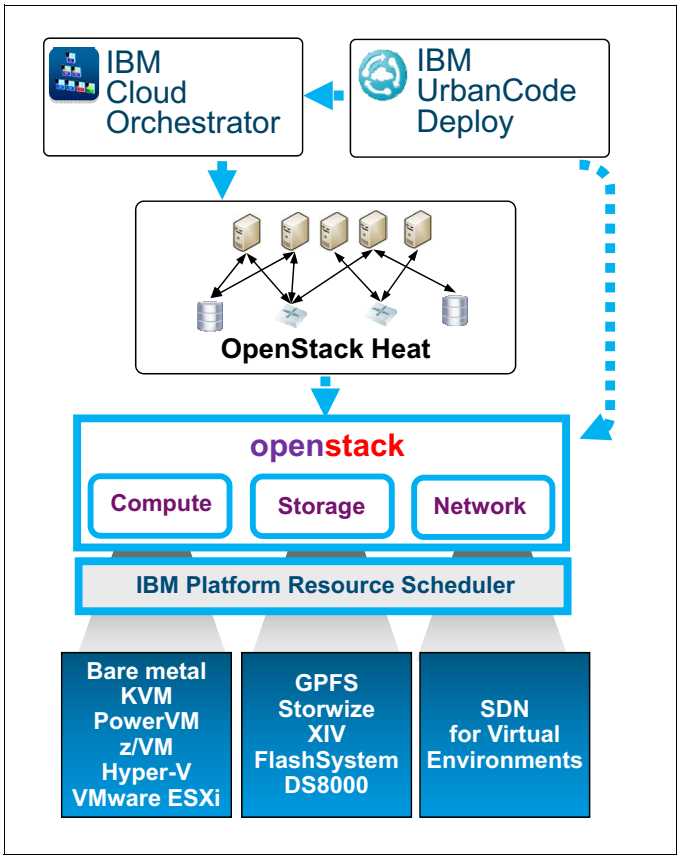


Figure 1-17 IBM SDE solution

IBM PowerVC

IBM PowerVC is very easy to install and use cloud management for IBM POWER servers. PowerVC is based on OpenStack running on the Linux operating system. It can manage POWER servers running PowerVM or PowerKVM hypervisor. It uses well-known technologies such as VIOS servers or HMC console. PowerVC integrates with IBM Cloud Manager so in case you have x86 and POWER servers you can use IBM Cloud Manager to manage both server architectures from one dashboard. For more information about PowerVC see these web pages:

- IBM PowerVC – Virtualization Center

  `http://www.ibm.com/systems/power/software/virtualization-management/`

- PowerVC Standard Edition in the IBM Knowledge Center

  `http://www.ibm.com/support/knowledgecenter/SSXK2N/welcome`

- *IBM PowerVC Version 1.2.1 Introduction and Configuration*, SG24-8199

  `http://www.redbooks.ibm.com/redpieces/abstracts/sg248199.html`

IBM Platform Resource Scheduler

IBM Platform Resource Scheduler (PRS) is a resource scheduling technology originating from the Platform Computing offering for high-performance computing (HPC). Now PRS is available for OpenStack cloud management and integrates with Cloud Manager, where you can find it under the new Resource Scheduler tab (see Figure 1-18).

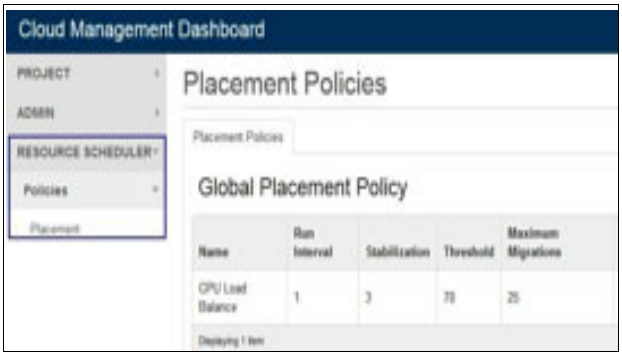


Figure 1-18 Platform Resource Scheduler

The IBM SDE solution showcases integration of policy-based resource scheduling via self-service orchestration. IBM Platform Resource Scheduler (PRS) is integrated with the IBM OpenStack Nova Scheduler to provide advanced scheduling features, and enables dynamic infrastructure resources allocation based on workload and resource availability and uses resource optimization placement policies rather than the default OpenStack compute scheduler filters. PRS provides flexibility by allowing polices to be set globally and prescriptively per workload instance.

The placement policies allow you to make decisions on where to deploy the virtual machines (VMs) with specified resource selection policies. The optimization policy is adjusted dynamically. This means it is monitored both at VM deploy (VM creation) and during the VM lifecycle. For example, when a new VM is deployed, the strategy monitoring system looks at where to place this VM based on the optimization policy selected. If a VM is deleted, the monitoring system will adjust accordingly, based on the optimization policy. This could trigger a VM migration that rearranges the distribution of VMs across the VM hypervisors.

PRS provides client value in several ways:

- Intelligent and policy-driven virtual server placement
- Policy support for automated virtual server deployment, relocation, and restart
- Server use and energy use
- Increased virtual server availability and resilience
- Multitenant resource sharing

PRS supports various placement policies:

- *The CPU Load Balance* policy allows your OpenStack cloud to evenly allocate the CPU load across the host systems.
- *The Memory Load Balance* policy enables optimized distribution of memory load across the host systems.
- *The Striping Policy* evenly distributes the number of virtual machines across the host systems.
- *The Packing Policy* packs VM instances as densely as possible to maximize the number and density of running VM instances.

- *The CPU Utilization Policy* balances the CPU use among the host systems. When selecting a policy, you will have a few policy attributes to manage:

  Run interval The runtime interval in which the policy will be executed. In Figure 1-19, we have selected a value of **1**. This means that the policy will run every minute.

  Stabilization The number of consecutive times in which a policy goal can exceed the policy threshold value before the policy action is triggered.

  Threshold The starting point for live migration. Depending on the policy, this can be a percentage of either CPU use memory or memory use.

  Maximum migration The number of live migration operations allowed for the cluster.



Figure 1-19 Platform Resource Scheduler CPU Utilization Policy

In addition, you can customize the default behavior of Platform Resource Scheduler by creating your own plug-ins and then integrating them into OpenStack Nova.

SDE Service Level Policies

The IBM SDE solution enables business policy and class of service association. This feature enables cloud administrators and cloud tenants to select policies based on cost and value ratio for their business needs.

Class of Service includes policy examples for Bronze and Gold services. The Bronze Service class provides a “Packing” policy that runs a denser set of VM on low-cost disk. The Gold service provides “Striping” and runs VMs on High Performance disk. Clients can readily use the SDE baseline policies to build their custom classes of services to meet their business needs.

Policies and class of service are delivered through the self-service catalog and can be used within a workload pattern.

For more information, see IBM Platform Resource Scheduler in the IBM Knowledge Center:

http://ibm.co/1D7JAMr

IBM Platform Cluster Manager, centralized SDE automation and management

IBM Platform Cluster Manager Advanced Edition (PCMAE) enables IT professionals to create a shared, automated infrastructure that enables users to deploy multiple cluster environments with great simplicity and efficiency. The following are some of key capabilities of PCMAE used for the IBM SDE solution:

- Administrators can use the PCMAE portal to create and manage cluster definitions, which provide a template for specifying how machines within the cluster environment should be provisioned and managed. Cluster definitions also provide great flexibility and capability to define which scripts to run on which machine at what time to prepare the environment for custom activities within the cluster.
- A cluster definition in PCMAE consists of tiers that work together and each tier within a cluster definition includes hardware layers and script layers along with dependencies between tiers. PCMAE provides a visual Cluster Designer to create or modify cluster definitions. The Cluster Designer enables you to visually define the infrastructure and provisioning tasks when creating or modifying cluster definitions. The Cluster Designer displays different machines in the cluster as tiers and displays the definition of the machine itself and any pre- or post-installation scripts within the machine as layers in each tier.
- PCMAE includes a provisioning engine that selects appropriate physical machines per customizable placement policies, performs bare metal provisioning on the physical machines and manages the physical machines that make up the cluster.
- Users can simply select a cluster definition as appropriate and deploy a cluster environment in a fully automated way.
- Users can add or remove a number of machines to or from a cluster manually or automatically according to rule-based policies.
- PCMAE helps with lifecycle management of the clusters and individual physical machines.

As an option in the IBM SDE solution, PCMAE is used as a starting point to quickly provision the whole software defined environment in a fully automated way and as a centralized platform for the infrastructure automation and management.

We defined a SDI cluster definition in PCMAE to provision a software defined infrastructure cluster in a fully automated way. This significantly increases the efficiency and reduces the skills required to create a software defined environment.

The software-defined infrastructure cluster consists of these components:

- IBM OpenStack
- IBM Spectrum™ Scale (formerly IBM General Parallel File System, GPFS™)
- IBM Platform Resource Scheduler (PRS)
- IBM Software Defined Network for Virtual Environments

The SDI cluster definition in PCMAE can provision the required number of physical machines with the specified Linux hypervisor versions installed, sets up the network environment and provisions an IBM OpenStack controller node and a specified number of Compute nodes on those machines via executing specified chef cookbooks in a fully automated way.

Further, post-provision scripts are used in the SDI cluster definition to configure IBM OpenStack and GPFS to work together. Specifically, GPFS is configured as the shared file system to store the OpenStack images in Glance, and the VM instances for every Compute node and GPFS are also configured as the backend volume storage for OpenStack Cinder via the GPFS cinder driver developed by IBM. VM live migration will be enabled for every Compute node in the OpenStack cluster.

Automated provisioning of OpenStack controller services on multiple nodes (for HA and distributed mode) will be implemented in the future.

Note: PCMAE is not currently installed or implemented in the IBM SDE solution. This information is provided here for explanatory purposes and as a next step toward advanced automation and orchestration.

Platform Resource Scheduler integration with Platform Cluster Manager Advanced Edition

The same technology that enables advance scheduling of virtual OpenStack instances is also integrated into Platform Cluster Manager Advance Edition (PCMAE). Here, it enables the user to define policies for placement of new workloads onto physical systems. It also enables the user to define a policy that can react to workload demand dynamically over time. PCMAE enables the user to "flex" workload clusters based on this policy. By default, PCMAE provides frameworks for CPU, disk, and memory use-based policies. Triggers from these policies result in provisioning additional physical resources for use by the workload.

The benefit provided by the IBM SDE solution here, is that it uses this capability to dynamically allocate resource to OpenStack cloud instances that are automatically based on the policies that clients set ahead of time.

1.7 Software Defined Compute

Software Defined Compute is a term used to describe a variety of hypervisor and virtualization technologies running on physical hosts and serving resources to guest virtual machines. The IBM Software Defined Environment is a flexible solution and can be used with many popular hypervisors, such as IBM PowerVM, KVM, Hyper-V, or ESXi. This section describes the most popular hypervisors.

1.7.1 PowerVM

IBM PowerVM is a well known, robust virtualization technology available on IBM POWER servers. It built in POWER servers firmware and can provide virtualized or physical resources to virtual machines (logical partitions, or LPARs) running AIX, IBM System i® or Linux operating system. PowerVM enables dynamic, non-disruptive, manual or automatic resource relocation, such as dynamic CPU or RAM reassignment between different LPARs. For more information about PowerVM, see these sources:

- Server virtualization with IBM PowerVM

  http://www.ibm.com/systems/power/software/virtualization/

- *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940

  http://www.redbooks.ibm.com/abstracts/sg247940.html

1.7.2 PowerKVM

IBM PowerKVM is a new virtualization technology available for IBM Power Systems, alongside PowerVM. It is supported only on IBM POWER8™ servers and uses Linux on Power operating systems with kernel-based virtual machine (KVM). Unlike PowerVM, PowerKVM doesn't need neither VIOS nor HMC to implement virtualization and supports only Linux on Power guest virtual machines. For more information about PowerKVM, see:

- IBM PowerKVM – open virtualization

  http://www.ibm.com/systems/power/software/linux/powerkvm/

- *IBM PowerKVM Configuration and Use*, SG24-8231

  http://www.redbooks.ibm.com/redpieces/abstracts/sg248231.html

1.7.3 Kernel-based virtualization

A kernel-based virtual machine (KVM) is a virtualization technology available on x86 servers. KVM runs on a Linux operating system and can provide virtualization for Linux and Microsoft Windows guest virtual machines.

For more information about KVM, see the Linux KVM web page:

http://www.linux-kvm.org/page/Main_Page

1.8 IBM z/VM

IBM z/VM® is a virtualization technology available on IBM System z servers. It supports enterprise class, top notch RAS - reliability, availability and scalability. It also highly secure enterprise infrastructure capable of support more guest virtual machines, running on a single piece of hardware, than any other platform on the market. z/VM supports IBM z/OS® and Linux operating systems. For more information, see these resources:

- The IBM z/VM web page

  http://www.vm.ibm.com/

- *The Virtualization Cookbook for IBM z/VM 6.3, RHEL 6.4, and SLES 11 SP3*, SG24-8147

  http://www.redbooks.ibm.com/abstracts/sg248147.html

1.9 Hyper-V

Hyper-V is a virtualization technology provided by Microsoft. It runs on x86 servers and supports Linux and Windows guest virtual machines. It is also supported by IBM Cloud Manager as one of the underlying cloud technologies. For more information, see the Virtualization page on Microsoft's website:

http://www.microsoft.com/en-us/server-cloud/solutions/virtualization.aspx

1.10 ESX

ESX is a virtualization technology provided by VMware. It runs on x86 servers and supports Linux and Windows guest virtual machines. It is also supported by IBM Cloud Manager as one of the underlying cloud technologies. For more information about ESX, see the VMware ESXi page on the VMware website:

http://www.vmware.com/products/esxi-and-esx/overview

1.11 Software-defined storage

Software-defined storage (SDS) is a method of providing storage for physical and virtual hosts. In SDS all storage functions are embedded in software so there is no need for sophisticated, dedicated storage equipment. SDS is much more flexible than traditional hard disks or disk arrays. In Software Defined Environments, you can use and mix different storage products and solutions. For example, you can use IBM SAN Volume Controller for PowerVC and IBM Spectrum Scale (was GPFS) for x86 KVM, all managed by a single GUI or CLI provided by Cloud Manager. This section describes different storage solutions.

1.11.1 IBM SAN Volume Controller

IBM SAN Volume Controller (SVC) is a well-known and proven, robust storage solution which scales up to 32 petabytes. SVC is a storage software running on cluster of two or more x86 servers. It provides massive number of block storage functions, such as IP or FC replication, mirroring, snapshots, thin provisioning, compression, Fibre Channel, and iSCSI connection, and so on. Also, it's very easy to use through a GUI or a by using a CLI for more complicated tasks. SVC can virtualize huge number of external disk arrays to provide common storage pool to hosts. Cloud Manager and PowerVC manage SVC via plug-in for OpenStack Cinder block storage module.

There is more helpful documentation about SVC, especially in these other IBM Redbooks publications:

- IBM Spectrum Virtualize - SAN Volume Controller

  http://www.ibm.com/systems/storage/software/virtualization/svc/
- *Implementing the IBM System Storage SAN Volume Controller V7.2*, SG24-7933

  http://www.redbooks.ibm.com/abstracts/sg247933.html
- *IBM SAN Volume Controller Stretched Cluster with PowerVM and PowerHA*, SG24-8142

  http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg248142.html
- *IBM SAN Volume Controller 2145-DH8 Introduction and Implementation*, SG24-8229

  http://www.redbooks.ibm.com/redpieces/abstracts/sg248229.html
- *IBM System Storage SAN Volume Controller and Storwize V7000 Best Practices and Performance Guidelines*, SG24-7521

  http://www.redbooks.ibm.com/redpieces/abstracts/sg247521.html
- *IBM SAN Volume Controller and Storwize Family Native IP Replication*, REDP-5103

  http://www.redbooks.ibm.com/redpieces/abstracts/redp5103.html
- *Implementing FlashSystem 840 with SAN Volume Controller*, TIPS1137

  http://www.redbooks.ibm.com/abstracts/tips1137.html

- IBM SAN Volume Controller section in the IBM Knowledge Center

  http://www.ibm.com/support/knowledgecenter/STPVGU/landing/SVC_welcome.html

1.11.2 IBM Storwize V7000

IBM Storwize® V7000 is a disk storage systems providing block storage to hosts or other V7000, or SVC. Storwize runs the same code as SVC does and has the same features as SVC. What function differs it from SVC is that it actually includes up to several hundreds of real physical disks, which can be gathered in RAID arrays and then provide common storage pool for physical or virtual servers. As SVC, Storwize V7000 is managed by Cloud Manager and PowerVC via plug-in for OpenStack Cinder block module. For more information about Storwize V7000, see the follow web pages and Redbooks publications:

- IBM Storwize V7000 Unified and Storwize V7000 Disk Systems

  http://www.ibm.com/systems/storage/disk/storwize_v7000/

- *Implementing the IBM Storwize V7000 V7.4*, SG24-7938

  http://www.redbooks.ibm.com/abstracts/sg247938.html

- *IBM System Storage SAN Volume Controller and Storwize V7000 Best Practices and Performance Guidelines*, SG24-7521

  http://www.redbooks.ibm.com/redpieces/abstracts/sg247521.html

- *IBM SAN Volume Controller and Storwize Family Native IP Replication*, REDP-5103

  http://www.redbooks.ibm.com/redpieces/abstracts/redp5103.html

- IBM Storwize V7000 section in the IBM Knowledge Center

  http://www.ibm.com/support/knowledgecenter/ST3FR7/landing/V7000_welcome.html

1.11.3 IBM XIV

IBM XIV® is a monolithic storage system providing block storage to physical or virtual host systems. What differs XIV from other storage systems is ease of use and reliability. In XIV all data are distributed evenly on all available disks. All data has a copy which also is distributed among all physical disks. This is done automatically so no disk array tuning is needed. XIV comes with configuration up to several hundreds of TB. XIV is supported by Cloud Manager via plug-in for OpenStack Cinder block storage module.

For more information about XIV, see the following web pages and Redbooks publication:

- IBM XIV Storage System

  http://www.ibm.com/systems/storage/disk/xiv/

- IBM XIV Storage System Architecture and Implementation

  http://www.redbooks.ibm.com/redpieces/abstracts/sg247659.html

- IBM XIV Storage System documentation

  http://ibm.co/1DA3WYL

1.11.4 IBM Spectrum Scale (was IBM General Parallel File System, or GPFS)

Many IT organizations want to use commodity storage hardware as part of their software-defined storage infrastructures. IBM General Parallel File System (GPFS) turns commodity local or external storage into elastic, highly available Enterprise cloud storage for cloud and big data applications. GPFS is a prominent part of the IBM SDE solution.

GPFS can use many different block storage devices, including SSDs, SAS HDDs, and NL SATA HDDs, to create a single shared file system. IBM General Parallel File System is distributed file system that can run on AIX, Windows and Linux operating systems. There's a large list that GPFS supports. GPFS storage pools can be used to create cost or performance optimized storage pools and volumes from those pools can be assigned to application deployments to help users maintain service level agreements (SLAs). The GPFS policy engine enables sophisticated rules to be defined to move data among the tiers as another means to manage SLAs.

This mature IBM storage technology can scale up to petabytes and billion of files in one file system without jeopardizing its very high performance. GPFS can be installed on physical host's local drives or on the drives provided by the external storage system, for example, SVC. GPFS has impressive number of features, such as snapshots, thin provisioning, SSD/Flash support, automatic tiering, replication, concurrent read/write from multiple hosts, encryption, quota management, and so on. GPFS is managed by Cloud Manager via plug-in from OpenStack Cinder block storage module. Additionally GPFS can be managed by OpenStack Swift object storage module. In this solution, SVC and General Parallel File System (GPFS) are used as elastic storage to provide backing devices to the instances. In our cloud, we used 2 x3650 M2 servers running Red Hat Enterprise Linux Server release 6.5 and KVM/QEMU version qemu-kvm-0.12.1.2-2.415. These servers are connected to both public and private networks and to the SVC storage. A GPFS Cluster was created in these 2 nodes using SVC disks. They are named cmo1.itso.ibm.com and cmo2.itso.ibm.com and are used as Compute nodes for deploying Windows and Linux instances.

The IBM SDE solution automates the deployment of a GPFS cluster. Typically, a storage administrator had to be familiar with the fine details of GPFS before they could manually install and configure a GPFS cluster.

Note: Please see 4.11.4, "Integrate Cinder with IBM Spectrum Scale" on page 387 for further details on how the GPFS cluster was created for this solution. This publication provides brief instructions on how to install and set up a GPFS Cluster. For more information and options setting up a GPFS Cluster, consider reviewing the following documentation:

Deploying a big data solution using IBM GPFS-FPO

http://public.dhe.ibm.com/common/ssi/ecm/en/dcw03051usen/DCW03051USEN.PDF

Implementing the IBM General Parallel File System (GPFS) in a Cross Platform Environment, SG24-7844

http://www.redbooks.ibm.com/abstracts/sg247844.html

By automating this process, GPFS deployment time is significantly reduced and enables users with limited GPFS knowledge to deploy GPFS clusters with ease. The internal disks of individual cloud servers can be part of a shared GPFS file system, and can be provisioned into hot and cold storage pools to satisfy quality of service requirements for VMs. Further information about using PCMAE to fully automate the provisioning of GPFS, see the citations that follow.

The SDI cluster definition for GPFS in PCMAE provisions the required number of physical machines with a specified Linux operating system version installed, sets up the network environment, NSD/disk servers, quorum nodes, and a specified number of client nodes in a fully automated way.

For more information about GPFS and using it with PCMAE, see these sources:

- IBM Spectrum Scale

  http://www.ibm.com/systems/storage/spectrum/scale/index.html
- General Parallel File System section of the IBM Knowledge Center

  http://www.ibm.com/support/knowledgecenter/SSFKCN/gpfs_welcome.html
- *IBM Platform Computing Solutions Reference Architectures and Best Practices*, SG24-8169

  http://www.redbooks.ibm.com/abstracts/sg248169.html
- *A Deployment Guide for IBM Spectrum Scale Object*, REDP-5113

  http://www.redbooks.ibm.com/redpieces/abstracts/redp5113.html

1.11.5 IBM Software Defined Network for Virtual Environments

As with software-defined storage, software-defined networking (SDN) means all networking tasks are done inside a software. In practice in means you don't need any sophisticated hardware to manage the network traffic because all this can be done by virtual switches, virtual firewalls, virtual load balancers, virtual routers and so on. SDN provides a solution to significantly reduce OPEX and time-to-market for new applications for the business,

The current reality of server virtualization technology and automation currently provides a way that servers can be provisioned in hours and in some cases minutes, but the bottleneck can often be the network, which requires days, if not weeks, to provision. SDN helps with this as another accelerated puzzle piece and brings network provisioning back on par with servers.

The IBM Software Defined Network for Virtual Environments has a strong set of capabilities. It provides integrated management of OpenFlow networks and Distributed Overlay of Virtual Ethernet (DOVE) networks.

This section describes the technologies supported by SDN.

Supported technologies

IBM Software Defined Network for Virtual Environments (SND VE) is software for networking implementation, based on the OpenDaylight open source platform. SDN VE enables creation of virtual networks that are isolated from physical networks in a way that is similar to how a virtual machine is separated from a physical host. SDN VE integrates with OpenStack Havana via the neutron network component of OpenStack.

IBM provides a neutron plug-in that allows actions performed in OpenStack to be reflected in SDN VE for seamless data center wide network management SDN VE is managed by Cloud Manager via plug-in for OpenStack Neutron networking module.

Distributed Overlay Virtual Ethernet (DOVE) networks are a key capability of the IBM SDE solution. DOVE networks are virtual highly secure layer-3 network connections between VMs. This enables creation of simple virtual networks between application tiers such as database and web server tiers, on a per application basis, that are fully secure, fully isolated, and can be created without network administrator assistance. Isolation between applications is achieved without the hassle of creating VLANs by the network administrator. It is even

possible for one DOVE network domain to use the same IP address range as another DOVE network's domain. DOVE networks are completely virtual networks, independent of the realities of physical networks beneath them, which transport the DOVE virtual network packets.

Another capability of SDN VE DOVE networks is the insertion of virtual or physical appliances, such as firewalls, intrusion prevention systems (IPS), and load balancers, easily in any segment of a DOVE network. This capability allows for network topologies that are much more difficult on physical networks.

The IBM SDE solution uses PCMAE to automate the deployment and configuration of DOVE network agents on individual OpenStack machines, so that OpenStack tenants on those machines can join DOVE virtual networks.

After DOVE networks are deployed in OpenStack, with separate DOVE networks for each cloud user application, IBM application deployment tools, UrbanCode Deploy, and Cloud Manager can specify those DOVE networks for application deployment.

Note: Software Defined Networks and DOVE networks were not implemented in the IBM lab environment that we used for this book, but they will be added to the SDE solution in the next release.

For more information about Software Defined Networks VE, see Appendix A, "IBM Software Defined Network for Virtual Environments" on page 711 and these sources:

- IBM Software Defined Networking

  http://ibm.co/1yeyuTU
- *Implementing IBM Software Defined Network for Virtual Environments*, SG24-8203

  http://www.redbooks.ibm.com/redpieces/abstracts/sg248203.html

1.12 Scenarios for an example company

This section provides use-based scenarios for enablement of the line-of-business cloud users, as well as for supporting cloud providers for a sample company called MyBank.

References to details in the supporting technologies section are provided that illustrate the various SDE technologies and software to support these scenarios.

1.12.1 Enable line-of-business cloud users

For the purposes of illustration, and example, we call the company MyBank. A line-of-business group at MyBank wants to get a customer retention application integrated into its daily business processes quickly. The developers of this application need a representative sandbox to develop and validate their queries of all the various client transactions across their business. Currently this data lives in several data sources, so they will need to also test the ingestion of this data.

The developers have chosen to use IBM BigInsights, with its BigSheets and BigSQL query capabilities based on Hadoop. Hadoop is at the core of IBM BigInsights, since it excels at large quantities of unstructured data.

Note[a]**:** *Apache Hadoop* is an open source software framework for storage and large-scale processing of data sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users[b].

The Apache Hadoop framework is composed of the following modules:

- Hadoop Common, which contains libraries and utilities needed by other Hadoop modules
- Hadoop Distributed File System (HDFS), a distributed file system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster
- Hadoop YARN, a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications
- Hadoop MapReduce, a programming model for large-scale data processing

Beyond HDFS, YARN, and MapReduce, the entire Apache Hadoop open stack is now commonly considered to consist of several related projects: Apache Pig, Apache Hive, Apache HBase, Apache Spark, and others.[c]

For the users, although MapReduce Java code is common, any programming language can be used with Hadoop streaming to implement the *map* and *reduce* parts of the program.[d] Apache Pig, Apache Hive, and Apache Spark, among other related projects, offer higher-level user interfaces, such as Pig Latin and a SQL variant, respectively. The Hadoop framework is mostly written in the Java programming language, with some native code in C and command-line utilities written as shell-scripts.

a. Source: http://en.wikipedia.org/wiki/Apache_Hadoop
b. “Applications and organizations using Hadoop.” Wiki.apache.org. 2013-06-19. Retrieved 2013-10-17.
c. “Hadoop-related projects” at Hadoop.apache.org. Retrieved 2013-10-17.
d. “[nlpatumd] Adventures with Hadoop and Perl.” Mail-archive.com. 2010-05-02. Retrieved 2013-04-05.

The value of the SDE solution combined the BigInsights and Hadoop environment is tighter, and more fine-tuned orchestration and policy management for the workloads which is a benefit for the management of the demands of structured and unstructured data especially for “fast and big” streaming data.

For their three-week prototype development cycle, the developers have chosen the IBM SoftLayer public cloud, where they can pay for the hours of time they actually use, and get started instantly, no need to wait for the IT staff to create IT infrastructure at this point.

Note: For the purposes of this scenario and example, a SoftLayer SDE-based solution is used, using UrbanCode Deploy (see UrbanCode Deploy section on Page 21 in this chapter). Later in this book, we will also illustrate a similar BigInsights scenario in our Redbooks Development Lab SDE environment, which is based on deq - x86 Linux (KVM), IBM PowerLinux™ (KVM), and Power AIX (PowerVM) in terms of platform options.

The developers are able to use UrbanCode Deploy to get a small, representative BigInsights cluster installed and running in SoftLayer in a couple hours.

Due to the sensitive nature of the data involved, the developers will place fictional test data in all the data sources, and quickly are able to test the ingestion of data from these sources into their BigInsights cluster file system.

With test data coming in regularly into their development cluster, they are able to develop their queries, as well as some of the lifecycle components, such as query and report automation, and removal of aged data that is no longer required. UrbanCode is also used to facilitate their development and sustaining of their query and data lifecycle automation code, so that it's easy to continuously feed the latest code refinements into their running cluster, and monitor the effect of those changes real time.

After three weeks of development, the developers are ready to move to a production on-premises cloud with real business data. The same UrbanCode application will be used, with an environment set up for the on-premises cloud at MyBank, as well as the prior public development cloud. Only a couple additional tweaks are required when deploying to the production cloud. A set of GPFS storage pools is configured that is designed to handle streaming of large quantities of data. This is a choice that wasn't applicable in the smaller, development public cloud. Even though this deployment is to an internal production cloud, there was no need to work with the IT department's cloud provider team, because the internal production cloud already has arrangements in place for specifying one of several GPFS storage pool types. The line-of-business developers have all of the tools they need for production deployment and lifecycle management of their client retention application.

After the production deployment is up and running, the inevitable requests for additional queries and reports start to come in. With UrbanCode Deploy and its continuous deployment capabilities, the developers can quickly spin up new instances of their BigInsights development cluster on SoftLayer, iterate a few times on their new queries, and then, in real time, push those new queries into production, and monitor that those queries are working fine in production. Gates within UrbanCode Deploy facilitate higher quality assurance and control when moving application versions between environments.

One of MyBank's recent acquisitions is already using a different application and infrastructure deployment tool set, and wants to keep using that toolset until a migration can be completed. The second tool set works just fine at MyBank, since that tool set also creates the same, industry standard HOT deployment documents as the IBM products. By adopting an industry standard deployment template in the IBM SDE solution, there can be wide interoperability of deployment tools and target clouds. The HOT template provides the "Contract" that all parties to can abide by, giving interoperability and freedom from vendor lock in. See 1.6.2, "IBM pattern engine strategy" on page 14, for more information.

1.12.2 Business expansion: Advanced orchestration

MyBank is finding a sharp increase in the frequency of new application deployments from several lines of business. Much of this increase is coming from recent acquisitions that MyBank has made (for example, Insurance and expansion of Wealth Management and private Banking) making MyBank a large, diverse enterprise. Further, MyBank is finding that several lines of business are now seemingly deploying fairly similar applications. To increase productivity, MyBank invests in IBM Cloud Orchestrator, where MyBank puts many of the common application patterns into the self-service catalog, so when a line-of-business needs to deploy another instance of a fairly common application, such as its customer retention application, in new geographies, or in newly acquired business units, the bronze, silver, or gold versions of the customer retention application can be cloned and adjusted to meet specific needs, and then used to deploy and manage another instance of the customer retention application, quickly, easily, and error free.

MyBank is able to use the application deployment and lifecycle capabilities of UrbanCode Deploy, when its lines of business use Cloud Orchestrator. Each line-of-business can use the Cloud Orchestrator linkages to UrbanCode Deploy to control the version of the IBM BigInsights application used in initial deployment into development and into production. This linkage is invaluable when it's time to move to a new patch level or version of IBM BigInsights. MyBank can use this linkage to maintain its own internally developed query and reporting applications easily and stress free.

1.13 Deploy and managing clouds

In this section, we describe scenarios for the MyBank cloud provider and how the advanced SDE supporting technologies will enable them to make life easier for the line-of-business cloud users.

1.13.1 Cloud provider advanced automation and orchestration

Getting back to our scenario with MyBank, the IT department at MyBank provides a steadily growing set of private, on-premises clouds for a myriad of line-of-business users. On-premises clouds are required for several MyBank applications, because regulatory requirements require certain data to be kept on premises (via an on-premises system of record hosted on IBM Power Servers or the IBM mainframe). Also, MyBank can provide production-level SLAs that might not be possible with hourly public cloud environments. In some cases, MyBank even finds that it is more cost effective for 24x7 production applications to be run on-premises, where it has tight control on actual costs of operation.

MyBank has found that it can standardize its racks of servers with three types of servers present: commodities compute servers, commodity data servers, and enterprise servers. The compute servers are useful as web servers, app servers, and load balancers. The data servers have 12 HDDs each, and can either be used as Hadoop servers or GPFS storage enclosures. Commodity servers are appropriate for compute and data servers, as application availability is achieved through server replication, such as five web servers, where any four web servers can handle the peak load, or the fact that GPFS can be set for multiple copies of data, so that the data is always available, even with a GPFS server down. The Enterprise servers are used for business critical database servers that require a highly available single server for each database.

MyBank has single racks containing only one type of server. MyBank is able to project its quarterly growth of total application and data volume, and purchases and deploys servers a few weeks ahead of their expected need. As new servers are required, these servers can be deployed individually into any of several different clouds easily. Having a consistent rack level architecture enables MyBank to use the IBM Intelligent Cluster process for ordering and fulfilling fully functional racks of equipment. Only external power and network connections are needed when the racks are set in the data center.

MyBank has chosen IBM Platform Cluster Manager Advanced Edition (PCMAE) for its capability to do highly tailored bare metal deployment and can manage multiple IBM OpenStack cloud clusters simultaneously. MyBank uses PCMAE's cluster definition capabilities to automatically deploy and grow various combinations of IBM OpenStack, GPFS, and Software Defined Networks VE. Within the GPFS definitions, MyBank can tailor the storage pool types into "bronze" cost effective storage, "silver" high bandwidth, streaming storage, such as for Hadoop applications, and "gold" high IOPS storage, for transaction-oriented applications. These multiple storage classes or personalities can all be accomplished on the same, standard commodity data server, so they offer significant

software define storage capability. MyBank finds that it is even able to deploy both IBM OpenStack and GPFS on to the same commodity data server for applications that strongly use data locality such as Hadoop. With significant storage capabilities built into the storage rich server, this server type is a great Swiss army knife for the cloud provider!

MyBank has found that it can greatly use IBM Platform Resource Scheduler (PRS) to auto-scale or “flex” individual clouds, with total automation, totally unattended. MyBank has decided to run its compute servers at 2:1 oversubscription, as it finds that the applications running on compute servers, such as web servers and app servers seldom all hit peaks at the same time, but do rather tend to become more busy as additional line-of-business applications are deployed in their cloud.

Further, MyBank has set a policy that says if one or more compute servers hits 75% CPU use, that PRS will provision and deploy another IBM OpenStack compute server, in the spare compute server rack, and have it join the cloud running at 75%, and then PRS will initiate live VM migrations to spread some web server or app server VMs to the newly added compute server, so that now each compute server is now well below 75% CPU use. This level of hands-free auto-scaling could happen all on its own, on an off shift, with no IT staff involvement.

This 2:1 packing of compute server VMs, complemented with auto-scaling provides MyBank with a very cost effective operating model with many workloads at CableCo, that don’t all peak at the same time. MyBank does also offer non-overcommitted 1:1 stripped clouds for its demanding “batch” analytics applications that will always drive near 100% CPU use on the resources provided. Line-of-business application owners choose IBM OpenStack instance sizes in these 1:1 stripped clouds that balance the need for results in a certain time frame with the higher department charge back of running larger, dedicated VMs full-time.

1.14 Benefits and outcomes

The strengths of the various IBM products in the IBM SDE solution are illustrated, but just as important, it is shown that the products integrate into an Open, full service, end-to-end solution.

This enables cloud users to rapidly deploy and manage infrastructure and applications and enables cloud providers to automate the deployment and management of complex clouds. These capabilities extend seamlessly across on-premises private clouds and off-premises private and public clouds.

Challenges across all industries, but especially the financial services and government industries, are similar: IT is measured on keeping the systems stable and functional. The business wants to develop and deploy new functions. There is a chasm to cross here between the two viewpoints (status quo versus new functions) and this is first and foremost a cultural or organizational problem to solve rather than a technology one.

The greatest impact will be on SLOs (service level objectives) and SLAs. Working with clients to understand their objectives and needs from the business perspective of their workloads is essential and becomes even more critical and important. SDE will provide the benefit of “discovery” of the workloads. This will be effective if business rules are rigorously defined up front, particularly from the nonfunctional requirements (NFRs) and the peak use needs.

One very important aspect with SDE is that workloads must be described as machine-readable patterns so that orchestration engines can interpret them, can instantiate the required software-defined resources, deploy the respective middleware and application workloads on-top and manage the complete workload to fulfill service level objectives (SLOs). SLOs should be very clearly defined.

In an industry that is very dynamic such as financial services, and with multiple initiatives potentially occurring in parallel in the business and data center merger and acquisition activity, regulatory, existing system modernization and transformation, big data and analytics, mobile, and social), and with a technology “complexity” multiplier of cloud services coming into the mix, the more technology enablement is needed to provide the shifting and changing business requirements and resources. These can be discovered through SDE workload and resource abstraction.

Table 1-1 summarizes some of the key challenges facing the financial services industry today and indicates how SDE can help generate improved outcomes.

Table 1-1 Key challenges facing financial services

Challenges	SDE benefits and outcomes for the cloud user and business	SDE benefits and outcomes for the cloud provider
Create value in the customer experience	Competitive advantage through increased differentiation by acceleration of new product launches New revenue opportunities by proactively and reactively acting faster to anticipated and unanticipated market changes (for example, financial technology startups, Amazon selling insurance, nontraditional retailers opening banking channels) Increased productivity and ROI by capturing new and preferred practices and patterns for workloads and infrastructure Results: Optimization of customer experience to increase loyalty	Faster orchestration and adaptation to enable time to market through multiple channels (for example, systems of engagement, Internet banking, omni-channel, mobile banking) Support business needs and time to market by automating process and infrastructure through DevOps Accelerate IT transformation activity with less impact on the LOB (back-office modernization to on- and off-premises clouds)
Business diversification	Improved benefits and faster realization of M&A and integration activity (example: a bank buys an insurer)	Faster and standardized enablement for shared services and platforms (for example, bank buys insurer, integration of common MDM platform)
Business expansion into new markets	Access to new products for the user and increased revenue and profits for the business by faster entry into new markets	SDE with OpenStack in a multiregion topology uses open architecture to provide faster, secure entries into new markets and geographies

Challenges	SDE benefits and outcomes for the cloud user and business	SDE benefits and outcomes for the cloud provider
Regulatory and compliance	Improved stability, fewer outages (planned or unplanned) Less revenue Loss or Increased revenue Ability for the business to proactively act and react faster to regulatory guidance changes Increased business continuity	Can provide improved capabilities for predictive analytics and modeling for compliance policy changes, industry stress tests, and improve consolidated market, credit, and operational risk positions through policy-based automation Improved HA automation and reduced DR recovery times provided through PRS and PCMAE with GPFS
Increased service levels through more customers, more users	Improved stability, fewer outages (planned or unplanned)	Faster implementation of policy changes with reduced impact as SDE is integrated SDE able to automatically meet new service levels and objectives through policy and threshold management
Efficiency	Faster application response times Less impact on users and the business. Better use of faciities and enablement of collocation for data centers to reduce costs	Faster implementation of Policy changes with reduced impact as SDE is integrated SDE able to automatically meet new service levels and objectives through policy and threshold management Optimization of data center, power efficiency, floor space reduction
Tighter business and IT integration	Faster access to companies new products for the customer Access anywhere Improved customer service and enablement of shared services	Faster enablement of shared services through heterogenous multiplatform (reduce the dependency on single platform or vendor) Optimization of data center locations

Challenges	SDE benefits and outcomes for the cloud user and business	SDE benefits and outcomes for the cloud provider
Business growth	Faster access to companies new products for the customer Access anywhere Improved customer service and enablement of shared services Customers gain entry into new product offerings faster Retention of customers	SDE provides faster vertical and horizontal auto scaling capabilities through policy management. Patterns enable predictive management, which enables IT to be more nimble and to act proactively, based on capacity projections from the business.

2

Reference architecture framework and methods

This chapter defines the reference architecture, describes the Enterprise Technology Framework, and provides the different implementation methods:

It includes the following sections:

- 2.1, “Various definitions of reference architecture” on page 48
- 2.2, “The Enterprise Technology Framework” on page 49
- 2.3, “Approaches and methods” on page 52

© Copyright IBM Corp. 2015. All rights reserved.

2.1 Various definitions of reference architecture

Different people are likely to define *reference architecture* according to their own perspectives and the context in the business and IT environment where they work.

According to the Wikipedia definition:

> "A reference architecture in the field of software architecture or enterprise architecture provides a template solution for an architecture for a particular domain. It also provides a common vocabulary for discussing implementations, often with the aim to stress commonality. A reference software architecture is a software architecture[1] where the structures and respective elements and relations provide templates for concrete architectures in a particular domain or in a family of software systems.
>
> "A reference architecture often consists of a list of functions and some indication of their interfaces (or APIs) and interactions with each other and with functions located outside of the scope of the reference architecture.
>
> "Reference architectures can be defined at different levels of abstraction. A highly abstract one might show different pieces of equipment on a communications network, each providing different functions. A lower-level one might demonstrate the interactions of procedures (or methods) within a computer program defined to perform a very specific task.
>
> "A reference architecture provides a template, often based on the generalization of a set of solutions. These solutions may have been generalized and structured for the depiction of one or more architecture structures based on the harvesting of a set of patterns that have been observed in a number of successful implementations. Further it shows how to compose these parts together into a solution. Reference architectures will be instantiated for a particular domain or for specific projects.
>
> "Adopting a reference architecture within an organization accelerates delivery through the re-use of an effective solution and provides a basis for governance to ensure the consistency and applicability of technology use within an organization."[2]

There are various IBM definitions:

- A reference architecture is a resource containing a consistent set of architectural preferred practices for use by all the teams in your organization.[3]
- A reference architecture is in essence, a predefined architectural pattern, or set of patterns, possibly partially or completely instantiated, designed, and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use. Often, these artifacts are harvested from previous projects. [4]
- A reference architecture is meant to ultimately drive consistency across business units in an organization, aligned to use cases and industry scenarios. Its purpose is to inform and guide based on preferred practices. It provides the critical linkage between strategy and implementation.
- Fundamentally, a reference architecture is a proven template-based solution for architecture for a particular domain.

---

[1] Clements, Paul Felix Bachmann; Len Bass; David Garlan; James Ivers; Reed Little; Paulo Merson; Robert Nord; and Judith Stafford. *Documenting Software Architectures: Views and Beyond*, Second Edition. (2010) Boston: Addison-Wesley. ISBN 0-321-55268-7.

[2] Wikpedia http://en.wikipedia.org/wiki/Reference_architecture

[3] Reed, Paul. "Reference architecture: The best of best practices,"including comments. IBM developerWorks, 2002. http://www.ibm.com/developerworks/rational/library/2774.html

[4] Reed, Paul. Cited previously.

2.2 The Enterprise Technology Framework

As one example of a reference architecture to start with and one that can be applied to many solutions, IBM has developed an Enterprise Technology Framework. For background, see “The Enterprise Technology Framework as applied to cloud initiatives” by William Genovese in the IBM “Thoughts on Cloud” blog archive:

http://ibm.co/1GyA1Tc

When you build a house, do you do it haphazardly, without requirements or specifications? And, of course, you want to ensure that the house has a solid foundation, running water, electricity, access to public services (or you will need to provide your own private services, in some cases), cable TV, Internet and telephone service, physical security services, and zones in the house. You also need to abide by specific zoning regulations of the town and state where you live. You might have selected a plot of land and have an idea in your mind about what the house will look like, and you will probably draw some diagrams or even hire an architect to provide a well-designed and detailed blueprint or set of blueprints. Those will show both your private services and the public services that you need to access, which are provided by the city or county (or province), state, or even involve the federal government.

The same holds true for corporations that are now venturing into cloud services. They use a well-defined strategy with well-defined services and a blueprint or framework. These specify building blocks that can help with investment decisions. They also stipulate what services might be further developed in determining public, private, or hybrid cloud models for the corporation.

An Enterprise Technology Framework is key in an enterprise-wide business and IT architecture as part of enterprise architecture. The framework defines the technology services and functions (IT capabilities) that are required to support the business applications and data, including common (or shared) application services, common data services, common system services, network services, security services, platform services, and the management tools used to support the delivery of IT service. It also helps to define the specifics for a line of business that might be required, or, in the case of a SDE that uses a hybrid cloud model, what system or application must stay in the data center at the corporation (for example, a system of record versus what might be hosted in a SaaS public model or accessed through a system of engagement on a mobile device). The Enterprise Technology Framework must be totally aligned with the business applications and data, if the business goals, objectives, and benefits are to be realized. This reference framework can also help define what are private or bounded services and the policies and patterns for these services in relation to the enterprise for SDE.

This work product may be produced to varying levels of detail, depending on the stage at which it is produced. It does not represent a complete functional specification of the future IT requirements. Rather, it defines them at a sufficient level to allow alternative technologies and solutions to be evaluated and investment decisions to be made before proceeding with the design and implementation of specific hardware and software components and application systems. That is very important for a SDE and cloud environments.

The Enterprise Technology Framework is used for the following tasks:

- Provide a repository of information about the technology (IT enablers and capabilities) that are required to support both the various parts of the business and the achievement of the business goals and objectives. This provides a framework for making IT investment decisions.
- Provide a repository of agreed upon technology principles, standards, products and components that can be selected at system design time and implemented.

- Reduce the amount of time spent by individual development projects in the evaluation and selection of products and components.
- Provide predefined combinations of implementable components, standards, and interfaces.
- Ensure that individual systems can be integrated effectively, including the sharing of common services, functions, middleware, and data.
- Provide a known technology base for service delivery planning (capacity, performance, and availability) and measurement to meet future business requirements.
- Provide the basis for the specification requirements for the proposed IT systems.

Figure 2-1 is a customer example. The blocks in red are the gaps that were agreed to be closed by the program funding, based on the enterprise initiative (in this case, an analytics with MDM initiative, public and private cloud, and portal, security, and infrastructure improvements provided by Software Defined Compute, networking, and storage). This is used to identify opportunities to enhance the current or new information technologies that can be benefit the IT environment.

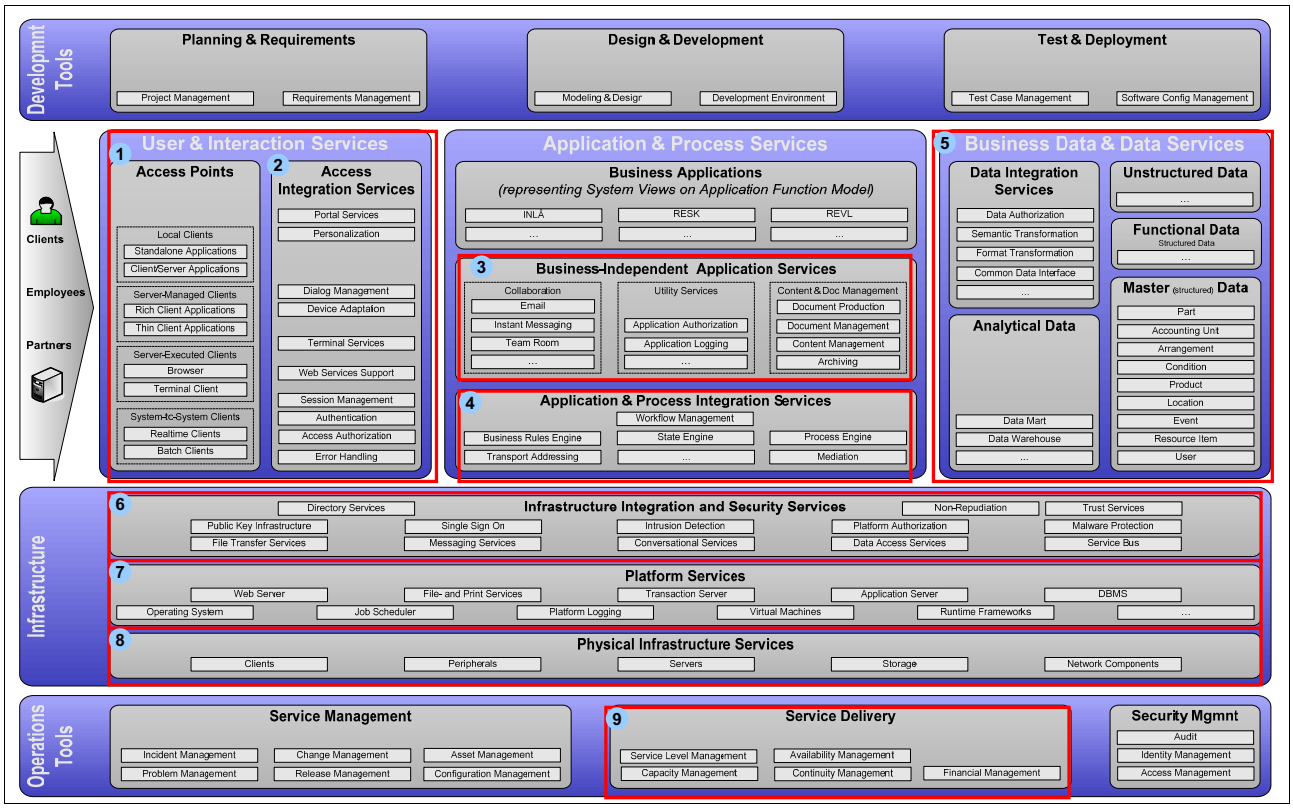


Figure 2-1 The Enterprise Technology Framework

Per Figure 2-1, the following content details insights on the Enterprise Technology Framework:

1. Access points. Develop and execute an user platform strategy, roadmap, and deliver a solution that addresses the needs of all international locations and drives toward a modern standards-based thin client model.

2. Access integration services. Develop and execute a strategy and deliver a solution for transforming existing user applications to use a portal shared services model.

3. Business-independent application services. Develop a strategy for externalizing business independent services (from their existing proprietary applications) and hosting them as a set of shared services.

4. Application and process integration services. Develop a strategy and solution for implementing a set of process integration services with appropriate handling of business events and provision of business activity and process monitoring.

5. Business data and data services. Externalize, decouple, and centralize Customer, Product, and Arrangement information as a set of master data management services in a solution. Centralize and integrate all data warehouses and data marts and their associated services (ETL, data quality management, and so on).

6. Infrastructure integration and security services. Develop a robust security architecture and solution aligned with corporate standards and business needs, and meeting the appropriate regulatory standards (for example: AICPA, SOC2, FedRamp, COBIT 5.0, ISO/IEC 27001:2013, NIST SP800-53 R3).

7. Platform services. Rationalize and consolidate workloads on a minimal set of compute nodes while meeting performance, scaling, availability, and disaster recovery requirements strategically through software-define compute, software-defined network, and software-defined storage.

8. Physical infrastructure services. Optimize use of servers, storage and network components through software-defined compute, software-defined network, software-defined storage.

9. Service delivery. Develop a strategy and implement a solution for systemic application-supported service level management, availability management, capacity management, continuity management, and financial management (ideal requirement for SDE).

Note: In the case of the financial services industry, every large financial institution is either beginning to or currently reviewing their current core systems and considering what to do to position their enterprise for the future. There are a variety of core systems being used by large enterprises. Some are packaged applications, many of which have had considerable customization over their years of use. Some have been custom-developed. More often than not, these systems are large, monolithic, tightly coupled environments where, although feature-rich, are difficult to enhance, costly to maintain, often do not fit the needs of the business, and will probably not be extensible, scalable, or meet the availability requirements for the current or future needs of a large enterprise.

The objective of this narrative is to give a point of view on what the next generation of core systems will look like from an architectural viewpoint. Technology is and will continue to evolve, but with some recent changes and the adoption of cloud computing and the various software as a service offering, it is critical to establish the correct architectural framework to be able to use the changes in technology as we move from the current generation of core systems so we are not facing the same challenges five years from now.

We have a once-in-a-generation chance, as technologists, to create core enterprise solutions. With the correct design philosophy, we can create the next generation of systems that have pluggable functional components using the right technology for the right task and enabling the business to enhance capabilities based on the functional needs.

2.3 Approaches and methods

There are a couple of core considerations as this re-engineering of the core is being considered.

First, the customer interaction models are changing which will drive us to transform our core systems to support the future of banking. The volume of transactions and authorizations will continue to grow dramatically. New offerings such as ApplePay will increase transaction volumes at a rate we have never seen before. Second, our ability to use new compute models and new technologies for the most appropriate workload is critical. When you decompose the function of a core system the transactional, functional, performance and operational characteristics will differ widely across the various domains. A flexible architecture, such as SDE, will allow us to place the right workload on the most appropriate computing platform.

Another key architectural principle is to simplify the current environment. Years of enhancements to our current core systems have created very complex environments. It is critical that we flatten the model and the underlying topology supporting this business function. An enterprise needs to look beyond just moving from package A to package B, programming language A to programming language B or technology stack A to technology stack B. Our objective is to create a flexible ecosystem where the purchased component, programming language or technology stack will meet the need of the individual business domain it supports. The cost and business risk of a modernization program is too high to not take all the right steps and do it right. Having the correct architectural vision and framework is as important as having an architectural blueprint for a skyscraper before starting construction.

So, how is this different? This new model is very different compared to the traditional core systems we have in place now. Most large banks have either a packaged system or a home-grown core banking system that has been in place for many years that is very rich in functions. These systems are generally monolithic, tightly integrated and have grown overly complex to maintain and enhance. The new model of a very componentized, loosely coupled, services centric and most importantly cloud enabled will enable the enterprise to have the business agility to change as the business demands and the flexibility to place use the appropriate infrastructure for the right workload.

2.3.1 The core

This section describes the components that make the solution possible.

SOR versus SOE versus SOI

From a workload standpoint, understanding the distinction and differences between the three computing workload patterns is important. The functional and technical characteristics that differentiate these three workload types will become the basis of how we decompose our systems. The descriptions below provide a high-level description of the three workload types. All business function will fit in one of the three types. The scope of the discussion about “core systems” falls into the systems of record category.

Systems of record

Systems of record are back-office systems that essentially run the company. In banking it would include the core banking, posting, authorization, customer information, product master, payment systems, and so on This is the “single source of truth” in terms of data.

Systems of engagement

The systems of engagement are where users/customers access their data. Ideally little or no data is persisted in this tier. There will also be multiple systems of engagement accessing the back-office systems. Examples include online banking systems, mobile systems, and teller systems.

Systems of insight

Systems of insight provide customers and business the information to make personal and business decisions. The data can be historical or real time depending upon the analytic requirement. It can be business-related or provide insight to technical information such as website usage patterns.

Decomposing our efforts into these three categories enables us to understand but, more importantly, to design our systems and data flows and to reduce the complexity of the environment.

2.3.2 The hybrid core

The idea of a hybrid core differs from what we know of our current core systems. It is common that our core system is a monolithic environment sitting on a common technology stack. If we decompose the functions that our core systems provide, the work performed across the various functions differs significantly. For core banking systems, financial authorizations, payments, profile updates, and postings, as examples, are all very different workload types with different interaction profiles, different data consistency needs, and different performance requirements.

Core design for the next generation of systems

In the design of our next-generation systems, the concept of a heterogeneous “system of systems” will become the basis of our future core platform. This architecture enables us to easily choose which components we build, purchase, or use as a Software as Service (SaaS) offerings from third-party providers.

The most critical part of the next generation core system is a business service tier. These coarse-grained services become the basis for how the enterprise fulfills customer requests. This services tier will serve requests from any channel and fulfill the requests from the underlying components, either within the enterprise or externally, where appropriate. Ideally, this layer is thin and stateless with the supporting components handling transactions.

The underlying components are where the hybrid element of the solution will become apparent. As an example, the posting engine and authorization engine might run on “big iron” where scalability, availability, extreme I/O requirements, and, especially, the transactional integrity of the work is critical. Less critical components where eventual consistency of data is acceptable might be federated across different infrastructure.

Decomposing the domains and understanding the workload characteristics is a relatively simple but important first task as we define the architecture.

Attributes of the hybrid core

The following are the attributes of the hybrid core.

Services centric

We have been talking about the benefits of a services centric, service-oriented architecture for more than a decade. This principle applies now more than ever. The focus for the business should be on a relatively small set of coarse-grained services that are exposed to the channels. Those services abstract the channel systems from the underlying systems that fulfill the requests. A service-centric architecture also enables the business to replace and enhance the underlying service provider non-disruptively as the channel is only concerned with the services tier.

Cloud-capable

There are two dimensions to consider with respect to cloud computing. Often we think of the traditional infrastructure tier and using the cloud to host applications. When it comes to our core systems we also must consider how our new solution is able to use business function from software as a service (SaaS) providers. Having the flexibility to use portions of our business function from external providers will give us the flexibility to use non-critical functions in a variable model to enable the business to focus on the components that distinguish the IBM approach. An example would be the hosting of documents with an external provider. The services tier and information lifecycle could still be owned by the enterprise but the hosting of the documents could be moved to an external provider, probably at a lower cost.

Loosely coupled

The functional components of the next generation system will consist of loosely coupled components. Our services and process tiers provide the isolation from the components, but also orchestrate service and transaction calls across the components. As we explain later, this model also enables us to deploy the functional components to the most appropriate infrastructure in a SDE environment based upon the function they provide.

Massive scalability requirements

Changes in banking channels and customer access will continue to drive transaction volumes. The evolution from phone banking, to Internet banking, to mobile and most recently to mobile payment offerings will continue to increase volumes at a rapid rate. The idea of “bite size banking”, where there will be high volume, low value transaction activity 24X7 will become the new norm. The transactional elements of the next generation system must accommodate for this continued evolution in customer access and customer transactional habits.

The right technology for the right function

As we consider a model with components providing the various business functions we have the opportunity to host those components on the SDE infrastructure that is most appropriate, and provision workload accordingly based on demands. We look at such characteristics such as transaction volumes, create, retrieve, update, and delete ratios, data consistency requirements, synchronous versus asynchronous business processes, and so on to determine how to most appropriately host each component. We might host some components as active/active across long geographic distances, but we might need to maintain a single instance for absolute data consistency. The hybrid model moves away from the one-size-fits all approach for component hosting. This applies to all elements of the technology stack, including technologies in memory databases, solid-state storage, column store, map reduce, messaging technologies, and so on. Application of the right technology to the right task will give us the characteristics which best fit the task at hand for each component.

If you consider the difference between a financial authorization system versus a system which maintains end of day net asset value of funds, they differ significantly in terms of performance, scalability, and so on. Application of the right technology for the right task provides the optimal solution for the task at hand.[5]

In conclusion, the Enterprise Technology Framework can help identify what is required before embarking on large-scale journeys into the technical unknown. In the absence of a defined framework of IT enablers, capabilities and requirements, many assumptions and design decisions may be made in a vacuum (especially at the project level), greatly increasing the risk that the business objectives, requirements and expectations may not be met, putting the enterprise at risk, OR the enterprise as a whole may miss out on greater opportunities that are only achieved at a project level.

Note: The Enterprise Technology Framework is an IBM asset and an architectural work product. For more information see this web page:

Artifact: Enterprise Technology Framework (ARC 308)

http://ibm.co/1yN7jiV

2.3.3 Benefits of reference architectures

"Great," you say, "but what are the benefits to an organization by embracing and using reference architectures?" There are both tangible and intangible benefits:

Tangible benefits

- Cost avoidance or cost reduction, or both
- Avoid purchase of incompatible architectures
- Shortened development and deployment times
- Uses installed architecture
- Reduce duplication and waste
- Accelerated preparation and delivery
- Delivers preferred practices in a standardized, methodical way
- Ensures consistency and quality across the development and delivery process
- Mitigates risk by taking an asset-based approach to solution development
- Reduced support and maintenance costs
- Use support staff
- Reduce learning cycle

Some of the greatest value comes from intangible things.

Intangible benefits

- Common vision and principles between the business and IT
- Consensus approach serving common goals
- Promotes healthy discussion, bringing issues and concerns out in the open
- Creates deliverables and processes as an organization
- Enhanced communications
- Common language
- Centralized information
- Common technology inventory
- Logical models of technology
- Increased knowledge base

[5] Next Gen Core System, Narrative Discussion, 28 October 2014, Tom Parette Client Technical Leader, Bank of America Account

3

SDE reference architecture

This chapter provides examples of SDE reference architecture implementations that are aligned with the use cases and scenarios in Chapter 1. We describe varying reference architecture solutions that are based on the ideal fit for these use cases and scenarios, including our Redbooks reference architecture implementation. This provides guidance for where to start for referential hardware and software building blocks to create the full-scale SDE solution.

We cover the following topics:

- 3.1, “A closer look at SDE reference architecture” on page 58
- 3.2, “SDE with OpenStack reference security architecture” on page 69
- 3.3, “Servers: x86, Power Systems, System z” on page 77
- 3.4, “Hypervisors xKVM, PowerKVM, PowerVM, z/VM, VMware” on page 102
- 3.5, “Storage GPFS, GPFS Storage Server, XIV, V7K, DS8K” on page 132
- 3.6, “Examples” on page 180

© Copyright IBM Corp. 2015. All rights reserved.

3.1 A closer look at SDE reference architecture

The SDE reference architecture and the supporting technologies (specifically in the context of this solution) are described in 1.6, “Supporting technologies” on page 9**.** In this section, we go into a further detail.

The business requirements of our clients are changing rapidly and are driving new complexity and volatility in IT. Existing models of business operations (enterprise applications, core data and transactions, and operational analytics) are fairly static, “codified” and hard-wired into silos in the data center. The new models of engagement (Mobile, Social, big data, analytics and cloud) and the need to transform digitally are creating larger, more complex, and more interconnected and holistic demands upon are clients. These new models of engagement require a more agile, programmable infrastructure that is more responsive and “elastic” to the business, and faster to react across a heterogeneous landscape to meet the new digital demands. And, it must be more secure and highly resilient than before.

We are at an inflection point; traditional environments no longer meet the needs of the business, and IT can longer respond fast enough due to the accelerated expansion of business needs, existing hard-wired IT complexity, and already constrained resources. Software Defined Environments simplify operations, minimize time-to-value, and maximize business outcomes through programmable infrastructure, sense and response automation, and resource optimization.

SDE is the next evolutionary step in IaaS. It is flexible, efficient, integrated and optimized based on “programmable infrastructure.” SDE uses virtualization to optimize the infrastructure as a starting point building block, but differently than traditional infrastructure. The traditional view of IT virtualization is hardware-centric, and uses a “bottoms-up” approach, with extensive manual process management of individual “islands” or “silos” of homogeneous compute silos and server, storage, and network silos. SDE begins as a “top-down” approach through workload awareness. Servers, storage and network are integrated and SDE manages cross platform through a heterogeneous compute federation. Virtualization is used to manage IT, through advanced programmed automation, and pools of systems are managed as a single system.

The diagram in Figure 3-1 on page 59 depicts the optimization capabilities that the programmable infrastructure of SDE provides.

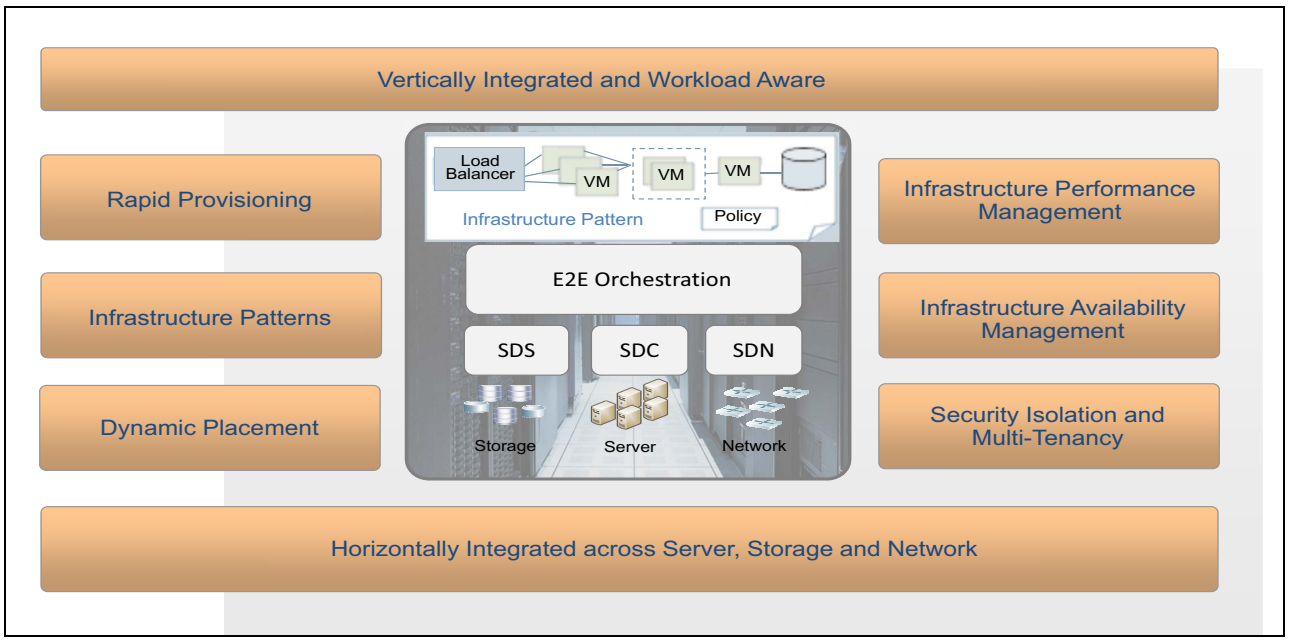


Figure 3-1 SDE optimization capabilities

SDE brings value to both the data center and cloud delivery of IT services, so, in summary the received client value from this new flexible, efficient infrastructure breaks down into these categories:

- Responsiveness:
  - Flexible and elastic infrastructure
  - Radically Improved time to value
- Resiliency and stability:
  - Repeatability and consistency via automation
  - Improved availability
- Cost reduction:
  - Higher use and efficiencies
  - Significantly reduced energy costs
- Security:
  - Security isolation supporting multitenancy

3.1.1 Relationship with the IBM Cloud Enabled Data Center and cloud computing reference architecture

IBM has published and updated a reference architecture for the Cloud Enabled Data Center (CEDC) and cloud computing:

Blueprint for Your Cloud: Cloud Computing Reference Architecture (CCRA)

http://ibm.co/1kh4lpj

In the most recent release (V4.0), the integration and context of SDE with the CEDC and the cloud reference architecture is explained. This is very important, because SDE is a key enabler for the new and next-generation data center and cloud computing.

SDE relates to the CEDC in the following ways:

- CEDC is an adoption pattern of the CCRA.
- SDE is covered in CEDC/CCRA as an emerging technology that raises infrastructure management to a higher level.
- SDE, as a core component of the CEDC, enables the dynamic optimization of the infrastructure resources that are driven through policies.
- SDE serves as the basis for CEDC automation and policy-driven infrastructure.
- SDE provides a set of *IaaS.next* services, enabled by technology components for compute, storage, and network management.
- SDE extends many core uses of the CEDC, and moves the data center model toward policy-driven and pattern-based infrastructure management.

SDE improves the CEDC by placing emphasis on the following factors:

- Abstractions and APIs that make the infrastructure *programmable*
- Orchestration and automation that makes the infrastructure more *dynamic*
- A better *alignment* of IT with the business
- Realizing infrastructure capabilities in software and hardware (rather than hardware alone)
- *Openness* with interfaces and implementations
- Providing extended value for both *traditional investments* and *cloud delivery* models

CEDC is an adoption pattern for cloud service delivery and a *user* of SDE.

3.1.2 SDE reference architecture, logical view

Figure 3-2 provides the logical or functional view of the SDE reference architecture.

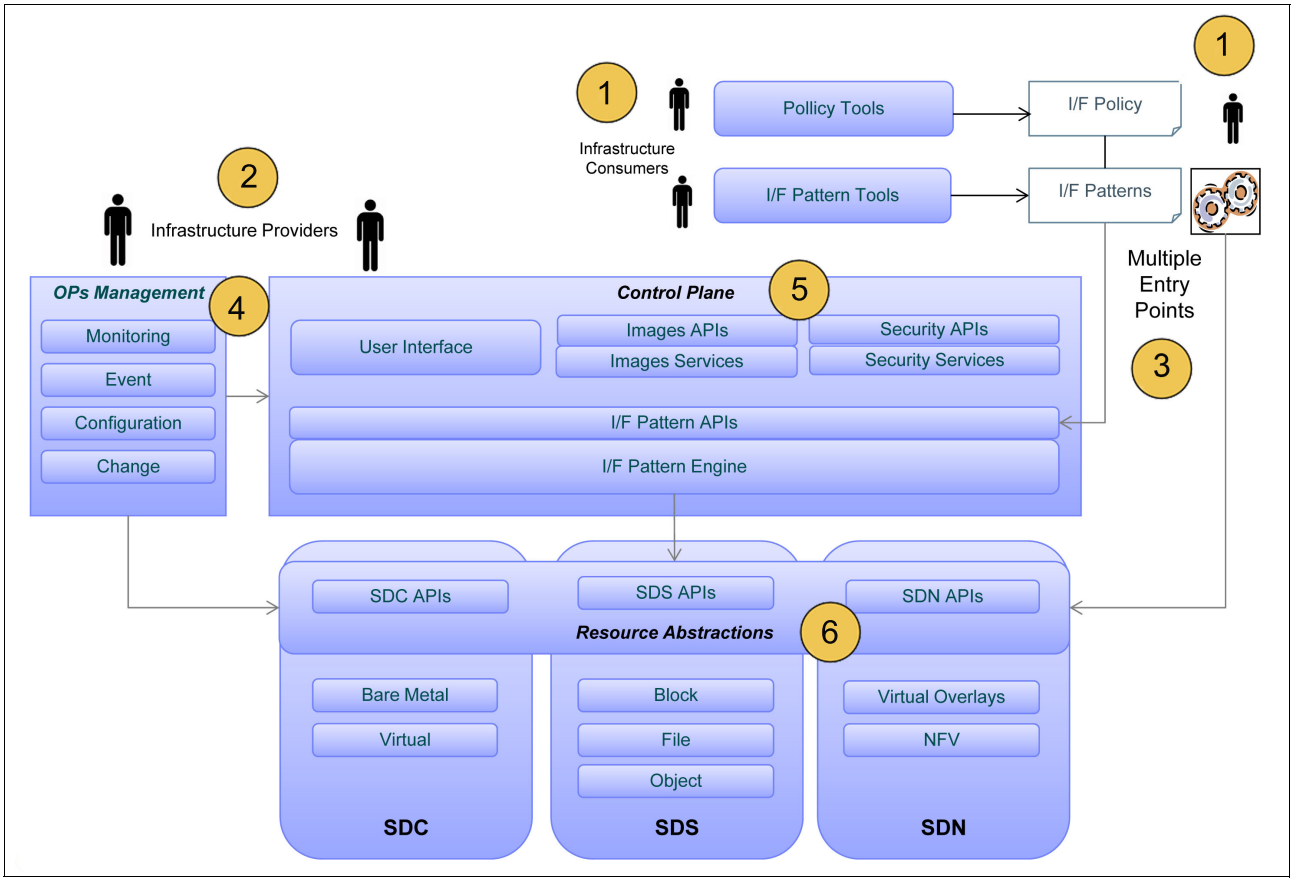


Figure 3-2 SDE Logical reference architecture

From Figure 3-2:

1, 2, 3. Infrastructure users (or cloud users) provide their application and infrastructure policies (for example, resource sizing, use sizing and thresholds, service level agreements (based on functional and non-functional requirements) in working with their infrastructure providers (for example, IT department, or cloud providers). These policies and patterns are typically entered into another application (for example, IBM UrbanCode Deploy, UrbanCode Deploy with Patterns, and Cloud Manager with OpenStack) through the HOT and multiple entry points, as explained in Chapter 1.

4. Operations Management (in the IT department) provides monitoring, event, configuration, and change information via an integrated feed or interface (for example, IBM Tivoli® Netcool® OMNIbus or individual tools' interfaces) to the Control Plane (5).

5. The Control Plane is the "brain" or control center for SDE. It provides a user interface to configure, manage, and deploy images, images services, security, and security services, all through patterns and integrated to the infrastructure layer by resource abstractions (and defined by policies) through application programming interfaces (APIs). In computer programming, an application programming interface (API) specifies a software component in terms of its operations, their inputs and outputs and underlying types.[1]

[1] Source: Wikipedia http://en.wikipedia.org/wiki/Application_programming_interface

6. These resource abstractions, defined by the policies, and automated and standardized through the patterns, provide the “instructions” and “types” for the workloads to the Software Defined Compute (bare metal or virtual) storage (block, file, object) and network (virtual overlay or network functions virtualization).

3.1.3 SDE reference architecture with OpenStack integration

Figure 3-3, provides a more detailed view from the logical architecture with the OpenStack components added.

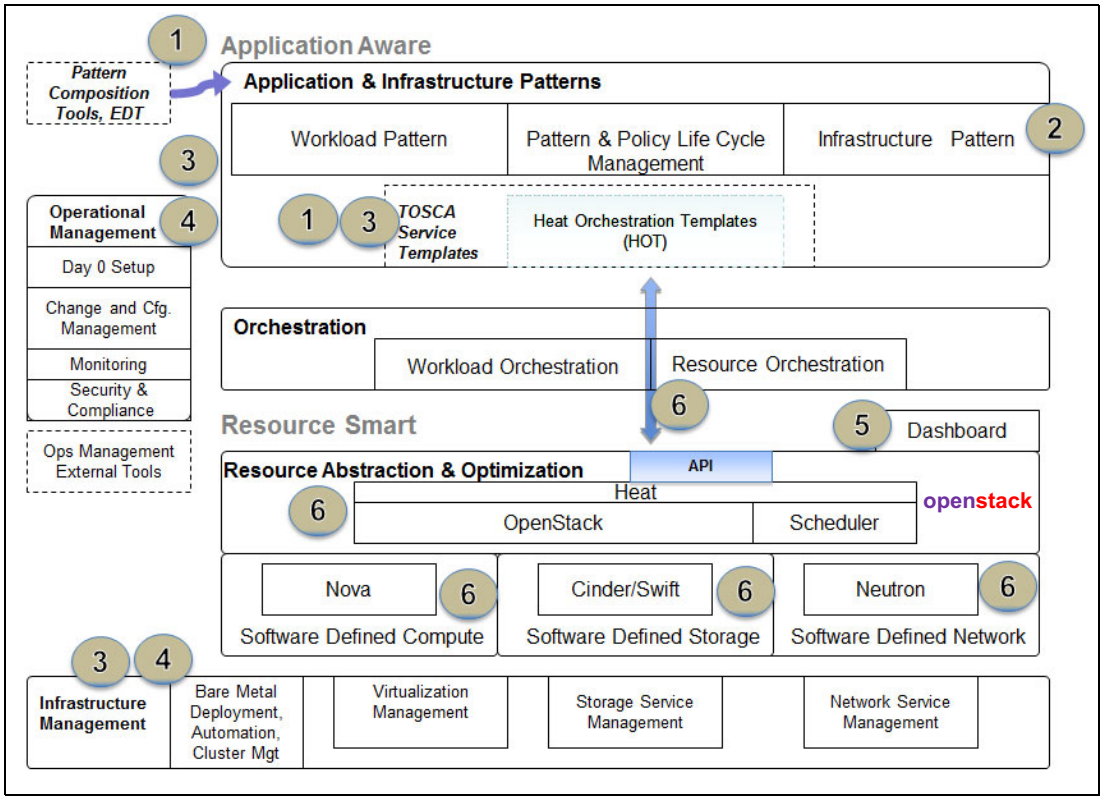


Figure 3-3 SDE detailed reference architecture with OpenStack

From Figure 3-3:

1, 2, 3. Infrastructure users (or cloud users) provide their application and infrastructure policies (for example, resource sizing, use sizing and thresholds, service level agreements (based on functional and non-functional requirements) in working with their infrastructure providers (for example, IT department, or cloud providers). These policies and patterns are typically entered into another application (for example, UrbanCode Deploy, UrbanCode Deploy with Patterns, or Cloud Manager with OpenStack) via the HOT templates, TOSCA service templates, and other entry points, as explained in Chapter 1.

4. Operations management (in the IT Department) provides Day 0 initial setup, monitoring, event, configuration, and change information, and security and compliance policies via an integrated feed or interface (for example, Tivoli Netcool/OMNIbus, or individual tools interfaces) to the Control Plane (5).

5. The control panel is the “brain” or control center for SDE. It provides a User Interface to configure, manage, and deploy images, images services, security, and security services, all through patterns and integrated to the infrastructure layer by resource abstractions (and

defined by policies) through application programming interfaces (APIs). In computer programming, an application programming interface (API) specifies a software component in terms of its operations, their inputs and outputs and underlying types.[2]

6. Workload and resource orchestration, via the APIs, provides resource abstractions, defined by the policies, and automated and standardized through the patterns, to provide the “instructions set” and “types” for the workloads to the Software Defined Compute (bare metal or virtual) storage (block, file, object) and network (virtual overlay or network functions virtualization).

SDE reference architecture and the OpenStack

The beauty of SDE with OpenStack is its openness and flexibility to meet dynamic, changing policies and corresponding patterns. However, with all solution architectures the right solution still comes down to meeting the needs and requirements of the customer, and at least providing the right infrastructure aligned and optimized for specific workloads based on workload characteristics. This is one of the key points and benefits of SDE when using the right infrastructure--the optimization cycle can be shorter, as the solution stack is application aware and can sense and respond quicker.

In this SDE solution, we defined a solution stack that is broad enough to meet most scenarios, with additional optimization-based workload and platform fit--specific to the software workload and transactional scenarios developed. In Chapter 3, Section 3.3, we explain the SDE Redbooks reference solution architecture topology (with the components), that were implemented. In this chapter, in the following sections we explain other reference architectures specific to the hardware platforms we used, hypervisors, storage, the specific workload-based reference architectures (software and hardware), and finally a starting point reference security architecture for SDE with OpenStack.

In the hybrid cloud created for this publication, three different virtualization technologies are used for server virtualization: KVM/QEMU, PowerKVM/QEMU and PowerVC/PowerVM. There are two Ethernet networks connected to all the physical servers, where the *public network* is for management only and the *private network* is for instances intercommunication. SVC and IBM Spectrum Scale (formerly called IBM General Parallel File System, GPFS) are used as elastic storage to provide backing devices to the instances. Figure 3-4 on page 64 illustrates the SDE reference architecture topology (with the components).

[2] Source: Wikipedia http://en.wikipedia.org/wiki/Application_programming_interface

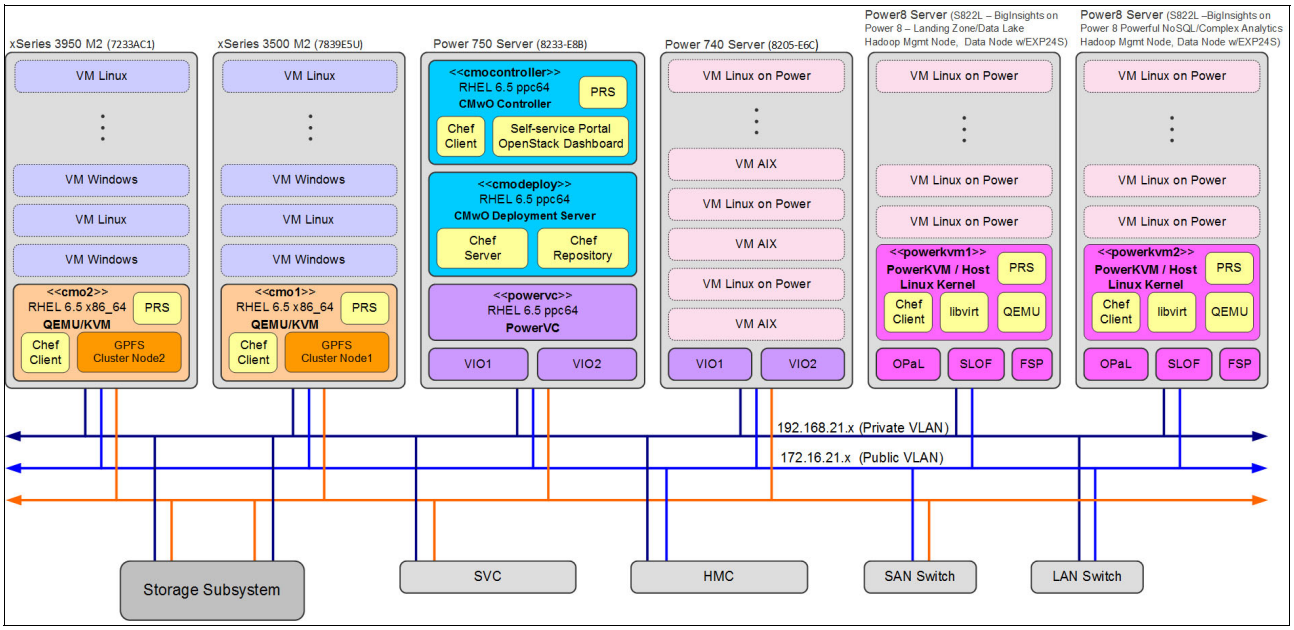


Figure 3-4 SDE topology reference architecture

The cloud topology (as shown in Figure 3-4) used for this residency is *Controller + n Compute nodes*. The topology of the architecture and configuration is summarized in Table 3-1 and also located in 4.3.2, “Cloud topology for this publication” on page 201. For more details, see 4.3, “Cloud setup for this publication” on page 198.

Table 3-1 Configuration topology summary

Node	Description
cmodeploy	Deployment server running RHEL 6.5 on POWER System in IBM Power 740+ Express server 8205-E6C LPAR with ICM 4.1.0.1*
cmocontroller	Controller node running RHEL 6.5 on POWER System LPAR in the IBM POWER 740+ Express server 8205-E6C.The Controller node manages cmo1 and cmo2 via the self-service portal OpenStack dashboard (IBM Cloud Manager 4.1.01*) as KVM Compute nodes, powerkvm1 and powerkvm2 as PowerKVM Compute nodes and it also has the PowerVC driver enabled to communicate with the PowerVC server. PRS 2.2 is also enabled.
IBM x3500 M2 (839E5U)	CMO 1 x86 server running VM Linux, VM Windows, QEMU/KVM, PRS, Chef Client, GPFS Cluster Node 1
IBM x3950 M2 (7233AC1)	CMO 2 x86 server running VM Linux, VM Windows, QEMU/KVM, PRS, Chef Client, GPFS Cluster Node 2
IBM PowerVC	PowerVC 1.2.1.1 server running RHEL 6.5 on POWER System LPAR in the p740 server and managing both p740 and p750.
IBM p750 (8233-E8B)	IBM POWER7+™ Server running IBM Cloud Manager Controller (composed of PRS, Chef Client, and self-service portal OpenStack dashboard), The Cloud Manager deployment server consists of the Chef server and Chef Repository, PowerVC, and VIOS1 and VIOS 2.
IBM p740 (8205-E6C	POWER7+ Server running PowerLinux VMs, AIX VMs, and VIOS1 and VIOS 2.
IBM p740_2_vio1	Virtual I/O Server 2.2.3.3 running in the p740+ to provide network virtualization and NPIV to the LPARs from p740 server.

Node	Description
IBM p740_2_vio2	Virtual I/O Server 2.2.3.3 running in the p740+ (8205-E6C)to provide network virtualization and NPIV to the LPARs from p740 server.
IBM p750_1_vio1	Virtual I/O Server 2.2.3.2 running in the IBM Power 750 Express server (8233-E8B) to provide network virtualization and NPIV to the LPARs from p750 server.
IBM p750_1_vio2	Virtual I/O Server 2.2.3.2 running in the IBM Power 750 Express server (8233-E8B) to provide network virtualization and NPIV to the LPARs from p750 server.
IBM p8 S822L	POWER8 Server running Linux VMs, PowerKVM, PRS, Chef Clients, libvirt, QEMU, OPAL, SLOF, FSP.
IBM p8 S822L	POWER8 Server running Linux VMs, PowerKVM, PRS, Chef Clients, libvirt, QEMU, OPAL, SLOF, FSP.
cmo1	KVM Compute node running RHEL 6.5 on System IBM System x3500 M2 server -(7839E5U).
cmo2	KVM Compute node running RHEL 6.5 on IBM System x3950 M2 -(7233AC1).
IBM powerkvm1	PowerKVM Compute node running PowerKVM 2.1.0 on POWER8 (IBM PowerLinux 8233-E8B) server.
IBM powerkvm2	PowerKVM Compute node running PowerKVM 2.1.0 on POWER8 (IBM PowerLinux 8233-E8B) Server.
QEMU	QEMU (short for “quick emulator”) is a free and open source-hosted hypervisor that performs hardware virtualization (not to be confused with hardware-assisted virtualization). QEMU is a hosted virtual machine monitor. It emulates central processing units through dynamic binary translation and provides a set of device models, enabling it to run a variety of unmodified guest operating systems. It also can be used together with KVM to run virtual machines at near-native speed (requiring hardware virtualization extensions on x86 machines). QEMU can also be used purely for CPU emulation for user-level processes, which allows applications that were compiled for one architecture to be run on another. QEMU is running on the x86 servers, and POWER8 servers in the IBM SDE solution.
libvirt	libvirt is a hypervisor management library. libvirt provides a stable, cross-platform interface for higher-level management tools: ► Used to manage guests, virtual networks and storage on the KVM host ► Provides APIs for management ► The configuration of each guest is stored in an XML file. ► Allows remote management of guests -Encryption, certificates (TLS), authentication (SASL) ► Communication between libvirt and tools management is done via a daemon called *libvirtd* -Check status: “*systemctl status libvirtd*”
IBM SVC	SVC connected to the PowerVC and Controller nodes running SVC 2145 model 8G4 (6.4.0.0).
IBM Spectrum Scale (was GPFS)	GPFS Cluster running on KVM Compute nodes on top of SVC disks. GPFS Version 4.1.0.0.
IBM HMC	HMC V8R8.1.0.0 under control of PowerVC.
IBM SAN Switch	SAN Switch IBM 2498-B40 running FOS v7.1.1 under control of PowerVC.
IBM AIX LPARs	Several AIX 7100-03-00 LPARs as instances being deployed in the cloud.

Node	Description
IBM UrbanCode Deploy	Virtual machine in the cloud running x86_64 RHEL 6.5 and UrbanCode Deploy 6.1.0.1.532639 and UrbanCode Deploy with Patterns 6.1.0.1.545068 as Developer tool.
Chef repository	The Chef repository is the location (in our solution on a p740 Linux partition as a component of the IBM Cloud Manager (formerly Cloud Manager with Orchestrator, or Cloud Manager, deployment server)) in which the following data objects are stored: ► Cookbooks (recipes, versions, cookbook attributes, resources, providers, libraries, and templates) ► Roles ► Data bags ► Environments ► Configuration files (for clients, workstations, and servers) The Chef repository is located on a workstation and should be synchronized with a version control system, such as Git. All of the data in the Chef repository should be treated like source code. Knife is used to upload data to the Chef server from the Chef repository. After it is uploaded, that data is used by the Chef client to manage all of the nodes that are registered with the Chef server and to ensure that the correct cookbooks, environments, roles, and other settings are applied to nodes correctly.
Chef server	The Chef server acts as a hub for configuration data. The Chef server stores cookbooks, the policies that are applied to nodes, and metadata that describes each registered node that is being managed by the chef-client. Nodes use the chef-client to ask the Chef server for configuration details, such as recipes, templates, and file distributions. The chef client then does as much of the configuration work as possible on the nodes themselves (and not on the Chef server). This scalable approach distributes the configuration effort throughout the organization. The Chef server is installed in our solution on a p740 Linux partition as a component of the IBM Cloud Manager.
Chef client	A chef client is an agent that runs locally on every node that is under management by Chef. When a chef client is run, it will perform all of the steps that are required to bring the node into the expected state, including: ► Registering and authenticating the node with the Chef server ► Building the node object ► Synchronizing cookbooks ► Compiling the resource collection by loading each of the required cookbooks, including recipes, attributes, and all other dependencies ► Taking the appropriate and required actions to configure the node ► Looking for exceptions and notifications, handling each as required **Note:** The Chef client executable file can be run as a daemon. The Chef clients are installed on the x3500, x3950 M2 servers, p750 server, and POWER8 servers.

Node	Description
IBM Platform Resource Scheduler (PRS)	IBM Platform Resource Scheduler (PRS) prioritizes resources to the most important applications, enables resource consolidation while preserving service level agreements, increases operational efficiency by optimizing compute and memory use, and decreases power use while reducing administrator requirements. IBM Platform Computing PRS is a significant differentiator as a technology component in the SDE solution with OpenStack, and in the software-defined infrastructure (SDI) solution family in an open architecture providing the highest levels of customization to meet your company-specific objectives with heterogeneous resource pools, and providing automated smart scheduling and optimization for service level agreements. Platform Resource Scheduler offers several benefits: ► A flexible hierarchical organizational structure and an intelligent allocation model that aligns resource capacity to departments for sharing, isolation, or optimizing quality of service while maximizing use. ► An intelligent policy-based allocation engine that places workloads based upon business objectives such as use, energy conservation, application performance, reliability, and availability. ► An optimization of runtime environments through policies for strategies such as packing, load balancing, energy use, and performance, combined with real-time monitoring, such as memory, compute, I/O, and network. ► Advanced capacity planning and reservations to make sure the resources will be available when needed. PRS is installed on the x3650 M2 servers, p750 server, and POWER8 servers. Please see Chapter 1 and supporting technologies where PRS is described in the introduction and further benefits are provided.
Firmware or OPAL (OPAL)	OPAL is the system firmware in the stack of POWER8 processor-based server, PowerKVM. It consists of three major parts, called skiboot, skiroot, and skirun: ► OPAL boot (skiboot) This is a binary component (a Load ID) loaded by the FSP which performs some system initialization and setup (PCIe and so on) and provides runtime services abstraction to a client operating system. Those services range from accessing the real-time clock, rebooting the system, various PCIe PCI Host Bridging (PHB) related functions and nvram. OPAL additionally loads and transfers control to the OPAL loader during boot. ► OPAL root (skiroot) This a Linux-based boot loader that discovers I/O devices and loads a Linux-based hypervisor from a disk or network ► OPAL runtime (skirun) OPAL runtime is a Linux system (kernel + ramdisk) that is stored in a separate local ID. This system is used to access I/O devices (such as disks, network, and so on) and locate and load the host Linux system to be run on the system in hypervisor mode.
Slimline Open Firmware (SLOF)	Slimline Open Firmware (SLOF) is initialization and boot source code based on the IEEE-1275 standard (also known as the Open Firmware standard). If you are a software engineer interested in IBM PowerPC®-based systems, it can be invaluable when developing boot firmware, an operating system, or low-level application code. For hardware engineers, it provides fast start of PowerPC systems, verification, and prototyping of hardware support and drivers. SLOF is important, too, in that it enables a vendor to separate the low-level firmware from the SLOF layer. SLOF is part of the Power8 servers architecture.

Node	Description
Flexible service processor (FSP)	The FSP is an advanced service processor that provides function to initialize the system chip sets, boot to a hypervisor run time, interface to the management module, and provide runtime support for advanced reliability, availability, and serviceability (RAS) functions. FSP is an additional layer in the hardware to closely monitor the other hardware components, and during the event, it notifies the system administrator or vendor to work proactively before the failure to avoid disruption in the service and so on. FSP is part of the POWER8 servers architecture.

Note: Due to hardware availability in our lab environment, each of the x86 servers and POWER servers do not match identically, in terms of the sizing and CPU processor speed. It is true, and it is best to have identically sized hardware platforms as a starting point in your solution. However, remember that with SDE, the approach is different; top-down workload and application aware. Servers, storage and network are integrated and SDE manages cross platform through a heterogeneous compute federation, so it can work with existing infrastructure and adjust accordingly, based on the sizings of individual servers. In a bottoms-up infrastructure approach, having different sizings for the infrastructure, can be even more problematic, and extra capacity is added as a buffer. Therefore, extensive manual process management of individual “islands” or “silos” of homogeneous compute silos and server, storage, and network silos occurs to “tune” and adjust at the infrastructure layer, based on dynamic workload demands.

Note: In this SDE solution, we deployed IBM Cloud Manager V4.1.0.1. Make sure that you are aware of the following product withdrawal and the new release information (V4.2):

- *Software withdrawal:* IBM SmartCloud Entry V3 and IBM Cloud Manager V4.1

  http://ibm.co/1cXnRv5

  Effective date: 9 January 2015
- *Announcement:* IBM Cloud Manager V4.2 delivers an enhanced OpenStack experience and simplified licensing

  http://ibm.co/1E8L8o7

  Planned availability date: 12 December 2014
- *Announcement:* IBM Cloud Manager V4.2 delivers a quick and easy multiplatform cloud solution for IBM Power Systems servers

  http://ibm.co/1QgTWx4

  Planned availability date: 12 December 2014

The following are some of the key new features for IBM Cloud Managerversion 4.2:

- *Juno support.* Full access to OpenStack Juno APIs, backed with IBM support

  Note: Trove and Swift are available via OpenStack APIs only. They are not configured by the deployer.

- *Simplified deployment.* A new simplified command-line method for deploying clouds in the KVM or QEMU, PowerKVM and PowerVC environments.
  - With this option of deploying clouds, you use an example cloud file that is configured with typical defaults.
  - You can customize the example cloud file to meet your needs.
- *Advanced scheduling.* Integrated advanced scheduling via IBM Platform Resource Scheduler v2.2 (available with the IBM Cloud Manager).
- *Seamless in-place upgrade.* Ability to seamless in-place upgrade the IBM Cloud Manager environment from version 4.1 to version 4.2 with Chef.
- *RabbitMQ support.* Added support for RabbitMQ as the default messaging queue.
- *PowerVC v1.2.2 support:*
  - Using maintenance mode to enable or disable a host system for maintenance
  - Using the Attach interfaces (os-interface) Nova API service to create, list, get details for, and deleted port interfaces.
  - Allocating a floating IP address to an OpenStack instance.
- *PowerKVM 2.1.1 support.* Added support for PowerKVM 2.1.1 with GRE network type support.
- *FIPS and NIST compliance.* Support for configuring your cloud topology to be compliant with the Federal Information Processing Standards (FIPS) and National Institute of Standards and Technology (NIST) for cryptography modules
- *Improving planning documentation.* Enhanced planning worksheets to help get you started.

3.2 SDE with OpenStack reference security architecture

In this section, we describe the SDE with OpenStack reference security architecture.

3.2.1 IBM dynamic cloud security portfolio

Cloud computing is transforming IT, resulting in greater operational efficiencies and lower costs than with many traditional IT deployments. However, cloud computing requires consistent protection against threats—whether the threats originate on-p remises or from an outside source.

The rapid adoption of cloud is taking place as businesses still struggle to safeguard their existing IT systems against attackers who are becoming increasingly sophisticated and more difficult to detect. Currently, 75% of security breaches take days, weeks or even months to be discovered, which significantly increases the damage inflicted by attackers.

New IBM cloud security tools use proven analytics to give companies a clear line of sight into the security status of their entire business—from private data centers, to the cloud—even to an individual employee's mobile device. This unprecedented "single-pane-of-glass" view shows exactly who is using the cloud, what data individuals are accessing and from where they are accessing it.

Designed to be used with different users, whether it be a developer or line-of-business executive, the portfolio also includes analytics and security intelligence for public cloud services such as IBM SoftLayer. In addition, the offering features IBM Managed Security Services platform, which can help secure the cloud for IBM clients as well as clients of companies such as Amazon Web Services and Salesforce.com.

Customers can also take advantage of the intelligence from more than 20 billion daily security events that the IBM Managed Security Services team monitors in more than 130 countries. With this insight, they can identify threats in real time and proactively defend their companies from sophisticated attacks on all fronts.

According to a new IBM study of nearly 150 Chief Information Security Officers (CISOs), although 85% say their organizations are now moving to cloud, almost half expect a major cloud provider to experience a security breach. Despite these concerns, critical workloads processing customer and sensitive data are still moving to the cloud.

The new IBM Dynamic Cloud Security portfolio addresses the security gaps that can exist between on-premises, cloud, software-as-a-service (SaaS) and mobile applications. The portfolio is focused on authenticating access, controlling data, improving visibility and optimizing security operations for the cloud. Developed over the past year by 200 engineers, the new tools can be deployed in the cloud or on-premises, matching the hybrid IT environments customers are managing.

The IBM Dynamic Cloud Security portfolio was created specifically to help enterprises use existing security investments and bridge a mix of traditional IT, private, public, and hybrid cloud models. Deployed in private and hybrid cloud environments, Dynamic Cloud Security provides layered protection, deep insight across the infrastructure and a flexible approach to tailor the security-to-the-cloud use case.

To safeguard the cloud, enterprises must focus on the following key areas:

- *Manage access.* Safely connect people, applications, and devices to the cloud with easy-to-use identity and access management tools built for the cloud.
- *Protect data.* Identify vulnerabilities and prevent attacks targeting sensitive data by using hardened underlying cloud infrastructure and applications.
- *Gain visibility.* Monitor the cloud for security breaches and compliance violations through advanced security analytics and threat intelligence from the cloud.
- *Optimize security operations.* Deploy intelligence-driven security maturity across all environments by assessing and optimizing security practices.

Figure 3-5 on page 71 illustrates the IBM Dynamic Cloud Security portfolio and its capabilities.

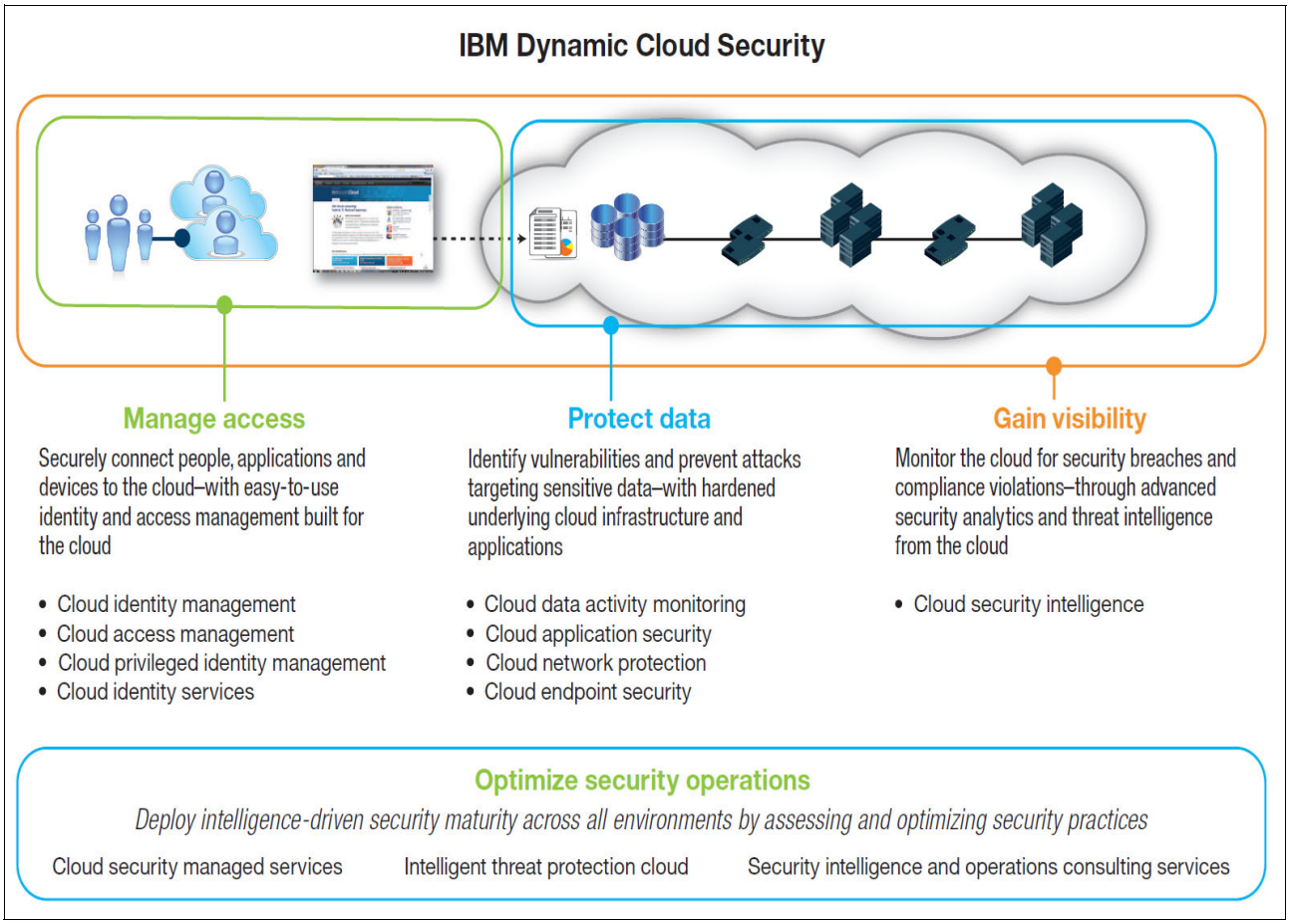


Figure 3-5 IBM Dynamic Cloud Security

Manage access and connect users to the cloud more securely

Some of the IBM Dynamic Cloud Security portfolio's new offerings help secure user access to cloud services as organizations extend their data centers to public cloud environments. They can centralize the granting of correct privileges to users and provide extra security around users who have administrator access to sensitive data. These offerings can monitor and track access to applications with the right level of authentication controls, including multifactor authentication. Another portfolio offering can also help developers build single-sign-on security into their applications using APIs.

Organizations need to provide context-aware access to data and applications to authorized users when they need them, while also blocking unauthorized access. As relationships extend outward to diverse communities of users, organizations also need strong provisioning and auditing capabilities for service and application entitlements.

Cloud identity management

IBM Security Identity and Access Assurance helps users gain access to cloud resources, while also monitoring, controlling and reporting on the identities of the systems, database administrators and other privileged users. Identity federation and rapid onboarding helps extend users to applications beyond the corporate firewall. IBM Federated Identity Manager provides authentication to multiple cloud applications with a single credential, providing self-service for identity creation and management. Built on a standards-based platform, this single sign-on solution helps simplify log-ons for both internally hosted applications and the cloud, enableing users to easily and quickly use cloud services. In addition, the IBM Single

Sign-On solution enables developers to easily add similar user authentication and single sign-on functions to applications running on the IBM Bluemix™ platform.

Lightweight Directory Access Protocol (LDAP) is a client/server protocol for accessing and managing directory information. Many enterprise applications use LDAP as the foundation for user authentication. (Implementations of LDAP include IBM Tivoli Directory Server, Microsoft Active Directory, and OpenLDAP.) IBM has developed and built integration to LDAP through enabling the Keystone LDAP back end in OpenStack.

For more information regarding the OpenStack Keystone integration for LDAP integration, see “Enable the Keystone LDAP back end in OpenStack” in the IBM developerWorks Technical Library:

http://www.ibm.com/developerworks/cloud/library/cl-ldap-keystone/

Cloud access management

IBM Security Access Manager provides a gateway solution to deliver user-level security for workloads deployed on the cloud. A virtual appliance deployment model, available on SoftLayer as well as other cloud platforms, helps administrators get started quickly and scale to thousands of users, maintaining the same level of protection when applications are moved to the cloud by moving access management with them. This enables enterprises to have a singular and consistent policy enforcement point across the data center, private and public clouds.

Cloud privileged identity management

In addition to database administrators and system administrators, cloud computing introduces a new tier of privileged users: operating personnel working for cloud providers. IBM Security Privileged Identity Manager helps manage and control access to critical cloud resources by the organization’s employees as well as personnel who work for cloud providers and have high-level privileged access.

In addition, as more and more enterprise customers begin to deploy OpenStack, it is becoming critical for these customers to seamlessly integrate OpenStack into their existing federated identity infrastructure. Fortunately, since the Icehouse release of OpenStack, new federation capabilities have been added to Keystone, the identity service manager subproject for OpenStack. These capabilities enable Keystone to integrate with identity provider products such as Tivoli Federated Identity Manager.

For more information regarding the OpenStack Keystone integration for federated identity manager, see “Integrate OpenStack Keystone with Tivoli Federated Identity Manager” in the IBM developerWorks Technical Library:

http://www.ibm.com/developerworks/cloud/library/cl-keystone-tfim/

For more information regarding Federated Identity and Federated Service Support for OpenStack Clouds, see the OpenStack Foundation video by that title on YouTube:

https://www.youtube.com/watch?v=5WG4DFXHT18

Cloud identity services

For organizations with capital and staff limitations that still want the benefits of an enterprise-class identity and access solution, IBM Cloud Identity Services provides robust, cloud-based identity and access built on best-in-class IBM software and global delivery capabilities. IBM services use the cloud to deploy identity solutions quickly, at lower ownership costs and with simplified administration. With IBM services, there is no infrastructure for clients to build, and solutions can accommodate the changing needs of dynamic organizations.

Lock down and protect data in the cloud

With the IBM Dynamic Cloud Security portfolio, enterprises can deploy proactive tools to automatically discover, classify and assess sensitive data stored in cloud-hosted repositories, including activity monitoring for both structured and unstructured data. Using the same type of data activity monitoring tools used in the private data center, teams can now monitor activity in the cloud and create centralized auditing for data sources deployed on cloud.

Data protection also involves discovering and remediating vulnerabilities in the applications that access that data, yet developers often lack the security expertise to spot vulnerabilities in their code. The new tools will quickly analyze both Web and mobile applications for security weaknesses. Developers can then remedy those vulnerabilities before they put the application into production or place it in an app store.

Sensitive or regulated data, including runtime and archived data in the cloud, must be protected from unauthorized access. IBM helps improve data governance through data discovery, access management, monitoring and reporting, and by blocking unauthorized access in several essential ways:

- *Cloud data activity monitoring.* IBM InfoSphere® Guardium® Data Security solutions extend protection from the data center to cloud-based structured and unstructured data, sensitive information and intellectual property on SoftLayer. InfoSphere Guardium solutions also provide an efficient and protected audit architecture for cloud- and non-cloud-based data sources, including a centralized security console across different database platforms.
- *Cloud application security.* Today's headlines are filled with news of data breaches that result from application security failures. Insecure coding practices and human error, combined with the relative ease of finding and using application-based vulnerabilities, often make application security a major point of weakness. IBM Security AppScan® provides one of the industry's most comprehensive sets of tools to protect today's enterprise applications. The dynamic analysis platform included in IBM Security AppScan Standard Edition enables continuous testing of production applications that are deployed to the cloud. IBM Security AppScan Source Edition provides source code-scanning capabilities that help development teams discover and remediate security issues in new and existing applications—before placing the applications into production. Today's DevOps environment requires agile application development. Developers who are unfamiliar with secure coding practices can gain application programming interface (API)-based access to security scanning services on Bluemix. IBM AppScan Mobile Analyzer and IBM AppScan Dynamic Analyzer produce vulnerability and remediation reports without the developer having to leave the Bluemix environment.
- *Cloud network protection.* Internet-facing cloud workloads require an advanced level of protection on the network and at endpoints. Powered by IBM X-Force® research and development, IBM Security Network Protection helps shield applications and network infrastructure from exploitation, including zero-day attacks, provides visibility into network traffic, identifies personally identifiable information (PII) and other confidential data, and provides granular application control that prevents users from opening up multiple attack vectors such as malicious URLs, instant messaging protocols and peer-to-peer file sharing to and from cloud resources.
- *Cloud endpoint security.* Unpatched systems, unnecessary services and poor configurations settings are a high risk to cloud deployments. IBM Endpoint Manager for Data Centers optimizes IT efficiency through higher levels of automation and enables users to find and fix problems in minutes across all endpoints (including physical and virtual servers), regardless of operating system. This helps ensure continuous configuration compliance with security and regulatory policies.

Gain visibility across the cloud

With organizations using public cloud services at an increasing rate, they must be able to dynamically analyze the security posture of users, applications, networks, mobile devices and other assets across their enterprise and in the cloud. The Dynamic Cloud Security portfolio extends the IBM QRadar® security analytics platform to the IBM SoftLayer cloud and other public cloud services, such as Amazon Web Services.

By design, clouds obscure underlying application traffic and user activity from their tenants, but this complicates activity monitoring and regulatory compliance reporting. Concerns about visibility and auditing can be roadblocks for many would-be adopters. Before making outsourcing commitments, organizations are also increasingly seeking advanced security incident search capabilities to support forensic investigations.

Cloud security intelligence

IBM QRadar Security Intelligence, anchored by IBM Security QRadar SIEM, answers these concerns by monitoring all traffic going into and out of the cloud using a variety of hardware, software or virtualized software appliances. Cloud activity can be collected and analyzed within the cloud, or it can simply be collected and exported to an on-premises security intelligence resource. Suspect events and high-probability incidents can then be aggregated with complementary security technologies, such as IBM Security Identity and Access Assurance, to correlate what's occurring in the cloud and who's responsible for the actions.

QRadar solutions integrate with a growing list of cloud services and applications including SoftLayer, Amazon Web Services, OpenStack, CloudTrail, Salesforce, CloudPassage, Zscaler, Qualys, and IBM Security Trusteer® Apex advanced malware protection. This enables organizations to embrace infrastructure as a service (IaaS), software as a service (SaaS) and hybrid cloud IT deployments with a combination of on-premises and in-the-cloud security intelligence appliances. QRadar solutions provide an audit trail for cloud-based applications when event and flow collectors are put into the cloud to forward security data to event and flow processors running within the client's data center. Organizations can also choose to implement a fully cloud-based security intelligence solution by collecting and forwarding on-premises data to a cloud-based instance of QRadar.

For VMware ESX and ESXi virtual environments, IBM Security QRadar VFlow Collector appliances provide Layer 7 monitoring and profiling support for more than 1,000 applications. The solution runs as a virtual host inside the hypervisor and can monitor traffic from the virtual switch as well as port-mirrored traffic from a physical switch, which provides visibility in both the traditional and virtual environments that comprise hybrid cloud environments.

A key feature of any cloud infrastructure is providing auditing capabilities for compliance with security, operational, and business processes. In the YouTube talk cited here, we provide an overview of the recent enhancements made in OpenStack projects to support API and security auditing using the DMTF Cloud Auditing Data Federation (CADF) standard. We describe how auditing is seamlessly enabled for Nova, Glance, Swift, Cinder, Neutron, and Keystone and illustrate what is audited, where it is stored, what the records contain and how this supports compliance. We finish by presenting some possible future directions such as extending the use of CADF beyond audit to facilitate event correlation and federation across multiple tiers.

For an overview of cloud auditing support for OpenStack, see the following YouTube video:

An Overview of Cloud Auditing Support for OpenStack

https://www.youtube.com/watch?v=TOBJK_ESpxo

The following video demonstrates how the CADF auditing support added in the IceHouse OpenStack Icehouse release by IBM can be used by tools such as IBM QRadar for security threat analysis and reporting:

http://www.youtube.com/watch?v=QqrzH4l_xHA

Optimize security operations for the cloud

IBM Intelligent Threat Protection Cloud is a managed services platform that monitors the cloud environment. Drawing upon its billions of security events, it includes advanced analytics with new correlation technologies and external data feeds. With data sources growing and companies and infrastructures becoming more dispersed, these analytics can provide real-time insight into what's happening in the cloud, enabling quick detection and response to intrusions into a client's network. This new platform can improve threat response times and help secure these new cloud workloads for any business. These services can be accessed from anywhere at any time and are built for a variety of IT environments in any combination.

IBM Cloud Security Managed Services help clients address the security challenges posed by a cloud environment without the costs involved in acquiring, configuring and managing new protection devices on premises. The offering addresses firewall, email and web security, as well as intrusion protection and event and log management. The service uses integrated security technologies, global threat intelligence and vulnerability research to meet customers' unique demands. Security professionals monitor cloud security around the clock, 365 days a year. Other offerings include emergency response services to help address suspected breaches in a cloud environment and a host-based intrusion detection system for servers.

IBM Security Intelligence and Operations Consulting Services help organizations develop security maturity in intelligence-driven operations across all environments. The IBM Security Intelligence and Operations Consulting Services team can help an organization assess, build or optimize its security intelligence and operations capabilities as it evolves its security organization to safely migrate its business to the cloud.

IBM Managed Security Services, powered by intelligent threat protection for cloud environments, provides big data insights, next-generation intelligence tools and powerful visualization of threats across hybrid cloud and on-premises environments.

For more information about the IBM Dynamic Cloud Security portfolio, see this web page:

http://www.ibm.com/security/cloud/

Open cloud and OpenStack-specific security

This section highlights some of the most recent advancements, vendor collaborations, and enhancements for open cloud and OpenStack security.

- Intel, Mirantis, and IBM SoftLayer enhancements for OpenStack security

  Together, Intel, Mirantis, and IBM SoftLayer have integrated a solution based on Intel Trusted Execution Technology, attestation and automation to enhance hybrid cloud security. Intel Trusted Execution Technology is an extension to the Intel Xeon processor and is designed to harden platforms against attacks to the hypervisor and BIOS, malicious rootkit installations, and other firmware and software attacks. Intel TXT establishes a root of trust, a hardware-based security foundation that is used to verify the integrity of the other system components, such as the hypervisor.

Intel TXT helps protect virtualized server environments through isolation and attestation. In Figure 3-6, at start, Intel TXT measures the hash value of the hypervisor and compares it with a known good value. If the measurements do not match, indicating that the hypervisor may have been compromised, Intel TXT alerts IT to the situation. IT staff can then define policies to respond to these alerts, such as blocking the launch. This enables the cloud service provider (or the private cloud) to establish pools of compute resources with proven server infrastructure integrity on which tenant virtual machines run.

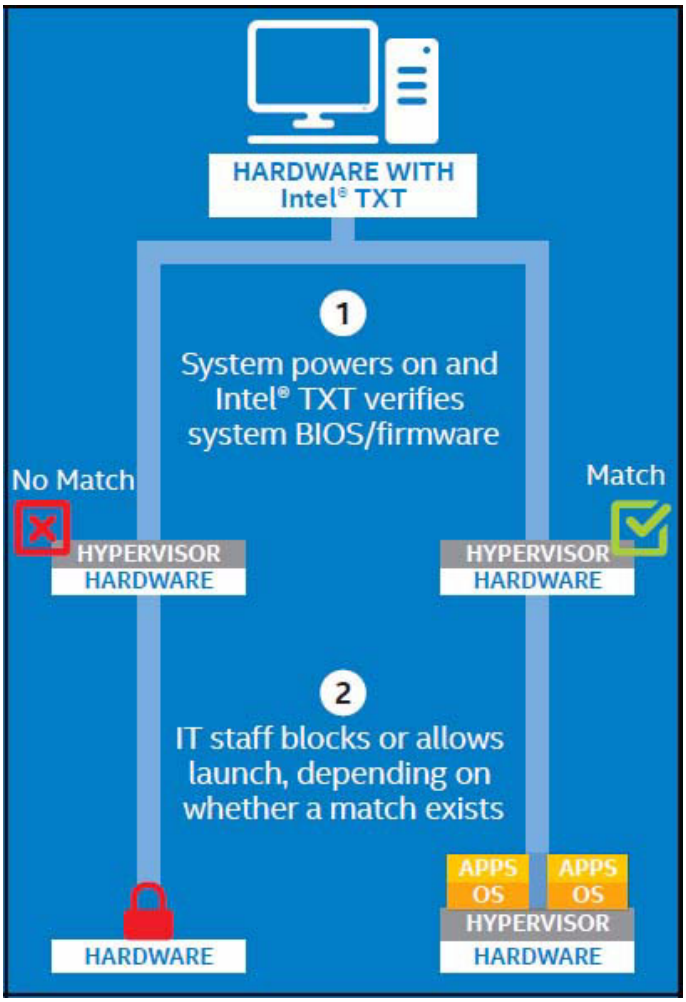


Figure 3-6 Intel TXT Technology

For more information, see the Intel Trusted Execution Technology Hybrid Cloud Security solution brief:

Improving OpenStack Hybrid Cloud Security

http://www.slideshare.net/DavidSlovensky1/improving-openstackhybridcloudsecuritybriefl

For detailed information about OpenStack security, read the *OpenStack Security Guide*:

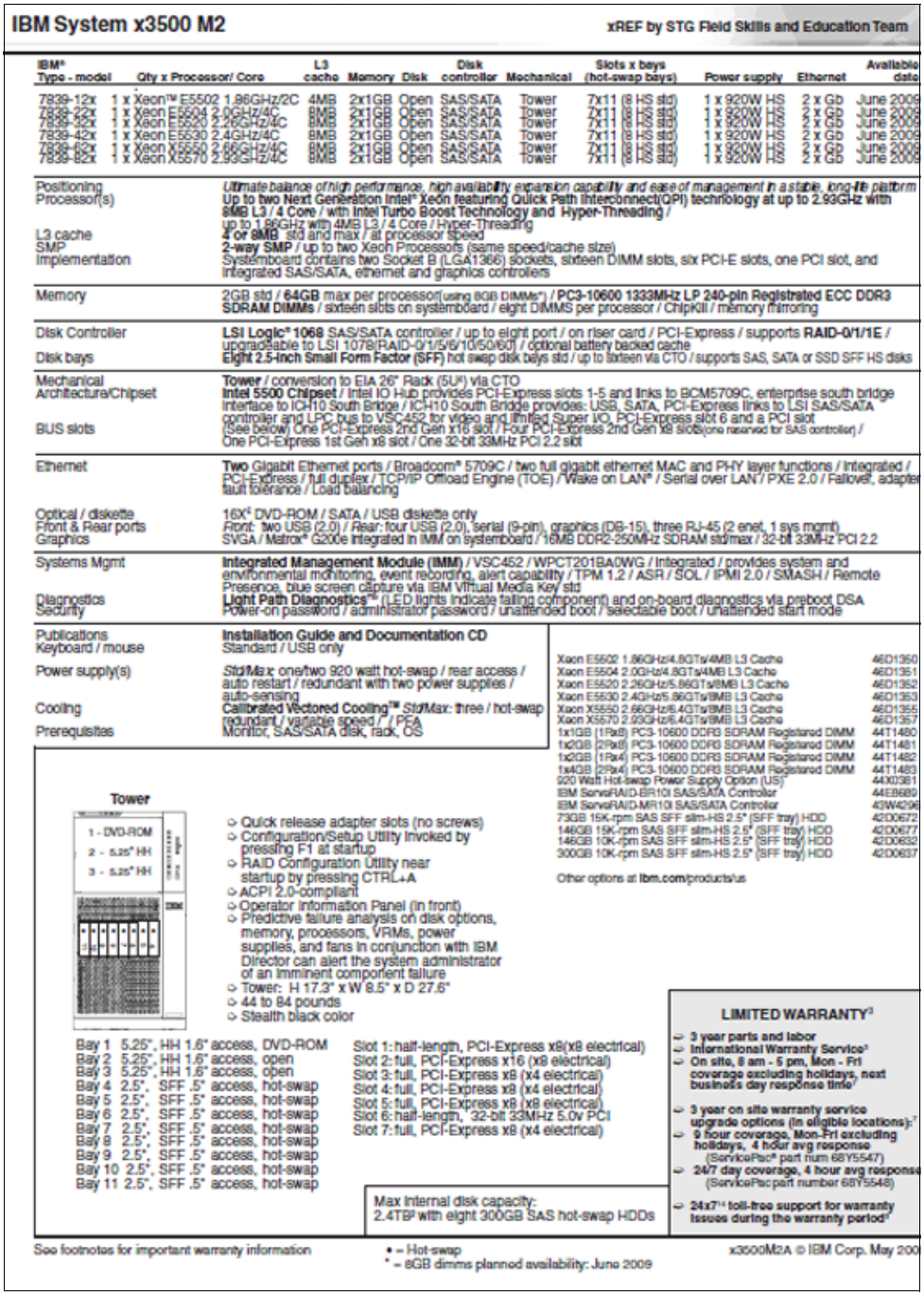
http://docs.openstack.org/sec/

3.3 Servers: x86, Power Systems, System z

For the SDE solution in this IBM Redbooks publication (and based on lab hardware availability), we have selected IBM x86 M series servers. The section that follows provides the reference detail for these x3500 and x3950 M2 servers.

3.3.1 x3500 7839 series

Figure 3-7 provides the reference detail for the x3500 server.



IBM System x3500 M2

xREF by STG Field Skills and Education Team

IBM® Type - model	Qty x Processor/ Core	L3 cache	Memory	Disk	Disk controller	Mechanical	Slots x bays (hot-swap bays)	Power supply	Ethernet	Available date
7839-12x	1 x Xeon™ E5502 1.86GHz/2C	4MB	2x1GB	Open	SAS/SATA	Tower	7x11 (8 HS std)	1 x 920W HS	2 x Gb	June 2009
7839-22x	1 x Xeon E5504 2.0GHz/4C	4MB	2x1GB	Open	SAS/SATA	Tower	7x11 (8 HS std)	1 x 920W HS	2 x Gb	June 2009
7839-32x	1 x Xeon E5520 2.26GHz/4C	8MB	2x1GB	Open	SAS/SATA	Tower	7x11 (8 HS std)	1 x 920W HS	2 x Gb	June 2009
7839-42x	1 x Xeon E5530 2.4GHz/4C	8MB	2x1GB	Open	SAS/SATA	Tower	7x11 (8 HS std)	1 x 920W HS	2 x Gb	June 2009
7839-62x	1 x Xeon X5550 2.66GHz/4C	8MB	2x1GB	Open	SAS/SATA	Tower	7x11 (8 HS std)	1 x 920W HS	2 x Gb	June 2009
7839-82x	1 x Xeon X5570 2.93GHz/4C	8MB	2x1GB	Open	SAS/SATA	Tower	7x11 (8 HS std)	1 x 920W HS	2 x Gb	June 2009

Positioning	*Ultimate balance of high performance, high availability expansion capability and ease of management in a stable, long-life platform*
Processor(s)	**Up to two Next Generation Intel® Xeon featuring Quick Path Interconnect(QPI) technology at up to 2.93GHz with 8MB L3 / 4 Core / with Intel Turbo Boost Technology and Hyper-Threading /** up to 1.86GHz with 4MB L3 / 4 Core / Hyper-Threading
L3 cache	**4 or 8MB** std and max / at processor speed
SMP	**2-way SMP** / up to two Xeon Processors (same speed/cache size)
Implementation	Systemboard contains two Socket B (LGA1366) sockets, sixteen DIMM slots, six PCI-E slots, one PCI slot, and integrated SAS/SATA, ethernet and graphics controllers
Memory	2GB std / **64GB** max per processor(using 8GB DIMMs*) / **PC3-10600 1333MHz LP 240-pin Registrated ECC DDR3 SDRAM DIMMs** / sixteen slots on systemboard / eight DIMMS per processor / ChipKill / memory mirroring
Disk Controller	**LSI Logic® 1068** SAS/SATA controller / up to eight port / on riser card / PCI-Express / supports **RAID-0/1/1E** / upgradeable to LSI 1078(RAID-0/1/5/6/10/50/60) / optional battery backed cache
Disk bays	**Eight 2.5-inch Small Form Factor (SFF)** hot swap disk bays std / up to sixteen via CTO / supports SAS, SATA or SSD SFF HS disks
Mechanical	**Tower** / conversion to EIA 26" Rack (5U²) via CTO
Architecture/Chipset	**Intel 5500 Chipset** / Intel IO Hub provides PCI-Express slots 1-5 and links to BCM5709C, enterprise south bridge interface to ICH10 South Bridge / ICH10 South Bridge provides: USB, SATA, PCI-Express links to LSI SAS/SATA controller and LPC bus to VSC452 for video and Winbond Super I/O. PCI-Express slot 6 and a PCI slot
BUS slots	(See below) One PCI-Express 2nd Gen x16 slot / Four PCI-Express 2nd Gen x8 slots(one reserved for SAS controller) / One PCI-Express 1st Gen x8 slot / One 32-bit 33MHz PCI 2.2 slot
Ethernet	**Two** Gigabit Ethernet ports / Broadcom® 5709C / two full gigabit ethernet MAC and PHY layer functions / Integrated / PCI-Express / full duplex / TCP/IP Offload Engine (TOE) / Wake on LAN® / Serial over LAN / PXE 2.0 / Failover, adapter fault tolerance / Load balancing
Optical / diskette	16X³ DVD-ROM / SATA / USB diskette only
Front & Rear ports	*Front:* two USB (2.0) / *Rear:* four USB (2.0), serial (9-pin), graphics (DB-15), three RJ-45 (2 enet, 1 sys mgmt)
Graphics	SVGA / Matrox® G200e integrated in IMM on systemboard / 16MB DDR2-250MHz SDRAM std/max / 32-bit 33MHz PCI 2.2
Systems Mgmt	**Integrated Management Module (IMM)** / VSC452 / WPCT201BA0WG / Integrated / provides system and environmental monitoring, event recording, alert capability / TPM 1.2 / ASR / SOL / IPMI 2.0 / SMASH / Remote Presence, blue screen capture via IBM Virtual Media Key std
Diagnostics	**Light Path Diagnostics™** (LED lights indicate failing component) and on-board diagnostics via preboot DSA
Security	Power-on password / administrator password / unattended boot / selectable boot / unattended start mode
Publications	**Installation Guide and Documentation CD**
Keyboard / mouse	Standard / USB only
Power supply(s)	*Std/Max:* one/two 920 watt hot-swap / rear access / auto restart / redundant with two power supplies / auto-sensing
Cooling	**Calibrated Vectored Cooling™** *Std/Max:* three / hot-swap redundant / variable speed / / PFA
Prerequisites	Monitor, SAS/SATA disk, rack, OS

Xeon E5502 1.86GHz/4.8GTs/4MB L3 Cache	46D1350
Xeon E5504 2.0GHz/4.8GTs/4MB L3 Cache	46D1351
Xeon E5520 2.26GHz/5.86GTs/8MB L3 Cache	46D1352
Xeon E5530 2.4GHz/5.86GTs/8MB L3 Cache	46D1353
Xeon X5550 2.66GHz/6.4GTs/8MB L3 Cache	46D1355
Xeon X5570 2.93GHz/6.4GTs/8MB L3 Cache	46D1357
1x1GB (1Rx8) PC3-10600 DDR3 SDRAM Registered DIMM	44T1480
1x2GB (2Rx8) PC3-10600 DDR3 SDRAM Registered DIMM	44T1481
1x2GB (1Rx4) PC3-10600 DDR3 SDRAM Registered DIMM	44T1482
1x4GB (2Rx4) PC3-10600 DDR3 SDRAM Registered DIMM	44T1483
920 Watt Hot-swap Power Supply Option (US)	44X0381
IBM ServeRAID-BR10i SAS/SATA Controller	44E8689
IBM ServeRAID-MR10i SAS/SATA Controller	43W4296
73GB 15K-rpm SAS SFF slim-HS 2.5" (SFF tray) HDD	42D0672
146GB 15K-rpm SAS SFF slim-HS 2.5" (SFF tray) HDD	42D0677
146GB 10K-rpm SAS SFF slim-HS 2.5" (SFF tray) HDD	42D0632
300GB 10K-rpm SAS SFF slim-HS 2.5" (SFF tray) HDD	42D0637

Other options at ibm.com/products/us

Tower

- Quick release adapter slots (no screws)
- Configuration/Setup Utility invoked by pressing F1 at startup
- RAID Configuration Utility near startup by pressing CTRL+A
- ACPI 2.0-compliant
- Operator information Panel (in front)
- Predictive failure analysis on disk options, memory, processors, VRMs, power supplies, and fans in conjunction with IBM Director can alert the system administrator of an imminent component failure
- Tower: H 17.3" x W 8.5" x D 27.6"
- 44 to 84 pounds
- Stealth black color

Bay 1	5.25", HH 1.6" access, DVD-ROM
Bay 2	5.25", HH 1.6" access, open
Bay 3	5.25", HH 1.6" access, open
Bay 4	2.5", SFF .5" access, hot-swap
Bay 5	2.5", SFF .5" access, hot-swap
Bay 6	2.5", SFF .5" access, hot-swap
Bay 7	2.5", SFF .5" access, hot-swap
Bay 8	2.5", SFF .5" access, hot-swap
Bay 9	2.5", SFF .5" access, hot-swap
Bay 10	2.5", SFF .5" access, hot-swap
Bay 11	2.5", SFF .5" access, hot-swap

Slot 1:	half-length, PCI-Express x8(x8 electrical)
Slot 2:	full, PCI-Express x16 (x8 electrical)
Slot 3:	full, PCI-Express x8 (x4 electrical)
Slot 4:	full, PCI-Express x8 (x4 electrical)
Slot 5:	full, PCI-Express x8 (x8 electrical)
Slot 6:	half-length, 32-bit 33MHz 5.0v PCI
Slot 7:	full, PCI-Express x8 (x4 electrical)

Max internal disk capacity: 2.4TB⁵ with eight 300GB SAS hot-swap HDDs

LIMITED WARRANTY⁷

- **3 year parts and labor**
- **International Warranty Service⁸**
- **On site, 8 am - 5 pm, Mon - Fri coverage excluding holidays, next business day response time⁹**
- **3 year on site warranty service upgrade options (in eligible locations):¹⁰**
- **9 hour coverage, Mon-Fri excluding holidays., 4 hour avg response** (ServicePac® part num 68Y5547)
- **24/7 day coverage, 4 hour avg response** (ServicePac part number 68Y5548)
- **24x7¹⁴ toll-free support for warranty issues during the warranty period¹⁰**

See footnotes for important warranty information

• = Hot-swap
* = 8GB dimms planned availability: June 2009

x3500M2A © IBM Corp. May 200

Figure 3-7 x3500 M2 reference detail

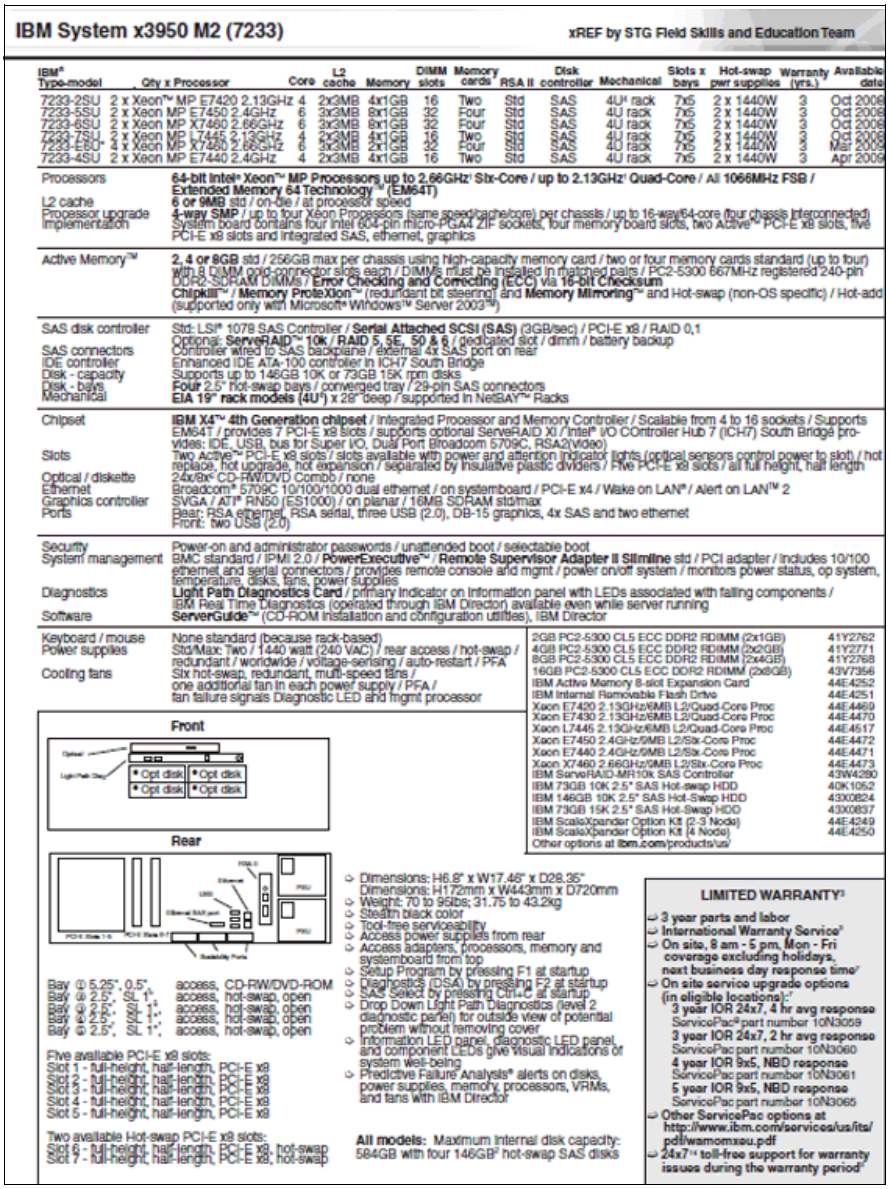
Figure 3-8 is a picture of the x3500 M2 server.



Figure 3-8 x3500 M2 server

3.3.2 x3950 Series 7233 Series

Figure 3-9 provides the reference detail for the x3950 M2 server.



IBM System x3950 M2 (7233)

xREF by STG Field Skills and Education Team

IBM® Type-model	Qty x Processor	Core	L2 cache	Memory	DIMM slots	Memory cards	RSA II	Disk controller	Mechanical	Slots x bays	Hot-swap pwr supplies	Warranty (yrs.)	Available date
7233-2SU	2 x Xeon™ MP E7420 2.13GHz	4	2x3MB	4x1GB	16	Two	Std	SAS	4U[4] rack	7x5	2 x 1440W	3	Oct 2008
7233-5SU	2 x Xeon MP E7450 2.4GHz	6	3x3MB	8x1GB	32	Four	Std	SAS	4U rack	7x5	2 x 1440W	3	Oct 2008
7233-6SU	2 x Xeon MP X7460 2.66GHz	6	3x3MB	8x1GB	32	Four	Std	SAS	4U rack	7x5	2 x 1440W	3	Oct 2008
7233-7SU	2 x Xeon MP L7445 2.13GHz	4	2x3MB	4x1GB	16	Two	Std	SAS	4U rack	7x5	2 x 1440W	3	Oct 2008
7233-8SU	4 x Xeon MP X7460 2.66GHz	6	3x3MB	8x1GB	32	Four	Std	SAS	4U rack	7x5	2 x 1440W	3	Mar 2009
7233-4SU	2 x Xeon MP E7440 2.4GHz	4	2x3MB	4x1GB	16	Two	Std	SAS	4U rack	7x5	2 x 1440W	3	Apr 2009

Processors	**64-bit Intel® Xeon™ MP Processors up to 2.66GHz[1] Six-Core / up to 2.13GHz[1] Quad-Core** / All **1066MHz FSB** / **Extended Memory 64 Technology™ (EM64T)**
L2 cache	**6 or 9MB** std / on-die / at processor speed
Processor upgrade	**4-way SMP** / up to four Xeon Processors (same speed/cache/core) per chassis / up to 16-way/64-core (four chassis interconnected)
Implementation	System board contains four Intel 604-pin micro-PGA4 ZIF sockets, four memory board slots, two Active™ PCI-E x8 slots, five PCI-E x8 slots and integrated SAS, ethernet, graphics
Active Memory™	**2, 4 or 8GB** std / 256GB max per chassis using high-capacity memory card / two or four memory cards standard (up to four) with 8 DIMM gold-connector slots each / DIMMs must be installed in matched pairs / PC2-5300 667MHz registered 240-pin DDR2-SDRAM DIMMs / **Error Checking and Correcting (ECC)** via **16-bit Checksum Chipkill™ / Memory ProteXion™** (redundant bit steering) and **Memory Mirroring™** and Hot-swap (non-OS specific) / Hot-add (supported only with Microsoft® Windows™ Server 2003™)
SAS disk controller	Std: LSI® 1078 SAS Controller / **Serial Attached SCSI (SAS)** (3GB/sec) / PCI-E x8 / RAID 0,1 Optional: **ServeRAID™ 10k / RAID 5, 5E, 50 & 6** / dedicated slot / dimm / battery backup
SAS connectors	Controller wired to SAS backplane / external 4x SAS port on rear
IDE controller	Enhanced IDE ATA-100 controller in ICH7 South Bridge
Disk - capacity	Supports up to 146GB 10K or 73GB 15K rpm disks
Disk - bays	**Four** 2.5" hot-swap bays / converged tray / 29-pin SAS connectors
Mechanical	**EIA 19" rack models (4U[4]) x 28"** deep / supported in NetBAY™ Racks
Chipset	**IBM X4™ 4th Generation chipset** / Integrated Processor and Memory Controller / Scalable from 4 to 16 sockets / Supports EM64T / provides 7 PCI-E x8 slots / supports optional ServeRAID XI / Intel® I/O COntroller Hub 7 (ICH7) South Bridge provides: IDE, USB, bus for Super I/O, Dual Port Broadcom 5709C, RSA2(video)
Slots	Two Active™ PCI-E x8 slots / slots available with power and attention indicator lights (optical sensors control power to slot) / hot replace, hot upgrade, hot expansion / separated by insulative plastic dividers / Five PCI-E x8 slots / all full height, half length
Optical / diskette	24x/8x[2] CD-RW/DVD Combo / none
Ethernet	Broadcom® 5709C 10/100/1000 dual ethernet / on systemboard / PCI-E x4 / Wake on LAN® / Alert on LAN™ 2
Graphics controller	SVGA / ATI® RN50 (ES1000) / on planar / 16MB SDRAM std/max
Ports	Rear: RSA ethernet, RSA serial, three USB (2.0), DB-15 graphics, 4x SAS and two ethernet Front: two USB (2.0)
Security	Power-on and administrator passwords / unattended boot / selectable boot
System management	BMC standard / IPMI 2.0 / **PowerExecutive™** / **Remote Supervisor Adapter II Slimline** std / PCI adapter / includes 10/100 ethernet and serial connectors / provides remote console and mgmt / power on/off system / monitors power status, op system, temperature, disks, fans, power supplies
Diagnostics	**Light Path Diagnostics Card** / primary indicator on information panel with LEDs associated with failing components / IBM Real Time Diagnostics (operated through IBM Director) available even while server running
Software	**ServerGuide™** (CD-ROM installation and configuration utilities), IBM Director
Keyboard / mouse	None standard (because rack-based)
Power supplies	Std/Max: Two / 1440 watt (240 VAC) / rear access / hot-swap / redundant / worldwide / voltage-sensing / auto-restart / PFA
Cooling fans	Six hot-swap, redundant, multi-speed fans / one additional fan in each power supply / PFA / fan failure signals Diagnostic LED and mgmt processor

2GB PC2-5300 CL5 ECC DDR2 RDIMM (2x1GB)	41Y2762
4GB PC2-5300 CL5 ECC DDR2 RDIMM (2x2GB)	41Y2771
8GB PC2-5300 CL5 ECC DDR2 RDIMM (2x4GB)	41Y2768
16GB PC2-5300 CL5 ECC DDR2 RDIMM (2x8GB)	43V7356
IBM Active Memory 8-slot Expansion Card	44E4252
IBM Internal Removable Flash Drive	44E4251
Xeon E7420 2.13GHz/6MB L2/Quad-Core Proc	44E4469
Xeon E7430 2.13GHz/6MB L2/Quad-Core Proc	44E4470
Xeon L7445 2.13GHz/6MB L2/Quad-Core Proc	44E4517
Xeon E7450 2.4GHz/9MB L2/Six-Core Proc	44E4472
Xeon E7440 2.4GHz/9MB L2/Six-Core Proc	44E4471
Xeon X7460 2.66GHz/9MB L2/Six-Core Proc	44E4473
IBM ServeRAID-MR10k SAS Controller	43W4280
IBM 73GB 10K 2.5" SAS Hot-swap HDD	40K1052
IBM 146GB 10K 2.5" SAS Hot-Swap HDD	43X0824
IBM 73GB 15K 2.5" SAS Hot-Swap HDD	43X0837
IBM ScaleXpander Option Kit (2-3 Node)	44E4249
IBM ScaleXpander Option Kit (4 Node)	44E4250

Other options at ibm.com/products/us/

Front

Rear

- Dimensions: H6.9" x W17.46" x D28.35" Dimensions: H172mm x W443mm x D720mm
- Weight: 70 to 95lbs; 31.75 to 43.2kg
- Stealth black color
- Tool-free serviceability
- Access power supplies from rear
- Access adapters, processors, memory and systemboard from top
- Setup Program by pressing F1 at startup
- Diagnostics (DSA) by pressing F2 at startup
- SAS Select by pressing Ctrl+C at startup
- Drop Down Light Path Diagnostics (level 2 diagnostic panel) for outside view of potential problem without removing cover
- Information LED panel, diagnostic LED panel, and component LEDs give visual indications of system well-being
- Predictive Failure Analysis® alerts on disks, power supplies, memory, processors, VRMs, and fans with IBM Director

Bay ① 5.25", 0.5", access, CD-RW/DVD-ROM
Bay ② 2.5", SL 1", access, hot-swap, open
Bay ③ 2.5", SL 1", access, hot-swap, open
Bay ④ 2.5", SL 1", access, hot-swap, open
Bay ⑤ 2.5", SL 1", access, hot-swap, open

Five available PCI-E x8 slots:
Slot 1 - full-height, half-length, PCI-E x8
Slot 2 - full-height, half-length, PCI-E x8
Slot 3 - full-height, half-length, PCI-E x8
Slot 4 - full-height, half-length, PCI-E x8
Slot 5 - full-height, half-length, PCI-E x8

Two available Hot-swap PCI-E x8 slots:
Slot 6 - full-height, half-length, PCI-E x8, hot-swap
Slot 7 - full-height, half-length, PCI-E x8, hot-swap

All models: Maximum internal disk capacity: 584GB with four 146GB[2] hot-swap SAS disks

LIMITED WARRANTY[3]

- **3 year parts and labor**
- **International Warranty Service[5]**
- **On site, 8 am - 5 pm, Mon - Fri coverage excluding holidays, next business day response time[7]**
- **On site service upgrade options (in eligible locations):[7]**
  - **3 year IOR 24x7, 4 hr avg response** ServicePac® part number 10N3059
  - **3 year IOR 24x7, 2 hr avg response** ServicePac part number 10N3060
  - **4 year IOR 9x5, NBD response** ServicePac part number 10N3061
  - **5 year IOR 9x5, NBD response** ServicePac part number 10N3065
- **Other ServicePac options at http://www.ibm.com/services/us/its/pdf/wamomxeu.pdf**
- **24x7[14] toll-free support for warranty issues during the warranty period[7]**

Figure 3-9 x3950 M2 reference detail

Figure 3-10 shows a picture of the x3950 M2 server.



Figure 3-10 x3950 M2 server

For this IBM SDE solution, we deployed IBM x86 M series servers, as shown in Figure 3-11.

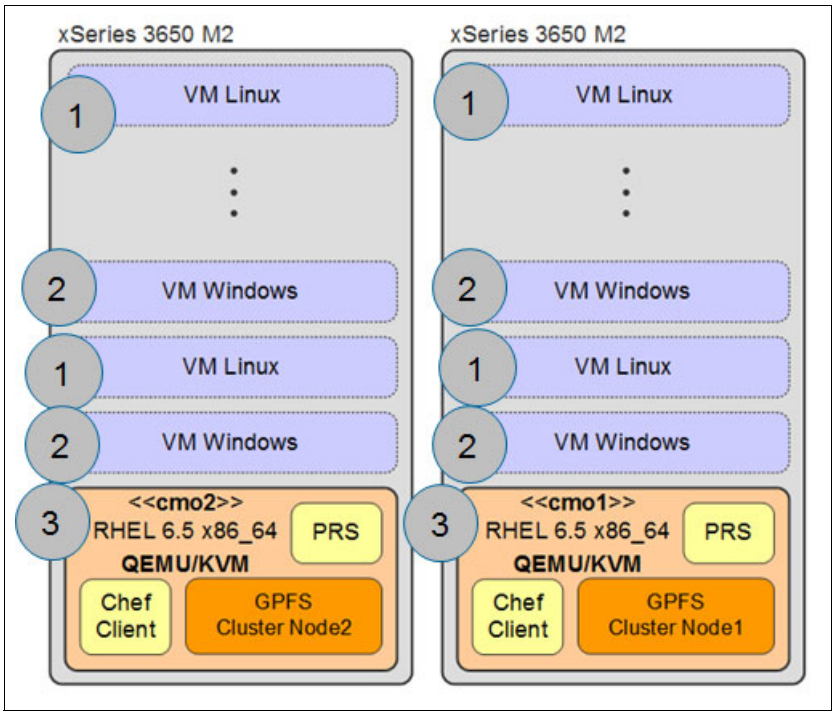


Figure 3-11 x86 SDE reference architecture

Key to the numbers in Figure 3-11 on page 79:

1, 2. Each server has an image for workload placement - for Linux and Windows. In IBM Cloud Manager with OpenStack, you use images to deploy your instances (virtual machines). The images are virtual machines that were previously installed and set up with appropriate operating system, additional software, and specific configurations (based on workload) for your environment. The OpenStack component that manages images is called *Glance*. Please see 4.12, “Manage images in IBM Cloud Manager” on page 412 for information about how to create images for deployment on KVM, PowerKVM and PowerVM Compute nodes in your OpenStack cloud.

3. In our SDE cloud environment, we used 2 x86 M2 servers running Red Hat Enterprise Linux Server release 6.5 and KVM/QEMU version qemu-kvm-0.12.1.2-2.415 for the CMO 1 and CMO 2 servers. The CMO 1, and 2 are the KVM Compute nodes running RHEL 6.5 on these System x3650 M2 Servers. These servers are connected to both public and private networks and to the SVC storage. A GPFS Cluster was created in these 2 nodes using SVC disks. They are named cmo1.itso.ibm.com and cmo2.itso.ibm.com and are used as Compute nodes for deploying Windows and Linux instances.

GPFS is a clustered file system that allows concurrent access to the files across several nodes. It can be used as an Spectrum Scale or software-defined storage in the cloud environment. Volumes for KVM Compute nodes instances can be file backed devices and the files can be placed in the GPFS file system. With the files available in the GPFS file system and the file system mounted on all KVM nodes, the instance can be migrated dynamically from one node to the other using solutions such as Platform Resource Scheduler (PRS).

In our cloud, one GPFS Cluster was created on KVM Compute nodes cmo1.itso.ibm.com and cmo2.itso.ibm.com using SVC disks.

Platform Resource Scheduler (PRS): As a resource-aware and workload-smart solution, IBM Platform Resource Scheduler is a key element of the IBM SDE portfolio, enabling dynamic allocation of infrastructure using intelligent resource scheduling. This new Platform Resource Scheduler is an extension to IBM Cloud Orchestrator that uses technology of IBM Platform Computing to simplify the management of cloud resources. By tying IBM Platform Resource Scheduler to Cloud Orchestrator for OpenStack environments, the solution can dynamically allocate resources on demand across thousands of virtual and physical servers. The Platform Resource Scheduler within an OpenStack deployment can choose the most appropriate server in which to place a new virtual machine based on policies crafted by the administrator. Criteria can include the number of times each machine is already being used in terms of its CPU use, memory use and other factors.

The IBM Cloud Manager can be installed on the following platforms:

- Intel x86_64 (running Linux)
- IBM POWER6®, IBM POWER7® or 7+ or POWER8 (running PowerLinux)

To complete this installation, there are minimum requirements that need to be met:

- 4 CPUs
- At least 4 GB of free space for `/opt/ibm/cmwo` (Cloud Manager directory. This doesn’t need to be a separate file system, but the parent file system `/opt` or */opt/ibm* must provide this free space.)
- At least 4.5 GB of temporary free space during the product installation
- Additional at least 5 GB of free space in /opt and 5 GB of free space in `/var`
- At least 4GB of RAM

The two x86 servers have the following configuration:

- Server 1 (CMO 1):
  - CPU: 16 CPUs Intel Xeon CPU E5520 at 2.27 GHz
  - Memory: 130 GB
  - Disk: 1 TB
- Server 2 (CMO 2):
  - CPU: 12 CPUs Intel Xeon CPU X7460 at 2.66 GHz
  - Memory: 130 GB
  - Disk: 300 GB

Table 3-2 provides a summary of the sizing for the servers, for each VM, and including the allocated sizing for CMO 1 and 2.

Table 3-2 Servers sizing summary

Server or image	CMO 1	CMO 2	Linux VM 1, 2	Windows 1, 2
Server 1/CMO 1	► CPU: .8 CPU ► Memory: 27 GB ► Disk: 81 GB	N/A	See following table or figure	Shown for illustrative purposes only. in our lab environment, we deployed only Linux instances. These are here to represent the “art of the possible,” and the flexibility of SDE with OpenStack.
Server 2/CMo 2	N/A	► .36 CPU ► 27 GB ► 33 GB	See following table or figure	Shown for illustrative purposes only; in our lab environment we only deployed Linux instances. These are here to represent the “art of the possible”, and the flexibility of SDE with OpenStack.

The VMs sizes are based on the “flavor” we choose when we deploy the instances. Figure 3-12 on page 82 shows an example in the Redbooks lab SDE environment, and the currently available flavors (but more can always be created) are illustrated.

```
[root@cmocontroller ~]# nova flavor-list
+--------------------------------------+-------------------------+-----------+------+-----------+------+-------+-------------+-----------+
| ID                                   | Name                    | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+--------------------------------------+-------------------------+-----------+------+-----------+------+-------+-------------+-----------+
| 1                                    | m1.tiny                 | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 1f6596de-2708-42b2-81fc-f324fbb3cde6 | PVC-m1.small            | 2048      | 50   | 0         |      | 1     | 1.0         | True      |
| 3                                    | m1.medium               | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4                                    | m1.large                | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 4610c174-f3b0-4175-9609-eb364b1e651d | big_flavor              | 10240     | 50   | 0         |      | 4     | 1.0         | True      |
| 5                                    | m1.xlarge               | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
| 714ebbee-c0ed-4a55-9bcd-8a2a7bea944c | PVC-m1.large            | 8192      | 50   | 0         |      | 4     | 1.0         | True      |
| PVC-6940e12a-ebf4-4e99-b66e-d7c45ac3a986 | PVC-big_powervc_flavor | 10240     | 50   | 0         |      | 4     | 1.0         | True      |
| d9599491-cc9c-435f-a363-852f1c1e6c94 | PVC-m1.medium           | 4096      | 50   | 0         |      | 2     | 1.0         | True      |
| e21d6835-cd1d-4fc1-8922-b441be11bf39 | PVC-m1.tiny             | 1024      | 50   | 0         |      | 1     | 1.0         | True      |
| e7c1f9e3-a725-460a-9fe5-d65b084030f5 | PVC-m1.xlarge           | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
| f252a58b-f05c-4c9f-b61a-0c74c2b770b2 | m1.small                | 2048      | 20   | 0         |      | 1     | 1.0         | False     |
+--------------------------------------+-------------------------+-----------+------+-----------+------+-------+-------------+-----------+

For the PowerVM (PowerVC) only the Flavors starting with PVC-* are valid.
```

Figure 3-12 SDE image (Glance) solution instance "flavor" listing

For more information about the configuration, see Chapter 4, "Cloud provider installation guidelines and implementation scenarios" on page 195.

IBM recognizes the importance of the x86 architecture for our client workloads. The following documents contain x86 reference architectures:

- *IBM Platform Computing Solutions Reference Architectures and Best Practices*, SG24-8169

  http://www.redbooks.ibm.com/abstracts/sg248169.html

- *IBM System x Reference Architecture for Hadoop: IBM InfoSphere BigInsights Reference Architecture,* REDP-5009
  (Also described in Chapter 2, Section 3.6.2, "Big data analytics workloads" on page 184)

  http://www.redbooks.ibm.com/abstracts/redp5009.html

- Solution brief: IBM System x reference architecture solutions for big data
  (Also described in Chapter 2, Section 3.6.2, "Big data analytics workloads" on page 184)

  http://ibm.co/1zIAaAi

- Solution brief: IBM System x Reference Architecture and Sizing Guide for Hadoop: MapR

  http://ibm.co/1DKpTHE

- Solution brief: IBM System x Solutions for Cloud Computing and Analytics

  http://ibm.co/1yPITpr

Additional information about these topics is available on the "System x literature" web page, which includes brochures, data sheets, solution briefs and reference architectures, white papers, case studies, and product documentation:

http://ibm.co/1IwBuLk

IBM Power servers

IBM Power servers with Linux are the perfect complement to the SDE solution, especially for big data and analytics workloads. Supported on every Power Systems server IBM makes, a Linux on Power Systems is the only infrastructure that offers both scale- out and scale -up choices that align to your needs. Optimize your workloads for emerging business challenges with industry standard Red Hat, SUSE and Ubuntu Linux distributions.

Open innovation is the future of business, as information and data (structured and unstructured) is more widely available than ever. Collaboration, co-creation, and knowledge sharing are quickening the pace of invention. Linux on Power Systems offers competitive advantages to organizations - open technology, performance, portability and scalability.

So, why Linux on Power, when Linux on x86 has always been the de facto standard, for many clients?

- Uses open technology:
  - Benefit from Power Systems open technology and an open ecosystem delivering community created innovation. IBM has partnered with Google, Ubuntu, Hitachi, Mellanox, Micron, NVIDIA, POWERCORE, Samsung and Tyan other companies to further develop and assist in driving the IBM Power Architecture® in the OpenPOWER Foundation.
  - The OpenPOWER Foundation was founded in 2013 as an open technical membership organization that will enable data centers to rethink their approach to technology. Member companies may use POWER for custom open servers and components for Linux based cloud data centers as well as optimizing Linux software on POWER.
  - TYAN Corporation has just released the first customer reference system from this OpenPOWER collaboration. The TYAN GN70-BP010 is a customer reference system which enables users to deploy software based on the OpenPOWER architecture tailored to their individual requirements. It provides another opportunity for users to run their applications in a way of cost effective and flexible way. It is an innovative and collaborative hardware solution for IT experts who are looking for a more open, flexible, customized, and intelligent IT deployment. The TYAN customer reference system product features can be seen in Figure 3-13.

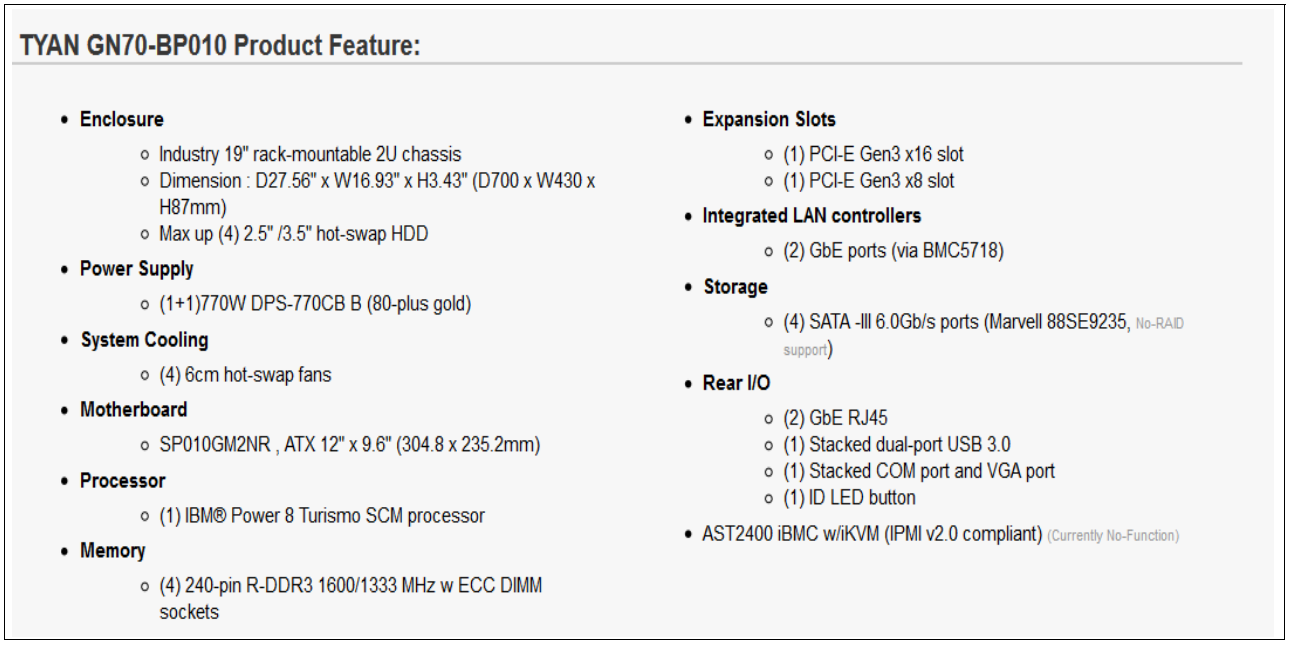


Figure 3-13 TYAN POWER8 customer reference system features and specifications

For more information about this system, see the OpenPOWER Customer Reference System page on the Tyan website:

http://www.tyan.com/campaign/openpower/

Note: For more information, see the OpenPOWER Foundation website:

http://openpowerfoundation.org/

Also see their OpenPOWER Introduction video:

http://openpower.kcdlhosting.com/video/video-openpower/

- Redefine Linux performance:
  - Power Systems run industry-standard Linux from Red Hat, SUSE or Canonical. Linux uses the advanced hardware and software capabilities of POWER8 technology, which provides economic advantages that scale as your business grows.
- Port applications with ease:
  - Clients can confidently run highly scalable, highly reliable, and highly flexible Linux environments on POWER processor-based servers. The benefits of Power Systems are realized whether a client is deploying new applications or wants to improve the performance of existing applications. With POWER8, moving Linux applications to Power has never been easier.
- Optimize emerging workloads:
  - Watson was built on Linux on Power Systems for the big data and analytics advantages POWER provides. IBM is investing heavily in building solution catalogs for emerging areas where POWER offers unique advantages to clients.
- Consolidate to rein in sprawling server farms:
  - Linux-only Power Systems combined with PowerKVM or PowerVM offerings help you to consolidate workloads from hundreds of servers to a few servers.
- Run Linux with Java, IBM middleware and open source components:
  - Power Systems offer double the performance of competitive platforms for Java-based solutions.
- The Power to choose:
  - 1U, single-core to 256-core servers, Power Systems offering choice and are ready for your business-critical Linux applications.
- Industry proven and acclaimed:
  - Supporting embedded systems to large supercomputers, POWER technology is pervasive today with a strong roadmap for the future.
- Reliability, Availability, Serviceability (RAS) that grows with you:
  - Gain reliability, availability and serviceability uncommon in severs running Linux OS. Toolkits and extensions are available to enhance your choice of Linux operating systems to support the RAS characteristics of POWER processor-based servers at a higher level.
- A range of computational power-on which to standardize:
  - Supports processing loads from deep computing to high transaction-oriented commercial systems. By offering multiple processors rates and SMP systems up to 256 cores and memory up to 2 terabytes, Power provides the architecture you need.
- End-to-end support:
  - IBM offers extensive assistance in the form of education, migration tools, services, and support to enable a low risk implementation or migration to Linux solutions.

For the SDE solution in this book, we have also selected IBM Power series servers to illustrate and demonstrate that the SDE solution works with, and across a heterogeneous infrastructure environment. The following provides the reference feature detail for the POWER7+ servers.

IBM Power 740 Express system

The following summary describes the standard features:

- Tower or rack-mount (4U) chassis
- Processors:
  - 4-core or 8-core configuration with one or two 3.3 GHz or 3.7 GHz 4-core processor
  - modules
  - 6-core or 12-core configuration with one or two 3.7 GHz 6-core processor modules
  - 8 or 16-core configuration with two 3.55 GHz 8-core processor modules
- Up to 256 GB of 1066 MHz DDR3 ECC memory
- Choice of two disk/media backplanes:
  - Six x 2.5-inch HDD, SSD, or media backplane, with one tape drive bay and one DVD bay
  - Eight x 2.5-inch HDD, SSD, or media backplane with one tape drive bay, one DVD bay
  - One external SAS port
- Choice of three Integrated Virtual Ethernet daughter cards:
  - Quad-port 1 Gb Copper IVE
  - Dual-port 10 Gb SFP+ TwinAx Cu IVE
  - Dual-port 10 Gb SFP+ Fiber IVE
- A maximum of eight PCIe slots:
  - Four PCIe x8 full height short card slots
  - Optional four PCIe x8 low profile short card slots
- Two GX++ slots
- Integrated:
  - Service processor
  - IBM EnergyScale™ technology
  - Hot-swap and redundant cooling
  - Three USB ports and two system ports
  - Two HMC ports and two SPCN ports
- Redundant, 1725 Watt ac hot-swap power supplies

Figure 3-14 on page 86 shows a picture of the p740+ (8205-E6C) server.



Figure 3-14 Power 740+ server

For the SDE solution, the Power 740+ (8205-E6C) server has the following configuration:

- 16-core configuration with two 3.55 GHz 8-core processor modules;
- 3 Power AIX VM instances (please see Figure 3-13 on page 83 for instance and flavor listings);
- 3 PowerLinux VM instances (please see Figure 3-13 on page 83 for instance and flavor listings);

For more information about the Power 740 server, see this Redbooks publication:

IBM Power 720 and 740 Technical Overview and Introduction, REDP-4984

http://www.redbooks.ibm.com/abstracts/redp4984.html

IBM Power 750 Express system

The following is a summary of standard features:

- Rack-mount (4U) configuration
- Processors:
  - 6-, 12-, 18-, and 24-core design with one, two, three, or four 3.3 GHz processor cards
  - 8-, 16-, 24-, and 32-core design with one, two, three, or four 3.0 GHz or 3.3 GHz
- Processor cards:
  - 8-, 16-, 24-, and 32-core design with one, two, three, or four 3.55 GHz processor cards
- Up to 512 GB of 1066 MHz ECC (error checking and correcting) memory, expandable to 128 GB per processor card
- 8 x 2.5-inch DASD, SSD, Media backplane with an external SAS port
  - 1 to 8 SFF (Small Form Factor) DASD or solid-state drives (mixing allowed)
- Choice of two Integrated Virtual Ethernet daughter cards:
  - Quad-port 1 Gb IVE
  - Dual-port 10 Gb IVE
- Two media bays:
  - One slim bay for a DVD-RAM (required)
  - One half-height bay for an optional tape drive or removable disk

- A maximum of five hot-swap slots:
  - Two PCIe x8 slots, short card length (slots 1 and 2)
  - One PCIe x8 slot, full card length (slot 3)
  - Two PCI-X DDR slots, full card length (slots 4 and 5)
  - One GX+ slot (shares same space as PCIe x8 slot 2)
  - One GX++ slot (shares same space as PCIe x8 slot 1)
- Integrated:
  - Service processor
  - Quad-port 10/100/1000 Mb Ethernet
  - EnergyScale technology
  - Hot-swap and redundant cooling
  - Three USB ports; two system ports
  - Two HMC ports; two SPCN ports
- Two Power Supplies, 1725 Watt AC, Hot-swap

Figure 3-15 shows the p750+ (8233-E8B) server.



Figure 3-15 Power 750+ server

For our SDE solution, the Power 750+ (8233-E8B) server has the following configuration:

- 16-core configuration with two 3.0 GHz 8-core processor modules
- cmocontroller (total / allocated):
  - CPU: 16 CPUs (5% used)
  - Memory: 32 GB (29 GB used)
  - Disk: 100 GB (52 G used)
- cmodeploy (total / allocated)
  - CPU: 16 CPUs (1% used)
  - Memory: 8 GB (4 GB Used)
  - Disk: 44 G (24 G used)
- powervc (total / allocated)
  - CPU: 20 CPUs (10% used)
  - Memory: 16 GB (11GB used)
  - Disk: 44 G (14 G used)

For more information about the Power 750 server, see the Redbooks publication titled *IBM Power 750 and 755 (8233-E8B, 8236-E8C) Technical Overview and Introduction*, REDP-4638:

http://www.redbooks.ibm.com/abstracts/redp4638.html

IBM Power Systems S812L and S822L

The IBM Power System S812L (8247-21L) and the IBM Power System S822L (8247-22L) servers use the latest POWER8 processor technology that is designed to deliver unprecedented performance, scalability, reliability, and manageability for demanding commercial workloads.

These Power Systems servers are optimized for Linux to help deliver new solutions and services faster, with higher quality. The server is also optimized for big data and analytics and to provide the ideal foundation for scale-out data and cloud environments in a compact 2U package. You can gain faster insights with the IBM POWER8 processor and smart acceleration, which is enabled by Coherent Accelerator processor Interface (CAPI) technologies.

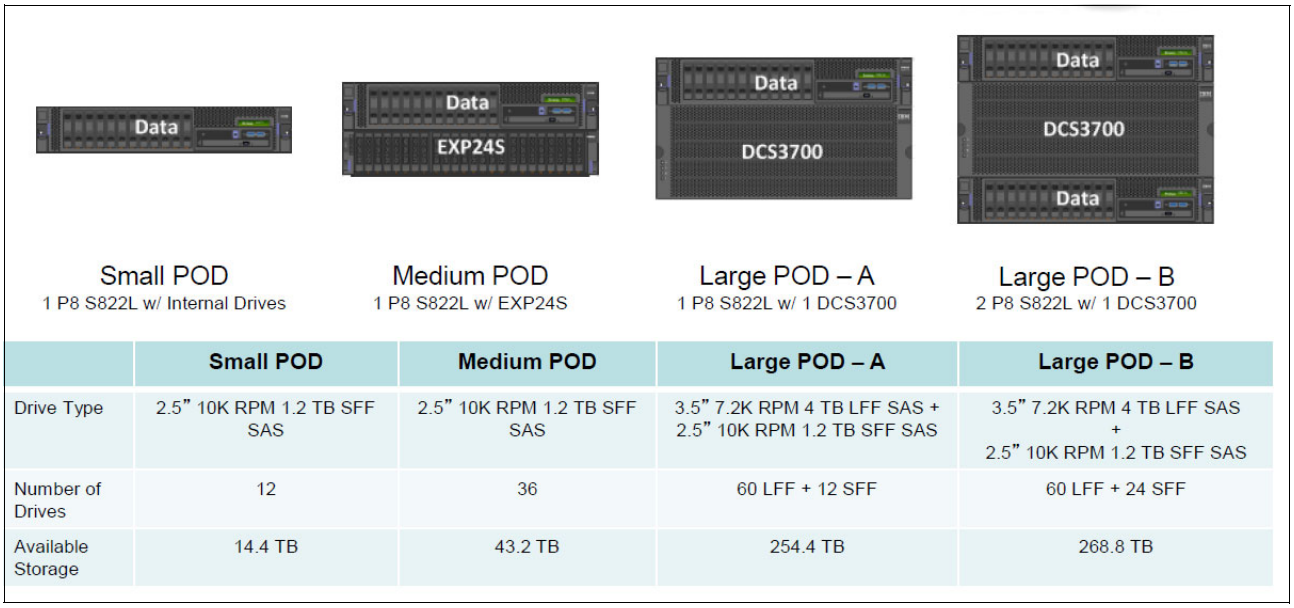
For customers that are committed to Linux and open source applications and infrastructures, the Power S812L and Power S822L servers provide the ideal foundation for private and public cloud infrastructure. Both servers, based on POWER8 technology, deliver superior throughput compared to x86-based offerings for comparable workloads, and provide superior economics for scale-out deployments.

For customers looking to deploy advanced analytics, the Power S812L and Power S822L servers can deliver superior response time for sorting and querying unstructured big data sets, and provide a superior number of business reports per hour for typical business analytics when compared to competing solutions built on x86 servers.

Designed to empower the ecosystem of open source development, these new servers support an increased Linux operating system (OS) ecosystem (Red Hat, SUSE, and Ubuntu) and have added support for open source virtualization technology with the announcement of PowerKVM. For Java workloads, the new servers deliver unprecedented performance. You can also choose between industry-recognized enterprise virtualization PowerVM or PowerKVM and reduce energy use by using advanced energy control.

Power big data clusters are built using a simple building block approach to tailor the mix of CPU and storage to application requirements.

Figure 3-16 on page 89 shows the various integrated building blocks. These are great starting points, and can be further optimized and balanced with SDE as your environment grows.



	Small POD	Medium POD	Large POD – A	Large POD – B
Drive Type	2.5" 10K RPM 1.2 TB SFF SAS	2.5" 10K RPM 1.2 TB SFF SAS	3.5" 7.2K RPM 4 TB LFF SAS + 2.5" 10K RPM 1.2 TB SFF SAS	3.5" 7.2K RPM 4 TB LFF SAS + 2.5" 10K RPM 1.2 TB SFF SAS
Number of Drives	12	36	60 LFF + 12 SFF	60 LFF + 24 SFF
Available Storage	14.4 TB	43.2 TB	254.4 TB	268.8 TB

Figure 3-16 POWER8 S822L Building Blocks

The IBM Power S822L is one of the newest servers in the POWER8 lineup. We have selected this model as it is ideally aligned for hosting IBM big data Hadoop and BigInsights software.

This summary describes the standard features of the Power S822L:

- Rack-mount (2U) chassis
- Dual processor module:
  - 10-core 3.42 GHz processor module
  - 2-core 3.02 GHz processor module
- Up to 1024 GB of 1600 MHz DDR3 ECC memory
- A choice of the following storage features:
  - Storage backplane with 12 SFF-3 bays and one DVD bay
  - Optionally, split the twelve SFF-3 bays and add a second integrated SAS controller without cache.
  - Storage backplane 8 with SFF-3 bays, a 1.8-inch SSD Attach, a DVD bay, dual IOA with write cache and Easy Tier functions, and the expansion capability for the EXP24S SFF Gen2-bay drawer
- Hot-swap PCIe Gen 3 slots
- Integrated features:
  - Service processor
  - EnergyScale technology
  - Hot-swap and redundant cooling
  - Four USB 3.0 ports for general use
  - Two USB 2.0 ports for non-general use
  - Two HMC 1GbE RJ45 ports
  - One system port with RJ45 connector
  - Two hot-plug, redundant power supplies

Figure 3-17 shows the front view of an Power S822L server.



Figure 3-17 Power S822L

For this IBM SDE solution, we have deployed POWER7 and POWER8 series servers, as shown in Figure 3-18.

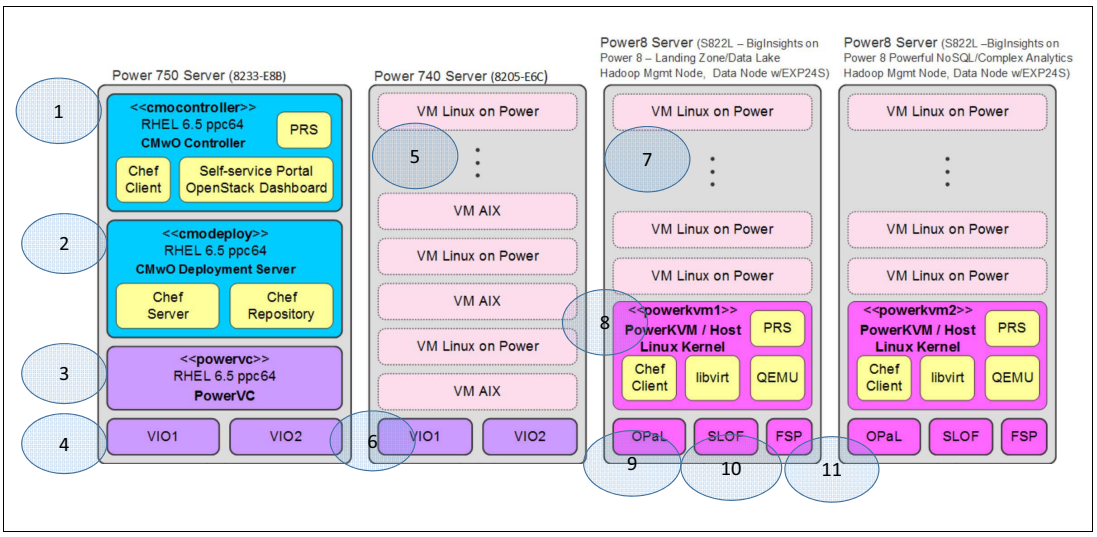


Figure 3-18 Power Systems SDE reference architecture

1. CMO (IBM Cloud Manager 4.1.0.1) Controller, the Controller node running RHEL 6.5 on POWER System LPAR in the IBM POWER 740 + Express server 8205-E6C.The Controller node manages cmo1 and cmo2 as KVM Compute nodes, powerkvm1 and powerkvm2 as PowerKVM Compute nodes and it also has the PowerVC driver enabled to communicate with the PowerVC server. PRS 2.2 is also enabled. It also includes PRS, a Chef Client, and the self-service portal OpenStack dashboard.
2. CMO (IBM Cloud Manager 4.1.0.1) Deploy, the deployment server running RHEL 6.5 on POWER System in IBM Power 740+ Express server 8205-E6C LPAR with IBM Cloud Manager 4.1.0.1. This also includes the Chef server and the Chef Repository.
3. PowerVC- Power Virtualization Center (VC) 1.2.1.1 server running RHEL 6.5 on POWER System LPAR in the p740 server and managing both p740 and p750.
   - IBM PowerVC Virtualization Center is an advanced virtualization management offering, built on OpenStack, that provides simplified virtualization management for IBM AIX, IBM i and Linux virtual machines (VMs) running on IBM Power Systems. PowerVC is designed to improve administrator productivity and simplify the management of VMs and LPARs on Power Systems servers.

PowerVC provides the foundation for Power Systems scalable cloud management, including integration to higher-level cloud managers based on OpenStack technology.

- PowerVC enables clients to capture and manage a library of VM images, enabling IT managers to quickly deploy a VM environment by launching a stored image of that environment rather than having to manually re-create a particular environment. By saving virtual images and centralizing image management, IT managers and administrators can migrate and move virtual images to available systems to expedite deployment.
- The architecture of PowerVC uses the OpenStack platform to provide virtualization management customized for Power Systems. This includes an IBM Power-specific user interface, and IBM-specific OpenStack drivers for Power Systems management interfaces such as the Hardware Management Console (HMC) and integrated virtualization manager (IVM), plus scheduling capabilities. Clients looking for a fully open virtualized infrastructure can use PowerVC to manage their PowerKVM on Power deployments as well.

  For more information about PowerVC Virtualization Center, see these web pages:

  - IBM PowerVC – Virtualization Center

    http://www.ibm.com/systems/power/software/virtualization-management/
  - IBM PowerVC 1.2 Overview

    http://www.youtube.com/watch?v=RFTbC6JW7YE
  - *IBM PowerVC Version 1.2.1 Introduction and Configuration*, SG24-8199

    http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg248199.html
  - *Implementing IBM Power Virtualization Center in Your Data Center*, TIPS1136

    http://www.redbooks.ibm.com/abstracts/tips1136.html
  - IBM developerWorks PowerVC community

    https://www.ibm.com/developerworks/community/groups/community/PowerVC/

4. Virtual I/O Server 2.2.3.2 running in the IBM Power 750 Express server (8233-E8B) to provide network virtualization and NPIV to the LPARs from p750 server
5. Power AIX and Linux VM instances (please see Figure 3-13 on page 83 for instance and flavor listings)
6. Virtual I/O Server 2.2.3.2 running in the IBM Power 740 Express server (8205-E6C) to provide network virtualization and NPIV to the LPARs from p750 server
7. PowerLinux VM instances (please see Figure 3-13 on page 83 for instance and flavor listings)
8. PowerKVM 1 and PowerKVM 2 Compute node running PowerKVM 2.1.0 on the POWER8 (IBM PowerLinux S822L) servers, including:
   - PRS- Platform Resource Scheduler (PRS) prioritizes resources to the most important applications, enables resource consolidation while preserving service level agreements, increases operational efficiency by optimizing compute and memory use, and decreases power use while reducing administrator requirements. An open architecture provides the highest levels of customization to meet your company-specific objectives with heterogeneous resource pools. This runs on both of the POWER8 (IBM PowerLinux S822L) servers.

- Chef client is an agent that runs locally on every node that is under management by Chef. When a chef-client is run, it will perform all of the steps that are required to bring the node into the expected state. This runs on both of the POWER8 (IBM PowerLinux S822L) servers.
- libbvirt is a hypervisor management library. libvirt provides a stable, cross-platform interface for higher-level management tools:
  - Used to manage guests, virtual networks and storage on the KVM host
  - Provides APIs for management
  - The configuration of each guest is stored in an XML file.
  - Allows remote management of guests
  - Encryption, certificates (TLS), authentication (SASL)
  - Communication between libvirt and tools management is done via a daemon called *libvirtd*
  - Check status: `systemctl status libvirtd`
- QEMU (short for “quick emuator”) is a free and open source-hosted hypervisor that performs hardware virtualization (not to be confused with hardware-assisted virtualization).QEMU is a hosted virtual machine monitor: It emulates central processing units through dynamic binary translation and provides a set of device models, enabling it to run a variety of unmodified guest operating systems. It also can be used together with KVM to run virtual machines at near-native speed (requiring hardware virtualization extensions on x86 machines). QEMU can also be used purely for CPU emulation for user-level processes, allowing applications compiled for one architecture to be run on another. This runs on both of the POWER8 (IBM PowerLinux S822L) servers.

9. OPAL is the system firmware in the stack of POWER8 processor-based server, PowerKVM. It consists of three major parts called skiboot, skiroot, and skirun.
   - OPAL boot (skiboot)

     This is a binary component (a Load ID) loaded by the FSP which performs some system initialization and setup (PCIe and so on) and provides runtime services abstraction to a client operating system. Those services range from accessing the real-time clock, rebooting the system, various PCIe PCI Host Bridging (PHB) related functions and nvram. OPAL additionally loads and transfers control to the OPAL loader during boot.
   - OPAL root (skiroot)

     This a Linux-based boot loader that discovers I/O devices and loads a Linux-based hypervisor from a disk or network.
   - OPAL runtime (skirun)

     OPAL runtime is a Linux system (kernel + ramdisk) stored in a separate local ID. This system is used to access I/O devices (such as disks, network, and so on) and locate and load the host Linux system to be run on the system in hypervisor mode. This firmware is part of the POWER8 architecture on both of the POWER8 (IBM PowerLinux S822L) servers.

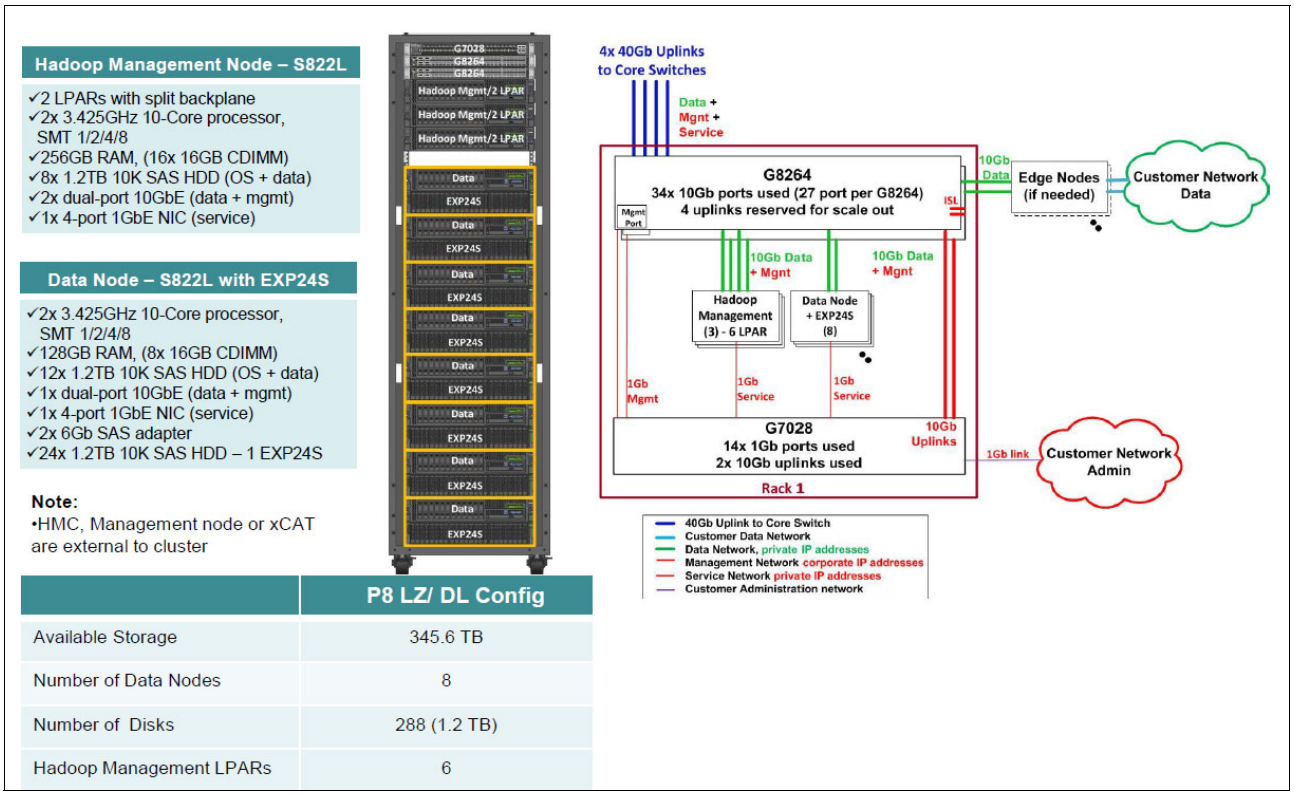
10. Slimline Open Firmware (SLOF) is initialization and boot source code based on the IEEE-1275 standard (also known as the Open Firmware standard). If you're a software engineer interested in PowerPC based systems, it can be invaluable when developing boot firmware, an operating system, or low-level application code. For hardware engineers, it provides fast start of PowerPC systems, verification, and prototyping of hardware support and drivers. SLOF is important, too, in that it enables a vendor to separate the low-level firmware from the SLOF layer. This firmware is part of the POWER8 architecture on both of the POWER8 (IBM PowerLinux S822L) servers.
11. The flexible service processor (FSP) is an advanced service processor that provides function to initialize the system chip sets, boot to a hypervisor run time, interface to the management module, and provide runtime support for advanced reliability, availability, and serviceability (RAS) functions. FSP is an additional layer in the hardware to closely monitor the other hardware components, and during the event, it notifies the system administrator or vendor to work proactively before the failure to avoid disruption in the service and so on. This advanced service processor is part of the POWER8 architecture on both of the POWER8 (IBM PowerLinux S822L) servers.

The sizing and configuration for the BigInsights Landing Zone/Data Lake solution (one of the POWER8 servers in this SDE solution) is aligned with IBM big data reference architecture, as shown in Figure 3-19 on page 94.

In addition, the Application Ready Solution for IBM InfoSphere BigInsights provides a powerful big data MapReduce analytics environment and reference architecture that is based on IBM PowerLinux servers, IBM Platform Symphony®, IBM GPFS, and integrated storage. The solution delivers balanced performance for data-intensive workloads, with tools and accelerators to simplify and speed application development. The solution is ideal for solving time-critical, data-intensive analytics problems in various industry sectors.

For more information, see *IBM Solution for Hadoop, Power Systems Edition: An IBM Reference Architecture for InfoSphere BigInsights*:

http://public.dhe.ibm.com/common/ssi/ecm/en/dcl12376usen/DCL12376USEN.PDF



	P8 LZ/ DL Config
Available Storage	345.6 TB
Number of Data Nodes	8
Number of Disks	288 (1.2 TB)
Hadoop Management LPARs	6

Figure 3-19 BigInsights on POWER8 -Landing Zone/Data Lake Architecture

The Power S822L is the ideal platform for supporting a general purpose, data landing zone or for a beginning “data lake.” Simply put, a data lake is a repository that holds a vast amount of raw data in its native format until it is needed. Although a hierarchical data warehouse stores data in files or folders, a data lake uses a flat architecture to store data.

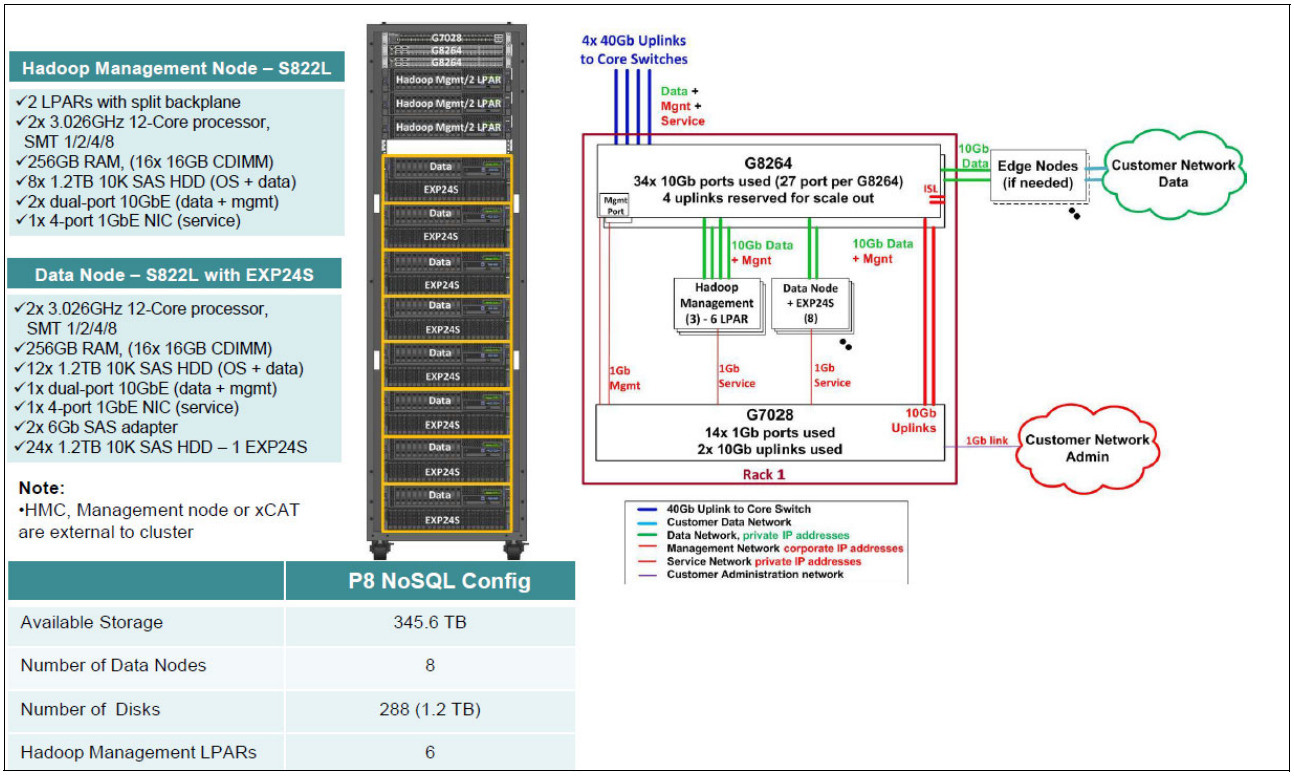
Note: IBM big data reference architecture will also be further explained in Section 2.6 of this chapter. 3.6.2, “Big data analytics workloads” on page 184.

This platform provides the following advantages:

- Rapid ingestion, analysis, and storage of unstructured data
  - ETL workloads including IBM InfoSphere DataStage®, Informatica
  - DataClick, BigInsights Console, and Scoop
  - Custom Map, Reduce Jobs, and Hive
  - Low cost per GB and fully protected
- Long-term storage of data with enterprise level protection
  - Balanced compute and storage resource, general purpose landing zone, data lake configuration has 345 TB storage
  - Higher storage than compute ratio for storage-intensive data lake has 1.272 PB storage
  - Compression and encryptions options
  - Flexible storage frameworks: HDFS or GPFS

- Flexible network and security options
  - Fully redundant routed or private networks
  - 10/40 Gb bonded Ethernet data network
  - 56 Gb FDR InfiniBand data network available
- Solid construction with hardware and software redundancy
  - OS, power, network, and name node service
  - Data protection using Hadoop replication
  - Hardware support for multiple storage configurations including hardware encryption

The sizing and configuration for the BigInsights NoSQL Complex Analytics solution (the second of the POWER8 servers in this SDE solution) is aligned with IBM big data reference architecture, as shown in Figure 3-20.



	P8 NoSQL Config
Available Storage	345.6 TB
Number of Data Nodes	8
Number of Disks	288 (1.2 TB)
Hadoop Management LPARs	6

Figure 3-20 BigInsights on POWER8 NoSQL/Complex Data Analytics

The POWER8 S822L is also an ideal platform for supporting powerful NoSQL and complex analytical data processing.

This platform supports and provides the following:

- Rapid Ingest, Store, Analyze and Query of unstructured data
  - DW and RDBMS replacement using BigSQL
  - HBase, Accumulo, Hive, and Custom Map/Reduce Jobs
  - BigR, R, or IBM SPSS® L
  - Fully protected storage

- Long-term storage of data with enterprise level protection
  - 345 TB raw storage
  - Compression and encryptions options
  - Flexible Storage frameworks: HDFS or GPFS
- Flexible network and security options
  - Fully redundant routed or private networks
  - 10/40 Gb bonded Ethernet data network
  - 56 Gb FDR InfiniBand data network available
- Solid construction with hardware and software redundancy
  - OS, Power, network, and name node service
  - Data protection using Hadoop replication
  - Each data node with 5 GB/s high IO bandwidth

For more information about the new 812L and S822L servers, see the IBM Redbooks publication titled *IBM Power Systems S812L and S822L Technical Overview and Introduction*, REDP-5098:

http://www.redbooks.ibm.com/abstracts/redp5098.html

Power Systems E870 and E880

The IBM Power System E870 (9119-MME) and IBM Power System E880 (9119-MHE) servers use the latest POWER8 processor technology that is designed to deliver unprecedented performance, scalability, reliability, and manageability for demanding commercial workloads.

The Power E870 is a highly-scalable rack-mount system optimized for running AIX, IBM i, and Linux workloads. The Power E870 is a modular-built system and uses one or two system nodes together with a system control unit. The Power E870 (9119-MME) server is a powerful POWER8-based system that scales up to eight sockets. Each socket contains a single 8-core or 10-core POWER8 processor. Thus, a fully-configured Power E870 can scale up to 64 or 80 cores. Each system node provides four processor sockets for POWER8 processors and 32 CDIMM slots supporting a maximum of 8 memory features. Using the 256 GB memory features, the system node can support a maximum of 2 TB of RAM. Thus a fully-configured Power E870 can support up to 4 TB of RAM. Each system node provides eight PCIe Gen3 x16 slots. Each system node is five EIA-units tall (5U) and the system control unit is two EIA-units (2U) tall. The Power E870 is housed in a 19-inch rack.

The Power E880 is a highly-scalable rack-mount system optimized for running AIX, IBM i, and Linux workloads. The Power E880 is a modular-built system. The Power E880 (9119-MHE) server is a powerful POWER8-based system that scales up to 16 sockets. Each socket contains a single 8-core POWER8 processor. Thus, a fully-configured Power E880 can scale up to 128 cores. In 2014 up to two system nodes per server are supported or a maximum of 64-cores.In 2014, the system is built of one or two system nodes together with a system control unit. Each system node provides four processor sockets for POWER8 processors and 32 CDIMM slots supporting a maximum of 8 memory features. Using the 512 GB memory features, the system node can support a maximum of 4 TB of RAM. Thus a two-node Power E880 can support up to 8 TB of memory. A four-node Power E880 can support up to 16 TB of RAM. Each system node provides eight PCIe Gen3 x16 slots. Each system node is five EIA-units tall (5U) and the system control unit is two EIA-units (2U) tall. The Power E880 is housed in a 19-inch rack.

For more information about these new E870 and E880 servers, see *IBM Power Systems E870 and E880 Technical Overview and Introduction*, REDP-5137:

http://www.redbooks.ibm.com/redbooks.nsf/RedpieceAbstracts/redp5137.html

For this IBM SDE solution, we have deployed IBM POWER7 and POWER8 series servers.

Alternate Power SDE-related systems reference architecture

There is an alternate Power Systems example SDE solution architecture with PowerFlex that will enable you to design, build and integrate your InfoSphere MDM Server Workbench into a very flexible building block, with System x86 and Power Systems on 1 chassis. Figure 3-21 is an illustration showing the components and layers in this “build your own” solution with even further flexibility at the Platform as a Service (PaaS) layer, based on customer requirements.

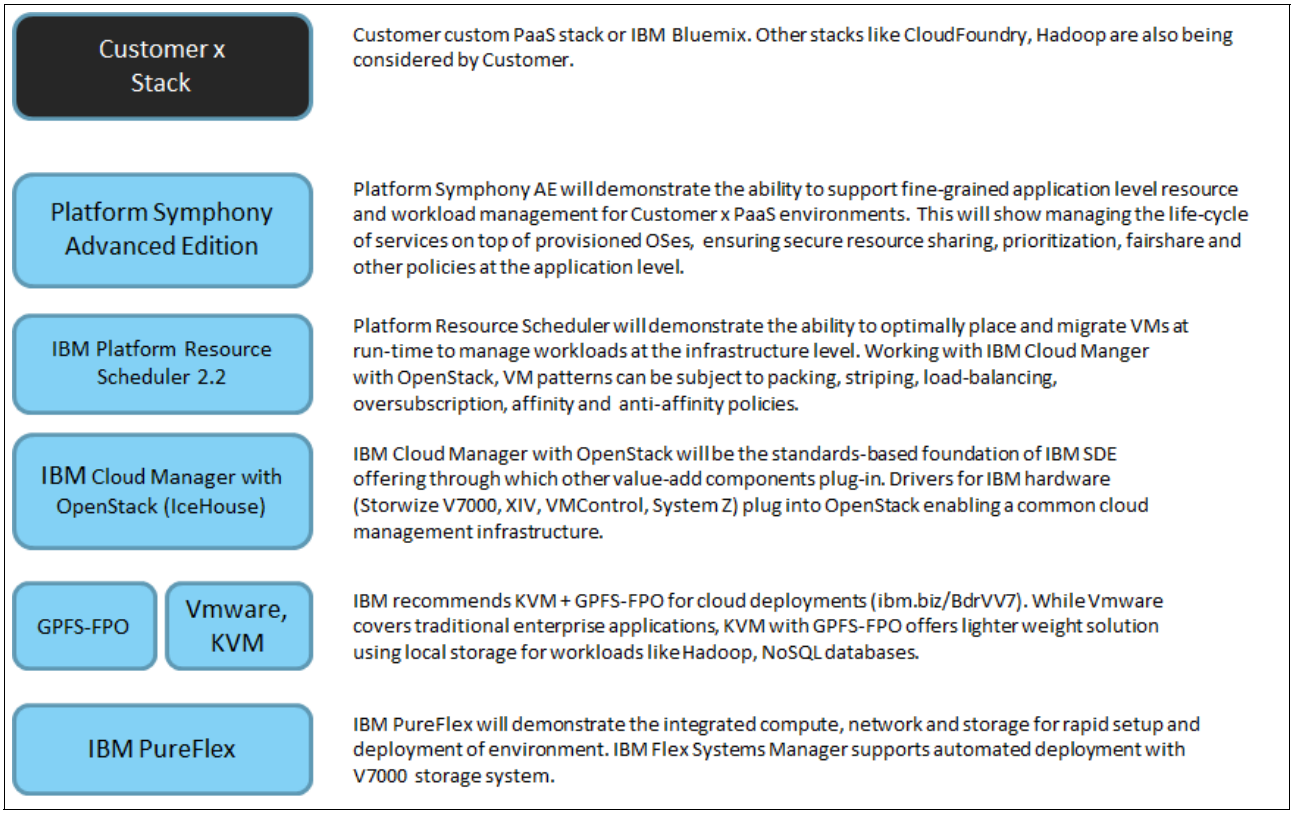


Figure 3-21 SDE IBM PureFlex® based alternate architecture

Figure 3-22 shows a picture of the IBM PureFlex System SDE solution.

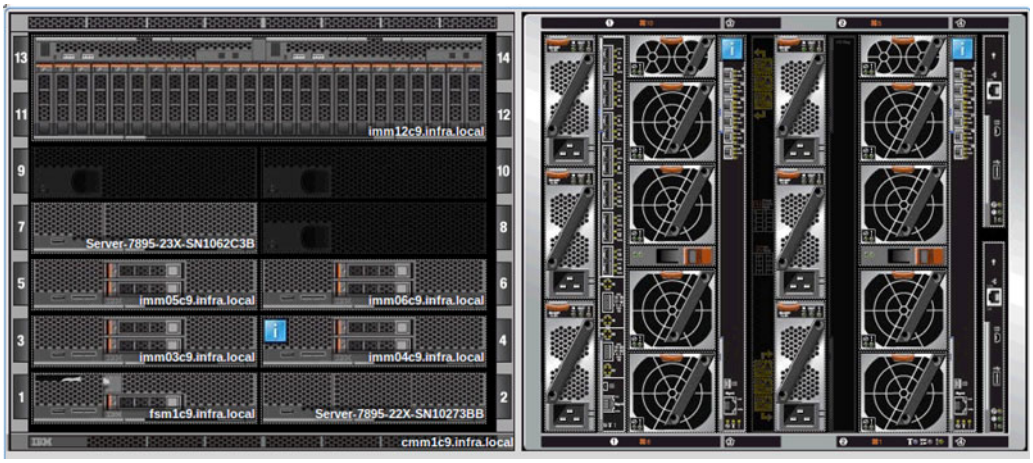


Figure 3-22 IBM SDE PureFlex Solution

Figure 3-22 has the following configuration:

- 1 Chassis
  - 2X -10G Ethernet switch
  - 2X -8G FC switches
- IBM Flex System® Manager (FSM)
- 9 x86 x240 Compute nodes
  - 2x -4 GHz Xeon CPUs (32 logical cores)
  - 256 GB RAM
  - IBM SmartCloud Entry installed on one blade
- 1 Power Compute node
  - 1 p270 - 2CPU/8 Cores (16 CPU) 4.1 GHz POWER7 + processor
  - 128 GB RAM
  - IBM AIX, IBM PowerVM Enterprise Edition
- 1x Flex V7000
  - 10x 500 GB SSD drives
  - 6x 600 GB SAS hard-disk drives
- Jump box
  - For remote access capability

IBM System z

So why System z for a cloud as part of the SDE solution architecture? The benefits of a mainframe in the data center are still relevant today, in terms of protecting and running critical business systems. Increasingly, organizations are turning to cloud computing to improve their IT responsiveness and to reduce costs. And for enterprise workloads, which require highly reliable and secure platforms to meet demanding service levels, many companies are building their clouds using a mainframe. Far from being out of date, mainframes are used around the world to build robust, trusted clouds.

Several recent IBM System z use cases are seeing striking results:

- An internal *business analytics* cloud service has enabled IBM to save more than USD $20 million over five years.
- Innovative mainframe *production applications* in Italy and Hawaii have achieved wide acclaim.
- A large insurance company built a dedicated *development and test* cloud to rapidly provision new environments.

- A new, cloud-based *social collaboration* platform at IBM supports 50 million instant messages daily and has lowered the related costs by 75%.
- A large IT service provider in Africa and a state government in the United States are deploying *next-generation* mainframe clouds to deliver scalable, secure, and customizable services.

A cloud built on a mainframe, such as IBM zEnterprise® or others in the IBM System z family, has several distinct advantages:

- **Performance:** System z mainframes build upon years of IBM leadership in virtualization and can host more virtual machines in a single footprint than any other platform. With its superior management services, a mainframe enables resources to be shared among workloads, achieving the highest possible use of the platform.1 Further, because a mainframe has dedicated processors for I/O operations, workloads with high I/O demand, such as transaction processing, can run more efficiently. And resources can be dynamically scaled without affecting ongoing operations.
- **Reliability:** A System z mainframe is designed to operate with the highest levels of reliability and availability. Each component is built to be resilient with support for non-disruptive configuration changes and dynamic replacement capabilities. Many organizations using IBM mainframes report years going without any unplanned downtime.[3]
- **Security:** The mainframe has garnered some of the highest marks for security in the industry. The latest IBM zEnterprise mainframe earned Evaluation Assurance Level (EAL) 5 certification for its logical partitions and EAL 4+ certification for its use of Linux. With its fine-grained, multilayered security architecture, the zEnterprise platform excels in providing secure workload isolation, data protection and privacy, and offers extensive security and audit reporting to meet evolving industry standards and regulations. [4]
- **Cost savings:** A recent IBM study found that the three-year total cost of ownership (TCO) for a System z mainframe cloud can be 75% less than a third-party provider's public cloud and as much as 49% less than an x86-based private cloud. On the mainframe, variety of workloads can be consolidated and virtualized on a smaller footprint, which results in lower software costs when compared to other solutions. Further, with the System z smaller footprint, costs for labor, electricity, and floor space can be lower than with an x86-based private cloud.

The cloud market today is growing at a rapid pace, but many CIOs continue to have concerns about security and availability. IBM announced in April of 2014, the IBM Enterprise Cloud System uniting leading IBM software, storage and server technologies into one simple, flexible and secure factory-integrated solution. This provides those delivering cloud services with the ability to rapidly deploy a fully automated OpenStack based cloud orchestration and monitoring environment via a Linux environment with mainframe qualities of service.

To align with MSP and CSP business models and make the mainframe more accessible to a broader number of clients, IBM announced new System z utility pricing for MSPs that will provide managed service providers a 'pay as you grow' pricing model and reduce the total cost of acquisition of acquiring System z technologies by as much as 65 percent. With the ability to start small and scale up to 6,000 virtual machines in a single footprint, the enterprise cloud system and new utility base enables more clients to provide superior cloud services for as low as half the cost of public cloud or x86 private cloud alternatives.

---

[3] *Comparing Virtualization Alternatives: What;s best for your business?* Solitaire Interglobal Report, 2012 http://public.dhe.ibm.com/common/ssi/ecm/en/zsl03192usen/ZSL03192USEN.PDF

[4] *Secure the Enterprise with Confidence Using aMainframe Infrastructure,* Forrester Report, March 2013 http://public.dhe.ibm.com/common/ssi/ecm/en/zsl03233usen/ZSL03233USEN.PDF

Note: System z in not in scope for this SDE solution nor is it currently deployed and integrated in our lab environment to our solution architecture. However, the next section explains how an IBM System z cloud can be integrated to the SDE solution through Cloud Orchestrator, which is a component in the Cloud Management Suite for System z, and enables infrastructure, platform, and advanced orchestration services. In addition, integration is also available through IBM Cloud Manager (Cloud Manager V4.2, which enables basis infrastructure cloud services. In addition, z Cloud integration to the SDE solution will be provided in an update to this book or another Redbooks publication.

Cloud Management Suite for System z is a single offering that includes IBM OMEGAMON® OMEGAMON XE for Messaging for z/VM and Linux, Tivoli Storage Manager Extended Edition and Cloud Orchestrator. It uses a simple pricing metric (z/OS VUs) enabling customers to rapidly deploy and manage a cloud on System z.

The suite contains the following components:

- IBM Orchestrator (formerly SmartCloud Cloud Orchestrator) drives the adoption of the Cloud on zEnterprise solution enabling business and infrastructure agility across the entire data center.
- IBM Tivoli OMEGAMON XE on z/OS provides detailed monitoring and problem management throughout IBM System z and the IBM zEnterprise system.
- Tivoli Storage Manager Extended Edition will provide back up and restore for the cloud workloads.
- IBM Tivoli Monitoring to monitor and manage system and network applications on a variety of operating systems, track the availability and performance of your enterprise system, and get reports to track trends and troubleshoot problems.

Key capabilities

- Orchestration and workflow management of request for new VM instances
- Tools to creating a set of standard workload patterns
- Provisioning VM instances and workload patterns to target hardware
- Centralized management of VM instances to service level agreements

For more information, see IBM Cloud Management Suite for System z V1.1.0 in IBM Knowledge Center:

http://ibm.co/1IHerMt

Figure 3-23 shows an illustration of the integrated z Cloud in an SDE environment, using IBM Cloud Orchestrator (formerly SmartCloud Orchestrator).

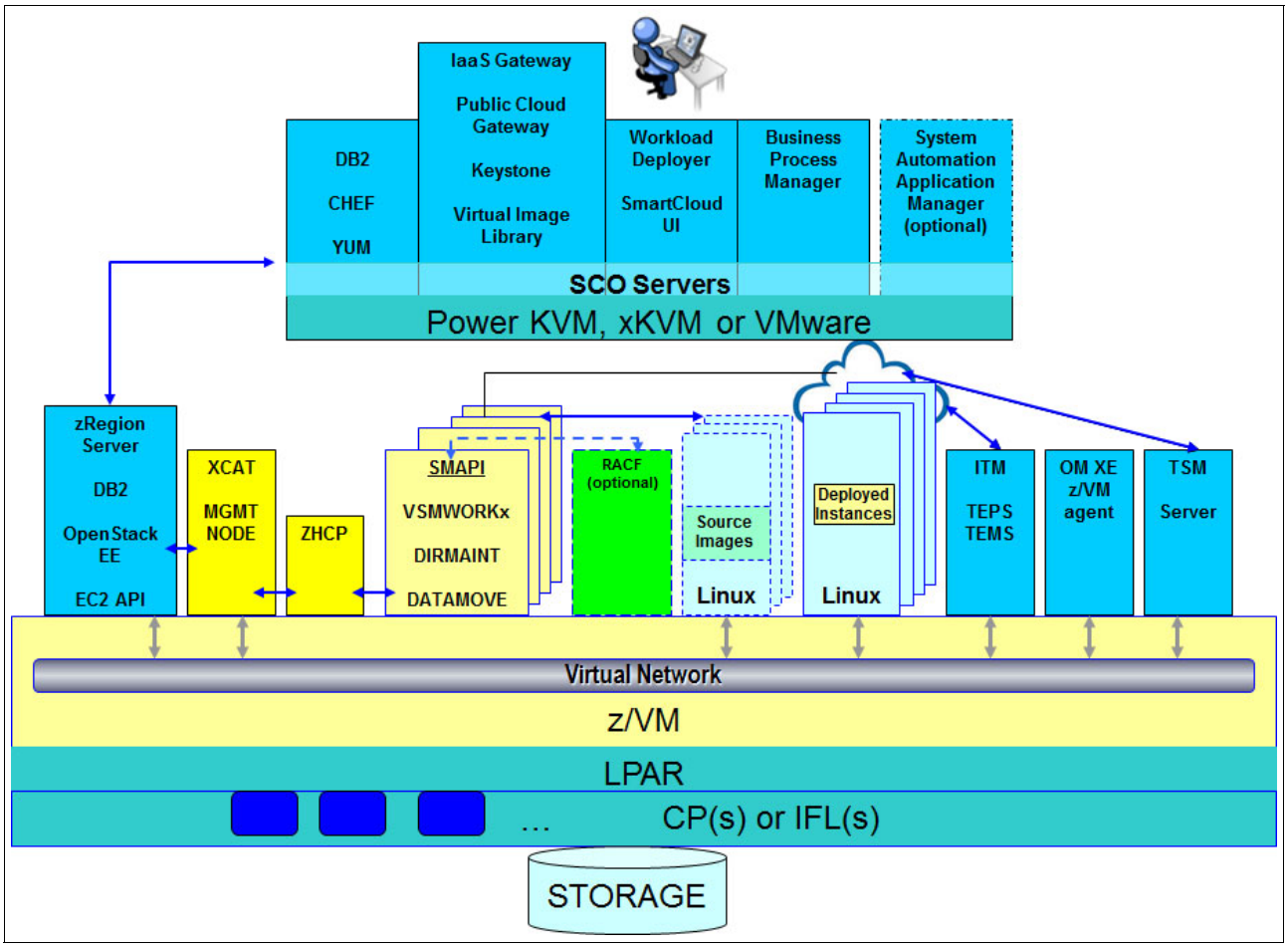


Figure 3-23 Cloud Management Suite for System z - SDE Integration

Note: xCAT, as seen above, is not a component in the Cloud Management Suite for System z, but it is a very integral and important piece of the solution that provides provisioning for the workloads. xCAT is the Extreme Cluster/Cloud Administration Toolkit. xCAT offers complete management for HPC clusters, render farms, grids, web farms, online gaming infrastructure, clouds, data centers, and whatever tomorrow's buzzwords are. It is agile, extensible, and based on years of system administration best practices and experience. It enables you to accomplish several key tasks:

- Provision operating systems on physical or virtual machines: Red Hat Enterprise Linux (RHEL), CentOS, Fedora, SLES, Ubuntu, IBM AIX, Microsoft Windows, VMware, kernel-based virtual machine (KVM), IBM PowerVM, IBM PowerKVM, IBM z/VM
- Provision using scripted install, stateless, statelite, iSCSI, or cloning
- Remotely manage systems: Lights-out management, remote console, and distributed shell support
- Quickly configure and control management node services: DNS, HTTP, DHCP, TFTP, NFS

For more information, see the SourceForge website:

http://sourceforge.net/p/xcat/wiki/Main_Page/

In the next section, hypervisors (xKVM, PowerKVM, PowerVM, z/VM) and respective virtualization technologies are described in the SDE solution architecture.

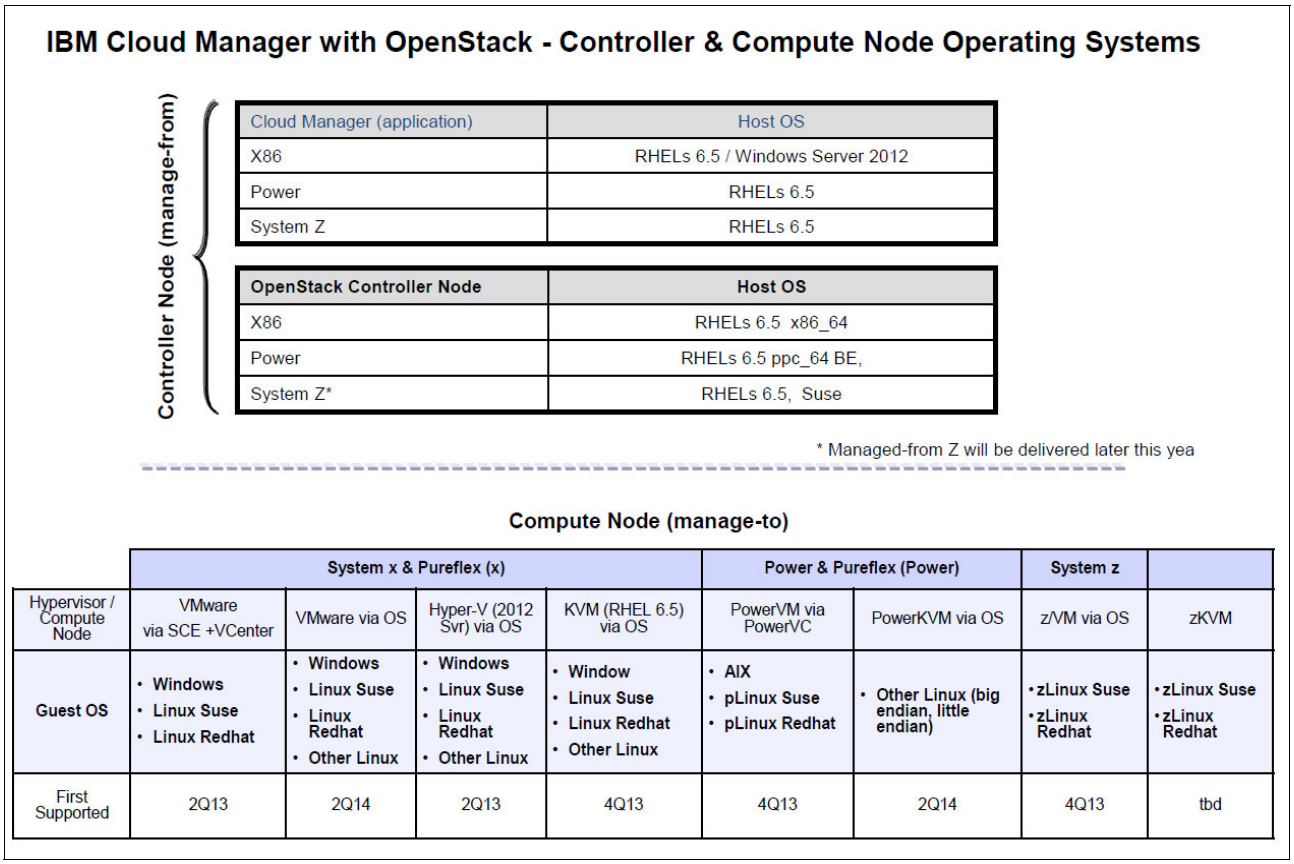
3.4 Hypervisors xKVM, PowerKVM, PowerVM, z/VM, VMware

One of the key strengths of the OpenStack and SDE solution is for the capability to work with the hypervisors that are available in the market today, and for the foreseeable future. A hypervisor or virtual machine monitor (VMM) is a piece of computer software, firmware or hardware that creates and runs virtual machines. A computer on which a hypervisor is running one or more virtual machines is defined as a host machine. Each virtual machine is called a guest machine.

The hypervisor market is becoming a commodity. IBM will continue to optimize our cloud platform with added differentiating capabilities needed by clients (optimized workloads, image management, DevOps, orchestration).

The other key benefit of OpenStack in context to hypervisors, is the capability to reduce costs with an open standardized IaaS layer which promotes hypervisor choice, so there is not really a need to be forced into, or locked into a specific technology for virtualization. In this section, we describe the reference architecture for each of the hypervisors that can work in the OpenStack and SDE solution.

Figure 3-24 illustrates the relationship between the supported Controller and Compute node operating systems:



IBM Cloud Manager with OpenStack - Controller & Compute Node Operating Systems

Controller Node (manage-from)

Cloud Manager (application)	Host OS
X86	RHELs 6.5 / Windows Server 2012
Power	RHELs 6.5
System Z	RHELs 6.5

OpenStack Controller Node	Host OS
X86	RHELs 6.5 x86_64
Power	RHELs 6.5 ppc_64 BE,
System Z*	RHELs 6.5, Suse

* Managed-from Z will be delivered later this yea

Compute Node (manage-to)

	System x & Pureflex (x)				Power & Pureflex (Power)		System z	
Hypervisor / Compute Node	VMware via SCE +VCenter	VMware via OS	Hyper-V (2012 Svr) via OS	KVM (RHEL 6.5) via OS	PowerVM via PowerVC	PowerKVM via OS	z/VM via OS	zKVM
Guest OS	• Windows • Linux Suse • Linux Redhat	• Windows • Linux Suse • Linux Redhat • Other Linux	• Windows • Linux Suse • Linux Redhat • Other Linux	• Window • Linux Suse • Linux Redhat • Other Linux	• AIX • pLinux Suse • pLinux Redhat	• Other Linux (big endian, little endian)	• zLinux Suse • zLinux Redhat	• zLinux Suse • zLinux Redhat
First Supported	2Q13	2Q14	2Q13	4Q13	4Q13	2Q14	4Q13	tbd

Figure 3-24 IBM Cloud Manager, Controller and Compute node operating systems

Before the explanation of the hypervisors, it is important to note just how prevalent Linux is in today's world and for companies, and why IBM is focusing on it terms of investment:

- Linux is the world's fastest-growing operating system

  http://www.ibm.com/systems/hardware/browse/linux/

- Over 90% of world's fastest supercomputers, including the top 10 in the Top 500 list, run on Linux

  http://top500.org/

- 8 of the world's top 10 websites, including Google, YouTube, Yahoo, Facebook, and Twitter, run on Linux

  http://ibm.co/1g3vx0i

KVM, the kernel virtual machine, consists of several components:

- Primarily, a kernel module, `kvm.ko`, that turns Linux into a hypervisor
- Brings core virtualization and hypervisor features to the Linux kernel
- A machine emulator and virtualizer for users called QEMU (short for quick emulator)
- Provides emulation and virtual devices + control mechanisms
- A standard interface library: `libvirt`
  - The standard library used to manage virtual machines
  - Provides an API

These pieces convert a Linux kernel into a hypervisor:

- Existing Linux scheduler and facilities used
- Virtual machines exist as userspace processes to the kernel/hypervisor
- This Linux kernel is designated as the “Host”
- Virtual machines are called “Guests”

KVM also:

- Requires hardware virtualization extensions, including paravirtualization where applicable

> **Note:** Paravirtualization – a virtualization technique that presents a software interface to virtual machines (VM) that is similar but not identical to that of the underlying hardware.

- Supports multiple architectures including PowerPC
- Provides competitive performance and feature set and advanced memory management

KVM approaches virtualization through the needs or requirements of the hypervisor, which are the following:

- A scheduler and memory management
- An I/O stack
- Device drivers
- A management stack
- Networking
- Platform support code

Linux has support for all of the above, and KVM reuses as much of the Linux-base code as possible. KVM's focus is on virtualization, and leaves the other components to the respective developers. KVM benefits (and will continue to benefit) from related advances in Linux as a technology.

3.4.1 xKVM

Linux for System x (informally called “x Linux” by some), or “Linux on x86,” KVM (for Kernel-based virtual machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor-specific module, kvm-intel.ko or kvm-amd.ko. KVM also requires QEMU.

- Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, and so on.
- The kernel component of KVM is included in mainline Linux, as of 2.6.20.
- KVM is open source software.

The KVM architecture and infrastructure consists of the following 2 primary types of nodes:

The host is the physical server running Linux with the KVM hypervisor. A host is a physical server running as a hypervisor to provide virtualized hardware interfaces to guest operating systems. These virtualized interfaces allow the guest to execute normal operations as if they were on physical hardware.

- KVM, as a hypervisor, is managed by an infrastructure of tools. Some of these tools may exist in the Linux vendor's ecosystem, but many of them will occur as third-party products.

- The Linux operating system supports a variety of hypervisors. Originally, both Red Hat and SUSE targeted Xen as its strategic hypervisor. However, Xen's earlier nature was such that its emulation layer was abstracted from the Linux kernel, which did not allow Xen to support the full extent of Linux's supported technologies without those modules being compiled and tested within the Xen kernel itself. Initially, only Red Hat turned to KVM, but eventually SUSE and other Linux distributions also went in that direction. SUSE continues to support Xen, but Red Hat has made it an alternative, leaving KVM as its focus.
- KVM is a Linux kernel module that is loaded into the running Linux kernel that provides virtual machine process and memory management integration into the existing Linux kernel function. This allows Linux to treat KVM virtual machines as additional processes within the Linux operating system.
- KVM was introduced as Red Hat's strategic hypervisor in RHEL5.4 and enhancements continued through today. With the release of RHEL6, KVM made a large leap in maturity, allowing it to become more competitive in the Enterprise Virtualization market.
- Hardware emulation is handled by QEMU. KVM allows QEMU unprivileged access to the full range of supported devices within the Linux kernel. The unprivileged access level of QEMU increases the security of the hypervisor due to the increased difficulty in guest-based attacks to affect host-based process through the QEMU layer. However, the integration of KVM into the kernel adds the entirety of the kernel's integrated hardware and API support to KVM itself, increasing the range of supported hardware for solution deployment. From a security perspective, this also includes SELinux integration with virtualization through sVirt in Red Hat's implementations.

Figure 3-25 illustrates the KVM architecture described above.

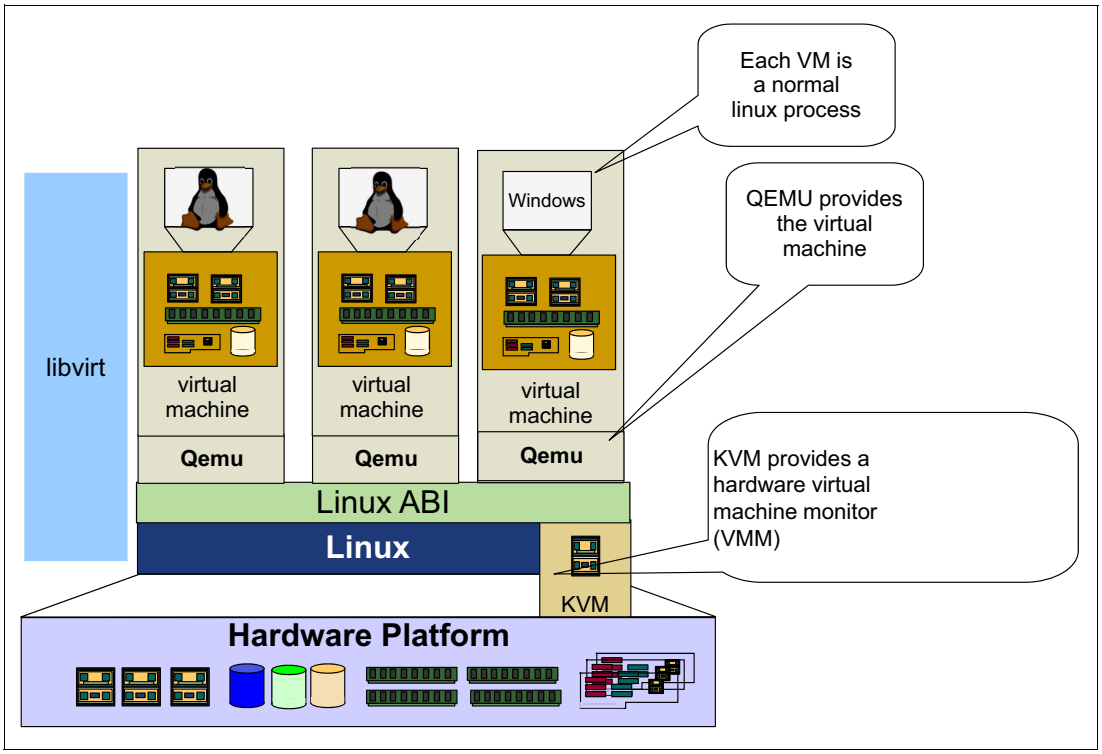


Figure 3-25 KVM architecture

KVM security is largely provided for by the existing policies governing Linux. However, additional policies for enhanced governance of the hypervisor components will need to be determined and documented. In RHEL, SELinux and sVirt provide the majority of the guest isolation. In SLES, AppArmor provides a subset of the same function. Depending on the implementation of Linux, other options may be configured.

Typically, one expects Linux to be installed as a full (thick) OS installation. This is true with most distributions. In the case of Red Hat, KVM is supported in two forms:

- RHEL Server
- Red Hat Enterprise Virtualization – Hypervisor (RHEV-H)

The primary difference between these versions is the level of configuration that is required during installation and that can be applied during steady state. Customers who are running RHEV-H have limited abilities to apply typical Linux security policies and patches. The RHEV-H server deployment is a reduced footprint hypervisor-only operating system that is designed to have a wide range of hardware support, similar to the native Linux kernel, but with limited additional software that would increase both operational costs and security risk. RHEV-H is managed only by the RHEV-Manager (RHEV-M) product at this time and explicitly uses the Virtual Desktop Server Manager (VDSM) agent to manage the virtualization components on the hypervisor, rather than the native libvirt. RHEL server is a full Linux deployment with a package set that can be tailored for your functions and security needs. Also, maintenance can be completed within the current strategic tools that you use on your servers.

For example, patching is completed through RHN or any proxy function that you are using to access RHN. RHEL Server often requires a couple of additional packages as dependencies for whichever management method is chosen. However, it can still be a far more limited package set than typically found with a full desktop environment.

For KVM-based solutions, 64-bit hardware is required that is Virtualization Technology (VT) capable, supports No Execute (NX) flag, and is enabled in the firmware. Both Intel and AMD VT-capable processors are acceptable, with the caveat that live migration between CPU vendors is not currently possible. In RHEL6.x, KVM supports 160 processor cores on a particular host. The theoretical limit is 4096. KVM hardware support is typically limited to the same list of supported hardware that is supported by RHEL. Use Red Hat's hardware compatibility listing or your hardware vendor's site to ensure support of the RHEL 5.5 OS:

http://www.redhat.com/rhel/compatibility/hardware/

Figure 3-26 illustrates a view of the KVM hypervisor and host (hardware).

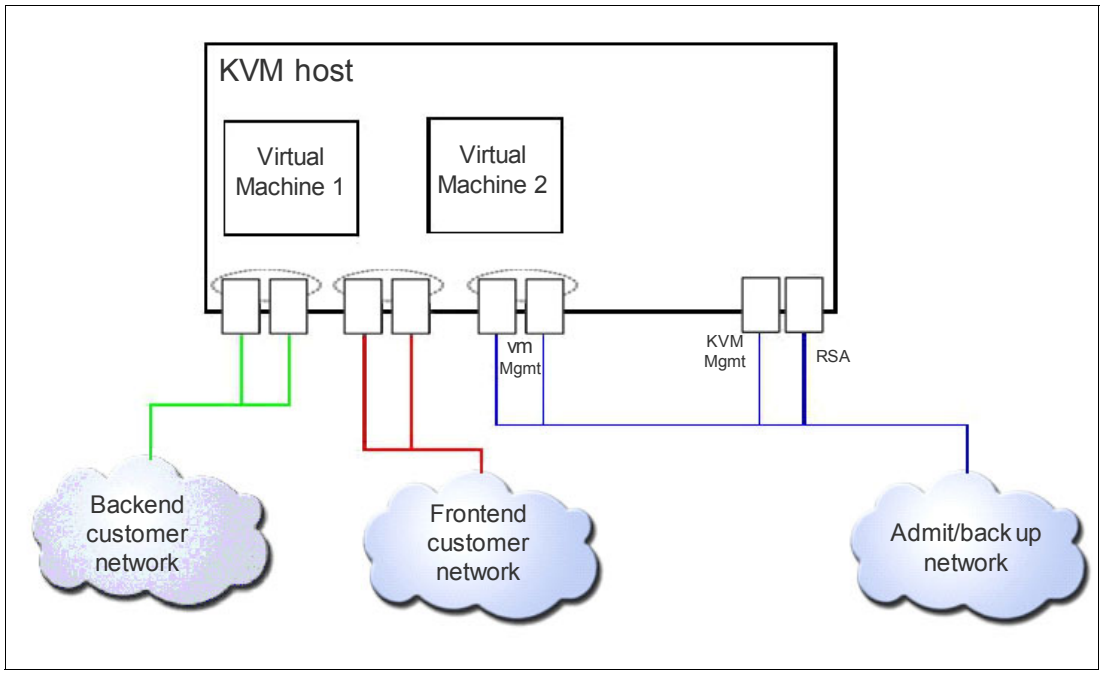


Figure 3-26 KVM and host

Figure 3-27, which would typically represent a medium-sized deployment, shows the KVM host is connected to the following networks:

- Front-end customer network

  Provides access for the customer. Several ports are bundled using NIC teaming to provide sufficient bandwidth or redundancy.

- Backend customer network

  Provides access for application traffic between servers, such as network traffic between database and application servers. Several ports are bundled using NIC teaming to provide sufficient bandwidth or redundancy.

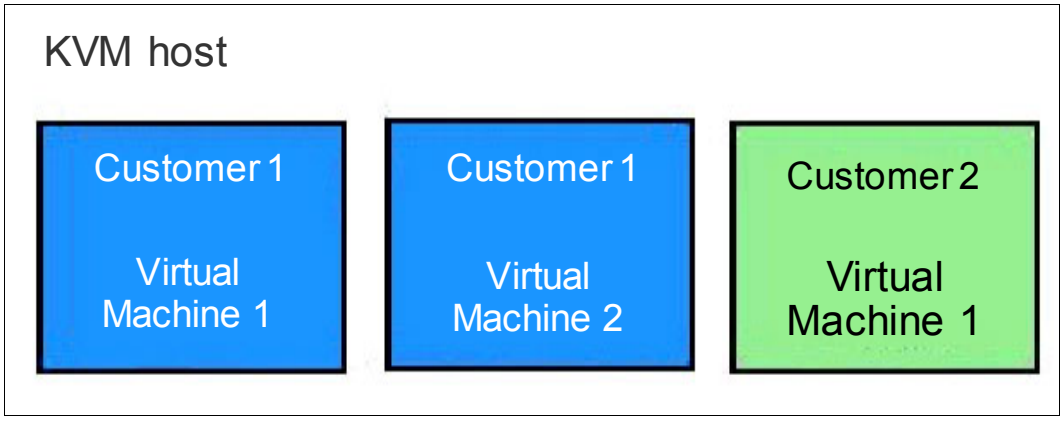


Figure 3-27 KVM cloud network topology

- Administration and backup network

  Provides access for system administrator login, systems management tools (for example, monitoring, metering, health checking, and so on) and backup.

- RSA network

  Provides access for hardware management through the RSA adapters.

Networking capabilities are provided by the kernel provided network modules. A single NIC can be configured as a single interface or the Linux bonding module can be used to “team” physical network interface cards (NICs) on the host into a single interface allowing redundancy or increased bandwidth. The interface can then be bound to a bridge adapter that is used to bridge traffic from the virtual network adapter interfaces in guests to the physical network. VLAN trunking (802.1q) is supported through the base Linux capabilities and can be used on top of the interfaces (bonded or physical) to provide bridging into trunked VLANs on a physical link.

Remote Console cards (RSA, IMM, IMMv2, and so on) should be used where available to allow remote management of the host server. The ID and passwords of these devices should be managed to standard shared password requirements. The IP addresses should be to an internal segment of the management network and not facing the customer or the Internet. Newer models of these devices support VLAN assignments.

3.4.2 KVM cloud network topology

Figure 3-28 illustrates the KVM cloud network topology, which would typically be implemented in a larger deployment in a public or private cloud.

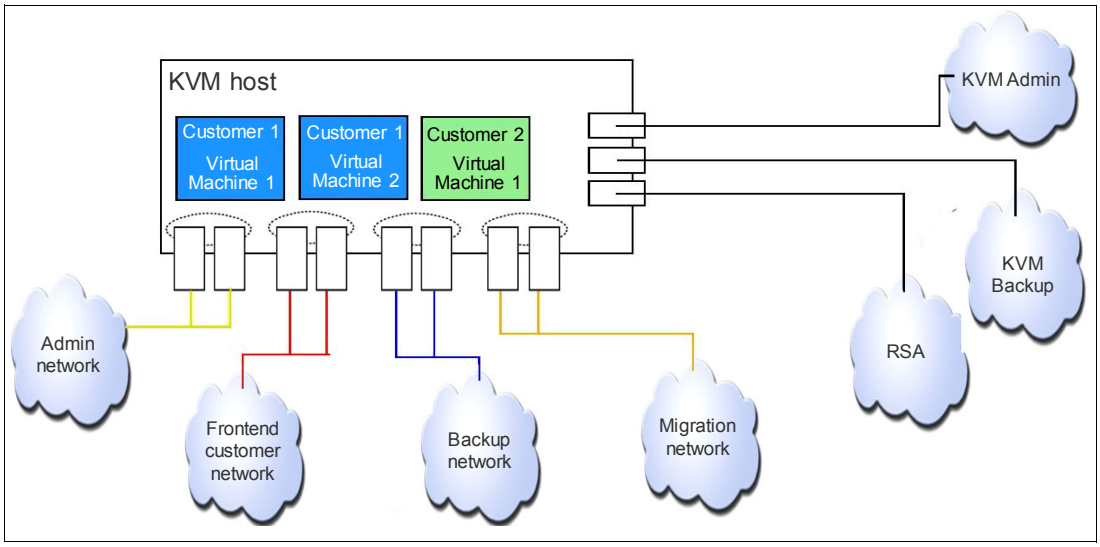


Figure 3-28 KVM cloud network topology

As Figure 3-28 shows, the KVM host is connected to the following networks:

Frontend customer network Provides access for the customer and customer applications. Several ports are bundled using NIC teaming to provide sufficient bandwidth or redundancy. Different customers are isolated from each other using VLANs.

Migration network	Is used during virtual machine migration operations to transfer image data from KVM host to another. Several ports are bundled using NIC teaming to provide sufficient bandwidth or redundancy.
Administration network	Provides access for system administrator login and systems management tools (for example monitoring, metering, health checking, and so on). Several ports are bundled using NIC teaming to provide sufficient bandwidth or redundancy. Different customers are isolated from each other using VLANs.
Backup network	Provides access for backup traffic. Several ports are bundled using NIC teaming to provide sufficient bandwidth or redundancy. Different customers are isolated from each other using VLANs.
RSA network	Provides access for hardware management through the RSA adapters.
The guest	A guest is the virtual machine instance that is hosted on a hypervisor. It interacts with virtualized hardware provided by the hypervisor that is eventually translated to the physical hardware devices where data is stored, computed, or transmitted.

- Guest OS support
  - Given that QEMU provides the hardware emulation, technically KVM can host many different x86-based guest operating systems, but only a few have been tested and verified as being stable. Each OS vendor providing KVM support has identified the Guest operating systems that they support within the operating system. Using supported Guest operating systems is not purely about tested stability, but also the paravirtualization driver packages are made available only for those operating systems. Here are the supported guests at the time of this writing:
  - For example, at the time of this writing, Red Hat Enterprise Virtualization (RHEV) 3.1 and Red Hat Enterprise Linux (RHEL) 6.x supports:
    - RHEL 3/4/5/6 in both 32 and 64bit
    - Windows XP as 32bit only
    - Windows 7/2003(sp2+)/2008 as 32 and 64bit
    - Windows 2008 R2 as 64bit only
  - SUSE Linux Enterprise Server (SLES) 11 SP2 is currently providing Guest support for:
    - SLES 9 SP4/10 SP4/11 SP1and2
    - Novell Open Enterprise Server 11 (Linux flavor only)
    - Windows 2003 SP2/ 2008 R2/ 2008 SP2
- Guest OS management
  - Guest operating systems should be managed in accordance with strategic tools and policies. There is little difference in the support of a virtualized OS versus a physically hosted OS. The users, processes, and services are all managed similarly. The differences lay in the tools that may be excluded from Guest installation due to their support within the hypervisor layer, the lack of firmware to be maintained for the Guest's hardware, and the ease of obtaining remote hardware management (ex. stop/start/console).

For more information about xKVM, see the following web pages:

http://www.linux-kvm.org/page/Main_Page

https://www.suse.com/documentation/sles11/singlehtml/book_kvm/book_kvm.html

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Virtualization/3.1/html/Administration_Guide/Supported_virtual_machines.html

https://access.redhat.com/certified-hypervisors

http://www.linux-kvm.org/page/Guest_Support_Status

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Host_Configuration_and_Guest_Installation_Guide/index.html

https://access.redhat.com/articles/906543

http://wiki.apparmor.net/index.php/Libvirt

3.4.3 PowerKVM

KVM runs on just about every platform that Linux is ported to, and now includes Power Systems. PowerKVM is an IBM product designed, developed, and built with embedded Linux and all KVM modules and functions:

- As an “appliance” (this appliance Linux OS is the hypervisor or host)
- Full shell (bash) provided
- Full access to fully compliant libvirt
- Built-in management tools and monitoring solutions:
  - Kimchi
  - Nagios
  - Ganglia
- Easy repository-based updates

The following installation options were included:

- Shipped preinstalled
- Optical media based install
- Network-based install
- Install media can also upgrade

Table 3-3 provides a comparison of terminology between PowerKVM, and PowerVM (which will be explained in the next section.

Table 3-3 Terminology comparison between PowerKVM and PowerVM

PowerKVM	PowerVM
Integrated Management Module (IMM)	FSP
Host, hypervisor	Hypervisor
Unified extensible firmware interface (UEFI) and the basic input/output firmware interface (BIOS)	PowerVM hypervisor driver (pHyp) firmware
KFM host userspace (qemu)	Virtual I/O Server (VIOS)

PowerKVM	PowerVM
Host userspace tools based on the libvirt API, including virsh	Integrated Virtualization Manager (IVM) Hardware Management Console (HMC)
KIMCHI or virt-manager	Integrated Virtualization Manager Hardware Management Console
Command-line message-based hardware management interface to manage IPMI-enabled devices on remote host with impitool	Integrated Virtualization Manager (IVM) Hardware Management Console (HMC)

Table 3-3 on page 110 provides a comparison of the unique features between PowerVM and PowerKVM.

So, how does all of this technology come together in PowerKVM and the IBM Power servers? If we look at this in the context to IBM Power Systems and virtualization, The Power platform consists of a vertical integration of hardware, firmware and software components that provide unmatched virtualization features using flexibility and performance.

The platform standards, guidelines and specifications are established by a governing body power.org *Power.org* defines:

- Processor ISA
- Memory management
- Architecture platform reference specifications
- POWER Architecture Platform Reference (PAPR)

  PAPR describes the environment in which a general-purpose operating system will run:

  - Bootstrap
  - Runtime
  - Shutdown function
  - Virtualization operation
  - Virtualization standards for the platform must be implemented using a combination of hardware, firmware and software.

Figure 3-29 on page 112 illustrates the reference logical Power virtualization architecture in terms of the starting point (before PowerKVM) software stack on POWER7.

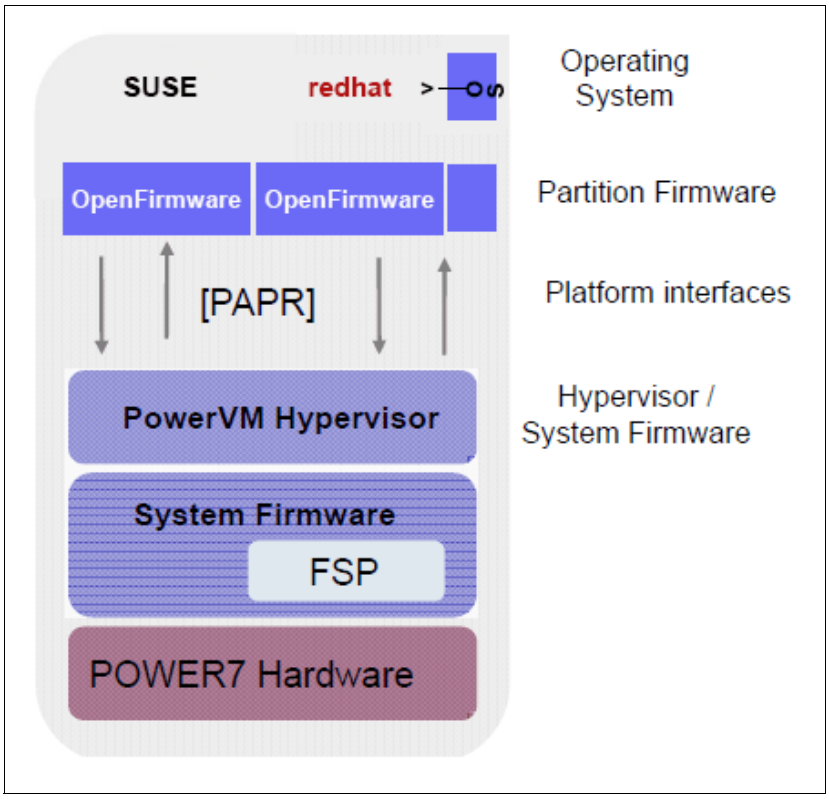


Figure 3-29 Power Virtualization Architecture - POWER7

Virtualization on POWER means the cooperation of hardware, firmware and software. This allows for efficient management of privileged hardware resources.

The hardware includes three privilege levels:

- Hypervisor
- Supervisor
- User

The hypervisor state includes partitioning/virtualization facilities via Special Purpose Registers control:

- MMU hash table access
- Interrupt control (which ones go to VM, which ones go to hypervisor)
- Entire platform designed for cooperation or *paravirtualization*
- Some aspects of the machine cannot be emulated or spoofed
- Operating systems have some virtualization responsibilities
- OS calls directly into the hypervisor for some things (hcalls)

The technology and architecture is always paravirtualized. The hypervisor runs in hypervisor mode (highest privilege level), and it has access to all memory and system resources. The operating systems in guests/VMs/LPARs run in supervisor mode:

- Virtualized operating systems must conform to the PAPR interfaces: AIX, IBM i, and ppc64 Linux kernel
- PAPR conformance gives knowledge of when to call into the hypervisor
- No need to trap and emulate privileged instructions
- These run at full hardware speed

- Hypervisor and VMs each have their own MMU hash tables

The above architecture and technology results in very fast, high performance with very low overhead virtualization.

The PowerVM hypervisor (pHyp) is the only software that runs in hypervisor mode on the processor. It has the following responsibilities:

- Managing CPU
- Managing memory
- Routing interrupts
- Some simple transports
- Scheduling of virtual machines
- Some platform management
- Error recovery

The pHyp provides interfaces for management, but does not allow a direct login, and it does **not** handle the virtualization of input and output devices. It has been deliberately kept as simple as possible, but has added functions over the years:

- Manages Non-Uniform Memory Architecture (NUMA) layouts
- Processor affinity
- Routing of virtualized networking between virtual machines on the same physical server

Now that we have described the Power Virtualization stack, it is good to see how this compares to the PowerKVM reference logical virtualization architecture software stack on POWER8, which can be seen in Figure 3-30.

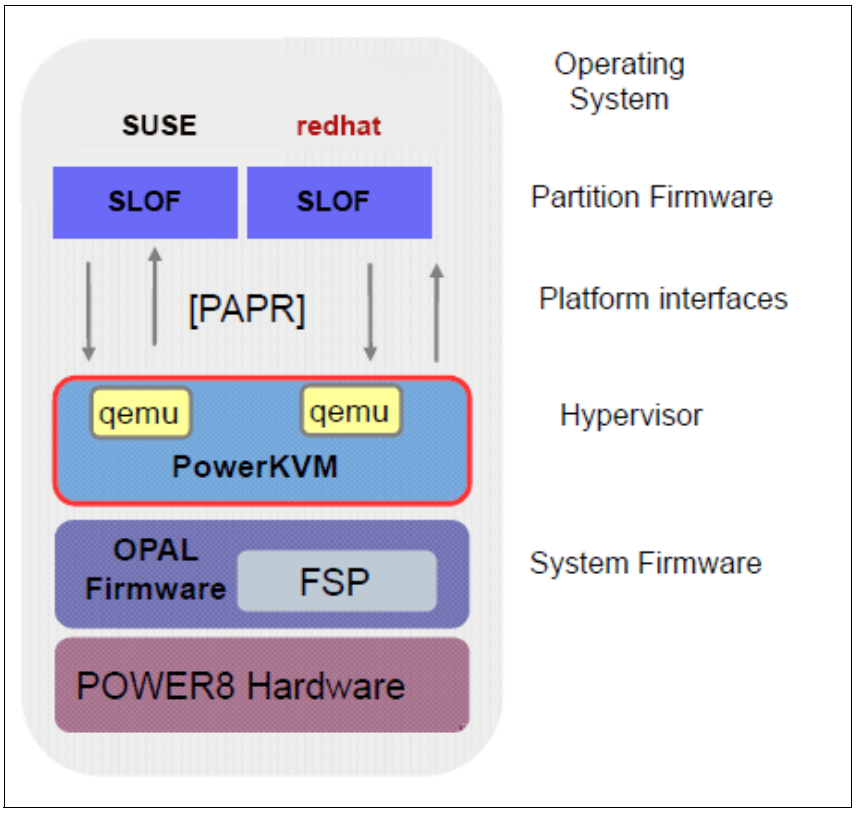


Figure 3-30 Power Virtualization - POWER8 with PowerKVM

The host OS runs in hypervisor mode on the processor:

- Guest kernels run in supervisor mode
- Host has access to all memory and machine resources
- Host does not trap or emulate privileged instructions from guests

There is special firmware required:

- That allows access to hypervisor mode
- Disables pHyp

KVM guests are paravirtualized using the PAPR interfaces:

- Same interfaces as PowerVM
- Existing Linux distributions for Power will work (SLES, RHEL)

Breaking the PowerKVM hypervisor down into further detail, we can see how IBM supports the various layers and components in the architecture as shown in Figure 3-31.

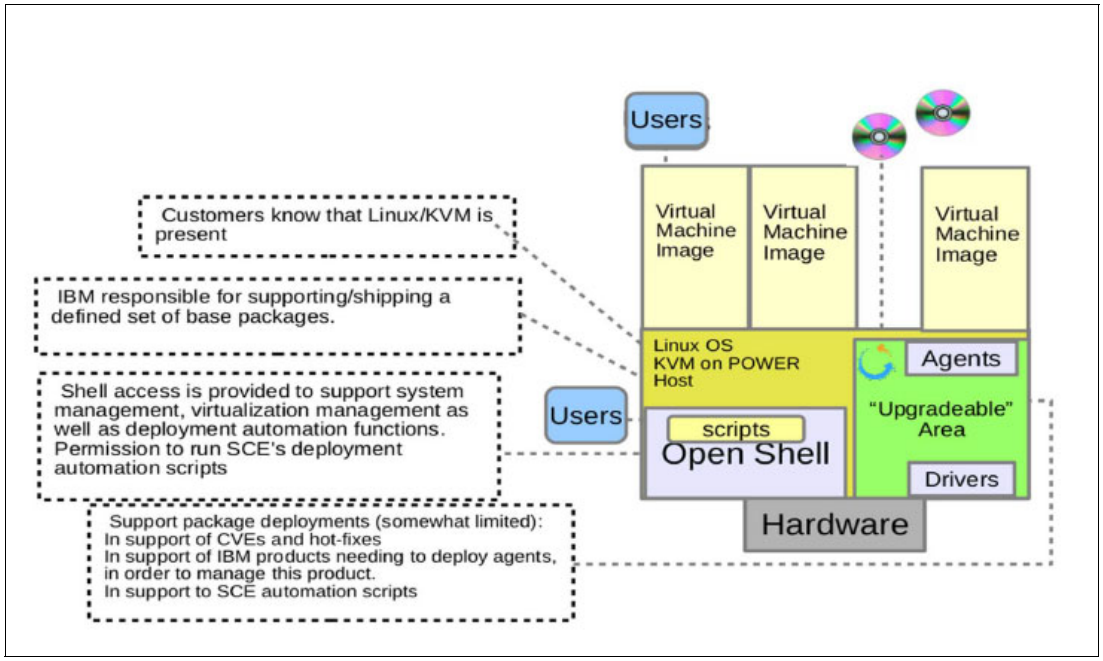


Figure 3-31 PowerKVM hypervisor

Now that we have explained the PowerKVM architecture, it is good to have a comparison of the PowerKVM versus PowerVM (both included in this SDE solution) features as shown in Table 3-4.

Table 3-4 Comparison of the features of PowerKVM and PowerVM

PowerVM unique features not in PowerKVM	PowerKVM unique features not in PowerVM
Compute: ► Dedicated processors ► Shared processor pools ► Shared dedicated processors ► Guaranteed minimum entitlement ► Hard capping of VMs ► Capacity on demand ► IFLs	Compute: ► Uses POWER8 microthreading

PowerVM unique features not in PowerKVM	PowerKVM unique features not in PowerVM
Security: ► vTPM ► Existing security certifications ► Firmware-based hypervisor	Security: ► N/A (no unique differences)
I/O: ► NPIV* ► SR-IOV* ► Dedicated I/O devices* ► Redundant I/O ► Virtualization (dual VIOS)	I/O: ► N/A (no unique differences)
Storage: ► N/A (no unique differences)	Storage: ► NFS storage support ► ISCSI storage support
Linux operating system ► N/A	Linux operating system ► Ubuntu
Systems management ► Requires HMC	Systems management ► No HMC needed
Configuration ► Dynamic LPAR (DLPAR*) ► Support for AIX and IBM i VMs ► System pools	Configuration ► N/A

*PowerKVM function planned

In summary, this is what is different with KVM on Power versus xKVM:

- KVM is open source.
- Companies (for example, Red Hat) offer commercial KVM hypervisor products.
- On IBM x86, it is also possible to enable KVM on an existing Linux installation (which turns that Linux OS into a hypervisor).
- Not all companies, distributors, or solutions officially support both models.

The following are the internal differences:

- No “full virtualization” on Power
- KVM implements PAPR
- No full CPU emulation
- Qemu device models:
  - Disk:
    - virtio-scsi
    - virtio
    - spapr-vscsi
    - No IDE
  - Network:
    - virtio
    - E1000 (Intel)
    - Rtl (Realtek)
    - spapr-vlan
- Graphics
  - VGA (VNC backend only)
  - No Spice (coming later)

Also, it is beneficial to know the Linux distributions that are available on IBM Power systems. Figure 3-32 provides this information.

Linux	Release	Endian	Dedicated LPAR	PowerVM Guest	PowerKVM Guest
Redhat	5.10	Big	✓	✓	×
Redhat	6.4	Big	✓	✓	×
Redhat	6.5	Big	✓	✓	✓
SUSE	11 SP3	Big	✓	✓	✓
Ubuntu*	14.04	Little	×	×	✓

*Exploits P8

Figure 3-32 Linux distributions on IBM Power systems

In making your choice in terms of the Linux distribution, follow these guidelines:

1. Select the applications that you want to run on Linux on Power. This SDE solution uses a top-down, application-aware approach, so it's good to determine up front what applications you'd like on Linux and whether they are supported.
2. Look at the Linux distributions that are available for that application.
3. Select your Linux distribution.

For more information about PowerKVM, see these web pages:

- *IBM PowerKVM Configuration and Use*, SG24-8231, an IBM Redbooks publication

  http://www.redbooks.ibm.com/abstracts/sg248231.html
- KVM model

  http://www.linux-kvm.org/page/Main_Page
- libvirt

  http://www.libvirt.org/index.html
- QEMU

  http://wiki.qemu.org/Main_Page

3.4.4 PowerVM

Power Systems with optional IBM PowerVM virtualization offerings enable businesses to consolidate servers and applications, virtualize system resources, and provide a more flexible, dynamic IT infrastructure. IBM PowerVM virtualization technology is a combination of hardware and software that supports and manages virtual environments on IBM POWER5, POWER5+, POWER6, and POWER7 and POWER8 processor-based systems. These systems are available on IBM Power Systems and IBM BladeCenter servers as optional editions, and are supported by the IBM AIX, IBM i, and Linux operating systems. With this set of comprehensive systems technologies and services, you can aggregate and manage resources with a consolidated, logical view.

By deploying PowerVM virtualization and IBM Power Systems, you can take advantage of the following benefits:

- Lower energy costs through server consolidation.
- Reduced cost of your existing infrastructure.
- Better management of the growth, complexity, and risk of your infrastructure.
- Deliver services with superior economics by consolidating virtualized workloads.
- Deliver services faster by automating deployment of virtual machines and storage.
- Optimize use of server and storage resources to control cost and boost ROI.
- Scale your virtualized deployments without paying underlying performance penalties.
- Eliminate scheduled downtime by deploying live mobility between servers.
- Deliver higher-quality services by improving virtual resource management.

Complexity can work its way into any IT infrastructure, driven by the rollout of new applications and unanticipated change. However, adding servers in response to each demand for new workloads drives the need for more data center space, power, cooling, network cabling, data storage and administrative resources. Such complexity leads to inefficiency. The answer is virtualization, which enables organizations to consolidate multiple operating systems and software stacks on a single platform.

PowerVM provides the industrial-strength virtualization solution for IBM Power Systems servers and blades. Based on more than a decade of evolution and innovation, PowerVM represents the state of the art in enterprise virtualization and is broadly deployed in production environments worldwide by most Power Systems owners.

The IBM Power Systems family of servers includes proven[5] workload consolidation platforms that help clients control costs while improving performance, availability and energy efficiency. With these servers and IBM PowerVM virtualization solutions, an organization can consolidate large numbers of applications and servers, fully virtualize its system resources, and provide a more flexible, dynamic IT infrastructure. In other words, IBM Power Systems with PowerVM delivers the benefits of virtualization without limits.

PowerVM provides industrial-strength virtualization for AIX, IBM i, and Linux environments on IBM POWER processor-based systems. IBM Power Systems servers integrated with PowerVM technology are designed to enable clients to build a dynamic infrastructure that will help them to reduce costs, manage risk and improve service levels.

PowerVM also offers a secure and resilient virtualization environment, built on the advanced RAS (reliability, availability and serviceability) features, extreme scalability and leadership performance14of the IBM Power platform, based on the outstanding POWER7 processors.

The IBM POWER Hypervisor™ enables the hardware to be divided into multiple partitions (LPARs), and ensures isolation between them. The hypervisor orchestrates and manages system virtualization, including creating LPARs and dynamically moving resources across multiple operating environments. The POWER Hypervisor also enforces partition security, and can provide interpartition communication that enables the VIOS virtual SCSI, virtual Fibre Channel and virtual Ethernet functions.

PowerVM Editions deliver advanced virtualization functions for AIX, IBM i, and Linux clients such as IBM Micro-Partitioning® technology, VIOS, Integrated Virtualization Manager and Live Partition Mobility. PowerVM features provide the ability to virtualize processor, memory, and I/O resources to increase asset use and reduce infrastructure costs. PowerVM also allows server resources to be dynamically adjusted to meet changing workload demands, without a server shutdown.

[5] PowerVM Case Studies: ibm.com/systems/power/success/index.html

PowerVM Editions provide VIOS technology to facilitate consolidation of LAN and disk I/O resources and minimizes the number of required physical adapters in a Power system. The VIOS actually owns the resources that are shared with clients. A physical adapter assigned to the VIOS partition can be shared by one or more other partitions. The VIOS can use both virtualized storage and network adapters, making use of the virtual SCSI, virtual Fibre Channel with NPIV, and virtual Ethernet facilities.

You can achieve continuous availability of virtual I/O by deploying multiple VIOS partitions (dual VIOS) on an HMC-managed system to provide highly available virtual services to client partitions.

3.4.5 Architecture and solution context

The challenges of IT revolve around *volume* (scale of data), *variety* (different forms of data), *velocity* (analysis of streaming data), and *veracity* (uncertainty in data) in terms of a data intensive world we are all experiencing.

The volume of servers is growing exponentially. The variety of information technology options provides the opportunity to bring a variety of new types of information to the system which needs it. The velocity of change in an IT organization is driving the need to implement the correct IT services. The data veracity brings uncertainty of data, because the more data sources you access and the greater the variety of data types within these sources, the higher the probability of error becomes.

OS infrastructures need to be open, flexible, on demand, and responsive to change. To align server use costs with project requirements, a comprehensive set of virtual servers are defined that are relative to cost, performance, manageability, availability, scalability, functionality, and recoverability. Implementation of PowerVM and PowerKVM virtualization are essential to meet the requirements and challenges that the Four Vs of data present. Previously, to further optimize management of AIX virtualization, automation was required to manage according to polices and minimize human error. This is where the SDE solution provides this automation.

Figure 3-33 illustrates the PowerVM AIX Virtualization Architecture Model, which is useful to describe the PowerVM technology.

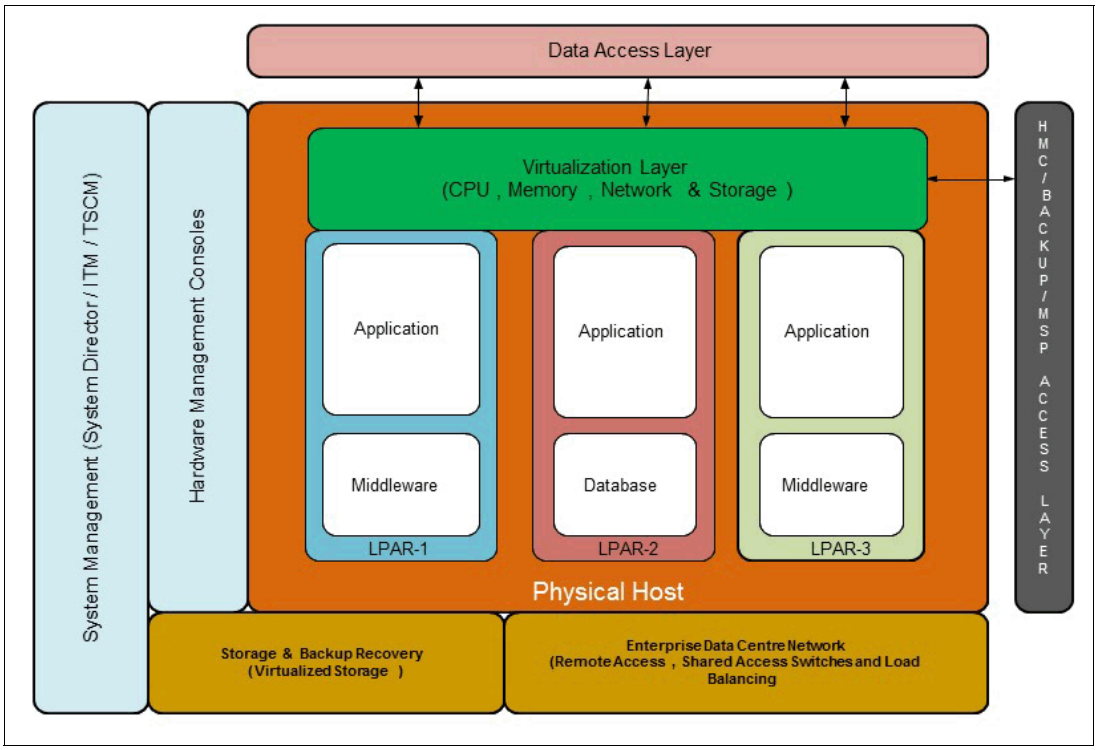


Figure 3-33 PowerVM AIX Virtualization Architecture Model

Note: This illustration is useful as a reference for key terms throughout this book.

Data access layer

This is where the applications access their data. There is no difference in this application access whether served by a physical or a virtualized server. The data access layer can also be referenced as a presentation layer. The presentation layer can come from the customer's application premises or the public network (Internet via DMZ).

Application

Examples are BigInsights, Internet Banking, Core Banking, CRM, Commerce applications, and so on that are running and hosted within the LPAR.

Middleware

Examples are IBM WebSphere® Application Server, IBM MQ, JBoss, and so on that are running within the operating system.

LPAR: Logical partition

The operating system is running inside a virtual machine or LPAR (logical partition) environment.

Virtualization layer (hypervisor)

Virtualization is a technique to decouple a physical configuration from the virtualized one. A virtualized entity provides flexibility and on-demand service attributes to changing business needs. Virtualized IT resources are grouped in resource pools to improve management and use. Virtualization is a trigger for increased standardization, consolidation, automation and infrastructure simplification. In PowerVM terminology, the hypervisor manages the virtualization layer.

Figure 3-34 provides a view of the hypervisor and virtualization layer.

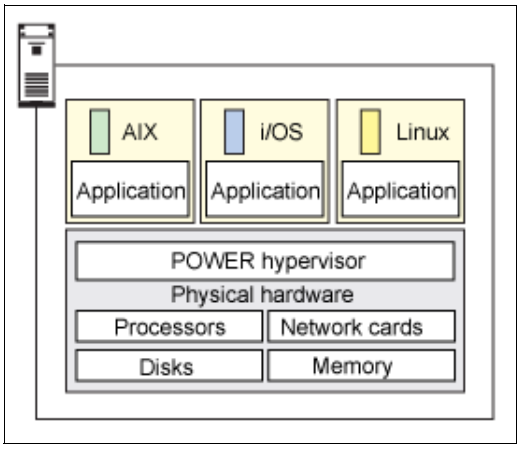


Figure 3-34 Architecture of the PowerVM hypervisor

3.4.6 Virtual I/O Server in relation to the PowerVM hypervisor and virtualization

The Virtual I/O Server (VIOS) is software that is in a Power Systems Server logical partition (LPAR). This software facilitates the sharing of physical I/O resources between client logical partitions within the server. The VIOS provides virtual SCSI target, virtual Fibre Channel, Shared Ethernet adapter, and PowerVM IBM Active Memory™ Sharing capability to client logical partitions within the system. As a result, client logical partitions can share SCSI devices, Fibre Channel adapters, Ethernet adapters, and expand the amount of memory available to logical partitions using paging space devices. The VIOS software requires that the logical partition be dedicated solely for its use. The VIOS is part of the PowerVM Editions hardware feature, and is part of the solution for this book, as standard building block and best practice for Power Systems virtualization.

Using the VIOS facilitates the following functions:

- Sharing of physical resources between logical partitions on the system
- Creating logical partitions without requiring additional physical I/O resources
- Creating more logical partitions than there are I/O slots or physical devices available with the ability for logical partitions to have dedicated I/O, virtual I/O, or both
- Maximizing use of physical resources on the system
- Helping to reduce the storage area network (SAN) infrastructure

The VIOS allows the sharing of physical resources between supported AIX, IBM i, and Linux partitions to enable more efficient use and flexibility for using physical storage and network devices.

The VIOS also provides functions for features such as Active Memory sharing or suspend and resume. When using the PowerVM Standard Edition and PowerVM Enterprise Edition, dual Virtual I/O Servers need to be deployed to provide maximum availability for client partitions when performing VIOS maintenance.

Figure 3-35 shows a VIOS configuration. This diagram only shows a small subset of the capabilities to illustrate the basic concept of how the VIOS works. As you can see, the physical resources such as the physical Ethernet adapter and the physical disk adapter are accessed by the client partition using virtual I/O devices.

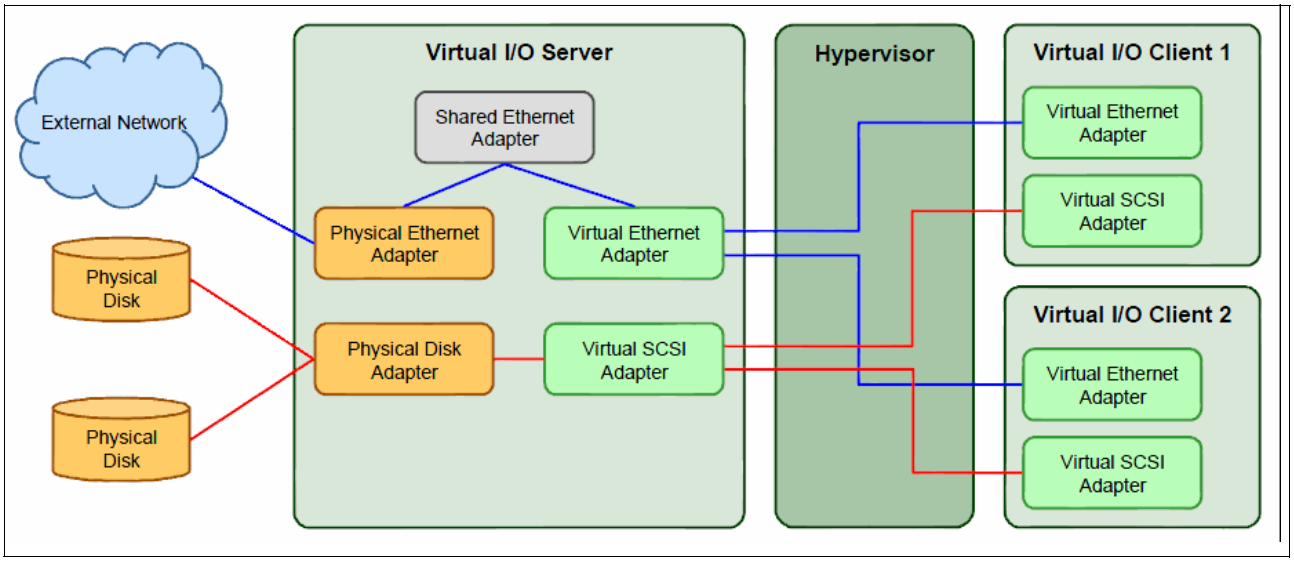


Figure 3-35 Virtual I/O Server configuration

For more information about VIOS, see the following IBM Redbooks publications:

- *IBM PowerVM Getting Started Guide*, REDP-4815

  http://www.redbooks.ibm.com/abstracts/redp4815.html
- *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940

  http://www.redbooks.ibm.com/abstracts/sg247940.html

3.4.7 Power Systems network virtualization

Power Systems and PowerVM architecture recommends and uses network virtualization from a VIOS and connectivity/adapter perspective. Following is a summary of virtual device requirements from a PowerVM technology perspective:

- Virtual devices that do not require a Virtual I/O Server:
  - Virtual Ethernet:
  - Partitions on the same system communicate without using physical Ethernet adapters.
  - Virtual Ethernet adapters are assigned to LPARs.
  - POWER Hypervisor enables internal virtual switch and virtual Ethernet protocol.

- Virtual devices that do require a Virtual I/O Server:
  - SEA (also requires a physical Ethernet adapter)
  - Layer two bridge function connects internal virtual Ethernet with external physical network.
  - VSCSI (also requires a physical disk attachment adapter):
  - Virtual I/O Server partition uses files, logical volumes, whole physical disks, or physical optical devices to provide the backing storage for VSCSI devices presented to client partitions.
  - Client sees an hdisk# or cd#.
- Hardware-assisted virtual device:
  - IVE adapter
  - Partitions can communicate internally and with external physical network without going through virtual
- Ethernet and SEA:
  - This is equivalent to a physical switch, direct communication from LPAR with no hypervisor interaction.
  - This is not supported for Live Partition Mobility.

VLAN: Tagged and untagged ports

PowerVM uses a Layer-2 implementation of the Open Systems Interconnection (OSI) model to create logical networks within switches. A single switch can support multiple VLANs, and VLANs can span across switches. The PowerVM hypervisor implements VLAN based on IEEE 802.1Q VLAN implementation, and can support 4094 VLANs per server. The maximum transmission unit (MTU) size is 65394.

A port on a VLAN-capable switch has a default port virtual LAN ID (PVID) that indicates the default VLAN the port belongs to. The switch adds the PVID tag to untagged packets that are received by that port from the host. In addition to a PVID, a port may belong to additional VLANs and have those VLAN IDs assigned to it that indicate the additional VLANs the port belongs to. A switch port with a PVID only is called an untagged port. Untagged ports are used to connect VLAN-unaware hosts.

A port with a PVID and additional VIDs is called a *tagged port*. Tagged ports are used to connect VLAN-aware hosts. VLAN-aware means that the host is IEEE 802.1Q compatible and can manipulate VLAN tags, and thus can interpret them, add them, and remove them from Ethernet frames.

Figure 3-36 provides an overview of a VLAN from a tagged and untagged port view.

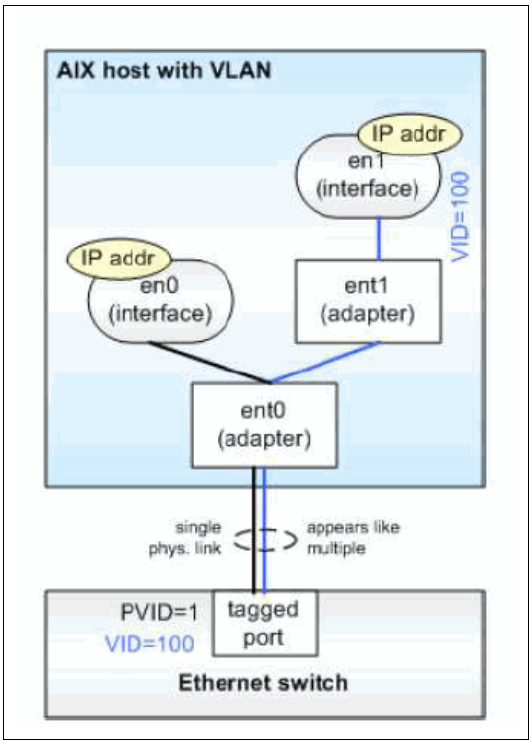


Figure 3-36 VLAN tagged and untagged ports

Virtual Ethernet

Virtual Ethernet provides inter-partition communication facility without the need for a physical adapter. Configuring a virtual Ethernet adapter results in configuring a port of virtual switch of the hypervisor. The hypervisor copies packets between the LPARs memory. Up to 256 virtual Ethernet adapters per partition are supported. One virtual Ethernet adapter can provide access to 21 VLANs (including PVID).

Power hypervisor Ethernet switch

The POWER Hypervisor is included with all POWER5, POWER6, POWER7, and POWER8 processor-based systems and has the following characteristics:

- Implements a software Ethernet switch and virtual Ethernet protocol
- Is consistent with IEEE 802.1Q
- Supports up to 4094 networks (VLAN IDs)
- Uses PVID tag for untagged packets
- Copies packets between partitions (memory to memory)

Multiple VSwitches can be defined in IBM Power Systems within these considerations:

- One VSwitch (VSwitch0) is defined by default.
- VSwitches are defined at the system level (hypervisor).
- VLANs cannot communicate through different VSwitches.
- VSwitch can be created from the Hardware Management Console (HMC) GUI.
- By creating a virtual Ethernet adapter and selecting VSwitch.

Shared Ethernet Adapter

The Shared Ethernet Adapter (SEA) connects virtual network to physical network through the VIOS. The SEA provides a Layer 2 bridge functions to transport network traffic between virtual and physical network. The configuration changes in the SEA do not require changes in the client.

SEA redundancy works in active/passive configuration, and different priorities should be set for Virtual Ethernet trunk adapters. A lower priority value indicates higher priority. A control channel (separate VLAN) needs to be created to exchange heartbeat packets. The control channel adapter should be placed on a dedicated VLAN that is not used for anything else.

Also, the IP address should be set to ping on the SEA.

Figure 3-37 provides an overview of the SEA in a failover configuration for high availability.

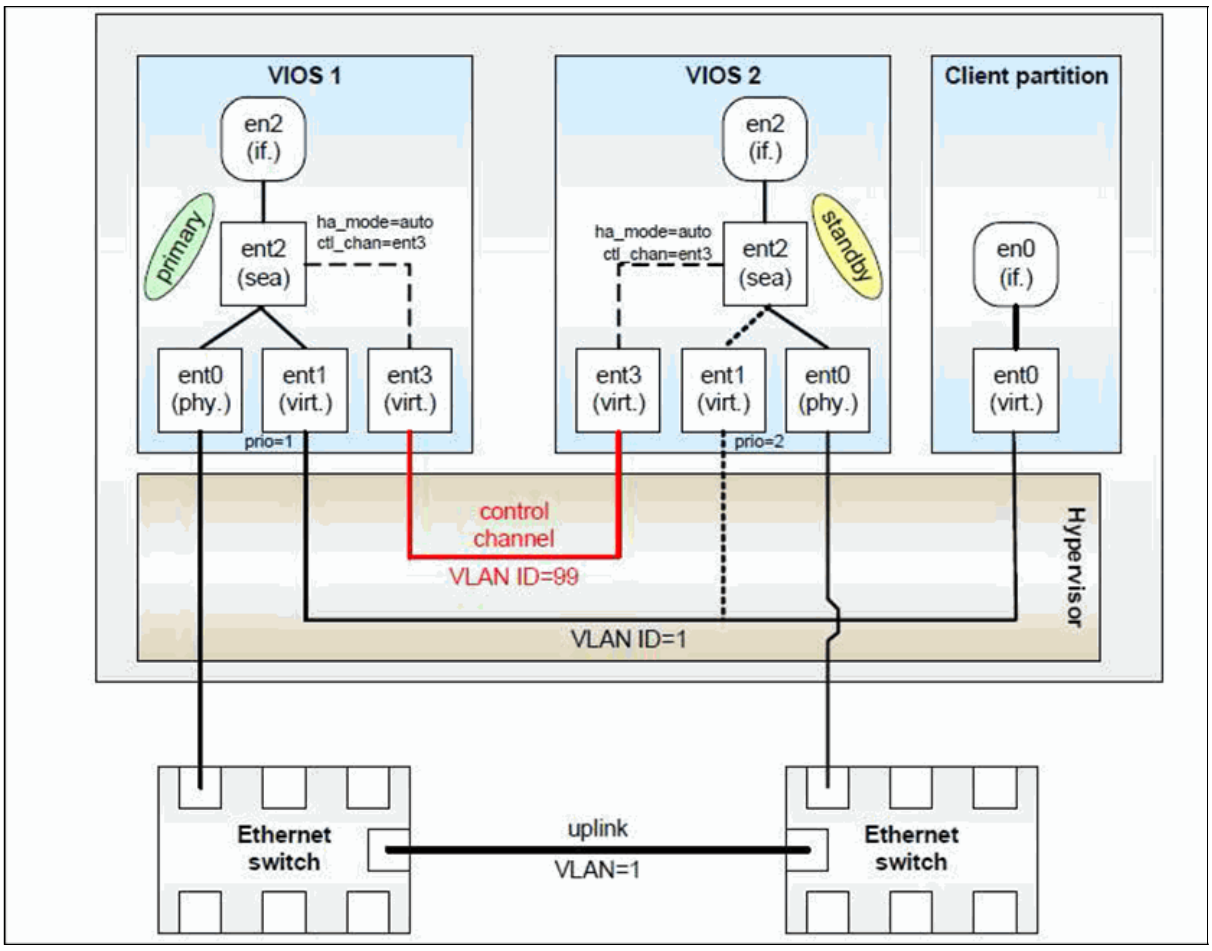


Figure 3-37 Shared Ethernet Adapter (SEA) failover configuration

Integrated virtual Ethernet

An integrated virtual Ethernet, or IVE, offers an alternative to the use of virtual Ethernet or the Shared Ethernet Adapter service of the VIOS. It is supported only on platforms that are entry, midrange platforms that are POWER6 or above.

IVE refers to the collective name referring to a number of technologies including Hardware - Host Ethernet Adapter (HEA), software components, and Power hypervisor functions. IVE offers greater throughput and lower latency than PCIe or PCI-X bus connected Ethernet adapters as it is connected to the GX+ bus of the Power System server. IVE also provides special hardware features that enable logical Ethernet adapters which can communicate directly with logical partitions running on the system.

Figure 3-38 provides a pictorial comparison between a SEA and IVE design. This is interesting to see the relationship to network integration through the hypervisor and without.

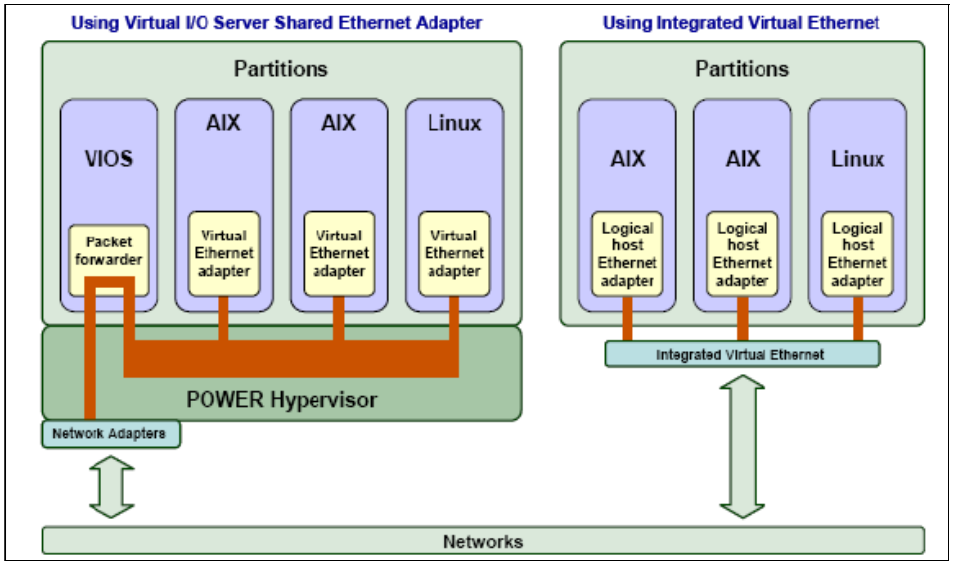


Figure 3-38 VIOS SEA versus IVE design

Figure 3-39 on page 126 provides a listed functional comparison between IVE and VIOS (SEA), which is helpful for determining the best fit solution based on your requirements. For the solution described in this book, we have implemented VIOS (SEA).

Virtualized function	IVE	VIOS
Basic functions		
Requires activation of the Advanced POWER Virtualization feature code		✓
Virtualizes Ethernet resources over a single server	✓	✓
Provides hardware acceleration	✓	
Managed by the HMC as a hardware device	✓	
Scalable and flexible for all virtualization requirements		✓
Operating system based - runs in an LPAR		✓
Enables advanced server functions, such as Live Partition Mobility		✓
Strategic solution for IBM Systems client requirements	✓	✓
Network functions		
Can be used as an SEA	✓	✓
Provides scalable threading for multiple-core systems	✓	
Provides 10 Gbps wire speed performance with very low latency	✓	
Provides secure network connections between LPARs	✓	✓
Supports industry standard networking attributes, such as VLAN, MTU, and so on	✓	✓
Does not requires a PCI-x or PCIe slot for external communication	✓	

Figure 3-39 VIOS SEA versus IVE - Functional comparison

For more information about network virtualization, the following resources are helpful:

- PowerVM Information Roadmap
  http://publib.boulder.ibm.com/eserver/roadmap_powervm.html
- *IBM PowerVM Virtualization Managing and Monitoring*, SG24-7590
  http://www.redbooks.ibm.com/abstracts/sg247590.html
- *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940
  http://www.redbooks.ibm.com/abstracts/sg247940.html
- *Integrated Virtual Ethernet Adapter Technical Overview and Introduction*, REDP-4340
  http://www.redbooks.ibm.com/abstracts/redp4340.html

3.4.8 System z virtual machine, z/VM

> "Everything should be made as simple as possible, but not simpler." — Albert Einstein

z/VM is the current version in the IBM VM family of virtual machine operating systems. z/VM was first released in October 2000[6] and remains in active use and development as of 2015. It is directly based on technology and concepts dating back to the 1960s, with IBM CP/CMS on the IBM System/360-67. z/VM 6.3 runs on IBM System z family of computers. It can be used to support large numbers (thousands) of Linux virtual machines.

[6] See http://www.vm.ibm.com for recent release history.

On July 23, 2013, IBM released z/VM Version 6.3 which requires IBM z/Architecture® 2 (ARCHLVL 3), which was implemented in IBM System z10® models.

The virtual machine (VM) has long been recognized as a robust computing platform, spanning the family of IBM mainframe servers.

Note: System z in not within the scope of this SDE solution, nor is it currently deployed and integrated in our lab environment for our solution architecture. Integration is also available through IBM Cloud Manager (formerly called Cloud Manager with OpenStack) version 4.2, which enables basis infrastructure cloud services. Also, zCloud integration to the SDE solution will be further provided in an update to this book or a new Redbooks publication for System z. However, because we are describing other hypervisors and virtual machines, and the key principle for SDE with OpenStack is to work and integrate with a heterogeneous platform environment, we are providing z/VM details and benefits as part of an SDE solution and a hybrid or private cloud.

z/VM provides a highly secure and scalable enterprise cloud infrastructure and an environment for efficiently running multiple diverse critical applications on System z with support for more virtual servers than any other platform in a single footprint. These virtual servers can run Linux or IBM z/OS operating systems.

Linux on System z offers a uniquely powerful, high-performance solution. Supreme efficiency for optimized workload deployment, innovative system management with IBM Wave for z/VM, and the legendary performance and scalability of z/VM come together to form a foundation for your Linux applications that is simple, cost-effective, efficient, and secure.

z/VM offers a base for clients who want to use IBM virtualization technology on one of theI BM System z family servers. With virtualization technology, clients can easily create many virtual machines that consist of a virtualized processor, communications, storage, networking, and I/O resources.

The z/VM hypervisor helps clients extend the business value of mainframe technology across the enterprise by integrating applications and data while providing exceptional levels of availability, security, and operational ease. z/VM virtualization technology enables the capability for clients to run hundreds to thousands of Linux servers on a single mainframe, running with other System z operating systems, such as z/OS, or as a large-scale, Linux-only enterprise server solution. z/VM can also help to improve productivity by hosting non-Linux workloads, such as z/OS, IBM z/VSE®, and IBM z/Transaction Processing Facility (z/TPF).

Enhancements to z/VM 6.3 now support 1 TB of real memory to help clients keep pace with increasing business demands while continuing to allow for the highest levels of shared resource use. This support may increase server consolidation ratios. z/VM 6.3 is the premier virtualization platform, supporting more virtual servers than any other in a single footprint. It provides industry leading capabilities to efficiently scale both horizontally and vertically, enabling clients to support growing business demands while lowering the cost of service delivery.

In addition to more memory, we have also addressed the increasing workload demands for processor cycles and for quicker access to memory. z/VM HiperDispatch does that by strengthening the affinity between where work is dispatched and where the data used by that work exists, thereby avoiding delays to retrieve that data. z/VM HiperDispatch provides improved throughput depending on workload characteristics, with memory-intensive workloads running on large numbers (16 to 32) of physical processors most likely to achieve the highest performance gains.

Virtualization is still very hot in the IT industry. The IBM mainframe, z/VM, and its predecessors have been doing virtualization for five decades. Today, it can be argued that the mainframe is the most functionally rich virtualization platform. When Linux came to the IBM mainframe in 2000, it was a natural fit to run under z/VM. You can run many tens or even low hundreds of Linux virtual servers on the same System z logical partition (LPAR) under z/VM.

With a z/VM and Linux infrastructure, you can reduce the time between deciding on the acquisition of new servers and then implementing them because new servers can be deployed in a matter of minutes. This powerful build and clone capability can enable you to launch new products and services without the exhaustive planning, purchasing, installing, and configuring new hardware and software that can be associated with conventional discrete hardware servers. Development groups who need test environments built and rebuilt rapidly to enable them to efficiently deliver their projects, handling change management in the process, can also benefit from this unique advantage.

The following capabilities are some of the best strengths of the mainframe and z/VM:

- Their virtualization capabilities are more mature and robust than any other hardware and hypervisor combination.
- The z/VM virtual switch makes networking Linux much simpler.
- Full volume backup of systems enables complete disaster recovery when another data center is available.
- z/VM is one of the easiest operating systems to customize at the base installation level.
- There is only a relatively small number of configuration files. Properly set up, z/VM can run for months with little maintenance or administration required.

Much functions were added to z/VM since version 5.2. The following summary lists the functions for v6.3.

Enhancements introduced in z/VM 6.3

z/VM 6.3, which became generally available in late July 2013, extends the mainframe virtualization platform to help you reshape and derive more value from your systems. It offers the following benefits:

- Improved economies of scale with z/VM support for 1 TB of real memory
  - Better performance for larger virtual machines
    - Quadruples memory scalability while continuing to maintain greater than 90% resource use
  - Additional vertical scalability to help reduce LPAR sprawl
    - Considerably more virtual machines can be consolidated into a single LPAR, depending on workload characteristics
  - Reduced administrative expense through managing a smaller number of large-capacity z/VM host servers
- Improved performance with z/VM HiperDispatch
- More efficient use of CPU hardware resources for dispatched work
- IBM contributions to the OpenStack project to enable z/VM 6.3 to be the first System z operating environment to be managed by these open cloud architecture-based interfaces
- Simplified migration to z/VM V6.3 with upgrade in place, which reduces the effect of an upgrade on active workloads

- Highly secure industry-standard support that is required for banking and financial-industry applications
- Support for the new IBM zEC12 and zBC12 servers

Single image system design

With the introduction of z/VM 6.2 in December 2011, the architecture of Linux solutions on System z has changed dramatically. Cross Systems Extensions (CSE) allowed for a type of clustering environment for Linux on System z before z/VM 6.2. However, CSE was not widely used nor was the architecture completely enabled for clusters. z/VM 6.2, with its single system image (SSI) and live guest relocation (LGR) functions has changed all of this. A z/VM system is no longer the most important “object” in the hierarchy.

With z/VM 6.1 and earlier, the system identifier of each z/VM system was the most important. With z/VM 6.2 and later, the SSI name is the highest level identifier.

Figure 3-40 shows a block diagram of a four-member SSI with default volume labels. The recommend scenario, a four member cluster with two members on two different central electronic complexes (CECs), is shown. In such a cluster, there are four z/VM systems and four system identifiers. However, there is only one SSI name.

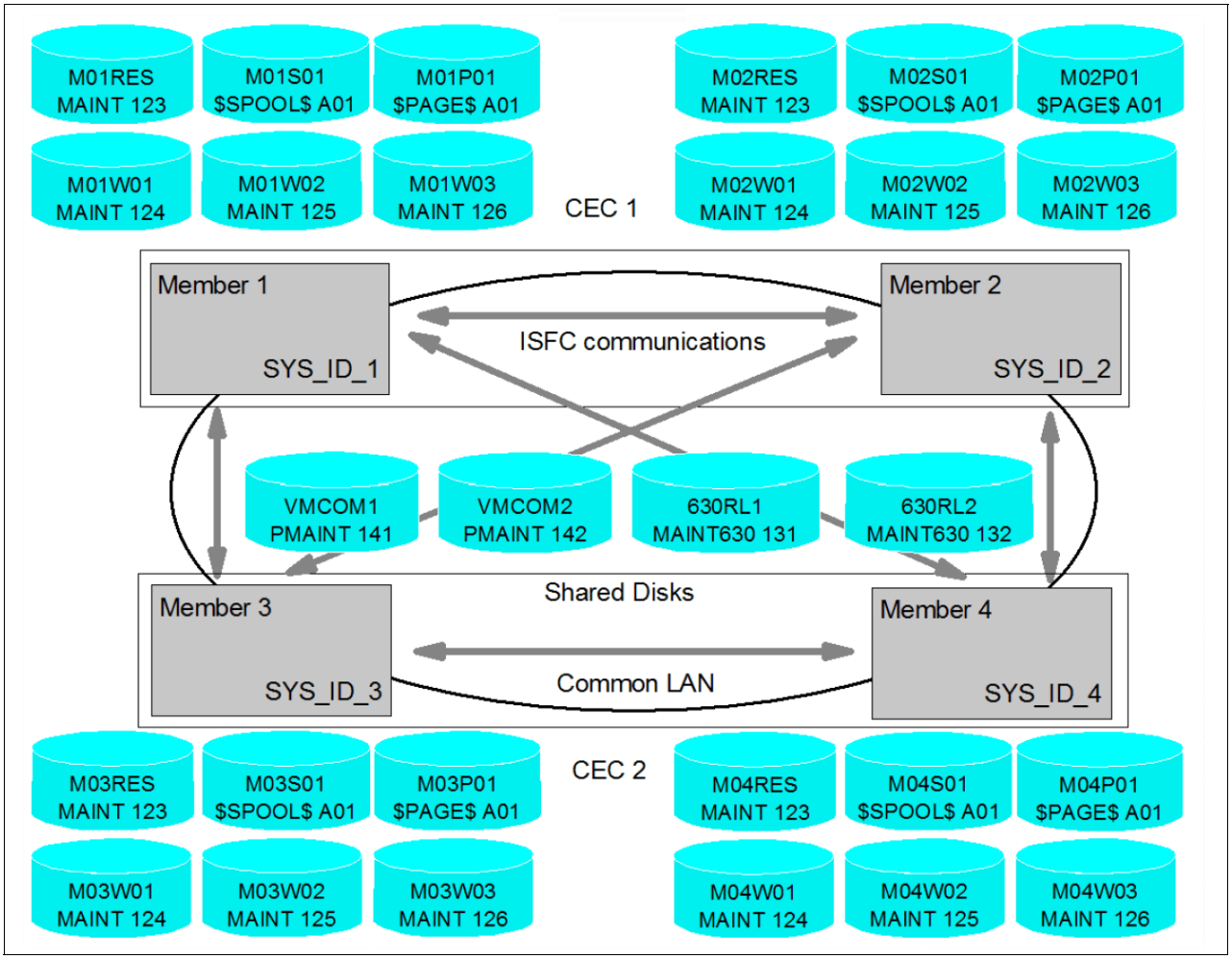


Figure 3-40 z/VM 6.3 single system image design block diagram

Infrastructure design

To install and configure z/VM, and to install, configure, and clone Linux or *provision virtual servers*, there must be a certain infrastructure design in place. A System z server with associated resources and the z/VM operating system define much of this infrastructure. Figure 3-41 shows a block diagram of a z114 with z/VM 6.2 on two LPARs. z/VM comes with many virtual machines predefined. The most important six IDs are shown in the z/VM LPAR above the dashed line. Below the dashed line, you see the virtual machines.

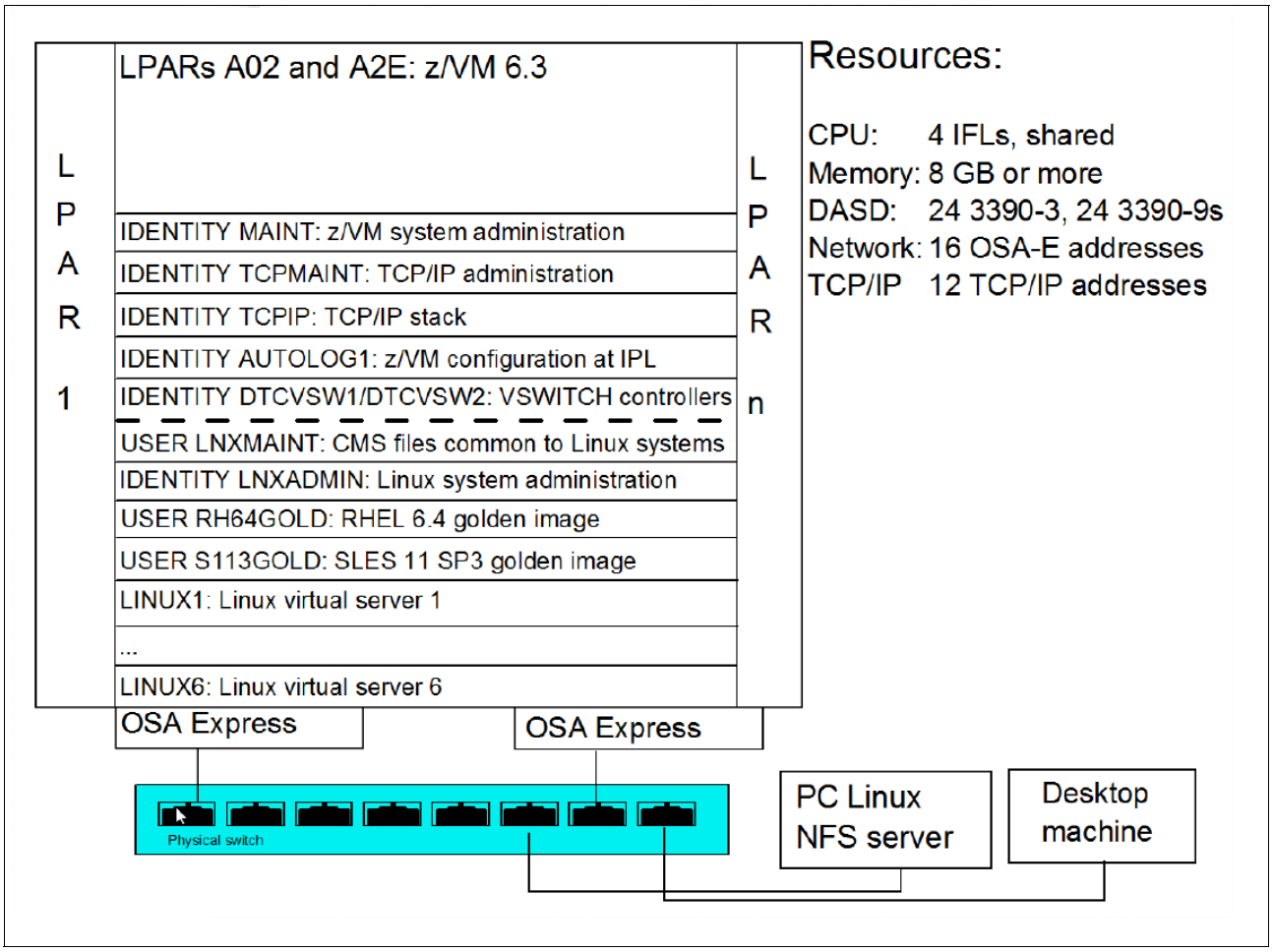


Figure 3-41 System infrastructure and z/VM virtual machines

The virtual machines that are described in this book have the following functions:

LNXMAINT	A virtual machine on which to store CMS files to be used by Linux
LNXADMIN	The Linux system administration server that exports install trees, clones systems, and performs other system administrative functions. This is an *identity* that can be logged on to all SSI members at the same time
RH64GOLD	A virtual machine that contains the RHEL 6.4 *golden image*
S113GOLD	A virtual machine that contains the SLES 11 SP3 *golden image*
LINUX1 – LINUX6	Six sample *worker* virtual machines

In addition to the two LPARs, two other machines are shown:

Linux PC NFS server A Linux system used for the first two installations of each distribution.

Desktop machine A workstation from where all of the work is done.

For more information about IBM Virtualization for z/VM 6.3, Red Hat Enterprise Linus (RHEL) V6.4 and SLES 11 SP3, see the following publications:

- *The Virtualization Cookbook for IBM z/VM 6.3, RHEL 6.4, and SLES 11 SP3*, SG24-8147

  http://www.redbooks.ibm.com/abstracts/sg248147.html
- *Set up Linux on IBM System z for Production*, SG24-8137

  http://www.redbooks.ibm.com/abstracts/sg248137.html
- The “Other z/VM Publications” library:

  http://www.vm.ibm.com/pubs/

For more information about System z and cloud computing, see the following Redbooks publications:

- *Deploying a Cloud on IBM System z*, REDP-4711

  http://www.redbooks.ibm.com/abstracts/redp4711.html
- *Maximize IBM System z ROI with Open Cloud Management and Policy-based Optimization*, REDP-5008

  http://www.redbooks.ibm.com/abstracts/redp5008.html
- *Cloud Workloads on the Mainframe*, REDP-5108

  http://www.redbooks.ibm.com/abstracts/redp5108.html
- *Getting Started With IBM Enterprise Cloud System*, SG24-8219

  http://www.redbooks.ibm.com/redpieces/abstracts/sg248219.html

VMware

VMware is out of scope of this SDE solution, but OpenStack does integrate with VMware in terms of a Compute node (Manage -to) and is supported on VMware through IBM Cloud Manager with OpenStack and vCenter. Guest operating systems that are covered are Microsoft Windows, Linux SUSE Linux, and RedHat Linux.

OpenStack Compute supports the VMware vSphere product family and enables access to advanced features, such as vMotion, High Availability, and Dynamic Resource Scheduling (DRS).

For more information, see the VMware vSphere page on the OpenStack.org website:

http://docs.openstack.org/juno/config-reference/content/vmware.html

This section has provided an overview of the hypervisors on many popular platform choices in the context to OpenStack. For more information about OpenStack architecture design considerations and hypervisor integration, see the following resources on the OpenStack.org website:

- OpenStack Architecture Design Guide

  http://docs.openstack.org/arch-design/content/
- HypervisorSupportMatrix

  https://wiki.openstack.org/wiki/HypervisorSupportMatrix

3.5 Storage GPFS, GPFS Storage Server, XIV, V7K, DS8K

This section describes the storage environment used and the options to deploy such configurations.

3.5.1 IBM Spectrum Scale

Today's never-ending data growth is challenging traditional storage and data management solutions. These outdated systems are expensive to administer and scale, in addition to limiting data access, performance and reliability that today's data-intensive computing environments require; particularly when data is accessed on a global scale. Application performance is affected by data access bottlenecks that delay schedules and waste expensive resources, and data growth is threatening exponential CAPEX and OPEX expansion that must be arrested.

IBM Spectrum Storage™ (formerly IBM General Parallel File System, or GPFS) is a high-performance clustered file system developed by IBM. It can be deployed in shared-disk or shared-nothing distributed parallel modes. It is used by many of the world's largest commercial companies, as well as some of the supercomputers on the Top 500 List.[7] For example, GPFS was the file system of the ASC Purple Supercomputer,[8] which was composed of more than 12,000 processors and has 2 petabytes of total disk storage spanning more than 11,000 disks. GPFS also is the basis for VTS (Virtual Tape server), Scale out File Services (SoFS) and IBM Scale Out Network Attached Storage (SONAS). The main differentiator in GPFS is that it is not a clustered file system but a *parallel* file system. This means that GPFS can scale almost infinitely. Today, installations with more than 2 PB in a single instance exist, running more than 11.000 disk drives.

In common with typical cluster file systems, GPFS provides concurrent high-speed file access to applications executing on multiple nodes of clusters. It can be used with IBM AIX clusters, Linux clusters, on Microsoft Windows Server, or a heterogeneous cluster of AIX, Linux, and Windows nodes. In addition to providing file system storage capabilities, GPFS provides tools for management and administration of the GPFS cluster and allows for shared access to file systems from remote GPFS clusters.

GPFS has been available on IBM AIX since 1998, on Linux since 2001, and on Microsoft Windows Server since 2008, and it is offered as part of the IBM System Cluster 1350. Since release 3.5, GPFS offers Active File Management to enable asynchronous access and control of local and remote files, which enables global file collaboration.

3.5.2 IBM software-defined storage vision within SDE

You might be wondering: Why software-defined storage (SDS)? Along with the same principles and existing client environments that have heterogeneous "islands of compute," there are also heterogeneous "islands of storage," based on varying applications and varying workload characteristics and demands, such as email, application development, HPC, analytics, grid, archive, and OLTP, to name a few.

---

[7] Schmuck, Frank and Roger Haskin (January 2002). "GPFS: A Shared-Disk File System for Large Computing Clusters" (PDF). "Proceedings of the FAST'02 Conference on File and Storage Technologies." Monterey, California, USA: USENIX. pp. 231–244. ISBN 1-880446-03-0. Retrieved 2008-01-18. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.7147

[8] For more information about GPFS and Spectrum Scale (previously called Elastic Storage), see this web page: http://www.ibm.com/systems/storage/spectrum/scale/

The principle and goal for SDS within SDE is to consolidate these islands of storage into a unified shared storage pool that is “elastic” (also known as a “data lake”) with the following capabilities:

- It can scale quickly and on demand.
- It can support diverse workloads.
- It uses storage tiering to optimize the technology, tailored to the workload characteristics.
- It can offer a hybrid deployment (reuse existing infrastructure, and add new infrastructure).

Spectrum Scale version 4.1 is software-defined storage for cloud, big data, and analytics introduces enhanced security, flash accelerated performance, and improved usability for the world’s most demanding data intensive enterprises. Spectrum Scale includes these features:

- *Enhanced security.* Native encryption and secure erase, NIST SP 800-131A encryption compliance.
- *Increased performance.* Server-side Spectrum Scale Flash caches increase IO performance up to 6X.
- *Improved usability.* Data migration, Active File Management (AFM), File Placement Optimizer (FPO), and backup/restore enhancements. Also, reliability, availability, and serviceability enhancements.

Spectrum Scale provides world-class storage management delivered in software and used on any hardware, with extreme scalability flash accelerated performance, and automatic policy-based storage tiering from flash through disk to tape, reducing storage costs 90% while improving security and management efficiency in cloud, big data and analytics environments. It removes data-related bottlenecks by providing parallel access to data, eliminating single filer choke points or hot spots. Spectrum Scale also simplifies data management at scale by providing a single namespace that can be scaled simply, quickly, and infinitely by simply adding more scale-out resources, which eliminates “filer sprawl” and its associated headache by providing the following functions:

- Parallel access from all clients to all storage: end-to-end and high performance access to small and large files.
- High-bandwidth access to single file:
  - Wide striping
  - Large block size
  - Parallel access to files from multiple nodes
  - Efficient deep pre-fetching: read ahead, write behind
  - Highly multithreaded daemon
- Provides high file system input/operations per second (IOPS):
  - Distributed metadata architecture
  - Option to configure dedicated fast disks to store file system metadata
  - Data for small files stored within the inode, thereby reducing disk I/O for small file workloads
- Integrated tiered storage:
  - Use SSD or high IOPS storage to store metadata or hot data for better IOPS
  - Use SSDs in client nodes as GPFS read cache: Local Read Only Cache (LROC)

Spectrum Scale empowers geographically distributed organizations by expanding that single global namespace, literally globally by placing critical data close to everyone and everything that needs it, no matter where they are in the world. Speeding data access to stakeholders around the world accelerates schedules and improves productivity.

Spectrum Scale simplifies data management ease at scale and data lifecycle automation bridges the gaping data growth - budget chasm, bringing storage costs into line. When integrated with IBM Tivoli Storage Manager or IBM Linear Tape File System™ (LTFS), elastic storage can manage the full data lifecycle, at substantially lower cost, through policy-driven automation and tiered storage management. Key features:

- Seamless capacity and performance scaling
- Rolling upgrades of individual nodes in the cluster while the file system remains online
- Quotas enable the administrator to manage file system use by users and groups
- SNMP interface allows monitoring by network management applications
- GPFS supports POSIX and NFS V4 access control lists (ACLs)

Spectrum Scale provides unsurpassed end-to-end data availability, reliability, and integrity with unique technologies including Spectrum Scale Native RAID (GNR) which uses advanced erasure coding to avoid the painful multiday rebuild times becoming common with today's multiterabyte drives, in addition to being able to withstand multiple device failures rather than the one or two failures conventional systems are capable of withstanding. Spectrum Scale also provides advance checksum protection to ensure data integrity from the disk platter all the way to the compute client.

Spectrum Scale provides fast data access and simple, cost effective data management and it offers the following benefits:

- Enhanced security. Protect data on disk from security breaches, unauthorized access, or being lost, stolen or improperly discarded with encryption of data at rest and enable HIPAA, Sarbanes-Oxley, EU, and various national data privacy laws compliance.
- Extreme performance. Server-side Spectrum Scale flash caches speed IO performance up to 6X, benefiting application performance, while still enjoying all the manageability benefits of shared storage.
- More cost-effective. Saves acquisition costs by using standard servers and storage rather than expensive, special purpose hardware.
- Limitless elastic data scaling. Scale out with relatively inexpensive standard hardware, while maintaining world-class storage management.
- Increased resource and operational efficiency. Pools redundant isolated resources and optimizing use.
- Greater IT agility. Be able to quickly react, provision, and redeploy resources in response to new requirements.
- Intelligent resource use and automated management. Automated, policy-driven management of storage reduces storage costs 90% and improves operational efficiencies.
- Geographically distributed workflows. Places critical data close to everyone and everything that needs it, accelerating schedules and time to market.

Figure 3-42, provides an illustration and description of the IBM vision of the integration of new and traditional models of storage data architecture.

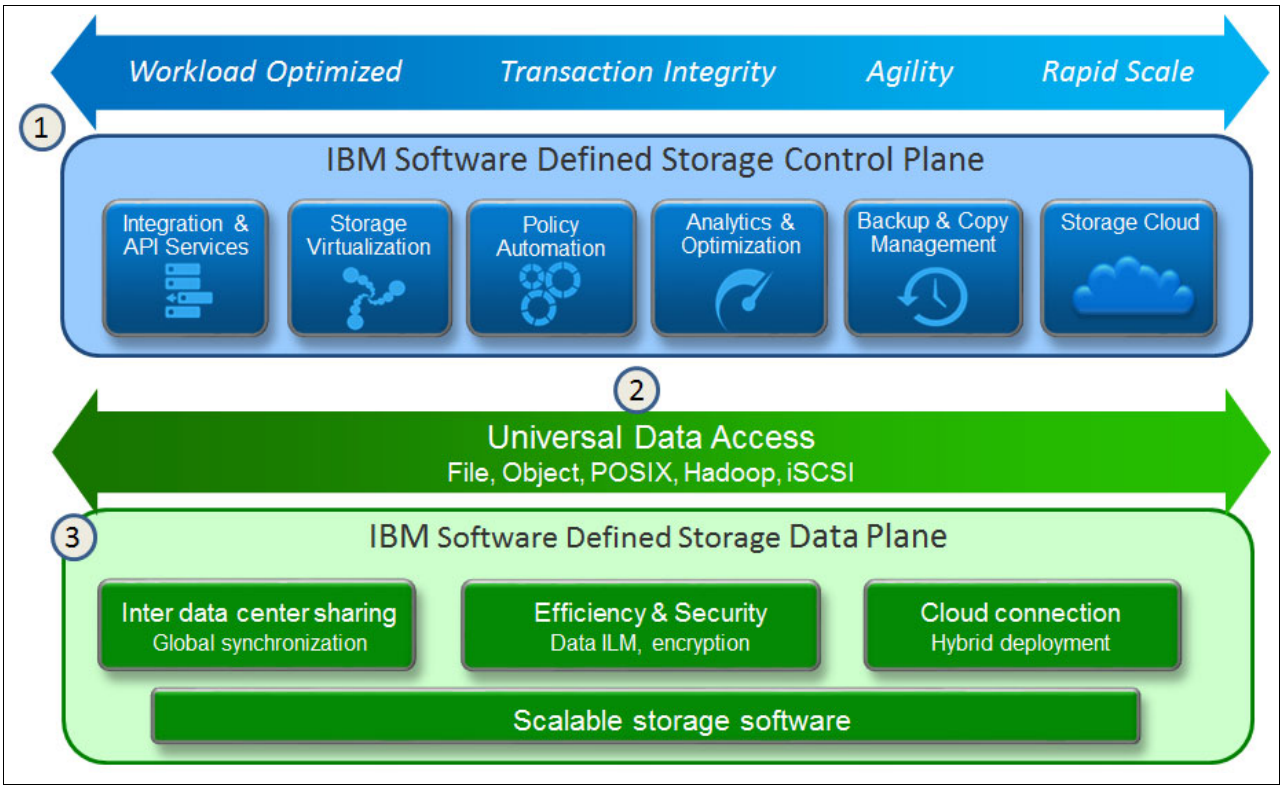


Figure 3-42 IBM Vision integrating new and traditional models

Once again, the business in many environments today demand more of an application aware solution architecture, to increase time to market for their products and enable them to deploy applications faster than ever before. However, this application awareness requirement applies throughout the entire solution architecture, including the storage “layers” as shown in the diagram in Figure 3-42:

1. The goals are to make the application function and work at its best performance and securely, as aligned with the storage infrastructure to get the most that storage can provide to the application architecture in terms of transaction integrity, agility, and rapid scale.

   Through the various control methods as shown in #1, above, storage is no longer a disparate black box. Access, virtualized storage, analytics, back up and copy management and storage cloud off-premises access versus on-premises storage is set by policy automation parameters and settings in the IBM Software Defined Storage Control Plane. The result is storage optimized for the specific application workload rather than “one size fits all.”

2. The objective of the Universal Data Access layer is to provide access to the heterogeneous storage environment. Just as the key principle in the SDE Compute layer is to provide access to heterogeneous virtualization and compute platforms, based on workload demands, the principle here is very similar. Universal data access is provided according to the policies set in the storage control pane.

Access can be provided through the following types of storage protocols:

- File
- Object
- Portable Operating System Interface[9] (POSIX) Hadoop

Note: POSIX defines the application programming interface (API), along with command-line shells and utility interfaces, for software compatibility with variants of UNIX and other operating systems.[a]

a. "POSIX 1003.1 FAQ Version 1.12". 2 February 2006. Retrieved 16 July 2006. "POSIX." Standards. IEEE.

For more information about POSIX, see:

http://www.opengroup.org/austin/papers/posix_faq.html

- Internet Small Computer System Interface (iSCSI)

Note: In computing, iSCSI is an acronym for Internet Small Computer System Interface, an Internet Protocol (IP)-based storage networking standard for linking data storage facilities.

By carrying SCSI commands over IP networks, iSCSI is used to facilitate data transfers over intranets and to manage storage over long distances. iSCSI can be used to transmit data over local area networks (LANs), wide area networks (WANs), or the Internet and can enable location-independent data storage and retrieval.

The protocol allows clients (called initiators) to send SCSI commands (CDBs) to SCSI storage devices (targets) on remote servers. It is a storage area network (SAN) protocol, which enables organizations to consolidate storage into data center storage arrays while providing hosts (such as database and web servers) with the illusion of locally attached disks.[a]

Unlike traditional Fibre Channel, which requires special-purpose cabling, iSCSI can be run over long distances using existing network infrastructure. [b]iSCSI was pioneered by IBM and Cisco in 1998 and submitted as draft standard in March 2000.[c]

a. Rouse, Margaret (May 2011). "iSCSI (Internet Small Computer System Interface)." SearchStorage. Retrieved 3 November 2012.
b. "ISCSI SAN: Key Benefits, Solutions and Top Providers Of Storage Area Networking." Trident Network Solutions. Retrieve
c. "iSCSI proof-of-concept at IBM Research Haifa." IBM. Retrieved 13 September 2013.

3. The IBM SDS Data Plane provides the foundational aspects needed for storage in today's cross-region or global data center environments. As determined by policy-driven automation in the Control Plane, the Data Plane provides:
   - Sharing between data centers through global synchronization.
   - Data integrity, efficiency, and security to provide encryption, and also assist with Data ILM through tiering based on workload types and storage tiering.
   - Support for on-premises to off-premises hybrid clouds, securely, safely and efficiently, where there might be more than one type of storage type involved.

[9] POSIX.1 FAQ." The Open Group. October 5, 2011

The result is a highly scalable solution for heterogeneous storage that helps eliminate typical inherent bottlenecks that clients often experience in NAS and filers that are very structured and rigid in terms of their architecture. The goal with Spectrum Scale is to provide a secure, reliable, cloud storage tier enabled by the underlying GPFS technology and architecture.

So why do this? Why change what is in aleady place in an environment? The motivation here is to start getting more ROI on storage investment through information lifecycle management (ILM), through managing data growth by placing data in the right tier at the right time according to its value, while being available under one common name space at any time using the economy of scale of the cloud.

Information lifecycle management (ILM) is not a product, but instead consists of the strategy, processes and technology (through SDE, SDS Spectrum Scale) to effectively manage information through the phases of its lifecycle on the most efficient and cost-effective IT infrastructure. the IBM approach to ILM spans a continuum that can deliver business value in a variety of ways, aligned with business objectives. ILM is not a quick file system management solution. It should be implemented in manageable stages, focusing first on the areas of greatest return.

In IBM client work, six best practice areas have emerged that can lead to more business value from storage and information management:

- *Data rationalization* establishes valid groups of information to apply effective data management standards and policies. Invalid data groups are identified during this process for storage reclamation and consolidation.
- *Tiered storage environment* aligns variable cost hardware types with information classes and classes of service to create a variable cost storage environment.
- *Information management* uses enterprise content management in conjunction with information integration capabilities to intelligently manage information.
- *Storage process, organization, technology and governance* are important components for improving and sustaining information management efficiency.
- *Archiving and information retention* enhances system performance while enabling organizations to better manage risk and streamline regulatory compliance.

GPFS can help you achieve ILM efficiencies through powerful policy-driven, automated tiered storage management. The GPFS ILM toolkit helps you manage sets of files and pools of storage, and also enables you to automate the management of file data.

Using these tools, GPFS can automatically determine where to physically store your data regardless of its placement in the logical directory structure. Storage pools, file sets and user-defined policies provide the ability to match the cost of your storage resources to the value of your data.

Spectrum Scale policy-based ILM tools help you with the following tasks:

- Create storage pools to provide a way to partition a file system's storage into collections of disks or a Redundant Array of Independent Disks with similar properties that are managed together as a group. Spectrum Scale has three types of storage pools:
  - A required system storage pool that you create and manage through Spectrum Scale
  - Optional user storage pools that you create and manage through Spectrum Scale
  - Optional external storage pools that you define with Spectrum Scale policy rules and manage through an external application, such as IBM Tivoli Storage Manager

- Create file sets to provide a way to partition the file system namespace to allow administrative operations at a finer granularity than that of the entire file system. See *File sets*, which follows.
- Create policy rules based on data attributes to determine initial file data placement and manage file data placement throughout the life of the file.

In addition, Spectrum Scale provides the following essentials:

- Storage pools

  Physically, a storage pool is a collection of disks or RAID arrays. Storage pools also enable you to group multiple storage systems within a file system.

- Policies and rules

  Spectrum Scale provides a means to automate the management of files using policies and rules. Properly managing your files enables you to efficiently use and balance your premium and less expensive storage resources.

- File sets

  In most file systems, a file hierarchy is represented as a series of directories that form a tree-like structure. Each directory contains other directories, files, or other file system objects such as symbolic links and hard links. Every file system object has a name associated with it, and is represented in the namespace as a node of the tree.

- Immutability and appendOnly restrictions

  To prevent files from being changed or deleted unexpectedly, Spectrum Scale provides immutability and appendOnly restrictions.

This results in the following value for the client of Spectrum Scale in cloud architecture and technology:

- Migrates files between disk and cloud seamlessly (cloud becomes a Spectrum Scale storage pool)
- Provides full file system backup for disasater recovery (premigrate all files and Spectrum Scale metadata to the cloud)
- Shares data efficiently between remote clusters via Active File Management (AFM)
- Multiple cloud storage tiers are possible (use multiple clouds)
- Removes data-related bottlenecks, which is a necessary prerequisite in any cloud environment
- Simplifies data management at scale, which is necessary for cloud investments
- Empowers global collaboration to gain more value from the cloud investments where and when needed
- Manages the full data lifecycle cost-effectively (this value increases in cloud environments)

The solutions provide end-to-end data availability, reliability, and integrity, which is critical for cloud solutions.

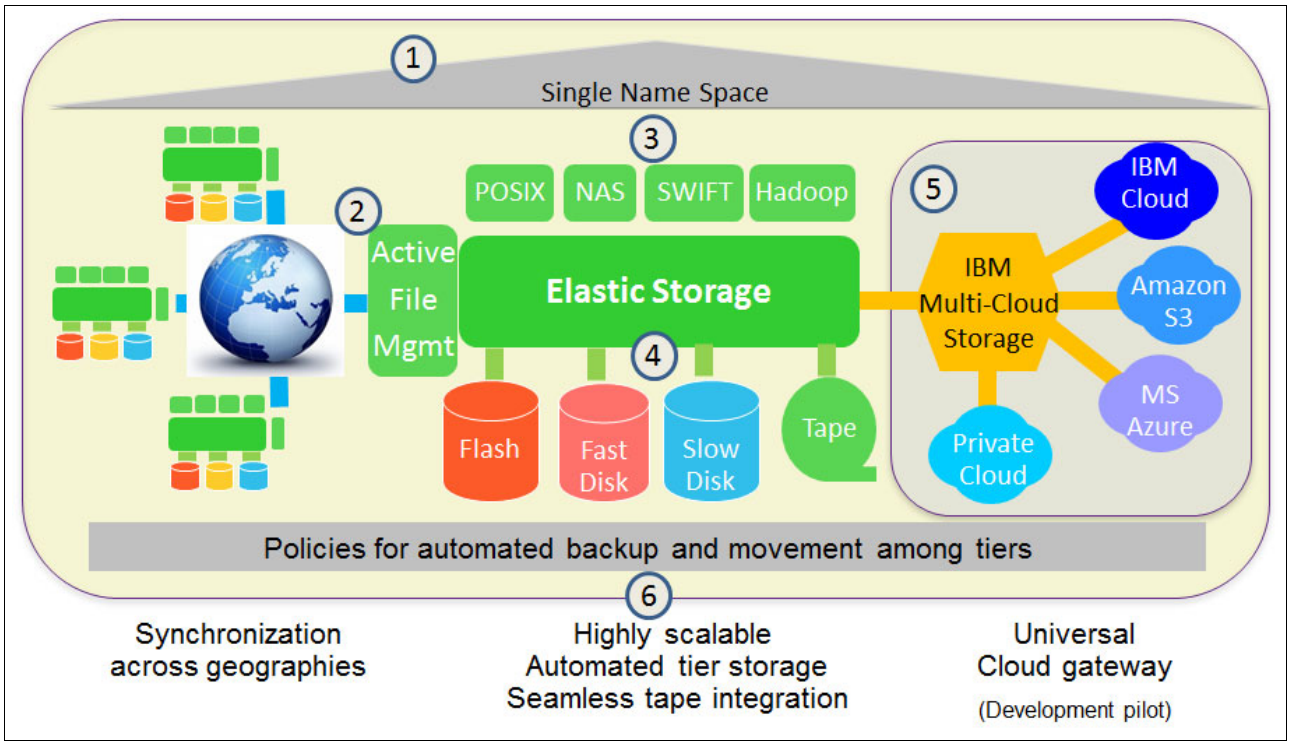


Figure 3-43 Spectrum Scale software vision: End-to-end global access

Figure 3-43 illustrates the Spectrum Scale software vision in the context of cloud technology, architecture, and solutions:

1. The key feature of SDS through Spectrum Scale and alignment point to this SDE solution is through the single name space and Swift (object storage) or Cinder (block storage). The access through the single name space can be shared across geographical boundaries, if needed, or contained within geographical boundaries through policy management.
2. Active File Management is a scalable, high-performance, file system caching layer integrated with the Spectrum Scale cluster file system. AFM enables you to create associations from a local cluster to a remote cluster or storage, and to define the location and flow of file data to automate the management of the data. This enables you to implement a single namespace view across sites around the world.
3. Shared or non-shared data access across locations is provided through a single name space and supported by POSIX, NAS, Switf, Cinder, and Hadoop, which allow the client to work with multiple existing storage types.
4. Files are scanned on the system, and based on policy engine settings (aligned with ILM-based business and application workload requirements and policies), Spectrum Scale dynamic tiering will determine the best placement for storage (Flash, fast disk, slow disk, tape).

Note: Nuance Communications, Inc.® is a leading provider of voice and language solutions for businesses and users around the world. Its technologies, applications and services make the user experience more compelling by transforming the way people interact with devices and systems. Every day, millions of users and thousands of businesses use Nuance's proven applications. Nuance R&D has a large worldwide user community who interacts with big data every day. Multiple petabytes of storage are managed with IBM Spectrum Scale.

Nuance has partnered with IBM and deployed GPFS in an information lifecycle-based approach with storage tiering, and IBM Flash solutions. The highest demanding workload for IOPs in Nuance's environment were deployed to Flash, based on the Policy Engine and automation in the Spectrum Scale solution.

5. Access to heterogeneous cloud storage is provided through the IBM Cloud Gateway (under development) which will provide access to off-premises IBM cloud (for example, SoftLayer), Amazon S3, Microsoft Azure, and on-premises private clouds.
6. Scalability is provided by metadata, and determined and driven by policies for automated backup among tiers and seamless integration to tape.

3.5.3 Spectrum Scale use cases

Figure 3-44 illustrates typical use cases for Spectrum Scale and how it accelerates data-intensive workflows.

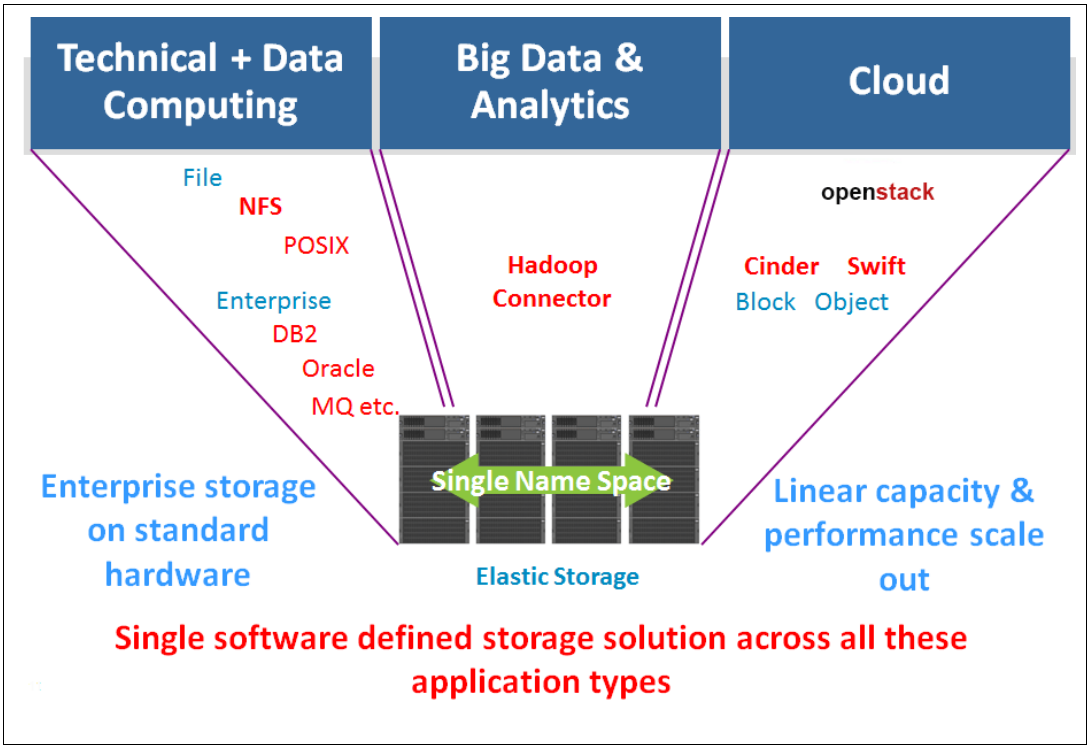


Figure 3-44 Spectrum Scale use cases

Massive amounts of data from high output devices, such as genome sequencers in life sciences to market data in financial services, is ingested in parallel into a cluster of elastic storage servers.

From there, that data is processed in the compute cluster on top, both with conventional computation and Hadoop-based MapReduce techniques. This is all on the same compute and storage infrastructure, which cuts costs and eliminates data movement.

After computation, data is accessed, analyzed, and added to by various local participants in the workflow, as shown previously at the bottom of the diagram in Figure 3-44 on page 140.

In addition to local data access, Spectrum Scale empowers global collaborative workflows by caching data on Spectrum Scale clusters anywhere in the world. This provides low-latency access to data anywhere that it is needed.

Also, Spectrum Scale can tier data automatically, which significantly lowers the costs of storing data. From flash to high-speed disks, to high-capacity disks and even tape, all data, no matter where it is, is still visible in the file system. Spectrum Scale puts the right data in the right place at the right time with the right performance and the right cost.

3.5.4 IBM Spectrum Scale architecture

Spectrum Scale provides high performance by allowing data to be accessed over multiple computers at once. Most existing file systems are designed for a single-server environment, and adding more file servers does not improve performance. Spectrum Scale provides higher input/output performance by “striping” blocks of data from individual files over multiple disks and reading and writing these blocks in parallel. Other features provided by GPFS include high availability, support for heterogeneous clusters, disaster recovery, security, DMAPI, HSM, and ILM.

According to Schmuck and Haskin, a file that is written to the file system is broken up into blocks of a configured size, less than 1 megabyte each. These blocks are distributed across multiple file system nodes, so that a single file is fully distributed across the disk array. This results in high reading and writing speeds for a single file because the combined bandwidth of the many physical drives is high. This makes the file system vulnerable to disk failures, and any one disk failing would be enough to lose data. To prevent data loss, the file system nodes have RAID controllers, so multiple copies of each block are written to the physical disks on the individual nodes. It is also possible to opt out of RAID-replicated blocks, and instead store two copies of each block on different file system nodes.

The file system also includes these features:

- Distributed metadata, including the directory tree. There is no single “directory controller” or “index server” in charge of the file system.
- Efficient indexing of directory entries for very large directories. Many file systems are limited to a small number of files in a single directory (often 65536 or a similar small binary number). Spectrum Scale does not have such limits.
- Distributed locking. This allows for full Posix file system semantics, including locking for exclusive file access.
- Partition Aware. A failure of the network may partition the file system into two or more groups of nodes that can only see the nodes in their group. This can be detected through a heartbeat protocol. When a partition occurs, the file system remains live for the largest partition formed. This offers a graceful degradation of the file system, and some machines will remain working.
- File system maintenance can be performed online. Most of the file system maintenance chores (adding new disks, rebalancing data across disks) can be performed while the file system is live. This ensures the file system is available more often, so keeps the supercomputer cluster itself available for longer.

It is interesting to compare this with Hadoop's HDFS file system. It is designed to store similar or greater quantities of data on commodity hardware, that is, data centers without RAID disks and a storage area network (SAN).

- HDFS also breaks files up into blocks and stores them on different file system nodes.
- HDFS does not expect reliable disks, so it stores copies of the blocks on different nodes. The failure of a node that contains a single copy of a block is a minor issue, dealt with by re-replicating another copy of the set of valid blocks to bring the replication count back up to the desired number. In contrast, although Spectrum Scale supports recovery from a lost node, it is a more serious event, one that might include a higher risk of data being temporarily lost.
- Spectrum Scale supports full Posix file system semantics. HDFS does not support full Posix compliance.
- Spectrum Scale distributes its directory indexes and other metadata across the file system. Hadoop, in contrast, keeps this on the primary and secondary name nodes, large servers that must store all index information in-RAM.
- Spectrum Scale breaks files up into small blocks. Hadoop HDFS uses blocks of 64 MB or more, because this reduces the storage requirements of the name node. Small blocks or many small files fill up a file system's indexes quickly, so they limit the file system's size.

Figure 3-45 illustrates the GPFS (Spectrum Scale) basic "common" building block architecture.

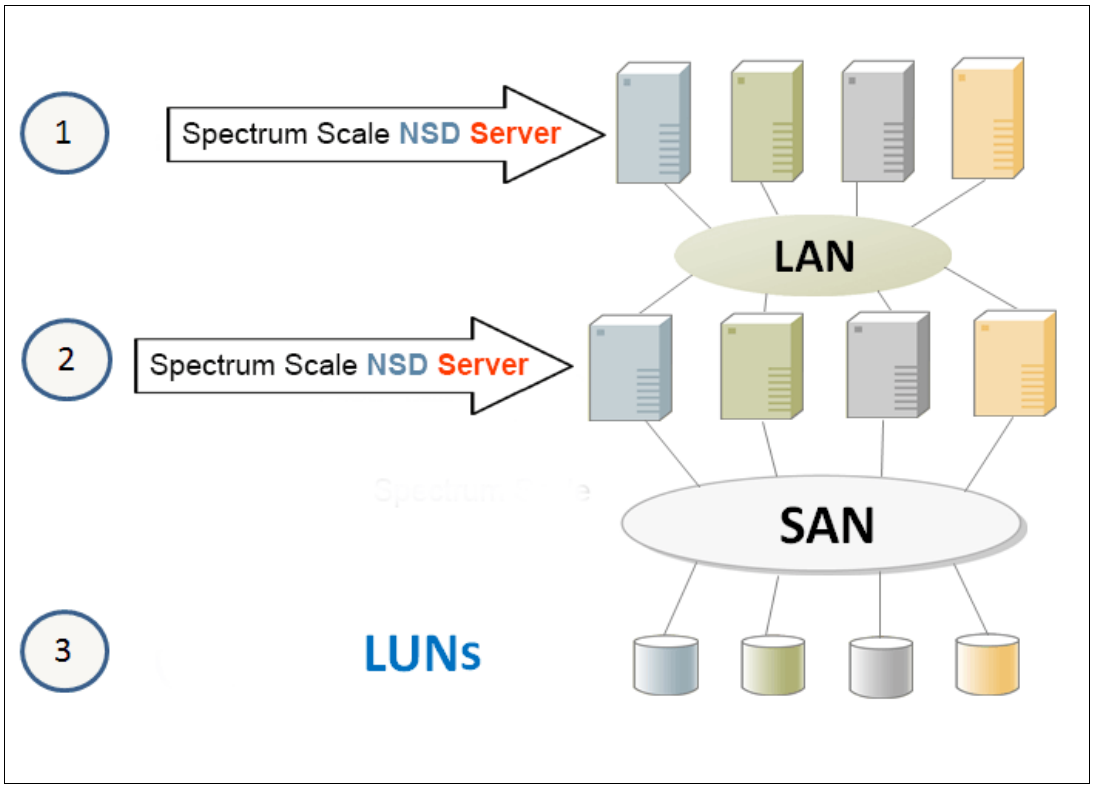


Figure 3-45 GPFS common architecture

1. Individual GPFS Network Shared Disk (NSD) Clients are installed on each server that is using GPFS.
2. The clients are assigned to the GPFS NSD servers.

3. An assignment is made to the logical unit numbers (LUNs) in the attached SAN (storage area network, shared SAS, twin-tailed, and so on). There is a 1:1 relationship between the NSD client and LUN as a building block starting point.

Next, as the environment grows, and individual islands of compute, network, and storage are optimized for the business, the GPFS architecture and technology can scale rapidly to support these demands. Figure 3-46 illustrates a typical GPFS based architecture in a customer environment.

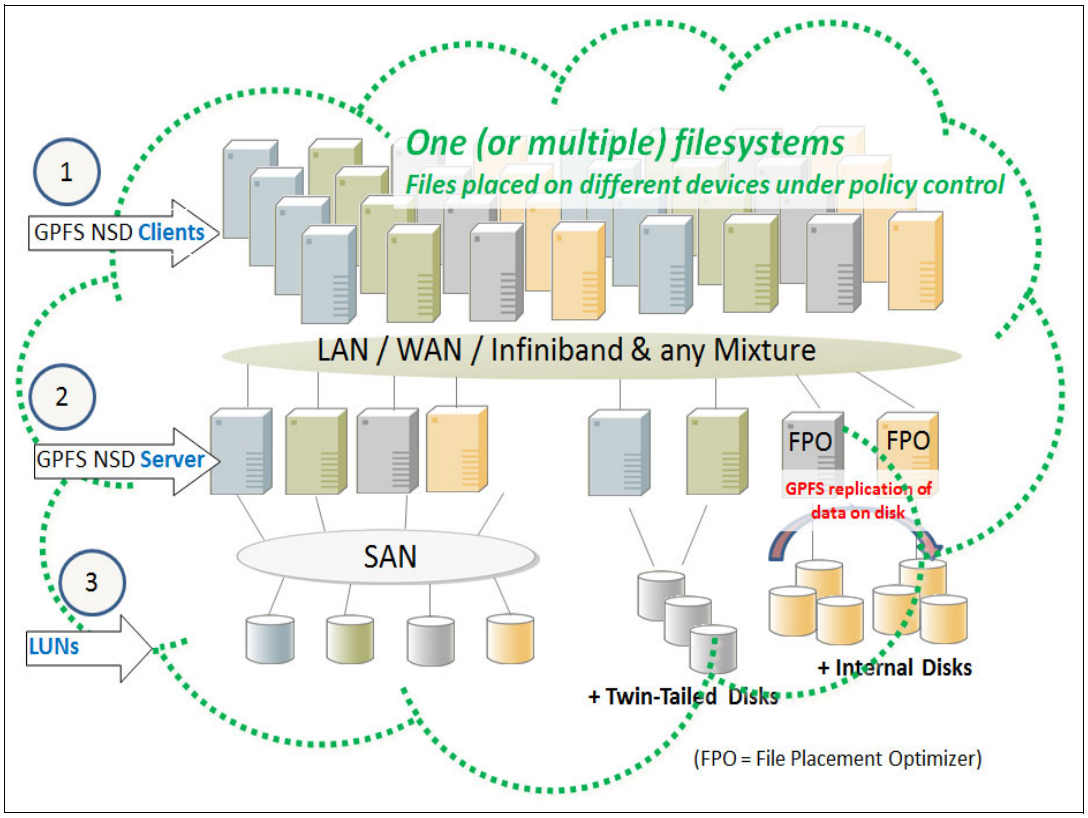


Figure 3-46 GPFS architecture - Typical deployment

1. Individual GPFS NSD dlients are installed on each server that is using GPFS. Files are placed on different devices under policy control.
2. Network assignments to the GPFS NSD servers are supported through LAN or WAN or through InfiniBand cross-site or at extended distance. Any mixture of these network types is supported. In addition, GPFS replication of data on disk to internal disks, and twin-tailed disks is supported through the File Placement Optimizer.

> **Note:** A *twin-tailed disk* is a disk or group of disks that are attached to two nodes. For recoverability purposes, only one of these nodes serves the disks at any given time. The secondary or backup node provides access to the disks if the primary node fails, is powered off, or if you need to change the server node temporarily for administrative reasons. Both must be in the same system partition.
>
> A twin-tailed volume group is a volume group that contains disks that are accessible to two nodes. Both nodes must be in the same system partition.

3. Assignment to LUNs via SAN + twin-tailed disks + internal disks is supported, and this can be a common and typical architectural pattern and deployment in customer data centers.

As Figure 3-47 shows, GPFS can support multisite data center architectures to the highest level of data loss protection through synchronous data replication via the LAN network to both sites. In this stretch cluster solution architecture, the file system is active-active across sites, and a quorum node is typically installed at a third site for tie-breaker and split-brain scenarios that involve in-flight writes between the sites.

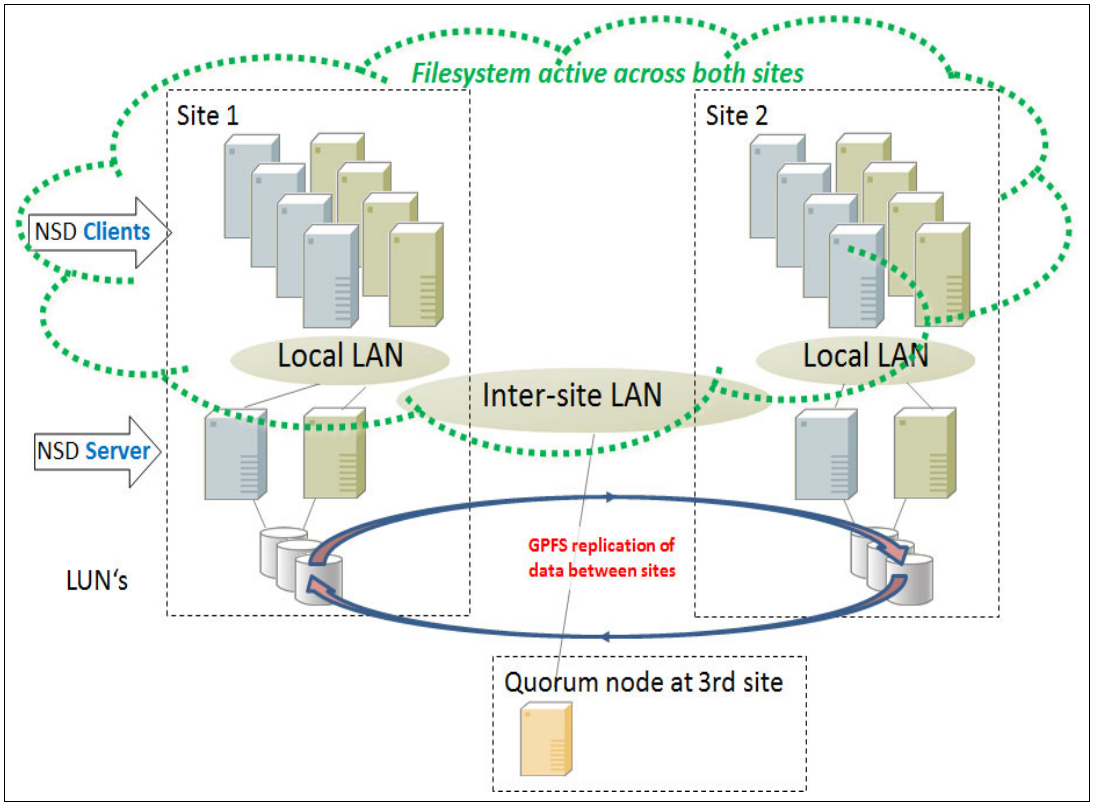


Figure 3-47 GPFS stretched cluster, synchronous

Also, as Figure 3-48 shows, GPFS can also support data loss protection remotely. This is through asynchronous cache from a local cluster to a remote cluster via read/write through AFM at the file system level across WANs.

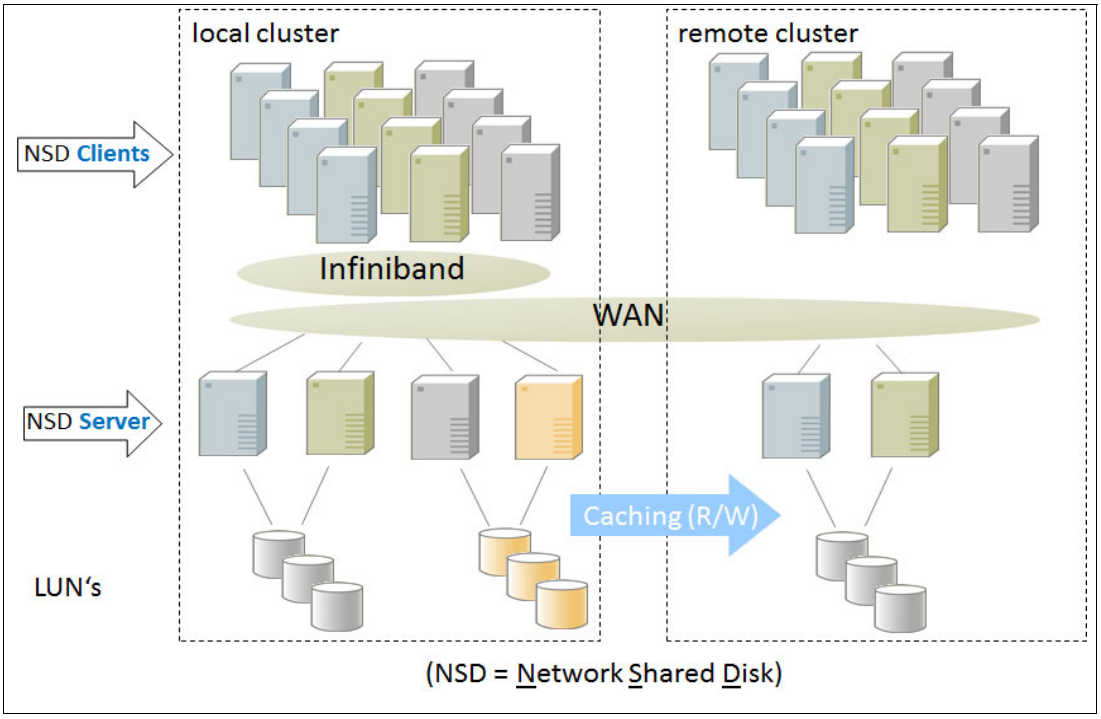


Figure 3-48 GPFS Active File Management (AFM)

3.5.5 System x GPFS Storage Server

Deploying solutions for technical computing, high-performance computing (HPC), analytics, and cloud environments can place a significant burden on IT. The IBM System x GPFS Storage Server, fulfilled through the IBM Intelligent Cluster, uses decades of IBM experience to reduce the complexity of deployment with integrated, delivered, and fully-supported solutions that match best-in-industry components with optimized solution design. With the GPFS Storage Server, clients can focus their efforts on maximizing business value rather than using valuable resources to design, optimize, install, and support the infrastructure required to meet business demands. As part of making these solutions simple, IBM introduces the GPFS Storage Server (GSS21, GSS22, GSS24 and GSS26). The GPFS Storage Server combines the performance of System x servers with industry-leading GPFS software to offer a high-performance, scalable building-block approach to modern storage needs. The GPFS Storage Server enables organizations to start small and build via incremental additions, providing expanded capacity and bandwidth with each additional GSS building-block.

The GPFS Storage Server features System x servers, storage enclosures and drives, software and networking components that allow for a wide choice of technology within an integrated, delivered solution. IBM thoroughly tests and optimizes each solution for reliability, interoperability and maximum performance, so clients can quickly deploy the system and get to work achieving their business goals. With GSS, you gain all the reliability, integration and interoperability of an Intelligent Cluster solution, all within a high-capacity, GPFS-based storage architecture.

In April of 2013, IBM announced IBM System x GPFS Storage Server (GSS), a new storage solution based on System x hardware and IBM General Parallel File System (GPFS). Here are five things to know about the GSS solution[10]:

1. GPFS Storage Server is powered by IBM GPFS software, running on many of the world's largest scientific supercomputers and commercial applications.

   GPFS is a high-performance shared-disk clustered file management solution. It provides fast and reliable access to a common set of data, online storage management, scalable access, and tightly integrated information lifecycle tools capable of managing petabytes of data and billions of files.

2. GPFS Storage Server is built for performance and scalability.

   GPFS Storage Server offers high performance on a scalable building-block approach. Performance and capacity increase as you add server and disk enclosures. Two models are available:

   – Model 24 - 2 x3650 M4 servers with 4 disk enclosures, with up to 232 disk drives
   – Model 26 - 2 x3650 M4 servers with 6 disk enclosures, with up to 348 disk drives

   The offering scales up to a high-density HPC option that comprises 6 x3650 M4 servers and 18 disk enclosures, all of which fit in two standard 42U racks. Such a configuration holds more than 1000 drives and as much as 3 petabytes of storage.

3. GPFS Storage Server is built on standard System x components. GPFS Storage Server is a single, integrated, fully supported IBM solution, built to use a strong GPFS software market.

   There are two models, Model 24 and Model 26. GPFS Storage Server comprises:

   – 4 (Model 24) or 6 (Model 26) NetApp DE6600 SAS Disk Controller JBOD disk enclosures
   – Choice of FDR InfiniBand or Ethernet interconnects, or both
   – GPFS software (RAID functions performed by the optional GPFS Native RAID feature)

   Each NetApp disk enclosure holds up to 58 drives each, a mixture of solid-state drives and spinning drives, with capacities of up to 3 TB per drive.

4. IBM Elastic Storage™ Server (ESS) runs on IBM System x3650 M4 servers.

   The IBM System x3650 M4 server was one of IBM's top selling two-socket Intel-based rack offerings. These servers provide outstanding performance for business-critical applications such as data management. Its energy-efficient design supports high performance, memory, and data capacity in a scalable 2U package that is easy to service and manage. With more computing power per watt and the latest Intel Xeon E5 processors, you can reduce costs while maintaining speed and availability.

5. The Elastic Storage Server is available as part of the IBM Intelligent Cluster portfolio.

   GSS is delivered as an integrated storage solution consisting of servers, hard disk drives (HDDs), solid-state drives (SSDs) and software. With it, users will have improved data availability and integrity with improved performance and efficiency, at on a cost competitive per terabyte price.

With the Intelligent Cluster portfolio, IBM provides single point-of-contact, solution-level support that includes both IBM and third-party components to deliver maximum system availability throughout the life of the system, so you can spend less time maintaining systems and more time delivering results.

[10] David Watts. "5 Things To Know About IBM System x GPFS Storage Server." IBM developerWorks, May 2012. https://www.ibm.com/developerworks/community/blogs/5things/entry/gpfs_storage_server

For more information about the IBM Elastic Storage Server, see the following web pages and publications:

- IBM United States Hardware Announcement 113-069, April 30, 2013 (announcements for other countries available from IBM Offering Information): IBM Intelligent Cluster portfolio now includes IBM Flex Systems and enhancements to DataDirect Networks, Mellanox, and NetApp offerings

  http://ibm.co/1b7uzNA

- IBM Elastic Storage Server product page

  http://www.ibm.com/systems/x/hardware/highdensity/gpfsstorage/index.html

- IBM Spectrum Scale product page

  www.ibm.com/systems/storage/spectrum/scale/index.html

- *IBM System x3650 M4*, TIPS0850, an IBM Redbooks Product Guide

  http://www.redbooks.ibm.com/abstracts/tips0850.html

You can see that Spectrum Scale is a very strong part of the SDE solution and fits with the application-aware strategy, architecture, and deployment patterns for today's customer environments—and tomorrow's.

For more information about GPFS, see the following sources:

- IBM Spectrum Scale product page

  www.ibm.com/systems/storage/spectrum/scale/index.html

- General Parallel File System (now called Spectrum Scale)

  http://www.ibm.com/software/products/en/software

- IBM Elastic Storage converged storage platform for emerging and traditional workload (Elastic Storage was the code name for what is now called Spectrum Scale)

  http://www.youtube.com/watch?v=2eDrB6ePbQ4

- IBM GPFS, IBM Platform Computing

  http://www.youtube.com/watch?v=kAJlPYqrms8

- General Parallel File System 4.1 in IBM Knowledge Center

  http://www.ibm.com/support/knowledgecenter/SSFKCN/gpfs_welcome.html

- IBM Redbooks publications
  - Implementing the IBM General Parallel File System (GPFS) in a Cross Platform Environment, SG24-7844

    http://www.redbooks.ibm.com/abstracts/sg247844.html
  - *A Deployment Guide for IBM Spectrum Scale Object*, REDP-5113

    http://www.redbooks.ibm.com/abstracts/redp5113.html
  - *IBM Private, Public, and Hybrid Cloud Storage Solutions*, REDP-4873

    http://www.redbooks.ibm.com/abstracts/redp4873.html
  - *IBM Platform Computing Solutions Reference Architectures and Best Practices*, SG24-8169

    http://www.redbooks.ibm.com/abstracts/sg248169.html

In the next section, we describe IBM XIV storage and how it fits in the SDE with OpenStack environment that is based on integration and its features, capabilities, and architecture.

3.5.6 IBM XIV storage

IBM XIV is a high-end disk storage system that stands out for tuning-free predictable high performance, an exceptional user experience, and excellent data economics. It is ideal for cloud and analytics, offering high service levels for dynamic workloads, easy hyper-scaling including in multitenant environments, and flexible use models – and robust integration with cloud orchestration platforms via OpenStack, RESTful API, VMware and other means. It fortifies security and data protection through three-way mirroring, hot encryption and PCI-DSS support. The unique XIV grid architecture is core to the system's far-reaching customer value. Industry benchmarks underscore stellar XIV performance and cost benefits.

IBM XIV Storage System is high-end, flash-optimized data storage built to deliver simplicity, resiliency, and predictability that are ideal for cloud and big data analytics environments, and integrates seamlessly in an SDE OpenStack environment. Designed to simplify storage, XIV Storage System is enabling thousands of organizations to take control of their storage and gain business insights from their data. XIV storage is deployed in diverse industries worldwide, including financial, healthcare, telecommunications, managed services, energy and manufacturing. It integrates easily with virtualization, email, database, analytics, data protection and other solutions from leading providers such as VMware, Microsoft, SAP, Oracle Solaris, SAS and Symantec.

Table 3-5 shows a summary of the key features and benefits of the IBM XIV Storage System.

Table 3-5 IBM XIV benefits

Features	Benefits
Grid architecture with InfiniBand interconnect and automated uniform data distribution within and across XIV systems	Fully used system resources at all times, including disks, CPU cores, RAM and cache result in: ► Hotspot-free, consistent and predictable high performance across dynamic workloads. without the need for tuning, ideal for cloud ► Rapid disk rebuild takes place in minutes per TB, with negligible performance impact.
Flash caching	► Up to 4.5 times performance boost for database-like workloads. ► Reduced need to access data on spindles due to sizable distributed flash cache, management-free.
Exceptionally simple management tools	► Highly intuitive GUI. IBM Hyper-Scale Manager for consolidated, integrated ease of management across up to 144 XIV frames, ideal for multitenant cloud deployments. Automatic notification alerts to mobile devices. Can also integrate with the OpenStack Management portal. ► Powerful command-line interface (CLI) for complex scripting. ► XIV mobile dashboard supporting Apple iOS and Google Android-based devices to provide flexible remote monitoring of performance, health, and capacity.

Features	Benefits
Built-in advanced features	► Thousands of snapshots with virtually no performance impact. IBM Hyper-Scale Consistency for concurrent, consistent snapshots of all the volumes of an application across multiple XIV systems, for enhanced data protection. ► Three-site replication featuring three concurrent copies of data for disaster recovery, using field-proven synchronous and asynchronous mirroring. ► IBM Hyper-Scale Mobility for online volume movement between XIV systems, nondisruptive to applications for operational agility. ► Easy-to-use thin provisioning for optimal use of storage resources. ► Quick-start synchronous mirroring through offline initialization for easy swapping between synchronous and asynchronous mirroring to meet varied service-level needs.
Cloud and hypervisor integration	► Open-standards tools support, including for OpenStack and IP SAN (10 Gbps), for robust and highly available and automated cloud deployments. ► IBM Cloud Storage Access, enabling cloud storage by providing self-service, on-demand capacity provisioning with metering and chargeback options, and improving employee productivity. ► XIV representational state transfer (REST) application programming interface (API) for integrating XIV monitoring and automated provisioning in cloud environments. ► Deep VMware integration, including VMware vCloud Suite and vStorage APIs for Array Integration (VAAI). Certified, robust disaster-recovery via VMware Site Recovery Manager (SRM). Advanced, IBM-native storage visibility and self-service provisioning through the new IBM Storage Integration Server via a vSphere Web Client plug-in. Profile-based storage provisioning and automated storage placement via vSphere APIs for Storage Awareness (VASA). Space reclamation via XIV support for VMware ESXi v5.5. ► Integration with Microsoft Hyper-V and Microsoft System Center virtual machine manager for simpler storage management.

Features	Benefits
Security	► Industry-standard data-at-rest encryption without performance impact. Uses AES 256-bit keys. Trusted Computing Group (TCG) storage specification-compliant. Separate, simplified key support by IBM Security Key Lifecycle Manager. ► USGv6 certification and support for the added security features available with IPv6 and IPSec. ► Secure multitenancy enabling IT organizations and service providers to consolidate and benefit from cloud economics.
IBM Tivoli Storage FlashCopy® Manager integration	► Automated, application-aware snapshots for fast and easy backup and restore of popular enterprise applications.

XIV architecture

IBM XIV Storage System Gen3 carries on the XIV tradition of providing several architecture design factors that contribute to its unique operational capabilities:

- Virtual, grid-based array design (to minimize storage complexity)
- Self-tuning and self-healing (to deliver performance and availability without manual tuning)
- Easy storage management and provisioning (an intuitive GUI to simplify the storage operation within a virtualized storage architecture)
- Enhanced connectivity and reliability (to strengthen the end-to-end performance without disruption to the service)
- Innovative caching with optional flash cache, coupled with high-density storage (to reduce the footprint and address energy efficiency while sustaining performance)

The XIV Storage System architecture incorporates various features designed to uniformly distribute data across internal resources. This unique data distribution method fundamentally differentiates the XIV Storage System from conventional storage subsystems, offering numerous availability, performance, and management benefits across both physical and logical elements of the system.

The XIV Storage System has a native virtual storage design that is both efficient and simple to use. It eliminates all physical disk Redundant Array of Independent Disks (RAID) management tasks. The XIV Storage System is an Enterprise-Class storage system in which storage management involves defining capacity (a logical unit number (LUN) or a volume) and assigning the capacity to a host system.

The XIV Storage System hardware architecture is a dramatic shift from traditional RAID-based parity data protection. From a performance perspective, the XIV Storage System is able to *automatically* involve all of the disk drives, the whole system cache including its optional flash cache extension, and processors in performing the work of servicing I/O.

Scalability is also a strength of the XIV Storage System. The XIV Storage System is able to scale without administrative involvement to redistribute data or tune for optimum performance. It takes care of these tasks automatically.

XIV hardware elements

To convey the conceptual principles that make up the XIV Storage System architecture, it is useful to first look at the physical design.

The IBM XIV Storage System configuration includes data modules, interface modules, interconnect switches, and uninterruptible power supply (UPS) units.

Modules

The primary components of the XIV Storage System are known as *modules*. Modules provide processing, cache (including optional flash cache), and host interfaces. They are composed of “off the shelf” systems that are based on Intel technology.

The modules are redundantly connected to one another through an internal switched network, as shown in Figure 3-49.

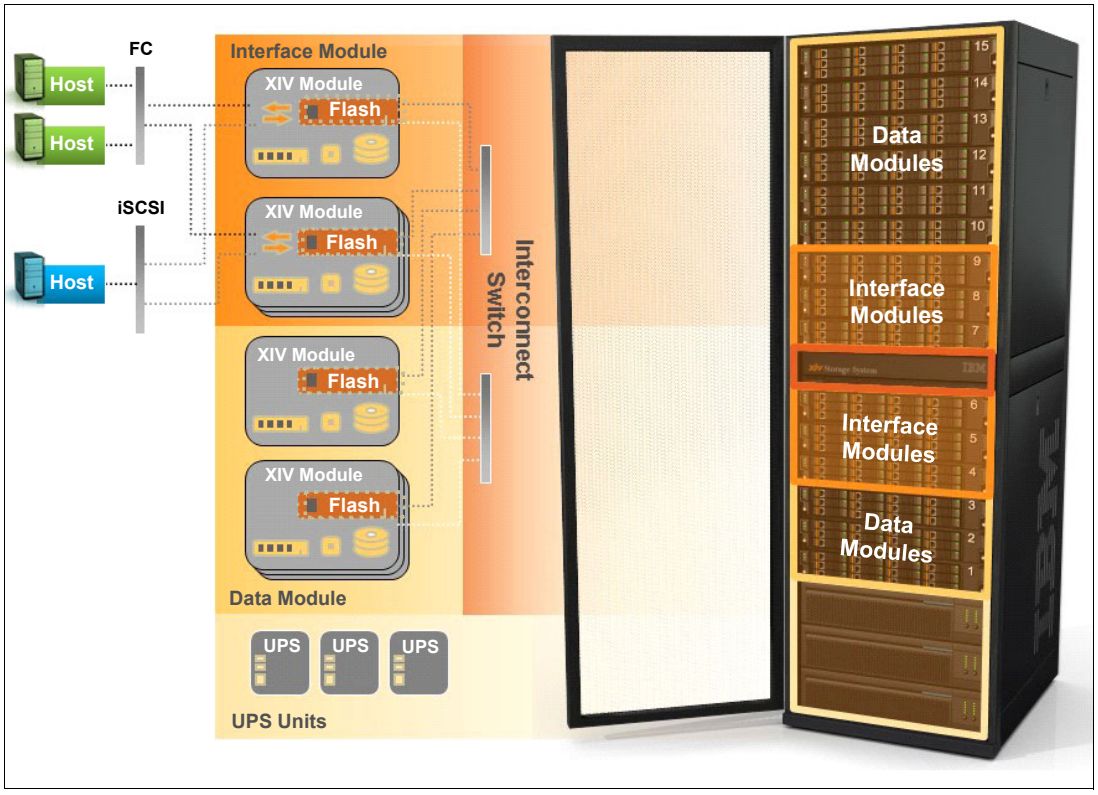


Figure 3-49 XIV Storage System major hardware elements

All of the modules work together concurrently as elements of a grid architecture, so the system harnesses the powerful parallelism inherent in such a distributed computing environment.

Important: Flash cache is an optional feature. However, if you opt for the flash cache extension, flash cache devices must be installed in *all* modules.

Data modules

At a conceptual level, data modules function as the elementary “building blocks” of the system, providing storage capacity, processing power, and caching, in addition to advanced system-managed services. The data modules’ ability to share and manage system software and services are key elements of the physical architecture, as shown in Figure 3-50.

Note: Figure depicts the conceptual architecture only. Do not misinterpret the number of connections or modules and other representations as a precise hardware layout.

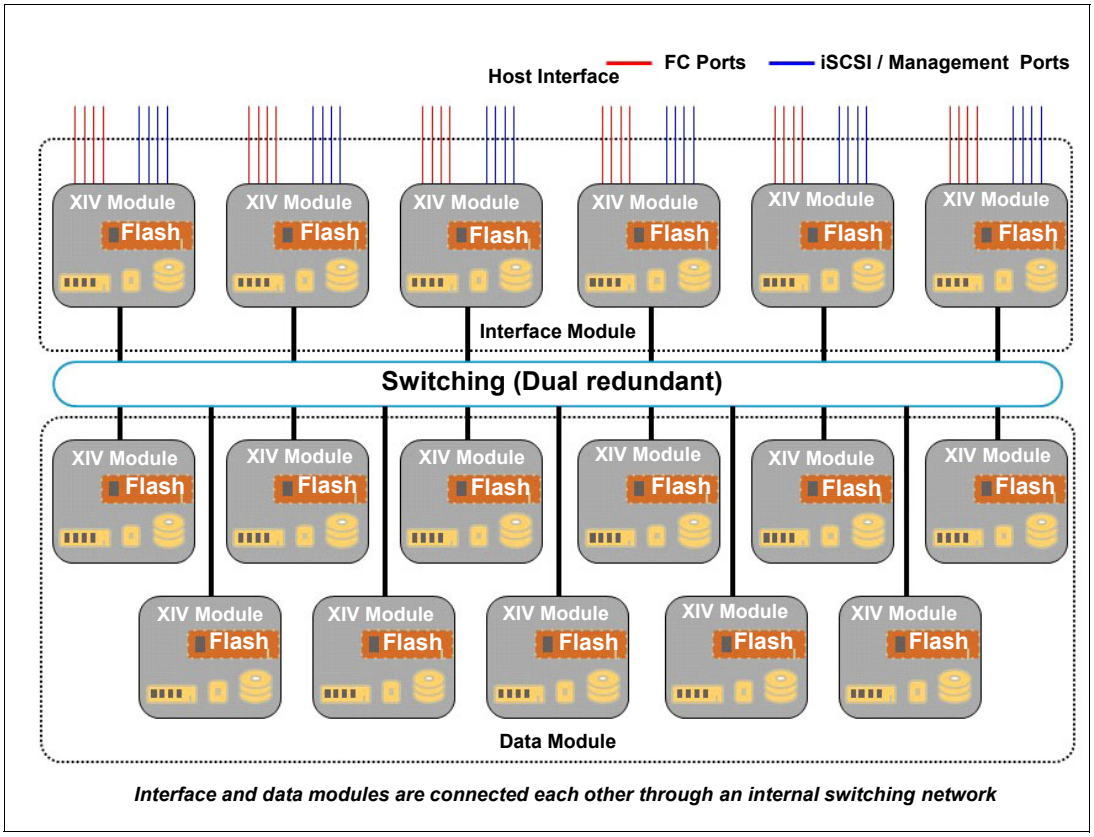


Figure 3-50 XIV Storage System architectural overview

Interface modules

Interface modules are equivalent to data modules in all aspects, with the following exceptions:

- In addition to disk, cache, and processing resources, interface modules are designed to include both Fibre Channel and IP network Small Computer System Interface (iSCSI) interfaces for host system connectivity, remote mirroring, and data migration activities. Figure 2-48 conceptually illustrates the placement of interface modules within the topology of the XIV Storage System architecture.
- The system services and software functions associated with managing external I/O is located exclusively on the interface modules.

Module interconnect switches

The XIV Storage System contains a redundant switched network, based on InfiniBand, that transmits both data and metadata traffic between the modules. Traffic can flow between two interface modules, between two data modules, and between an interface module and a data module.

Parallelism

The concept of *parallelism* pervades all aspects of the XIV Storage System architecture by using a balanced, redundant data distribution scheme with a pool of distributed (or grid) computing resources. To explain the principle of parallelism further, it is helpful to consider the ramifications of both the hardware and software implementations independently.

Note: The XIV Storage System uses parallelism at *both* hardware and software levels.

Hardware parallelism and grid architecture

The XIV grid design (Figure on page) has the following characteristics:

- Both interface and data modules work together in a distributed computing sense. However, the interface modules also have additional functions and features associated with host system and remote mirroring connectivity.
- The modules communicate with each other through an internal, redundant switched network.
- The software services and distributed computing algorithms running within the modules collectively manage all aspects of the operating environment.

Design principles

The XIV Storage System grid architecture, by virtue of its distributed topology, ensures that the following design principles are possible:

- Performance:
  - The relative effect of the loss of a single component is minimized.
  - All modules are able to participate equally in servicing the total workload.

  This design principle is true regardless of access patterns. The system architecture inherently enables excellent load balancing, even if certain applications access certain volumes, or certain parts within a volume, more frequently.
- Compatibility:
  - Modules consist of standard “off the shelf” components.

  Because components are not specifically engineered for the system, the resources and time required for the development of newer hardware technologies are minimized. This benefit, coupled with the efficient integration of computing resources into the grid architecture, enables the system to realize the rapid adoption of newer hardware technologies available without the need to deploy a whole new subsystem.
- Scalability:
  - Computing resources can be dynamically changed.
  - The architecture can be “scaled out” by adding new modules to accommodate both new capacity and new performance demands.
  - The architecture can also be “scaled up” by adding more modules, up to a total of 15 modules.

Figure 3-51 provides a conceptual depiction of the scalable grid architecture of the XIV Storage System.

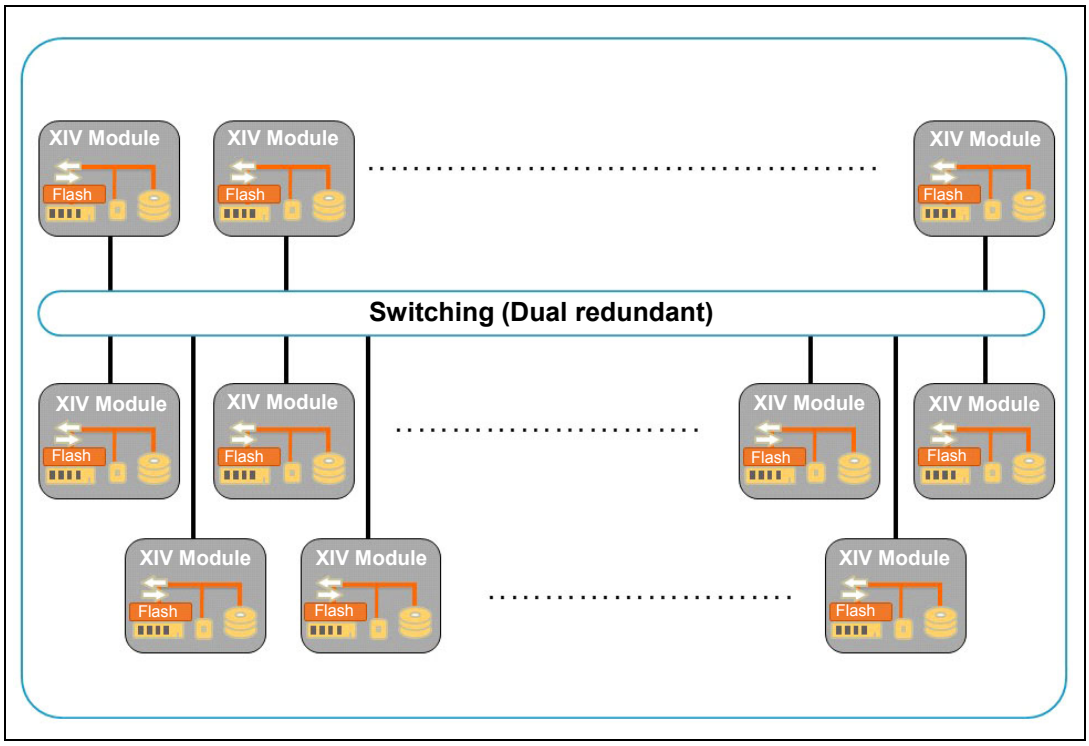


Figure 3-51 XIV Storage System scalable conceptual grid architecture

Proportional scalability

Within the XIV Storage System, each module contains all of the pertinent hardware elements that are necessary for a grid topology (processing, caching, and storage). All modules are connected through a scalable network. This aspect of the grid infrastructure enables the relative proportions of cache, processor, disk, and interconnect bandwidth to remain optimal even when modules are added or removed:

- Linear cache growth. The total system cache size and cache bandwidth increase linearly with disk capacity because every module is a self-contained computing resource that houses its own cache. The cache bandwidth scales linearly in terms of both host-to-cache and cache-to-disk throughput, and the close proximity of cache, processor, and disk is maintained.

  Cache: Flash cache also increases linearly with total memory capacity. Flash cache drives are not used to expand the disk capacity, but rather to extend the memory capacity allocated to the caching of read I/Os.

- Proportional interface growth

  Interface modules house Ethernet and Fibre Channel host interfaces and can access both local resources within the module and the entire system. With every interface module added, the system proportionally scales both the number of host interfaces and the bandwidth to the internal resources.

- Constant switching capacity

  The internal switching capacity is designed to scale proportionally as the system grows, preventing bottlenecks regardless of the number of modules. This capability ensures that internal throughput scales proportionally to capacity.

- Embedded processing power

  Because each module incorporates its own processing power with cache and disk components, the ability of the system to perform processor-intensive tasks (such as aggressive prefetch caching, sophisticated cache updates, snapshot management, and data distribution) is always maintained, regardless of the system capacity.

Software parallelism

In addition to the hardware parallelism, the XIV Storage System also employs sophisticated algorithms to achieve optimal software parallelism.

Modular software design

The XIV Storage System internal operating environment consists of a set of software functions that are loosely coupled with the hardware modules. These software functions reside on one or more modules and can be redistributed among modules as required, therefore ensuring resiliency under changing hardware conditions.

An example of this modular design is located specifically in the interface modules. All six interface modules actively manage system services and software functions associated with managing external I/O. Also, three of the interface modules deliver the system's management interface service for use with the XIV Storage System.

Data distribution algorithms

Data is distributed across all drives in a *pseudo-random* fashion. Patented algorithms provide a uniform yet random distribution of data, which is divided into 1 MB partitions across all available disks, to maintain data resilience and redundancy.

Figure 3-52 shows how all drives are used evenly by partition units regardless of applications or the size of assigned logical volumes. The storage administrator does not need to worry about data placement impact on performance.

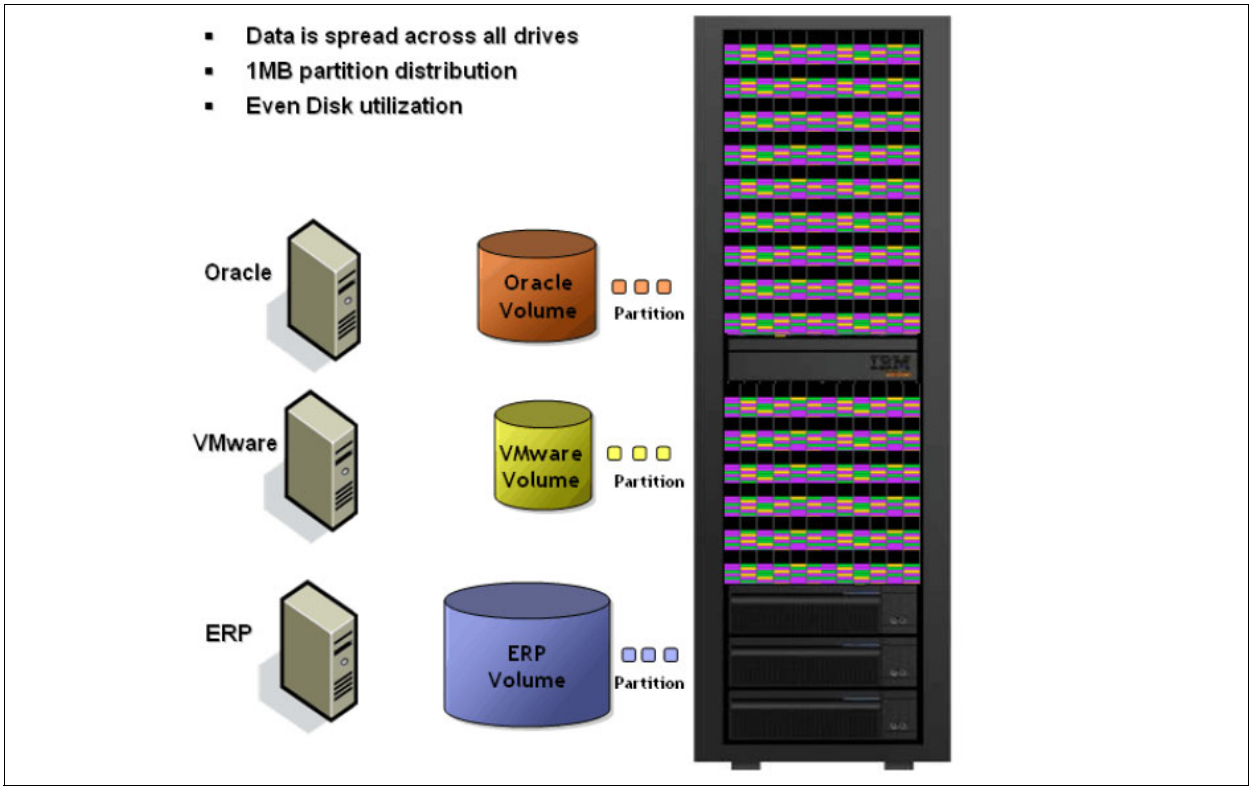


Figure 3-52 XIV Storage System pseudo-random data distribution

Flash caching architecture overview

Starting with the XIV System software Version 11.1.0, the XIV Storage System Gen3 supports an optional flash caching feature that significantly improves performance for small block, random-read I/O workloads.

XIV flash caching is implemented as an extension of the primary cache layer. Flash cache is used exclusively as a read cache. Each module supports 400 GB cache, which allows the caching layer to be expanded up to 6 TB. Starting with software Version 11.4, a flash cache of 800 GB per module for a total of 12 TB in a fully populated system with 4 TB drives can be used optionally to further boost performance.

With flash caching, as illustrated in Figure 3-53 on page 157, there is no need to relocate data. Because flash cache is used as read cache only when data in cache is no longer accessible, it can simply be dropped and replaced by more relevant data.

This approach allows for the flash cache to be used very effectively in highly dynamic environments where data patterns are constantly changing. To have a holistic solution, the caching architecture must also deliver good write performance.

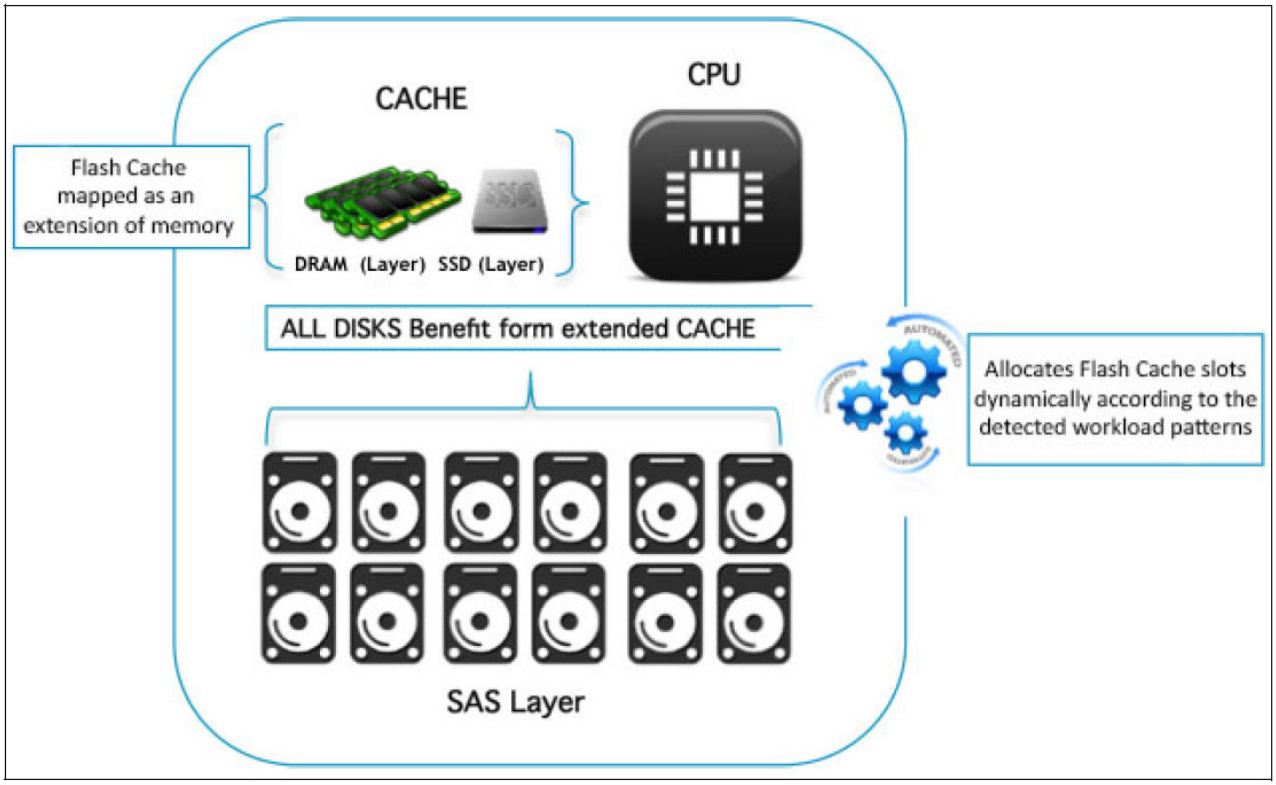


Figure 3-53 XIV flash caching overview

The caching algorithm is embedded in the XIV System software (firmware) and makes the flash cache integration completely transparent to the user or storage administrator. No tuning of the cache is required to get the potential performance boost.

The flash caching feature is supported automatically by the XIV unique caching algorithm, which dynamically adapts to detected I/O request patterns.

Flash caching algorithm and concepts

Caching operations are split into two groups, main and extended:

- The *main cache* handles host write I/Os and then destages them directly to the disk drive.
- The *extended cache* handles the caching of random read miss operations less than 64 KB.

Sequential read prefetches (larger than 64 KB) are handled in main dynamic random access memory (DRAM) cache.

Flash caching: For more details about flash caching, see the IBM Redpaper™ publication titled *Solid-State Drive Caching in the IBM XIV Storage System*, REDP-4842:

`http://www.redbooks.ibm.com/abstracts/redp4842.html`

A flash cache map is built as read misses occur in the DRAM cache. The process, known as *flash cache learning,* is depicted in Figure 3-54.

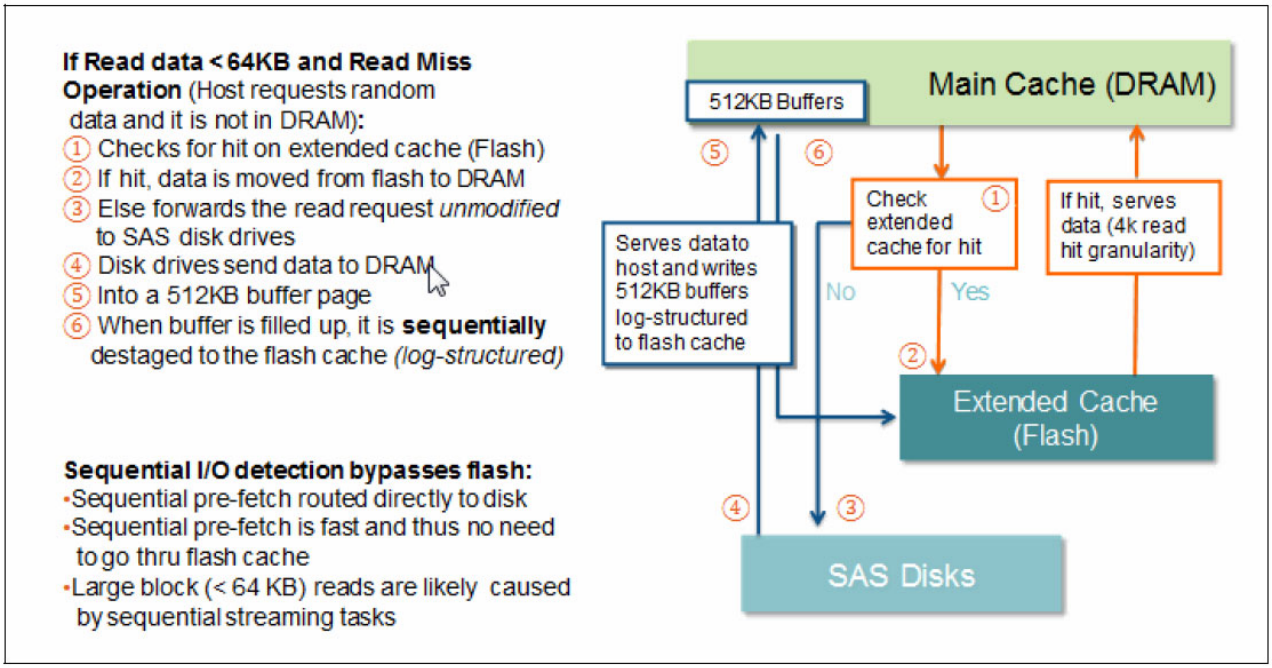


Figure 3-54 XIV flash cache learning

The cache node immediately checks the extended cache for the requested I/O. If the requested I/O exists in the extended flash cache, it is served to the host through the main cache. The I/O operation is now complete and is recorded as a flash cache read hit.

If the operation results in a true read miss (not in the DRAM cache and not in extended flash cache), the request is forwarded in an unmodified state to the disk drive (SAS layer). The I/O is retrieved from the disk drive and served to the host through the main cache. From a host perspective, the I/O operation is now complete and is recorded as a read miss. The related pages are copied into reserved buffers in the main cache.

Important: Any read larger than 64 KB bypasses the extended flash cache.

When the buffer reaches 512 KB, it is written sequentially to the flash cache as a log-structured write. This method helps to prolong the life of the flash cache.

Note: The XIV retains the data in the flash cache between system restarts and code upgrades.

XIV Storage System Software Version 11.2 introduced improved flash caching algorithms that boost performance up to 4.5 times over systems without flash cache for random database-type workloads. This boost is accomplished by storing and computing all flash cache-related data integrity checking tasks in DRAM, rather than on the flash cache.

XIV hardware architecture and configuration, IBM XIV Storage System Gen3

The XIV Storage System Gen3 Model 214 shares all of the concepts and software architecture of its predecessor Model 114. However, the Model 214 offers hardware upgrades that deliver significant performance improvements. The primary hardware component upgrades are in the interface and data modules:

- Processor
- Memory
- Power supply unit
- 1 TB[11], 2 TB, 3 TB, and 4 TB serial-attached SCSI (SAS) self-encrypting drives (SEDs)
- 400 GB flash drive cache option or 800 GB flash drive with 4 TB configurations
- Two-port 10 GbE adapter

The XIV Storage System Gen3, shown in Figure 3-55 on page 160, is a scalable enterprise storage system based on a grid array of hardware components. The architecture offers the highest performance through maximized and balanced use of all disks, a true distributed cache implementation, and exceptional performance characteristics. It also offers superior reliability through its distributed architecture, redundant components, self-monitoring, and self-healing attributes.

[11] The 1 TB drives are actually stripped-down 2 TB drives.



Figure 3-55 IBM XIV Storage System Gen3: Front and rear views

The XIV Storage System Gen3 can use flash drives, which are also referred to as solid-state drives (SSDs), in each module. Flash drives are used as secondary read cache between disks and the dynamic random access memory (DRAM) cache. Data is always placed in both cache units, and disk access is avoided when data is in the SSD. Because the SSD cache is multiple times larger than the DRAM cache, the flash cache is highly effective for the read cache hit ratio. XIV Storage software V11.1 or later is a prerequisite for using SSDs in Model 114.

Figure 3-56 on page 161 shows a table that lists the main hardware components of the IBM XIV Storage System Gen3.

Component	Model 114	Model 214 (1 GbE)	Model 214 (10 GbE)
Rack	IBM T42 42U	IBM T42 42U	IBM T42 42U
All Modules	Intel Quad core CPU 24 GB DDR memory 12x SAS drives 2x Redundant Power Supplies/module 2U	Intel Six core CPU 48 GB DDR3 1.3 GHz memory 12x SAS drives 2x High Efficient. Power Supplies 2U	Intel Six core CPU 48 GB DDR3 1.3 GHz memory 12x SAS drives 2x High Efficient. Power Supplies 2U
Interface Module	In addition to data module: - 2x iSCSI Port on Module 4 - 4x iSCSI Ports on active Modules - 4x 8 Gbps FC ports	In addition to data module: - 2x iSCSI Port on Module 4 - 4x iSCSI Ports on active Modules - 4x 8 Gbps FC ports	In addition to data module: - 2x iSCSI Ports 10 GbE - 4x 8 Gbps FC ports
Switch	- 2x 36 ports InfiniBand	- 2x 36 ports InfiniBand	- 2x 36 ports InfiniBand
Maintenance Module and Modem	- 1U Maintenance Module - 1x Modem	- 1U Maintenance Module - 1x Modem	- 1U Maintenance Module - 1x Modem
UPS	- 3x UPS	- 3x UPS	- 3x UPS
ATS	- 1x ATS	- 1x ATS	- 1x ATS
Full Rack	- 15x Modules - 180x disks - 2 TB disks = 161 TB - 3 TB disks = 243 TB - 360 GB memory (cache) - 6 TB flash cache - 6x interface modules - 24x FC ports - 22x iSCSI ports - 9x data modules	- 15x Modules - 180x disks - 2 TB disks = 161 TB - 3 TB disks = 243 TB - 4 TB disks = 325 TB - 360 GB cache with 2 TB, 3 TB disks - 720 GB cache with 4 TB disks - 6 TB flash cache - 12 TB flash cache with 4 TB disks - 6x interface modules - 24x FC ports - 22x iSCSI ports - 9x data modules	- 15x Modules - 180x disks - 2 TB disks = 161 TB - 3 TB disks = 243 TB - 4 TB disks = 325 TB - 360 GB cache with 2 TB, 3 TB disks - 720 GB cache with 4 TB disks - 6 TB flash cache - 12 TB flash cache with 4 TB disks - 6x interface modules - 24x FC ports - 12x iSCSI ports - 9x data modules

Figure 3-56 IBM XIV Storage System Gen3 hardware overview

All XIV Gen3 hardware components are delivered preinstalled in a standard IBM T42 19-inch rack. Data and interface modules provide the processing, caching, and storing of data. All modules can be considered data modules in that they each contain a processor, memory, and 12 serial-attached SCSI (SAS) drives. The SAS drives can be 2 TB and 3 TB. The SAS-SED drives can be 2 TB, 3 TB, or 4 TB. The interface modules are data modules but with more capabilities. The interface modules have unique software and hardware that provide Fibre Channel and IP network Small Computer System Interface (iSCSI) host connectivity.

Important: Disks of different sizes cannot be intermixed within the same XIV Storage System configuration. XIV Storage System Software Version 11.0.1 or later is required for using 3 TB disks, and Version 11.4.0 or later is required for using the SAS-SED drives in 2 TB, 3 TB, or 4 TB.

The bottom of the rack contains three uninterruptible power supplies (UPSes) that supply power to all XIV Storage System components. They also provide enough battery backup power for emergency XIV Storage System shutdown if main power is lost for more than 30 seconds.

A maintenance module and two InfiniBand switches are installed near the middle of the rack, just above module 6. The InfiniBand network provides redundant and fast communications paths between the modules. This grid network ensures communication between all modules even if one of the switches or a cable connection fails. Furthermore, this grid network provides the capabilities for parallelism and the execution of a data distribution algorithm that contributes to the performance of the XIV Storage System.

All cabling between the modules and switches, including the internal power connections, are fully redundant using two sets of cables. All cables use industry standard plugs.

As you can see, the IBM XIV Storage System offers an added benefit to the SDE solution with OpenStack. It seamlessly integrates for storage management, and either the IBM Cloud Manager portal may be used or the XIV GUI. Regardless, the same information is provided in both places, for ease of administration and consistency. See the “XIV for Cloud with OpenStack demo” on YouTube:

`http://www.youtube.com/watch?v=RgAYTUNrWe0`

The XIV grid-based architecture is superbly aligned to an SDE-based solution and cloud environments. It is low-management and comes with flash caching architecture to support ILM for valuable and critical workloads needing optimal performance. This “extra level” of intelligent architecture is perfect in an SDE environment.

For more information about XIV, see the following web pages and publications:

- *IBM XIV Storage System Architecture and Implementation*, SG24-7659

  http://www.redbooks.ibm.com/redpieces/abstracts/sg247659.html
- *Solid-State Drive Caching in the IBM XIV Storage System*, REDP-4842

  http://www.redbooks.ibm.com/abstracts/redp4842.html
- *Using XIV in OpenStack Environments*, REDP-4971

  `http://www.redbooks.ibm.com/abstracts/redp4971.html`
- *IBM XIV Storage System: Host Attachment and Interoperability*, SG24-7904

  http://www.redbooks.ibm.com/abstracts/sg247904.html

In the next section, we explain how IBM V7K (Storwize V7000) storage fits in the SDE with OpenStack environment that is based on integration and its features, capabilities, and architecture.

3.5.7 IBM Storwize V7000 storage systems

IBM Storwize V7000 Unified and IBM Storwize V7000, also known informally as V7K, are virtualized, enterprise-class storage systems that provide the foundation for implementing an effective storage infrastructure and transforming the economics of data storage. With hardware accelerated IBM Real-time Compression™, they can reduce the cost of storage by up to half while maintaining application performance.

IBM Storwize V7000 Unified and Storwize V7000 provide the latest storage technologies for unlocking the business value of stored data, including virtualization and Real-time Compression. The systems also include a powerful new hardware platform that can support the massive volumes of data created by today's demanding cloud and analytics applications. The Storwize systems deliver outstanding efficiency and ease of use and dependability for organizations of all sizes. Table 3-6 provides an overview of the key features and benefits.

Table 3-6 Storwize features and benefits

Features	Benefits
IBM Systems Director	► Provides integrated management for physical and virtual server infrastructures, including monitoring and repairing for higher availability and operational efficiency.
IBM Flex System Manager®	► With IBM PureFlex Systems, provides management integration of physical and virtual compute, storage and networking resources from a single vantage point.
IBM Storage Mobile dashboard	► Provides basic monitoring capabilities to securely check the health and performance of Storwize family systems.
Dynamic migration	► Provides efficiency and business value in a nondisruptive migration function.
IBM Easy Tier	► Provides automatic migration of frequently accessed data elements across tiers of storage including flash.
Clustered systems	► Enable growth from the smallest configurations up to systems with 1,056 drives (up to 6 PB capacity).
Thin provisioning	► Supports business applications that need to grow dynamically, while using only the necessary space.
Unified storage	► Combines block and file storage requirements into a single system for simplicity and efficiency. ► Single user interface for managing both block and file storage streamlines administration.
IBM FlashCopy	► Creates a near-instant copy of active data that can be used for backup or for parallel processing activities; supports backup for recovering corrupted data.
Flash drives	► For applications that demand high disk speed, quick access to data and support for tiered storage environments.
IBM Active Cloud Engine®	► Reduces costs through policy-based management of files through use of tiered storage and improves data governance. ► Automates movement of less-frequently used files to lower-cost tiers of storage, including tape in an IBM Tivoli Storage Manager system. ► Automates deletion of unwanted or expired files. ► Caches files at remote locations to help improve performance and reduce networking cost.
IBM Tivoli Storage FlashCopy Manager	► Helps minimize the impact of backups, provides near-instant restore, shortens backup and recovery times, simplifies management and automates routine tasks.

Features	Benefits
IBM Metro Mirror and IBM Global Mirror	► Allows synchronous or asynchronous data replication between any Storwize family systems for maximum flexibility. ► Helps optimize use of network bandwidth with integrated Bridgeworks SANrockIT technology to reduce network costs or speed replication cycles, improving the accuracy of remote data.
IBM Real-time Compression	► Helps improve efficiency by storing up to five times as much active primary data in the same physical disk space;* by significantly reducing storage requirements, you can keep up to five times more information online, use the improved efficiency to reduce storage costs, or achieve a combination of greater capacity and reduced cost. ► Delivers unique hardware compression acceleration to help transform the economics of data storage.
Block data encryption	► Helps provide advanced protection of data at rest.

IBM Storwize V7000 architecture

The IBM Storwize V7000 solution incorporates some of the top IBM technologies that are typically found only in enterprise-class storage systems, which raises the standard for storage efficiency in midrange disk systems. This storage system extends the comprehensive storage portfolio from IBM and can help change the way organizations address the ongoing information explosion.

The Storwize V7000 solution provides a modular storage system that includes the capability to virtualize external SAN-attached storage and its own internal storage. The Storwize V7000 solution is built upon the IBM SAN Volume Controller technology base and uses technology from the IBM System Storage® DS8000 family.

An Storwize V7000 system provides several configuration options that are aimed at simplifying the implementation process. It also provides automated wizards, called *Directed Maintenance Procedures (DMP)*, to assist in resolving any events that might occur. An Storwize V7000 system is a clustered, scalable, and midrange storage system, and an external virtualization device.

Figure 3-57 shows the Storwize V7000 high-level overview.

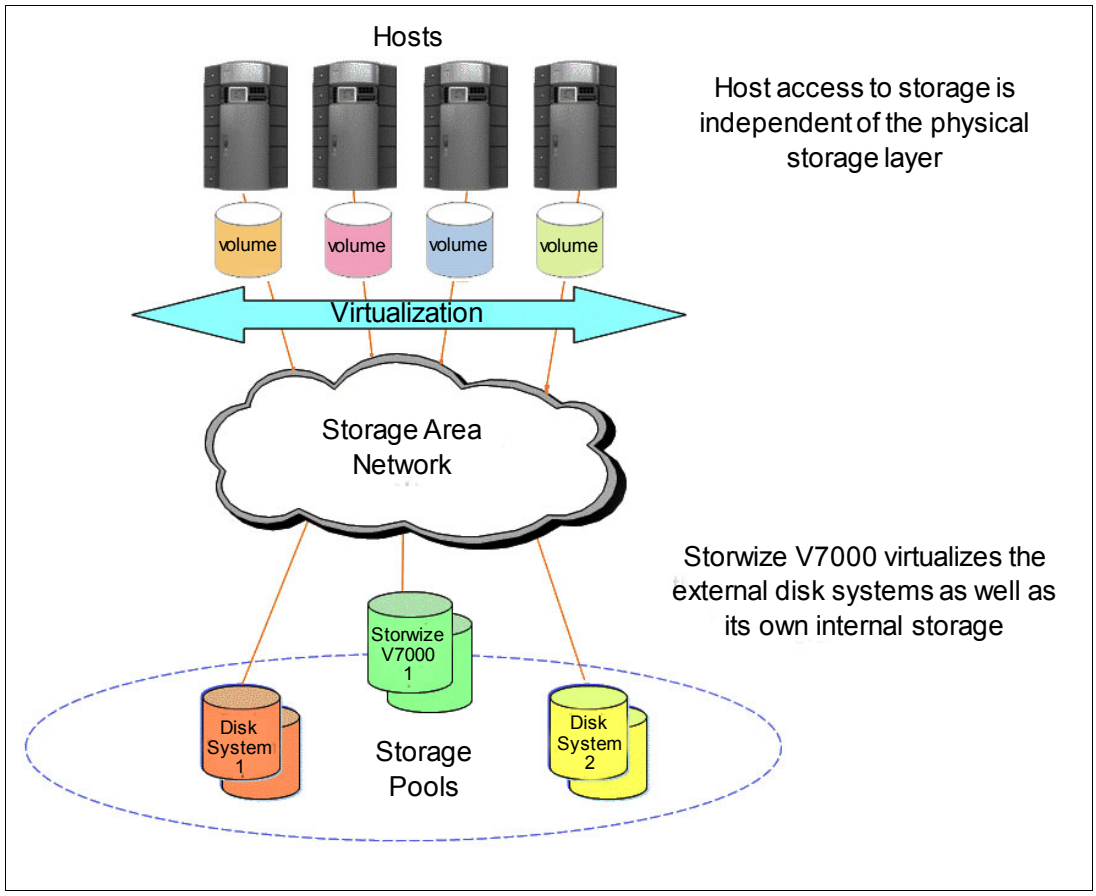


Figure 3-57 IBM Storwize V7000 overview

Included with an Storwize V7000 system is a graphical user interface (GUI) that enables storage to be deployed quickly and efficiently. The GUI runs on the Storwize V7000 system, so there is no need for a separate console. The management GUI contains a series of preestablished configuration options that are called *presets* and that use common settings to quickly configure objects on the system. Presets are available for creating volumes and IBM FlashCopy mappings and for setting up a RAID configuration.

The Storwize V7000 solution provides a choice of up to 480 x 3.5 inch or 960 x 2.5 inch serial-attached SCSI (SAS) drives for the internal storage in a clustered system and uses SAS cables and connectors to attach to the optional expansion enclosures. In a clustered system, the V7000 can provide 1.92 PiB capacity.

When virtualizing external storage arrays, an Storwize V7000 system can provide up to 32 PiB of usable capacity. A Storwize V7000 system supports a range of external disk systems similar to what the SAN Volume Controller supports today.

The Storwize V7000 solution consists of one to four control enclosures and, optionally, up to 36 expansion enclosures (and supports the intermixing of the different expansion enclosures). Within each enclosure are two canisters. Control enclosures contain two node canisters, and expansion enclosures contain two expansion canisters.

IBM Storwize V7000 models

The Storwize V7000 platform consists of enclosures and drives. Each enclosure contains two canisters that, although they can be replaced independently, are seen as part of the enclosure.

Additional IBM Storwize V7000 information: For the most up-to-date information about the features, benefits, and specifications of Storwize V7000 models, see this web page:

IBM Storwize V7000 Unified and Storwize V7000 Disk Systems

http://www.ibm.com/systems/storage/disk/storwize_v7000/index.html

The information in this book is valid at the time of writing, but as the Storwize V7000 matures, expect to see new features and enhanced specifications.

The Storwize V7000 models are described in Figure 3-58.

Model	Cache	FC / iSCSI / SAS ports	Drive slots	Power supply
2076-112 (with two node canisters)	16 GB	8x8 Gb / 4x1 Gb / 4x6 Gb	12 x 3.5-inch	Integrated dual power supplies with battery backup
2076-124 (with two node canisters)	16 GB	8x8 Gb / 4x1 Gb / 4x6 Gb	24 x 2.5-inch	Integrated dual power supplies with battery backup
2076-312 (with two node canisters)	16 GB	8x8 Gb / 4x1 Gb+4x10Gb / 4x6 Gb	12 x 3.5-inch	Integrated dual power supplies with battery backup
2076-324 (with two node canisters)	16 GB	8x8 Gb / 4x1 Gb+4x10Gb / 4x6 Gb	24 x 2.5-inch	Integrated dual power supplies with battery backup
2076-212 (with two expansion canisters)	Not applicable (N/A)	-- / -- / 4x6 Gb	12 x 3.5-inch	Integrated dual power supplies
2076-224 (with two expansion canisters)	N/A	-- / -- / 4x6 Gb	24 x 2.5-inch	Integrated dual power supplies

Figure 3-58 IBM Storwize V7000 models

Storwize V7000 attributes

For a complete and updated list, see “V7.2 Configuration Limits and Restrictions for IBM Storwize V7000” in the IBM Support Portal:

http://www.ibm.com/support/docview.wss?uid=ssg1S1004511

Storwize V7000 functions

The following functions are available with Storwize V7000:

- Thin provisioning (no license required)

  Traditional fully allocated volumes allocate real physical disk capacity for an entire volume even if that capacity is never used. Thin-provisioned volumes allocate real physical disk capacity only when data is written to the logical volume.

- Volume mirroring (no license required)

  Provides a single volume image to the attached host systems while maintaining pointers to two copies of data in separate storage pools. Copies can be on separate disk storage systems that are being virtualized. If one copy is failing, Storwize V7000 provides continuous data access by redirecting I/O to the remaining copy. When the copy becomes available, automatic resynchronization occurs.

- FlashCopy (included with the base Storwize V7000 license)

  Provides a volume level point-in-time copy function for any storage being virtualized by Storwize V7000. This function creates copies for backup, parallel processing, testing, and development, and have the copies available almost immediately.

  The Storwize V7000 includes the following FlashCopy functions:

  – Full or incremental copy

    This function copies only the changes from either the source or target data since the last FlashCopy operation and enables completion of point-in-time online backups faster than using traditional FlashCopy.

  – Multitarget FlashCopy

    Storwize V7000 supports copying of up to 256 target volumes from a single source volume. Each copy is managed by a unique mapping and, in general, each mapping acts independently and is not affected by other mappings sharing the source volume.

  – Cascaded FlashCopy

    This function is used to create copies of copies and supports full, incremental, or no copy operations.

  – Reverse FlashCopy

    This function enables data from an earlier point-in-time copy to be restored with minimal disruption to the host.

  – FlashCopy nocopy with thin provisioning

    This function provides a combination of using thin-provisioned volumes and FlashCopy together to help reduce disk space requirements when making copies.

  This option has two variations:

  – Space-efficient source and target with background copy, which copies only the allocated space

  – Space-efficient target with no background copy, which copies only the space used for changes between the source and target and is generally referred to as *snapshots*.

    This function may be used with multitarget, cascaded, and incremental FlashCopy.

- Consistency groups

  Consistency groups address the issue where application data is on multiple volumes. By placing the FlashCopy relationships into a consistency group, commands can be issued against all of the volumes in the group. This action enables a consistent point-in-time copy of all of the data, even though it might be on a physically separate volume.

  FlashCopy mappings can be members of a consistency group, or they can be operated in a stand-alone manner, that is, not as part of a consistency group. FlashCopy commands can be issued to a FlashCopy consistency group, which affects all FlashCopy mappings in the consistency group, or to a single FlashCopy mapping if it is not part of a defined FlashCopy consistency group.

- Metro Mirror (licensed is based on the number of enclosures and includes both Metro and Global Mirror)

  Provides a synchronous remote mirroring function up to approximately 300 kilometers between sites. As the host I/O only completes after the data is cached at both locations, performance requirements might limit the practical distance. Metro Mirror provides fully synchronized copies at both sites with zero data loss after the initial copy is completed.

  Metro Mirror can operate among multiple IBM Storwize systems.

- Global Mirror (licensed is based on the capacity being mirrored and includes both Metro and Global Mirror)

  Provides long-distance asynchronous remote mirroring function up to approximately 8,000 kilometers between sites. With Global Mirror, the host I/O completes locally and the changed data is sent to the remote site later. This function i maintains a consistent recoverable copy of data at the remote site, which lags behind the local site.

  Global Mirror can operate among multiple IBM Storwize systems.

- Data migration (no charge for temporary use)

  Storwize V7000 provides a data migration function that can be used to import external storage systems to the Storwize V7000 system. You can use it for these tasks:

  - Move volumes nondisruptively onto a newly installed storage system
  - Move volumes to rebalance a changed workload
  - Migrate data from other back-end storage to Storwize V7000 managed storage

- IBM System Storage Easy Tier (no charge)

  Provides a mechanism to seamlessly migrate hot spots to the most appropriate tier within the Storwize V7000 solution. This migration could be to internal drives within Storwize V7000 or to external storage systems that are virtualized by Storwize V7000.

  Easy Tier is another great technical optimization point in a SDE environment. As part of an ILM strategy, application workloads can be assigned to specific host volumes. The System Storage Easy Tier function is shown in Figure 3-59 on page 169.

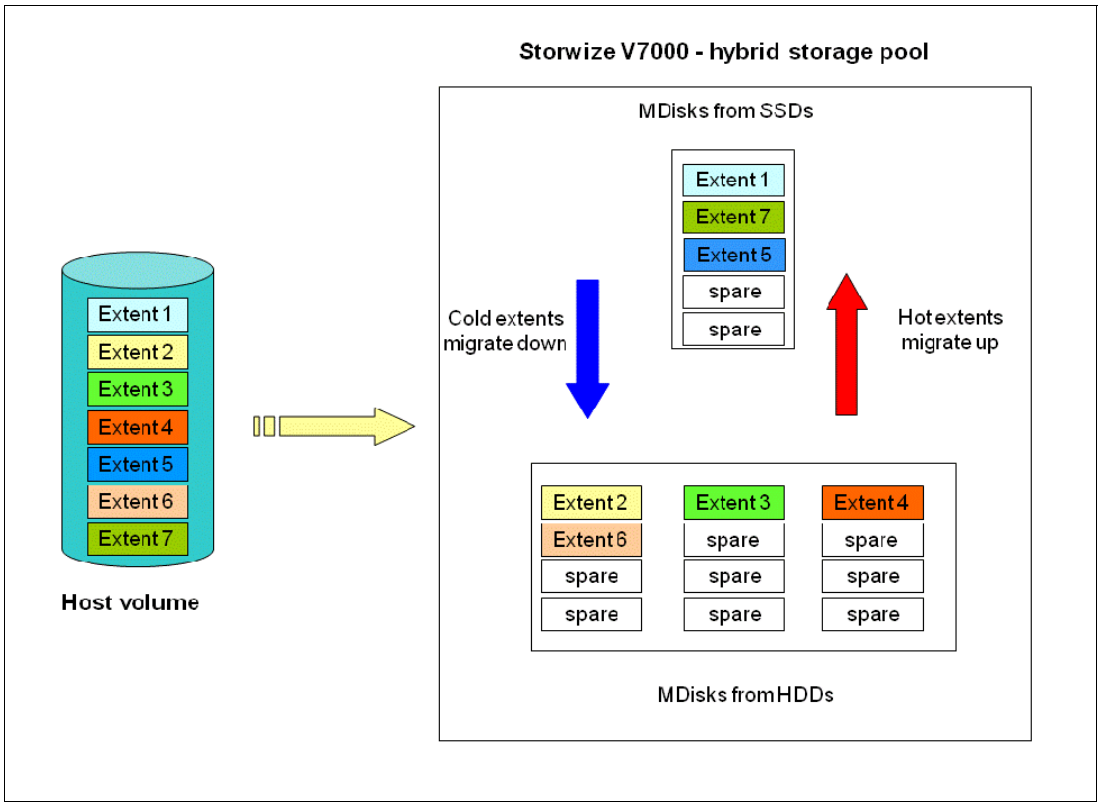


Figure 3-59 Easy Tier overview

IBM Storwize V7000 hardware

The Storwize V7000 solution is a modular storage system that is built on a common enclosure (control enclosure and expansion enclosure) that is compliant with Storage Bridge Bay (SBB).

Note: SBB is a specification that is created by a nonprofit working group that defines a mechanical and electrical interface between a passive backplane drive array and the electronics packages that characterize the array.

Figure 3-60 shows an overview of the hardware components of the Storwize V7000 solution.

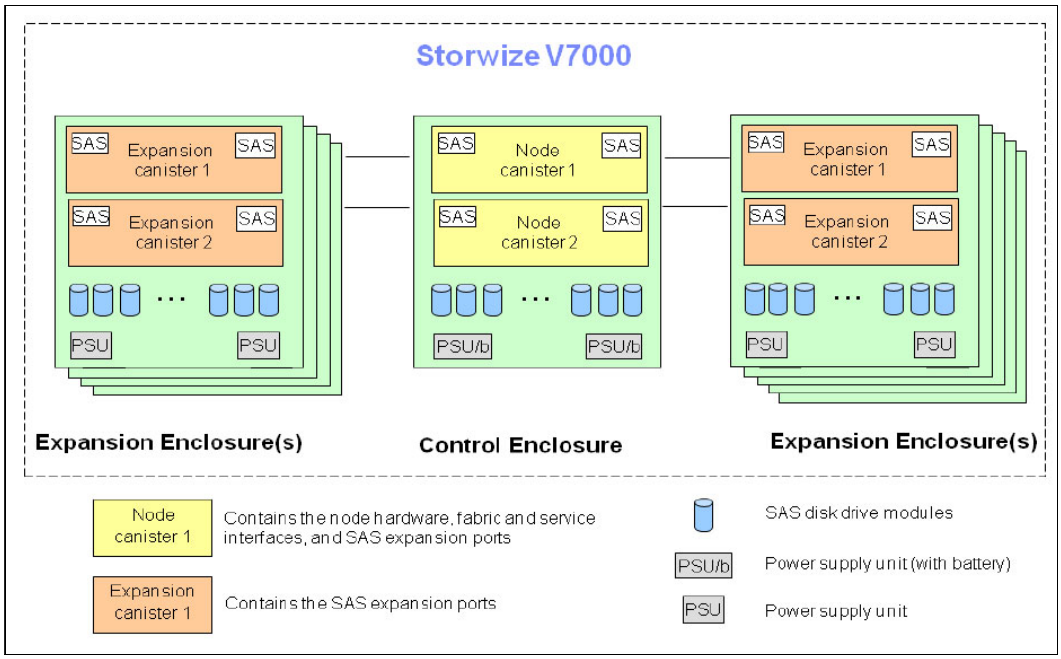


Figure 3-60 IBM Storwize V7000 hardware components

Note: For specifics about the configuration of SVC or Storwize V7000 in this SDE with OpenStack solution, see 4.11, “Storage configuration in IBM Cloud Manager” on page 357.

For more information, see the following web pages and publications:

- *Implementing the IBM Storwize V7000 V7.2*, SG24-2938

  http://www.redbooks.ibm.com/redpieces/abstracts/sg247938.html
- IBM Storwize V7000 Support page

  http://ibm.co/1nyBrTn
- IBM Storwize V7000 Unified and Storwize V7000 Disk Systems

  http://www.ibm.com/systems/storage/news/center/storwize_v7000/index.html
- V7.2.x Supported Hardware List, Device Driver, Firmware and Recommended Software Levels for IBM Storwize V7000

  http://www.ibm.com/support/docview.wss?uid=ssg1S1004450
- V7.2 Configuration Limits and Restrictions for IBM Storwize V7000

  http://www.ibm.com/support/docview.wss?uid=ssg1S1004511
- IBM Knowledge Center
  - IBM Storwize V7000

    http://pic.dhe.ibm.com/infocenter/storwize/ic/index.jsp
  - IBM Storwize V7000 documentation

    http://ibm.co/1QBwmvc
- Product support content for IBM Storwize V7000 (2076)

  http://ibm.co/1nxgxFF

- More publications about IBM Storwize V7000 are available on the IBM Redbooks website:
  http://www.redbooks.ibm.com/cgi-bin/searchsite.cgi?query=v7000

IBM Storwize V7000 learning videos on YouTube

Several helpful videos that describe the Storwize V7000 system are available on YouTube at the links that are listed in Table 3-7.

Table 3-7 Informational videos

Description of the video	Web address
IBM Storwize V7000 Storage Virtualization Terminology overview	http://www.youtube.com/watch?v=I2rzt3m2gP0
IBM Storwize V7000 Interface Tour	http://www.youtube.com/watch?v=FPbNRs9HacQ
IBM Storwize V7000 Volume Management	http://www.youtube.com/watch?v=YXeKqH8Sd9o
IBM Storwize V7000 migration	http://www.youtube.com/watch?v=dXxnUN6dk74
IBM Storwize V7000 Introduction to FlashCopy	http://www.youtube.com/watch?v=MXWgGWjBzG4
IBM Storwize V7000 Introduction to Remote Copy Part I: Creating Relationships and Consistency Groups	http://www.youtube.com/watch?v=koFYm--gnEc
IBM Storwize V7000 Introduction to Remote Copy Part II: Recovery and Return	http://www.youtube.com/watch?v=kMOFFS3NYas
IBM Storwize V7000 Rapid Application Storage Solution: Tivoli Storage FlashCopy Manager	http://www.youtube.com/watch?v=1zgEljRVDSc
IBM SAN Volume Controller and Storwize V7000 Performance Panel Sped-up! (HD)	http://www.youtube.com/watch?v=7noC71tLkWs
IBM Real-time Compression and Storwize V7000 and SAN Volume Controller demonstration	http://www.youtube.com/watch?v=rgKj75kn2JO
Storage efficiency with IBM Easy Tier	http://www.youtube.com/watch?v=aggudOKG9go

In the next section, we describe the DS8K storage system.

3.5.8 IBM System Storage DS8000 series (DS8K)

The IBM Enterprise class System Storage DS8000 series is designed for high performance, reliability, and flexibility and works in a range of open server operating environments and the IBM System z mainframe. The DS8000 includes a range of features that automate performance optimization and application quality of service, as well as provide high levels of reliability and system uptime. The DS8000 offers specialized advanced functions optimized for IBM Power Systems and IBM System z servers. The DS8000 also can use self-encrypting drives for every drive tier to help secure data at rest.[12] The DS8000 series is a great choice for SDE solutions with OpenStack and for enterprise private, hybrid, and public cloud solutions.

[12] "DS8000 disk encryption." IBM System Storage DS8000 Information Center. IBM. Retrieved 4 October 2012.

Summary of DS8000 series storage systems

Table 3-8 provides a historical summary overview of the DS8000 line. Each release of the DS8000 storage has resulted in improved capacity, performance, and resiliency for enterprise workloads. As seen below, the IBM POWER processor is a key architectural component to deliver optimal performance, and each release of the DS8K storage includes the latest advancements in Power Systems processor technology. IBM has continuously engineered and enhanced the DS8000 Enterprise Storage line with every release.

Table 3-8 Overview of the DS8000

DS8000 model	Key specifications and configuration options
DS8100[a]	► Dual 2-core IBM POWER5+ based controllers ► Can contain up to 384 drives ► Drives can be Fibre Channel or SATA
DS8300[b]	► Dual 4-core POWER5+ based controllers (based on Power 570 servers) ► Can contain up to 1024 drives ► Drives can be Fibre Channel or SATA
DS8700[c]	► Dual 2- or 4-core IBM POWER6 based controllers ► 3.5-in. Fibre Channel drives or solid-state disk ► Can contain up to 1,024 drives ► Drives can be 3.5” 15K RPM drives and enterprise flash drives
DS8800[d]	► Dual 2- or 4-core IBM POWER6+™ based controllers ► Can contain up to 1,536 drives ► 2.5" 10 K and 15 K RPM drives and SSD enterprise flash SAS-2 drives ► 3.5" Nearline-SAS drives
DS8870[e]	► Dual 2-, 4-, 8- or 16-core IBM POWER7 based controllers ► Since December 2013, based on POWER7+ ► Can contain up to 1,536 drives plus 120 1.8-inch flash cards in the High-performance Flash Enclosure (HPFE) ► 2.5-in. 10K and 15K RPM Drives and enterprise flash SAS-2 drives ► 3.5-in. Nearline-SAS drives ► High-performance Flash Enclosure: integrates and optimizes flash technology in the DS8870 ► Up to 4 flash enclosures per system: 48 TB raw per system ► High-performance Flash Enclosure fits into an existing DS8870 bay

a. “DS8000 disk encryption,” IBM Knowledge Center http://ibm.co/1DiTIwH
b. “DS8300 (Models 922, 932, 9A2, 9B2, 92E, 9AE),” IBM Knowledge Center http://ibm.co/1IJ7MTQ
c. “DS8700 (Models 941 and 94E),” IBM Knowledge Center http://ibm.co/1K5Jep3
d. “DS8800 (Models 951 and 95E),” IBM Knowledge Center http://ibm.co/1bjH2OW
e. “IBM System Storage DS8870 (Machine type 2422) Models 961 and 96E with two-year warranty,” IBM Announcement Letter, 3 October 2012 http://ibm.co/1d1646b

The newest offering in the IBM DS8000 line is the IBM DS8870. Smarter storage, such as DS8870, can help organizations spend less time managing storage and more time exploring ways to use data to grow their businesses. Quick and reliable data access is the driving force behind important data initiatives such as cloud, business analytics, and mobile computing, and the sixth-generation DS8870 system sets the standard for what an enterprise storage system should be with extraordinary performance, reliability and agility.

The DS8870 is the most advanced model in the DS8000 series and is equipped with IBM POWER7+ based controllers. Various configuration options are available that scale from dual 2-core systems up to dual 16-core systems with up to 1 TB of cache.

The DS8870 features an integrated high-performance flash enclosure with flash cards that can deliver up to 250,000 IOPS and up to 3.4 GBps bandwidth. A high performance all-flash drive configuration is also available. The DS8870 also features enhanced 8 Gbps device adapters and host adapters. Connectivity options, with up to 128 Fibre Channel or IBM FICON® ports for host connections, make the DS8870 suitable for multiple server environments in open systems and IBM System z environments.

The DS8870 supports advanced disaster recovery solutions, business continuity solutions, and thin provisioning. All disk drives in the DS8870 storage system have the Full Disk Encryption (FDE) feature. The DS8870 also can be integrated in a Lightweight Directory Access Protocol (LDAP) infrastructure.

The DS8870 can automatically optimize the use of each storage tier, particularly flash drives and flash cards, through the IBM Easy Tier feature, which is available at no extra charge.

As the latest model in the IBM DS8000 series, DS8870 can simplify your data architecture with self-optimizing performance for both random and sequential application workloads for the mainframe and an extensive variety of distributed server platforms. And with the agility and automation to manage all these workloads simultaneously, DS8870 can greatly reduce the challenge of managing your increasingly complex storage environment as your business requirements change.

To support your most demanding, business-critical applications, DS8870 combines rock-solid resiliency and intelligent flash performance to deliver consistent microsecond application response times.[13] The ability to select from all-flash, hybrid-flash, or traditional disk configuration options gives you the confidence needed to support the dynamic applications that sit at the heart of your business. To that end, DS8870 delivers a range of self-tuning features, such as intelligent caching algorithms, automated quality of service management, and advanced storage tiering, that can optimize data placement between the DS8870 and the enterprise servers attached to it. DS8870 also offers high availability and exclusive integration with IBM enterprise servers.

For organizations that are seeking an ideal combination of performance, high availability, resilience, and agility, DS8870 is a logical choice. The DS8870 is a high-performance, high-capacity, secure storage system designed to deliver the highest levels of performance, flexibility, scalability, resiliency, and total value for the most demanding, heterogeneous storage environments.

[13] Measurements taken from both IBM internal performance benchmark results and customer testimonials using the high-performance flash enclosure.

Table 3-9 is an overview of the key features and benefits of the DS8700 storage system.

Table 3-9 DS8700 features and benefits

Features	Benefits
Scalable all-flash and hybrid-flash options	Provides high flexibility to deploy flash performance for critical workloads. Delivers extreme performance for mission-critical applications with the high-performance flash enclosure.
Multiple IBM Power-based SMP controllers	Balances controller processors dynamically across I/O requests for balanced data throughput.
High-capacity cache with innovative caching algorithms	Helps optimize real-time workload requirements with dynamically self-tuning cache use.
Flexible configuration options with enterprise SAS, nearline SAS and ultra-fast flash storage	Helps support high-end consolidation and multitenant environments.
IBM Easy Tier	Automates placement of data across and within drive tiers to help optimize current workload requirements.
Storage Tier Advisor	Easy-to-use tool provides volume-based recommendations on where to deploy flash by analyzing current workload performance.
I/O Priority Manager (QoS manager)	Automates quality of service (QoS) management for all application volumes for effective consolidation and multitenant environments.
Thin provisioning	Reduces storage costs by automating the process of capacity provisioning and increasing capacity use.
Full Disk Encryption (FDE) drives	Protects sensitive information from internal and external threats.
IBM System z and IBM Power integration	Helps improve performance, efficiency and high availability for System z mainframes and for Power servers.
IBM FlashCopy	Helps ensure data recovery if of source data loss with point-in-time data copies for backup or replication. Enables data to be copied in the background while making both source and copied data available to users almost immediately.
BM FlashCopy SE	Provides a space-efficient snapshot capability that can greatly reduce the storage capacity needed for point-in-time copies.
IBM Tivoli Storage FlashCopy Manager	Provides application-aware FlashCopy and remote mirroring for popular enterprise applications.
IBM Metro Mirror, Global Mirror, Metro/Global Mirror, IBM z/OS Metro/Global Mirror and Global Copy	Creates duplicate copies of data at remote sites. Provides flexible options to support multisite business-continuity scenarios. Provides users access to data during planned and unplanned system and site outages.

Features	Benefits
Intuitive graphical user interface (GUI)	Simplifies and centralizes storage management, enabling more effective consolidation.
Support for Internet Protocol version 6 (IPv6) and US Government standards profile for IPv6	Supports next-generation Internet Protocol with increased security, addressing space and routing performance and services.

DS8870 architecture

The DS8870 Model 961 is a great storage system for enterprise private cloud, hybrid or even public cloud environments where the utmost is required in terms of performance, resiliency, and reliability. The DS8870 storage systems include the DS8870 Model 961 base frame and optional DS8870 Model 96E expansion frames. The DS8870 is available in three different configurations so that clients can tailor it to their specific performance and growth requirements.

As Figure 3-61 shows, the DS8870 is designed to support the most demanding business applications with its exceptional all-around performance and data throughput. The DS8870 architecture is server-based. Powerful IBM POWER7+ processor-based servers manage the cache to streamline disk I/O to maximize performance and throughput. These capabilities are further enhanced with the availability of high-performance flash enclosures and a high-performance All-flash configuration option.



Figure 3-61 DS8870 Enterprise Class and All-flash base frames

Combined with world-class business resiliency and encryption features, the DS8870 provides a unique combination of high availability, performance, and security. The DS8870 is equipped with encryption-capable disk drives. Encryption-capable solid-state flash drives (SSDs) and high- performance encryption-capable flash card storage are also available.

The DS8870 is tremendously scalable, has broad server support, and virtualization capabilities. These features can help simplify the storage environment by consolidating multiple storage systems. High-density storage enclosures offer a considerable reduction in footprint and energy use.

DS8870 includes a power architecture that is based on a Direct Current Uninterruptible Power System (DC-UPS). DC-UPS converts incoming AC line voltage to rectified AC, and contains an integrated battery subsystem. DC-UPS enables the DS8870 to achieve the highest energy efficiency in the DS8000 series. The DS8870 complies with the emerging ENERGY STAR specifications. ENERGY STAR is a joint program of the US Environmental Protection Agency and the US Department of Energy and helps save money and protect the environment through energy efficient products and practices. For more information, see this website:

http://www.energystar.gov

Using the IBM DS8870 in a SDE with OpenStack cloud environment

This section describes the use of the DS8870 in a cloud environment.

Business value

OpenStack simplifies deployment and makes DS8870 capabilities available for cloud environments in key ways:

- OpenStack automates the provisioning of high-end storage to a cloud environment.
- The storage in OpenStack cloud environments can be provisioned by using a self-service model that provides storage on an as-needed basis. Using this model, IBM clients get and pay only for what they really need, and the cloud provider does not have a situation where cloud storage space is wasted because of over-provisioning.
- Connection is just the first step. OpenStack can also potentially further integrate with advanced features, such as automated tiering with IBM Easy Tier or enablement of differentiated type of capabilities through quality of service (QoS).

With the availability of the IBM Storage Driver for OpenStack, the IBM DS8870 storage system can offer a range of capabilities that enable more effective storage automation deployments in to private or public clouds. Enabling OpenStack with DS8870 makes storage available whenever it is needed without the traditional associated cost of highly skilled administrators and infrastructure.

The IBM Storage Driver for OpenStack enables OpenStack clouds to access the DS8870 storage system. The driver integrates with the OpenStack environment and enables OpenStack applications to take advantage of the DS8870 enterprise-class storage.

Figure 3-62 shows the solution for integrating the DS8870 in an OpenStack infrastructure.

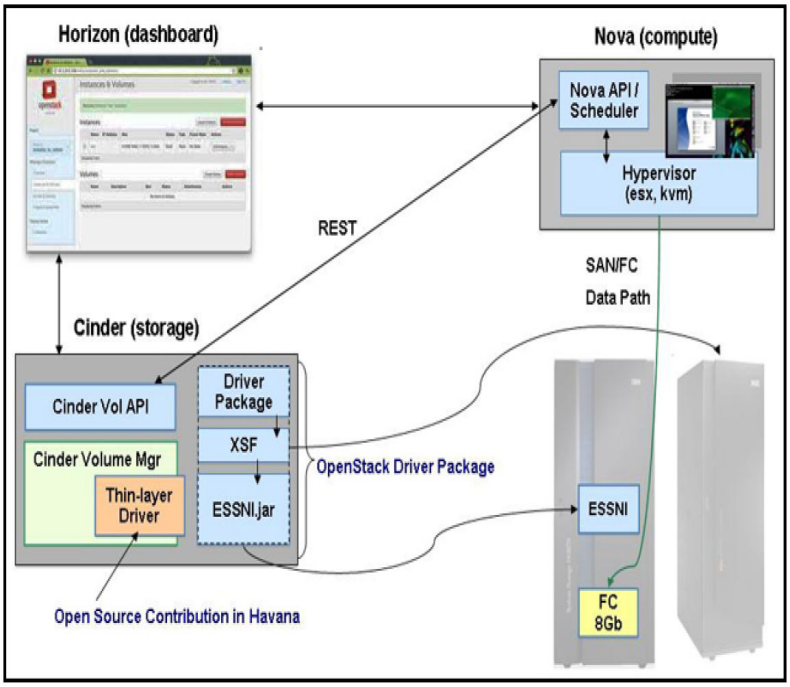


Figure 3-62 DS8870 and OpenStack infrastructure

As seen above, OpenStack includes the following major components:

- Glance (image) provides a catalog and repository for disk images that are mostly used in OpenStack Compute.
- Nova (compute) provides virtual machines (VM) or instances. Nova interacts with several OpenStack services, such as Keystone, Horizon, or Glance. The API process can upload and query Glance while the Nova compute service downloads images for use in starting images. Nova is the most complex and distributed component of OpenStack. Many processes are involved to handle user API requests and start virtual machines.
- Horizon (dashboard) provides a modular web-based user interface for all the OpenStack services. With this web GUI, you can perform most operations on your cloud. You can create volumes, start an instance, assign IP addresses, and set access controls.

> **Note:** In the IBM SDE solution, we have deployed IBM Cloud Manager with OpenStack, and as part of this software package, the IBM self-service portal OpenStack dashboard (IBM Cloud Manager 4.1.01). See “SDE reference architecture and the OpenStack” on page 63.

- Cinder (block storage) provides persistent block storage to guest VMs. This project was created from code that was originally in Nova (the nova-volume service). The Storage Driver for OpenStack is fully supported by Cinder 2012.11 and provides “block storage as a service” through iSCSI and Fibre Channel to VMs.

OpenStack Cinder was split from the Nova project to increase the focus on storage. Cinder's role is to take requests from the system through API calls and provision persistent storage to specific compute resources:

- Granularity is a LUN that is sized based on request.
- The LUN is persistent until it is deleted.
- Cinder is attached (mapped) to a single VM, and is not shared to multiple hosts.
- Cinder is control path only; I/O is run directly from the compute resources.

For more information about Cinder and the Storage Driver for OpenStack, see the Cinder page on the OpenStack wiki:

http://wiki.openstack.org/Cinder

Solution architecture

Figure 3-62 on page 177 shows the OpenStack architecture for DS8870. The OpenStack Cinder driver provides an infrastructure for managing volumes and is the interface to the DS8870. Because Cinder uses XSF, a gateway into the DS8870 ESSNI service, it can work with multiple IBM System Storage DS8000 storage systems.

The IBM self-service portal OpenStack dashboard provides a web-based user interface for managing OpenStack services for both users and administrators. The OpenStack Cinder driver attaches or uses extent pools only, so other workloads and data can also be on the same DS8870 in other pools (including CKD and other distributed data).

Infrastructure setup requirements

An OpenStack solution with a DS8870 storage system consists of the following components:

- One or more DS8870 storage systems to host the entire data set. Arrays, Ranks, and Extent Pools must be created.
- A storage area network (SAN) that is used to provide connectivity across application servers and storage systems.
- Network connectivity from OpenStack nodes to HMC.
- One or more fixed block extent pools to be used for OpenStack.

The OpenStack driver attaches or uses extent pools only, so other workloads and data can also be on the same DS8870 in other pools (including CKD and other distributed data).

The IBM Storage Driver for OpenStack package is available for download from the IBM Fix Central website:

http://www.ibm.com/support/fixcentral/

Configuration example

To illustrate the use of OpenStack with the DS8870, here is a configuration example. Assume that the DS8870 and an OpenStack environment are in place and that there is network connectivity between OpenStack nodes and the DS8870 HMC.

For this example, we have defined two Fixed Blocks extent pools with ID P1 and P2 for use with OpenStack. We create a stanza (shown in Figure 3-63 on page 179) called `ibm_ds8k_1` in the `cinder.conf` configuration file to configure access to the P1 and P2 pools in OpenStack. The parameters in the stanza help configure the driver and are used to initialize control the XSF and ESSNI communication to the DS8870.

```
[ibm_ds8k_1]
san_password = none
san_clustername = none
san_login = none
san_ip = none
xiv_ds8k_connection_type = fibre_channel
xiv_ds8k_proxy = xiv_ds8k_openstack.ds8k_nova_proxy.DS8KNovaProxy
san_is_local = False
debug = true
verbose = true
volume_backend_name = ibm_ds8k_1
xiv_ds8k_array_jar_lib_path = /home/user/xsf/jars
xiv_ds8k_array_java_path = /usr/bin/java
xiv_ds8k_array_storage_unit = IBM.2107-xxx
xiv_ds8k_vol_pool = P1,P2
xiv_ds8k_pass = test
xiv_ds8k_user = test
xiv_ds8k_address = 9.10.11.12
volume_driver = cinder.volume.drivers.xiv_ds8k.XIVDS8KDriver
```

Figure 3-63 DS8870 and OpenStack configuration example

The driver needs the following information:

- The IP address of the HMC (`xiv_ds8k_address`)
- The HMC login and password (`xiv_ds8k_user` and `xiv_ds8K_pass`)
- The storage system ID of the storage facility image that the HMC should control (`xiv_ds8K_storage _unit`)
- Various paths to the Java environment (`xiv_ds8k_array_jar_lib_path` and `xiv_ds8k_array_java_path`)
- The name of the OpenStack driver (`volume_driver`)
- DS8870 pools to use for creating the volume (`volume_back_end_name`) with OpenStack (`xiv_ds8k_vol_pool`)

Use scenarios

The OpenStack cloud connects to the DS8870 storage system over an iSCSI or Fibre Channel connection. Remote cloud users can issue requests for storage volumes from the OpenStack cloud. These requests are transparently handled by the Storage Driver. The driver communicates with the DS8870 Storage System and controls the storage volumes on it.

In addition to traditional enterprise-class storage technology, many organizations now have various storage needs with varying performance and price requirements. OpenStack has support for both object storage and block storage, with many deployment options for each depending on the use case:

- Cinder can migrate volumes between back ends.
- Users can initiate full backups of detached volumes to the OpenStack Swift Object store.
- Users can initiate incremental or differential backups of detached volumes.

Integration

The IBM Storage Driver for OpenStack is a software component that integrates with the OpenStack cloud environment and enables the use of storage resources that are provided by the following IBM storage systems:

- DS8870
- IBM XIV
- IBM SAN Volume Controller
- IBM Storwize V7000

Related publications and links

- *IBM System Storage DS8870: Architecture and Implementation*, SG24-8886

  http://www.redbooks.ibm.com/abstracts/SG248886.html
- *Using the IBM DS8870 in an OpenStack Cloud Environment,* TIPS1150

  http://www.redbooks.ibm.com/abstracts/tips1150.html
- OpenStack home page

  https://wiki.openstack.org/wiki/Main_Page
- OpenStack Cinder driver download

  https://github.com/openstack/cinder/tree/stable/havana/cinder/volume/drivers
- OpenStack Planet (blogs by OpenStack technical folks on OpenStack topics)

  http://planet.openstack.org
- IBM Offering Information page (search announcement letters, sales manuals, or both)

  http://pic.dhe.ibm.com/infocenter/strhosts/ic/index.jsp

In the next section, we describe an example of a three-tier commerce transactional workload that aligns with the SDE OpenStack solution architecture.

3.6 Examples

This section provides sample scenarios of using the IBM Software Defined Environment.

3.6.1 Three-tier commerce transactional workload

In this section, we describe an example from various user scenarios perspectives with an SDE-based solution in a hybrid cloud environment.

We explain the solution and deployment architecture of a mobile application to a hybrid cloud built on IBM Power Systems and running KVM by using OpenStack technologies incorporated into IBM products.

The key benefits for an enterprise are the ability to rapidly and consistently develop and deploy applications to the hybrid cloud. Easy management of the cloud infrastructure, using the suite of tools and products that we show in these examples, is also an important benefit.

For many of our enterprise clients, an off-premises isolated cloud is not an option because of their existing IT investments and in-house data. Therefore, a hybrid environment that connects in-house or outsourced systems with a cloud is very popular with these clients.

So on one side we have your traditional, back-office applications and workloads. We call these “systems of record.” If you are a retailer, think of your warehouse inventory records; if you are a healthcare provider, think of your existing patient records; if you are an insurer, your claims records. There are other traditional IT applications for customer relationship management and for finance and accounting systems. These workloads and IT expense are not going away. Many of our clients want to move these traditional workloads to a cloud-based environment for greater efficiency, flexibility, and access.

On the other side, we have newer “systems of engagement.” Think of the new mobile applications for things such as filing accident claims from your phone and so on. The development of these new mobile and social applications requires an infrastructure where they can quickly be developed, deployed, modified, and redeployed. Then, of course, you want to turn the data generated from these applications into insights that can be acted on. For a traditional enterprise, such as an insurance company, you need to be able to connect your data that is generated through the mobile application with existing customer data.

An example of how we’re helping a client do both is *UniCredit*. IBM signed a 10-year strategic sourcing agreement to help them rationalize their existing environment and using a private cloud for efficiency. Additionally, we are working with them on front-office projects such as using advanced analytics with cloud to monitor trends in the banking and IT industries and identify emerging disruptive technologies. Through a joint venture, we are also proactively marketing and delivering similar services to other financial services companies in Europe.

So value can clearly be obtained from both pools, but it is becoming clear that the highest value is derived when we bring the systems of engagement and systems of record data together to generate deeper “systems of insight.”

Acme Air solution

Figure 3-64 provides another illustration of four cloud users in a different SDE environment. Michael, the infrastructure administrator needs to manage the off-premises and on-premises cloud resources. Maureen, the DevOps pro needs to provision applications, and finally, Joe and Sally are the typical cloud users who interact with applications in different ways, as seen between and with the on-premises systems of record versus the off-premises systems of engagement.

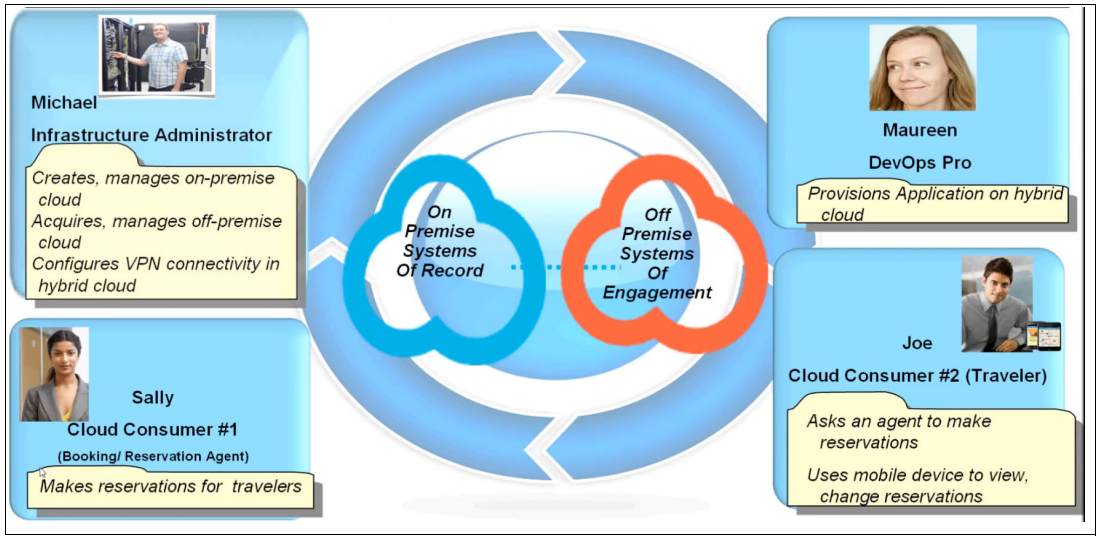


Figure 3-64 Hybrid cloud users - Commerce transaction

In this configuration, we have deployed the systems of record to an on-premises cloud running on KVM on IBM Power Systems. We also have a system of engagement deployed to an off-premises cloud running on SoftLayer.

We have chosen a sample airlines reservation system application called Acme Air for this set of transactional workloads. The sample Acme Air application is a built using IBM WebSphere eXtreme scale technology on the systems of record end. For the Systems of engagement end, it is built using the IBM MobileFirst™ server to develop and enable a mobile front end.

Figure 3-65 shows the solution architecture.

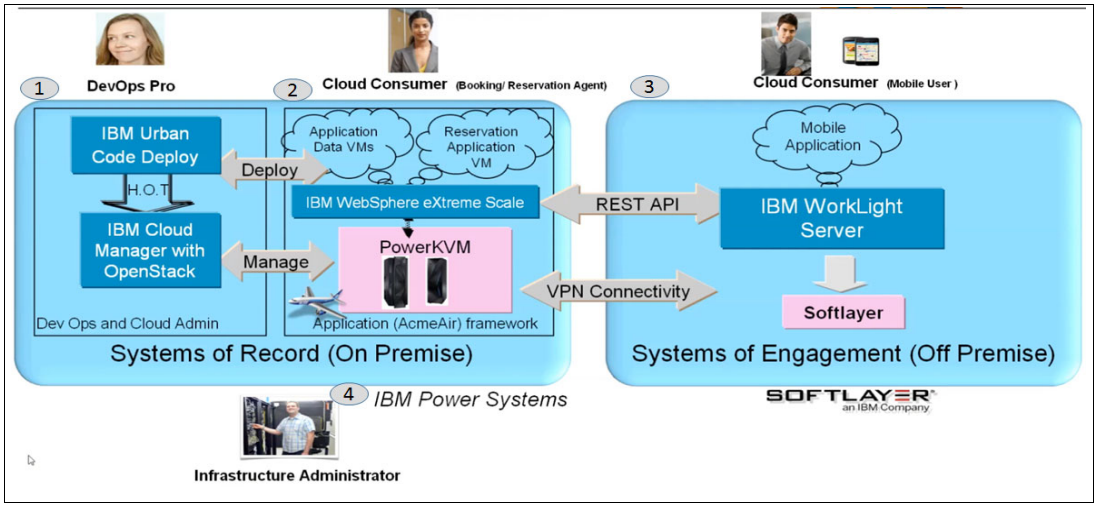


Figure 3-65 Three tier commerce solution architecture

In this solution architecture, we used IBM products that use Heat Orchestration Templates from OpenStack, such as IBM UrbanCode Deploy with Patterns and IBM Cloud Manager with OpenStack. IBM UrbanCode Deploy with Patterns is used to build blueprints of applications and deploy them to the hybrid cloud. IBM Cloud Manager is used to monitor and manage the cloud infrastructure.

The on-premises systems of record are built using IBM Power Systems and PowerKVM (Linux). Maureen will deploy her application data VMs and reservation application VMs to WebSphere to build and deploy the application using UrbanCode Deploy with patterns for running the application at peak performance using IBM WebSphere eXtreme Scale. IBM WebSphere eXtreme Scale is an elastic, scalable, in-memory data grid (IMDG). It is designed to help you handle exponential growth of transactions. With WebSphere eXtreme Scale, you can expect an enhanced quality of service in high-performance computing environments.

WebSphere eXtreme Scale provides a full set of IMDG functions. This approach provides a solution to scalability issues through caching and grid technology. The data grid dynamically caches, partitions, replicates and manages application data and business logic throughout multiple servers.

WebSphere eXtreme Scale integrates with other middleware products such as IBM WebSphere Application Server, IBM WebSphere Commerce and IBM WebSphere Portal Server, and supports big data, mobile and cloud computing.

WebSphere eXtreme Scale can perform massive volumes of transaction processing with high efficiency and linear scalability, and is definitely an added benefit in a SDE with OpenStack hybrid cloud environment. WebSphere eXtreme Scale is an essential tool for elastic scalability that provides these valuable benefits:

- Processes massive volumes of transactions with extreme efficiency and linear scalability.
- Rapidly builds a seamless, flexible, highly available elastic grid that scales out as applications scale, removing the performance limits of the database.
- Provides high availability and security with redundant copies of cache data, and authentication schemas that help ensure system security.
- Reduces total cost of ownership by enabling your existing backend systems to support significantly more applications.

  For more information about IBM WebSphere eXtreme Scale, see the WebSphere eXtreme Scale web page, including the Downloads tab:

  http://www.ibm.com/software/products/en/websphere-extreme-scale

  In addition, the HOT format is used and deployed for cloud instances (Glance) and infrastructure provisioning to the IBM Power Systems, which are managed by the Infrastructure Administrator (#4). The application framework for Acme Air is hosted on the Power Systems, and the application and data VMs are provisioned to Power Systems via UrbanCode Deploy with patterns.

6. The cloud user (in this case, the Acme Air Reservation/Booking Agent), accesses the reservation applications via the on-premises private cloud hosted on IBM Power Systems. This is secure, and built on-premises as a core system of record for data privacy protection for Acme Air's customers.
7. Mobile applications are built and deployed in the off-premises systems of engagement cloud in the highly secured SoftLayer cloud. The mobile applications that the cloud user accesses via his or her mobile device are built and deployed onto SoftLayer infrastructure via the IBM MobileFirst Server with mobile application patterns to accelerate deployment. Connection to the Acme Air reservations system deployed in the on-premises private cloud is made via a Representational State Transfer (REST) based API. Network connectivity between the on-premises private cloud and off-premises SoftLayer cloud is via a secure VPN connection.

For more information about IBM MobileFirst, see this web page:

http://www.ibm.com/software/products/en/worklight-foundation

You'll find downloads, white papers, analyst reports, and IBM Redbooks under the Downloads tab on this web page:

http://www.ibm.com/software/products/en/worklight-foundation

We describe this scenario further in 7.4, "Acme Air scenario" on page 668.

In the next section, we explain the IBM big data analytics vision and strategy, architecture, and give an example of big data SDE workloads and the most common use cases for big data and analytics.

3.6.2 Big data analytics workloads

Note: In this section, we provide detail regarding a big data analytics workload environment. As background, and for further preparation for this, we have also provided:

- An overview of what big data and analytics is, its impact in today's world and trends
- An explanation of how to get the most from big data and analytics from foundation starting points, from maturity, and from increasing value chain perspectives
- Overview of IBM big data and analytics portfolio and big data reference architecture
- Common big data use cases and workload examples

The traditional landscape for analytics

The traditional approach for analytics consisted of creating an analytical environment - *single version of the truth* that focused on delivering application independent and trusted insights to the business through a data warehouse. Traditionally, organizations implemented their data warehousing and analytical environment by sourcing structured data from their transactional, operational and third-party data systems. Batch, semi-batch or near-real-time processes extracted and sent data from the systems into a data ingestion process where data transformations were applied. The data was then loaded into relational repositories under a predefined schema to support the reporting and analytical needs of an organization. In this traditional approach, the analytical sources were created based on specific business needs. As new business requirements arose, a new data process was needed to generate the data and make it available for users or applications.

Challenges

The traditional landscape with a data warehouse for analytics has supported the analytical needs of many organizations very well for many years. However, in today's world, the technological evolution with its proliferation of new data from mobile devices and social networks requires new analytics for use cases such as retaining users who are now much more demanding, and easily and quickly influenced by social networks. In addition, the globalization of the economy challenges organizations to respond quickly to competition from new and different markets.

To overcome these challenges, organizations need to improve their business efficiency with better, faster and more informed decisions. They need to stay competitive by reducing cost and mitigating risk. Employees need access to better information and insights to implement new processes to achieve competitive advantage.

The traditional data warehousing and analytics systems need to evolve and adapt to these new business models to better support the analytical requirements of the organizations. Traditional IT landscapes are becoming too expensive to scale and maintain to meet the continuous exponential growth of data, both structured and unstructured. Data truly is driving the need for innovative and more cost effective solutions. The key challenges with the traditional analytical approach are:

- Cost
  - The data warehouse is a platform that stores all the "gold data" (trusted data) of an organization. It needs to scale to support large, complex workloads and to deliver information in a time manner. A data warehouse system is built from expensive hardware and software components and as the size of data and workloads increases the cost associate with the solution also increases.

- Another cost factor for data warehouse systems is running the data transformation using SQL language processes (ELT – Extract, Load and Transform) inside the expensive data warehousing platform to improve performance and scalability.
- An additional challenge with the traditional data warehouse is related to the complexity and management of the workload. It is challenging to deal with complex and mixed workloads on the same environment, therefore large organizations need to deploy multiple systems to offer adequate performance to their users. This increases the cost of the solution.

- Flexibility and limited exploratory capabilities:
  - Because of the nature of how data is stored and maintained in a data warehouse, it is hard to support the needs of the advanced users (data scientists, statisticians) to perform exploratory analytics or advanced analytics in a timely manner. The data is stored in a rigid schema so new data elements need to be modeled and transformed before being made available to the users. It is modeled for business intelligence and guided analysis, but not necessarily modeled appropriately for predictive analytics.
  - To keep size and cost manageable a data warehouse frequently stores only months of data and keeps data at some level of aggregation. However, the advanced users need to have access to more years of historical data and data in its natural, raw transaction level format.

A traditional data warehouse stores mostly structured data, but today's analytical requirements need to use new types of unstructured and semi-structured information. To overcome these challenges, many organizations started to build alternative platforms to perform analytics. One example of this is a data mining solution to support predictive analytics. Because the advanced users needed to have access to more detailed data (data in its natural form, as it was generated by the transactional systems), they started to extract data directly from the transactional systems and to build new platforms with detail data across a longer time period.

This alternative approach did not solve all the limitations of the traditional data warehouse. In fact, it can make the problems even worse. In many cases, this infrastructure became bigger than the traditional data warehouse environment. Governance and cost issues increased as the data and infrastructure multiplied within departments and was no longer centrally governed and managed by the IT organization. Productivity issues ensued as data and processes became duplicated everywhere requiring more resources to manage the systems. Furthermore, this alternative did not resolve the need to use new types of unstructured and semi-structured data for analytics.

From an architecture perspective, the challenge is to enable the new advanced analytical demands in an architecture that offers better governance, flexibility and agility while storing all the types of data needed for advanced analysis. The following sections of this document describe an architecture overview that meets the challenge by using big data and analytics technologies.

The new information landscape for big data and analytics

The big data technological evolution is causing a huge transformation in the business processes of many organizations. Automation is becoming a critical element to improve agility and efficiency. New and emergent technologies such as mobile devices and social networks are empowering users and completely transforming how they interact with business and services providers. In today's world, users quickly and easily exchange information among themselves. They express their sentiment and opinion about products and services, influence other users, and search for better, alternative offerings from anywhere, at any time.

Our highly instrumented and interconnected world is creating a huge impact on information management. The variety of the data sources is growing at a rapid rate. The generation of digital and machine data from social media interactions, rich media files, geospatial data, click-stream data, real-time sensor devices, and smart meters is growing much faster than ever before.

To deal with these new paradigms, organizations are forced to become more agile and innovative. Analytics capabilities are becoming a key competitive differentiator for organizations. The ability to have better knowledge and insight into customers is critical. Understanding customer needs and behaviors is a value to almost every organization in any industry (healthcare, retail, telco, travel, education, finance, food, and so on). This enables businesses to be more assertive in their investments, improve customer experience (better products and services), attract new customers, retain existing customers, and increase profitability.

In today's and tomorrow's business and personal worlds, four powerful forces are converging:

Social	The new ways we are all interacting around decisions.
Mobile	The ability to know what new value to put in people's hands.
Cloud	A way to innovate iteratively and with less investment, and
Big data and analytics	The way to turn all of the forces at work into competitive advantage. Between 2005 to 2020, the digital universe will grow by a factor of 300, from 130 to 40,000 exabytes (10^{18} bytes, or one billion gigabytes).

What is changing in the realm of data and analytics:

- Data is emerging as the world's newest resource for a competitive advantage.
- Decision-making is moving from the elite few to the empowered many.
- As the value of data continues to grow, current systems won't keep pace.

This leaves us with compelling reasons to act for the following key reasons:

- To outperform in your industry
- To manage risk
- To create IT agility to underpin the business

In today's competitive marketplace, executive leaders are racing to convert data-driven insights into meaningful results. Successful leaders are infusing analytics throughout their organizations to drive smarter decisions, enable faster actions and optimize outcomes. These are among the key findings from the 2013 IBM Institute for Business Value research study on how organizations around the globe are using key capabilities to amplify their ability to create value from big data and analytics.

An organization's ability to extract insights from this immense volume and variety of data and quickly deliver it to the appropriate personnel at the right time facilitates significant business opportunities and competitive advantage.

"Big data"—which means many things to many people—is no longer confined to the realm of technology. Today it is a business priority, given its ability to profoundly affect commerce in the globally integrated economy. In addition to providing solutions to long-standing business challenges, big data inspires new ways to transform processes, organizations, entire industries and even society itself. Yet extensive media coverage makes it hard to distinguish hype from reality to tell what is really happening. Our newest research finds that organizations are using big data strategically to target customer-centric outcomes, tap into internal data and build a better information ecosystem.

How you can act strategically:

- Build a culture that infuses analytics everywhere. Develop a curiosity-driven and evidence-inspired workforce. Infuse analytics into everything employees touch.
- Be proactive about privacy, security, and governance. Adopt forward-thinking approaches to maximize impact yet balancing risk.
- Invest in a big data and analytics platform, and design and build against a master plan. Cover all types of data and all types of analytics to cover a full range of business outcomes.

3.6.3 The IBM business analytics and optimization capabilities

As an organization moves up the value chain of a business intelligence solution, they move from collecting and reporting data, to integration, to analytics and finally to actionable embedded analytics, as shown in Figure 3-66. Using predictive analytics and embedding it into business processes makes analytics pervasive across the enterprise and empowers many more employees to make better and informed decision and results in a more effective and competitive business.

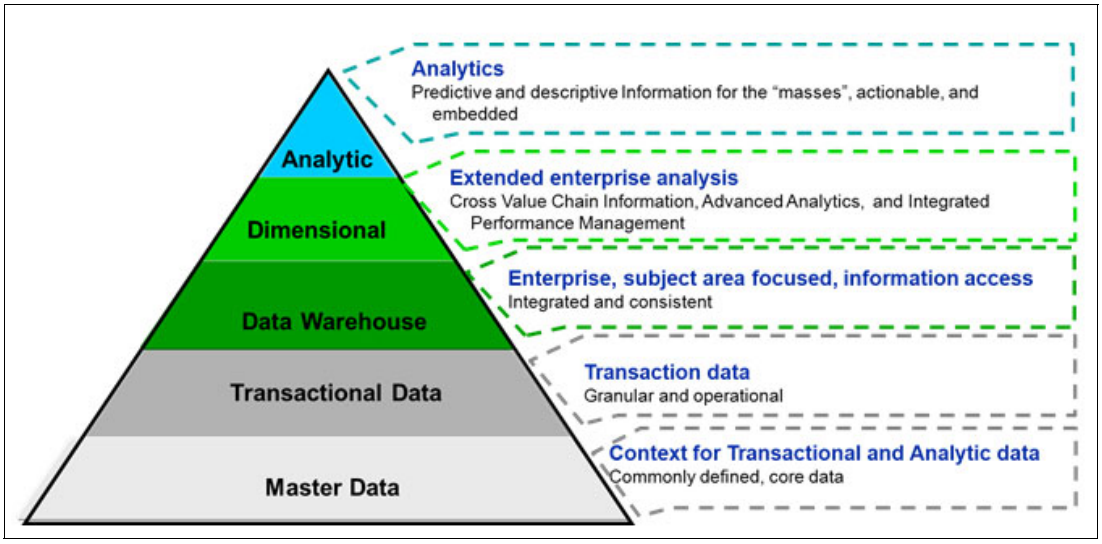


Figure 3-66 Business analytics and optimization capabilities value chain

The IBM Business Analytics and Optimization high-level architecture overview

The Business Analytics and Optimization (BAO) reference architecture is the framework that ties all of IBM industry data models, data integration models and business solution templates together from a business and functional perspective. It describes the major foundational components of an end-to-end business analytics architecture solution.

The BAO reference architecture is evolutionary in nature, but it provides key benefits by developing a common language and methodology over time. The BAO reference architecture:

- Describes the major foundational components or candidate building blocks of an end-to-end target architecture.
- Spans all industries and all solution areas.
- Forms a common language for describing architectural components.

- Offers a framework for scope identification, roadmap definition, risk assessment, and gap assessment.
- Provides a common framework for conceptualizing the problem area.

Figure 3-67 illustrates the core architecture components of the IBM Business Analytics and Optimization reference architecture.

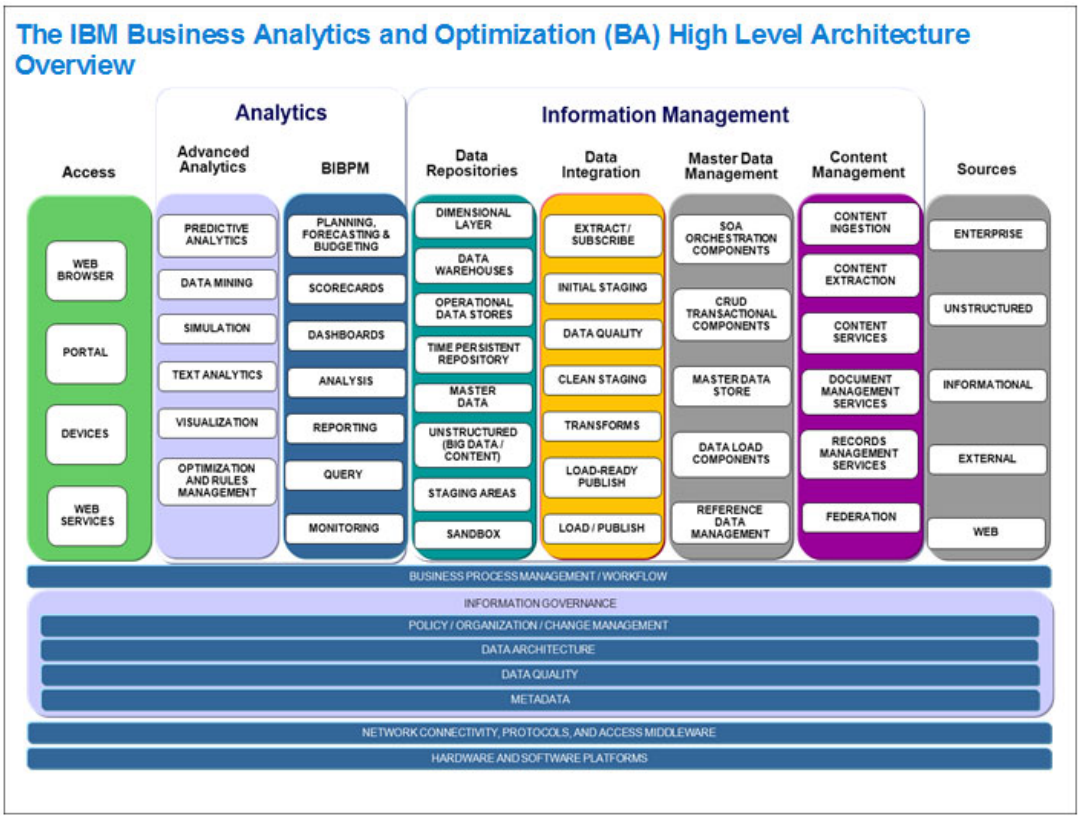


Figure 3-67 Business Analytics and Optimization high-level architecture overview

Access layer

The Access layer provides the mechanism for user or external system interaction with the analytical, informational, and reporting applications of the IBM Information on Demand® environment. The interaction may be human or automated through a web service.

Advanced Analytics layer

The Advanced Analytics layer uses data and models to provide insight to guide decisions. Advanced analytics empowers clients to make decisions that are more effective. This layer builds productive systems based on complete data, consideration of all available options, and careful predictions of outcomes and estimates of risk. The Advanced Analytics layer is typically composed of various technological components destined to meet specific needs, and is usually built from top software and tools, such as IBM SPSS and IBM ILOG®.

Business Intelligence Business Performance Management layer

The Business Intelligence Business Performance Management (BIBPM) layer empowers decision making and improved business performance through the timely access, analysis and reporting of actionable, accurate, and personalized information. A variety of applications may be supported, from static reporting, to balanced scorecards, to monitoring tools that are embedded within an operational process. This layer is typically composed of various technological components destined to meet specific needs, and is usually built from the best software and tools, such as data access models, query tools, reporting tools for online analytical processing (OLAP).

Data Repository layer

The Data Repository layer contains the databases, data stores and related components that provide most of the storage for the data needed to support a Business Analytics and Optimization (BAO) environment. The Data Repositories layer's repositories are not a replacement or replica of operational databases that reside on the Data Source layer, but are a complementary set of data repositories that reshape data into formats necessary for making decisions and managing a business. These database structures are represented by conceptual, logical, and physical data models and data model types (for example, third normal form, star/snowflake schemas, unstructured, and so on).

A semantic component sits on top of a high normal form (HNF) data store and serves to make it more useful by defining views, materialized views, and propagated (materialized) structures.

This is the application-specific, business user interface of the data warehouse environment. The key benefits of the semantic component are:

- Reduces physical storage requirements
- Reduces redundancy by using views as opposed to physical materialization
- Reduces data latency by eliminating downstream ETL processes
- Isolates the impact to downstream systems of data warehouse updates to a set of views as opposed to ETL systems
- Provides capability to build multipurpose structures for broader use
- Ensures dependency upon the underlying HNF data store
- Allows for the creation of a flexible, yet more complex, third normal form physical model by exposing the data to the user in a business friendly manner
- Allows for performance tuning based on query and data access patterns
- Ensures data visualization to be valid through connecting directly to the enterprise data warehouse (EDW)
- Allows for direct enablement of the OLAP capability in the scaled-out EDW
- Brings the applications to the data, rather than the data to the applications

Data Integration layer

The Data Integration layer focuses on the processes and environments that deal with capture, qualification, processing, and movement of data to prepare it for storage in the Data Repository layer, which is subsequently shared with the Analytical and Access applications and systems. This layer may process data in scheduled batch intervals or in near real-time or “just-in-time” intervals, depending on the nature of the data and the business purpose for use.

Master Data Management layer

The Master Data Management, or MDM, is a set of disciplines, technologies, and solutions to create and maintain consistent, complete, contextual, and accurate business data for all stakeholders across and beyond the enterprise.

Content Management layer

The Content Management layer contains the services, technologies and processes used to capture, manage, store, preserve, and deliver unstructured content. It provides global access and management of digital assets used to collaborate and share information between a company and its customers, suppliers, employees and business partners.

Cross-functional layer

The Cross-functional layer architecture provides the cohesive foundation whereupon the BAO pillars can function in an integrated and seamless fashion. This layer handles several functions:

- BPM and workflow

  Business process and workflow management is based on continually improving and optimizing business processes, using business value and technology as guides and using a systematic approach for continuous improvement.

- Information governance

  Information governance is the orchestration of people, processes, and technology to enable an organization to use data as an enterprise asset.

- Policy

  Policy provides strategic and operational direction to the enterprise. Policy recognizes that corporate data is a critical corporate resource and will be managed as such. Policies are intended to guide and affect real world decisions by placing limitations on a particular decision or process.

- Organization

  The administrative and functional structures required to achieve the objectives of the Organization’s program. Included in this module are the executive, management, and operational organizational structures and stewardship roles.

- Change management

  Change Management activities provide the strategy for the creation and distribution of meaningful and appropriate communication of messages on required changes to program stakeholders, including executives, as well as program and project level communities.

- Data architecture

  Includes the data, application, and technology structures needed to support BAO processes and improve information management capabilities.

- Data quality

  Data quality management is defined by how effectively the data supports the data quality requirements of the transactions and decisions needed to meet an organization's strategic goals and objectives, as embodied in its ability to control its assets and conduct its core operations.

- Metadata

  Metadata management is the capture, versioning, approval, use, and analysis of the different types of metadata found in an Information Management environment.

- Network, connectivity, protocols, and access middleware

  The platform and technology that allows communication between components, formal description of message formats and the rules for exchanging those messages is provided. Security plays an important role in this sub-layer ensuring that the appropriate controls and accesses are in place and continuously monitored. Data storage and network connectivity is the foundation for the BAO platform that is needed to build the specific applications that will allow the suite of applications to interact with the user community.

- Hardware and software platforms

  The physical platforms for a BAO environment include the hardware components and the software applications that run on the platform. The infrastructure includes the physical hardware, foundational software such as operating systems, and security protocols. A key consideration in the BAO arena is the development of workload optimized systems as provided and specified by Software Defined Environments and appliances that are designed and built for specific purposes. IBM PureData™ Systems for Operational Analytics (PDOA) and IBM PureData Systems for Analytics (PDA) are good examples.

Big data requires a new view on business intelligence, data management, governance, and delivery. So now that we have explained the capabilities and high-level BAO framework, you can get ready to gain the benefits big data and analytics:

- Big data exploration

  Find, visualize and understand big data to improve decision making.

- Enhanced 360-degree view of the customer

  Extend existing customer views by incorporating additional internal and external information sources.

- Security and intelligence extension

  Reduce risk, detect fraud and monitor cyber security in real time.

- Operations analysis

  Analyze a variety of machine data for improved business results and operational efficiency.

- Data warehouse augmentation

  Integrate big data and traditional data warehouse capabilities to gain new business insights while optimizing existing warehouse infrastructure.

The Four Vs of big data

From traffic patterns to music downloads, to web history and medical records, data is recorded, stored and analyzed, to enable the technology and services that the world relies on every day. But what exactly is big data, and how can these massive amounts of data be used? As a leader in the sector, IBM data scientists break big data into four dimensions: *Volume*, *velocity*, *variety*, and *veracity*. The following examples of metrics and the impact of the 4 Vs are relevant:

- Volume (scale of data):
  - It is estimated that 2.5 quintillion bytes (2.3 trillion gigabytes) are created each day.
  - 40 zettabytes of data will be created by 2020; and increase of 300X since 2005.
  - In a world that has a population of 7 billion people, 6 billion people have cell phones.
  - Most companies in the US have at least 100 terabytes (100,000 gigabytes) of data stored.
- Velocity (analysis of streaming data):
  - The New York Stock Exchange captures 1 tetrabyte of trade information during each trading session.
  - By 2016, it is projected that there will be 18.9 billion network connections, which is almost 2.5 connections per person on Earth.
  - Modern cars can have close to 100 sensors that monitor items such as fuel level and tire pressure.
- Variety (different forms of data)
  - As of 2011, the global size of data in healthcare was estimated to be 150 Exabytes (150 billion gigabytes).
  - 30 billion pieces of content are shared on Facebook each month.
  - By 2014, it's anticipated there will be 420 million wearable wireless health monitors.
  - 4 Billion + hours of video are watched on YouTube each month.
  - 400 million tweets are sent per day by about 200 million monthly active users.
- Veracity (uncertainty of data)
  - 1 in 3 business leaders don't trust the information they have and use to make decisions.
  - Poor data quality costs the United States economy around $3.1 Trillion a year.
  - 27% of respondents were unsure of how much of their data was inaccurate

It is also beneficial to further look at your data and identify its classification or taxonomy in context to big data and its characteristics as shown in Figure 3-68 on page 193.

Volume	Velocity	Variety	Veracity	Horizontal Scalability ?	SQL limitation?	Big Data?	Comments
No	No	No	Yes	No	No	No	Data validation is required
No	No	No	No	No	No	No	Data validity/ lineage / provenance needs to be established
No	No	Yes	No	Maybe	Maybe	Maybe	Data validity + non-relational data store is required
No	Yes	No	No	Maybe	No	Maybe	Continuous Ingest capability is required
Yes	**No**	**No**	**No**	**Yes**	**Maybe**	**Type 1**	**Traditional DW**
No	No	Yes	Yes	Maybe	Yes	Type 3	Non relational data repository is required
No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Type 3,4**	**Non relational data repository with continuous data ingest**
Yes	Yes	No	No	Yes	Maybe	Type 1,4	Continuous ingest with horizontal scalability of relational data repository with data validation
Yes	Yes	Yes	No	Yes	Yes	Type 1,3,4	Continuous ingest with horizontal scalability of non relational data repository
Yes	Yes	No	Yes	Yes	Maybe	Type 1,4	Continuous ingest with horizontal scalability of relational data repository
Yes	Yes	Yes	Yes	Yes	Yes	Type 1,3,4	Continuous ingest with horizontal scalability of non relational data repository with data validation
Yes*	Yes*	Yes*	No	Yes	Yes	Type 2,4	Highly recursive social data with continuous ingest, non-relational data store and data validation

Figure 3-68 Big data taxonomy

Depending on the industry and organization, big data encompasses information from multiple internal and external sources such as transactions, social media, enterprise content, sensors, and mobile devices. Companies can use their data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue. By 2015, 4.4 million IT jobs will be created globally to support big data, with 1.9 million in the United States.

So, this leads to several questions that need answers:

- Is your architecture and technology ready for this onslaught of big data that is already occurring?
- How can you best deal with this onslaught, in terms of technology and platform choices?
- How can you use existing investments to at least try to keep pace?
- How can investments be accelerated and optimized without “ripping and replacing,” which takes time and a lot of money. in many cases?

Well, through Software Defined Environments, of course. SDEs can use existing infrastructure and middleware investments to build upon as a starting building block. When coupled with a SDE, the IBM big data and analytics portfolio helps our clients meet the most demanding needs in today’s data intensive landscape and effectively manage the Four Vs for competitive advantages.

4

Cloud provider installation guidelines and implementation scenarios

This Software Defined Environments (SDE) implementation focuses on IBM Cloud Manager architecture for deploying and managing an OpenStack cloud.

This chapter presents the Cloud Manager installation and integration with different types of hypervisors (Compute nodes), network configuration, and integration with storage components (including software-defined storage). It includes Cloud Manager images and instances operations, deployment and use of IBM Platform Resource Scheduler (PRS), and steps to enable the self-service portal. It then provides the IBM UrbanCode Deploy with Patterns installation instructions.

This chapter includes the following sections:

- 4.1, “Structure of the IBM Cloud Manager installation” on page 196
- 4.2, “Cloud installation” on page 198
- 4.3, “Cloud setup for this publication” on page 198
- 4.4, “Deployment server installation” on page 202
- 4.5, “Deploy the Controller node” on page 218
- 4.6, “Add KVM Compute nodes to the cloud” on page 236
- 4.7, “Add PowerKVM Compute nodes to the cloud” on page 253
- 4.8, “Add PowerVM Compute nodes to the cloud” on page 274
- 4.9, “Enable the self-service portal” on page 311
- 4.10, “Network configuration in IBM Cloud Manager” on page 326
- 4.11, “Storage configuration in IBM Cloud Manager” on page 357
- 4.12, “Manage images in IBM Cloud Manager” on page 412
- 4.13, “Manage instances in IBM Cloud Manager” on page 443
- 4.14, “Manage flavors in IBM Cloud Manager” on page 480
- 4.15, “Use SSH key pairs in IBM Cloud Manager” on page 485
- 4.16, “Developer tool installation: UrbanCode Deploy with Patterns” on page 492

© Copyright IBM Corp. 2015. All rights reserved.

4.1 Structure of the IBM Cloud Manager installation

Cloud Manager provides the required tools, mechanisms and features to create, manage and operate your cloud. It can be used by a cloud service provider (CSP) or to set up an on-premises private cloud, using your own infrastructure.

The Cloud Manager installation package contains the tools to deploy your OpenStack cloud, including the Controller and different types of Compute nodes, that can be integrated under the Controller node management.

The installation of the Cloud Manager and deployment of your cloud is based on multiple nodes using Chef server or client architecture. The Cloud Manager must be installed in one Linux host, which will be your Deployment Server node. This node will host the Chef server, which is the tool that contains all necessary templates (cookbooks) and mechanisms (such as knife command) to install packages in your Compute nodes (which will be Chef clients). Chef server will manage all the configuration steps in these client nodes to make them available for a quick deployment of instances in your cloud.

Note: In the Cloud Manager environment, virtual machines are called instances.

Chef server can be described as an automation tool that installs and configures products across different servers in a cloud. It improves the efficiency of the cloud deployment primarily by avoiding configuration mistakes made manually. It contains three major components:

- Chef server
- Chef workstation (which, in the Cloud Manager installation, also runs in the deployment server along with the Chef server)
- Chef nodes (or clients)

The Chef server uses cookbooks with recipes to complete certain actions that assign roles to the nodes and defines their attributes. For example, an action such as the nova, neutron, cinder, and openvswitch installation and configuration in a KVM node, enables it to be part of the OpenStack cloud. This action will assign a Controller or Compute role to the node and Chef will execute specific configurations accordingly on these nodes to create a cloud. We work with the Chef server using its command-line interface, called knife. By using the knife command with the specified environment and topology configuration files, the Chef server will automate the installation and configuration actions in the nodes and deploy your cloud.

Figure 4-1 on page 197 provides an overview of the Cloud Manager installation and how it uses Chef to communicate with your Controller and Compute nodes to easily deploy your cloud.

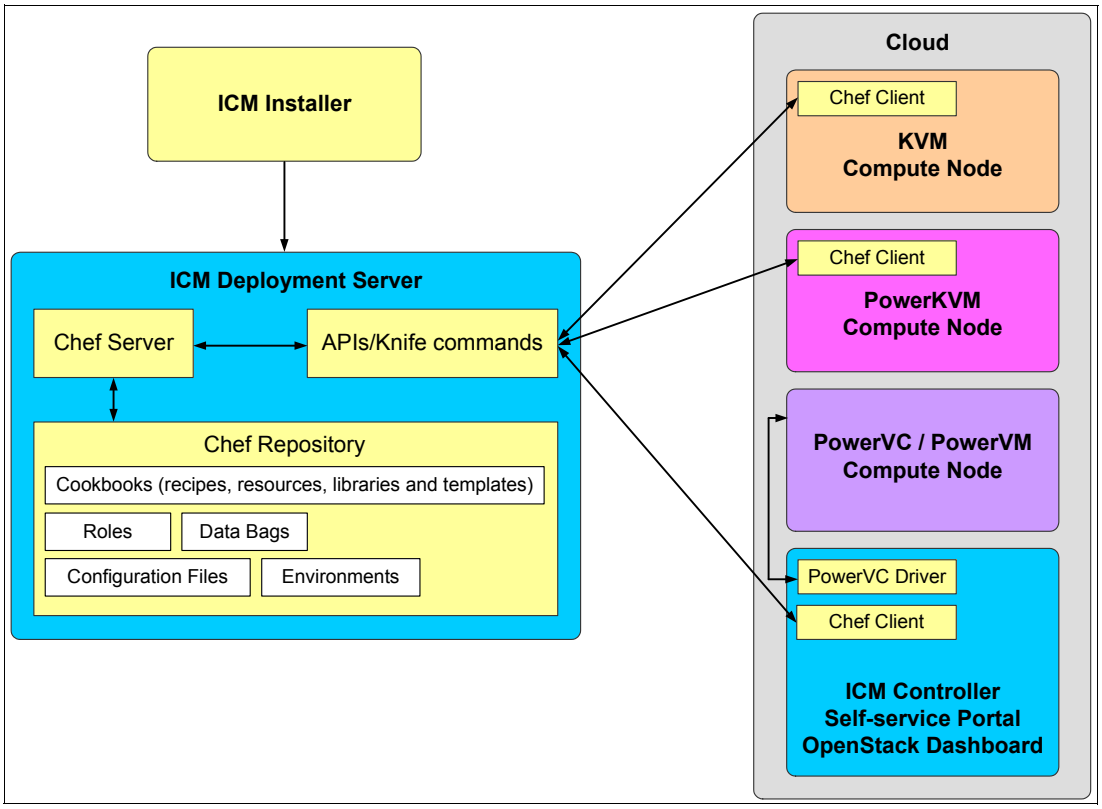


Figure 4-1 The Cloud Manager Chef components

IBM Cloud Manager is installed on the deployment server which contains the Chef server, the cookbooks and recipes in the Chef repository. It will use **`knife`** commands to apply templates on the nodes. With these templates and recipes, the deployment server will once more use **`knife`** commands across the Controller and Compute nodes to automatically perform installation and configuration without manual intervention on the Compute nodes. This involves installing Chef client and Cloud Manager components (Nova, Glance, Cinder, Keystone, and Neutron) in the nodes (Controller and Compute nodes), and configuring them. Most of the configuration of your cloud is accomplished by changing the files in the deployment server and using the knife commands to distribute the information across the nodes, including nova, cinder and neutron configuration files.

Note: In some specific configurations, it might require manual modification of the component configuration file inside the Controller or Compute nodes. For example, to configure multiple storage backends for cinder, you need to manually modify the /etc/cinder/cinder.conf file in the Controller node.

For more information about Chef, visit the All About Chef website:

`http://docs.getchef.com/`

You can also read more details about the Cloud Manager installation structure in IBM Cloud Manager technical overview in the IBM Knowledge Center:

`http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_techoverview.html`

4.2 Cloud installation

The Cloud Manager implementation consists of installing the deployment server and then using it to deploy your cloud, including the Controller and Compute nodes.

There are three different topologies you can use to deploy your cloud. These topologies are explained in Table 4-1.

Table 4-1 IBM Cloud Manager possible topologies

Topology	Description
Minimal	For evaluation only. This means that a single host can run the Deployment Server, Controller and Compute node. In this case the Compute node can only be a KVM host.[a]
Controller + n Compute nodes	This is the most common scenario, where you have one Deployment Server, one Controller node and as many Compute nodes you want, including different technologies such as KVM, PowerKVM, PowerVM, Hyper-V and z/VM.[b]
Distributed database	This topology is similar to the Controller + n Compute nodes, with the difference that the database service for Cloud Manager will run in a separate node.

a. In the minimal topology you also have an option of running the deployment server on a separate node, keeping only Controller and Compute function in a single host.
b. In this topology, the deployment server and Controller node are two separate servers that can run on Intel x86_64 Linux or PowerLinux (on POWER6, POWER7 or POWER8).

Note: In this IBM Redbooks publication, we used the Controller + n Compute nodes topology. The deployment server and the Controller node are running PowerLinux on 2 logical partitions in a POWER7 server. Additionally, we also showcase an all-in-one topology, where the deployment server is running in a VMware virtual machine and the Controller node is also a Compute node running Red Hat Linux with KVM (QEMU).

4.3 Cloud setup for this publication

We performed the product installation, configuration and demonstration on the infrastructure represented in Figure 4-2 on page 199. The residency team created a hybrid cloud, containing POWER7 servers, POWER8 servers, System x servers, SAN Volume Controller storage and GPFS cluster. This section provides instructions and examples on how to implement Cloud Manager in different servers, enabling the readers of the IBM Redbooks to implement a cloud in hybrid or homogeneous environments.

This hybrid cloud, allows us to enable developer tools such as the UrbanCode Deploy and UrbanCode Deploy with Patterns and use these applications to easily deploy complex environments, such as the InfoSphere BigInsights. Other important IBM and non-IBM applications can also be easily deployed in the cloud after corectly setting up images containing the products, including IBM DB2®, IBM Informix®, WebSphere Application Server, WebSphere Message Broker, InfoSphere DataStage, Tivoli Storage Manager, IBM HTTP Server, Oracle Database, SAS, SAP, and so on. With multiple hardware architecture available in the cloud, we can use them to deploy applications running on top of several operating systems, including: Linux on Power (several distributions), Linux x86_64 (several distributions), Windows and AIX.

During this residency, we explored multiple cloud options to demonstrate how the Software Defined Environments (SDEs) can bring benefits to your environment. This section explains how to implement some of the components that will provide a SDE, especially with Cloud Manager. We also show the installation process of Developer tools such as UrbanCode Deploy and UrbanCode Deploy with Patterns. More information and demonstration on these tools and your cloud to deploy applications are provided in Chapter 7, “Cloud user deployment and use scenarios” on page 563.

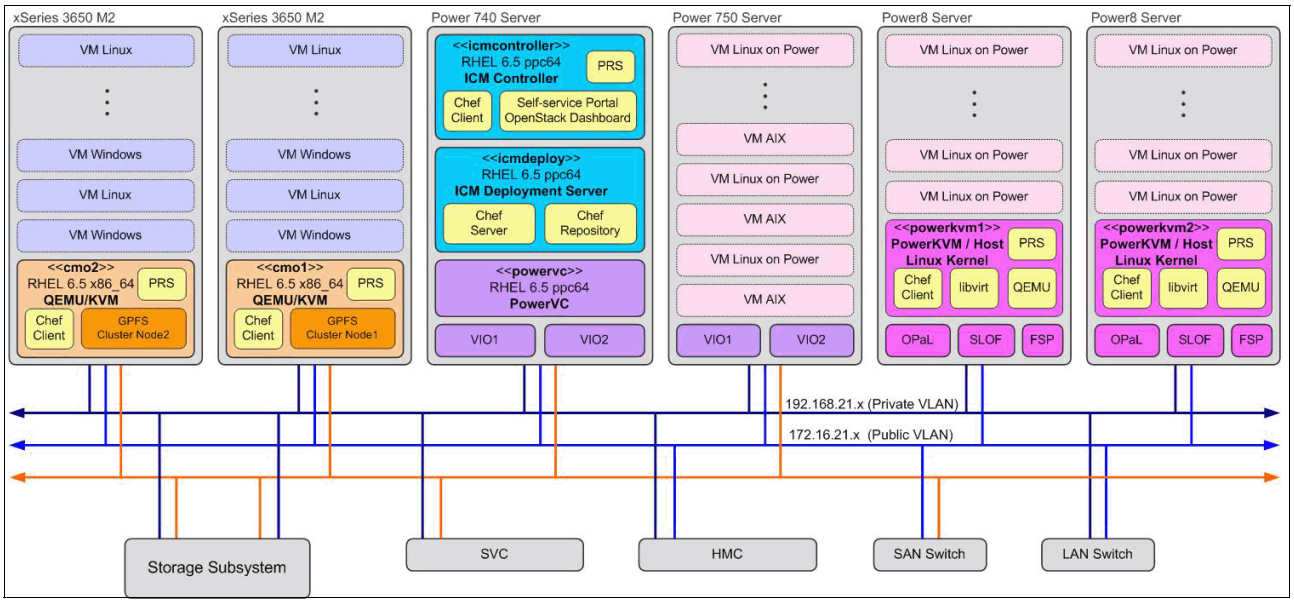


Figure 4-2 Topology of the residency cloud environment

4.3.1 Hardware and software setup for this publication

In the hybrid cloud created for this publication, three different virtualization technologies are used for server virtualization: KVM/QEMU, PowerKVM/QEMU and PowerVC/PowerVM. There are two Ethernet networks connected to all the physical servers, where the *public network* is for management only and the *private network* is for instances intercommunication. SAN Volume Controller (SVC) and General Parallel File System (GPFS) are used as elastic storage to provide backing devices to the instances.

KVM/QEMU

Kernel-based virtual machine (KVM) is an open source software that provides a virtualization structure for x86 or x86_64 servers with virtualization extensions.

In our cloud, we used 2 x3650 M2 servers running Red Hat Enterprise Linux Server release 6.5 and KVM or QEMU version qemu-kvm-0.12.1.2-2.415. These servers are connected to both public and private networks and to the SVC storage. A GPFS cluster was created in these 2 nodes using SVC disks. They are named cmo1.itso.ibm.com and cmo2.itso.ibm.com and are used as Compute nodes for deploying Microsoft Windows and Linux instances.

PowerKVM

PowerKVM is a server virtualization choice including Linux open source technology of KVM Virtualization, based on POWER8 technology. For more information about PowerKVM, its features, functionalities and how to implement it, see *IBM PowerKVM Configuration and Use*, SG24-8231.

In our cloud, we used 2 POWER8 servers running IBM_PowerKVM release 2.1.0 build 17 Gold. These servers are connected to both public and private networks. These servers are not connected to the SAN Switch or IBM SAN Volume Controller (SVC), but they can access the SVC disks using iSCSI. They are named `powerkvm1.itso.ibm.com` and `powerkvm2.itso.ibm.com` and are used as Compute nodes for deploying Linux running on Power Systems.

PowerVM

PowerVM consists of a combination of hardware, firmware and software running on POWER5, POWER6, POWER7 or POWER8 servers, providing CPU, network, Fibre Channel and disk virtualization features. For more information about PowerVM, its features and functionalities, see *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940.

Important: Although PowerVM can run on IBM POWER5 Servers, IBM Cloud Manager and IBM Power Virtualization Center (PowerVC) are not supported on POWER5 Systems.

In our cloud, we used one POWER7 server p740 (8205-E6C) running two Virtual I/O Servers 2.2.3.3 and one POWER7 server p750 (8233-E8B) running two Virtual I/O Servers 2.2.3.2. They both have Shared Ethernet Adapter failover (SEA failover) configured and providing access to both public and private networks. The Virtual I/O Servers (VIOS) have HBAs connected to the SAN Switches and to the SVC. The HBAs are NPIV Capable, so the logical partitions in these PowerVM Servers can use Virtual Fibre Channel Devices to access the SAN and SVC. These servers are managed by the same Hardware Management Console (HMC). These servers are used as Compute nodes to deploy Linux (ppc64) and AIX instances. The Controller node and deployment server are also two logical partitions (LPARs) running Red Hat Enterprise Linux Server release 6.5 (ppc64) operating in the P740 server).

Note: PowerVM environments cannot be directly managed by the Cloud Manager. The Cloud Manager can manage the PowerVM Servers using Power Virtualization Center (PowerVC).

Power Virtualization Center (PowerVC)

IBM POWER Virtualization Center (PowerVC) is an offering for Power Systems also based on OpenStack technology. It provides a tool to manage POWER servers, allowing you to capture images, deploy virtual machines and manage them. For more information about PowerVC, see *IBM PowerVC Version 1.2.1 Introduction and Configuration*, SG24-8199.

The Cloud Manager uses its PowerVC driver to communicate with PowerVC for deploying and managing instances, networks and volumes on the PowerVM Servers.

In our cloud, we created 1 LPAR running Red Hat Enterprise Linux Server release 6.5 (ppc64) and installed the PowerVC 1.2.1.1 on this LPAR. We used it to manage the HMC and both POWER7 servers (p750 and p740), the SVC Storage, SAN switch and both public and private networks being using in this environment. The PowerVC is then integrated with Cloud Manager, allowing it to deploy AIX and Linux (ppc64) instances in our cloud over both POWER7 servers.

SAN Volume Controller

IBM SAN Volume Controller (SVC) is a storage virtualization appliance that allows you to virtualize IBM and non-IBM Storage subsystems. For more information about SVC, see *IBM SAN Volume Controller 2145-DH8 Introduction and Implementation*, SG24-8229.

In our cloud, the SVC is under PowerVC management to provide volumes for instances deployed in the PowerVM environment. The operating systems (AIX and Linux) are installed on SVC disks, and they are captured as images in the PowerVC and Cloud Manager environment. Both PowerVC and Cloud Manager use FlashCopy technology in the SVC to capture and deploy instances. The Controller node also integrates with the SVC by configuration of Cinder (an OpenStack component that manages volumes) to provide volumes to the instances to use SVC as software-defined storage (SDS). SVC disks are also presented to the KVM Compute nodes where a GPFS cluster is created, `cmo1.itso.ibm.com` and `cmo2.itso.ibm.com`. One DS4800 storage subsystem is used to provide managed disks in our SVC.

IBM Spectrum Scale file system

Note: IBM Spectrum Scale was originally called IBM General Parallel File System (GPFS).

Spectrum Scale is a clustered file system that allows concurrent access to the files across several nodes. It can be used as Spectrum Scale or software-defined storage in the cloud environment. Volumes for x86_64 KVM Compute node instances can be file-backed devices, and the files can be placed in the GPFS file system. With the files available in the GPFS file system and the file system mounted on all x86_64 KVM nodes, the instance can be migrated dynamically from one node to the other by using solutions such as IBM Platform Resource Scheduler (PRS).

In our cloud, one GPFS cluster was created on x86_64 KVM Compute nodes:

`cmo1.itso.ibm.com and cmo2.itso.ibm.com using SVC disks.`

4.3.2 Cloud topology for this publication

The cloud topology used for this publication is *Controller + n Compute nodes*. The topology configuration can be summarized in the Table 4-2.

Table 4-2 Residency cloud topology summary

Node	Description
cmodeploy	Deployment server running RHEL 6.5 on POWER System in p740 server LPAR with Cloud Manager 4.1.0.1.
cmocontroller	Controller node running RHEL 6.5 on POWER System LPAR in the p740 server. The Controller node manages cmo1 and cmo2 as KVM Compute nodes, powerkvm1 and powerkvm2 as PowerKVM Compute nodes and it also has the PowerVC driver enabled to communicate with the PowerVC server. PRS 2.2 is also enabled.
powervc	PowerVC 1.2.1.1 server running RHEL 6.5 on POWER System LPAR in the p740 server and managing both p740 and p750.
p740_2_vio1	Virtual I/O Server 2.2.3.3 running in the p740 to provide network virtualization and NPIV to the LPARs from p740 server.
p740_2_vio2	Virtual I/O Server 2.2.3.3 running in the p740 to provide network virtualization and NPIV to the LPARs from p740 server.
p750_1_vio1	Virtual I/O Server 2.2.3.2 running in the p750 to provide network virtualization and NPIV to the LPARs from p750 server.
p750_1_vio2	Virtual I/O Server 2.2.3.2 running in the p750 to provide network virtualization and NPIV to the LPARs from p750 server.

Node	Description
cmo1	KVM Compute node running RHEL 6.5 on System x3650 M2 server.
cmo2	KVM Compute node running RHEL 6.5 on System x3650 M2 server.
powerkvm1	PowerKVM Compute node running PowerKVM 2.1.0 on POWER8 server.
powerkvm2	PowerKVM Compute node running PowerKVM 2.1.0 on POWER8 server.
SVC	SVC connected to the PowerVC and Controller nodes running SVC 6.4.0.0.
GPFS	GPFS cluster running on KVM Compute nodes on top of SVC disks. GPFS Version 4.1.0.0.
HMC	HMC V8R8.1.0.0 under control of PowerVC.
SAN Switch	SAN Switch IBM 2498-B40 running FOS v7.1.1 under control of PowerVC.
AIX LPARs	Several AIX 7100-03-00 LPARs as instances being deployed in the cloud.
UCD	Virtual machine in the cloud running x86_64 RHEL 6.5 and UrbanCode Deploy 6.1.0.1.532639 and UrbanCode Deploy with Patterns 6.1.0.1.545068 as Developer tool.

4.4 Deployment server installation

The deployment server is where the Cloud Manager installation package will be initially installed (and then used to deploy your cloud). IBM Cloud Manager version 4.1 can be installed on Red Hat Enterprise Linux version 6.4 or 6.5, either on ppc64 or x86_64 platforms.

To install the Deployment Server, complete the following steps:

1. Create the logical partition (required only if you plan to run Cloud Manager Deployment node in POWER System LPAR) or virtual machine (in KVM or VMware environment).
2. Install Red Hat Enterprise Linux 6.5 in the LPAR.
3. Customize Red Hat Enterprise Linux to meet the Cloud Manager requirements.
4. Install the Cloud Manager software.

4.4.1 IBM Cloud Manager hardware prerequisites

Cloud Manager can be installed on the following platforms:

- Intel x86_64 (running Linux).
- POWER6, POWER7, or POWER8 (running PowerLinux)

To complete this installation, these minimum requirements must be met:

- 4 CPUs
- At least 4GB of free space for `/opt/ibm/cmwo` (Cloud Manager directory. This does not need to be a separate file system, but the parent file system `/opt` or `/opt/ibm` must provide this free space.)
- At least 4.5 GB of temporary free space during the product installation
- Additional at least 5GB of free space in `/opt` and 5 GB of free space in `/var`
- At least 4 GB of RAM

For more information related to prerequisites for installing, see IBM Cloud Manager prerequisites:

http://ibm.co/1DCYLqs

4.4.2 Create the virtual machine to host the Cloud Manager deployment server

Create the virtual machine (VM) to host the Cloud Manager deployment server using same procedure that you use for any other workload. Remember to check the requirements from section 4.4.1, “IBM Cloud Manager hardware prerequisites” on page 202.

Create a logical partition on Power Systems using HMC

To create the logical partition, in the Navigation Panel, select the Managed System where you want to place the logical partition and click **Configuration > Create Logical Partition**.

Follow the instructions in the Create Partition Wizard to create the logical partition and its Profile.

Note: After creating the logical partition, you need to install the Red Hat Enterprise Linux 6.4 or 6.5 (ppc64) operating system to host the deployment server.

Create a virtual machine on System X server

If the deployment server will run on a System x server, it can run directly in the physical host or can be virtualized using KVM or VMware. Follow the instructions according to the environment that you will use.

Note: After creating the virtual machine, you must install the Red Hat Enterprise Linux 6.4 or 6.5 (x86_64) operating system to host the deployment server.

4.4.3 Install Red Hat Enterprise Linux

Install the Red Hat Enterprise Linux 6.4 or 6.5 in the LPAR that will host the deployment server. For detailed instructions, see the Red Hat Enterprise Linux 6 Installation Guide on the RedHat website:

http://red.ht/1xEzlge

Note: You do not need to install additional packages on RHEL after the base operating system installation completes. The Cloud Manager deployment server installer will install the required packages if a Yum repository with the base installation media is available.

4.4.4 Customize Red Hat Enterprise Linux to host the deployment server

Some requirements must be attended to host the Cloud Manager deployment server. The next topics show how to complete them.

Install IBM Installation Toolkit for PowerLinux

If you are running the deployment server in a Power Systems LPAR, you should install the IBM Installation Toolkit for PowerLinux to install additional packages for RMC communication with the HMC. This will enable important features such as dynamic logical partitions (DLPARs).

The IBM Installation Toolkit for Linux on Power is available on the following web page:

https://www-304.ibm.com/webapp/set2/sas/f/lopdiags/installtools/home.html

For detailed instructions on how to install the IBM Installation Toolkit for PowerLinux, see the *IBM Installation Toolkit User's Guide*:

https://www.ibm.com/webapp/set2/sas/f/lopdiags/installtools/install/home.html

Use a fully qualified domain name

It is required to use fully qualified domain name (FQDN) on your nodes, including the deployment server. For example, an FQDN is *cmodeploy.itso.ibm.com*, not *cmodeploy*.

Make sure the output of the **hostname** command shows the FQDN. Example 4-1 shows the output of the hostname command with a fully qualified domain name.

Example 4-1 hostname output from deployment server with FQDN

```
[root@cmodeploy ~]# hostname
cmodeploy.itso.ibm.com
```

The deployment server should also be configured to keep using fully qualified domain name after the reboot. The `HOSTNAME` entry in the `/etc/sysconfig/network` configuration file can be modified to configure the FQDN to persist after reboot. Example 4-2 shows a network configuration file which will allow the server to continue using FQDN after reboot.

Example 4-2 Sample /etc/sysconfig/network file with FQDN

```
[root@cmodeploy ~]# cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=cmodeploy.itso.ibm.com
GATEWAY='172.16.20.1'
```

Configure host name resolution

Host name resolution must be configured across all the hosts that will participate in the cloud management (only physical hosts, not virtual machines), including deployment server, Controller node, and Compute nodes (hypervisors). This name resolution can be accomplished by using the `/etc/hosts` file, but DNS is recommended, especially if you have an enterprise cloud. If you plan to use `/etc/hosts`, make sure that you configure it with FQDN on all of the components, as shown in Example 4-3.

Example 4-3 Sample /etc/hosts with FQDN

```
127.0.0.1     localhost.localdomain        localhost
172.16.21.136 cmodeploy.itso.ibm.com       cmodeploy
172.16.21.137 cmocontroller.itso.ibm.com   cmocontroller
172.16.21.131 cmo1.itso.ibm.com            cmo1
172.16.21.132 cmo2.itso.ibm.com            cmo2
172.16.21.143 powerkvm1.itso.ibm.com       powerkvm1
172.16.21.144 powerkvm2.itso.ibm.com       powerkvm2
172.16.21.135 powervc.itso.ibm.com         powervc
```

To make sure your deployment server hostname is resolvable, run this command:

```
hostname -f
```

If the hostname is resolvable, it will return an output similar to Example 4-4 on page 205.

Example 4-4 Example of hostname -f command

```
[root@cmodeploy ~]# hostname -f
cmodeploy.itso.ibm.com
```

Synchronize the date and time

Date and time must be synchronized as close as possible on all physical hosts in the cloud (a few seconds of difference is acceptable). You can use an NTP server to accomplish this requirement, or simply configure your servers to match the date and time using regular Linux configuration commands.

Run the following command to configure date and time in your system:

```
[root@cmodeploy ~]# system-config-date
```

If you plan to use your system as NTP Client, enable the option `Synchronize date and time over the network`, and then click `Add` and provide your NTP server address as shown in Figure 4-3.

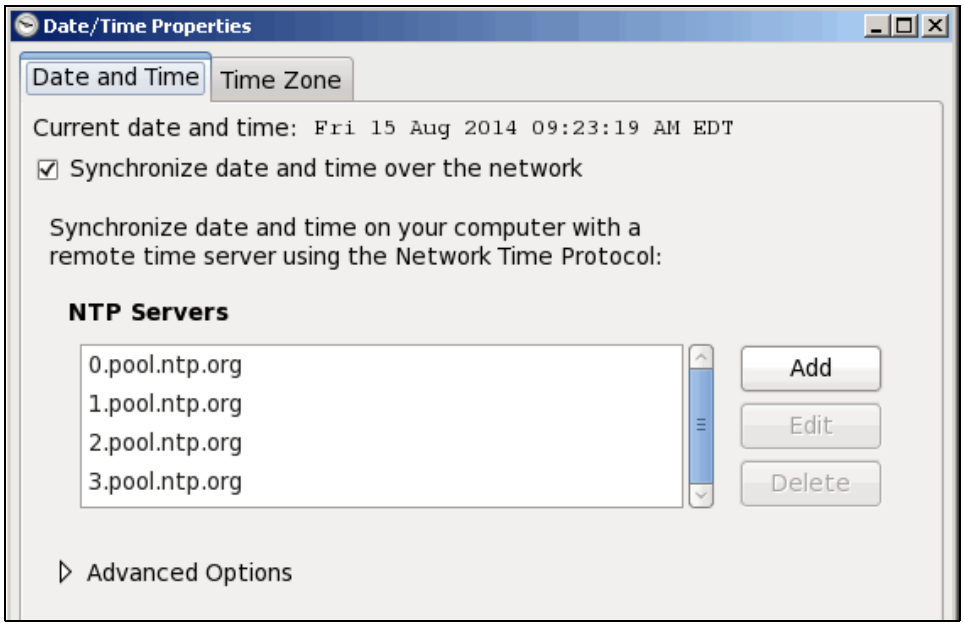


Figure 4-3 NTP client configuration

If you do not plan to use NTP, select the correct time zone in the `Time Zone` tab and provide the correct date and time in the `Date and Time` tab as shown in Figure 4-4 on page 206.

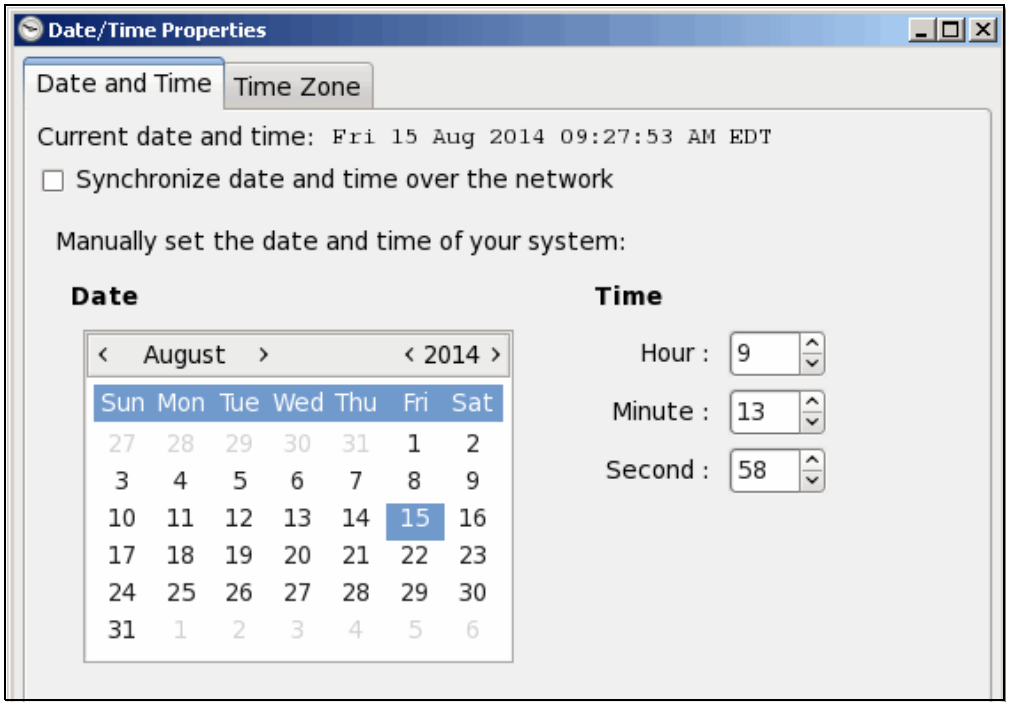


Figure 4-4 Time and date configuration without NTP

Note: Make sure to provide the time and date as close as possible to all nodes in the cloud. Only a few seconds of difference is acceptable.

Define the same root password on the nodes

Although it is not a requirement, it can be a convenience, in case security policies allow it, to use the same root password on all physical nodes. This will simplify the administration and avoid mistakes when providing the root password in the deployment server configuration files when you are deploying your cloud.

Provide network connection to the management network

The deployment server needs to access the management network, which is used exclusively for management purposes. No network bridge is created on this network, so it won't be available to the instances in your cloud. The Controller and the Compute nodes should also have access to the management network. Make sure an IP is configured on the management network and that all of the nodes can ping each other.

Configure the Yum repository

The Cloud Manager installation package, will automatically install the dependency packages by using Yum. A Yum repository with the Red Hat Enterprise Linux installation media must be corectly configured and available during installation time:

1. To configure the Yum repository, copy the ISO file of the installation media that you used to install the deployment server to a file system in the host. Then, add the following line to the `/etc/fstab` file to mount this ISO image during boot time:

```
/root/RHEL6.5-20131111.0-Server-ppc64-DVD1.iso  /media/cdrom    iso9660
loop,ro,auto 0 0
```

Note: The first entry is the full path to the ISO file which was copied to the deployment server. This entry must match the name and location you used in your system. The second entry is where the ISO will be mounted, so this directory must exist in your system.

2. Run the **mount -a** command to mount the ISO image:

```
[root@cmodeploy ~]# mount -a
```

3. Copy the `media.repo` file from your ISO image to the `/etc/yum.repos.d/` directory, and edit this file as shown in Example 4-5.

Example 4-5 How to set Yum repository

```
[InstallMedia]
name=Red Hat Enterprise Linux 6.5
mediaid=1384198011.520581
metadata_expire=-1
enabled=1
gpgcheck=0
cost=500
baseurl=file:///media/cdrom/Server
```

4. Verify that the repository you created is available to Yum as shown in Example 4-6.

Example 4-6 Yum repolist output from the deployment server

```
[root@cmodeploy ~]# yum repolist
repo id          repo name                       status
InstallMedia     Red Hat Enterprise Linux 6.5   3,396
IBMIT_LOCAL_REPO IBMIT LOCAL REPO               58
ibmit-local      ibmit-local                    58
repolist: 3,512
```

Note: Make sure that all the repositories which are currently enabled are working. If possible, disable other repositories temporarily, keeping only the installation media enabled. In case other repositories are enabled but not working, the Cloud Manager installation might fail because it tries to use the **yum** command.

4.4.5 Install IBM Cloud Manager 4.1 on the deployment server

This section explains how to install the Cloud Manager on your deployment server. Before you start the installation, make sure you meet the prerequisites described in topic 4.4.1, “IBM Cloud Manager hardware prerequisites” on page 202. Also make sure you have followed the instructions in topics 4.4.2, “Create the virtual machine to host the Cloud Manager deployment server” on page 203, 4.4.3, “Install Red Hat Enterprise Linux” on page 203 and 4.4.4, “Customize Red Hat Enterprise Linux to host the deployment server” on page 203.

Running the Cloud Manager installation package, sets up the system as deployment server. During this step, the following happens:

- The Chef server is installed and configured.
- Yum repository files is created in the system.
- Cookbooks, roles, data bags and sample environment files are uploaded to the Chef repository.

- The knife command is installed, providing a command-line interface to interact with Chef to deploy your cloud.

The installation steps will use the tar.gz file obtained from the download site. Contact your sales representative to obtain the Cloud Manager license and installation files. The trial version of Cloud Manager can be obtained at the following site:

http://www.ibm.com/developerworks/servicemanagement/cvm/sce/downloads.html

There are three possible types of installation for the Cloud Manager:

- Console installation
- Graphical installation
- Silent installation

This publication will use the graphical installation. For more information about the other installation types, see the *IBM Cloud Manager with OpenStack 4.1 Administrator Guide* available at:

http://www-01.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html

A simple example of the silent installation can also be found at the *IBM Cloud Manager with OpenStack 4.1 Quick Start Guide* available at:

http://www.ibm.com/developerworks/servicemanagement/cvm/sce/i/CMOQuickStartGuide.pdf

Additional Linux packages need to be installed before installing the Cloud Manager, in case you plan to use Graphical installation:

- For PowerLinux, use `libX11.ppc` and `libX11-devel`.
- For x86_64 Linux, use `libX11.i686` and `libX11-devel`.

To do this installation, run the following **`yum`** command, which will obtain the installation packages from the Yum repository configured in “Configure the Yum repository” on page 206:

```
yum install <package_name1> <package_name2>
```

Example 4-7 shows the output from a PowerLinux system.

Example 4-7 Installing libX11.ppc and libX11-devel packages

```
# yum install libX11.ppc libX11-devel
Loaded plug-ins: product-id, refresh-packagekit, security, subscription-manager
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package libX11.ppc 0:1.5.0-4.el6 will be installed
--> Processing Dependency: libxcb.so.1 for package: libX11-1.5.0-4.el6.ppc
---> Package libX11-devel.ppc64 0:1.5.0-4.el6 will be installed
--> Processing Dependency: pkgconfig(xcb) >= 1.1.92 for package:
libX11-devel-1.5.0-4.el6.ppc64
--> Processing Dependency: pkgconfig(xproto) for package:
libX11-devel-1.5.0-4.el6.ppc64
--> Processing Dependency: pkgconfig(xcb) for package:
libX11-devel-1.5.0-4.el6.ppc64
--> Processing Dependency: pkgconfig(kbproto) for package:
libX11-devel-1.5.0-4.el6.ppc64
--> Running transaction check
---> Package libxcb.ppc 0:1.8.1-1.el6 will be installed
--> Processing Dependency: libXau.so.6 for package: libxcb-1.8.1-1.el6.ppc
```

```
---> Package libxcb-devel.ppc64 0:1.8.1-1.el6 will be installed
--> Processing Dependency: pkgconfig(xau) >= 0.99.2 for package:
libxcb-devel-1.8.1-1.el6.ppc64
---> Package xorg-x11-proto-devel.noarch 0:7.6-25.el6 will be installed
--> Running transaction check
---> Package libXau.ppc 0:1.0.6-4.el6 will be installed
---> Package libXau-devel.ppc64 0:1.0.6-4.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch      Version        Repository       Size
================================================================================
Installing:
 libX11                 ppc       1.5.0-4.el6    InstallMedia    585 k
 libX11-devel           ppc64     1.5.0-4.el6    InstallMedia    1.0 M
Installing for dependencies:
 libXau                 ppc       1.0.6-4.el6    InstallMedia     25 k
 libXau-devel           ppc64     1.0.6-4.el6    InstallMedia     14 k
 libxcb                 ppc       1.8.1-1.el6    InstallMedia    115 k
 libxcb-devel           ppc64     1.8.1-1.el6    InstallMedia    174 k
 xorg-x11-proto-devel   noarch    7.6-25.el6     InstallMedia    274 k

Transaction Summary
================================================================================
Install       7 Package(s)

Total download size: 2.2 M
Installed size: 7.2 M
Is this ok [y/N]: y
Downloading Packages:
--------------------------------------------------------------------------------
Total                                           10 MB/s | 2.2 MB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Warning: RPMDB altered outside of yum.
  Installing : xorg-x11-proto-devel-7.6-25.el6.noarch                      1/7
  Installing : libXau-1.0.6-4.el6.ppc                                      2/7
  Installing : libxcb-1.8.1-1.el6.ppc                                      3/7
  Installing : libXau-devel-1.0.6-4.el6.ppc64                              4/7
  Installing : libxcb-devel-1.8.1-1.el6.ppc64                              5/7
  Installing : libX11-1.5.0-4.el6.ppc                                      6/7
  Installing : libX11-devel-1.5.0-4.el6.ppc64                              7/7
  Verifying  : libXau-devel-1.0.6-4.el6.ppc64                              1/7
  Verifying  : libXau-1.0.6-4.el6.ppc                                      2/7
  Verifying  : libX11-1.5.0-4.el6.ppc                                      3/7
  Verifying  : libxcb-devel-1.8.1-1.el6.ppc64                              4/7
  Verifying  : libxcb-1.8.1-1.el6.ppc                                      5/7
  Verifying  : xorg-x11-proto-devel-7.6-25.el6.noarch                      6/7
  Verifying  : libX11-devel-1.5.0-4.el6.ppc64                              7/7
Installed:
  libX11.ppc 0:1.5.0-4.el6                    libX11-devel.ppc64 0:1.5.0-4.el6
```

```
Dependency Installed:
  libXau.ppc 0:1.0.6-4.el6     libXau-devel.ppc64 0:1.0.6-4.el6     libxcb.ppc
0:1.8.1-1.el6     libxcb-devel.ppc64 0:1.8.1-1.el6     xorg-x11-proto-devel.noarch
0:7.6-25.el6

Complete!
```

Note: If these packages are not corectly installed, the installation using Graphical Interface will fail with the following error message:

```
PuTTY X11 proxy: wrong authorisation protocol attempted
Graphical installers are not supported by the VM. The console mode will be used
instead...
```

Even with this error, the installation will proceed by using console installation rather than graphical installation.

After completing this step, extract the Cloud Manager installation package, which will have a name similar to:

- For PowerLinux: `cloud_mgr_openstack_4.1_aix_rhel.tar.gz`
- For x86_64 Linux: `cloud_mgr_openstack_4.1_wdws_rhel.tar.gz`

Use this command to extract the package:

```
# tar -zxvf <package_name>.tar.gz
```

After extracting, the contents will be available in this location:

- For PowerLinux: `/<directory_where_you_extracted>/install/power/linux`
- For x86_64 Linux: `/<directory_where_you_extracted>/install/x86-64/linux`

Example 4-8 shows the expected content for this directory for a PowerLinux installation.

Example 4-8 PowerLinux Cloud Manager installation package

```
-rwxrwxr-x. 1 1000 1000  696659183 May 28 13:39 cmwo410_plinux_install.bin
-rw-rw-r--. 1 1000 1000 1009412262 May 28 13:57 cmwo410_plinux_install_pkg_01.tar.gz
-rw-rw-r--. 1 1000 1000  818247595 May 28 13:51 cmwo410_plinux_install_pkg_02.tar.gz
-rw-rw-r--. 1 1000 1000 1070820294 May 28 13:46 cmwo410_plinux_install_pkg_03.tar.gz
-rw-rw-r--. 1 1000 1000       2852 May 28 13:46 cmwo-install-sample.rsp
```

Example 4-9 shows the expected content for this directory for an x86_64 Linux installation.

Example 4-9 x86_64 Cloud Manager installation package

```
-rwxrwxr-x. 1 1000 1000  692858128 May 28 13:23 cmwo410_xlinux_install.bin
-rw-rw-r--. 1 1000 1000 1009412262 May 28 13:35 cmwo410_xlinux_install_pkg_01.tar.gz
-rw-rw-r--. 1 1000 1000  818247595 May 28 13:31 cmwo410_xlinux_install_pkg_02.tar.gz
-rw-rw-r--. 1 1000 1000 1070820294 May 28 13:27 cmwo410_xlinux_install_pkg_03.tar.gz
-rw-rw-r--. 1 1000 1000       2852 May 28 13:19 cmwo-install-sample.rsp
```

Change to the directory where the files are extracted, and run the following command to start the graphical installation of Cloud Manager:

For PowerLinux `./cmwo410_plinux_install.bin -i gui`
For x86_64 Linux `./cmwo410_xlinux_install.bin -i gui`

Example 4-10 shows the expected messages to appear during the installation process in a PowerLinux system.

Example 4-10 Starting the Cloud Manager installation on a PowerLinux system

```
[root@cmodeploy linux]# ./cmwo410_plinux_install.bin -i gui
Preparing to install...
Extracting the JRE from the installer archive...
Unpacking the JRE...
Extracting the installation resources from the installer archive...
Configuring the installer for this system's environment...

Launching installer...
```

Note: If you are installing the Cloud Manager remotely, to use the graphical interface, you must export X11 and allow X11 forwarding for SSH connections. The workstation from where you are installing the Cloud Manager must also have a working X server.

In the installation screen shown in Figure 4-5, click **OK** to select English during the Cloud Manager installation process.

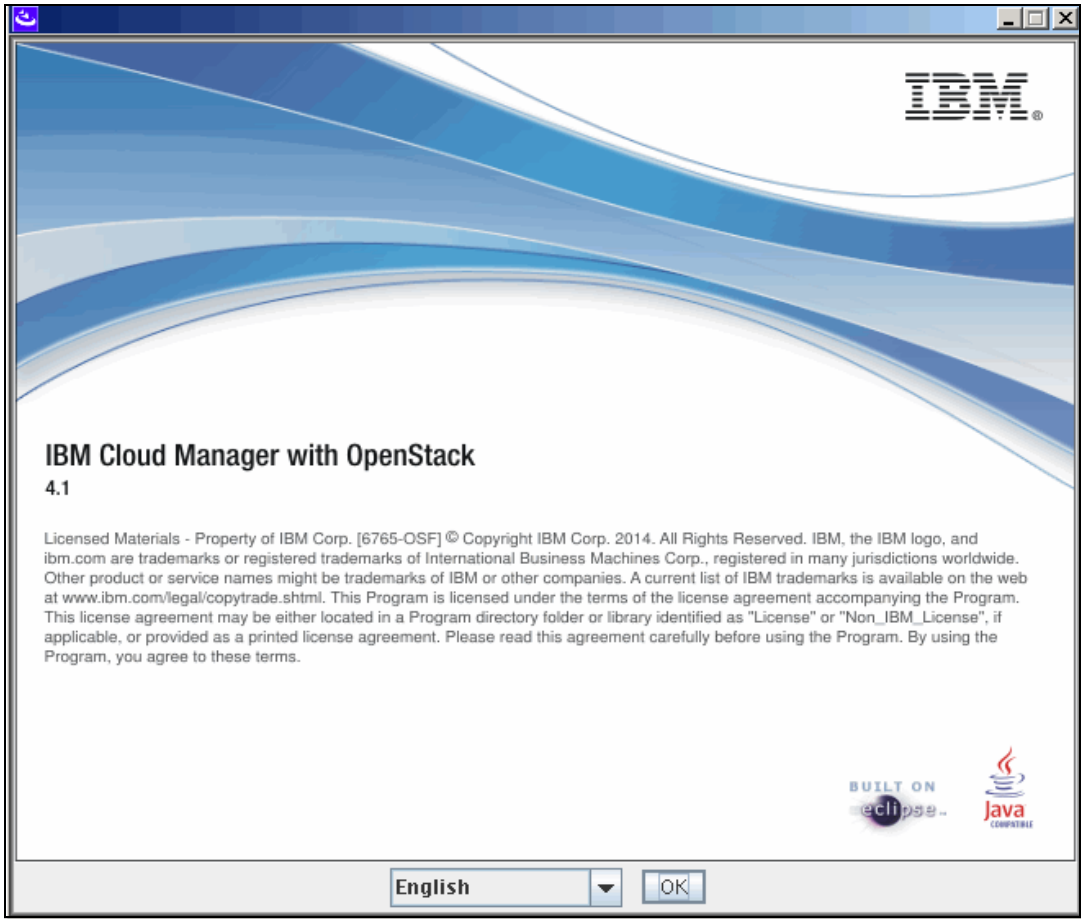


Figure 4-5 Language selection during installation process

Read the introduction, and then click **Next**, as shown in Figure 4-6.

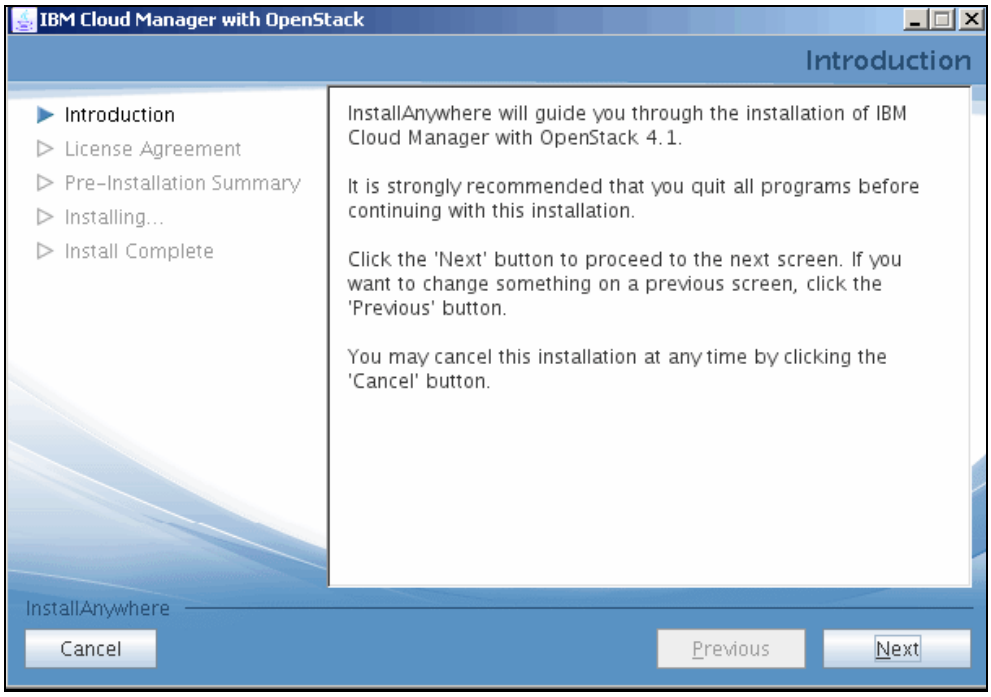


Figure 4-6 Introduction to Cloud Manager installation

Read and accept the license terms and click **Next**, as shown in Figure 4-7.

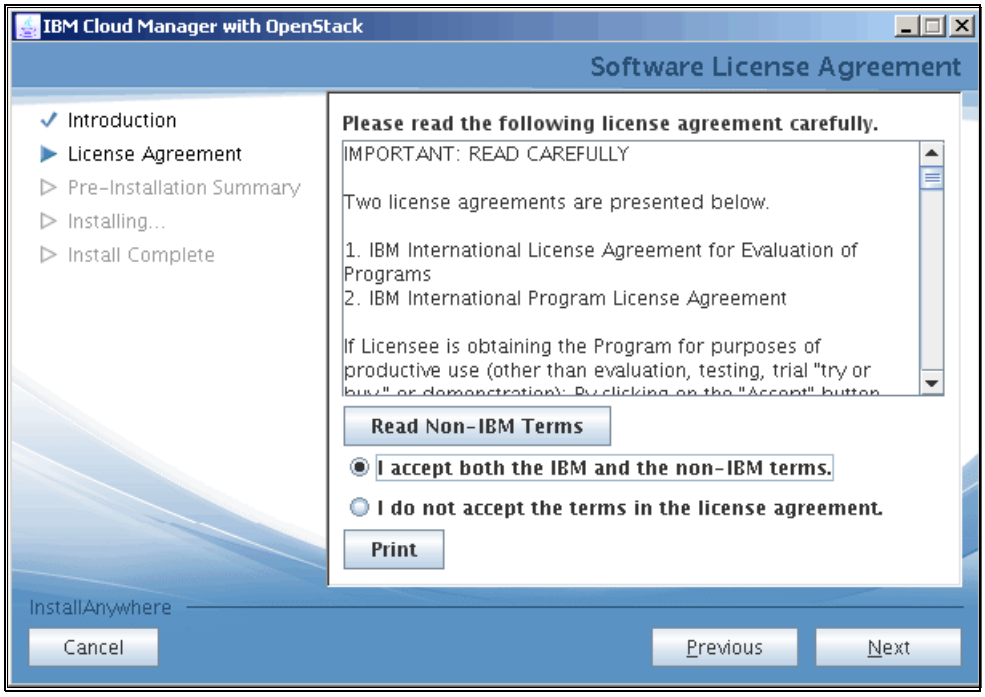


Figure 4-7 License terms during Cloud Manager installation

Click **Install** to proceed with the Cloud Manager installation, as shown in Figure 4-8.

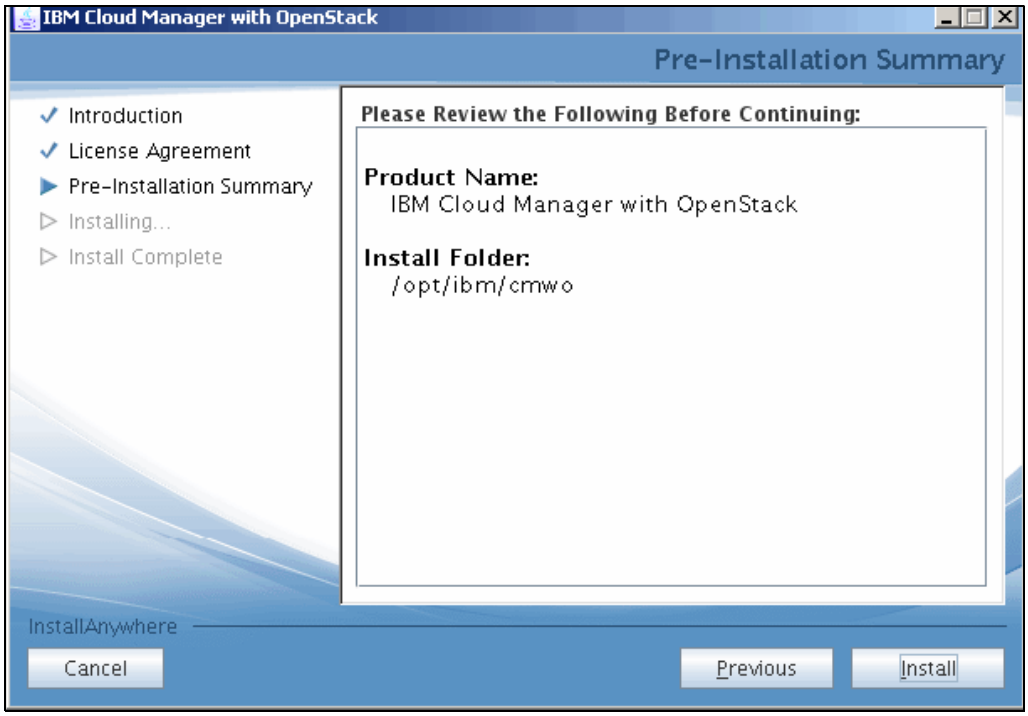


Figure 4-8 Initiating the Cloud Manager installation

Wait until the installation is finished, and then click **Done**, as shown in Figure 4-9.



Figure 4-9 Completing the Cloud Manager installation

The installation logs are in the following directory:

`/opt/ibm/cmwo/_installation/Logs/`

Low-level detail of the installation is in the following log file:

`/tmp/cmwo-installer.log`

4.4.6 Update the Cloud Manager to the latest level

The Cloud Manager Fix Packs are available at the Fix Central:

http://www.ibm.com/eserver/support/fixes/

1. Under Product Group, select **Other Software**.
2. Under Select from Other Software, select **Cloud Manager**.
3. Under Installed Version, select **4.1.0**.
4. Under Platform, select the appropriate platform.

Note: You might notice a **Windows 64-bit, x86** option under **Platform**. The IBM Cloud Manager is available only for Linux. This option will allow you to download updates to the self-service portal, which can be installed in the Windows systems.

Download the latest available version from Fix Central. The download will have a name similar to `cmwo_fixpack_4.1.0.2.tar.gz` (in this case, Fix Pack 2). Save the file in your deployment server, and extract it by using the following command:

```
# tar -zxvf cmwo_fixpack_4.1.0.2.tar.gz
```

Note: At the time of writing this IBM Redbooks publication, the latest available Fix Pack was Fix Pack 3 (IBM Cloud Manager 4.1.0.3).

After extraction, the available files will be similar to Example 4-11.

Example 4-11 Files in Cloud Manager fix pack

```
-rw-r-----. 1 root root 1275517134 Aug  6 10:51 cmwo_fixpack_4.1.0.2.tar.gz
-r--r--r--. 1 root root        672 Jun 29 08:07 fixpack.properties
-r-xr-xr-x. 1 root root      11656 Jun 29 07:58 install_cmwo_fixpack.sh
drwxrwxrwx. 8 root root       4096 Jun 29 08:07 product_files
-r--r--r--. 1 root root       8305 Jun 29 08:07 README.txt
drwxrwxrwx. 2 root root       4096 Jun 29 07:58 resources
```

Execute the following script to update the Cloud Manager:

```
# ./install_cmwo_fixpack.sh
```

You will see messages similar to Example 4-12.

Example 4-12 Cloud Manager update process

```
07/25/2014 04:55:54 PM Installed version is 4.1.0.0-201405281003.
07/25/2014 04:55:54 PM Copying product_files...
07/25/2014 04:56:04 PM Copy successful.
07/25/2014 04:56:04 PM Running post-install fix pack scripts...
07/25/2014 04:57:25 PM Post-install scripts completed successfully.
07/25/2014 04:57:25 PM Installation of fix pack completed successfully.
```

After the installation is complete, the log files will be available in the following file:

```
/opt/ibm/cmwo/version/install_cmow_fixpack_<date_and_time>_logs.zip
```

Validate the installation by running the following commands:

```
# chef-server-ctl status
# knife cookbook list
# knife role list
# knife environment list
```

The expected output is similar to Example 4-13.

Example 4-13 Validating the Cloud Manager installation in the Deployment Server

```
[root@cmodeploy fixpack]# chef-server-ctl status
run: bookshelf: (pid 8566) 556s; run: log: (pid 8565) 556s
run: chef-expander: (pid 8491) 567s; run: log: (pid 8490) 567s
run: chef-server-webui: (pid 8848) 534s; run: log: (pid 8847) 534s
run: chef-solr: (pid 8403) 573s; run: log: (pid 8402) 573s
run: erchef: (pid 9365) 501s; run: log: (pid 8655) 550s
run: nginx: (pid 9319) 503s; run: log: (pid 9318) 503s
run: postgresql: (pid 8272) 580s; run: log: (pid 8271) 580s
run: rabbitmq: (pid 7782) 601s; run: log: (pid 7781) 601s

[root@cmodeploy fixpack]# knife cookbook list
apache2                              1.9.6
apt                                  2.3.8
aws                                  1.0.0
build-essential                      1.4.2
chef_handler                         1.1.5
database                             2.0.0
db2                                  0.2.2
erlang                               1.4.2
homebrew                             1.5.4
ibm-openstack-appliance-migration    0.1.23
ibm-openstack-common                 9.5.4
ibm-openstack-iaas-gateway           0.1.4
ibm-openstack-iptables               9.2.1
ibm-openstack-powervc-driver         9.2.0
ibm-openstack-prs                    0.1.5
ibm-openstack-roles                  9.0.1
ibm-openstack-simple-token           9.0.0
ibm-openstack-yum-server             9.0.1
ibm-openstack-zvm-driver             0.1.0
ibm-sce                              0.1.9
iptables                             0.13.2
logrotate                            1.5.0
mysql                                4.1.2
ntp                                  1.5.4
openssl                              1.1.0
openstack-block-storage              9.2.2
openstack-common                     9.2.2
openstack-compute                    9.2.2
openstack-dashboard                  9.0.3
openstack-identity                   9.2.0
openstack-image                      9.0.3
```

```
openstack-network                   9.0.8
openstack-ops-database              9.0.1
openstack-ops-messaging             9.0.1
openstack-orchestration             9.1.4
openstack-telemetry                 9.1.0
pacman                              1.0.4
postgresql                          3.3.4
python                              1.4.6
qpid                                0.3.0
rabbitmq                            3.0.4
selinux                             0.8.0
windows                             1.30.0
xfs                                 1.1.0
yum                                 3.1.4
yum-epel                            0.3.4
yum-erlang_solutions                0.1.4

[root@cmodeploy fixpack]# knife role list
allinone-compute
ibm-os-allinone-kvm
ibm-os-base
ibm-os-cache-server-node
ibm-os-client-node
ibm-os-compute-node-kvm
ibm-os-compute-node-powerkvm
ibm-os-dashboard-branding
ibm-os-dashboard-node
ibm-os-database-client-node
ibm-os-database-server-node
ibm-os-messaging-server-node
ibm-os-powervc-driver-node
ibm-os-prs-base
ibm-os-prs-compute-node
ibm-os-prs-controller-node
ibm-os-prs-ego-candidate
ibm-os-prs-ego-master
ibm-os-prs-ha-service
ibm-os-prs-horizon-gui
ibm-os-prs-notification-service
ibm-os-prs-scheduler-service
ibm-os-single-controller-distributed-database-node
ibm-os-single-controller-node
ibm-os-single-controller-powervc-driver
ibm-os-zvm-driver-node
ibm-sce-node
os-base
os-block-storage
os-block-storage-api
os-block-storage-scheduler
os-block-storage-volume
os-compute-api
os-compute-api-ec2
os-compute-api-metadata
os-compute-api-os-compute
os-compute-cert
```

```
os-compute-conductor
os-compute-scheduler
os-compute-setup
os-compute-single-controller
os-compute-single-controller-no-network
os-compute-vncproxy
os-compute-worker
os-dashboard
os-identity
os-image
os-image-api
os-image-registry
os-image-upload
os-network
os-network-dhcp-agent
os-network-l3-agent
os-network-metadata-agent
os-network-openvswitch
os-network-server
os-ops-database
os-ops-messaging
os-orchestration
os-orchestration-api
os-orchestration-api-cfn
os-orchestration-api-cloudwatch
os-orchestration-engine
os-telemetry
os-telemetry-agent-central
os-telemetry-agent-compute
os-telemetry-agent-notification
os-telemetry-alarm-evaluator
os-telemetry-alarm-notifier
os-telemetry-api
os-telemetry-collector

[root@cmodeploy fixpack]# knife environment list
_default
example-ibm-os-allinone
example-ibm-os-single-controller-n-compute
example-ibm-sce
```

4.5 Deploy the Controller node

The deployment of the Controller node involves using the deployment server to automatically install and configure all packages that are required to setup a controller node. In this process, you will use the **knife** command to tell the Chef server to setup your Controller node.

The Controller node can run on Red Hat Enterprise Linux version 6.4 or 6.5, either on ppc64 (Power) or x86_64 platforms. If you are running on ppc64 platform, your Controller node must be dedicated for the Controller role, which means that no instance (virtual machine) can run inside the Controller node. If you are running on a x86_64 platform, you can choose if your Controller node will be dedicated for the controller role, or if it will also be a KVM node, where instances will run inside the Controller node (in this case, the controller will be able to execute two roles: Controller and Compute node).

The following steps are required to deploy your Controller node:

1. Create the logical partition (required only if you plan to run the Controller node in a POWER System LPAR).
2. Install Red Hat Enterprise Linux 6.5.
3. Customize Red Hat Enterprise Linux to meet the Controller node requirements.
4. Configure the network interfaces in the Controller node.
5. Deploy the Controller node from the deployment server.

4.5.1 Hardware prerequisites for the Controller node

The Controller node can be deployed in the following hardware:

- Intel x86_64 (running Linux)
- POWER6, POWER7 or POWER8 (running PowerLinux)

To complete this deployment, the following minimum hardware requirements must be provided:

- 4 CPUs
- 16GB physical memory
- 6GB of free disk space for installed code and empty databases

Note: If you add more CPUs to the Controller node, you should also consider increasing the memory size proportionally, because with more CPUs, more openstack threads will be able to run at the same time, increasing the amount of memory requirements.

For more information about prerequisites for the Controller node, see the IBM Cloud Manager with OpenStack 4.1 Administrator Guide prerequisites:

http://www.ibm.com/support/knowledgecenter/api/content/SST55W_4.1.0/liaca/liaca_ui_support.html

4.5.2 Create the logical partition to host the Controller node

Follow the instructions from “Create a logical partition on Power Systems using HMC” on page 203 to create the LPAR that will perform the Controller role.

4.5.3 Install Red Hat Enterprise Linux

Follow the instructions from 4.4.3, “Install Red Hat Enterprise Linux” on page 203, to install the Red Hat Enterprise Linux 6.4 or 6.5 on the Controller node.

Important: Do not enable SELinux in the Controller node. If you need instructions on how to disable it, see “SELinux setting configuration” on page 287.

4.5.4 Customize Red Hat Enterprise Linux to host the Controller node

The same customization done in the Deployment Server is required in the Controller node. Follow the instructions from 4.4.4, “Customize Red Hat Enterprise Linux to host the deployment server” on page 203 to complete the requirements for the Controller node.

4.5.5 Network considerations and requirements

The following network types can be implemented in your cloud:

Local network Provides only local communication inside the hypervisor or Compute node. Only instances running inside the same Compute node will be able to communicate. This is usually used in a minimal topology deployment for tests purposes.

VLAN VLAN tagged networks where the same physical port in the switch can handle traffic from multiple VLANs. In this case, the packets must be tagged with the VLAN ID, so the switch can route them to the correct VLAN.

Flat Uses the network adapters in the Compute nodes, configured as bridges (in this case, using openvswitch) to allow communication between instances residing in different Compute nodes.

GRE Tunneling protocol called Generic Routing Encapsulation (GRE) that can encapsulate several different protocols inside IP packets, which can be routed between nodes in your cloud. This avoids VLAN tagging configuration requirements.

Note: Only VLAN and Flat networks are covered in this publication.

Some restrictions apply to your cloud depending on the types of hypervisors that will be used. Table 4-3 explains the network types you can define based on the hypervisors from your cloud.

Table 4-3 Supported network configuration

Hypervisor	Number of network interfaces	Network type
Hyper-V	Two[a]	VLAN[d] or Flat
KVM	One[b] or Two[a]	Local, GRE[c], Flat or VLAN[d]
PowerKVM	Two[a]	VLAN[d] or Flat
PowerVC	Two[a]	VLAN[d] only
z/VM	Two[a]	VLAN[d] or Flat

a. When you use two network adapters, the first network adapter (usually `eth0`) must be dedicated for the management network, which means that only a single IP must be configured on top of it, without creating any bridge. The instances won't use this network interface to access the network. The second network adapter will be used to provide a bridge to the instances. It is called the *virtual machine data network* and, in this publication, it is also referred as *private network* during our network customizations.
b. In KVM Compute nodes with a single network adapter, only local or GRE networks can be used.
c. GRE networks can only be used when all your Compute nodes are KVM running x86_64. If you have any Compute node that is not a KVM, a GRE network cannot be used.
d. A DHCP server can be configured in your controller node to automatically provide IPs to the deployed instances. If you have a VLAN network and you want to use a DHCP server, the network adapter used for the virtual machine data network must use a physical adapter. This means that if your Controller node is running on a Power Systems LPAR, you must dedicate a physical adapter to the Controller LPAR.

For more information about network considerations and restrictions, see "Network Considerations" in the IBM Cloud Manager with OpenStack 4.1 Administrator Guide:

`http://ibm.co/1PT7mhI`

Provide two network adapters to your Controller node. One of the adapters will be dedicated for the management network (in our topology called Public Network) and one will be dedicated for the virtual machines data network (in our topology, called Private Network). In case you plan to use VLAN networks, the network adapter dedicated for virtual machines data network must a physical adapter.

Note: If you have PowerVC in your environment to manage IBM Power Systems, the only possible network type is VLAN, so you must use a physical adapter for the virtual machines data network.

4.5.6 Configure network interfaces in the Controller node

The network configuration of the Controller node consists of two steps:

1. Configuring the management network interface.
2. Configuring the virtual machines data network interface.

Configure the management network interface

The management network is used to communicate with the physical nodes of your cloud, including the deployment server and Compute nodes. It must have a dedicated network adapter, without any bridge configured on top of it.

Important: The names of the network adapters can be changed in the Cloud Manager configuration files, but if you can provide `eth0` for the management network and `eth1` for the virtual machines data network, you can reduce the amount of configuration changes required to deploy your cloud. This publication will consider that `eth0` is used as the management network interface.

Run the following command to configure the management network:

```
# system-config-network
```

Press Enter for **Device Configuration** and Enter on your network interface that is dedicated for the management network (preferably `eth0`). Provide the IP address, network mask, default gateway, and DNS configuration for your network, as shown in Figure 4-10.

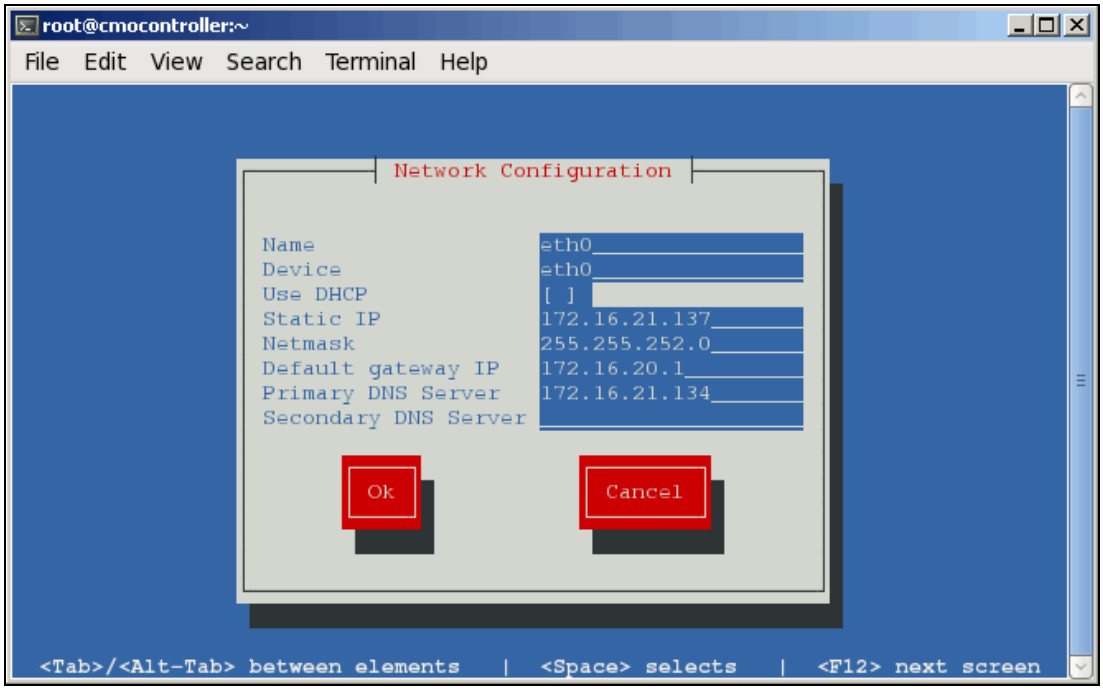


Figure 4-10 Management network configuration of the Controller node

After completing the network configuration, press **<tab>** to reach the `OK` button and conclude this step.

Modify the network configuration file to make the adapter available after every reboot and to remove the Network Manager control over this adapter:

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

Change the **ONBOOT** parameter to `yes` and the **NM_CONTROLLED** to `no`.

Run the following command to restart the network services:

```
# service network restart
```

After completing this step, the `/etc/sysconfig/network-scripts/ifcfg-eth0` configuration file should be similar to Example 4-14.

Example 4-14 /etc/sysconfig/network-scripts/ifcfg-eth0 configuration file on Controller node

```
[root@cmocontroller ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=none
NM_CONTROLLED=no
ONBOOT=yes
TYPE=Ethernet
UUID="e7b16f9f-854c-43a9-9144-67ce79d1b25d"
IPADDR=172.16.21.137
PREFIX=22
GATEWAY=172.16.20.1
DEFROUTE=yes
```

```
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="System eth0"
USERCTL=no
HWADDR=2a:e6:a3:e2:d0:02
DNS1=172.16.21.134
DOMAIN='itso.ibm.com privitso.ibm.com'
LAST_CONNECT=1406123738
NETMASK=255.255.252.0
```

The **ifconfig** command should also show the eth0 configured as shown in Example 4-15.

Example 4-15 ifconfig output for eth0 in the Controller node

```
[root@cmocontroller ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 2A:E6:A3:E2:D0:02
          inet addr:172.16.21.137  Bcast:172.16.23.255  Mask:255.255.252.0
          inet6 addr: fe80::28e6:a3ff:fee2:d002/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13351321239 errors:0 dropped:0 overruns:0 frame:0
          TX packets:40027245191 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1587354839454 (1.4 TiB)  TX bytes:4751101113205 (4.3 TiB)
          Interrupt:18
```

Make sure that your management network is working by using the **ping** command to communicate with the deployment server.

Configure the virtual machines data network interface

The data network interface is used to create a bridge to communicate with the instances (virtual machines) in your cloud. The controller node will also have a bridge in the data network, even if it does not perform a Compute node role. This bridge is created by Open vSwitch.

The Open vSwitch is a open source multilayer virtual switch, which works in hardware virtualization environments and supports several protocols and network standards, such as VLAN Tagged networks.

Note: In the cloud created for this publication, Open vSwitch is used in the Controller node, KVM and PowerKVM hypervisors. PowerVC does not use Open vSwitch.

The data network requires a dedicated network interface. It cannot have any bridge configuration and cannot have any IP Address, but the adapter must be up. The deployment server will automatically configure this adapter and the Open vSwitch bridge during the deployment process of the Controller node.

Note: This publication uses eth1 as the data network interface.

As mentioned in 4.5.5, “Network considerations and requirements” on page 219, you must use a physical adapter in case you plan to use VLAN Networks and DHCP server for your instances. In the cloud created for this publication, PowerVC was added for managing PowerVM hypervisor. We will also use DHCP to provide IP for the instances running in KVM and PowerKVM, so the eth1 used in this cloud must be physical.

If you need to add a physical adapter to the LPAR, use the HMC and select the logical partition and click **Dynamic Partitioning > Physical Adapters > Add**. Select the physical adapter and click **OK**. After the operation completes, remember to save the configuration profile by selecting the logical partition and click **Configuration > Save Current Configuration**, select your profile and click **OK**.

To check whether the eth1 interface is a physical adapter, you can use the **dmesg** and **lspci -v** commands, as shown in Example 4-16.

Example 4-16 Verifying whether the adapter is physical

```
[root@cmocontroller ~]# dmesg | grep eth1
e1000e 0000:03:00.0: eth1: (PCI Express:2.5GT/s:Width x4) 00:14:5e:e9:e1:61
e1000e 0000:03:00.0: eth1: Intel(R) PRO/1000 Network Connection
e1000e 0000:03:00.0: eth1: MAC: 0, PHY: 4, PBA No: E39336-004

[root@cmocontroller ~]# lspci -v
03:00.0 Ethernet controller: Intel Corporation 82571EB Gigabit Ethernet Controller
(Copper) (rev 06)
        Subsystem: IBM Device 0368
        Flags: bus master, fast devsel, latency 0, IRQ 37
        Memory at 3c04ffd40000 (32-bit, non-prefetchable) [size=128K]
        Memory at 3c04ffd60000 (32-bit, non-prefetchable) [size=128K]
        I/O ports at fe800 [disabled] [size=32]
        Expansion ROM at 3c04ffd80000 [disabled] [size=128K]
        Capabilities: [c8] Power Management version 2
        Capabilities: [d0] MSI: Enable+ Count=1/1 Maskable- 64bit+
        Capabilities: [e0] Express Endpoint, MSI 00
        Capabilities: [100] Advanced Error Reporting
        Capabilities: [140] Device Serial Number 00-14-5e-ff-ff-e9-e1-60
        Kernel driver in use: e1000e
        Kernel modules: e1000e
```

After providing the physical adapter, run the following command to configure the management network:

```
# system-config-network
```

Press Enter for **Device Configuration** and press Enter on your network interface that is dedicated for the data network (preferably `eth1`). *Do not* select Use DHCP, and leave all the fields blank, as shown in Figure 4-11 on page 224.

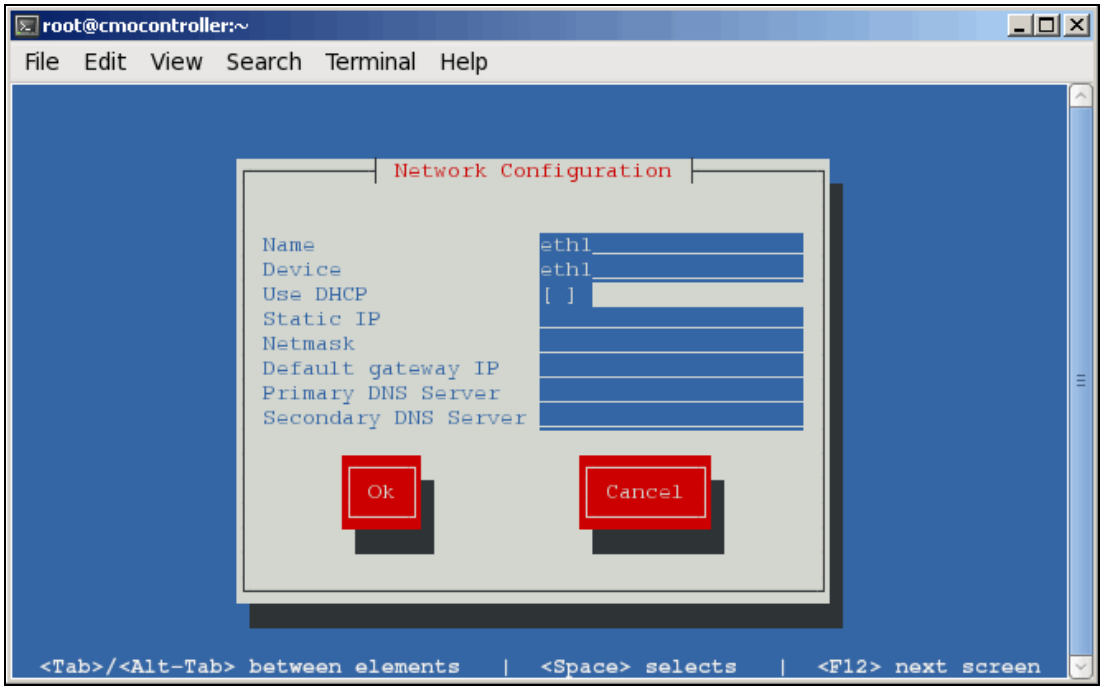


Figure 4-11 Data network configuration

Modify the configuration file to make the adapter available after every reboot and specify `none` as the protocol:

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth1
```

Change the `BOOTPROT0` parameter to `none` and the `ONBOOT` parameter to `yes`.

Run the following command to restart the network services:

```
# service network restart
```

After completing this step, the `/etc/sysconfig/network-scripts/ifcfg-eth1` configuration file should be similar to Example 4-17.

Example 4-17 Data network configuration file

```
[root@cmocontroller ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
TYPE=Ethernet
UUID=0e753f9b-4f2a-4469-abf2-22f3e6ac0999
BOOTPROTO=none
ONBOOT=yes
HWADDR=00:14:5E:E9:E1:62
DEFROUTE=no
IPV4_FAILURE_FATAL=yes
NAME="System eth1"
```

The **`ifconfig`** command should show no IP configured, but the **`mii-tool`** command should show that the link is up, as shown in Example 4-18 on page 225.

Example 4-18 ifconfig and mii-tool output from the data network in Controller node

```
[root@cmocontroller ~]# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:14:5E:E9:E1:62
          inet6 addr: fe80::214:5eff:fee9:e162/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1060668 errors:0 dropped:0 overruns:0 frame:0
          TX packets:775 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:93928964 (89.5 MiB)  TX bytes:121381 (118.5 KiB)
          Interrupt:33 Memory:3c04ffea0000-3c04ffec0000

[root@cmocontroller ~]# mii-tool eth1
eth1: negotiated 100baseTx-FD flow-control, link ok
```

4.5.7 Deploying the Controller node

To use the deployment server to deploy the Controller and the Compute nodes, you must use input files. These files will be used by the deployment server to execute specific configurations on the Controller and Compute nodes.

There are two main files that you need to prepare:

env.json Environment file, which contains most of the customizations and configuration options for your cloud

topo.json Topology file, which contains the topology definition for your cloud, including all the nodes that will participate in the cloud and their respective roles

These input files are based on the JavaScript Object Notation (JSON) format. JSON is a human-readable text notation that is used to transmit information by using attribute-value pairs. JSON files are used as an alternative to XML files.

Prepare the environment file, env.json

The `env.json` file must be prepared in the deployment server.

1. Log in to the deployment server and create a directory structure to maintain your cloud configuration files:

   ```
   # mkdir -p /root/cmo/itso-sde-deployment
   # chmod 600 /root/cmo/itso-sde-deployment
   # cd /root/cmo/itso-sde-deployment
   ```

   Note: There is no requirement regarding where to save the configuration files. Choose a safe place to keep the files. *Do not* place them in any temporary directory, such as `/tmp`, and back up the files regularly.

2. Rather than manually creating the `env.json` file from scratch, use one of the sample files available in the Chef repository. You can use the following command to query the available environments from Chef:

   ```
   # knife environment list
   ```

The expected output is shown in Example 4-19.

Example 4-19 Environments available in the Chef repository

```
[root@cmodeploy ~]# knife environment list
_default
example-ibm-os-allinone
example-ibm-os-single-controller-n-compute
example-ibm-sce
```

The `example-ibm-os-allinone` file provides only a minimal topology to your cloud, so the instances will only be able to communicate to other instances running inside the same Compute node. This is only useful for testing purposes, where a single host is used and the controller node runs x86_64 Linux, working both as a Controller and a KVM node.

The `example-ibm-os-single-controller-n-compute` file is the adequate topology for a cloud with multiple Compute nodes.

3. Use the **`knife`** command to dump the `example-ibm-os-single-controller-n-compute` environment file to the `env.json` file that you will customize for your cloud (no line break):

   ```
   # knife environment show example-ibm-os-single-controller-n-compute -d -Fjson > env.json
   ```

   Tip: This *single-line* command dumps `example-ibm-os-single-controller-n-compute` from the Chef repository to the `env.json` text file.

4. Modify the contents of the environment file by providing details of your cloud as a base for your cloud deployment by the Chef server running in the deployment server:

   ```
   # vi env.json
   ```

5. Change the Name and Description fields to information for your cloud (Example 4-20).

Example 4-20 Modifying the name and description fields in the env.json file

```
1 {
2   "name": "ITSO-SDE-CLOUD",
3   "description": "ITSO CLOUD",
```

Note: The same name must be provided later in the topology file (topo.json). The name must match, otherwise the deployment of your cloud will fail.

6. Change the `openstack.endpoints.network-openvswitch.bind_interface` and `openstack.endpoints.compute-vnc-bind.bind_interface` fields to match your management interface name, as shown in Example 4-21.

Example 4-21 Modifying the Management interface name in the env.json file

```
89     "openstack": {
90       "endpoints": {
91         "network-openvswitch": {
92           "bind_interface": "eth0"
93         },
94         "compute-vnc-bind": {
95           "bind_interface": "eth0"
96         }
97       },
```

Important: Your Management interface name does not need to be eth0. In case you were able to use eth0 for the Management interface, this section will be already correct.

7. Change the `network.openvswitch.bridge_mappings` and `network.openvswitch.bridge_mapping_interface` fields to match the interface that is being used for the data network, as shown in Example 4-22.

Example 4-22 Modifying the data network fields in the env.json file

```
107       "network": {
108         "verbose": false,
109         "debug": false,
110         "openvswitch": {
111           "bridge_mappings": "default:br-eth1",
112           "bridge_mapping_interface": "br-eth1:eth1"
113         }
114       },
```

The Chef server uses this information to set up the Open vSwitch in the Controller and Compute nodes. The `br-eth1` bridge is created, and the `eth1` interface is used to provide external network access to instances running in each KVM or PowerKVM Compute node.

Note: If Open vSwitch is already manually installed in the Compute nodes and bridges to the data network were already created, you can set this configuration to **“ ”**. If specific network configuration is required for the data network, such as using bonding, you must change this configuration to **“”** and complete the setup manually. This option is not covered in this book.

Important: The PowerKVM nodes use a different naming convention for the network interfaces, so the data network interface will not be called eth1, but will have a name according to its slot number, for example `enP3p1s0f1`. This will be modified later by using a node-specific attribute file, which will be only relevant for the PowerKVM nodes. You should match the interface name used in step 7 to your Controller node data network interface.

8. Change the `endpoints.host`, `endpoints.bind-host`, `endpoints.mq.host`, and `endpoints.db.host` fields to match the IP address of the Controller node (Example 4-23).

Example 4-23 Changing the Controller IP address in the env.json file

```
215       "endpoints": {
216         "host": "172.16.21.137",
217         "bind-host": "172.16.21.137",
218         "mq": {
219           "host": "172.16.21.137",
220           "port": "5671"
221         },
222         "db": {
223           "host": "172.16.21.137",
224           "port": "50000"
225         }
226       },
```

9. Unless you plan to use GRE networks (available only when all nodes from your OpenStack cloud are x86_64 KVM Compute nodes), disable tunneling. Change the `openstack.openvswitch.enable_tunneling` field to `False` as shown in Example 4-24.

Example 4-24 Disable tunneling in the env.json file

```
273         "openvswitch": {
274           "tenant_network_type": "vlan",
275           "network_vlan_ranges": "default:1:4094",
276           "enable_tunneling": "False",
277           "tunnel_type": "gre",
278           "tunnel_id_ranges": "1:1000"
279         }
280       },
```

With these modifications, your environment file (`env.json`) is ready for the cloud deployment. More modifications to this file will be done when we start adding hypervisors to the cloud. For more information about possible configurations and modifications to this file, see Chapter 5 of the IBM Cloud Manager with OpenStack 4.1 Administrator Guide:

http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html

10. Run the following command to upload the modified environment file (`env.json`) to the Chef server:

```
# knife environment from file env.json
Updated Environment ITSO-SDE-CLOUD
```

Create the topology file, topo.json

The topology file is where you describe the cloud topology for the deployment server to use Chef to deploy it. Initially you will provide only the Controller node and deploy your cloud. Then, you will add the hypervisors to your cloud by modifying this file (and the `env.json`), adding hypervisor details and updating or redeploying your cloud.

1. Create a simple topology file by copying and pasting the content of Example 4-25 to your deployment server. The topology file can have any name, but you should use some name that reminds you of its role, such as topo.json. You should also place this file inside the directory structure you create in step 1 on page 225.

Example 4-25 Sample topo.json file

```
{
  "name":"topo-$CLOUD_NAME",
  "description":"topo-$CLOUD_NAME",
  "environment":"$CLOUD_NAME-env",
  "run_sequentially":false,
  "nodes": [
    {
      "fqdn":"$NODE1_HOSTNAME",
      "password":"<root_password>",
      "quit_on_error":true,
      "run_order_number":1,
      "runlist": [
        "role[ibm-os-allinone-kvm]"
      ]
    }
  ]
}
```

2. Modify this file to provide details of your topology:

```
# vi topo.json
```

3. Change the `name`, `description` and `environment` entries as shown in Example 4-26.

Example 4-26 Modifying name, description and environment information in the topo.json file

```
1 {
2   "name":"topo-ITSO-SDE-CLOUD",
3   "description":"topo-ITSO-SDE-CLOUD",
4   "environment":"ITSO-SDE-CLOUD",
```

Important: The `environment` field from the topo.json file must match the `name` field from the env.json file. In case the contents do not match, the deployment will fail with the following error message:

```
Environment 'ITSO-SDE-CLOUD-env' could not be found.
Would you like to create the environment 'ITSO-SDE-CLOUD-env' now? (y/n):
[n] n
ERROR: Topology verification error: Environment 'ITSO-SDE-CLOUD-env' not
found.
ERROR: RuntimeError: Topology verification failed.
```

4. Change the `nodes.fqdn`, `nodes.password`, `nodes.runlist.role` and provide, respectively, the fully qualified domain name, root password and role of the Controller node.

Important: There are two possible roles for your Controller node:

- Role `ibm-os-allinone-kvm`: Use this role if you are running your Controller node on x86_64 RHEL 6.5 Linux and you plan to use the Controller node as a Compute node also. In this case, the Controller node will have two roles: the Controller role and a KVM role.
- Role `ibm-os-single-controller-node`: Use this role if you are running your Controller node either on x86_64 or ppc64 (Power) RHEL 6.5 Linux and you plan to use the Controller node as dedicated for the Controller role.

In this publication, we use the `ibm-os-single-controller-node` role.

Example 4-27 shows a topo.json file prepared to deploy a single controller node.

Example 4-27 Topology file prepared to deploy a single controller node

```
1 {
2   "name":"topo-ITSO-SDE-CLOUD",
3   "description":"topo-ITSO-SDE-CLOUD",
4   "environment":"ITSO-SDE-CLOUD",
5   "run_sequentially":false,
6   "nodes": [
7     {
8       "fqdn":"cmocontroller.itso.ibm.com",
9       "password":"itsoadmin",
10      "quit_on_error":true,
11      "run_order_number":1,
12      "runlist": [
13        "role[ibm-os-single-controller-node]"
14      ]
```

```
15     }
16   ]
17 }
```

Prepare the data bags

A data bag is a group of configuration files that follow the JSON format to provide authentication information for the Chef server. With this information, the Chef server is able to log in to the Controller and Compute nodes to install and configure the required components to deploy your cloud. The data bags contain user IDs, passwords and secret keys for hypervisors and services that run in each of them.

The Cloud Manager installation provides example data bags, which we can use as a base to our environment rather than creating them from scratch. In the following steps, we will copy these sample files to the directory that contains our deployment structure (env.json and topo.json files) and modify the passwords, if needed. These data bags will be referenced in the env.json and topo.json files to deploy your cloud.

For more information about data bags, see the “About Data Bags” page on the Chef website:

http://docs.chef.io/data_bags.html

1. Access the directory that contains the structure of your cloud (`env.json` and `topo.json` files):

   ```
   # cd /root/cmo/itso-sde-deployment
   ```

2. Create a secret key for encrypting the data bags. A secret key can be created in many ways. Example 4-28 uses the OpenSSL **`rand`** command to generate pseudo-random bytes that can be used as the secret key.

   Example 4-28 Creating the secret key to encrypt the data bags

   ```
   [root@cmodeploy itso-sde-deployment]# openssl rand -base64 512 | tr -d .\r\n. >
   itso-secret-key
   ```

   Note: There is no line break in the command above.

3. Set the permissions of the secret keys file to `600`:

   ```
   # chmod 600 itso-secret-key
   ```

4. Copy the example data bags to your cloud directory, and set the permissions to 600.

   ```
   # cd /root/cmo/itso-sde-deployment
   # mkdir data_bags
   # cp -r /opt/ibm/cmwo/chef-repo/data_bags/db_passwords
   data_bags/itso_sde_db_passwords
   # cp -r /opt/ibm/cmwo/chef-repo/data_bags/secrets data_bags/itso_sde_secrets
   # cp -r /opt/ibm/cmwo/chef-repo/data_bags/service_passwords
   data_bags/itso_sde_service_passwords
   # cp -r /opt/ibm/cmwo/chef-repo/data_bags/user_passwords
   data_bags/itso_sde_user_passwords
   # chmod -R 600 data_bags/
   ```

   Note: There is an additional data bag named `appliance32_passwords`, but it is only required when you are migrating from IBM SmartCloud Entry to IBM Cloud Manager.

   These steps assume that the default Cloud Manager installation path was used on the deployment server (`/opt/ibm/cmwo`).

5. If you need to change the user and password for the Cloud Manager services, you can change the JSON files in the data bags. In this publication, we kept the default users and passwords.
6. Upload the data bags to the Chef server. Run the following commands from the parent of the `data_bags` directory that you created in step 4 on page 230:

```
# knife data bag create itso_sde_db_passwords
Created data_bag[itso_sde_db_passwords]
# knife data bag create itso_sde_secrets
Created data_bag[itso_sde_secrets]
# knife data bag create itso_sde_service_passwords
Created data_bag[itso_sde_service_passwords]
# knife data bag create itso_sde_user_passwords
Created data_bag[itso_sde_user_passwords]

# knife data bag from file itso_sde_db_passwords --all --secret-file
itso-secret-key
Updated data_bag_item[itso_sde_db_passwords::keystone]
Updated data_bag_item[itso_sde_db_passwords::horizon]
Updated data_bag_item[itso_sde_db_passwords::glance]
Updated data_bag_item[itso_sde_db_passwords::nova]
Updated data_bag_item[itso_sde_db_passwords::powervc]
Updated data_bag_item[itso_sde_db_passwords::ceilometer]
Updated data_bag_item[itso_sde_db_passwords::cinder]
Updated data_bag_item[itso_sde_db_passwords::neutron]
Updated data_bag_item[itso_sde_db_passwords::heat]

# knife data bag from file itso_sde_secrets --all --secret-file itso-secret-key
Updated data_bag_item[itso_sde_secrets::openstack_metering_secret]
Updated data_bag_item[itso_sde_secrets::openstack_vmware_secret_name]
Updated data_bag_item[itso_sde_secrets::neutron_metadata_secret]
Updated data_bag_item[itso_sde_secrets::openstack_simple_token]
Updated data_bag_item[itso_sde_secrets::openstack_identity_bootstrap_token]

# knife data bag from file itso_sde_service_passwords --all --secret-file
itso-secret-key
Updated data_bag_item[itso_sde_service_passwords::openstack-block-storage]
Updated data_bag_item[itso_sde_service_passwords::openstack-ceilometer]
Updated data_bag_item[itso_sde_service_passwords::openstack-orchestration]
Updated data_bag_item[itso_sde_service_passwords::openstack-compute]
Updated data_bag_item[itso_sde_service_passwords::openstack-image]
Updated data_bag_item[itso_sde_service_passwords::openstack-powervc-driver]
Updated data_bag_item[itso_sde_service_passwords::openstack-iaas-gateway]
Updated data_bag_item[itso_sde_service_passwords::openstack-network]

# knife data bag from file itso_sde_user_passwords --all --secret-file
itso-secret-key
Updated data_bag_item[itso_sde_user_passwords::pvcqpid]
Updated data_bag_item[itso_sde_user_passwords::db2inst1]
Updated data_bag_item[itso_sde_user_passwords::zlinuxroot]
Updated data_bag_item[itso_sde_user_passwords::xcat]
Updated data_bag_item[itso_sde_user_passwords::admin]
Updated data_bag_item[itso_sde_user_passwords::db2fenc1]
Updated data_bag_item[itso_sde_user_passwords::qpidadmin]
Updated data_bag_item[itso_sde_user_passwords::sceagent]
```

```
Updated data_bag_item[itso_sde_user_passwords::pvcadmin]
Updated data_bag_item[itso_sde_user_passwords::qpidclient]
Updated data_bag_item[itso_sde_user_passwords::qpidssl]
Updated data_bag_item[itso_sde_user_passwords::db2das1]
```

7. Change the `openstack.secret.service_passwords_data_bag`, `openstack.secret.db_passwords_data_bag`, `openstack.secret.user_passwords_data_bag and openstack.secret.secrets_data_bag` attributes in the env.json files to match the data bags you created on step 4 on page 230:

   ```
   # vi /root/cmo/itso-sde-deployment/env.json
   ```

   Example 4-29 shows how these attributes should be changed to match your data bags names.

   Example 4-29 Modifying the data bags names in the env.json file

   ```
   183       "secret": {
   184         "key_path": "/etc/chef/encrypted_data_bag_secret",
   185         "service_passwords_data_bag": "itso_sde_service_passwords",
   186         "db_passwords_data_bag": "itso_sde_db_passwords",
   187         "user_passwords_data_bag": "itso_sde_user_passwords",
   188         "secrets_data_bag": "itso_sde_secrets"
   189       },
   ```

   a. Modify your topology file (`topo.json`) to add the `secret_file` attribute to it:

   ```
   # vi /root/cmo/itso-sde-deployment/
   ```

   Add the "`secret_file`" attribute between the "`environment`" and "`run_sequentially`" attributes and point it to the secret file created in step 2 on page 230 as shown in Example 4-30.

   Example 4-30 Adding secret_file attribute to the topology file topo.json

   ```
   1 {
   2   "name":"topo-ITSO-SDE-CLOUD",
   3   "description":"topo-ITSO-SDE-CLOUD",
   4   "environment":"ITSO-SDE-CLOUD",
   5   "secret_file":"/root/cmo/itso-sde-deployment/itso-secret-key",
   6   "run_sequentially":false,
   7   "nodes": [
   8      {
   9        "fqdn":"cmocontroller.itso.ibm.com",
   10       "password":"itsoadmin",
   11       "quit_on_error":true,
   12       "run_order_number":1,
   13       "runlist": [
   14          "role[ibm-os-single-controller-node]"
   15       ]
   16     }
   17   ]
   18 }
   ```

8. Run the following command to upload the modified environment file (`env.json`) to the Chef server:

   ```
   # knife environment from file env.json
   Updated Environment ITSO-SDE-CLOUD
   ```

Important: This step is essential, otherwise the information uploaded to the Chef server will be incorrect, and the deployment of your cloud may fail with the following error:

```
Error decrypting data bag value: 'bad decrypt'. Most likely the provided
key is incorrect
```

9. Run the **knife environment list** command to check whether the Chef server contains your environment details. The expected output will be similar to Example 4-31.

Example 4-31 Output of the command knife environment list

```
[root@cmodeploy itso-sde-deployment]# knife environment list
ITSO-SDE-CLOUD
_default
example-ibm-os-allinone
example-ibm-os-single-controller-n-compute
example-ibm-sce
```

Temporarily stop the iptables service on the Controller node

Before deploying your cloud, stop the iptables service to avoid problems with the Chef communication:

```
# service iptables stop
```

Note: Do not disable the iptables service with the **chkconfig** command. The Chef server will configure additional rules and use iptables for the neutron component. According to the experience of the IBM Redbooks residency team for this book, we can avoid problems accessing the Cloud Manager dashboard after the cloud deployment by temporarily disabling the iptables. It will be automatically enabled with the new rules after the deployment.

Use the knife command to deploy your cloud

Now you are ready to deploy your OpenStack cloud. In this step, the deployment server will effectively install the Cloud Manager components in the Controller node, configure and start all the necessary services. Before proceeding, read the following important notes.

Important: It is critical to complete all the steps described in 4.5.4, “Customize Red Hat Enterprise Linux to host the Controller node” on page 219, especially setting up the Yum repository. Make sure that you have completed these steps before proceeding. Otherwise, the cloud deployment might fail.

Important: There is no support to uninstall a deployed topology from the Cloud Manager. In this case, you need to reinstall or reset the node back to the state it had before deploying your cloud. You cannot redeploy the same physical node before setting it back to its predeployment state. Make sure to back up your node by using backup or snapshot capabilities before deploying you cloud in case you cannot reinstall it if needed.

1. Access the directory where you have the `topo.json` file:

```
# cd /root/cmo/itso-sde-deployment/
```

2. Use the **knife** command to deploy your OpenStack cloud by using the topology file (topo.json), which will also point to the environment that was uploaded to the Chef server on step 8 on page 232:

```
# knife os manage deploy topology topo.json
```

The expected output is too long to fully show it in this publication, but Example 4-32 partially shows the expected output of the **knife os manage deploy topology topo.json** command.

Example 4-32 Sample output of the Controller node deployment

```
[root@cmodeploy itso-sde-deployment]# knife os manage deploy topology topo.json
Deploying topology 'topo-ITSO-SDE-CLOUD' ...
Deploying to nodes with run_order_number '1'
Bootstraping nodes with environment ITSO-SDE-CLOUD...
Bootstrapping node ...
Connecting to cmocontroller.itso.ibm.com
cmocontroller.itso.ibm.com INFO: Adding trusted certificate for Chef server:
cmodeploy.itso.ibm.com
cmocontroller.itso.ibm.com Starting Chef Client, version 11.12.4
.
. <many lines were removed>
.
cmocontroller.itso.ibm.com Running handlers complete
cmocontroller.itso.ibm.com
cmocontroller.itso.ibm.com Chef Client finished, 375/537 resources updated in
718.413788029 seconds
All bootstrapped nodes with run_order_number '1' deployed.

Results for deploy of topology 'topo-ITSO-SDE-CLOUD'
Results for nodes with run_order_number '1'
Deploy of node at cmocontroller.itso.ibm.com was successful.
Deploy of topology 'topo.json' completed in 1002 seconds.
```

Access the Controller node using your browser

After these steps, use your browser with https protocol to point to the Controller node and it should be accessible:

```
https://<your_controller_node_ip_address>
```

You should be prompted for a `user name` and `password` to log in to the Cloud Management dashboard, as shown in Figure 4-12 on page 235. The Cloud Management dashboard is the main web interface for cloud management, where you can create images and volumes, set up the network, deploy (create) and terminate (delete) instances based on the images and other management tasks.

Note: The Cloud Management dashboard is also referred as the *Horizon* dashboard because it is based on the Horizon OpenStack project. For more information about Horizon, see the Horizon project website:

http://docs.openstack.org/developer/horizon/

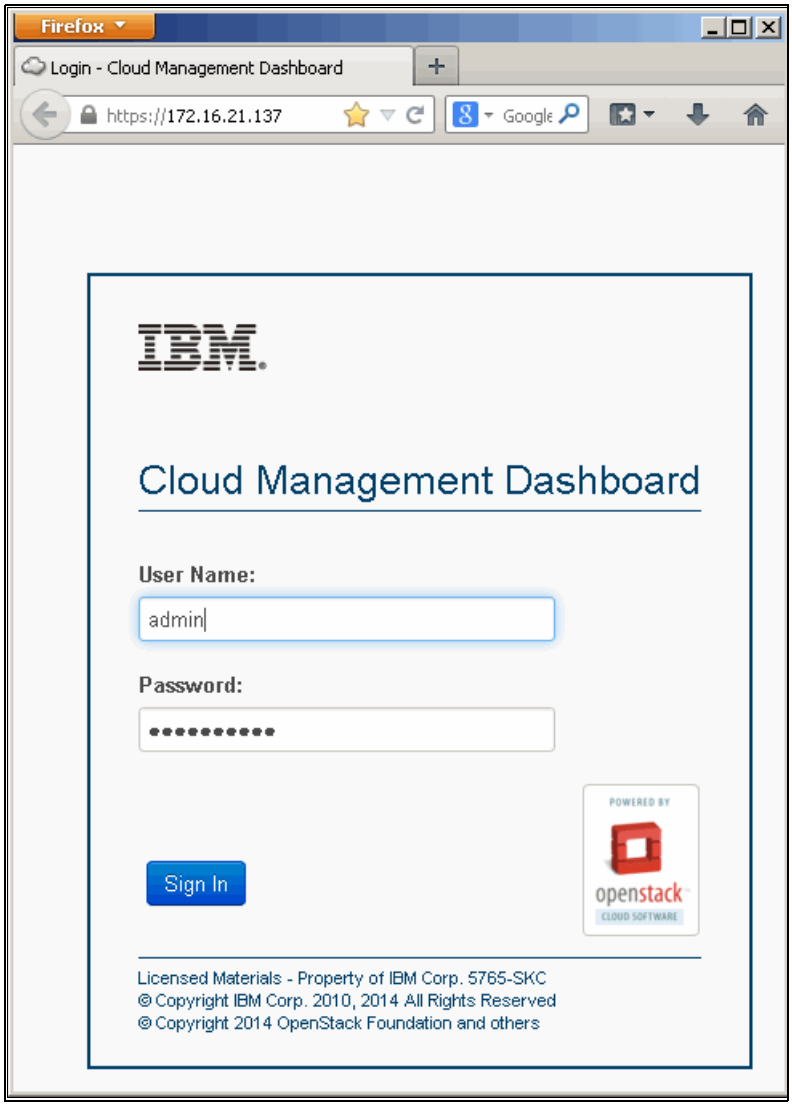


Figure 4-12 Cloud Management (Horizon) dashboard login

The user name for administration purposes is `admin` and the default initial password is `openstack1` (unless you modified this password in the step 5 on page 231). The file `/root/openrc` in the Controller node was created by Chef and contains the admin username and password that you can use to log in to the Cloud Manager dashboard, as shown in Example 4-33.

Example 4-33 Sample output of the /root/openrc file

```
[root@cmocontroller ~]# cat /root/openrc
# This file autogenerated by Chef
# Do not edit, changes will be overwritten

# COMMON OPENSTACK ENVS
export OS_USERNAME=admin
export OS_PASSWORD=openstack1
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://172.16.21.137:5000/v2.0
export OS_REGION_NAME=RegionOne
```

Note: Do not remove the `/root/openrc`. It contains very restrictive permissions and will be readable only by the root user in the Controller node. This file is required to use the command-line interface for OpenStack components such as Nova, Neutron, Cinder, Glance, and Keystone.

4.6 Add KVM Compute nodes to the cloud

The process of adding KVM Compute nodes (KVM or QEMU hypervisors) to the OpenStack cloud involves preparing the physical host by installing and customizing the Red Hat operating system on the System x server and using the deployment server to add the Compute node to the cloud. In this process, the Chef server will automatically install and configure the required packages in the KVM Compute node, including installing the Open vSwitch and creating the network bridge to provide external access to the instances that will run in this node. It will also add the Compute node information to the Controller node, bringing the KVM Compute node under the management of the Controller node. After this process, the Compute node will be available for instances deployment in the Cloud Management dashboard.

The KVM software can run on x86 or x86_64 servers that support hardware-assisted virtualization. Run the following command to determine if your system processor supports KVM:

```
# grep -E 'vmx|svm' /proc/cpuinfo
```

The expected output is similar to what is shown in Example 4-34.

Example 4-34 Checking if the processor supports KVM

```
[root@cmo1 ~]# grep -E 'vmx|svm' /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm
constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf pni
dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm dca sse4_1 sse4_2 popcnt
lahf_lm dts tpr_shadow vnmi flexpriority ept vpid
```

The server must run Red Hat Enterprise Linux 6.4 or 6.5, and the KVM software (available in the Red Hat installation media) must be previously installed in the server. The installation process is very simple if you do not have any virtual machine or any specific KVM configuration in the server, so the Chef server will be able to do all of these configuration steps automatically. You cannot have any Linux bridge configured, given that the OpenStack architecture uses Open vSwitch to set up the bridge.

The following steps are required to add the KVM Compute node to your OpenStack cloud:

1. Enable KVM support on your hardware.
2. Install Red Hat Enterprise Linux on the physical host.
3. Customize RHEL to meet the Cloud Manager requirements.
4. Configure management and data network interfaces.
5. Set up a Yum repository.
6. Install KVM packages on the physical host.
7. Deploy the KVM Compute node using the deployment server.

4.6.1 Verify software prerequisites for KVM Compute nodes

To add KVM Compute nodes to your cloud, the Compute node must meet the following software prerequisites:

- Red Hat Enterprise Linux 6.4 or 6.5
- A libvirt version 0.9.11 or later
- Open vSwitch version 2.0 (Chef server will create a Yum repository in the Compute node and automatically install the Open vSwitch software from this Yum repository, which is available at the deployment server)
- Python version 2.6.x

Note: Verify whether the KVM host device drivers are compatible with Open vSwitch, and update the host device drivers if necessary. The hardware used in this publication did not require any device driver update to work with Open vSwitch. IBM testing with Red Hat Enterprise Linux 6.4 or 6.5 on an IBM PureFlex System required the following device drivers updates:

```
kmod-be2iscsi-4.6.267.4-1.x86_64
kmod-tg3-3.129d-1.x86_64
kmod-elx-lpfc-8.3.7.29-1.x86_64
kmod-be2net-4.6.267.4-1.x86_64
kmod-brocade-bfa-3.2.1.1-0.x86_64
kmod-qlgc-qla2xxx-8.04.00.12.06.0_k3-1.x86_64
```

Updates to the IBM hardware device drivers can be found at the following website:

`http://ibm.com/support`

Also, consider reviewing the network details on 4.5.5, “Network considerations and requirements” on page 219.

4.6.2 Enable KVM support on your hardware

You need to enable KVM support on your hardware using the BIOS/Setup utility. This step may vary according to the hardware being used as your KVM Compute node. In general, the steps should be similar to the ones that follow:

1. Power off your System x server.
2. Power on the server and enter the Setup/BIOS utility (usually by pressing the F1 key during boot).
3. In the BIOS menu, select **Advanced Setup > CPU Options**.
4. Confirm that the Intel Virtualization Technology option is **Enabled**.
5. Save the settings and exit the BIOS utility.

Important: If you already have installed Red Hat Enterprise Linux operating system in this hardware, you can use the instructions in 4.6, “Add KVM Compute nodes to the cloud” on page 236. If the **`grep -E 'vmx|svm' /proc/cpuinfo`** command already provides the expected output, this means that this feature is enabled and you can skip this step.

4.6.3 Install Red Hat Enterprise Linux on the physical host

Obtain the Red Hat Enterprise Linux 6.4 or 6.5 for x86_64 architecture and install it in the physical host that will be your KVM Compute node in your OpenStack cloud. For detailed instructions on how to install RHEL, see the Red Hat Enterprise Linux Installation Guide

http://red.ht/1xEzlge

Note: You do not need to install additional packages at this time. You will be instructed to set up a Yum repository on this host in 4.6.6, “Set up a Yum repository” on page 242 and to install KVM packages by using this repository in 4.6.7, “Install KVM packages on the physical host” on page 243. Other required packages will be automatically installed by Chef server using Yum repositories during the deployment.

4.6.4 Customize RHEL to meet the IBM Cloud Manager requirements

Some requirements must be attended to add the KVM Compute node to your OpenStack cloud. The following topics will guide you to complete these requirements.

Use a fully qualified domain name

Follow the instructions from “Use a fully qualified domain name” on page 204 to configure your KVM Compute node to use fully qualified domain name (FQDN).

Configure hostname resolution

Follow the instructions from “Configure host name resolution” on page 204 to configure your KVM Compute node to be able to resolve hostnames to IP and IP to hostnames using DNS or `/etc/hosts`.

Synchronize the date and time

Follow the instructions from “Synchronize the date and time” on page 205 to configure your KVM Compute node with the same date and time as the other nodes in your cloud. You can use NTP client or manually setup the date and time to complete this requirement.

Define the same root password on the nodes

It can be a convenience to apply the same root password on the nodes, in case the security policies allow it. Consider applying the same recommendations from “Define the same root password on the nodes” on page 206 in your KVM Compute node.

4.6.5 Configure management and data network interfaces

The network configuration of the KVM Compute node is similar to the steps performed in 4.5.6, “Configure network interfaces in the Controller node” on page 220. It consists of two steps:

1. Configuring the management network interface.
2. Configuring the data network interface.

Configure the management network interface

The management network is used exclusively for management purposes, including communication between deployment server, Controller node and Compute nodes. It must have a dedicated network adapter without any bridge configured on top of it.

Important: The name of the management network interface must match what was specified on “Prepare the environment file, env.json” on page 225, more specifically step 6 on page 226, otherwise it would require creating a node-specific attribute file before deploying your cloud. This publication will consider that eth0 is used as the management network interface, matching the configuration in the env.json file in the deployment server.

Run the following command to configure the management network:

```
# system-config-network
```

Press Enter on **Device Configuration** and Enter on your network interface that is dedicated for the management network (preferably `eth0`). Provide the IP address, network mask, default gateway and DNS configuration suitable for your network as shown in Figure 4-13.

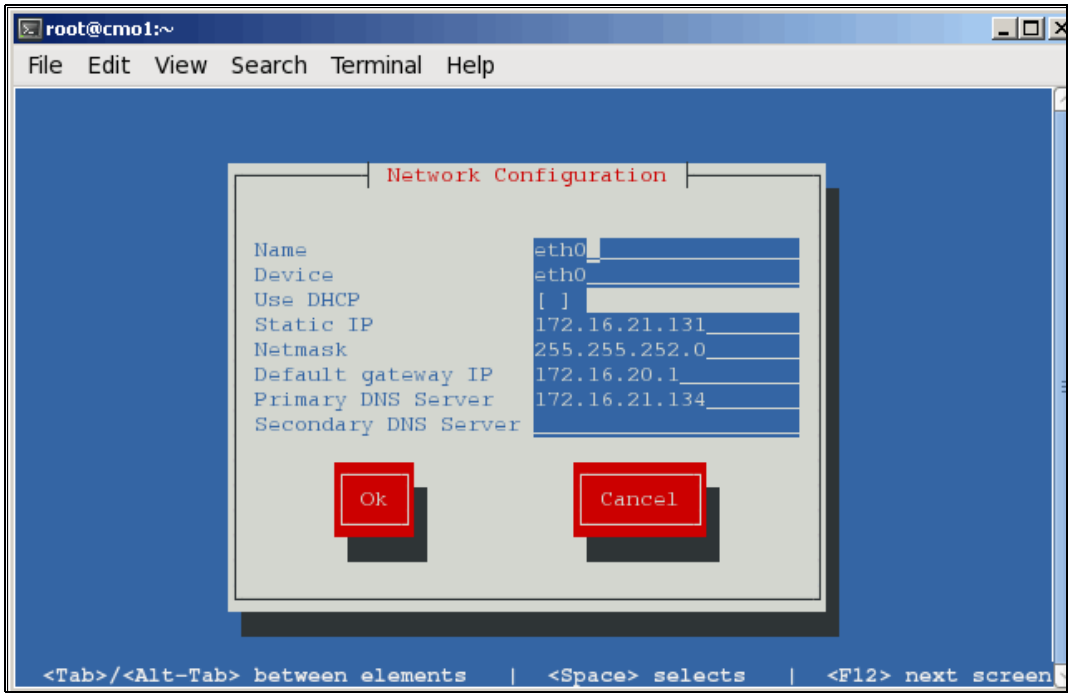


Figure 4-13 Management network configuration on KVM Compute node

After completing the network configuration, press Tab to reach the **OK** button, and click it to finish this step.

Modify the network configuration file to make the adapter available after every reboot and remove the Network Manager control over this adapter:

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

Change the `ONBOOT` parameter to `yes` and **`NM_CONTROLLED`** to `no`.

Run the following command to restart the network services:

```
# service network restart
```

After completing this step, the /etc/sysconfig/network-scripts/ifcfg-eth0 configuration should be similar to Example 4-35.

Example 4-35 /etc/sysconfig/network-scripts/ifcfg-eth0 configuration on KVM Compute node

```
[root@cmo1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=00:1a:64:b4:7c:f0
TYPE=Ethernet
UUID=65aa1ab9-48e7-4142-b3b4-8d36d9a0082f
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=none
IPV6INIT=no
USERCTL=no
IPADDR=172.16.21.131
NETMASK=255.255.252.0
GATEWAY=172.16.20.1
DNS1=172.16.21.134
```

The **ifconfig** command should also show the eth0 configured as shown in Example 4-36.

Example 4-36 ifconfig output for eth0 in the KVM Compute node

```
[root@cmo1 ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:1A:64:B4:7C:F0
          inet addr:172.16.21.131  Bcast:172.16.23.255  Mask:255.255.252.0
          inet6 addr: fe80::21a:64ff:feb4:7cf0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23908359992 errors:0 dropped:0 overruns:0 frame:0
          TX packets:71153166671 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2951030925784 (2.6 TiB)  TX bytes:8765702196536 (7.9 TiB)
```

Make sure that your management network is working by using the **ping** command to communicate with the deployment server and Controller node.

Configure the data network interface

The data network interface is used to create a bridge to provide external access to the instances (virtual machines) running in this Compute node. This bridge *should not* be manually created by modifying the /etc/sysconfig/network-scripts/ifcfg-ethX file. The deployment server will automatically install the Open vSwitch in this Compute node and create the network bridge during the step 4.6.8, “Deploy the KVM Compute node using the deployment server” on page 245.

The data network requires a dedicated network interface. It cannot have any bridge configuration and cannot have any IP address, but the adapter must be up. The bridge will be created by the deployment server using the configuration specified in the env.json file in “Prepare the environment file, env.json” on page 225 specifically in step 7 on page 227.

From Example 4-22 on page 227, we can see the following configuration:

```
107       "network": {
108         "verbose": false,
109         "debug": false,
110         "openvswitch": {
111           "bridge_mappings": "default:br-eth1",
112           "bridge_mapping_interface": "br-eth1:eth1"
113         }
114       },
```

This means that the deployment server will use Chef to install the Open vSwitch software in this KVM Compute node and will automatically create a bridge called `br-eth1` connected to an Open vSwitch network called `default` and it will use the physical adapter `eth1` to provide external access to this network.

Note: It is important to make sure that the network interface used for the data network matches the name specified in the sections `network.openvswitch.bridge_mappings` and `network.openvswitch_interface` in the `env.json` file in the deployment server, otherwise this configuration would require a node-specific attribute file.

This publication uses `eth1` as the data network, matching the configuration specified in the `env.json` file.

Run the following command to configure the data network:

```
# system-config-network
```

Press Enter on **Device Configuration** and press Enter on your network interface that it dedicated for the data network (preferably eth1). *Do not* select Use DHCP, and leave all of the fields blank, as shown in Figure 4-14.

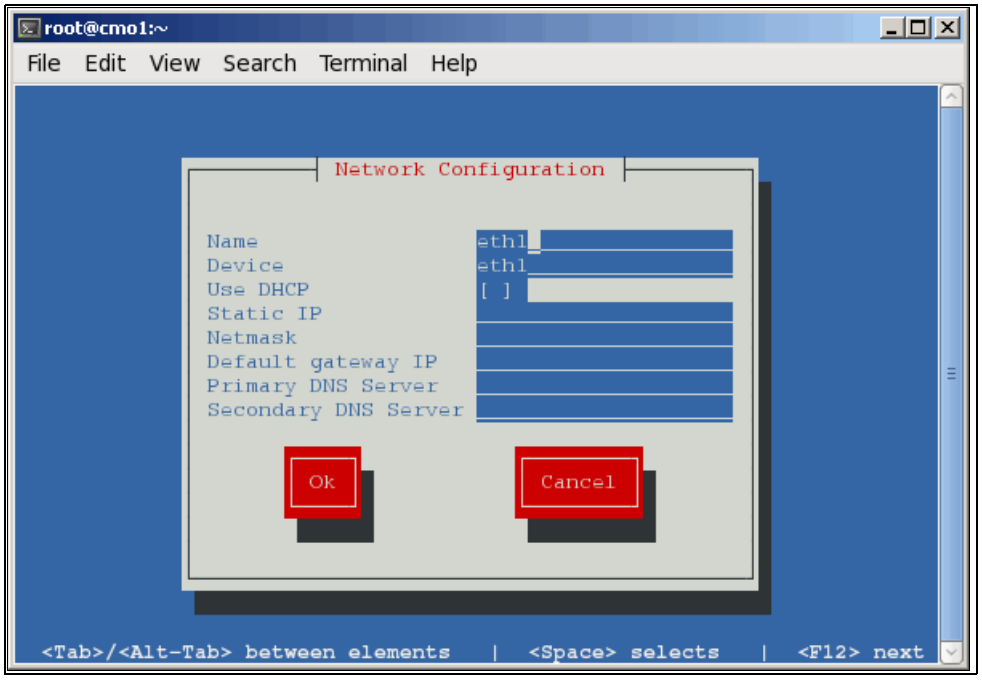


Figure 4-14 Data network interface configuration in the KVM Compute node

Use the **<tab>** key to reach the `OK` button and conclude this process.

Modify the configuration file to make the adapter available after every reboot and specify `none` as the protocol:

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth1
```

Change the **`BOOTPROTO`** parameter to `none` and the **`ONBOOT`** parameter to `yes`.

Run the following command to restart the network services:

```
# service network restart
```

After completing this step, the /etc/sysconfig/network-scripts/ifcfg-eth1 configuration file should be similar to Example 4-37.

Example 4-37 /etc/sysconfig/network-scripts/ifcfg-eth1 configuration file in KVM Compute node

```
[root@cmo1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
HWADDR=00:1a:64:b4:7c:f2
TYPE=Ethernet
UUID=d5ad8624-78c4-4c5d-865e-dd6c1776de4f
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=static
IPV6INIT=no
USERCTL=no
```

The **`ifconfig`** command shows no IP configured, but the **`mii-tool`** command shows that the link is up, as shown in Example 4-38.

Example 4-38 ifconfig and mii-tool output of the data network interface in the KVM Compute node

```
[root@cmo1 ~]# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:1A:64:B4:7C:F2
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4472259 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26368 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:367995287 (350.9 MiB)  TX bytes:2278397 (2.1 MiB)

[root@cmo1 ~]# mii-tool eth1
eth1: negotiated 100baseTx-FD flow-control, link ok
```

4.6.6 Set up a Yum repository

The deployment server will use Yum to automatically install the dependency packages when adding your KVM Compute node to the OpenStack cloud. During the deployment process one Yum repository will be automatically setup by Chef server to install OpenStack packages, but it also need to install packages from the Red Hat Enterprise Linux installation media, so you need to manually configure a Yum repository to the installation media before adding your KVM Compute node to the OpenStack cloud by using the deployment server.

Follow these steps to configure a Yum repository with the Red Hat Enterprise Linux installation media:

1. Copy the ISO file of the installation media which you used to install the KVM Compute node to a file system in the host. Make sure that the media is compatible with the architecture of the physical host (in this case x86_64). Then add the following line to the `/etc/fstab` file to mount this ISO image during boot time:

   ```
   /root/RHEL6.5-20131111.0-Server-x86_64-DVD1.iso /media/cdrom/   iso9660
   loop,ro,auto 0 0
   ```

   Note: The first entry is the full path to the ISO file which was copied to the KVM Compute node. This entry must match the name and location you used in your system. The second entry is where the ISO will be mounted, so the directory must exist in your system.

2. Run the **`mount -a`** command to mount the ISO image:

   ```
   # mount -a
   ```

3. Copy the `media.repo` file from your ISO image to the `/etc/yum.repos.d/` directory:

   ```
   # cp /media/cdrom/media.repo /etc/yum.repos.d/
   ```

4. Modify the /etc/yum.repos.d/media.repo file as shown in Example 4-39.

Example 4-39 media.repo Yum repository in the KVM Compute node

```
[InstallMedia]
name=Red Hat Enterprise Linux 6.5
mediaid=1384196515.415715
metadata_expire=-1
enabled=1
gpgcheck=0
cost=500
baseurl=file:///media/cdrom/Server
```

5. Verify that the repository you created is available to Yum as shown in Example 4-40.

Example 4-40 Yum repolist output from the KVM Compute node

```
[root@cmo1 ~]# yum repolist
Loaded plug-ins: product-id, refresh-packagekit, security, subscription-manager
repo id                  repo name                                      status
InstallMedia             Red Hat Enterprise Linux 6.5                    3,690
repolist: 4,260
```

Note: Make sure that all the repositories which are currently enabled are working. If possible, disable other repositories temporarily, keeping only the installation media enabled. In case other repositories are enabled but not working, the deployment process to add the KVM Compute node to your OpenStack cloud might fail, because it will try to use the **`yum`** command, which might fail.

4.6.7 Install KVM packages on the physical host

At this point, you can install KVM-related packages by using the Yum repository you have setup in section 4.6.6, “Set up a Yum repository” on page 242. Use the following commands to install all KVM-related packages:

```
# yum groupinstall Virtualization
# yum groupinstall "Virtualization Client"
# yum groupinstall "Virtualization Platform"
# yum groupinstall "Virtualization Tools"
# yum install qemu-guest-agent qemu-kvm-tools
# yum install fence-virtd-libvirt fence-virtd-multicast fence-virtd-serial
libvirt-cim libvirt-java libvirt-snmp perl-Sys-Virt
# yum install libguestfs-java libguestfs-tools virt-v2v
# yum install libvirt-python
```

After the installation is complete, use the following command to make sure that you have installed all required packages:

```
# rpm -qa | egrep
"qemu-kvm|hypervkvpd|qemu-guest-agent|qemu-kvm-tools|python-virtinst|virt-manager|
virt-viewer|virt-top|libvirt|libvirt-client|virt-who|fence-virtd-libvirt|fence-vir
td-multicast|fence-virtd-serial|libvirt-cim|libvirt-java|libvirt-snmp|perl-Sys-Vir
t|libguestfs|libguestfs-java|libguestfs-tools|virt-v2v|libvirt-python"
```

Note: This is a single-line command. There are no line breaks and no spaces between words inside the quotes.

The expected output is similar to what is shown in Example 4-41.

Example 4-41 KVM packages installed in KVM Compute node

```
[root@cmo1 ~]# rpm -qa | egrep
"qemu-kvm|hypervkvpd|qemu-guest-agent|qemu-kvm-tools|python-virtinst|virt-manager|
virt-viewer|virt-top|libvirt|libvirt-client|virt-who|fence-virtd-libvirt|fence-vir
td-multicast|fence-virtd-serial|libvirt-cim|libvirt-java|libvirt-snmp|perl-Sys-Vir
t|libguestfs|libguestfs-java|libguestfs-tools|virt-v2v|libvirt-python"
qemu-kvm-0.12.1.2-2.415.el6.x86_64
libguestfs-tools-1.20.11-2.el6.x86_64
virt-top-1.0.4-3.15.el6.x86_64
perl-Sys-Virt-0.10.2-5.el6.x86_64
libvirt-python-0.10.2-29.el6.x86_64
libguestfs-java-1.20.11-2.el6.x86_64
libvirt-cim-0.6.1-9.el6.x86_64
fence-virtd-serial-0.2.3-15.el6.x86_64
virt-viewer-0.5.6-8.el6.x86_64
libvirt-snmp-0.0.2-4.el6.x86_64
qemu-kvm-tools-0.12.1.2-2.415.el6.x86_64
fence-virtd-multicast-0.2.3-15.el6.x86_64
libvirt-client-0.10.2-29.el6.x86_64
libvirt-0.10.2-29.el6.x86_64
libguestfs-1.20.11-2.el6.x86_64
virt-v2v-0.9.1-4.el6.x86_64
virt-manager-0.9.0-19.el6.x86_64
fence-virtd-libvirt-0.2.3-15.el6.x86_64
libvirt-java-0.4.9-1.el6.noarch
hypervkvpd-0-0.12.el6.x86_64
python-libguestfs-1.20.11-2.el6.x86_64
python-virtinst-0.600.0-18.el6.noarch
virt-who-0.8-9.el6.noarch
libguestfs-tools-c-1.20.11-2.el6.x86_64
qemu-guest-agent-0.12.1.2-2.415.el6.x86_64
```

After you complete this step, run the following command to launch the virtual machine Manager:

```
# virt-manager
```

You should be able to see the virtual machine Manager, and it will be connected to the local host (QEMU), as shown in Figure 4-15.

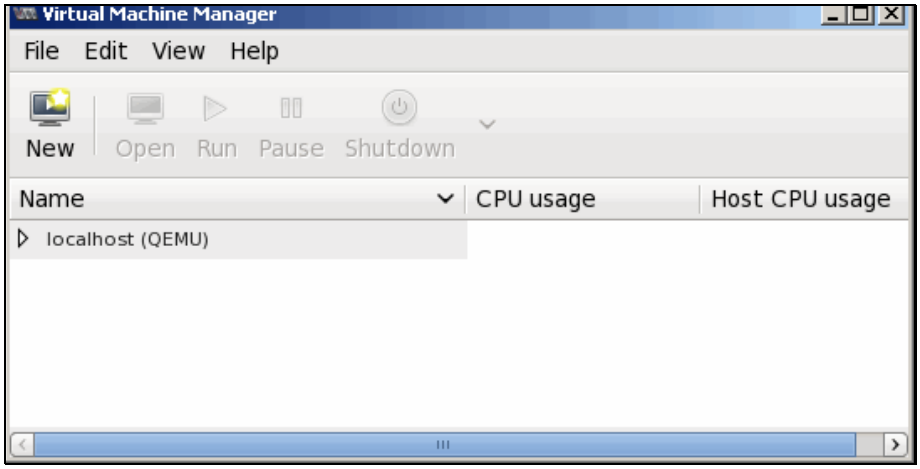


Figure 4-15 Virtual machine Manager on KVM Compute node

If this step is completed successfully, it means that the KVM Compute node is ready to join your OpenStack cloud.

4.6.8 Deploy the KVM Compute node using the deployment server

To add the KVM Compute node to your OpenStack cloud, you need to use the deployment server to complete this task. The deployment server will use the same input files (env.json and topo.json) to install packages in your Compute nodes, execute specific configuration steps and add them to the Controller node management.

Next, you will follow these steps to add the KVM Compute node to your OpenStack cloud:

1. Update the environment file (`env.json`).
2. Update the topology file (`topo.json`).
3. (Optional) Use node-specific attribute file.
4. Use the **`knife`** command to update your cloud topology.

Update the environment (env.json) file

Unless you have specific needs for the configuration of the KVM Compute nodes, all the customization required in the env.json file is already done. The default env.json file obtained through the example-ibm-os-single-controller-n-compute sample file in the deployment server in section 4.5.7, “Deploying the Controller node” on page 225 already consider using KVM/QEMU as Compute nodes.

In the following steps, we will check if the env.json file already meets the requirements for this configuration:

1. Log in to your deployment server and open the env.json file that contains the environment definitions for your cloud:

```
# cd /<path_to_your_cloud_directory_structure>
# vi env.json
```

2. Verify that the `openstack.compute.libvirt.virt_type` section is set to `kvm` or `qemu` as shown in Example 4-42.

Example 4-42 Verifying the libvirt.virt_type configuration in env.json

```
98        "compute": {
99          "debug": false,
100         "verbose": true,
101         "state_path": "/var/lib/nova",
102         "driver": "libvirt.LibvirtDriver",
103         "libvirt": {
104           "virt_type": "kvm"
105         }
106       },
```

2. Verify that the `openstack.network.openvswitch.bridge_mappings` and `openstack.network.openvswitch.bridge_mapping_interface` matches the name of your Compute node data network interface as shown in Example 4-43.

Example 4-43 Verifying the bridge_mappings configuration in the env.json file

```
107       "network": {
108         "verbose": false,
109         "debug": false,
110         "openvswitch": {
111           "bridge_mappings": "default:br-eth1",
112           "bridge_mapping_interface": "br-eth1:eth1"
113         }
114       },
```

If these configurations are correct, no modification is required in the `env.json` file.

Important: The configuration specified in this file is global and valid to all the nodes in your cloud that will use Open vSwitch bridges (Controller node, KVM Compute nodes and PowerKVM Compute nodes). With this configuration, the Chef server running in the deployment server will try to configure a bridge named `br-eth1` in each of those nodes, using eth1 as the physical interface to provide network access.

If your data network interface in the KVM Compute nodes is not eth1, but in the Controller node is eth1, do not change the env.json, otherwise it will affect the Controller node network bridge. In this case you need to create a node-specific attributes file.

The PowerKVM hosts do not use ethX as naming convention for network interfaces, so they will require a node-specific attribute file. Details are provided in the instructions to add the PowerKVM node to the OpenStack cloud.

In this publication, we use eth1 as data network interface.

For additional customization options in the environment file (`env.json`), see the Chapter 5 of the IBM Cloud Manager with OpenStack 4.1 Administrator Guide:

`http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_deploy_kvm.html`

Update the topology file (topo.json)

You need to modify your topology file to add the KVM Compute nodes to your cloud. After completing this step, you will use the **knife** command to tell the deployment server to update your cloud topology, adding the KVM Compute nodes to your OpenStack cloud.

The following steps guide you on how to modify the topology file:

1. Log in to your deployment server and open the `topo.json` file that contains your cloud topology:

```
# cd /<path_to_your_cloud_directory_structure>
# vi topo.json
```

2. Add your KVM Compute node entry in the “nodes” section of the file. Each Compute node should have an entry similar to Example 4-44.

Example 4-44 Sample KVM Compute node entry in the topo.json file

```
{
      "fqdn":"$KVM_Compute_node_FQDN",
      "password":"<root_password>",
      "quit_on_error":true,
      "run_order_number":1,
      "runlist": [
        "role[ibm-os-compute-node-kvm]"
      ]
}
```

Important: Every node entry should be placed between braces: { }

If there is another node after the node you are adding to the `topo.json` file, the braces (or curly brackets) should have also a comma after them: { },

If this is the last node, the braces are not followed by a comma: { }

3. In your `topo.json` file, the KVM Compute node should be added after the Controller node, so the closing braces from the Controller node entry must receive a comma, as shown in Example 4-45.

Example 4-45 Adding the KVM Compute node to the topo.json file

```
1 {
2   "name":"your_topology_name",
3   "description":"your_topology_description",
4   "environment":"your_environment_name",
5   "secret_file":"/path_to_your_secret_key_file",
6   "run_sequentially":false,
7   "nodes": [
8     {
9       "fqdn":"<controller_node_fqdn>",
10       "password":"<root_password>",
11       "quit_on_error":true,
12       "run_order_number":1,
13       "runlist": [
14         "role[ibm-os-single-controller-node]"
15       ]
16     },
17     {
```

```
18       "fqdn":"<kvm_compute_node_fqdn>",
19       "password":"<root_password>",
20       "quit_on_error":true,
21       "run_order_number":2,
22       "runlist": [
23         "role[ibm-os-compute-node-kvm]"
24       ]
25     }
26   ]
27 }
```

Note: Notice that there is a `node_order_number` entry. This will determine the order of your cloud deployment. The Controller node should always be the first and the Compute nodes should be run after the Controller node deployment. If you add more than one KVM Compute node at the same time in the `topo.json` file and they both have the same `run_order_number` the Chef server will run packages installation and configuration on both nodes in parallel, given that the `run_sequentially` is set to `false`.

In the cloud setup for this publication, we have two KVM Compute nodes (`cmo1.itso.ibm.com` and `cmo2.itso.ibm.com`) and we added both Compute nodes at the same time in the `topo.json` configuration for deployment. Example 4-46 shows the topo.json file used to add both KVM Compute nodes to the cloud prepared for this publication.

Example 4-46 topo.json file used to add both KVM Compute nodes to the residency cloud

```
1 {
2   "name":"topo-ITSO-SDE-CLOUD",
3   "description":"topo-ITSO-SDE-CLOUD",
4   "environment":"ITSO-SDE-CLOUD",
5   "secret_file":"/root/cmo/itso-sde-deployment/itso-secret-key",
6   "run_sequentially":false,
7   "nodes": [
8     {
9       "fqdn":"cmocontroller.itso.ibm.com",
10       "password":"itsoadmin",
11       "quit_on_error":true,
12       "run_order_number":1,
13       "runlist": [
14         "role[ibm-os-single-controller-node]"
15       ]
16     },
17     {
18       "fqdn":"cmo1.itso.ibm.com",
19       "password":"itsoadmin",
20       "quit_on_error":true,
21       "run_order_number":2,
22       "runlist": [
23         "role[ibm-os-compute-node-kvm]"
24       ]
25     },
26     {
27       "fqdn":"cmo2.itso.ibm.com",
28       "password":"itsoadmin",
29       "quit_on_error":true,
30       "run_order_number":2,
```

```
31       "runlist": [
32         "role[ibm-os-compute-node-kvm]"
33       ]
34     }
35   ]
36 }
```

If you do not have any specific requirement for the KVM Compute nodes, the topo.json file is ready to be used for deploying your OpenStack cloud, adding the Compute nodes to your existing cloud, so you can go to the section “Use the knife command to update your cloud topology”. If your KVM Compute nodes do not use the same interface name specified in the `openstack.network.openvswitch.bridge_mapping_interface` in the `env.json` file you need to configure a node-specific attribute file, see “Optional step: Use node-specific attribute file” for instructions on how to do it.

Optional step: Use node-specific attribute file

This step is only required if you need to use the deployment server to provide specific configuration to your KVM Compute nodes that do not apply to other nodes in the cloud. For example, if the data network interface from your Controller node is `eth1` and in your KVM Compute nodes is `eth2`, you cannot change the env.json file, or it would affect the bridge configuration from the Controller node, so you must create this node-specific attribute file to tell the Chef server running in the deployment server that specifically for this node, the bridge configuration will be different.

The following steps use this example of different data network interface name to show how to use the node-specific attribute file, but the same logic applies for any configuration that you need to provide specifically to a single node and does not apply to the entire cloud.

1. Log in to your deployment server and create a node-specific attribute file using a relevant name that allows you to identify that the file belongs to a certain node (recommended using the node name in the file with the `.json` extension):

   ```
   # cd /<path_to_your_cloud_directory_structure>
   # vi env-<kvm_compute_node_name>.json
   ```

2. Provide attributes that are specific to this node, preserving the same structure of braces from the `env.json` file. Example 4-47 shows how to provide the openstack.network.openvswitch.bridge_mappings and openstack.network.openvswitch.bridge_mapping_interface information, using the node-specific attribute file.

   Example 4-47 Node-specific attribute file used to provide attributes information specific to a node

   ```
   1 {
   2 "openstack": {
   3       "network": {
   4         "verbose": false,
   5         "debug": false,
   6         "openvswitch": {
   7           "bridge_mappings": "default:br-eth2",
   8           "bridge_mapping_interface": "br-eth2:eth2"
   9         }
   10       }
   11    }
   12 }
   ```

In Example 4-47 on page 249, rather than creating a bridge called `br-eth1` using `eth1` as the data network interface (as specified in the `env.json` file), the Chef server will create a bridge called `br-eth2` using `eth2` as the data network interface specifically in the node that uses this node-specific attribute file.

3. Modify your topology file (`topo.json`) to provide the node-specific attribute file information to the node that needs this specific configuration. Add an **"**`attribute_file`**"** entry with the full path to the node-specific attribute file, between the **"**`password`**"** and **"**`quit_on_error`**"** entries from this node as shown in Example 4-48.

Example 4-48 Providing a node-specific attribute file information in the topo.json file

```
{
  "name":"topo-ITSO-SDE-CLOUD",
  "description":"topo-ITSO-SDE-CLOUD",
  "environment":"ITSO-SDE-CLOUD",
  "secret_file":"/root/cmo/itso-sde-deployment/itso-secret-key",
  "run_sequentially":false,
  "nodes": [
    {
      "fqdn":"cmocontroller.itso.ibm.com",
       "password":"itsoadmin",
       "quit_on_error":true,
       "run_order_number":1,
       "runlist": [
         "role[ibm-os-single-controller-node]"
       ]
    },
    {
      "fqdn":"cmo1.itso.ibm.com",
      "password":"itsoadmin",
      "attribute_file":"/root/cmo/itso-sde-deployment/env-cmo1.json",
      "quit_on_error":true,
      "run_order_number":2,
      "runlist": [
        "role[ibm-os-compute-node-kvm]"
      ]
    },
    {
      "fqdn":"cmo2.itso.ibm.com",
      "password":"itsoadmin",
      "quit_on_error":true,
      "run_order_number":2,
      "runlist": [
        "role[ibm-os-compute-node-kvm]"
      ]
    }
  ]
}
```

The `/root/cmo/itso-sde-deployment/env-cmo1.json` file contains the information provided in Example 4-47 on page 249, the `topo.json` file contains the information from Example 4-48, and the `env.json` file is configured as shown in Example 4-43 on page 246. Therefore, when you use the **`knife`** command to add the KVM Compute nodes to your OpenStack cloud, the Chef server will create a bridge called `br-eth1` by using `eth1` as the data network interface in

the Controller node and Compute node `cmo2.itso.ibm.com`, and it will create a bridge called `br-eth2` by using `eth2` as data network interface in the Compute node `cmo1.itso.ibm.com`.

After completing this optional step, follow “Use the knife command to update your cloud topology” on page 251 to add the KVM Compute nodes to your cloud.

Use the `knife` command to update your cloud topology

Now you are ready to use the deployment server to add the KVM Compute nodes to your OpenStack cloud. In this step, the deployment server will effectively install and configure Cloud Manager components in the KVM Compute nodes that you added in the topo.json file and start all the necessary services. It will also add the Compute nodes information to the Controller node, allowing it to manage the Compute nodes components.

1. Log in to your deployment server and access the directory where you have the env.json, topo.json and optional node-specific attribute files:

   ```
   # cd /root/cmo/itso-sde-deployment/
   ```

2. Use the **`knife`** command to load the env.json file to the Chef server repository in case it was modified in “Update the environment (env.json) file” on page 245:

   ```
   # knife environment from file env.json
   Updated Environment ITSO-SDE-CLOUD
   ```

3. Use the **`knife`** command to update the topology of your OpenStack cloud with the modified topo.json file:

   ```
   # knife os manage update topology topo.json
   ```

For more information, see “Updating a deployed topology”:

http://ibm.co/1UCa1j4

The expected output is too long to fully show in this publication, but Example 4-49 shows partially the expected output from the **`knife os manage update topology topo.json`** command.

Example 4-49 Sample output updating the cloud topology by adding KVM Compute nodes

```
[root@cmodeploy itso-sde-deployment]# knife os manage update topology topo.json
Updating topology 'topo-ITSO-SDE-CLOUD' ...

Deploying to nodes with run_order_number '2'
Bootstrapping nodes with environment ITSO-SDE-CLOUD...
Bootstrapping node ...Bootstrapping node ...

Connecting to cmo2.itso.ibm.com
Connecting to cmo1.itso.ibm.com
cmo2.itso.ibm.com INFO: Adding certificate for Chef server:
cmodeploy.itso.ibm.com:14443
cmo2.itso.ibm.com depth=0 C = US, ST = WA, L = Seattle, O = YouCorp, OU =
Operations, CN = cmodeploy.itso.ibm.com, emailAddress = you@example.com
cmo2.itso.ibm.com verify error:num=18:self signed certificate
cmo2.itso.ibm.com verify return:1
cmo2.itso.ibm.com depth=0 C = US, ST = WA, L = Seattle, O = YouCorp, OU =
Operations, CN = cmodeploy.itso.ibm.com, emailAddress = you@example.com
cmo2.itso.ibm.com verify return:1
cmo1.itso.ibm.com INFO: Adding certificate for Chef server:
cmodeploy.itso.ibm.com:14443
cmo2.itso.ibm.com DONE
```

```
cmo1.itso.ibm.com depth=0 C = US, ST = WA, L = Seattle, O = YouCorp, OU =
Operations, CN = cmodeploy.itso.ibm.com, emailAddress = you@example.com
cmo1.itso.ibm.com verify error:num=18:self signed certificate
cmo1.itso.ibm.com verify return:1
cmo1.itso.ibm.com depth=0 C = US, ST = WA, L = Seattle, O = YouCorp, OU =
Operations, CN = cmodeploy.itso.ibm.com, emailAddress = you@example.com
cmo1.itso.ibm.com verify return:1
cmo1.itso.ibm.com DONE
.
. <many lines were removed>
.
All bootstrapped nodes with run_order_number '2' deployed.

Results for deploy of topology 'topo-ITSO-SDE-CLOUD'

Results for nodes with run_order_number '2'
Deploy of node at cmo1.itso.ibm.com was successful.
Deploy of node at cmo2.itso.ibm.com was successful.
Deploy of topology 'topo.json' completed in 900 seconds.
```

Note: You can also use the **`knife os manage deploy topology topo.json`** in this task, but it will run unnecessary commands in the Controller node, trying to redeploy what was already deployed in section 4.5.7, “Deploying the Controller node” on page 225, and take longer to complete.

Important: If you use a node-specific attribute file and for some reason you need to modify the configuration in this file after the topology is deployed, the **`knife os manage update topology topo.json`** command will not read the node-specific attribute file. Besides modifying the file to keep track of the changes, you should use the command **`knife node edit <node_name_fqdn>`**. For example:

```
# knife node edit cmo1.itso.ibm.com
```

This will bring the node-specific attribute file loaded in the Chef repository, so you can modify it directly in the Chef repository.

4. Log in to the Controller node and run the **`nova hypervisor-status-list`** command to verify if the Compute nodes were successfully added to your OpenStack cloud:

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# nova hypervisor-status-list
+----+-----------------------+-----------------+-----------+--------------------+
| id | hypervisor_hostname   | hypervisor_type | ha_status | maintenance_status |
+----+-----------------------+-----------------+-----------+--------------------+
| 1  | cmo1.itso.ibm.com     | QEMU            | ok        | ok                 |
| 3  | cmo2.itso.ibm.com     | QEMU            | ok        | ok                 |
+----+-----------------------+-----------------+-----------+--------------------+
```

Note: It is required to source the /root/openrc file before running the command, to provide the `OS_USERNAME` and `OS_PASSWORD` variables to the OpenStack commands.

5. Log in to the Cloud Management dashboard using the Controller node IP address and click **Admin > Hypervisors** as shown in Figure 4-16.

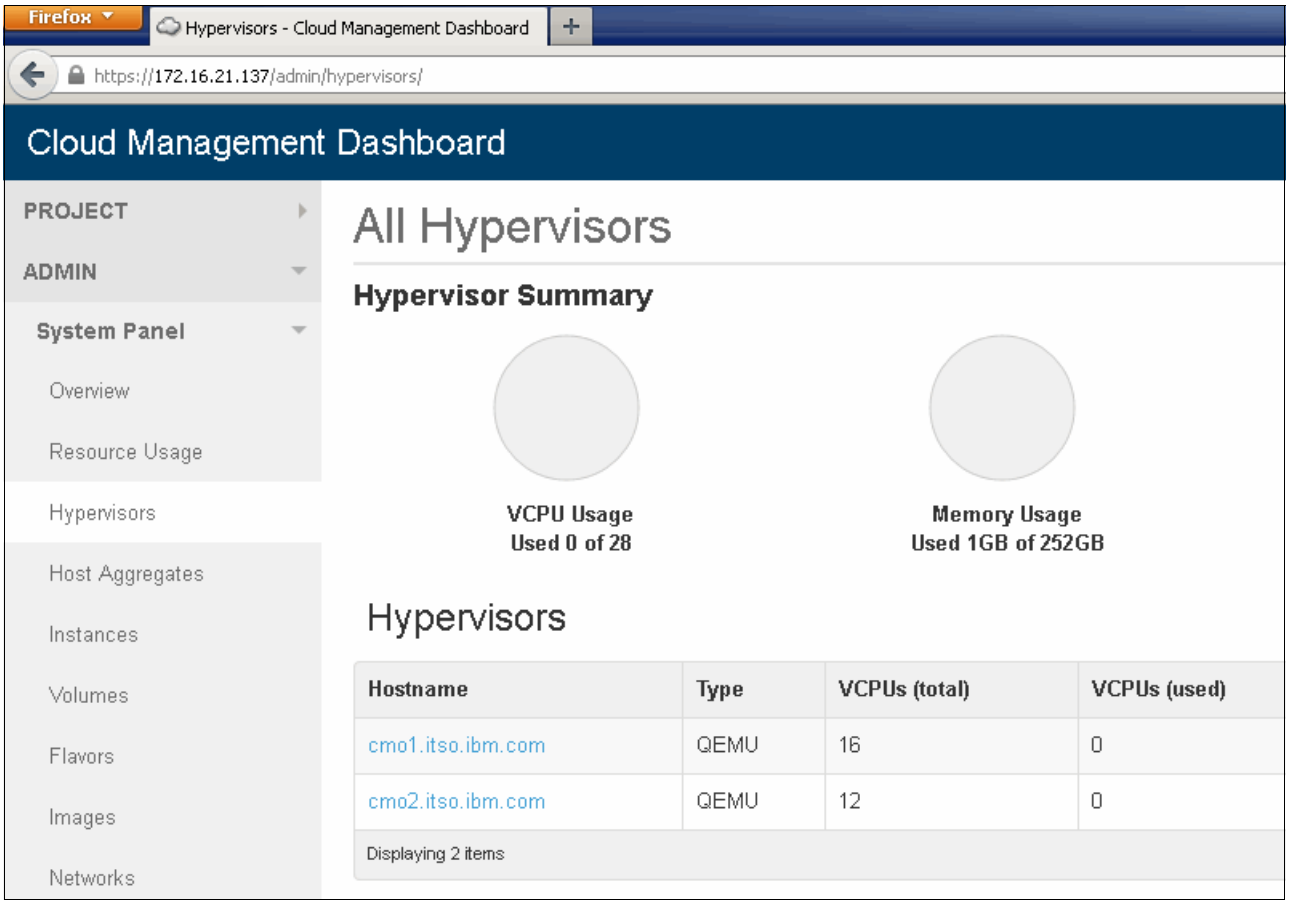


Figure 4-16 KVM Compute nodes in the Cloud Management dashboard

4.7 Add PowerKVM Compute nodes to the cloud

The process of adding PowerKVM Compute nodes (PowerKVM/QEMU hypervisors) to the OpenStack cloud involves preparing the physical host by installing and customizing PowerKVM 2.1 operating system on the POWER8 server and using the deployment server to add the Compute node to the OpenStack cloud. In this process, the Chef server will automatically install and configure required packages in the PowerKVM Compute node, including installing Open vSwitch, creating the network bridge required to provide external access to the instances (or virtual machines), installing and configuring OpenStack components such as nova, neutron and cinder, and adding the PowerKVM Compute node information to the Controller node. After the end of this process, the Compute node will be available for instances deployment in the Cloud Management dashboard.

PowerKVM is an excellent choice for Power Virtualization for Linux workloads, specially for system administrators not familiar with the Power Systems architecture. It allows them to benefit from the Power Systems features, using the same technology that is applied in the x86_64 KVM servers. It is compatible with Scale Out POWER8 Linux servers (for example IBM POWER8 servers S812L and S822L). The PowerKVM V2.1 operating system must be installed in the physical host and, at the time of writing this IBM Redbooks, it supports Red Hat, SUSE, and Ubuntu Linux guests.

For more information about PowerKVM, see *IBM PowerKVM Configuration and Use*, SG24-8231:

http://www.redbooks.ibm.com/redpieces/abstracts/sg248231.html

With Cloud Management with OpenStack, you can add PowerKVM hosts under its management, making even simpler and quick to deploy instances on your cloud using this virtualization technology.

Important: PowerKVM can also be managed by PowerVC, but you cannot add PowerKVM managed by PowerVC to the Cloud Manager. At the time of writing this IBM Redbooks, the PowerKVM must be directly managed by the Cloud Manager.

You can connect your PowerVC to the Cloud Manager to manage PowerVM hypervisors. In summary:

- Cloud Manager manages PowerKVM.
- Cloud Manager connects to PowerVC that manages PowerVM.

To add PowerKVM Compute nodes to the OpenStack cloud, the following steps need to be accomplished:

1. Verify that your hardware supports PowerKVM.
2. Enable PowerKVM in the Advanced System Management Interface (ASM).
3. Install PowerKVM.
4. Customize PowerKVM to meet Cloud Manager requirements.
5. Configure management and data network interfaces.
6. Deploy the PowerKVM Compute node using the deployment server.
7. (Optional) Modify the nova configuration file to enable the Kimchi console.

4.7.1 Prerequisites for PowerKVM Compute nodes

IBM Cloud Manager supports IBM PowerKVM Compute nodes. PowerKVM Compute nodes must be physical hosts (nested virtual machines are not supported). The PowerKVM must satisfy the following requirements:

- Operating system: IBM PowerKVM 2.1.0
- Hardware: POWER8 server with the PowerKVM hypervisor configured

4.7.2 Verify that your hardware supports PowerKVM

At the moment of writing this publication, only IBM PowerLinux Systems built on POWER8 processors are compatible with PowerKVM as the hypervisor. The following server models currently support PowerKVM:

- 8247-21L (IBM Power System S812L)
- 8247-22L (IBM Power System S822L)

If you have any of these servers, your hardware is compatible with PowerKVM.

Note: Optionally, PowerKVM GA (General Availability) offering is also orderable with IBM PowerKVM system preinstalled. If this is the case in your PowerKVM environment, you can go straight to 4.7.5, “Customize PowerKVM to meet the Cloud Manager requirements” on page 255.

4.7.3 Enable PowerKVM in Advanced System Management Interface (ASMI)

In general, the Power System S812L and 822L servers are already configured to use PowerKVM as the hypervisor mode. Use the following steps to connect to the ASMI interface and check to be sure that this configuration is correct.

1. Connect to the ASMI web interface using a workstation, and provide the `admin` user ID and password (if this is the first time that you are connecting to the ASMI web interface, the default `admin` password is `admin`).
2. From the Main Menu, select **System Configuration > Hypervisor Configuration**.
3. Select **PowerKVM (or OPAL)** as your hypervisor mode, set up a password for the IPMI session, and click **Continue**.
4. If you are setting up the ASMI interface for the first time, you need to configure network access, so you can use IPMI console to install the PowerKVM server. From the Main Menu, select **Services > Network Configuration** and follow the Configuration wizard to complete this step.

For more information about configuring the ASMI to install the PowerKVM hypervisor, see the Quick Start Guide for configuring IBM PowerKVM on Power Systems servers:

`http://www.ibm.com/support/knowledgecenter/linuxonibm/liabq/kvmquickstart_guide.pdf`

4.7.4 Install IBM PowerKVM

Obtain the IBM PowerKVM 2.1 installation media (available in ISO format), burn the DVD, and install in the physical host that will be the PowerKVM Compute node in your OpenStack cloud. For detailed instructions on how to install PowerKVM, see the IBM Redbooks publication titled *IBM PowerKVM Configuration and Use*, SG24-8231:

`http://www.redbooks.ibm.com/redpieces/abstracts/sg248231.html`

Important: During the installation process, you are prompted to set up the network configuration. *Do not* create any network bridge. The deployment server will automatically install the Open vSwitch software and set up the network bridge during the deployment process.

Note: Unlike the requirements for the Controller node and KVM Compute nodes, for the PowerKVM you do not need to set up a Yum repository with the PowerKVM installation media. All of the packages available in the installation media are already installed in your PowerKVM host when you complete the installation process. During the deployment process, the deployment server will set up Yum repositories in the PowerKVM Compute node to install OpenStack components.

4.7.5 Customize PowerKVM to meet the Cloud Manager requirements

Some requirements must be attended to add the PowerKVM Compute node to your OpenStack cloud. The following topics will guide you to complete these requirements.

Use a fully qualified domain name

It is required to use a fully qualified domain name (FQDN) on all your nodes. For example, an FQDN is *powerkvm1.itso.ibm.com*, not *powerkvm1*.

Make sure the output of the **hostname** command shows the FQDN. Example 4-50 shows the output of the **hostname** command with a fully qualified domain name.

Example 4-50 hostname output from PowerKVM Compute node with FQDN

```
[root@powerkvm1 ~]# hostname
powerkvm1.itso.ibm.com
```

In case your PowerKVM server is not configured to use FQDN, you can change this configuration using the **configure-system** tool:

1. Run the following command to open the **configure-system** tool:

   ```
   # configure-system
   ```

2. Use the keyboard arrows to move to **DNS Configuration** and press Enter (see Figure 4-17).

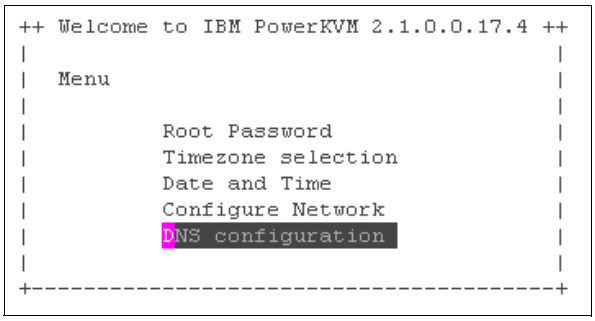


Figure 4-17 PowerKVM hostname configuration

3. Provide the fully qualified domain name in the `hostname` field (you can also use this tool for DNS configuration). Use the Tab key to move to **OK**, and press Enter, as shown in Figure 4-18.

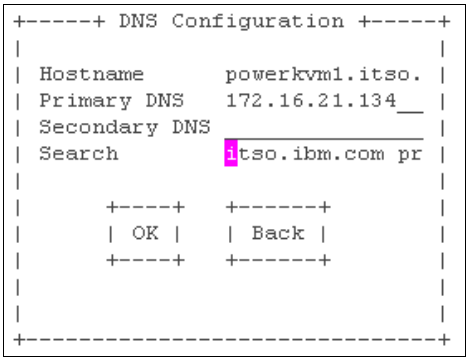


Figure 4-18 PowerKVM FQDN configuration

4. Use the keyboard arrows to reach the **Exit** option, and press Enter, as shown in Figure 4-19.

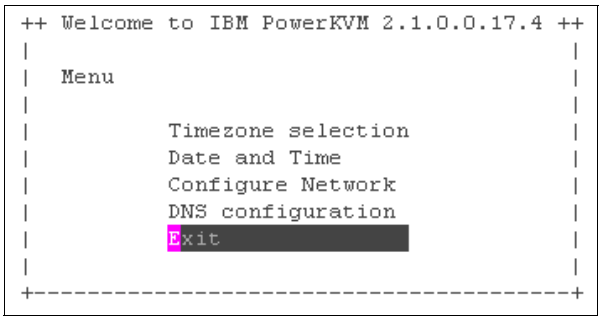


Figure 4-19 Exit the PowerKVM hostname configuration

5. Verify that the `/etc/hostname` file is configured with the fully qualified domain name, as shown in Example 4-51. If it is incorrect, use the **`vi`** command to modify this file, providing the FQDN.

Example 4-51 Verifying hostname configuration in the PowerKVM host

```
[root@powerkvm1 ~]# cat /etc/hostname
powerkvm1.itso.ibm.com
```

6. Configure host name resolution

Host name resolution must be configured across all of the hosts that will participate in the cloud (only physical hosts, not virtual machines), including the PowerKVM Compute nodes. This name resolution can be accomplished by using the `/etc/hosts` file, but DNS is recommended, especially if you have an enterprise cloud. If you plan to use `/etc/hosts`, make sure you configure it with FQDN, as shown in Example 4-52.

Example 4-52 Sample /etc/hosts from PowerKVM node

```
127.0.0.1     localhost localhost.localdomain localhost4
localhost4.localdomain4
::1           localhost localhost.localdomain localhost6
localhost6.localdomain6
172.16.21.136 cmodeploy.itso.ibm.com     cmodeploy
172.16.21.137 cmocontroller.itso.ibm.com cmocontroller
172.16.21.131 cmo1.itso.ibm.com          cmo1
172.16.21.132 cmo2.itso.ibm.com        cmo2
172.16.21.143 powerkvm1.itso.ibm.com     powerkvm1
172.16.21.144 powerkvm2.itso.ibm.com     powerkvm2
172.16.21.135 powervc.itso.ibm.com       powervc
```

If you plan to use this server as a DNS client, you can use the **`configure-system`** tool to accomplish this configuration:

```
# configure-system
```

a. In the main menu, use the arrows in your keyboard to move to **DNS Configuration** and press Enter, as shown in Figure 4-17 on page 256.

b. Provide the `Primary DNS`, `Secondary DNS` (if available), and `Search` order (if available), use the Tab key to reach **OK**, and press Enter, as shown in Figure 4-18 on page 256.

c. Use the arrows in your keyboard to move to the **Exit** option and press Enter, as shown in Figure 4-19.

To make sure your PowerKVM Compute node hostname is resolvable by running the command:

```
# hostname -f
```

If the hostname is resolvable, it will return an output similar to Example 4-53.

Example 4-53 Example of hostname -f output in PowerKVM Compute node

```
[root@powerkvm1 ~]# hostname -f
powerkvm1.itso.ibm.com
```

Synchronize the date and time

Date and time must be synchronized as close as possible on all physical hosts in the cloud (a few seconds of difference is acceptable). You can use an NTP server to accomplish this requirement or simply configure your servers to match the date and time using regular configuration tools.

Configure your system as NTP client

The following steps help to configure your NTP client:

1. If you plan to configure your system as an NTP client, modify the following configuration:

   ```
   # vi /etc/ntp.conf
   ```

2. Insert one or more NTP servers with the following format:

   server <hostname_of_ntp_server> iburst

> **Note:** The `iburst` option is used to speed up the initial synchronization with the NTP server.

Example 4-54 shows a sample `/etc/ntp.conf` configuration file in a PowerKVM host. In this case, it is using four public NTP servers.

Example 4-54 Sample /etc/ntp.conf configuration file in PowerKVM host

```
[root@powerkvm1 ~]# cat /etc/ntp.conf
# For more information about this file, see the man pages
# ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5), ntp_mon(5).

driftfile /var/lib/ntp/drift

# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery
restrict -6 default kod nomodify notrap nopeer noquery

# Permit all access over the loopback interface.  This could
# be tightened as well, but to do so would affect some of
# the administrative functions.
restrict 127.0.0.1
restrict -6 ::1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
```

```
server 0.pool.ntp.org iburst
server 1.pool.ntp.org iburst
server 2.pool.ntp.org iburst
server 3.pool.ntp.org iburst

#broadcast 192.168.1.255 autokey        # broadcast server
#broadcastclient                        # broadcast client
#broadcast 224.0.1.1 autokey            # multicast server
#multicastclient 224.0.1.1              # multicast client
#manycastserver 239.255.254.254         # manycast server
#manycastclient 239.255.254.254 autokey # manycast client

# Enable public key cryptography.
#crypto

includefile /etc/ntp/crypto/pw

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8

# Enable writing of statistics records.
#statistics clockstats cryptostats loopstats peerstats
```

3. After configuring this file, run the following commands to start the NTP daemons:

```
# systemctl start ntpd.service
# systemctl start ntpdate.service
```

4. Run the following commands to enable these services during reboot:

```
# systemctl enable ntpd.service
# systemctl enable ntpdate.service
```

After completing this step, use the **date** command to make sure that the date and time are correct on all of your nodes.

Configure the date and time manually without an NTP server

The following steps help to configure the date and time manually without an NTP server:

1. If you do not want to configure your PowerKVM Compute nodes as NTP clients, run the following command to start the date and time configuration:

```
# configure-system
```

2. Use the arrows in your keyboard to move to **Timezone selection**, as shown in Figure 4-20 on page 260.

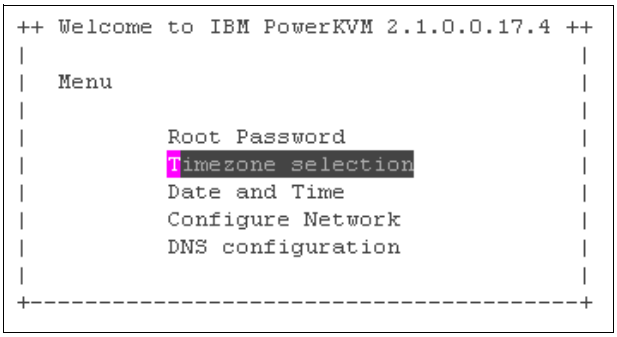


Figure 4-20 Timezone selection in configure-system tool

3. Select the correct `Timezone`, use the **<tab>** key to move to the next option, select `System clock uses UTC` if appropriate and press **<enter>** on the OK button as shown in Figure 4-21.

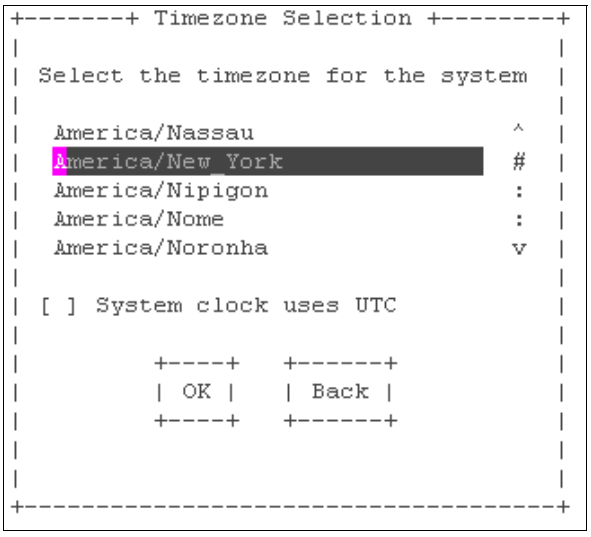


Figure 4-21 Timezone selection in PowerKVM system

4. Now select `Date and Time` as shown in Figure 4-22.

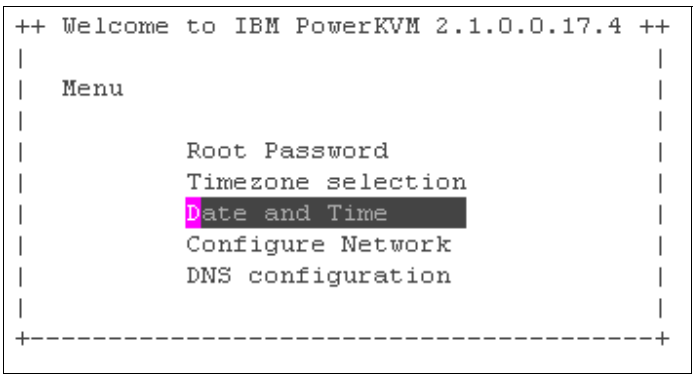


Figure 4-22 Date and time configuration in PowerKVM system

5. Provide the correct date and time information and press **<enter>** on the `OK` button as shown in Figure 4-23 on page 261.

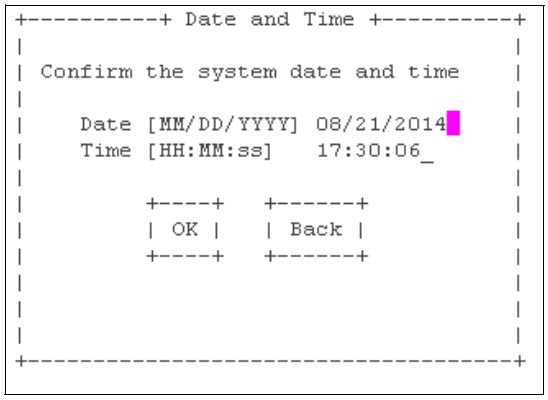


Figure 4-23 Providing date and time values in PowerKVM

6. Now use the keyboard arrows and conclude this configuration by pressing **<enter>** on the `Exit` button.

> **Note:** Make sure to provide the time and date as close as possible to all nodes in the cloud. Only a few seconds of difference is acceptable.

Define the same root password on the nodes

Consider applying the same recommendations from “Define the same root password on the nodes” on page 206 in your PowerKVM Compute node.

4.7.6 Configuring management and data network interfaces

The network configuration of the PowerKVM Compute node consists of two steps:

- Configuring the management network interface.
- Configuring the data network interface.

Configure the management network interface

The management network interface is used exclusively for management purposes, including communication between deployment server, Controller node and Compute nodes. It must have a dedicated network adapter without any bridge configured on top of it.

> **Important:** The naming convention for network interfaces in PowerKVM is based on the physical slot number where the adapter is placed within the hardware. For example, a valid interface name would be `enP3p1s0f0`. For this reason, the name of the management network interface in the PowerKVM node will not match the name provided in the environment (env.json) file on “Prepare the environment file, env.json” on page 225, so it will require a node-specific attribute file, which will be described on section 4.7.7, “Deploy the PowerKVM Compute node using the deployment server” on page 266.

1. Run the following command to configure the management network:

```
# configure-system
```

2. Use the arrow keys in your keyboard to move to `Configure Network` and hit **<enter>** as shown in Figure 4-24.

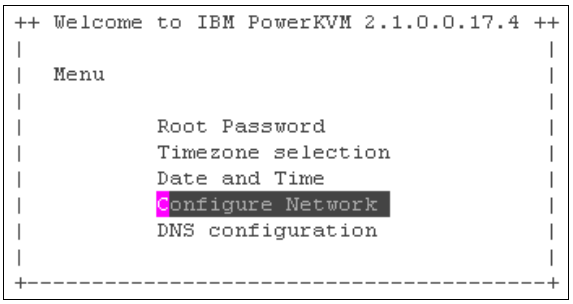


Figure 4-24 Network configuration on PowerKVM node

3. Select your network interface dedicated for management purposes, use the **<tab>** key to move to the `OK` button and hit **<enter>** as shown in Figure 4-25.

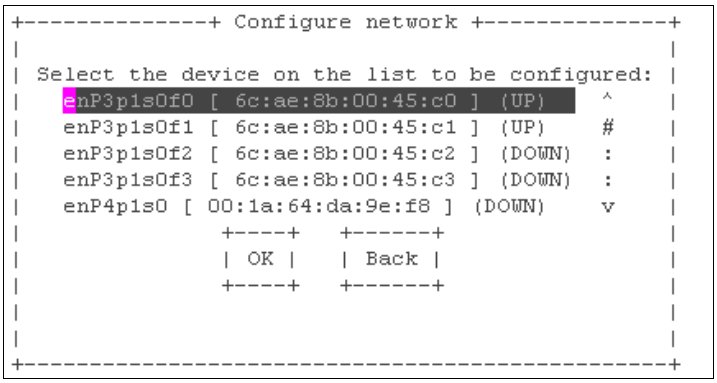


Figure 4-25 Management network interface selection in PowerKVM

Note: In this publication, we used the first available interface for management network (`enP3p1s0f0`).

4. Provide the IP address, network mask and default gateway. Check the `Enable` option and *do not check* the Create Network Bridge option. Use the Tab key to move to **Save**, and then press **Enter,** as shown in Figure 4-26.

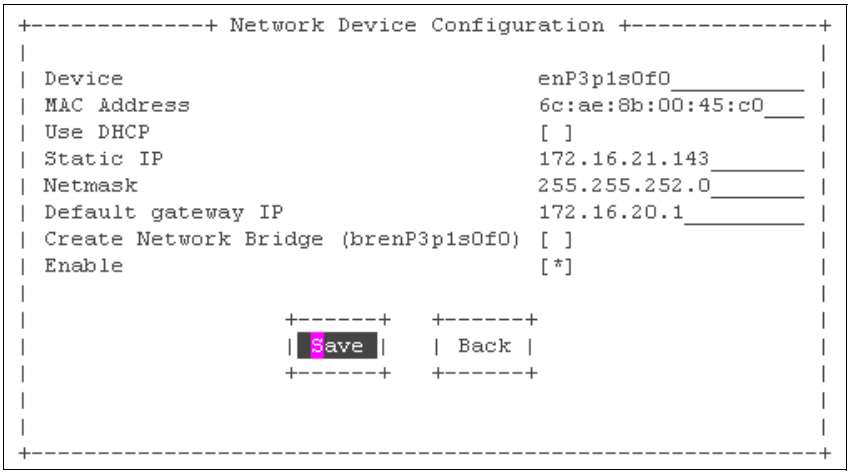


Figure 4-26 Management network interface configuration in PowerKVM

5. Use the `Exit` option to conclude this operation.

After completing this operation, the **`ifconfig`** command should show the interface configured as shown in Example 4-55.

Example 4-55 ifconfig output for PowerKVM management network interface

```
[root@powerkvm1 ~]# ifconfig enP3p1s0f0
enP3p1s0f0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.21.143  netmask 255.255.252.0  broadcast 172.16.23.255
        inet6 fe80::6eae:8bff:fe00:45c0  prefixlen 64  scopeid 0x20<link>
        ether 6c:ae:8b:00:45:c0  txqueuelen 1000  (Ethernet)
        RX packets 15811353  bytes 1834551453 (1.7 GiB)
        RX errors 0  dropped 108  overruns 0  frame 0
        TX packets 2980  bytes 844510 (824.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 22
```

The network configuration file `/etc/sysconfig/network-scripts/ifcfg-enP3p1s0f0` should be similar to Example 4-56.

Example 4-56 Management network configuration file on PowerKVM Compute node

```
[root@powerkvm1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-enP3p1s0f0
# Generated by IBM powerkvm installer
IPADDR=172.16.21.143
ONBOOT=yes
DEVICE=enP3p1s0f0
NETMASK=255.255.252.0
BOOTPROTO=none
HWADDR=6c:ae:8b:00:45:c0
TYPE=Ethernet
GATEWAY=172.16.20.1
```

Make sure that your management network is working by using the **ping** command to communicate with the deployment server and Controller node.

Now that the Management interface is configured, you can open the web browser and point to the following address to access the Kimchi interface and make sure the PowerKVM is working:

```
https://<your_powerkvm_ip_address>:8001
```

Provide the root user ID and password as shown in Figure 4-27 and you will be able to navigate to the Kimchi web interface.

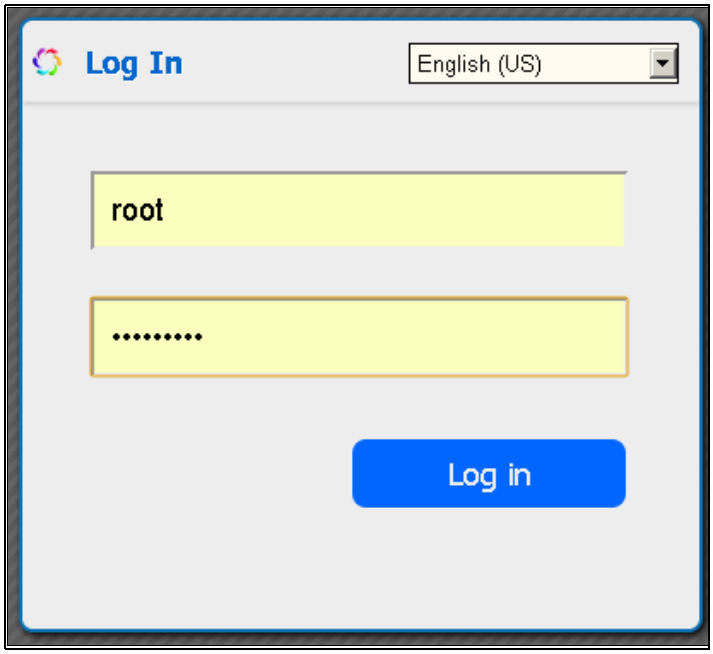


Figure 4-27 Kimchi login

Configure the data network interface

The data network interface is used to create a bridge to provide external access to the instances (virtual machines) running in this Compute node. This bridge *should not* be manually created by modifying the `/etc/sysconfig/network-scripts/ifcfg-enX` file or by using the **configure-system** tool. The deployment server will automatically install the Open vSwitch in this Compute node and create the network bridge during the step 4.7.7, "Deploy the PowerKVM Compute node using the deployment server" on page 266.

The data network requires a dedicated network interface. It cannot have any bridge configuration and cannot have any IP address, but the adapter must be up. The bridge will be created by the deployment server using the configuration specified in the node-specific attribute file, described in section 4.7.7, "Deploy the PowerKVM Compute node using the deployment server" on page 266.

1. Run the following command to configure the data network:

   ```
   # configure-system
   ```

2. Use the arrow keys in your keyboard to move to **Configure Network** and press Enter, as shown in Figure 4-24 on page 262.

3. Select your network interface dedicated for data network, press the Tab key to move to **OK**, and then press **Enter**, as shown in Figure 4-28.

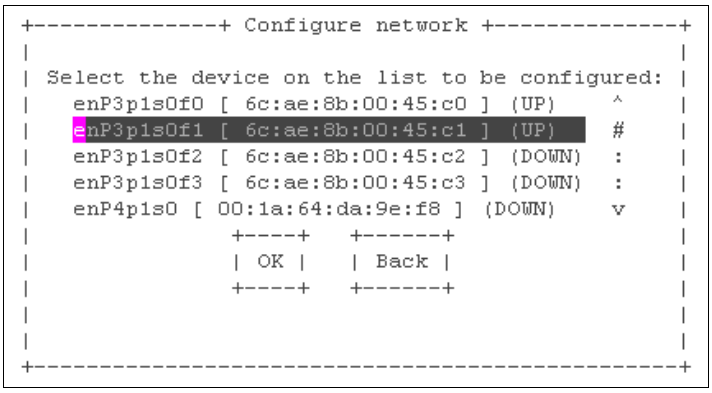


Figure 4-28 PowerKVM data network interface selection

4. Clear the Use DHCP box, *do not provide* any Static IP, Netmask or Default gateway IP information, *do not* check the Create Network Bridge box. Check only the **Enabl**e box, and then use the Tab key to reach **Save**, and then press **Enter**, as shown in Figure 4-29.

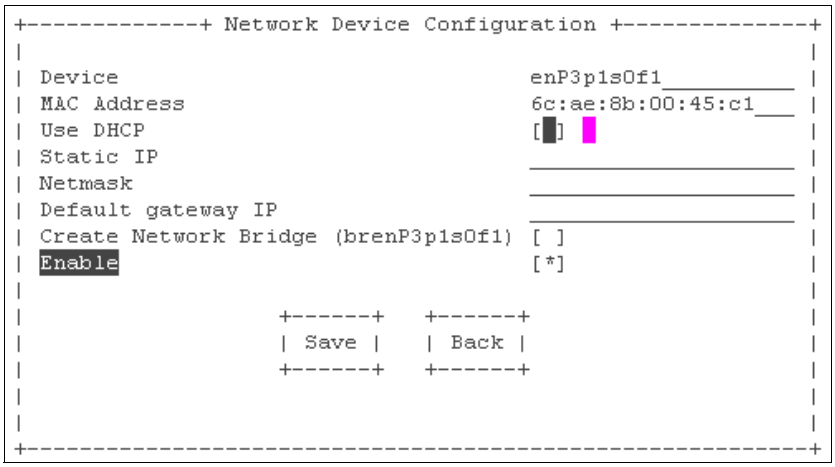


Figure 4-29 Data network interface configuration in PowerKVM

5. Conclude the process by using the `Exit` option.

After concluding this step, the configuration file for the data network interface /etc/sysconfig/network-scripts/ifcfg-enX file should be similar to the Example 4-57. Verify if the **`BOOTPROTO`** is set to `none` and the **`ONBOOT`** is set to `yes`. If they are not correct, modify it accordingly.

Example 4-57 Data network configuration file in PowerKVM Compute node

```
[root@powerkvm1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-enP3p1s0f1
# Generated by IBM powerkvm installer
DEVICE=enP3p1s0f1
BOOTPROTO=none
HWADDR=6c:ae:8b:00:45:c1
TYPE=Ethernet
ONBOOT=yes
```

The **ifconfig** command should show no IP configured, but the **mii-tool** command should show that the link is up, as shown in Example 4-58.

Example 4-58 ifconfig and mii-tool output of the data network interface in PowerKVM node

```
[root@powerkvm1 ~]# ifconfig enP3p1s0f1
enP3p1s0f1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::6eae:8bff:fe00:45c1  prefixlen 64  scopeid 0x20<link>
        ether 6c:ae:8b:00:45:c1  txqueuelen 1000  (Ethernet)
        RX packets 5420418  bytes 593839614 (566.3 MiB)
        RX errors 0  dropped 47802  overruns 0  frame 10
        TX packets 713  bytes 88347 (86.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 23

[root@powerkvm1 ~]# mii-tool enP3p1s0f1
enP3p1s0f1: negotiated 1000baseT-FD flow-control, link ok
```

4.7.7 Deploy the PowerKVM Compute node using the deployment server

To add PowerKVM Compute nodes to your OpenStack cloud, you need to use the deployment server to complete this task. The deployment server will use the same input files (env.json and topo.json) to install packages in your Compute nodes, execute specific configuration steps and add them to the Controller node management.

The following steps will add the PowerKVM Compute node to your OpenStack cloud:

1. Update the environment file (env.json).
2. Update the topology file (topo.json).
3. Create a node-specific attribute file.
4. Use the knife command to update your cloud topology.

Update the environment (env.json) file

Unless you have specific needs for the configuration of the PowerKVM Compute nodes, all the customization required in the env.json file is already done. The configuration executed in section 4.5.7, “Deploying the Controller node” on page 225 covers all required fields in the file to support a PowerKVM Compute node.

Note: You may notice that the information provided in the `openstack.endpoints.network-openvswitch.bind_interface`, `openstack.endpoints.compute-vnc-bind.bind_interface`, `openstack.network.openvswitch.bridge_mappings` and `openstack.network.openvswitch.bridge_mapping_interface` fields in the env.json file does not match the name of the management and data network interfaces in the PowerKVM Compute nodes. This will be addressed in the section “Create a node-specific attribute file”.

For more information about specific customization available in the env.json file for PowerKVM nodes, see the IBM Cloud Manager with OpenStack 4.1 Administrator Guide available at:

`http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html`

Update the topology (topo.json) file

You need to modify your topology file to add the PowerKVM Compute nodes to your cloud. You also need to set up node-specific attribute files to the PowerKVM nodes to address the naming convention for network interfaces in PowerKVM. After completing both steps you will use the **knife** command to tell the deployment server to update your cloud topology, adding the PowerKVM Compute nodes to your OpenStack cloud.

The following steps guide you on how to modify the topology file to achieve this objective:

1. Log in to your deployment server and open the topo.json file, that contains your cloud topology:

```
# cd /<path_to_your_cloud_directory_structure>
# vi topo.json
```

2. Add your PowerKVM Compute node entry in the “nodes” section of the file. Each Compute node should have an entry similar to Example 4-59.

Example 4-59 Sample PowerKVM Compute node entry in the topo.json file

```
{
      "fqdn":"$PowerKVM_Compute_node_FQDN",
      "password":"<root_password>",
      "attribute_file":"/path/to/node-specific-attribute-file.json",
      "quit_on_error":true,
      "run_order_number":2,
      "runlist": [
        "role[ibm-os-compute-node-powerkvm]"
      ]
}
```

Important: Every node entry should be placed between braces: `{ }`

If there is another node after the node you are adding to the topo.json file, the braces (or curly brackets) should have also a comma: `{ },`

If this is the last node, tthe braces are not followed by a comma: `{ }.`

Note: The node-specific attribute file specified in the `attribute_file` field will be created in the section “Create a node-specific attribute file”, but in this field you should place the full path to the file that you plan to create for the node you are adding.

3. In your topo.json file, the PowerKVM Compute node should be added after the Controller node or another Compute node (for example a KVM Compute node), so the closing braces from the previous node entry must receive a comma, as shown in Example 4-60.

Example 4-60 Adding the PowerKVM Compute node to the topo.json file

```
1 {
2   "name":"your_topology_name",
3   "description":"your_topology_description",
4   "environment":"your_environment_name",
5   "secret_file":"/path_to_your_secret_key_file",
6   "run_sequentially":false,
7   "nodes": [
8     {
```

```
9       "fqdn":"<controller_node_fqdn>",
10       "password":"<root_password>",
11       "quit_on_error":true,
12       "run_order_number":1,
13       "runlist": [
14         "role[ibm-os-single-controller-node]"
15       ]
16     },
17     {
18       "fqdn":"<kvm_compute_node_fqdn>",
19       "password":"<root_password>",
20       "quit_on_error":true,
21       "run_order_number":2,
22       "runlist": [
23         "role[ibm-os-compute-node-kvm]"
24       ]
25     },
26     {
27       "fqdn":"<powerkvm_compute_node_fqdn>",
28       "password":"<root_password>",
29       "attribute_file":"/path/to/node-specific-attribute-file.json",
30       "quit_on_error":true,
31       "run_order_number":2,
32       "runlist": [
33         "role[ibm-os-compute-node-powerkvm]"
34       ]
35     }
36   ]
37 }
```

Note: Notice that there is a `node_order_number` entry. This will determine the order of your cloud deployment. The Controller node should always be the first and the Compute nodes should be run after the Controller node deployment. If you add more than one PowerKVM Compute node at the same time in the topo.json file and they both have the same `run_order_number` the Chef server will run packages installation and configuration on both in parallel, given that the `run_sequentially` is set to `false`.

In the cloud setup for this publication, we have two PowerKVM Compute nodes (`powerkvm1.itso.ibm.com` and `powerkvm2.itso.ibm.com`). We added the `powerkvm1.itso.ibm.com` Compute node initially and then added the `powerkvm2.itso.ibm.com` Compute node in our cloud. Example 4-61 shows the `topo.json` file used to add both PowerKVM Compute nodes to the cloud prepared for this publication. Notice that they were added after both x86_64 KVM Compute nodes were already in the cloud.

Example 4-61 topo.json file used to add both PowerKVM Compute nodes to the Residency cloud

```
1 {
2   "name":"topo-ITSO-SDE-CLOUD",
3   "description":"topo-ITSO-SDE-CLOUD",
4   "environment":"ITSO-SDE-CLOUD",
5   "secret_file":"/root/cmo/itso-sde-deployment/itso-secret-key",
6   "run_sequentially":false,
7   "nodes": [
8     {
9       "fqdn":"cmocontroller.itso.ibm.com",
```

```
10        "password":"itsoadmin",
11        "quit_on_error":true,
12        "run_order_number":1,
13        "runlist": [
14          "role[ibm-os-single-controller-node]"
15        ]
16      },
17      {
18        "fqdn":"cmo1.itso.ibm.com",
19        "password":"itsoadmin",
20        "quit_on_error":true,
21        "run_order_number":2,
22        "runlist": [
23          "role[ibm-os-compute-node-kvm]"
24        ]
25      },
26      {
27        "fqdn":"cmo2.itso.ibm.com",
28        "password":"itsoadmin",
29        "quit_on_error":true,
30        "run_order_number":2,
31        "runlist": [
32          "role[ibm-os-compute-node-kvm]"
33        ]
34      },
35      {
36        "fqdn":"powerkvm1.itso.ibm.com",
37        "password":"itsoadmin",
38        "attribute_file":"/root/cmo/itso-sde-deployment/env-powekvm1.json",
39        "quit_on_error":true,
40        "run_order_number":2,
41        "runlist": [
42          "role[ibm-os-compute-node-powerkvm]"
43        ]
44      },
45      {
46        "fqdn":"powerkvm2.itso.ibm.com",
47        "password":"itsoadmin",
48        "attribute_file":"/root/cmo/itso-sde-deployment/env-powekvm2.json",
49        "quit_on_error":true,
50        "run_order_number":2,
51        "runlist": [
52          "role[ibm-os-compute-node-powerkvm]"
53        ]
54      }
55    ]
56  }
```

Create a node-specific attribute file

The node-specific attribute file is optional for other hypervisors, such as the x86_64 KVM, but is required in the PowerKVM Compute nodes. This file is used to provide information to the Chef server that will only apply to a specific node, and not to the whole OpenStack cloud. Given that the PowerKVM uses a different naming convention for the network interfaces, the configuration provided in the env.json file for the management and data networks do not apply to PowerKVM Compute nodes. Therefore, there is a requirement to provide a node-specific attribute file to inform to the Chef server the name of the management and data network interfaces.

Now, create the node-specific attribute file for your PowerKVM Compute node.

1. Log in to your deployment server and create a node-specific attribute file using a relevant name that allows you to identify that the file belongs to a certain node (recommended using the node name in the file with the `.json` extension):

   ```
   # cd /<path_to_your_cloud_directory_structure>
   # vi env-<powerkvm_compute_node_name>.json
   ```

   Important: The name and full path to the node-specific attribute file must match the value specified in the **"**`attribute_file`**"** field in the `topo.json` file on "Update the topology (topo.json) file" on page 267 specifically on step 2 on page 267.

2. Provide attributes that are specific to this node, preserving the same brackets / braces structure from the `env.json` file. Example 4-62 shows how to provide the management network interface name to the `openstack.endpoints.network-openvswitch.bind_interface` and `openstack.endpoints.compute-vnc-bind.bind_interface`, provide the bridge name to the field `openstack.network.openvswitch.bridge_mappings` and provide the data network interface name and bridge name to the field `openstack.network.openvswitch.bridge_mapping_interface` using the node-specific attribute file. These are the required attributes to be provided in this file for PowerKVM Compute nodes due to the network interface naming convention.

   Example 4-62 Node-specific attribute file used to provide network interface names in PowerKVM

   ```
   {
   "openstack": {
         "endpoints": {
           "network-openvswitch": {
             "bind_interface": "enP3p1s0f0"
           },
           "compute-vnc-bind": {
             "bind_interface": "enP3p1s0f0"
           }
         },
         "network": {
           "verbose": false,
           "debug": false,
           "openvswitch": {
             "bridge_mappings": "default:br-ext",
             "bridge_mapping_interface": "br-ext:enP3p1s0f1"
           }
         }
      }
   }
   ```

Example 4-62 on page 270, rather than creating a bridge called `br-eth1` using `eth1` as the data network interface (as specified in the `env.json` file) and using `eth0` for the Management interface (as specified in the `env.json` file), the Chef server creates a bridge called `br-ext` (as in external) using `enP3p1s0f1` as the data network interface and will use `enP3p1s0f0` as the management network interface specifically in the node that uses this node-specific attribute file.

Important: Notice that in this example, we use `br-ext` as the bridge name rather than using the `br-<interface_name>` convention used in the Controller node and KVM Compute nodes. This is because of a limitation in the number of characters for the bridge interface name accepted by neutron (OpenStack network component). The bridge name cannot be bigger than 11 characters, and `br-enP3p1s0f1` would have been 13 characters. This would prevent the deployment of instances in the PowerKVM Compute nodes.

The following explanation is extracted from the `/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini` from the PowerKVM Compute node:

```
# (ListOpt) Comma-separated list of <physical_network>:<bridge> tuples
# mapping physical network names to the agent's node-specific OVS
# bridge names to be used for flat and VLAN networks. The length of
# bridge names should be no more than 11. Each bridge must
# exist, and should have a physical network interface configured as a
# port. All physical networks listed in network_vlan_ranges on the
# server should have mappings to appropriate bridges on each agent.
```

Use the knife command to update your cloud topology

Now you are ready to use the deployment server to add the PowerKVM Compute nodes to your OpenStack cloud. In this step, the deployment server will effectively install and configure Cloud Manager components in the PowerKVM Compute nodes that you added in the topo.json file and start all the necessary services. It will also add the Compute node information to the Controller node, allowing it to manage the Compute nodes components.

1. Log in to your deployment server and access the directory where you have the env.json, topo.json and node-specific attribute files for the PowerKVM Compute nodes:

   ```
   # cd /root/cmo/itso-sde-deployment/
   ```

2. Use the **`knife`** command to load the `env.json` file to the Chef server repository in case it modified in “Update the environment (env.json) file” on page 266:

   ```
   # knife environment from file env.json
   Updated Environment ITSO-SDE-CLOUD
   ```

3. Use the **`knife`** command to update the topology of your OpenStack cloud with the modified `topo.json` file:

   ```
   # knife os manage update topology topo.json
   ```

Important: Before running the **`knife`** command, make sure that there is no unresponsive Yum repository in the PowerKVM Compute node. Run the **`yum list`** command in PowerKVM to make sure it completes successfully. Although it is not required any additional Yum repository in the PowerKVM Compute node to deploy it in your OpenStack cloud, if there is any unresponsive Yum repository, it might prevent the deployment.

For more information, see “Updating a deployed topology” in IBM Knowledge Center:

http://ibm.co/1UCa1j4

The expected output is too long to fully show in this publication, but Example 4-63 shows partially the expected output from the **knife os manage update topology topo.json** command.

Example 4-63 Sample output updating the cloud topology by adding PowerKVM Compute node

```
[root@cmodeploy itso-sde-deployment]# knife os manage update topology topo.json
Updating topology 'topo-ITSO-SDE-CLOUD' ...

Deploying topology 'topo-ITSO-SDE-CLOUD' ...
Deploying to nodes with run_order_number '2'
Bootstrapping nodes with environment ITSO-SDE-CLOUD...
Bootstrapping node ...
Connecting to powerkvm1.itso.ibm.com
.
. <many lines were removed>
.
powerkvm1.itso.ibm.com Chef Client finished, 84/114 resources updated in
181.523516415 seconds
All bootstrapped nodes with run_order_number '1' deployed.

Results for deploy of topology 'topo-ITSO-SDE-CLOUD'
Results for nodes with run_order_number '1'
Deploy of node at powerkvm1.itso.ibm.com was successful.
Deploy of topology 'top-powerkvm1.json' completed in 197 seconds.
```

Note: You could use the **knife os manage deploy topology topo.json** in this task, but that takes longer because it runs unnecessary commands in the Controller node, trying to redeploy what was deployed in section 4.5.7, “Deploying the Controller node” on page 225 and 4.6.8, “Deploy the KVM Compute node using the deployment server” on page 245.

Important: When you use a node-specific attribute file and needs to modify the topology for some reason (after the topology has already been deployed), the **knife os manage update topology topo.json** command will not read the node-specific attribute file. Besides modifying the file to keep track of the changes, you should use the command **knife node edit <node_name_fqdn>**. For example:

```
# knife node edit powerkvm1.itso.ibm.com
```

This will bring the node-specific attribute file loaded in the Chef repository, so you can modify it directly in the Chef repository.

4. Log in to the Controller node and run the **nova hypervisor-status-list** command to verify if the Compute nodes were successfully added to your OpenStack cloud:

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# nova hypervisor-status-list
+----+-----------------------+-----------------+-----------+--------------------+
| id | hypervisor_hostname   | hypervisor_type | ha_status | maintenance_status |
+----+-----------------------+-----------------+-----------+--------------------+
| 1  | cmo1.itso.ibm.com     | QEMU            | ok        | ok                 |
| 3  | cmo2.itso.ibm.com     | QEMU            | ok        | ok                 |
| 21 | powerkvm1.itso.ibm.com | QEMU            | ok        | ok                 |
| 22 | powerkvm2.itso.ibm.com | QEMU            | ok        | ok                 |
+----+-----------------------+-----------------+-----------+--------------------+
```

Note: It is required to source the `/root/openrc` file before running the command, to provide the `OS_USERNAME` and `OS_PASSWORD` variables to the OpenStack commands.

5. Log in to the Cloud Management dashboard using the Controller node IP address and click **Admin > Hypervisors** as shown in Figure 4-30.

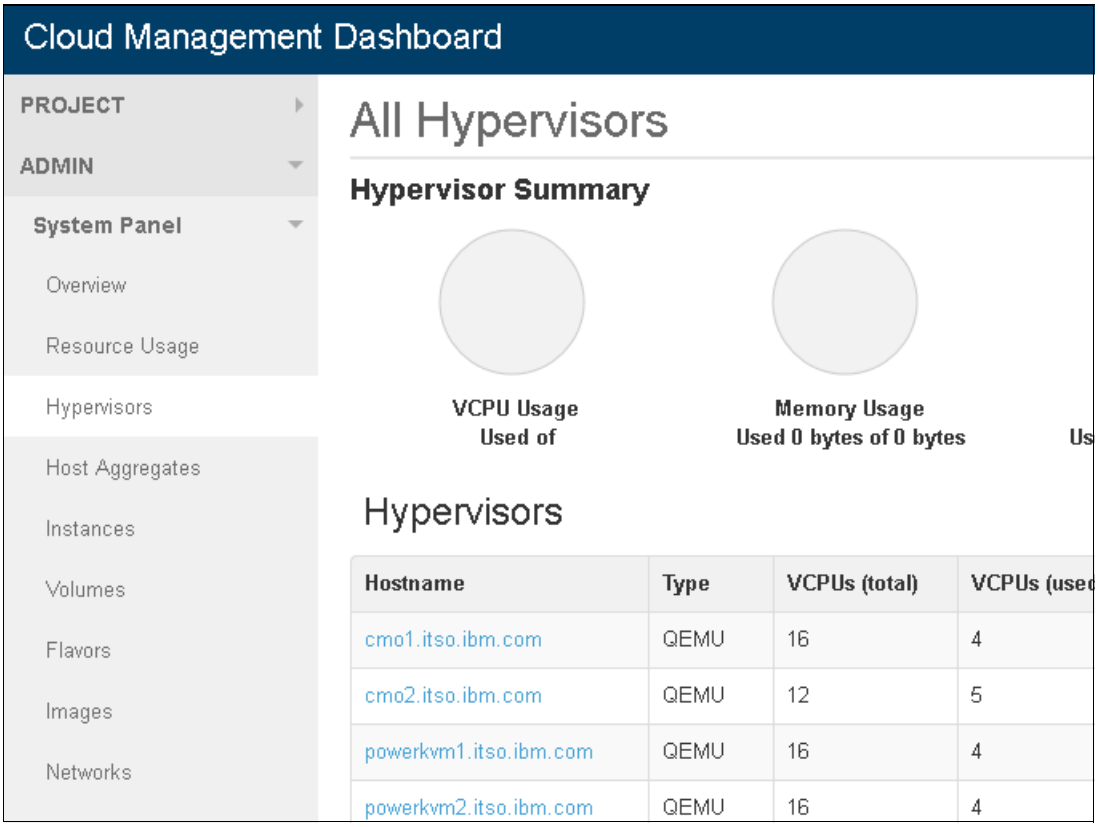


Figure 4-30 PowerKVM Compute nodes in the Cloud Management dashboard

4.7.8 Enable the Kimchi console in nova configuration file (optional)

While preparing the cloud for this publication, we noticed that instances deployed in the PowerKVM Compute nodes through the Cloud Manager did not have the console available in Kimchi (PowerKVM administration web interface). In general, you do not need this console, because you can access your instance using SSH network connection, but if you want to use the console, additional steps are required to enable the console.

1. Log in to your PowerKVM Compute node and open the `/etc/nova/nova.conf` file:

   ```
   # vi /etc/nova/nova.conf
   ```

2. Modify the `vncserver_listen` configuration to:

   ```
   vncserver_listen=0.0.0.0
   ```

3. Run the following command to restart the `openstack-nova-compute` service:

   ```
   # service openstack-nova-compute restart
   ```

Note: If you use the deployment server to update your topology, the nova.conf file in the PowerKVM Compute node will be overwritten and you need to apply this configuration again.

4. If you have instances running in the PowerKVM Compute node, you will need to modify the instance XML configuration file using the **virsh** command:

```
# virsh edit <instance_name>
```

5. Change the vnc section according to the following:

```
<graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0' keymap='en-us'>
      <listen type='address' address='0.0.0.0'/>
</graphics>
```

After these modifications, you can use the Kimchi to open a console for the instances running in the PowerKVM Compute node.

4.8 Add PowerVM Compute nodes to the cloud

The process of adding PowerVM Compute nodes (or hypervisors) to the OpenStack cloud involves setting up and preparing a Power Virtualization Center (PowerVC) and connecting the PowerVC to Cloud Manager. PowerVM Compute nodes cannot be managed directly by Cloud Manager,. They require PowerVC as an intermediary cloud management solution.

IBM Power Virtualization Center (PowerVC) is an enterprise virtualization management tool for Power Systems. It provides an enterprise simple graphical user interface (GUI) with excellent integration to PowerVM technologies. PowerVC is also built on OpenStack, using the same components as the Cloud Manager (nova, ego, neutron, keystone and cinder). By using the same components, Cloud Manager can easily integrate with PowerVC. Cloud Manager provides a PowerVC Driver, which connects Cloud Manager to the PowerVC, allowing you to operate PowerVC activities from the Cloud Management dashboard or command-line interface, integrating the PowerVM environment to your OpenStack cloud.

The PowerVM and PowerVC environment must be manually setup before adding PowerVM Compute nodes to your OpenStack cloud. After this step is completed, you will use the deployment server to add the PowerVM Compute nodes to your OpenStack cloud. In this process, the Chef server will configure the PowerVC driver in the Cloud Manager and connect with the PowerVC server. After this step is completed, you can operate your PowerVM environment from the Cloud Manager (for example creating and deleting instances and volumes), which will trigger commands to the PowerVC to implement in the PowerVM environment.

To add PowerVM Compute nodes to the OpenStack cloud, the following steps need to be accomplished:

1. Setup Virtual I/O Servers.
2. Setup Power Virtualization Center (PowerVC) server.
3. Add PowerVM Compute nodes to the OpenStack cloud.

4.8.1 Prerequisites for PowerVM Compute nodes

At the time of writing this publication, IBM Cloud Manager can connect to IBM Power Virtualization Center versions 1.2.0.1, 1.2.0.2, and 1.2.X. IBM Power Virtualization Center 1.2.1 can run on Red Hat Enterprise Linux (RHEL) 6.4 or 6.5 for IBM Power or x86_64. For more information about PowerVC planning and requirements, see the IBM Redbooks publication titled *IBM PowerVC Version 1.2.1 Introduction and Configuration*, SG24-8199:

http://www.redbooks.ibm.com/abstracts/sg248199.htm

Note: PowerVC 1.2.1 is recommended to take advantage of the latest PowerVC features. In the cloud created for this publication, PowerVC 1.2.1.1 was used.

IBM Cloud Manager 4.1 includes the following capabilities for managing PowerVC:

- Support for Power Systems hosts that are managed by a Hardware Management Console (HMC)
- Support for storage area networks
- Support for multiple Virtual I/O Servers in each physical host
- Support for multiple storage connectivity groups (for using Shared Storage Pools and V7000 SAN connectivity)

Important: PowerVC Express Edition is not supported from Cloud Manager, which means that physical hosts running Integrated Virtualization Manager (IVM) cannot be added to your OpenStack cloud at this time. Only PowerVC Standard Edition managing PowerVM nodes through an HMC be connected to the Cloud Manager. PowerKVM hosts must be directly managed by the Cloud Manager, without PowerVC intervention.

4.8.2 Set up Virtual I/O Servers

This section explains how to install and configure Virtual I/O Servers in your Power Systems, allowing them to be managed by IBM Power Virtualization Center (PowerVC). It includes a simple PowerVM VIO Server setup. For detailed information about implementing and configuring PowerVM VIO Servers, see *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940:

`http://www.redbooks.ibm.com/abstracts/sg247940.html`

Check whether Power System is connected to an HMC

Cloud Manager can manage PowerVM hosts connected to a Hardware Management Console (HMC). Check if your Power System is connected to your HMC by logging in as `hscroot` in your HMC and verifying if your managed systems appear under `Systems Management` → `Servers`. Also check whether they have `Operating` as the `status` as shown in Figure 4-31.

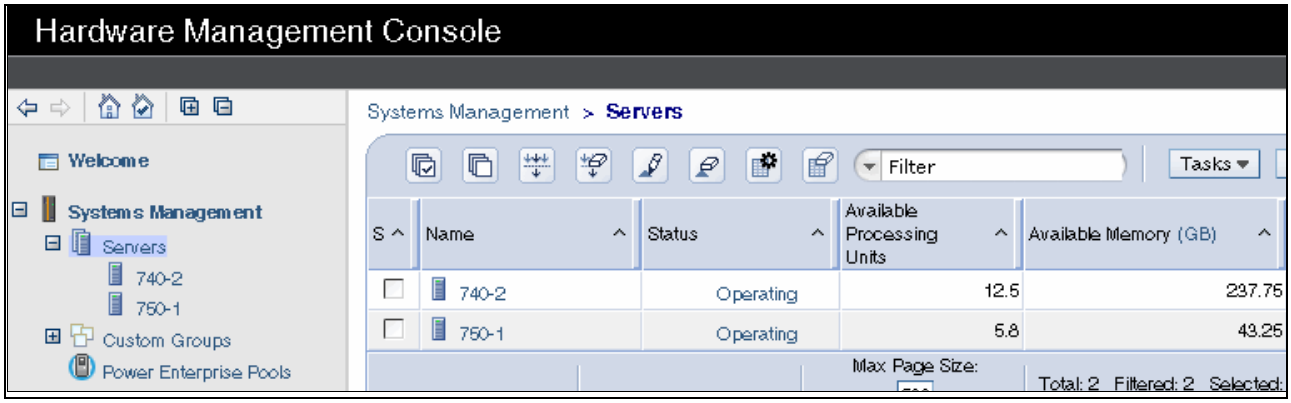


Figure 4-31 Checking HMC connection in PowerVM environment

Create the Virtual I/O Server logical partitions

Usually two Virtual I/O Servers are required to provide redundancy to the instances which will run in this physical server (network and disk redundancy). Use the following steps to create the logical partitions where the Virtual I/O Server (VIOS) will be installed:

1. Log in to your Hardware Management Console and, in the navigation panel, click **Systems Management > Servers** and select the managed system where you will create the logical partition.
2. Click **Configuration > Create Partition > VIO Server**, as shown in Figure 4-32.

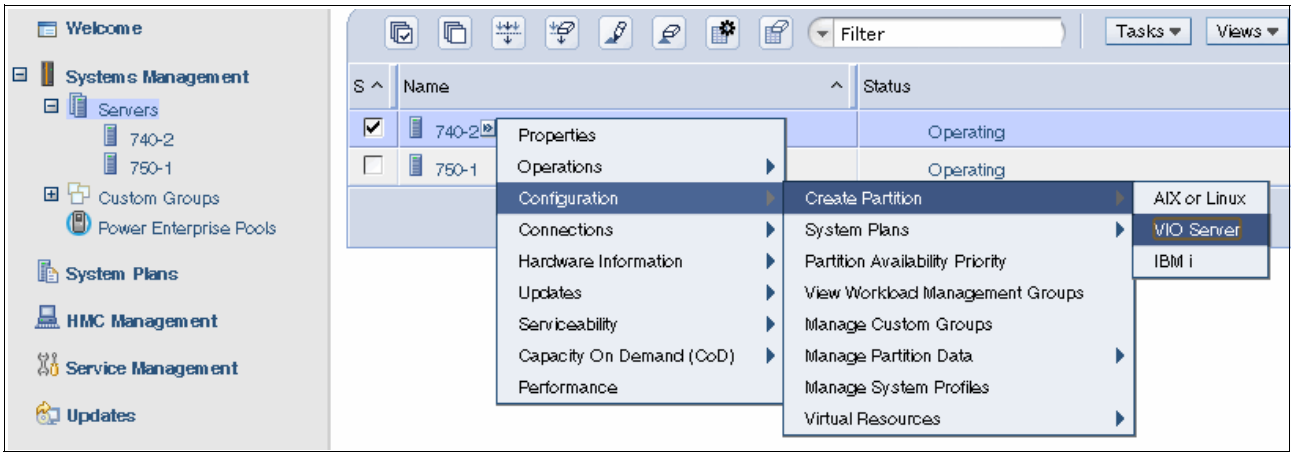


Figure 4-32 Creating VIO LPAR in Power System

3. Provide the Partition ID and Partition Name. Select the **Mover Service Partition** (to allow this VIOS to control the Live Partition Mobility operations, in case you plan to allow instance migration between physical servers).

For Sync Current configuration capability, select **Sync turned ON** (recommended for PowerVC managed systems), as shown in Figure 4-33.

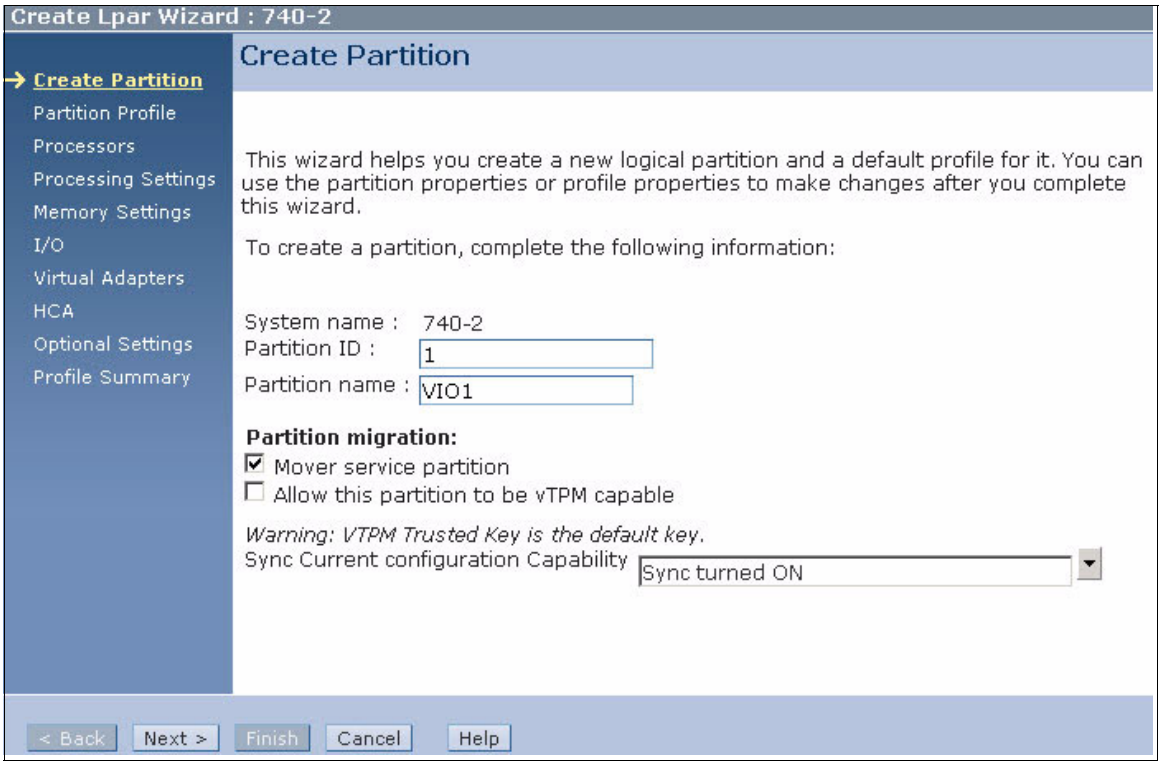


Figure 4-33 Creating VIO logical partition

4. Follow the Create LPAR wizard instructions to set up processor and memory configuration for the VIO LPAR.
5. Under I/O (physical I/O), provide at least two physical network adapters (to set up Shared Ethernet Adapter for management and data networks). Provide also at least one physical 8 Gigabit Fibre Channel adapter for NPIV.
6. Under Virtual Adapters, set up at least the following Virtual Ethernet Adapters (VEA):
   a. VEA for the management network

      Virtual Ethernet Adapter that will provide external network bridge to the management network for the instances running in this physical server. If you plan to add multiple hypervisors in your Cloud Manager, including PowerVM, VLAN networks is a requirement, so consider configuring the Virtual Ethernet Adapters with IEEE 802.1q enabled.

      Provide the Adapter ID, a non-valid Port Virtual Ethernet (VLAN ID), enable the **IEEE 802.1q compatible adapter** option, provide the valid VLAN IDs from your network (the same configuration must match in the physical port from your network switch), enable the option to **Use this adapter for Ethernet bridging**, and specify the Priority. (This priority configuration must be different in each Virtual I/O Server that will have a SEA configured in this network. Consider using 1 for VIO1 and 2 for VIO2, for example.)

Figure 4-34 shows an example of a valid configuration for this Virtual Ethernet Adapter.

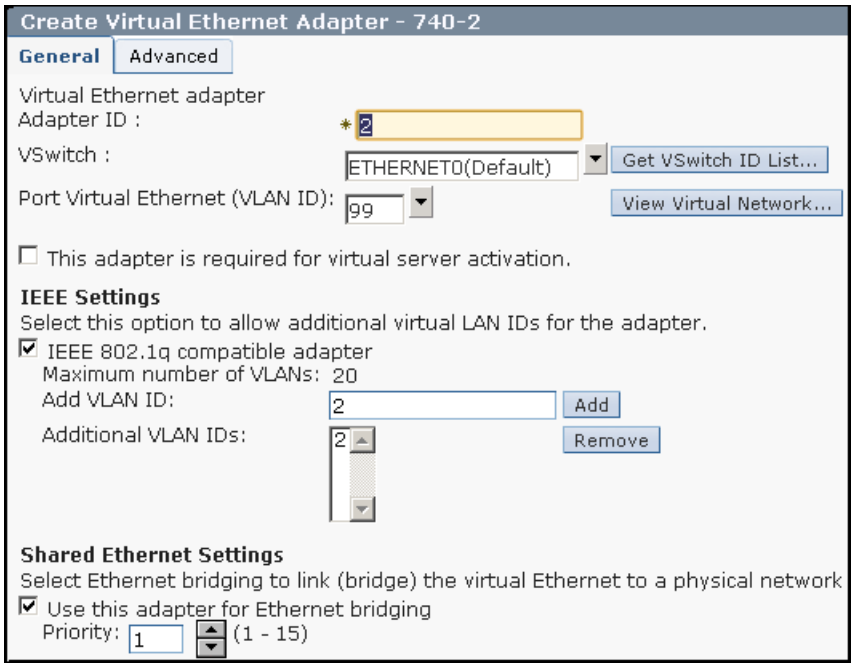


Figure 4-34 Virtual Ethernet Adapter configuration in PowerVM environment

> **Important:** You cannot provide the same priority to both Virtual I/O Servers Virtual Ethernet Adapters. Doing so might cause a loop in your physical network (broadcast storm).
>
> The Additional VLAN IDs provided here must match the VLAN ID configuration in your physical switch port.

b. VEA for the control channel for the management network

This Virtual Ethernet Adapter will be used for communication between Virtual I/O Servers for Shared Ethernet Adapter failover setup. Create a simple VEA and provide the Adapter ID and a non-valid Port Virtual Ethernet (VLAN ID), which must be different from the one used in step a on page 277.

Figure 4-35 shows an example of a valid configuration for this Virtual Ethernet Adapter.

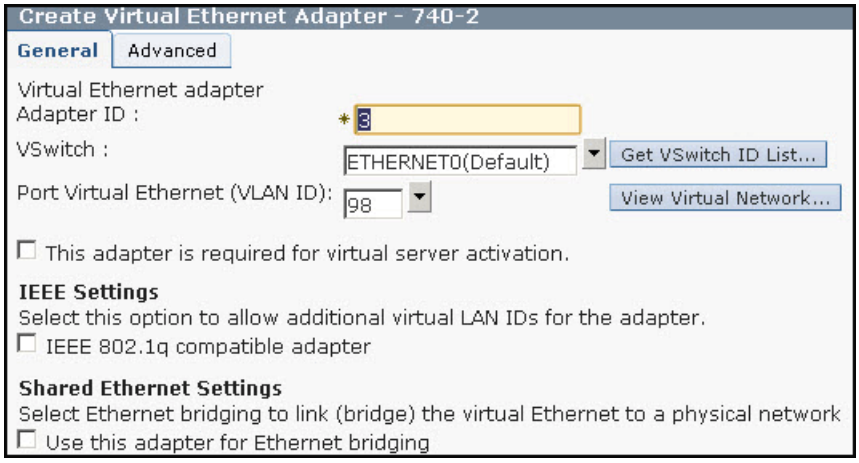


Figure 4-35 Control Channel Adapter configuration in PowerVM environment

c. VEA for data network

   This is the Virtual Ethernet Adapter that will be used to provide external network bridge to the data network for the instances running in this physical server. Consider using the same instructions from step a on page 277 with the appropriate values for each of its fields. The VLAN IDs provided here must match the configuration in your physical switch port for the data network.

d. VEA for control channel for the data network

   This is the Virtual Ethernet Adapter that will be used as Control Channel for communication between Virtual I/O Servers (VIOS) in the Shared Ethernet Adapter failover setup for the data network. Consider using the same instructions from step b on page 278 with the appropriate values for each of its fields.

Important: Unlike the instances deployed in KVM and PowerKVM Compute nodes, the instances deployed in PowerVM Compute nodes will be able to obtain IP address from both management and data networks. It is important to provide communication between instances and the Hardware Management Console (HMC) so you can make dynamic changes to your instances using DLPAR. This is also a requirement for migrating instances between physical servers using Live Partition Mobility.

In the cloud setup for this publication, the HMC is only connected to the management network, so it is required to provide both networks to the instances to allow them to communicate to the HMC (through the management network) and communicate to the instances from other Compute nodes (through the data network).

Note: In the cloud setup for this publication, due to physical switch limitations, we used untagged networks, setting Port VLAN ID 1 for the management network and Port VLAN ID 2 for the data network. The option IEEE 802.1Q was disabled. to allow communication between instances from PowerVM, PowerKVM and KVM, we manually added a tag with VLAN ID 2 in the packets in Open vSwitch at PowerKVM and KVM Compute nodes.

Install the Virtual I/O Server in the logical partitions

Obtain the Virtual I/O Server installation media and install in the logical partitions created on "Create the Virtual I/O Server logical partitions" on page 276. For detailed instructions on how to install Virtual I/O Server, see the Chapter 12 of *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940:

http://www.redbooks.ibm.com/abstracts/sg247940.html

Set up network communication in Virtual I/O Servers

This step provides you brief guidance on using the physical and virtual network adapters added to the Virtual I/O Server (VIOS) logical partition on section "Create the Virtual I/O Server logical partitions" on page 276 to create the Shared Ethernet Adapters (with SEA failover technology) and configuring an IP address for communication with the VIOS. For more information about SEA configuration and additional options for configuring it, see Chapter 16 from *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940.

1. Log in as `padmin` to your VIOS, and run the following command to identify the Virtual Ethernet Adapters available to this logical partition:

   ```
   $ lsdev -virtual | grep ent
   ```

   You should be able to see four virtual Ethernet adapters, as shown in Example 4-64.

   Example 4-64 List of Virtual Ethernet Adapters

   ```
   $ lsdev -virtual | grep ent
   ent2             Available   Virtual I/O Ethernet Adapter (l-lan)
   ent3             Available   Virtual I/O Ethernet Adapter (l-lan)
   ent9             Available   Virtual I/O Ethernet Adapter (l-lan)
   ent11            Available   Virtual I/O Ethernet Adapter (l-lan)
   ```

2. Run the following command to obtain the Location code for each Virtual Ethernet Adapter, so you can match this information with the Adapter ID provided in section "Create the Virtual I/O Server logical partitions" on page 276:

   ```
   $ lsmap -all -net | grep ent
   ```

   In Example 4-65 you can see the physical location for each of the virtual Ethernet adapters. The adapters ent2 and ent3 are used to set up the Shared Ethernet Adapter (SEA) for management network. The same steps should be used for setting up the data network SEA.

   Example 4-65 Identifying the virtual Ethernet adapters

   ```
   $ lsmap -all -net | grep ent
   ent2   U8233.E8B.061AA6P-V1-C2-T1
   ent3   U8233.E8B.061AA6P-V1-C3-T1
   ent9   U8233.E8B.061AA6P-V1-C9-T1
   ent11  U8233.E8B.061AA6P-V1-C10-T1
   ```

3. Identify the physical adapters available to the VIOS by running the following command:

   ```
   $ lsdev -type adapter | grep ent
   ```

   Example 4-66 on page 281 shows the expected output for this command.

Example 4-66 Physical and virtual adapters available for the VIOS

```
$ lsdev -type adapter | grep ent
ent0             Available   2-Port 10/100/1000 Base-TX PCI-X Adapter (14108902)
ent1             Available   2-Port 10/100/1000 Base-TX PCI-X Adapter (14108902)
ent2             Available   Virtual I/O Ethernet Adapter (l-lan)
ent3             Available   Virtual I/O Ethernet Adapter (l-lan)
ent5             Available   4-Port 10/100/1000 Base-TX PCI-Express Adapter (14106803)
ent6             Available   4-Port 10/100/1000 Base-TX PCI-Express Adapter (14106803)
ent7             Available   4-Port 10/100/1000 Base-TX PCI-Express Adapter (14106803)
ent8             Available   4-Port 10/100/1000 Base-TX PCI-Express Adapter (14106803)
ent9             Available   Virtual I/O Ethernet Adapter (l-lan)
ent11            Available   Virtual I/O Ethernet Adapter (l-lan)
```

4. Create the Shared Ethernet Adapter for the management networks with SEA failover technology as shown in Example 4-67.

Example 4-67 Setting up Shared Ethernet Adapter

```
$ mkvdev -sea ent0 -vadapter ent2 -default ent2 -defaultid 2 -attr ha_mode=auto
ctl_chan=ent3
ent4 Available
en4
et4
```

Note: Replace `ent0` for the physical adapter being used in your VIOS, `ent2` for the Virtual Ethernet Adapter configured for bridging the external access, `2` for the default Port VLAN ID and ent3 for the Virtual Adapter used as the Control Channel.

Important: The Ethernet switch port connected to the physical adapter must be corectly configured to use IEEE 802.1q VLAN tagged networks.

5. Create a VLAN adapter to tag the packets with correct VLAN ID information, so you can set up and IP address for your Virtual I/O Server, as shown in Example 4-68. Replace `ent4` for your Shared Ethernet Adapter and `2` for the appropriate VLAN ID from your management network.

Example 4-68 Creating VLAN Adapter

```
$ mkvdev -vlan ent4 -tagid 2
ent12 Available
en12
et12
```

6. Set up an IP address on the VLAN Tagged adapter created on step 5, ent12 in this case, using the following command:

```
$ mktcpip -hostname <vio_hostname> -inetaddr <ip_address> -netmask
<network_mask> -gateway <gateway_ip_address> -interface enX -start
```

Important: Make sure that your VIOS can ping the deployment server and the Controller node.

Note: Repeat the same steps to set up the management and data networks in the other Virtual I/O Server from your Power System.

Check whether Fibre Channel adapters are connected to the SAN

Make sure that the FC adapters that you will use to provide NPIV communication to your instances are connected to the storage area network.

1. Run the **lsnports** command to check whether they are available and connected to the SAN switch (if the ports are not NPIV capable or not connected to a SAN switch, this command will not show the ports). The expected output is similar to what is shown in Example 4-69:

Example 4-69 Using lsnports to check if adapter is connected to SAN

```
$ lsnports
name             physloc                        fabric tports aports swwpns  awwpns
fcs0             U5802.001.0086848-P1-C2-T1          1     64     59   2048    2027
fcs1             U5802.001.0086848-P1-C2-T2          1     64     60   2048    2028
```

2. Identify the WWPN of your FC adapter by running the following commands:

```
$ oem_setup_env
$ lscfg -vl fcsX
```

The expected output is shown in Example 4-70. The WWPN can be observed in the Network Address field.

Example 4-70 Obtaining the WWPN of a physical adapter

```
$ oem_setup_env
# lscfg -vl fcs0
  fcs0   U5802.001.0086848-P1-C2-T1  8Gb PCI Express Dual Port FC Adapter (df1000f114108a03)

        Part Number.................10N9824
        Serial Number...............1B02104269
        Manufacturer................001B
        EC Level....................D76482B
        Customer Card ID Number.....577D
        FRU Number..................10N9824
        Device Specific.(ZM)........3
        Network Address.............10000000C99FC71E
        ROS Level and ID............02781174
        Device Specific.(Z0)........31004549
        Device Specific.(Z1)........00000000
        Device Specific.(Z2)........00000000
        Device Specific.(Z3)........09030909
        Device Specific.(Z4)........FF781116
        Device Specific.(Z5)........02781174
        Device Specific.(Z6)........07731174
        Device Specific.(Z7)........0B7C1174
        Device Specific.(Z8)........20000000C99FC71E
        Device Specific.(Z9)........US1.11X4
        Device Specific.(ZA)........U2D1.11X4
        Device Specific.(ZB)........U3K1.11X4
        Device Specific.(ZC)........00000000
        Hardware Location Code......U5802.001.0086848-P1-C2-T1
```

3. Log in to your SAN switch and check whether the WWPN of your physical adapter is connected to the port where the fiber was physically plugged. Example 4-71 shows the output of the **portshow** command from an IBM 2498-B40 SAN switch, where we can find this information in the `portWwn of device(s) connected` field. Use the appropriate command for your SAN switch.

Example 4-71 Checking FC adapter connection in SAN switch

```
sansw1:admin> portshow 4
portIndex:   4
portName: port4
portHealth: HEALTHY

Authentication: None
portDisableReason: None
portCFlags: 0x1
portFlags: 0x1024b03     PRESENT ACTIVE F_PORT G_PORT U_PORT NPIV LOGICAL_ONLINE
LOGIN NOELP LED ACCEPT FLOGI
LocalSwcFlags: 0x0
portType:  17.0
POD Port: Port is licensed
portState: 1    Online
Protocol: FC
portPhys:  6    In_Sync         portScn:   32   F_Port
port generation number:    5468
state transition count:    17

portId:    010400
portIfId:    4302000e
portWwn:   20:04:00:05:1e:b5:a4:d6
portWwn of device(s) connected:
        10:00:00:00:c9:9f:c7:1e
Distance:  normal
portSpeed: N8Gbps

Credit Recovery: Inactive
Aoq: Inactive
FAA: Inactive
F_Trunk: Inactive
LE domain: 0
FC Fastwrite: OFF
Interrupts:        0          Link_failure: 0          Frjt:         0
Unknown:           221        Loss_of_sync: 0          Fbsy:         0
Lli:               94         Loss_of_sig:  9
Proc_rqrd:         381022     Protocol_err: 0
Timed_out:         0          Invalid_word: 0
Rx_flushed:        0          Invalid_crc:  0
Tx_unavail:        0          Delim_err:    0
Free_buffer:       0          Address_err:  0
Overrun:           0          Lr_in:        10
Suspended:         0          Lr_out:       0
Parity_err:        0          Ols_in:       0
2_parity_err:      0          Ols_out:      10
CMI_bus_err:       0
```

```
Port part of other ADs: No
```

If you plan to use IBM System Storage Storwize V7000 or SVC as a backup storage device for your instances, see section 4.8.3, “Set up the IBM Power Virtualization Center” on page 284.

Configure a Shared Storage Pool cluster (Optional)

If you do not have an IBM System Storage Storwize V7000 or IBM SAN Volume Controller (SVC) in your environment, you must configure a Shared Storage Pool in your Virtual I/O Server to use as a backup storage device for your instances. Even if you have the IBM storage subsystems mentioned, you have the optoin to configure the Shared Storage Pools in your environment.

Set up a Shared Storage Pool cluster using all Virtual I/O Servers from the physical servers where you want to place your instances. For example, if you have two physical servers with two Virtual I/O Servers each, and you want to be able to live-migrate your instances between these two physical servers, you need to set up a Shared Storage Pool cluster with four nodes (four VIO Servers). For more instructions on how to set up a Shared Storage Pool cluster, see Chapter 16 in the IBM Redbooks publication*IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940:

http://www.redbooks.ibm.com/abstracts/sg247940.html

Important: In the cloud setup for examples in this publication, Shared Storage Pools are not used. The PowerVC is connected directly to the SVC.

4.8.3 Set up the IBM Power Virtualization Center

This section explains how to install and configure the IBM Power Virtualization Center (PowerVC) and provides a simple implementation scenario. For more information about PowerVC, see *IBM PowerVC Version 1.2.1 Introduction and Configuration*, SG24-8199:

http://www.redbooks.ibm.com/abstracts/sg248199.html

Create the PowerVC logical partition

To create the logical partition to install the IBM Power Virtualization Center (PowerVC), log in to the Hardware Management Console (HMC). In the navigation panel, select the managed system where you want to place the logical partition, and click **Configuration > Create Logical Partition**.

Follow the Create Partition wizard instructions to create the logical partition and its profile.

Note: IBM Power Virtualization Center (PowerVC) can also be installed in a x86_64 server.

Install Red Hat Enterprise Linux

Install the Red Hat Enterprise Linux 6.4 or 6.5 in the LPAR that will host the IBM Power Virtualization Center (PowerVC). For detailed instructions on how to install RHEL, see the Installation Guide on the Red Hat website:

http://red.ht/1xEzlge

Configure a network connection to the management and data networks

The IBM Power Virtualization Center server needs to access the management and data networks. No network bridge is created in this server, so you can set up network configuration normally by using the **system-config-network** tool. Follow the instructions provided in 4.5.6, “Configure network interfaces in the Controller node” on page 220, to configure your network adapters.

Important: You can configure an IP on the data network interface. The PowerVC will not use network bridges with Open vSwitch, so you can normally configure an IP address on this interface. The instances get access to data network through VIOS SEAs.

Make sure an IP is configured in the management and data network as shown in Example 4-72.

Example 4-72 PowerVC network configuration

```
[root@powervc ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 2A:E6:AE:DC:E0:02
          inet addr:172.16.21.135  Bcast:172.16.23.255  Mask:255.255.252.0
          inet6 addr: fe80::28e6:aeff:fedc:e002/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8913135 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6719945 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4190487874 (3.9 GiB)  TX bytes:2348165685 (2.1 GiB)
          Interrupt:18

eth1      Link encap:Ethernet  HWaddr 2A:E6:AE:DC:E0:03
          inet addr:192.168.21.135  Bcast:192.168.21.255  Mask:255.255.255.0
          inet6 addr: fe80::28e6:aeff:fedc:e003/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1628296 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:151773207 (144.7 MiB)  TX bytes:1376 (1.3 KiB)
          Interrupt:19

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:144850197 errors:0 dropped:0 overruns:0 frame:0
          TX packets:144850197 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:75985859776 (70.7 GiB)  TX bytes:75985859776 (70.7 GiB)
```

The network configuration files for management and data network interfaces should be similar to Example 4-73 on page 286.

Example 4-73 Network configuration files in PowerVC server

```
[root@powervc ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=2a:e6:ae:dc:e0:02
TYPE=Ethernet
UUID=044Ocba7-a45d-41cc-9769-5Oca56c13bf0
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=none
IPADDR=172.16.21.135
NETMASK=255.255.252.0
GATEWAY=172.16.20.1
IPV6INIT=no
USERCTL=no

[root@powervc ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
HWADDR=2a:e6:ae:dc:e0:03
TYPE=Ethernet
UUID=9d8b9e5d-O05e-4614-86d8-fe22b74a7dca
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=none
IPADDR=192.168.21.135
NETMASK=255.255.255.0
IPV6INIT=no
USERCTL=no
```

Install IBM Installation Toolkit for PowerLinux

Follow the instructions from “Install IBM Installation Toolkit for PowerLinux” on page 203 to install the IBM Installation Toolkit for PowerLinux and enable RMC communication between the PowerVC LPAR and the HMC, enabling dynamic changes in the PowerVC LPAR (DLPAR).

Use a fully qualified domain name

Follow the instructions from “Use a fully qualified domain name” on page 204 to configure your PowerVC server to use a fully qualified domain name (FQDN).

Configure host name resolution

Follow the instructions from “Configure host name resolution” on page 204 to configure your PowerVC server to be able to resolve host names to IP and IP to host names by using DNS or `/etc/hosts`.

Synchronize date and time

Follow the instructions from “Synchronize the date and time” on page 205 to configure your PowerVC server with the same date and time as the other nodes in your cloud. You can use NTP client or manually set up the date and time to complete this requirement.

Configure the Yum repository

Follow the instructions from “Configure the Yum repository” on page 206 to set up a Yum repository in your PowerVC server. This Yum repository will be used during PowerVC installation to install prerequisite packages.

Important: Setting up Yum repository with the RHEL installation media is a requirement for PowerVC installation.

Define the same root password on the nodes

Consider applying the recommendations from “Define the same root password on the nodes” on page 206 in your PowerVC server.

SELinux setting configuration

For PowerVC to work corectly, SELinux must be set to `permissive` or `disable`. To verify your SELinux setting, run the **`sestatus`** command or check the `/etc/selinux/config` file and ensure that SELINUX is set up to `permissive` or `disabled`, as shown in Example 4-74. If your server does not have the appropriate configuration, modify the `/etc/selinux/config` file and reboot the server.

Example 4-74 SELinux configuration in PowerVC server

```
[root@powervc-temp-fabio ~]# sestatus
SELinux status:                 disabled

[root@powervc-temp-fabio ~]# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Install IBM Power Virtualization Center (PowerVC)

After meeting all requirements, you are now ready to start the PowerVC installation. Download the PowerVC installation binary files from the following IBM Entitled Systems Support web page:

https://www.ibm.com/servers/eserver/ess/

IBM Power Virtualization Center is licensed software, so you must be entitled to download it from the IBM website. Entitlement is bound to your Power System serial number. Down the PowerVC package by following these instructions:

1. Log in with your IBM ID.
2. Select **Software downloads**.
3. Select the **Power (AIX)** brand.
4. Select your **IBM Customer Number** (ICN).
5. Under 5692-A6P, select the PowerVC edition that you purchased, and click **Continue.**
6. Download either the `ppc64` or `x86_64 .tgz` file, according to the hardware where you plan to install the PowerVC software.

Important: You must register your IBM ID with your IBM Customer Number before downloading the PowerVC software.

7. After downloading the appropriate `.tgz` file, upload it to your PowerVC LPAR and extract it by using the following command:

```
# tar -zxvf powervc-install-ppc-rhel-1.2.1.1.tgz
```

8. Switch to the extracted directory and run the `install` script:

```
# cd powervc-1.2.1.1
# ./install
```

Important: PowerVC installation will use the IP configured in `eth0` as the network interface. If the management network IP address is not set to `eth0`, you must modify this information by using the *`HOST_INTERFACE`* variable. Export this variable with the correct network adapter before starting the PowerVC installation:

```
# export HOST_INTERFACE=eth1
```

9. Select the type of the installation. In this case it must be **2 - Standard managing PowerVM**:

```
Select the offering type to install:
   1 - Express  (IVM support)
   2 - Standard managing PowerVM
   3 - Standard managing PowerKVM
   9 - Exit
```

10. Read and accept the license agreements and PowerVC installation will start. The output from the installation is too big to add to this publication, but Example 4-75 shows partially the expected output from the installation.

Example 4-75 PowerVC installation output

```
################################################################################
#
Starting the IBM PowerVC 1.2.1.0 Installation on:
2014-08-25T15:51:45-04:00
################################################################################
#

LOG file is /opt/ibm/powervc/log/powervc_install_2014-08-25-155139.log

Checking system prerequisites.
Installation task 1 of 13
.
. <many lines were removed>
.
********************************************************************************
PowerVC installation successfully completed at 2014-08-25T16:03:44-04:00.
 Refer to
 /opt/ibm/powervc/log/powervc_install_2014-08-25-155139.log
 for more details.
********************************************************************************
Use a web browser to access IBM PowerVC at
   https://powervc.itso.ibm.com
```

Configure IBM Power Virtualization Center (PowerVC)

Now that PowerVC is corectly installed, configure it to manage your Power System:

1. Log in to your PowerVC web interface by using the following address:

   `https://<your_powervc_ip_address>`

2. Provide your root user ID and password and click **Log In**.
3. Click **Add Host**, as shown in Figure 4-36.

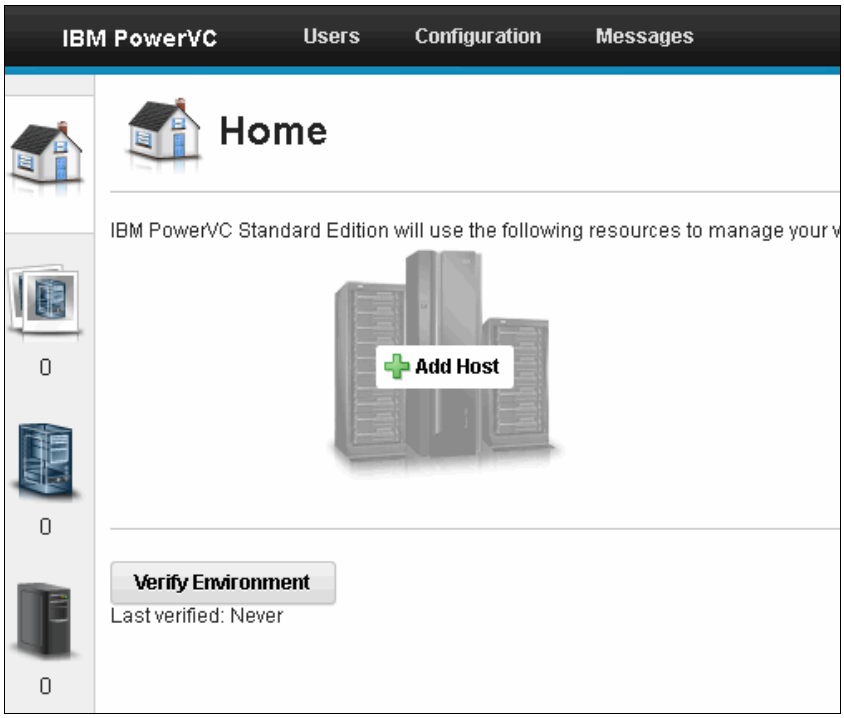


Figure 4-36 PowerVC Add Host screen

4. Provide the HMC IP address, `hscroot` user ID and `password`, and a name to identify your HMC on the PowerVC web interface, as shown in Figure 4-37.

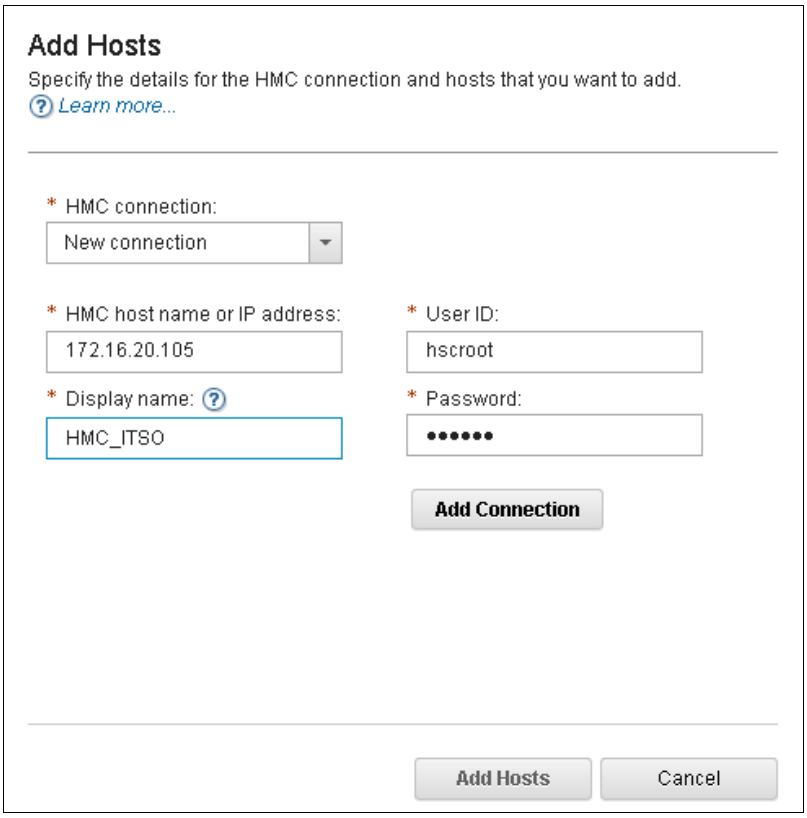


Figure 4-37 Adding HMC connection to the PowerVC

5. Click **Connect** to add the HMC, as shown in Figure 4-38.

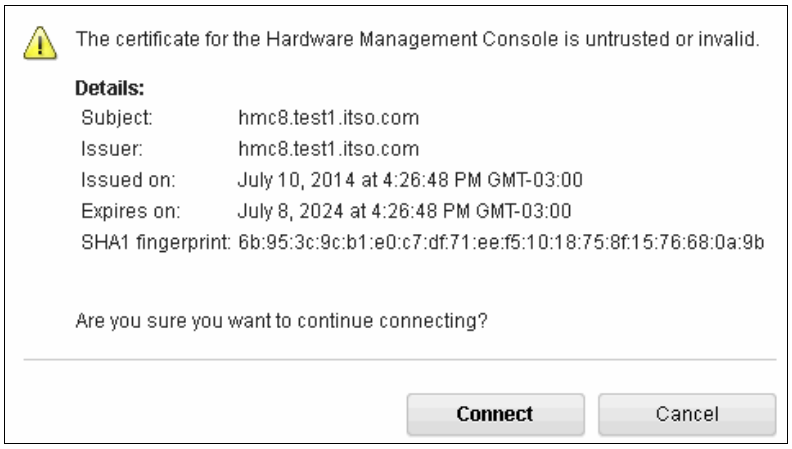


Figure 4-38 Accepting HMC connection to the PowerVC

6. PowerVC will automatically discover the Power Systems managed by this HMC and provide you a list. Select the managed systems that you want to add to your OpenStack cloud and click **Add Hosts**, as shown in Figure 4-39.

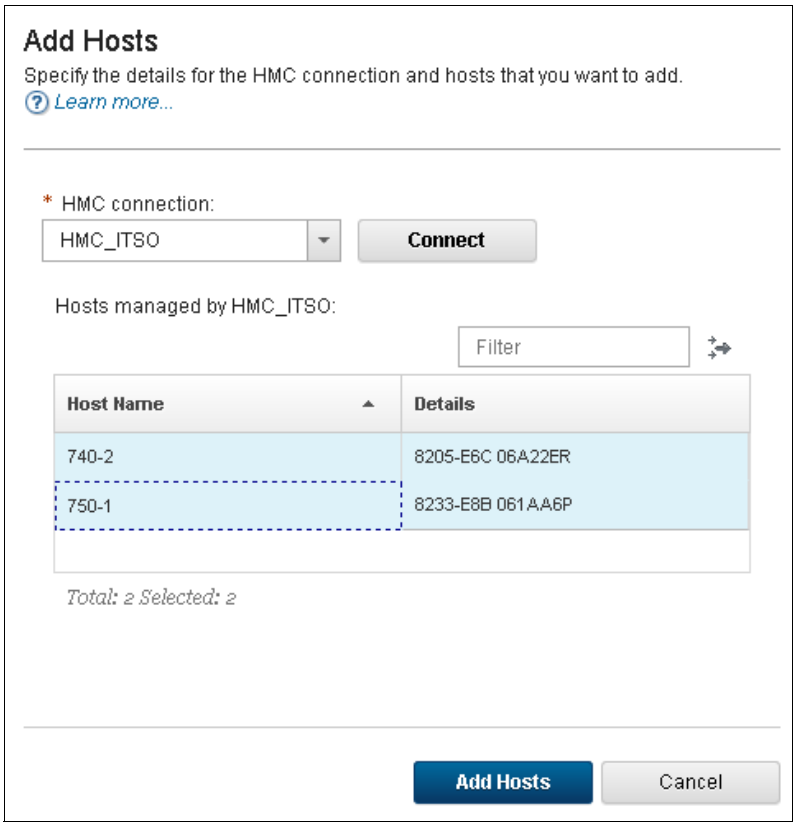


Figure 4-39 Adding managed systems to PowerVC

7. Now click **Add Storage** in the main screen of your PowerVC web interface, as shown in Figure 4-40.

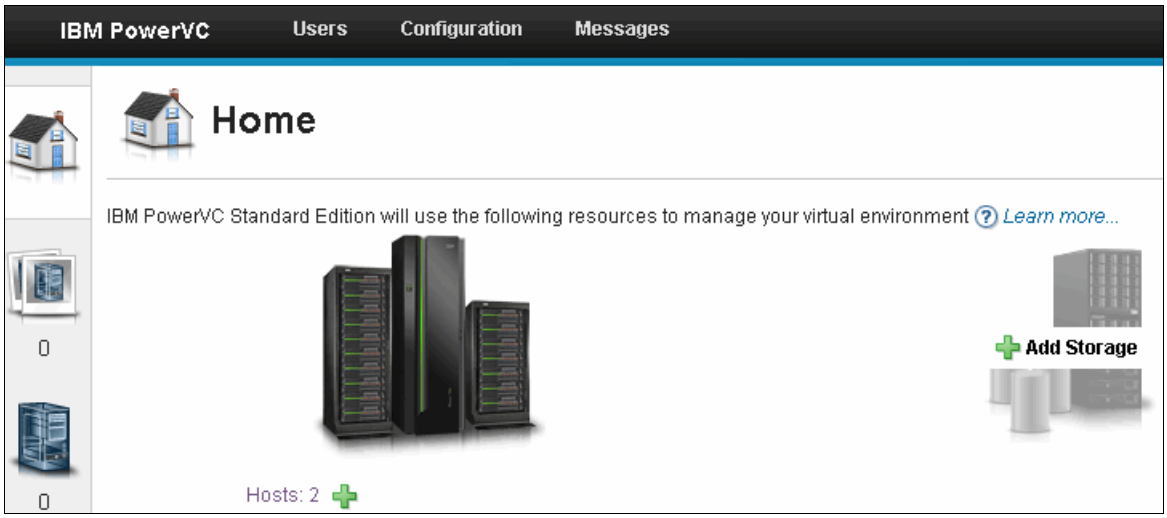


Figure 4-40 Managing Storage in PowerVC

8. Provide the IP address of your SVC or V7000 storage subsystem, user ID (superuser), password, and a display name to identify your storage in the PowerVC web interface, as shown in Figure 4-41. Then, click **Connect**.

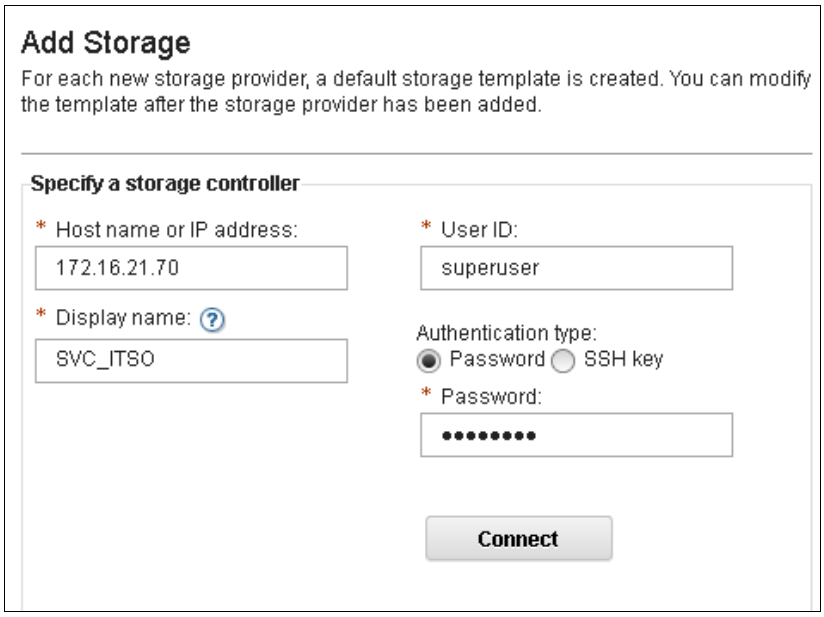


Figure 4-41 Adding SVC or V7000 Storage to PowerVC

9. Select your storage pool where you want to place the volume for the instances, and click **Add Storage**, as shown in Figure 4-42 on page 293.

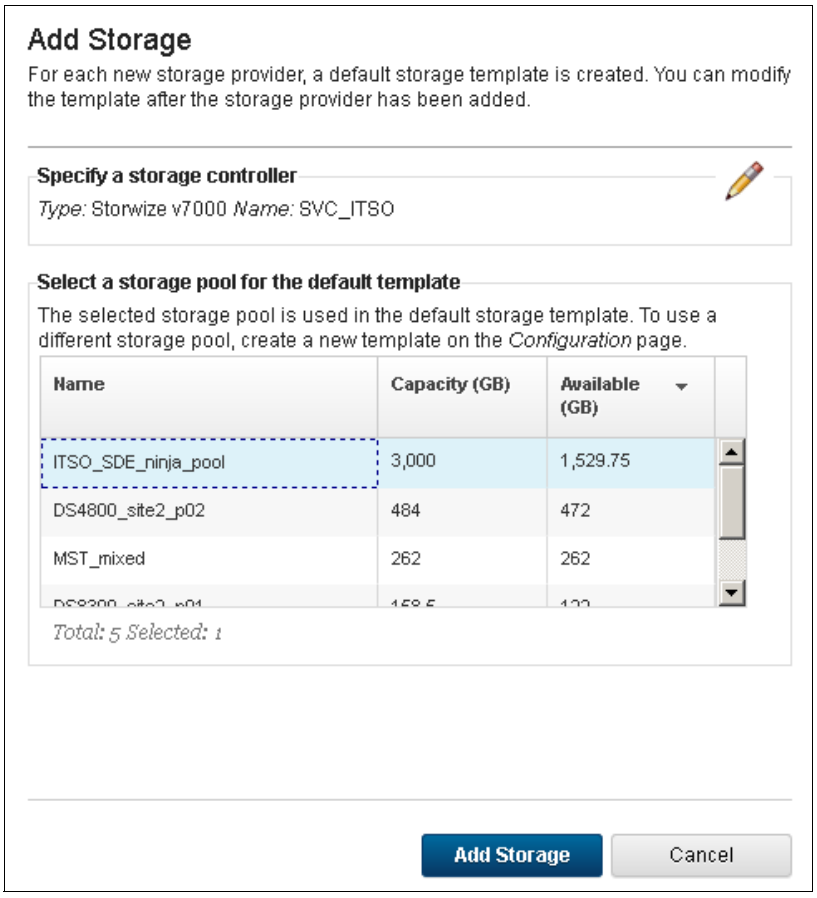


Figure 4-42 Selecting your storage pool from SVC or V7000

10. PowerVC will automatically prompt you to Add Fibre Channel Fabrics. Click `Add Fabric` as shown in Figure 4-43.



Figure 4-43 Adding Fabric to the PowerVC

11. Provide the SAN switch IP address, user ID, password, and display name to identify your SAN switch in the PowerVC, as shown in Figure 4-44, and click **Add Fabric**.

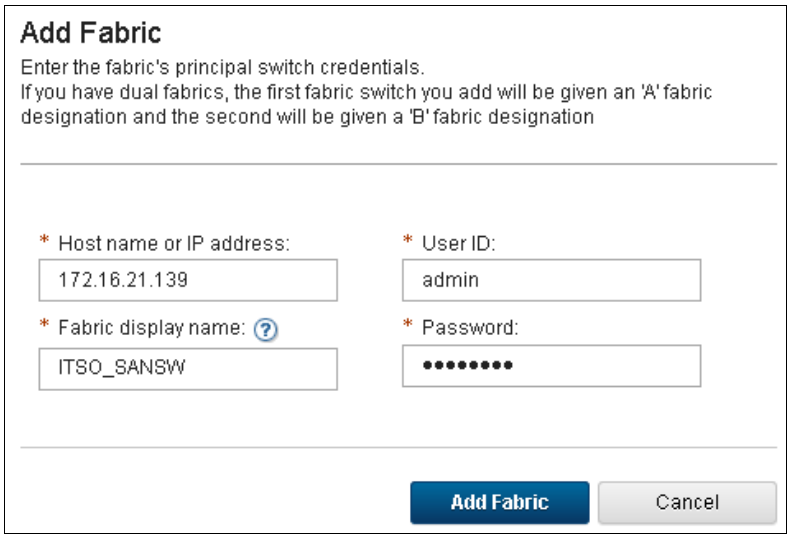


Figure 4-44 Providing fabric credentials

12. After completing this task, click **Hosts** at the left side of the PowerVC web interface, select your managed system, and click **Manage Existing virtual machines**, as shown in Figure 4-45.

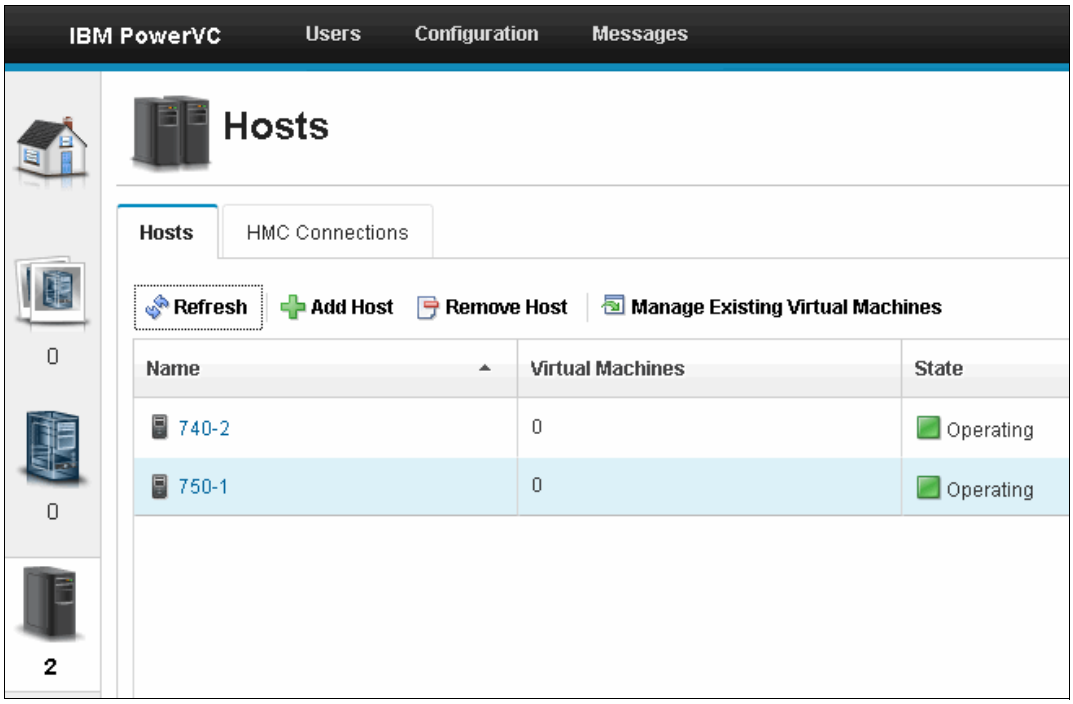


Figure 4-45 Managing existing virtual machines

13. Select **Manage any supported virtual machines that are not currently being managed by PowerVC** and click **Manage**.

14. By completing this task, the networks currently configured in your virtual machines will be automatically added to the PowerVC, but they cannot be used while deploying new virtual machines, so click in **Add symbol on Networks** to provide your network information, as shown in Figure 4-46.

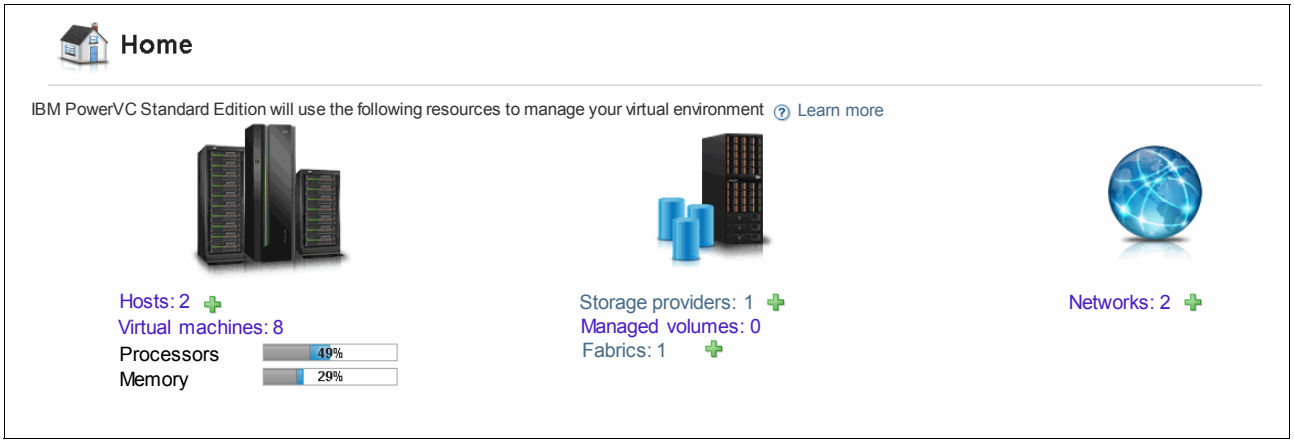


Figure 4-46 Adding networks to the PowerVC

15. Provide your network configuration details, including Name, VLAN ID, Subnet mask, Gateway, Primary and Secondary DNS configuration, and then click **Add Network**, as shown in Figure 4-47.

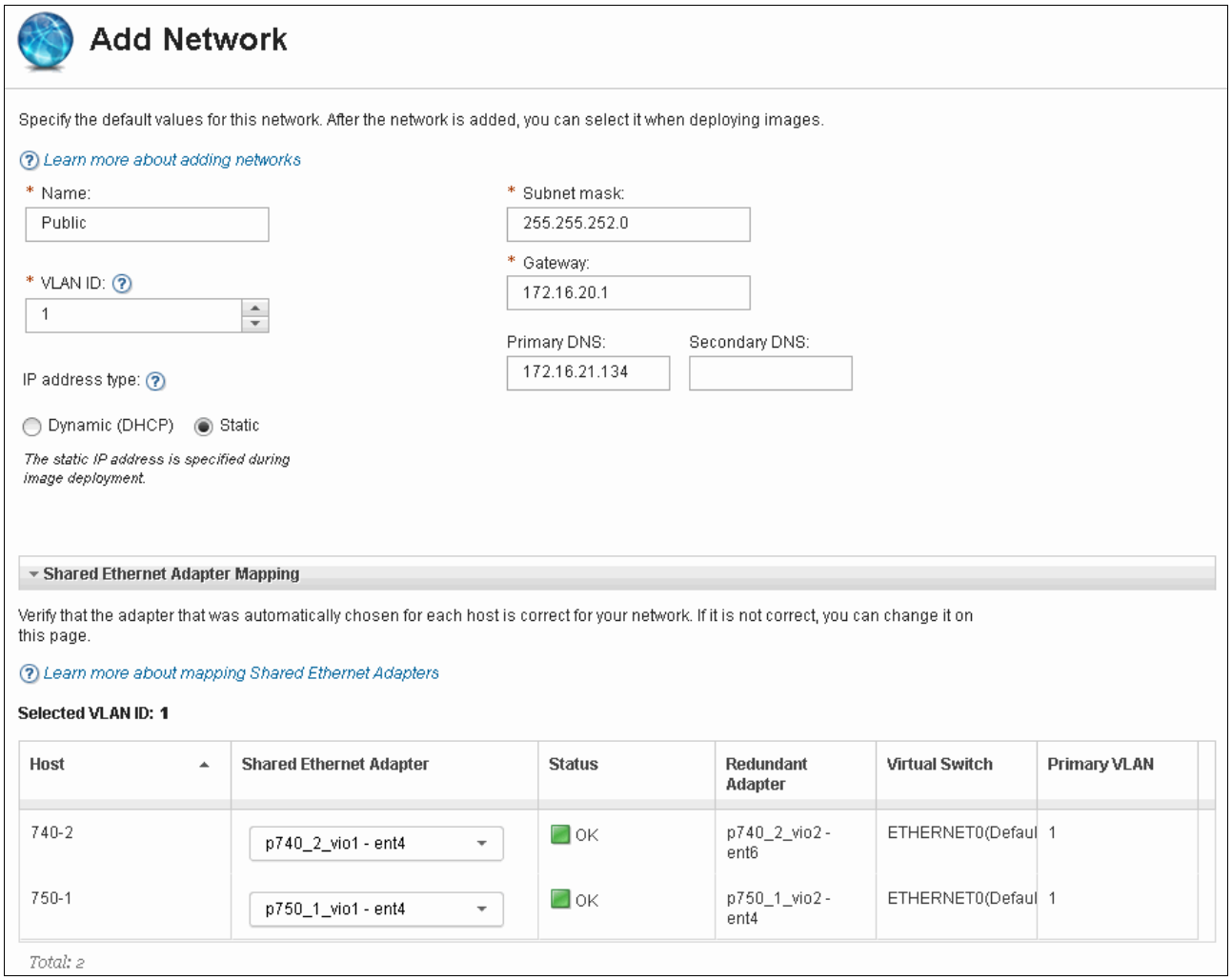


Figure 4-47 Adding the network configuration to the PowerVC

Note: When you select the `VLAN ID`, the `Shared Ethernet Adapter Mapping` section will be automatically updated with the Virtual I/O Servers and Shared Ethernet Adapters capable of providing network access to that `VLAN ID`. Even if your network does not have a Default Gateway, you should provide one IP address in this field, so that PowerVC will know the valid range of IP addresses for that network.

Repeat these steps to all Power Systems, storage subsystems, virtual machines, and networks. After completing this process, the PowerVC is ready to be connected to your Cloud Manager installation.

4.8.4 Add PowerVM Compute nodes to the OpenStack cloud

To add PowerVM Compute nodes to your OpenStack cloud, you need to use the deployment server to complete this task. The deployment server will use the same input files (`env.json` and `topo.json`) to connect the Controller node to the PowerVC server by installing and configuring the PowerVC driver in the Controller node.

The following steps add the PowerVM Compute nodes to your OpenStack cloud:

1. Update the environment file (`env.json`).
2. Update the data bags.
3. Create a node-specific attribute file.
4. Update the topology file (`topo.json`).
5. Customize the `nova.conf` file.
6. Use the knife command to update your cloud topology.

Update the environment file, env.json

Some specific modifications must be made to the `env.json` file to add PowerVM Compute nodes. Some of these changes affect other Compute nodes in your OpenStack cloud. We provide you instructions about what changes can be made in the `env.json` file directly, without affecting the other Compute nodes in your cloud. The changes that would affect them are made later by using a node-specific attribute file.

1. Log in to your deployment server and open the env.json file that contains the environment definitions for your cloud:

```
# cd /<path_to_your_cloud_directory_structure>
# vi env.json
```

2. Check if the name and description is correct, according to your cloud definition, as shown in Example 4-76.

Example 4-76 Name and description from your cloud in the env.json file

```
1 {
2   "name": "ITSO-SDE-CLOUD",
3   "description": "ITSO CLOUD",
```

3. Make sure that the endpoints.host, endpoints.bind-host, endpoints.mq.host and endpoints.db.host fields matches the IP address of the Controller node as shown in Example 4-77.

Example 4-77 endpoints configuration in the env.json file

```
215       "endpoints": {
216         "host": "172.16.21.137",
217         "bind-host": "172.16.21.137",
218         "mq": {
219           "host": "172.16.21.137",
220           "port": "5671"
221         },
222         "db": {
223           "host": "172.16.21.137",
224           "port": "50000"
225         }
226       },
```

4. Modify the following fields:
   - `ibm-openstack.powervc-driver.powervc.admin.user`

     Specify the administrator user name for your PowerVC server.
   - `ibm-openstack.powervc-driver.powervc.auth_url`

     Specify the authentication url for PowerVC server.
   - `ibm-openstack.powervc-driver.powervc.qpid.host`

     Specify the hostname for the PowerVC server.
   - `ibm-openstack.powervc-driver.db_create`

     Set to `true`.

Example 4-78 shows this section of the `env.json` file in the cloud prepared for this publication. The PowerVC server host name is `powervc.itso.ibm.com`, and the `root` user is the admin.

Example 4-78 PowerVC driver configuration in env.json file

```
151       "powervc-driver": {
152         "powervc": {
153           "admin_user": "root",
154           "auth_url": "https://powervc.itso.ibm.com/powervc/openstack/identity/v3",
155           "qpid": {
156             "host": "powervc.itso.ibm.com"
157           }
158         },
159         "db_create": true
160       },
```

Important: The `ibm-openstack.powervc-driver.powervc.auth_url` is not the URL that you use in your browser to access the PowerVC server. It is the URL for the authentication engine in PowerVC. In general, you should modify only the host name from your PowerVC server and preserve the rest of the URL:

`https://<provide_your_powervc_hostname_here>/powervc/openstack/identity/v3`

Example 4-79 shows the default initial content for the `ibm-openstack.powervc-driver.powervc.auth_url` entry from the `example-ibm-os-single-controller-n-compute` environment in the Chef repository (you used this example in section 4.5.7, “Deploying the Controller node” on page 225, to create your `env.json` file). This is what you will see in your `env.json` file before you modify it. In this case, replace only `powervc_host` for your PowerVC server IP address.

Example 4-79 Default ibm-openstack.powervc-driver.powervc.auth_url entry in the env.json file

```
151       "powervc-driver": {
152         "powervc": {
153           "admin_user": "root",
154           "auth_url": "https://powervc_host/powervc/openstack/identity/v3",
155           "qpid": {
156             "host": "powervc_host"
157           }
158         },
159         "db_create": false
160       },
```

5. Check whether your `openstack.network.core_plugin` is set to `neutron.plugins.ml2.Plug-in.Ml2Plugin` (default configuration). If so, make sure that `openstack.network.ml2.tenant_network_types` list `vlan` in the value list and `openstack.network.ml2.network_vlan_ranges` list `default:1:4094` in the value list, as shown in Example 4-80.

Example 4-80 Network plug-in configuration

```
242       "network": {
243         "core_plugin": "neutron.plugins.ml2.Plug-in.Ml2Plugin",
244         "use_namespaces": false,
245         "allow_overlapping_ips": true,
246         "rpc_thread_pool_size": 128,
247         "rpc_conn_pool_size": 60,
248         "rpc_response_timeout": 600,
249         "service_plugins": [
250           "neutron.services.l3_router.l3_router_plugin.L3RouterPlugin"
251         ],
252         "dhcp": {
253           "ovs_use_veth": false
254         },
255         "quota": {
256           "driver": "neutron.db.quota_db.DbQuotaDriver",
257           "items": "network,subnet,port",
258           "default": -1,
259           "network": 10,
260           "subnet": 10,
261           "port": 50,
262           "security_group": 10,
263           "security_group_rule": 100
264         },
265         "ml2": {
266           "type_drivers": "local,flat,vlan,gre",
267           "tenant_network_types": "vlan",
268           "mechanism_drivers": "openvswitch",
269           "flat_networks": "default",
270           "network_vlan_ranges": "default:1:4094",
271           "tunnel_id_ranges": "1:1000"
272         },
```

Note: There are other possibilities of neutron configuration. For more information, see IBM Cloud Manager with OpenStack 4.1 Administrator Guide available at:

http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html

6. Make sure that `openstack.network.openvswitch.enable_tunneling` is set to False as shown in Example 4-81.

Example 4-81 Disable tunneling

```
273         "openvswitch": {
274           "tenant_network_type": "vlan",
275           "network_vlan_ranges": "default:1:4094",
276           "enable_tunneling": "False",
277           "tunnel_type": "gre",
278           "tunnel_id_ranges": "1:1000"
279         }
280       },
```

7. Note the file names for your data bags, because you will need to modify them later. The data bags file names can be obtained in the `openstack.secret` section as shown in Example 4-82.

Example 4-82 Data bags information from env.json file

```
183       "secret": {
184         "key_path": "/etc/chef/encrypted_data_bag_secret",
185         "service_passwords_data_bag": "itso_sde_service_passwords",
186         "db_passwords_data_bag": "itso_sde_db_passwords",
187         "user_passwords_data_bag": "itso_sde_user_passwords",
188         "secrets_data_bag": "itso_sde_secrets"
189       },
```

Update the data bags

To allow the Controller node to connect with the PowerVC server, you have provided the authentication URL and administrator username in the `env.json` file, but for security reasons, the password was not provided. This password must be provided through the data bags, so it will be encrypted and securely stored in the Controller node.

There are two passwords that must be provided for the Controller node, so it can connect to the PowerVC server using the PowerVC driver: The administrator password and the `qpid` password. Two data bag files must be modified to provide these credentials.

The `qpid` daemon is a messaging tool that enables communication between hosts using AMQP protocol. For more information about qpid, see the Apache.org website:

`https://qpid.apache.org/`

Both the Controller node and the PowerVC server use qpid services for communication between OpenStack components in the Controller and Compute nodes.

The following steps will guide you to update the data bags with the appropriate credentials:

1. Access the directory where your data bags were prepared in section “Prepare the data bags” on page 230. If you do not remember where the files are, you can check the location in the **“**`secret_file`**”** attribute from your topology file (`topo.json`), as shown in Example 4-83 on page 301.

Example 4-83 Accessing the data bags directory

```
[root@cmodeploy itso-sde-deployment]# grep secret_file topo.json
  "secret_file":"/root/cmo/itso-sde-deployment/itso-secret-key",

[root@cmodeploy itso-sde-deployment]# cd
/root/cmo/itso-sde-deployment/data_bags/
```

2. Both files that you need to modify are from the `user_passwords_data_bag`. You have obtained the directory for this data bag in step on page 300 from section “Update the environment file, env.json” on page 297:

   ```
   # cd itso_sde_user_passwords
   ```

3. Modify the `pvcadmin.json` file and provide the password of the administrator user for PowerVC server. This is the password from the user specified in step 4 on page 298 on section “Update the environment file, env.json” on page 297:

   ```
   # vi pvcadmin.json
   ```

   Example 4-84 shows the file modified to provide the PowerVC administrator user password.

   Example 4-84 Providing the administrator password for PowerVC server in data bag

   ```
   [root@cmodeploy itso_sde_user_passwords]# cat pvcadmin.json
   {
     "id": "pvcadmin",
     "pvcadmin": "itsoadmin"
   }
   ```

   Important: Only the password should be provided in this file by modifying it in the `pvcadmin` field. *Do not* change the `id` field. Note that the file does not mention `root` as the `id`. This field will link the data bag with the user specified in the attribute `ibm-openstack.powervc-driver.powervc.admin.user` in the `env.json` file.

   For example, in the configuration above, the Controller node will use the `ibm-openstack.powervc-driver.powervc.admin.user` (`root`) as the user ID to log in the PowerVC server and `itsoadmin` as the root password (obtained from the data bag).

4. Log in to the PowerVC server and obtain the qpid password from the file `/opt/ibm/powervc/data/pw.file` as shown in Example 4-85.

   Example 4-85 Obtaining the qpid password from the PowerVC server

   ```
   [root@powervc ~]# cat /opt/ibm/powervc/data/pw.file
   cdb3e29615b8849630b6
   ```

5. Back on the deployment server, modify the `pvcqpid.json` file to provide the qpid password from PowerVC server, which you obtained in step 4:

   ```
   # vi pvcqpid.json
   ```

Example 4-86 shows the file modified to provide the qpid password.

Example 4-86 Modifying the pvcqpid.json file

```
[root@cmodeploy itso_sde_user_passwords]# cat pvcqpid.json
{
  "id": "pvcqpid",
  "pvcqpid": "cdb3e29615b8849630b6"
}
```

Important: Only the password should be provided in this file by modifying it in the `pvcqpid` field. *Do not* change the `id` field.

6. Access the parent directory of the data_bags directory and run the **`knife data bag from file <your_user_passwords_file>`** to update the `user_passwords_data_bag` with the modified files as shown in Example 4-87.

Example 4-87 Updating the data bags in the Chef repository

```
# cd /root/cmo/itso-sde-deployment
# knife data bag from file itso_sde_user_passwords --all --secret-file
itso-secret-key
Updated data_bag_item[itso_sde_user_passwords::pvcqpid]
Updated data_bag_item[itso_sde_user_passwords::db2inst1]
Updated data_bag_item[itso_sde_user_passwords::zlinuxroot]
Updated data_bag_item[itso_sde_user_passwords::xcat]
Updated data_bag_item[itso_sde_user_passwords::admin]
Updated data_bag_item[itso_sde_user_passwords::db2fenc1]
Updated data_bag_item[itso_sde_user_passwords::qpidadmin]
Updated data_bag_item[itso_sde_user_passwords::sceagent]
Updated data_bag_item[itso_sde_user_passwords::pvcadmin]
Updated data_bag_item[itso_sde_user_passwords::qpidclient]
Updated data_bag_item[itso_sde_user_passwords::qpidssl]
Updated data_bag_item[itso_sde_user_passwords::db2das1]
```

Important: You must run this command from the parent directory of the `data_bags` directory (in the cloud setup for this publication, this means `/root/cmo/itso-sde-deployment`).

7. Upload the modified environment file (env.json) to the Chef repository as shown in Example 4-88.

Example 4-88 Uploading the env.json file to the Chef repository

```
[root@cmodeploy itso-sde-deployment]# knife environment from file env.json
Updated Environment ITSO-SDE-CLOUD
```

Creating a node-specific attribute file

The node-specific attribute file is used to provide information to the Chef server that will only apply to a specific node (overwriting what is specified in the env.json file specifically for that node). If your Cloud Manager is managing exclusively PowerVM Compute nodes, you do not need to use the node-specific attribute file, and you can modify the parameters from this section directly in the environment file (env.json). If you have other nodes (for example KVM Compute nodes or PowerKVM Compute nodes), you cannot modify the env.json file directly, because the configurations provided in this section would affect the other nodes.

1. Log in to your deployment server and create a node-specific attribute file using a relevant name that allows you to identify that the file belongs to the PowerVC Driver (for example `env-powervc.json`):

```
# cd /<path_to_your_cloud_directory_structure>
# vi env-powervc.json
```

Important: Take a note of the file name and full path to the node-specific attribute file, because you will need to provide it in the “attribute_file” field in the topology (`topo.json`) file on “Update the topology file, topo.json”.

2. Provide attributes that are specific to the PowerVC driver, preserving the same brackets / braces structure from the `env.json` file. You need to modify the `openstack.block-storage.volume.create_volume_group` to `false` as shown in Example 4-89. The default cinder (block storage driver) configuration in the `env.json` file is to use LVM iSCSI to create volumes and provide to the instances, so you need to modify this configuration to allow the Cloud Manager to create volumes using the PowerVC driver and attach to your instances running in PowerVM Compute nodes.

Example 4-89 Modifying the create_volume_group attribute

```
{
      "block-storage": {
        "debug": false,
        "verbose": true,
        "volume": {
          "create_volume_group": false,
          "volume_group_size": 40,
          "iscsi_ip_address": "127.0.0.1"
        }
      }
}
```

Important: The node-specific attribute file should contain only the attribute lines specific to that node as shown in Example 4-89 on page 303. *Do not* copy all the content from the `env.json` file.

Note: If your Cloud Manager is managing PowerVC only hosts (PowerVM Compute node), this modification can be done directly in the `env.json` file. There are also additional configurations that you can modify to simplify your configuration. For more information, see the IBM Cloud Manager with OpenStack 4.1 Administrator Guide:

`http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html`

Update the topology file, topo.json

You need to modify your topology file to add the PowerVM Compute nodes to your OpenStack cloud. After completing this step, you will use the **`knife`** command to tell the deployment server to update your cloud topology, adding the PowerVM Compute nodes under the management of the Controller node.

Unlike the other nodes previously added (KVM and PowerKVM Compute nodes), you will not add the PowerVM Compute nodes directly in the topo.json file. You will add an additional role to the Controller node, allowing it to connect to the PowerVC server using the PowerVC driver.

1. Log in to your deployment server, and open the `topo.json` file that contains your cloud topology:

```
# cd /<path_to_your_cloud_directory_structure>
# vi topo.json
```

2. Modify your Controller node by adding the `ibm-os-powervc-driver-node` role and the `attribute_file` field, pointing to your node-specific attribute file, as shown in Example 4-90.

Example 4-90 Sample Controller node with PowerVC driver role

```
{
      "fqdn":"$Controller_node_FQDN",
      "password":"<root_password>",
      "attribute_file":"/path/to/node-specific-attribute-file.json",
      "quit_on_error":true,
      "run_order_number":1,
      "runlist": [
        "role[ibm-os-single-controller-node]",
        "role[ibm-os-powervc-driver-node]"
      ]
},
```

Important: Given that you are adding a new role, the previous role (`ibm-os-single-controller-node`) *must end with a comma*.

Every node entry should be placed between braces: { }

If there another node after the node you are modifying, the braces (or curly brackets) should have also a comma: { },

If this is the last node, the braces are not followed by a comma: { }.

Note: The node-specific attribute file specified in the `attribute_file` field must be the full path to the file created in step “Creating a node-specific attribute file” on page 302.

Example 4-91 shows the topo.json file used in the cloud setup for this publication to add the PowerVC driver role to the Controller node.

Example 4-91 JSON file used to add the PowerVC driver role to the Controller node

```
1 {
2   "name":"topo-ITSO-SDE-CLOUD",
3   "description":"topo-ITSO-SDE-CLOUD",
4   "environment":"ITSO-SDE-CLOUD",
5   "secret_file":"/root/cmo/itso-sde-deployment/itso-secret-key",
6   "run_sequentially":false,
7   "nodes": [
8     {
9       "fqdn":"cmocontroller.itso.ibm.com",
10       "password":"itsoadmin",
11       "attribute_file":"/root/cmo/itso-sde-deployment/env-powervc.json",
12       "quit_on_error":true,
13       "run_order_number":1,
14       "runlist": [
```

```
15          "role[ibm-os-single-controller-node]",
16          "role[ibm-os-powervc-driver-node]"
17        ]
18      },
19      {
20        "fqdn":"cmo1.itso.ibm.com",
21        "password":"itsoadmin",
22        "quit_on_error":true,
23        "run_order_number":2,
24        "runlist": [
25          "role[ibm-os-compute-node-kvm]"
26        ]
27      },
28      {
29        "fqdn":"cmo2.itso.ibm.com",
30        "password":"itsoadmin",
31        "quit_on_error":true,
32        "run_order_number":2,
33        "runlist": [
34          "role[ibm-os-compute-node-kvm]"
35        ]
36      },
37      {
38        "fqdn":"powerkvm1.itso.ibm.com",
39        "password":"itsoadmin",
40        "attribute_file":"/root/cmo/itso-sde-deployment/env-powekvm1.json",
41        "quit_on_error":true,
42        "run_order_number":2,
43        "runlist": [
44          "role[ibm-os-compute-node-powerkvm]"
45        ]
46      },
47      {
48        "fqdn":"powerkvm2.itso.ibm.com",
49        "password":"itsoadmin",
50        "attribute_file":"/root/cmo/itso-sde-deployment/env-powekvm2.json",
51        "quit_on_error":true,
52        "run_order_number":2,
53        "runlist": [
54          "role[ibm-os-compute-node-powerkvm]"
55        ]
56      }
57    ]
58  }
```

Customize the nova.conf file

There is an additional modification that must be made in the `/etc/nova/nova.conf` configuration file in the Controller node, which involves including an additional attribute to automatically confirm when an instance is resized.

- Instructions for PowerVC only environments:

  If you only have PowerVM Compute nodes in your environment, you can add this attribute directly in the `env.json` file in the deployment server before deploying your OpenStack cloud, by including the attribute resize_confirm_window=1 in the `openstack.compute.misc_nova` section as shown in Example 4-92 on page 306.

Example 4-92 Including the resize_confirm_window attribute using Chef

```
312         "misc_nova": [
313           "api_rate_limit=false",
314           "flat_injected=true",
315           "use_ipv6=true",
316           "resize_confirm_window=1"
317         ],
```

Important: This is only valid if you have exclusively PowerVM Compute nodes in your environment. Do not add this attribute to the env.json if you have a hybrid cloud. In the cloud prepared for this publication, this attribute was not added in the `env.json` file.

If you have only PowerVM Compute nodes and you are modifying the env.json, make sure to add a comma to the previous attribute, as you are adding one more attribute to the section.

- Instructions for hybrid environments:

  If you have other types of hypervisors in your OpenStack cloud, you need to manually modify the /etc/nova/nova.conf file in the Controller node:

  a. Log in to the Controller node and open the /etc/nova/nova.conf file:

     ```
     # vi /etc/nova/nova.conf
     ```

  b. Look for a section called `##### THIRD PARTY ADDITIONS #####` and add the `resize_confirm_window=1` entry after the `use_ipv6=true` as shown in Example 4-93.

     Example 4-93 Including the resize_confirm_window attribute in nova.conf

     ```
     ##### THIRD PARTY ADDITIONS #####

     api_rate_limit=false
     flat_injected=true
     use_ipv6=true
     resize_confirm_window=1
     ```

     It is important to complete this step before running the **`knife`** command to update your topology, because it will restart the nova services that will read the updated configuration in the `/etc/nova/nova.conf` file.

Use the knife command to update your cloud topology

Now you are ready to use the deployment server to add the PowerVM Compute nodes to your OpenStack cloud by connecting the Controller node to the PowerVC server using the PowerVC driver. In this step, the deployment server will use Chef server to install and configure additional packages in the Controller node and connect it to the PowerVC server using the credentials provided in the `env.json` file and the data bag. After completing this step, the Cloud Management dashboard will allow you to manage your PowerVM Compute nodes from the Cloud Manager.

1. Log in to your deployment server and access the directory where you have the env.json, topo.json and node-specific attribute files for the PowerVC Driver:

   ```
   # cd /root/cmo/itso-sde-deployment/
   ```

2. Use the **`knife`** command to load the modified env.json file to the Chef repository:

   ```
   # knife environment from file env.json
   Updated Environment ITSO-SDE-CLOUD
   ```

c. Given that the Controller node was previously deployed (on section 4.5, “Deploy the Controller node” on page 218), you need to use the **knife** command to apply directly to the Chef server the same modifications you made in the node-specific attribute file:

```
# knife node list
# knife node edit <Controller_node_FQDN>
```

Example 4-94 shows the command used in the cloud setup for this publication modify the attributes of the Controller node directly in the Chef repository.

Example 4-94 Using the knife node edit command

```
[root@cmodeploy itso-sde-deployment]# knife node list
cmo1.itso.ibm.com
cmo2.itso.ibm.com
cmocontroller.itso.ibm.com
powerkvm1.itso.ibm.com
powerkvm2.itso.ibm.com

[root@cmodeploy itso-sde-deployment]# knife node edit
cmocontroller.itso.ibm.com
```

Modify the `openstack.block-storage.volume.create_volume_group` to `false` as shown in Example 4-95.

Example 4-95 Changing the openstack.block-storage.volume.create_volume_group to false

```
"block-storage": {
      "debug": false,
      "verbose": true,
      "volume": {
        "create_volume_group": false,
        "volume_group_size": 40,
        "iscsi_ip_address": "127.0.0.1"
      }
},
```

d. Use the knife command to update the topology of your OpenStack cloud with the modified topo.json file:

```
# knife os manage update topology topo.json
```

The expected output is too long to fully show in this publication, but the Example 4-96 partially shows the expected output from the **knife os manage update topology topo.json** command for connecting the Controller node to the PowerVC by using the PowerVC driver.

Example 4-96 Sample output updating the cloud topology by adding PowerVM Compute nodes

```
[root@cmodeploy itso-sde-deployment]# knife os manage update topology
top-powervc.json
Updating topology 'topo-ITSO-SDE-CLOUD' ...
Updating nodes with run_order_number '1'
Updating node at cmocontroller.itso.ibm.com...
Setting environment for node cmocontroller.itso.ibm.com to 'ITSO-SDE-CLOUD'...
Setting run list for node cmocontroller.itso.ibm.com...
cmocontroller.itso.ibm.com:
  run_list:
    role[ibm-os-single-controller-node]
    role[ibm-os-powervc-driver-node]
```

```
Bootstrapping node ...
Connecting to cmocontroller.itso.ibm.com
cmocontroller.itso.ibm.com Chef client version on the node: 11.12.4
.
. <many lines were removed>
.
cmocontroller.itso.ibm.com Running handlers:
cmocontroller.itso.ibm.com Running handlers complete
cmocontroller.itso.ibm.com
cmocontroller.itso.ibm.com Chef Client finished, 46/468 resources updated in
196.530152054 seconds
All nodes with run_order_number '1' updated.

Results for update of topology 'topo-ITSO-SDE-CLOUD'
Results for nodes with run_order_number '1'
Update of node at cmocontroller.itso.ibm.com was successful.
Update of topology 'top-powervc.json' completed in 200 seconds.
```

Note: You can also use the **knife os manage deploy topology topo.json** command in this task, but it will run unnecessary commands in the Controller node and Compute nodes, trying to redeploy what was already deployed in the previous sections, taking longer to complete.

In case you need to use the **deploy**, rather than **update** option for a specific reason, you should create a separate topology file specifically to the node which you plan to modify, and use this file as input file in the **knife** command.

Important: When you use a node-specific attribute file and needs to modify the topology for some reason (after the topology has already been deployed), the **knife os manage update topology topo.json** command will not read the node-specific attribute file. Besides modifying the file to keep track of the changes as done in step 2 on page 303, you should use the command **knife node edit <node_name_fqdn>** as used in step c on page 307. This will bring the node-specific attribute file loaded in the Chef repository, so you can modify it directly in the Chef repository.

e. Log in to the Controller node and run the **nova hypervisor-status-list** command to verify if the PowerVM Compute nodes were added to your OpenStack cloud:

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# nova hypervisor-status-list
+----+-----------------------+----------------+-----------+-----------------
--+
| id | hypervisor_hostname    | hypervisor_type | ha_status |
maintenance_status |
+----+-----------------------+----------------+-----------+-----------------
--+
| 1  | cmo1.itso.ibm.com      | QEMU            | ok        | ok
|
| 3  | cmo2.itso.ibm.com      | QEMU            | ok        | ok
|
| 21 | powerkvm1.itso.ibm.com | QEMU            | ok        | ok
|
| 22 | powerkvm2.itso.ibm.com | QEMU            | ok        | ok
|
```

```
| 41 | 8205E6C_06A22ER        | powervm         | ok        | ok
|
| 61 | 8233E8B_061AA6P        | powervm         | ok        | ok
|
+----+------------------------+-----------------+-----------+-----------------
--+
```

Note: It is required to source the `/root/openrc` file before running the command, to provide the `OS_USERNAME` and `OS_PASSWORD` variables to the OpenStack commands.

f. Verify if the following services are running in the Controller node:

```
# service openstack-cinder-powervc status
openstack-cinder-powervc (pid  10594) is running...
# service openstack-glance-powervc status
openstack-glance-powervc (pid  23068) is running...
# service openstack-neutron-powervc status
openstack-neutron-powervc (pid  10400) is running...
# service openstack-nova-powervc status
openstack-nova-powervc (pid  10365) is running...
```

g. Log in to the Cloud Management dashboard using the Controller node IP address and click **Admin > Hypervisors** as shown in Figure 4-48 on page 310.

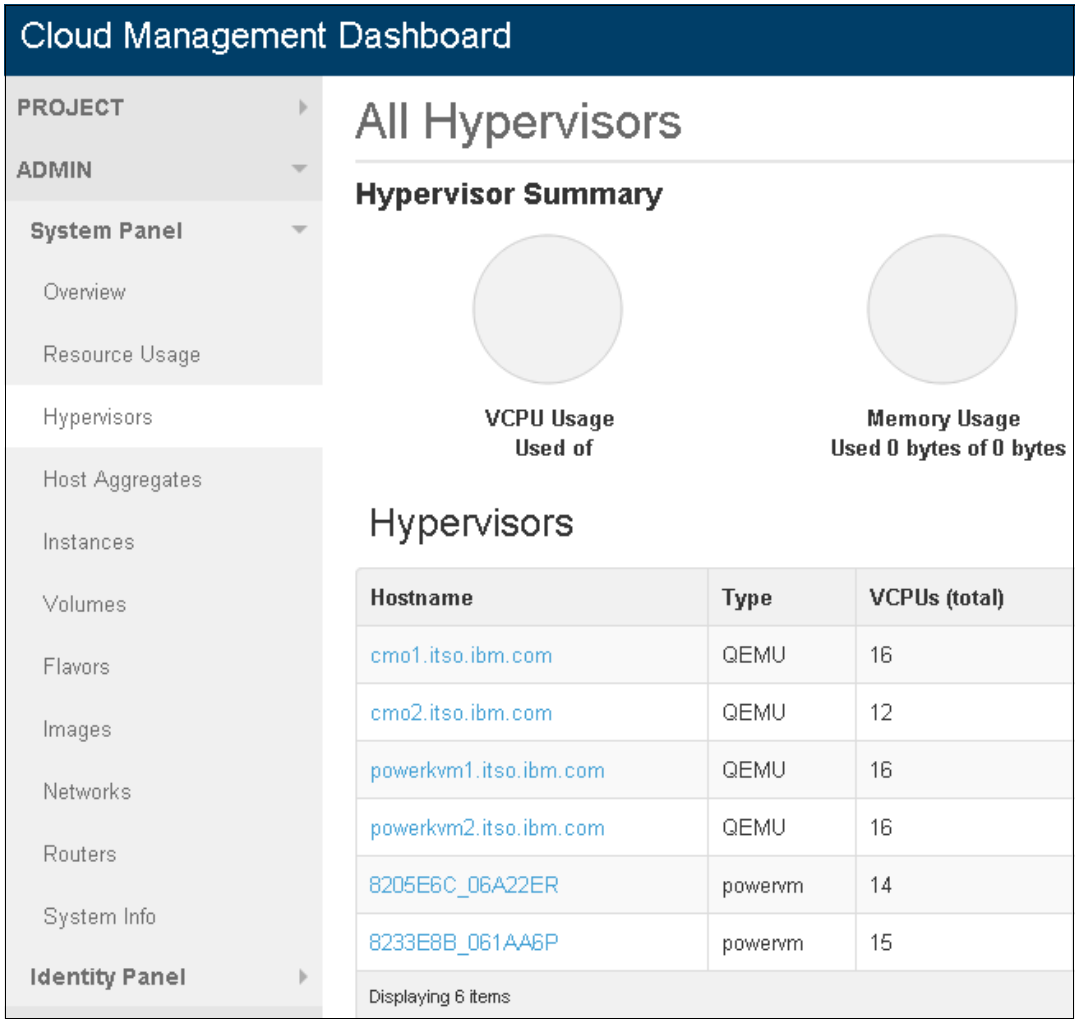


Figure 4-48 PowerVM Compute nodes in the Cloud Management dashboard

3. The deployment process will discover your instances (virtual machines) running in the PowerVM Compute nodes and populate the **Admin > Instances** section, as shown in Figure 4-49.

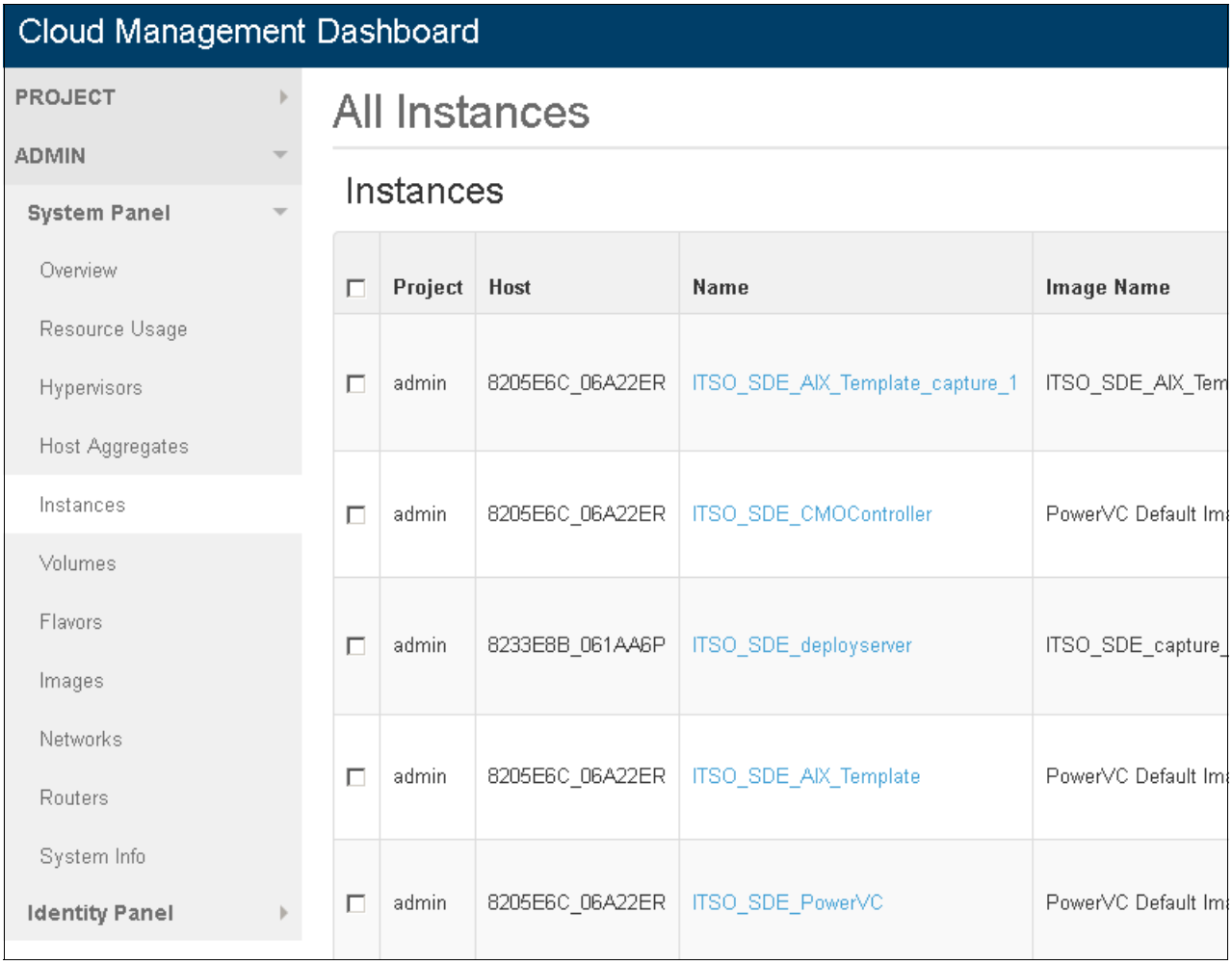


Figure 4-49 PowerVM instances from the Cloud Management dashboard

4. It will also discover information from the following locations:
   - Volumes from these instances: **Admin > System Panel > Volumes**
   - Additional flavors from PowerVC: **Admin > System Panel > Flavors**
   - Images (in case you have any image created in PowerVC): **Admin > System Panel > Images**.

4.9 Enable the self-service portal

The self-service management portal is an optional feature of the Cloud Manager as an alternative to the Cloud Management dashboard. You can use the easy-to-use web interface to manage to perform several operations on your cloud:

- Deploy and terminate instances on your cloud
- Provide access to multiple clouds from a single portal (in case you have more than one Cloud Manager cloud in your environment)

- Start and stop instances
- Resize existing instances
- Create projects to give team-specific access to instances
- Manage network configuration

Note: The self-service portal is also referred to as Smart Cloud Entry web interface.

For more information about the self-service portal, see the “Self-service management portal (optional)” page in the IBM Knowledge Center:

http://ibm.co/1Gud6rp

Important: It is possible to install the self-service portal on a separate server or virtual machine running Microsoft Windows. This publication will provide instructions on how to enable the self-service portal in the Controller node. If you plan to install self-service portal on Microsoft Windows, see the IBM Cloud Manager with OpenStack 4.1 Administrator Guide available at:

http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html

During the process of enabling the self-service portal, you will modify the input files in your deployment server (`env.json` and `topo.json`) and use the **`knife`** command to tell the Chef server to install and configure specific packages in the Controller node. To complete this task, the following steps are required:

1. Prepare the environment file (env.json).
2. Prepare the topology file (topo.json).
3. Use the **`knife`** command to update your cloud topology.
4. Connect the self-service portal to your OpenStack cloud.
5. Install IBM Cloud Manager License.

4.9.1 Prepare the environment (env.json) file

A simple specific modification is required in the `env.json` file to enable the self-service portal. This section will guide you to perform such change.

1. Log in to your deployment server and open the env.json file that contains the environment definitions to your cloud:

   ```
   # cd /<path_to_your_cloud_directory_structure>
   # vi env.json
   ```

2. Set the `ibm-sce.service.enabled` attribute to `true`. This attribute modification will result in the following actions:

   a. Replace the default OpenStack policy files with the updated self-service portal policy files.

   b. Register the `sceagent` user with the `admin` role in the OpenStack cloud.

Example 4-97 on page 313 shows the modified parameter in the env.json file.

Example 4-97 Enabling the self-service portal in the env.json file

```
349     "ibm-sce": {
350       "os": {
351         "user": "sce",
352         "group": "sce"
353       },
354       "config": {
355         "prop": {
356           "silent": "1"
357         }
358       },
359       "user": {
360         "input": {
361           "authentication": {
362             "username": "admin",
363             "name": "Administrator"
364           }
365         },
366         "install": {
367           "folder": "/opt/ibm"
368         },
369         "prop": {
370           "folder": "/var/opt/ibm"
371         },
372         "shortcuts": "/root"
373       },
374       "choose": {
375         "license": {
376           "type": {
377             "boolean": "0"
378           }
379         }
380       },
381       "license": {
382         "path": ""
383       },
384       "service": {
385         "enabled": true
386       },
```

4.9.2 Prepare the topology (topo.json) file

You need to modify your topology file to provide an additional role to your Controller node, allowing it to be your self-service portal node. After completing this step, you will use the **knife** command to tell the deployment server to update your cloud topology, installing and configuring the self-service portal packages in the Controller node and starting its services.

1. Log in to your deployment server and open the `topo.json` file, that contains your cloud topology:

   ```
   # cd /<path_to_your_cloud_directory_structure>
   # vi topo.json
   ```

2. Modify your Controller node, by adding the `ibm-sce-node` role. If your Controller node is also using the PowerVC driver to connect to a PowerVC server, the Controller node section should be similar to Example 4-98.

Example 4-98 Adding the ibm-sce-node role to your Controller node

```
{
      "fqdn":"$Controller_node_FQDN",
      "password":"<root_password>",
      "attribute_file":"/path/to/node-specific-attribute-file.json",
      "quit_on_error":true,
      "run_order_number":1,
      "runlist": [
        "role[ibm-os-single-controller-node]",
        "role[ibm-os-powervc-driver-node]",
        "role[ibm-sce-node]"
      ]
},
```

Note: If your Controller is not connected to a PowerVC server to manage PowerVM Compute nodes, you do not need the `ibm-os-powervc-driver-node` role and the `attribute_file` parameter.

Important: Given that you are adding a new role, the previous role (ibm-os-single-controller-node) must end with a comma.

Every node entry should be placed between braces: `{ }`

If there is another node after the node you are modifying, the braces (or curly brackets) should have also a comma: `{ },`

If this is the last node, the braces are not followed by a comma: `{ }`

Example 4-99 shows the `topo.json` file used in this publication modified to enable the self-service portal.

Example 4-99 topo.json file used in this publication to add self-service portal

```
1 {
2   "name":"topo-ITSO-SDE-CLOUD",
3   "description":"topo-ITSO-SDE-CLOUD",
4   "environment":"ITSO-SDE-CLOUD",
5   "secret_file":"/root/cmo/itso-sde-deployment/itso-secret-key",
6   "run_sequentially":false,
7   "nodes": [
8     {
9       "fqdn":"cmocontroller.itso.ibm.com",
10       "password":"itsoadmin",
11       "attribute_file":"/root/cmo/itso-sde-deployment/env-powervc.json",
12       "quit_on_error":true,
13       "run_order_number":1,
14       "runlist": [
15         "role[ibm-os-single-controller-node]",
16         "role[ibm-os-powervc-driver-node]",
17         "role[ibm-sce-node]"
```

```
18        ]
19      },
20      {
21        "fqdn":"cmo1.itso.ibm.com",
22        "password":"itsoadmin",
23        "quit_on_error":true,
24        "run_order_number":2,
25        "runlist": [
26          "role[ibm-os-compute-node-kvm]"
27        ]
28      },
29      {
30        "fqdn":"cmo2.itso.ibm.com",
31        "password":"itsoadmin",
32        "quit_on_error":true,
33        "run_order_number":2,
34        "runlist": [
35          "role[ibm-os-compute-node-kvm]"
36        ]
37      },
38      {
39        "fqdn":"powerkvm1.itso.ibm.com",
40        "password":"itsoadmin",
41        "attribute_file":"/root/cmo/itso-sde-deployment/env-powekvm1.json",
42        "quit_on_error":true,
43        "run_order_number":1,
44        "runlist": [
45          "role[ibm-os-compute-node-powerkvm]"
46        ]
47      },
48      {
49        "fqdn":"powerkvm2.itso.ibm.com",
50        "password":"itsoadmin",
51        "attribute_file":"/root/cmo/itso-sde-deployment/env-powekvm2.json",
52        "quit_on_error":true,
53        "run_order_number":1,
54        "runlist": [
55          "role[ibm-os-compute-node-powerkvm]"
56        ]
57      }
58    ]
59  }
```

4.9.3 Use the knife command to update your cloud topology

Now you are ready to use the deployment server to install, configure and start the self-service portal in the Controller node. After completing this step, you will be able to access the self-service portal web interface in the Controller node and connect it to your OpenStack cloud.

1. Log in to your deployment server and access the directory where you have the `env.json` and `topo.json` files:

   ```
   # cd /root/cmo/itso-sde-deployment/
   ```

2. Use the **knife** command to load the modified env.json file to the Chef repository:

```
# knife environment from file env.json
Updated Environment ITSO-SDE-CLOUD
```

3. Use the **knife** command to update your OpenStack cloud topology with the modified topo.json file:

```
# knife os manage update topology topo.json
```

The expected output is too long to fully show in this publication, but the Example 4-100 shows partially the expected output from the **knife os manage update topology topo.json** command for enabling the self-service portal in the Controller node.

Example 4-100 Sample output updating the topology by enabling the self-service portal

```
[root@cmodeploy itso-sde-deployment]# knife os manage update topology
top-powervc.json
Updating topology 'topo-ITSO-SDE-CLOUD' ...
Updating nodes with run_order_number '1'
Updating node at cmocontroller.itso.ibm.com...
Setting environment for node cmocontroller.itso.ibm.com to 'ITSO-SDE-CLOUD'...
Setting run list for node cmocontroller.itso.ibm.com...
cmocontroller.itso.ibm.com:
  run_list:
    role[ibm-os-single-controller-node]
    role[ibm-os-powervc-driver-node]
    role[ibm-sce-node]
Bootstrapping node ...
Connecting to cmocontroller.itso.ibm.com
cmocontroller.itso.ibm.com Chef client version on the node: 11.12.4
cmocontroller.itso.ibm.com Chef bootstrap version: 11.12.4
.
. <many lines were removed>
.
cmocontroller.itso.ibm.com Running handlers:
cmocontroller.itso.ibm.com Running handlers complete
cmocontroller.itso.ibm.com
cmocontroller.itso.ibm.com Chef Client finished, 96/518 resources updated in
320.18458355 seconds
All nodes with run_order_number '1' updated.

Results for update of topology 'topo-ITSO-SDE-CLOUD'
Results for nodes with run_order_number '1'
Update of node at cmocontroller.itso.ibm.com was successful.
Update of topology 'top-powervc.json' completed in 324 seconds.
```

4.9.4 Connect the self-service portal to your OpenStack cloud

At this point, the self-service portal is installed and running in your Controller node, but you still need to connect it to your existing cloud. The following steps will guide you to complete this task:

1. Open a browser and use the following URL to access the self-service portal in your Controller node:

```
http://<Controller_node_IP_Address>:18080/cloud/web/login.html
```

Important: Notice that this connection uses `http` rather than `https`.

2. Log in by using `admin` as user ID and the same password you use to log in to the Cloud Management dashboard, as shown in Figure 4-50.

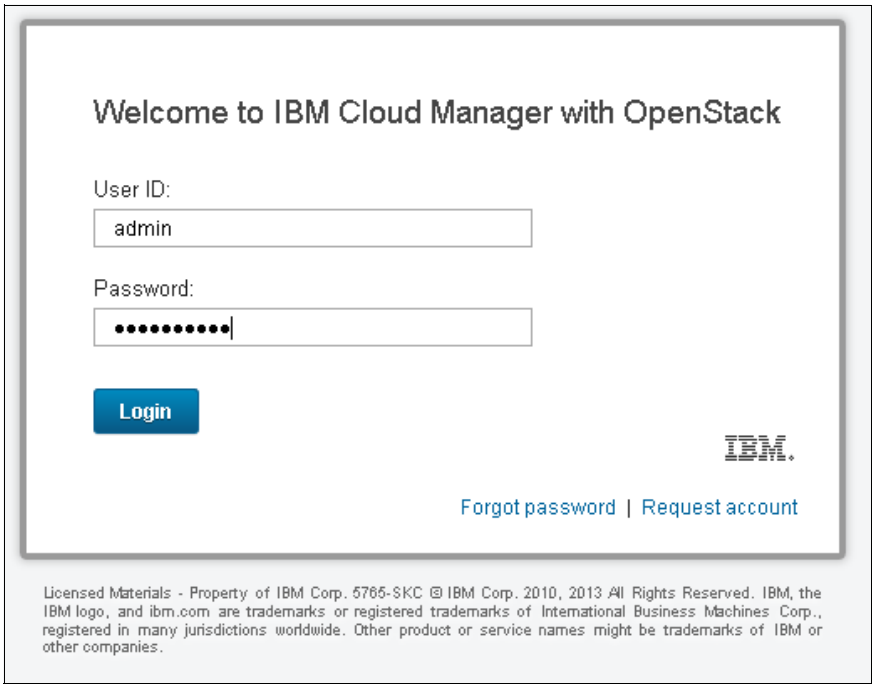


Figure 4-50 Self-service portal login interface

Note: Usually, the default admin password is openstack1. You can find this password in the `/root/openrc` file in the Controller node:

```
[root@cmocontroller ~]# cat /root/openrc

# This file autogenerated by Chef
# Do not edit, changes will be overwritten


# COMMON OPENSTACK ENVS
export OS_USERNAME=admin
export OS_PASSWORD=openstack1
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://172.16.21.137:5000/v2.0
export OS_REGION_NAME=RegionOne
```

3. Click **Configure the Cloud**, as shown in Figure 4-51.

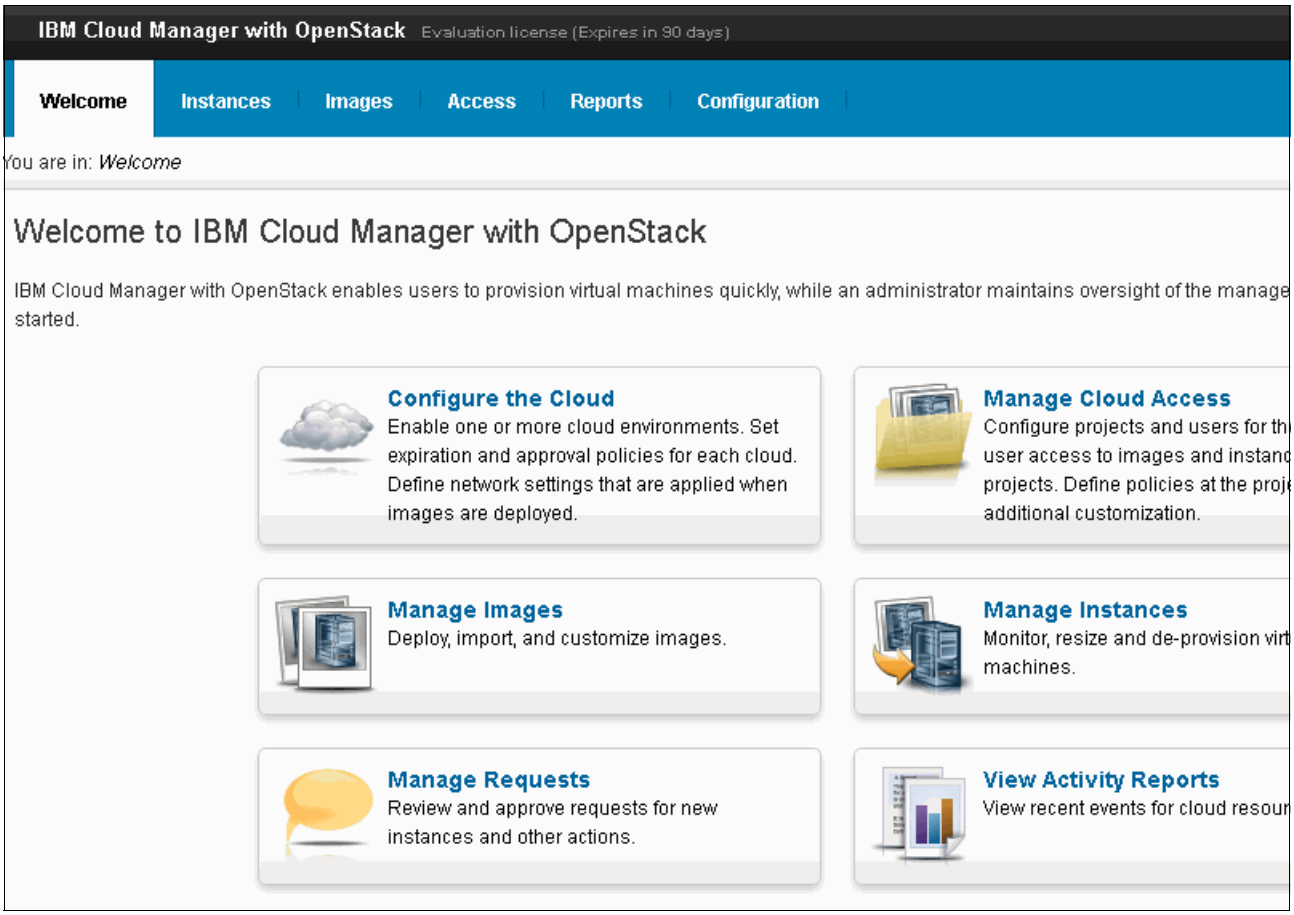


Figure 4-51 Configure the Cloud option in self-service portal

4. Click `Setup the IaaS Gateway` as shown in Figure 4-52.

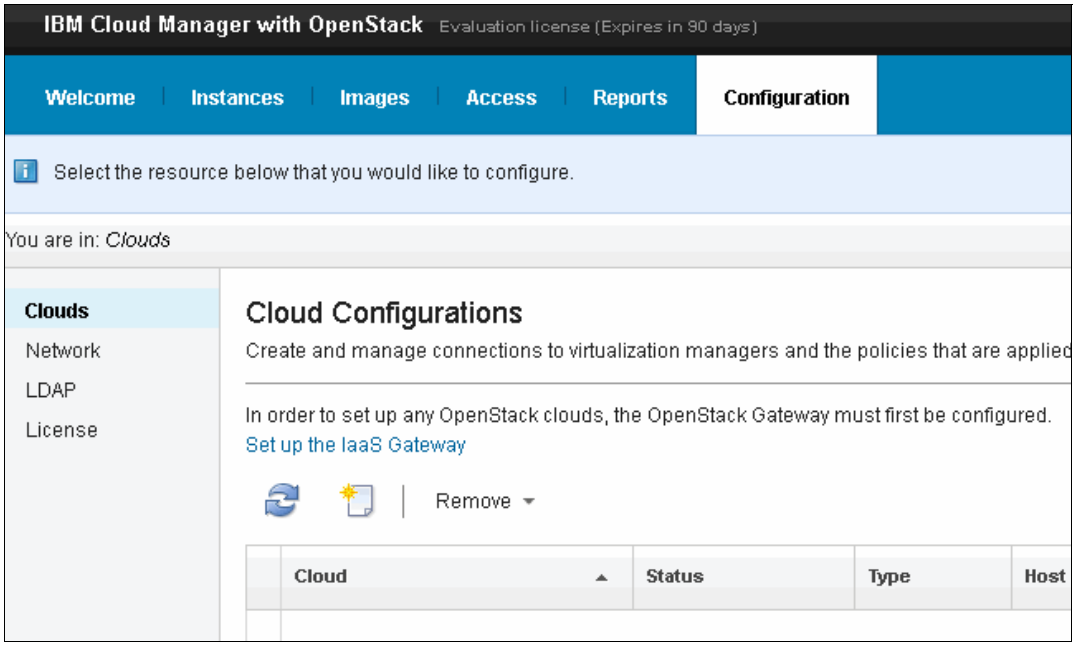


Figure 4-52 Click Setup the IaaS Gateway

5. In the Hostname field, provide the fully qualified domain name of your Controller node. The Administrator ID will be `sceagent` and the default sceagent password is `openstack1` (unless you have changed this password in the data bags). The Port field will be automatically filled with the port number 9973 (you should not change it).

   Click **Test Connection**. If you get a green check mark with the message “Connection attempt for OpenStack Gateway configuration <Controller_node_FQDN> completed successfully,” as shown in Figure 4-53 on page 320, you can click **Add**.

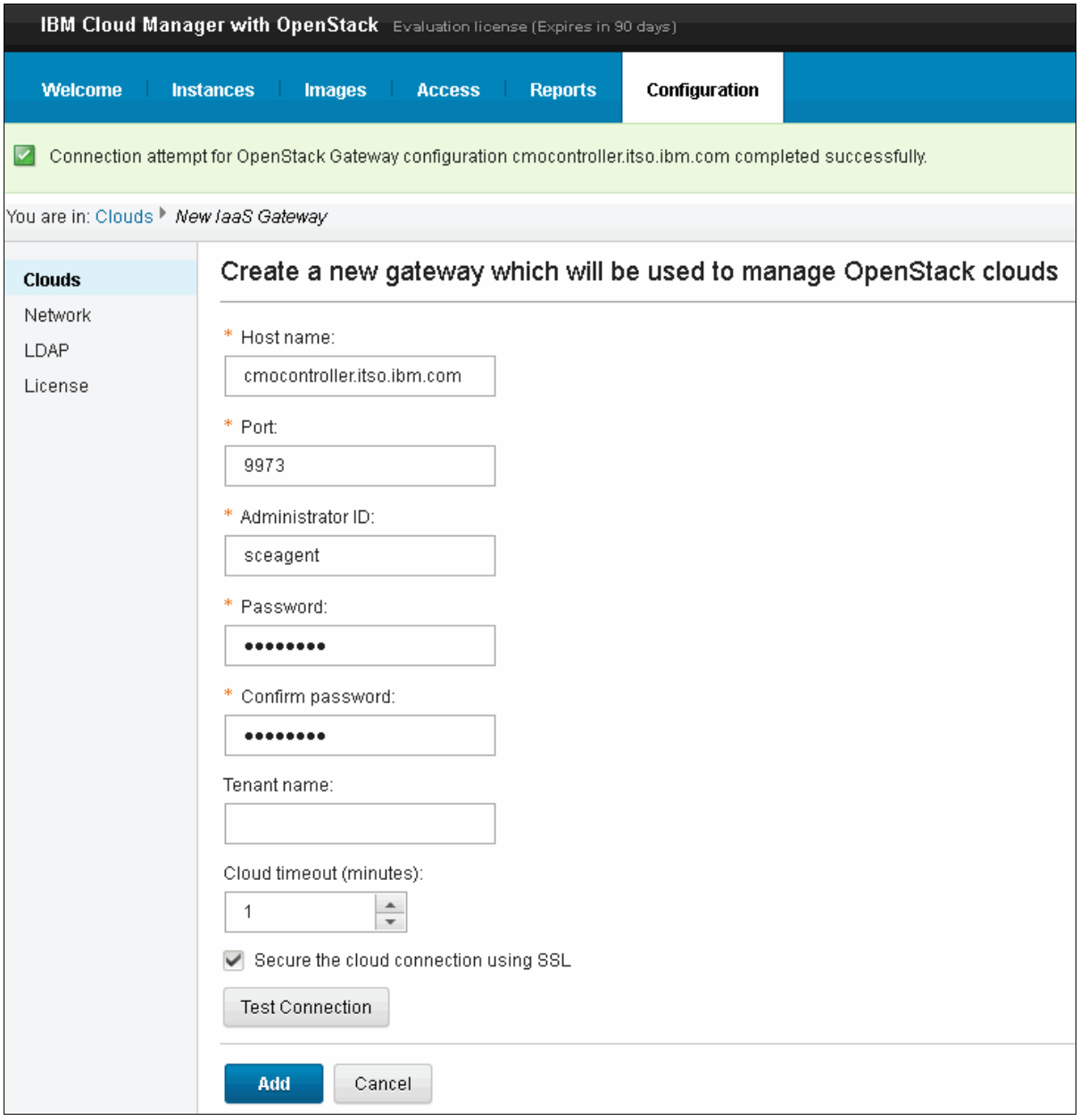


Figure 4-53 Providing the IaaS Gateway credentials

Important: The sceagent password can be found in the `user_passwords_data_bag`:

```
[root@cmodeploy ~]# cat
/root/cmo/itso-sde-deployment/data_bags/itso_sde_user_passwords/sceagent.json
{
  "id": "sceagent",
  "sceagent": "openstack1"
}
```

6. Click **Accept Certificate**, as shown in Figure 4-54.

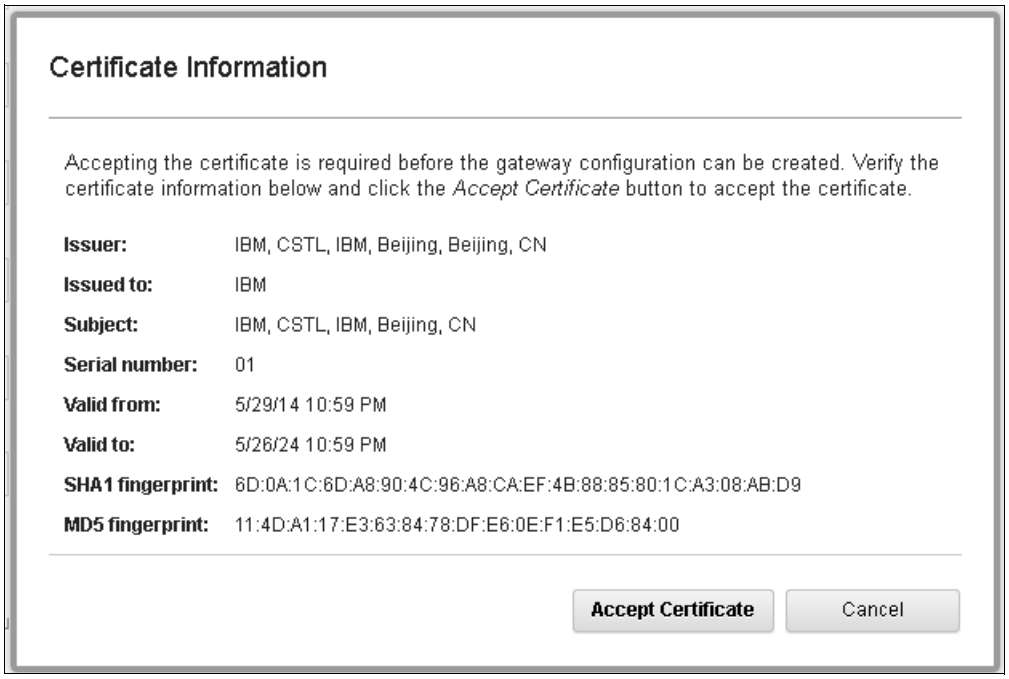


Figure 4-54 Accepting the certificate

7. Now click the `Create a new cloud configuration` button as shown in Figure 4-55.

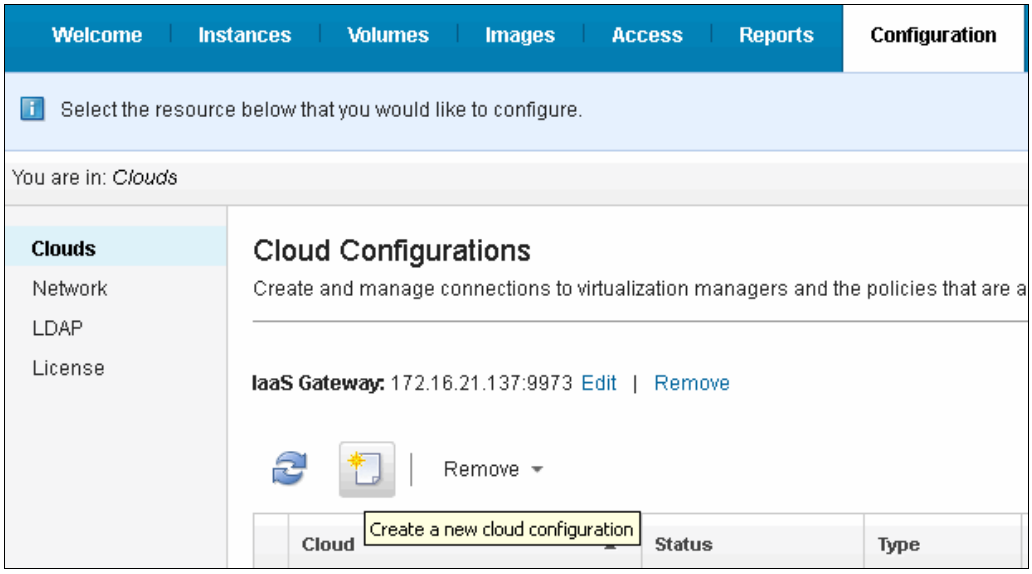


Figure 4-55 Create a new cloud configuration

8. Provide a name and a description (optional) for your cloud, select `OpenStack` as the `Type`, select the region (which you can obtain from your `/root/openrc` file in the Controller node). In the Qpid Settings section, provide the Controller node fully qualified domain name in the `Hostname` field, keep the port configuration as is (`5671`), preserve the `Secure the cloud connection using SSL` checked and provide the Apache Qpid user ID and password, which is usually `qpidclient` as user ID and `openstack1` as password (the Qpid client user ID and password can also be obtained from the data bags). Click `Test`

Connection and, if you receive a green check mark with the message Connection attempt for cloud configuration <your_cloud_name> completed successfully as shown in Figure 4-56 on page 323, you can click Add.

Note: You can get the Region from the /root/openrc file in the Controller node:

```
[root@cmocontroller ~]# cat /root/openrc
# This file autogenerated by Chef
# Do not edit, changes will be overwritten

# COMMON OPENSTACK ENVS
export OS_USERNAME=admin
export OS_PASSWORD=openstack1
export OS_TENANT_NAME=Public
export OS_AUTH_URL=http://172.16.21.137:5000/v2.0
export OS_REGION_NAME=RegionOne
```

Note: You can get the Qpid user ID and password from the user_passwords_data_bag in the deployment server:

```
[root@cmodeploy ~]# cat
/root/cmo/itso-sde-deployment/data_bags/itso_sde_user_passwords/qpidclient.j
son
{
  "id": "qpidclient",
  "qpidclient": "openstack1"
}
```

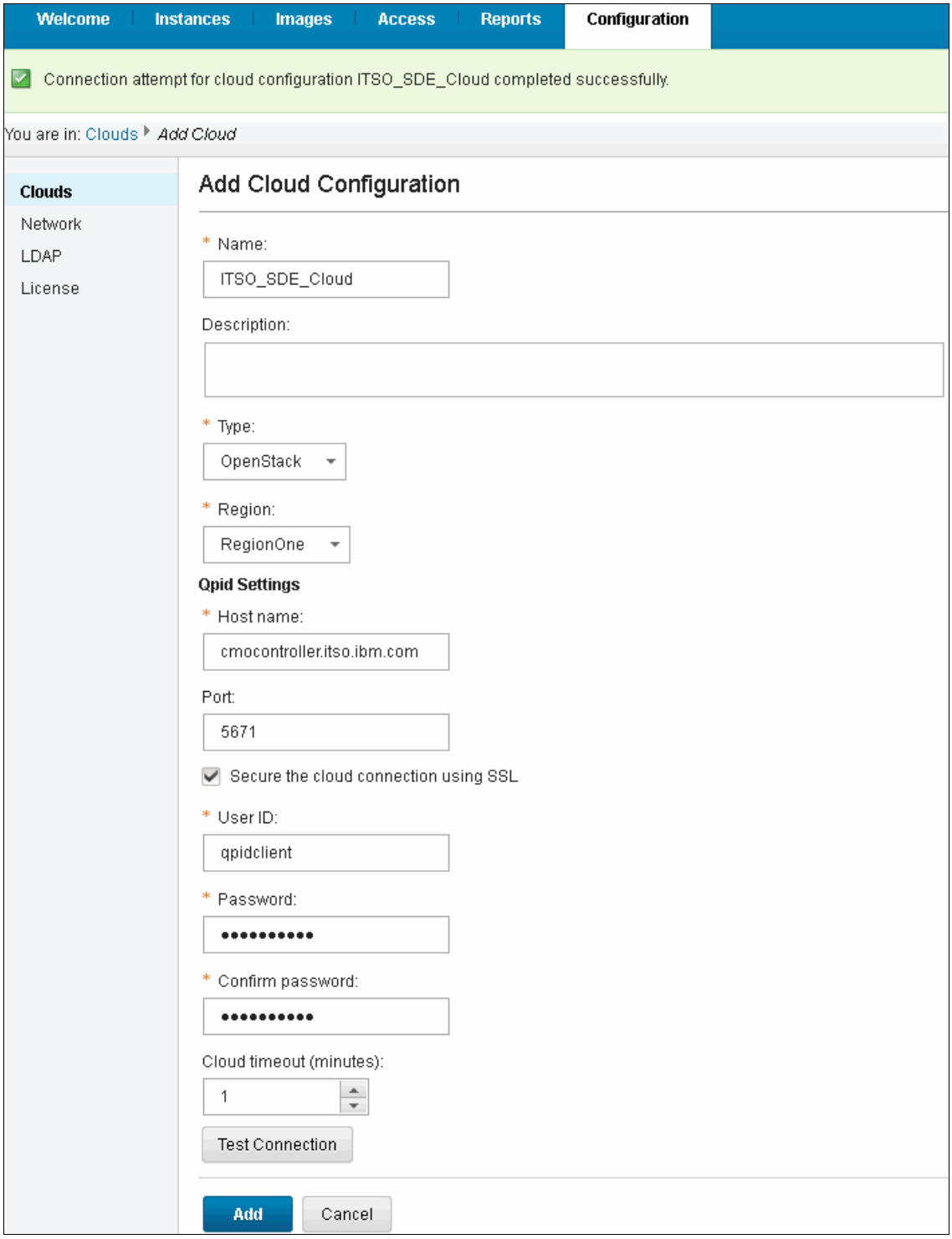


Figure 4-56 Adding your cloud to the self-service portal

9. Accept the certificate, as shown in Figure 4-57.

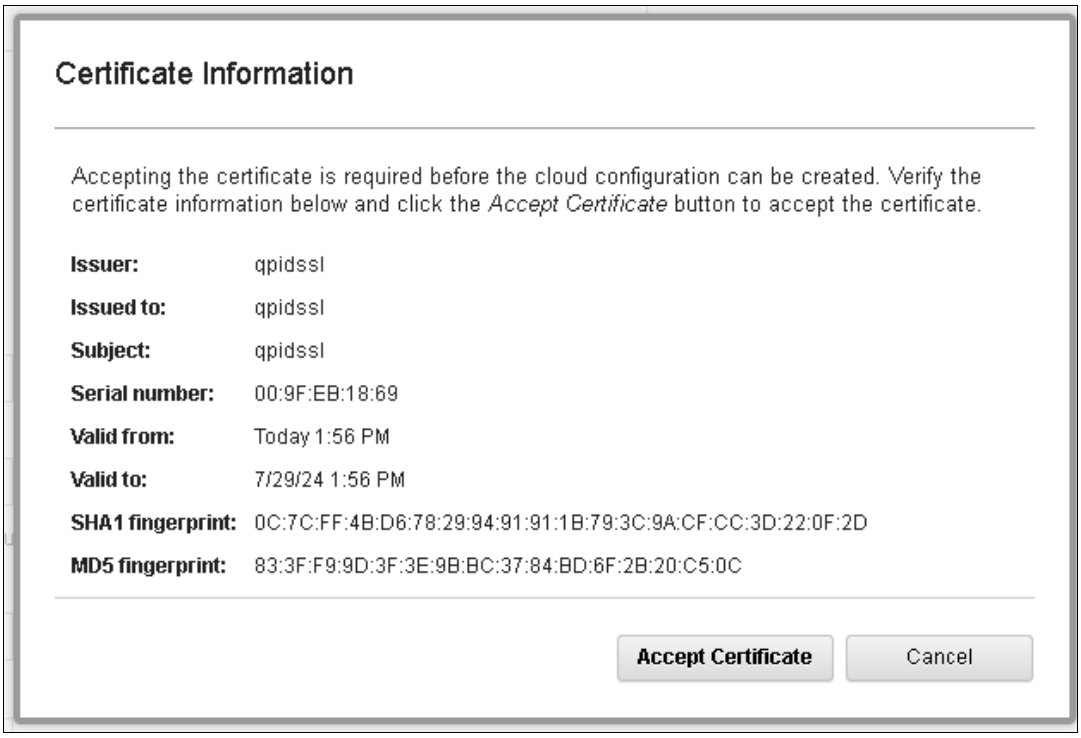


Figure 4-57 Accepting the certificate to add the cloud to the self-service portal

After completing these steps, you should be able to manage your instances, volumes, and images through the self-service portal, as shown in Figure 4-58.

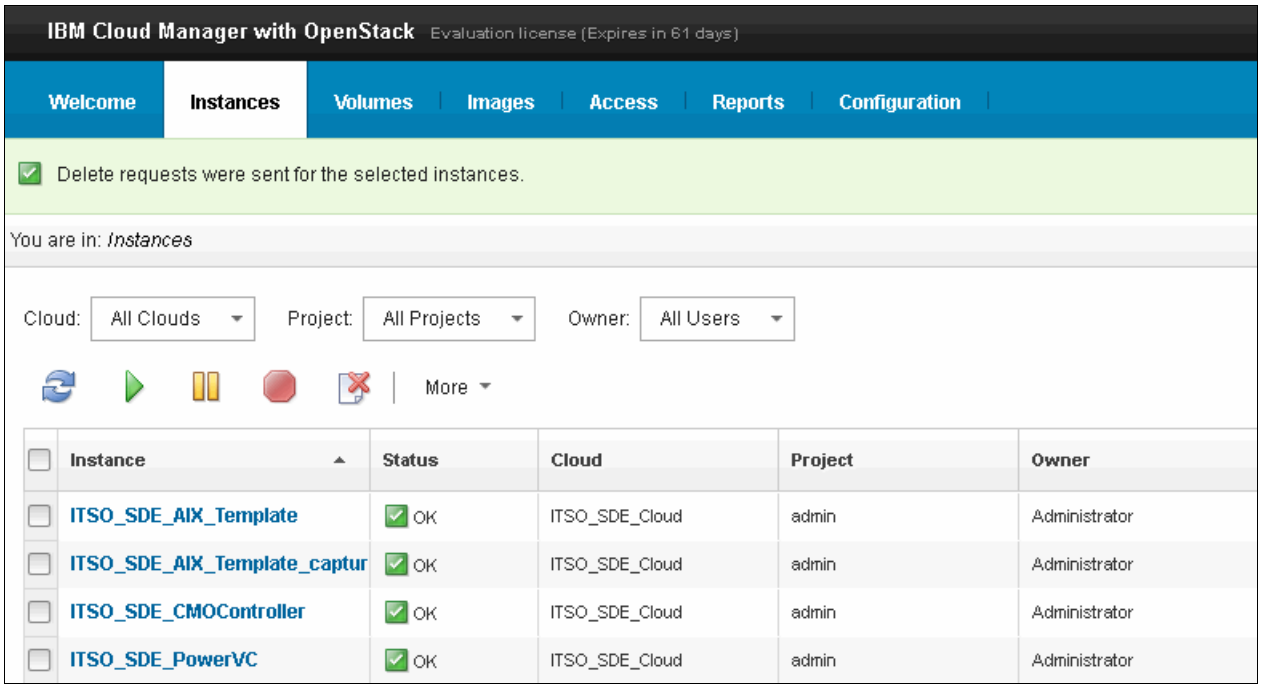


Figure 4-58 Self-service portal interface

4.9.5 Install the IBM Cloud Manager license

To complete the self-service portal, install the Cloud Manager license.

1. Obtain your license file (usually `cmwo_4.1.lic`) from your Sales Representative and copy the file to your workstation.
2. Log in to the self-service portal web interface at the following address:

   `http://<Controller_node_IP_Address>:18080/cloud/web/login.html`
3. Authenticate by using the admin user ID and appropriate password.
4. Click **Configure the Cloud**, as shown in Figure 4-51 on page 318.
5. Click **License** and then **Update License**, as shown in Figure 4-59.

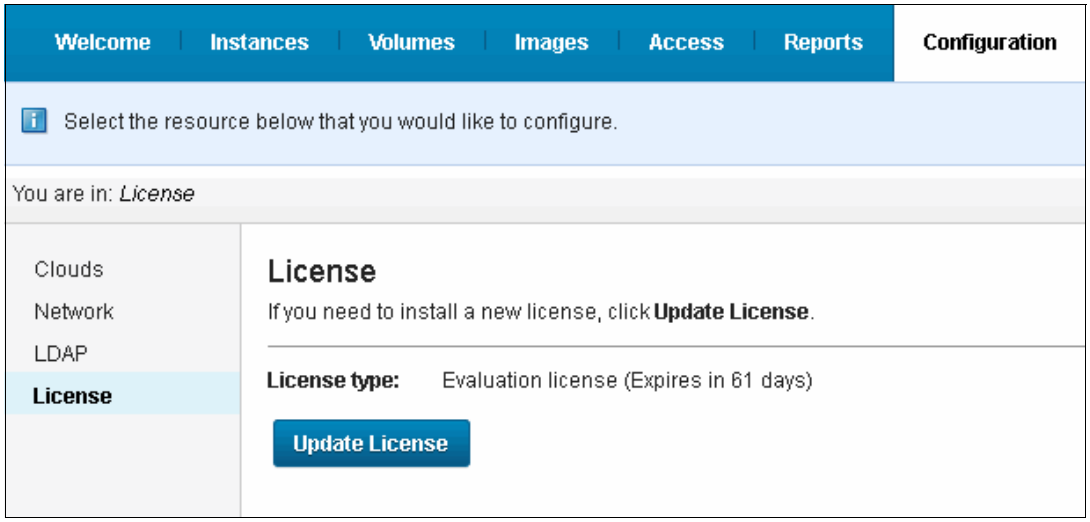


Figure 4-59 Updating the IBM Cloud Manager license

6. Click **Browse**, select the license file obtained in step 1, and then click **Apply**, as shown in Figure 4-60.



Figure 4-60 Applying the license file

Note: An evaluation license valid for 90 days is available from the deployment server:
`/opt/ibm/cmwo/properties/cmwo_4.1.lic`

4.10 Network configuration in IBM Cloud Manager

The OpenStack component that handles network communication is called `neutron`. Additional network setup is required in the Cloud Manager to allow your instances to connect to external network. During this setup you will create the network in the Cloud Management dashboard and relate this network with the Open vSwitch network bridge (in KVM and PowerKVM Compute nodes) and Shared Ethernet Adapter (in the PowerVM Compute nodes). Before starting this setup, you need to understand that there are four network types that you can create and configure in Cloud Manager, but restrictions applies according to the hypervisors that are running in your cloud. Consider reviewing the information from section 4.5.5, “Network considerations and requirements” on page 219 before proceeding.

The network types available in Cloud Manager are GRE, local, flat, and VLAN. For more details about each network type, see 4.5.5, “Network considerations and requirements” on page 219. This publication will provide instructions for flat and VLAN network configuration.

Note: If you have PowerVC driver connected to PowerVC server in your Controller node (to manage PowerVM Compute nodes), the only possible network type is VLAN network. In the cloud setup for this publication VLAN network was used.

Most of the network configuration required in your Compute nodes was already done by Chef during deployment steps in sections 4.5, “Deploy the Controller node” on page 218, 4.6, “Add KVM Compute nodes to the cloud” on page 236, 4.7, “Add PowerKVM Compute nodes to the cloud” on page 253 and 4.8, “Add PowerVM Compute nodes to the cloud” on page 274. During these steps, the deployment server has used Chef to prepare the Open vSwitch bridges in the Controller node, KVM and PowerKVM Compute nodes to allow your instances to connect to the network, and you have manually setup the Shared Ethernet Adapter in the PowerVM Compute nodes. The following topics describe these technologies:

- “Open vSwitch” on page 327
- “Shared Ethernet Adapters” on page 329

The 4.10.3, “Network configuration overview” on page 330” explains how the configurations made in the `env.json`, `topo.json`, and node-specific attribute files were deployed in the Controller and Compute nodes.

After completing this step, the next topics will help you to configure network in Cloud Manager:

- “Create networks in IBM Cloud Manager” on page 339
  - “Create flat networks” on page 339
  - “Create VLAN networks” on page 344
  - “Modify networks imported from PowerVC server” on page 347

For a better understanding of the cloud setup used in this publication, consider reviewing:

- “Network setup in the cloud prepared for this publication” on page 350

4.10.1 Open vSwitch

Open vSwitch is an open source, multilayer distributed virtual switch that supports standard management protocols and interfaces. It can be distributed across several physical machines. Open vSwitch has APIs for automation, which provide an excellent integration clouds.

IBM Cloud Manager uses Open vSwitch networks by default for Controller node, KVM and PowerKVM Compute nodes. Other bridges are allowed, but they are not covered by this publication. For more information about other possible network bridges configuration, see IBM Cloud Manager with OpenStack 4.1 Administrator Guide available at:

http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html

The Open vSwitch package was automatically installed by Chef server in the Controller node, KVM and PowerKVM Compute nodes during the deployment steps. You can see the package installed by using the **rpm** command, as shown in Example 4-101.

Example 4-101 Checking if openvswitch is installed

```
[root@cmocontroller ~]# rpm -q openvswitch
openvswitch-2.0.0-5.ibm.el6.ppc64
```

An Open vSwitch network bridge was also already configured by the Chef server during the deployment process, according to the configuration you provided in the `env.json` and node-specific attribute files. For more information about how the configuration from the `env.json` and node-specific attribute file affected the network bridges configuration, see section 4.10.3, “Network configuration overview” on page 330.

Figure 4-61 on page 328 briefly shows the topology in a Compute node with Open vSwitch configured, allowing you to understand how the network packets flow between the instance (virtual machine) and the external network.

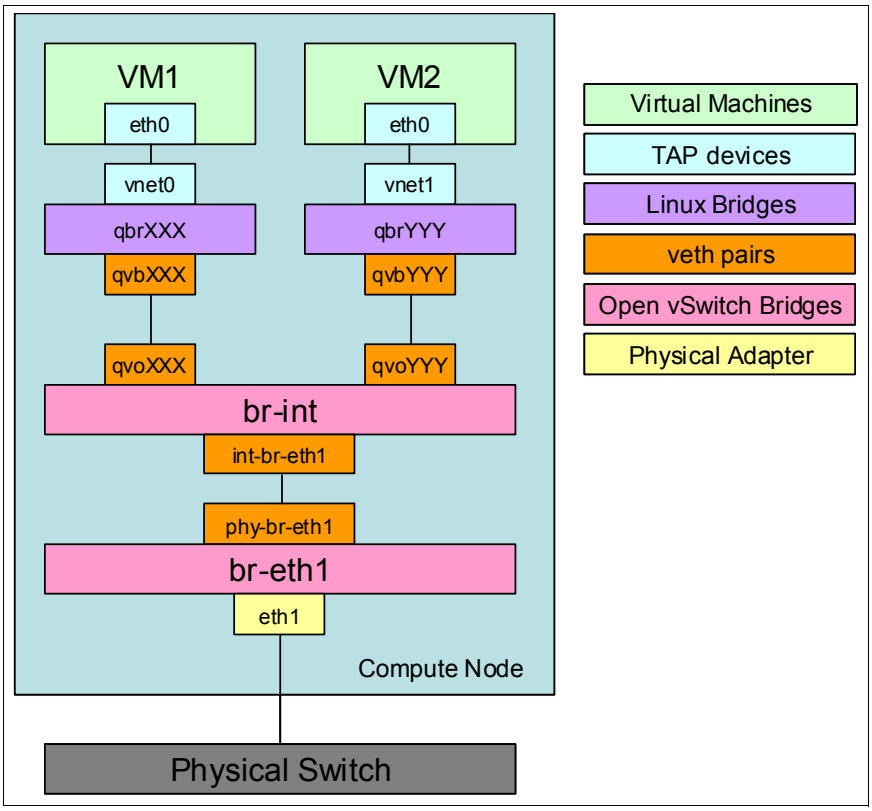


Figure 4-61 Open vSwitch topology

In this illustration, you can see four different virtual networking devices: TAP devices, veth pairs, Linux bridges and Open vSwitch bridges. For a network packet to go from the instance to the external network, it needs to cross nine devices inside the Compute node (TAP device `vnet0`, Linux bridge `qbrXXX`, veth pair `qvbXXX` and `qvoXXX`, Open vSwitch bridge `br-int`, veth pair `int-br-eth1` and `phy-br-eth1`, open vSwitch bridge `br-eth1` and physical adapter `eth1`).

The following components are involved in this communication:

TAP devices A *TAP device* is how the Compute node (PowerKVM or KVM) implements a virtual network interface. A packet sent to the TAP device is received by the virtual machine.

veth pairs A *veth pair* is a pair of virtual network interfaces directly connected together. The OpenStack network uses veth pairs as virtual patch cables to create connections between bridges.

Linux bridges A Linux bridge works similarly to a hub. You can connect multiple network interfaces to a Linux bridge, and the packets received in the bridge will be transmitted to all components connected to it.

Open vSwitch bridges The open vSwitch bridges work corectly as a physical switch, allowing multiple ports connections and VLAN segmentation.

In Figure 4-61, notice two open vSwitch bridges, the integration bridge (`br-int`), and the physical connectivity bridge (`br-eth1`). The integration bridge `br-int` is where all of the instances from the Compute node are connected, and it is connected to the physical connectivity bridge, `br-eth1`, by using veth pairs (virtual patch cables) to provide access to the external network for the instances.

For more information about the OpenStack network topology in Open vSwitch bridges, see the OpenStack documentation:

http://bit.ly/19N4K4H

Note: Although no instance will run in a dedicated Controller node (`ibm-os-single-controller-node`), it will still create an Open vSwitch bridge to be used by the DHCP server process to provide IP addresses to your instances.

4.10.2 Shared Ethernet Adapters

A Shared Ethernet Adapter (SEA) is used in PowerVM Compute nodes to bridge a physical network adapter to a virtual network adapter, allowing logical partitions using the same VLAN ID as the virtual network adapter to access the external network. The SEA is hosted in a Virtual I/O Server and you have manually setup this SEA in section 4.8.2, “Set up Virtual I/O Servers” on page 275.

Figure 4-62 briefly shows the network topology in a PowerVM Compute node, explaining how the network packets flow between the virtual machines (instances) and external network.

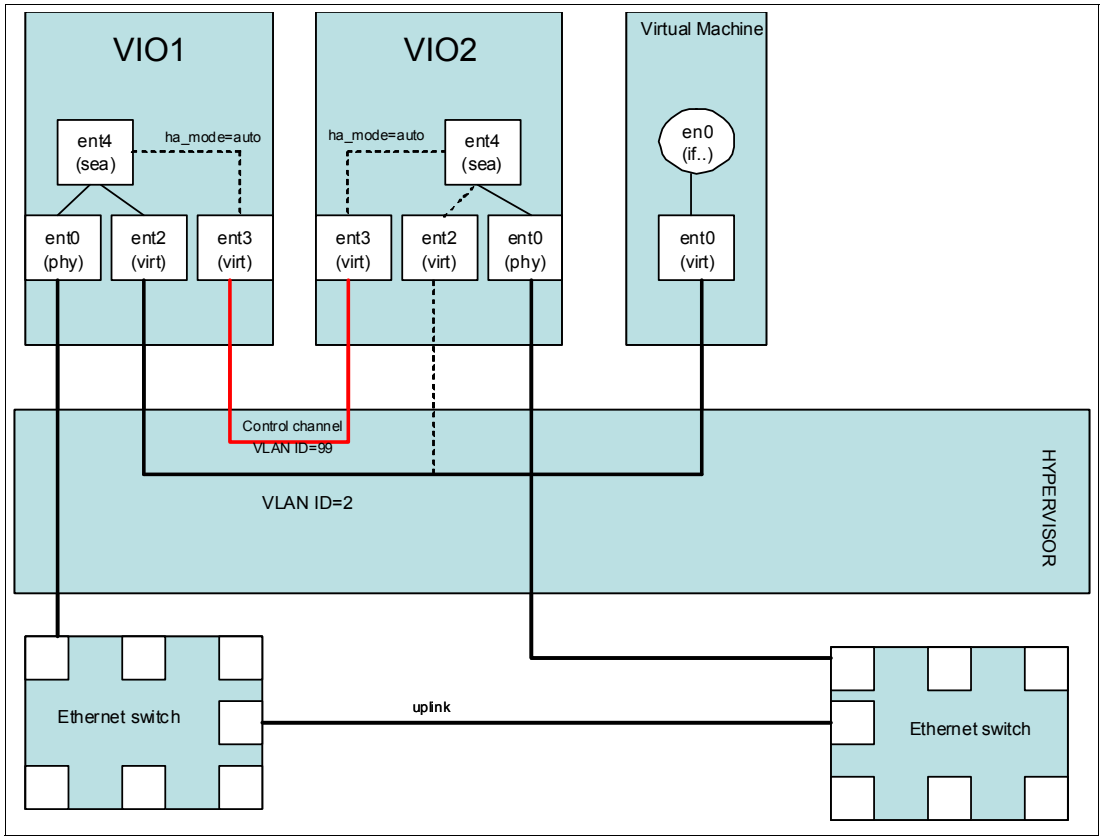


Figure 4-62 Shared Ethernet Adapter topology

The instance will receive a Virtual Ethernet Adapter with a certain VLAN ID (in the example above, VLAN ID is 2). This will allow this Virtual Ethernet Adapter to communicate with the Virtual I/O Server (VIOS) Virtual Ethernet Adapter, which is also using the same VLAN ID and has the `Use this adapter for Ethernet bridging` option enabled, allowing it to be part of a Shared Ethernet Adapter. In the VIOS, this Virtual Ethernet Adapter is connected to the physical adapter, creating a Shared Ethernet Adapter, which is a bridge between the Virtual side and the physical side, allowing instances to access the external network.

For more information about Shared Ethernet Adapter, see *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940:

`http://www.redbooks.ibm.com/abstracts/sg247940.html`

4.10.3 Network configuration overview

As mentioned before, most of the network configuration was already completed by Chef server during deployment steps. The Chef server has cookbooks and recipes in the Chef repository with instructions to install and prepare the network configuration in the Controller node and all the Compute nodes.

Important: In this section 4.10.3, “Network configuration overview” on page 330 you will not modify any configuration in your OpenStack cloud. The objective of this section is only to examine the configurations completed by Chef during the deployment step, according to the information provided in the `env.json`, `topo.json` and node-specific attribute files.

During the deployment steps, you have configured the data network interface in the Controller node (at section Figure on page 222), KVM Compute node (at section “Configure the data network interface” on page 240) and PowerKVM Compute nodes (at section “Configure the data network interface” on page 264) to be up after every reboot, but without any IP address. This is the first step for allowing Chef server to automatically prepare the data network configuration. After completing this, the following modifications were made in the `env.json`, `topo.json` and node-specific attribute file.

Configure tunneling

Given that no GRE network is used in the environment, the `openstack.openvswitch.enable_tunneling` configuration was modified to `False` in the `env.json` file as shown in Example 4-102.

Example 4-102 Disable tunneling configuration in env.json

```
273         "openvswitch": {
274           "tenant_network_type": "vlan",
275           "network_vlan_ranges": "default:1:4094",
276           "enable_tunneling": "False",
277           "tunnel_type": "gre",
278           "tunnel_id_ranges": "1:1000"
279         }
280       },
```

This reflects on Chef making a modification in the `/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini` file in the Controller node, KVM Compute nodes and PowerKVM Compute nodes as shown in Example 4-103 on page 331.

Example 4-103 Disable tunneling configuration in the openvswitch configuration file

```
30 # (BoolOpt) Set to True in the server and the agents to enable support
31 # for GRE or VXLAN networks. Requires kernel support for OVS patch ports and
32 # GRE or VXLAN tunneling.
33 #
34 # Default: enable_tunneling = False
35 enable_tunneling = False
```

This will prevent the use of GRE networks and avoid configuration conflicts in your OpenStack cloud. The Open vSwitch will read this file every time it is restarted. The Chef server will modify this file and automatically restart the Open vSwitch daemon during the deployment step.

Note: If you plan to use GRE networks, which is not described in this publication, you should not disable tunneling. Keep in mind that GRE network is only available when all the nodes from your OpenStack cloud are x86_64 KVM Compute nodes.

Open vSwitch network bridges

During the deployment steps, the Open vSwitch network bridge was already created in the nodes.

Network bridges in the Controller node and KVM Compute nodes

Given that your Controller node has a node-specific attribute file, but it does not contain any information related to network bridges, the information of how to set up the network bridge was obtained from the `openstack.network.openvswitch` section in `env.json` file as shown in Example 4-104. The KVM Compute nodes have no node-specific attribute files, so the network configuration was also obtained from the `env.json` file.

Example 4-104 Open vSwitch network bridge configuration from the env.json file

```
107       "network": {
108         "verbose": false,
109         "debug": false,
110         "openvswitch": {
111           "bridge_mappings": "default:br-eth1",
112           "bridge_mapping_interface": "br-eth1:eth1"
113         }
114       },
```

The `openstack.network.openvswitch.bridge_mappings` section instructed the Chef server to create a network called *default* in the `neutron` OpenStack component and attach this `default` network with an Open vSwitch network bridge called *br-eth1*. This configuration was made by Chef server in the `/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini` file of the Controller node and KVM Compute nodes, as shown in Example 4-105 on page 332.

Example 4-105 bridge_mappings in the Open vSwitch configuration file

```
77 # (ListOpt) Comma-separated list of <physical_network>:<bridge> tuples
78 # mapping physical network names to the agent's node-specific OVS
79 # bridge names to be used for flat and VLAN networks. The length of
80 # bridge names should be no more than 11. Each bridge must
81 # exist, and should have a physical network interface configured as a
82 # port. All physical networks listed in network_vlan_ranges on the
83 # server should have mappings to appropriate bridges on each agent.
84 #
85 # Default: bridge_mappings =
86 # Example: bridge_mappings = physnet1:br-eth1
87 bridge_mappings = default:br-eth1
```

In the same file, you also see the name of the Open vSwitch integration bridge, as shown in Example 4-106. All the virtual network interface card (or TAP devices) from the instances are connected to this Integration bridge.

Example 4-106 Integration bridge configuration in the Open vSwitch configuration file

```
51 # Do not change this parameter unless you have a good reason to.
52 # This is the name of the OVS integration bridge. There is one per hypervisor.
53 # The integration bridge acts as a virtual "patch bay". All VM VIFs are
54 # attached to this bridge and then "patched" according to their network
55 # connectivity.
56 #
57 # Default: integration_bridge = br-int
58 integration_bridge = br-int
```

The `openstack.network.openvswitch.bridge_mapping_interface` section instructed the Chef server to create an Open vSwitch network bridge called `br-eth1` using `eth1` as the physical adapter to provide external network access to the instances. Open vSwitch commands were used by the Chef server according to its cookbooks and recipes to create such bridge. You can verify that the bridge was corectly created using the **`ovs-vsctl show`**, command, as shown in Example 4-107.

Example 4-107 ovs-vsctl show command output from KVM Compute node

```
[root@cmo1 openvswitch]# ovs-vsctl show
f16dd0ea-3225-49d6-aab2-bc7461333ce7
Bridge br-tun
        Port "gre-ac101582"
            Interface "gre-ac101582"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.131", out_key=flow,
remote_ip="172.16.21.130"}
        Port "gre-ac101590"
            Interface "gre-ac101590"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.131", out_key=flow,
remote_ip="172.16.21.144"}
        Port "gre-ac101589"
            Interface "gre-ac101589"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.131", out_key=flow,
remote_ip="172.16.21.137"}
```

```
        Port br-tun
            Interface br-tun
                type: internal
        Port "gre-ac10158f"
            Interface "gre-ac10158f"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.131", out_key=flow,
remote_ip="172.16.21.143"}
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
    Bridge "br-eth1"
        Port "br-eth1"
            Interface "br-eth1"
                type: internal
        Port "phy-br-eth1"
            Interface "phy-br-eth1"
        Port "eth1"
            tag: 2
            Interface "eth1"
    Bridge br-int
        Port "qvo8f2167e3-6e"
            tag: 3
            Interface "qvo8f2167e3-6e"
        Port "qvo1380dec7-56"
            tag: 4095
            Interface "qvo1380dec7-56"
        Port "qvo55d46868-c6"
            tag: 3
            Interface "qvo55d46868-c6"
        Port "qvo623b69f2-1c"
            tag: 4095
            Interface "qvo623b69f2-1c"
        Port "qvo08a14a92-62"
            tag: 4095
            Interface "qvo08a14a92-62"
        Port "qvoaabf7c00-58"
            tag: 4095
            Interface "qvoaabf7c00-58"
        Port br-int
            Interface br-int
                type: internal
        Port "qvo899ee5be-22"
            tag: 4095
            Interface "qvo899ee5be-22"
        Port "qvo65424435-b1"
            tag: 1
            Interface "qvo65424435-b1"
        Port "qvo313a21e5-45"
            tag: 1
            Interface "qvo313a21e5-45"
        Port "qvo5acc9012-24"
            tag: 3
            Interface "qvo5acc9012-24"
```

```
        Port "int-br-eth1"
            Interface "int-br-eth1"
    ovs_version: "2.0.0"
```

From the output shown in Example 4-107 on page 332 you can see the Open vSwitch network bridge `br-eth1` connected to the physical port `eth1`. You can also see the veth pair (virtual patch cable) `phy-br-eth1` connected to this bridge. We can also see the integration bridge `br-int` where all instances are connected to, using its TAP devices (virtual network interface card). The other side of the phy-br-eth1 virtual patch cable, called int-br-eth1 is connected to the br-int network bridge. This is what allows the instances running in this Compute node to access the external network. This configuration shows the topology explained in topic 4.10.1, “Open vSwitch” on page 327.

Note: You may notice GRE interfaces in the Example 4-107 on page 332. This is because Tunneling was disabled, but GRE networks are still in the `openstack.network.ml2.type_drivers` section of the env.json file in the deployment server.

Although the Controller node does not have any instance (virtual machine) in the topology created for this publication, it also needs an Open vSwitch network bridge to use a TAP device dedicated for the DHCP server, allowing it to communicate with the instances running in other Compute nodes, providing an IP address for each of them. Example 4-108 shows the network interface being used for the DHCP server in the Controller node and how it is configured in the Open vSwitch network bridge.

Example 4-108 Open vSwitch network bridge and DHCP server in the Controller node

```
[root@cmocontroller openvswitch]# ps -ef | grep dnsmasq
nobody   22895     1  0 Aug28 ?        00:00:05 /usr/sbin/dnsmasq --no-hosts
--no-resolv --strict-order --bind-interfaces --interface=tap9560a619-76
--except-interface=lo
--pid-file=/var/lib/neutron/dhcp/d828cdb1-e926-4f94-b276-29a41471e5b6/pid
--dhcp-hostsfile=/var/lib/neutron/dhcp/d828cdb1-e926-4f94-b276-29a41471e5b6/host
--addn-hosts=/var/lib/neutron/dhcp/d828cdb1-e926-4f94-b276-29a41471e5b6/addn_hosts
--dhcp-optsfile=/var/lib/neutron/dhcp/d828cdb1-e926-4f94-b276-29a41471e5b6/opts
--leasefile-ro --dhcp-range=set:tag0,192.168.21.0,static,86400s
--dhcp-lease-max=256 --conf-file=/etc/neutron/dnsmasq.conf --domain=openstacklocal

[root@cmocontroller openvswitch]# ovs-vsctl show
98ddf3b8-564c-473a-8913-37ce37e9f1a3
    Bridge br-int
        Port "tap9560a619-76"
            tag: 1
            Interface "tap9560a619-76"
                type: internal
        Port "int-br-eth1"
            Interface "int-br-eth1"
        Port br-int
            Interface br-int
                type: internal
    Bridge br-tun
        Port "gre-ac101582"
            Interface "gre-ac101582"
                type: gre
```

```
                options: {in_key=flow, local_ip="172.16.21.137", out_key=flow,
remote_ip="172.16.21.130"}
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
        Port "gre-ac10158f"
            Interface "gre-ac10158f"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.137", out_key=flow,
remote_ip="172.16.21.143"}
        Port br-tun
            Interface br-tun
                type: internal
        Port "gre-ac101583"
            Interface "gre-ac101583"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.137", out_key=flow,
remote_ip="172.16.21.131"}
        Port "gre-ac101590"
            Interface "gre-ac101590"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.137", out_key=flow,
remote_ip="172.16.21.144"}
    Bridge "br-eth1"
        Port "phy-br-eth1"
            Interface "phy-br-eth1"
        Port "br-eth1"
            Interface "br-eth1"
                type: internal
        Port "eth1"
            tag: 2
            Interface "eth1"
    ovs_version: "2.0.0"
```

Note: DHCP server is an optional configuration. For more information see 4.10.4, “Create networks in IBM Cloud Manager” on page 339.

Network bridges in the PowerKVM Compute nodes

The same kind of configuration was accomplished by Chef server in the PowerKVM Compute nodes, but given that node-specific attribute files were used, the Chef server has considered the information provided in these files to create the network bridge in these nodes.

The `openstack.network.openvswitch.bridge_mappings` section in the node-specific attribute file, tells Chef to create a neutron network called `default` and connect it to an Open vSwitch bridge called `br-ext`. The `openstack.network.openvswitch.bridge_mapping_interface` tells chef to create an Open vSwitch network bridge called `br-ext` connected to the `enP3p1s0f1` network interface, as shown in Example 4-109 on page 336.

Example 4-109 PowerKVM node-specific attribute file

```
1 {
2 "openstack": {
3       "endpoints": {
4         "network-openvswitch": {
5           "bind_interface": "enP3p1s0f0"
6         },
7         "compute-vnc-bind": {
8           "bind_interface": "enP3p1s0f0"
9         }
10       },
11       "network": {
12         "verbose": false,
13         "debug": false,
14         "openvswitch": {
15           "bridge_mappings": "default:br-ext",
16           "bridge_mapping_interface": "br-ext:enP3p1s0f1"
17         }
18       }
19  }
20 }
```

You can see this configuration setup in the /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini configuration file from the PowerKVM Compute nodes, as shown in Example 4-110.

Example 4-110 Open vSwitch network configuration file in the PowerKVM Compute nodes

```
77 # (ListOpt) Comma-separated list of <physical_network>:<bridge> tuples
78 # mapping physical network names to the agent's node-specific OVS
79 # bridge names to be used for flat and VLAN networks. The length of
80 # bridge names should be no more than 11. Each bridge must
81 # exist, and should have a physical network interface configured as a
82 # port. All physical networks listed in network_vlan_ranges on the
83 # server should have mappings to appropriate bridges on each agent.
84 #
85 # Default: bridge_mappings =
86 # Example: bridge_mappings = physnet1:br-eth1
87 bridge_mappings = default:br-ext
```

You can also see the Open vSwitch network bridge created with the **ovs-vsctl show** command, as shown in Example 4-111.

Example 4-111 ovs-vsctl show command from PowerKVM Compute node

```
[root@powerkvm1 openvswitch]# ovs-vsctl show
614c8a53-3ca8-438b-99e7-c5b2dde6c251
    Bridge br-int
        Port "qvoba0f4685-4c"
            tag: 9
            Interface "qvoba0f4685-4c"
        Port "qvoa5af8afc-33"
            tag: 8
            Interface "qvoa5af8afc-33"
        Port int-br-ext
```

```
            Interface int-br-ext
        Port "qvo497e3e1c-20"
            tag: 8
            Interface "qvo497e3e1c-20"
        Port "qvob6842b31-94"
            tag: 9
            Interface "qvob6842b31-94"
        Port br-int
            Interface br-int
                type: internal
    Bridge br-ext
        Port "enP3p1s0f1"
            tag: 2
            Interface "enP3p1s0f1"
        Port br-ext
            Interface br-ext
                type: internal
        Port phy-br-ext
            Interface phy-br-ext
    Bridge br-tun
        Port "gre-ac101582"
            Interface "gre-ac101582"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.143", out_key=flow,
remote_ip="172.16.21.130"}
        Port "gre-ac101589"
            Interface "gre-ac101589"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.143", out_key=flow,
remote_ip="172.16.21.137"}
        Port "gre-ac101590"
            Interface "gre-ac101590"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.143", out_key=flow,
remote_ip="172.16.21.144"}
        Port "gre-ac101583"
            Interface "gre-ac101583"
                type: gre
                options: {in_key=flow, local_ip="172.16.21.143", out_key=flow,
remote_ip="172.16.21.131"}
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
        Port br-tun
            Interface br-tun
                type: internal
    ovs_version: "2.0.0"
```

Networks imported from PowerVC

If you have PowerVM Compute nodes in your environment, and used PowerVC driver in the Controller node to connect to the PowerVC server, the networks configured in PowerVC in step 15 on page 296 were automatically imported to Cloud Manager. The following steps allow you to review this configuration:

1. Log in to your PowerVC server, and click **Networks** on the left side of the web interface.
2. You should see two networks which were automatically imported after you managed existing virtual machines and two networks which you manually created in step 15 on page 296, as shown in Figure 4-63.

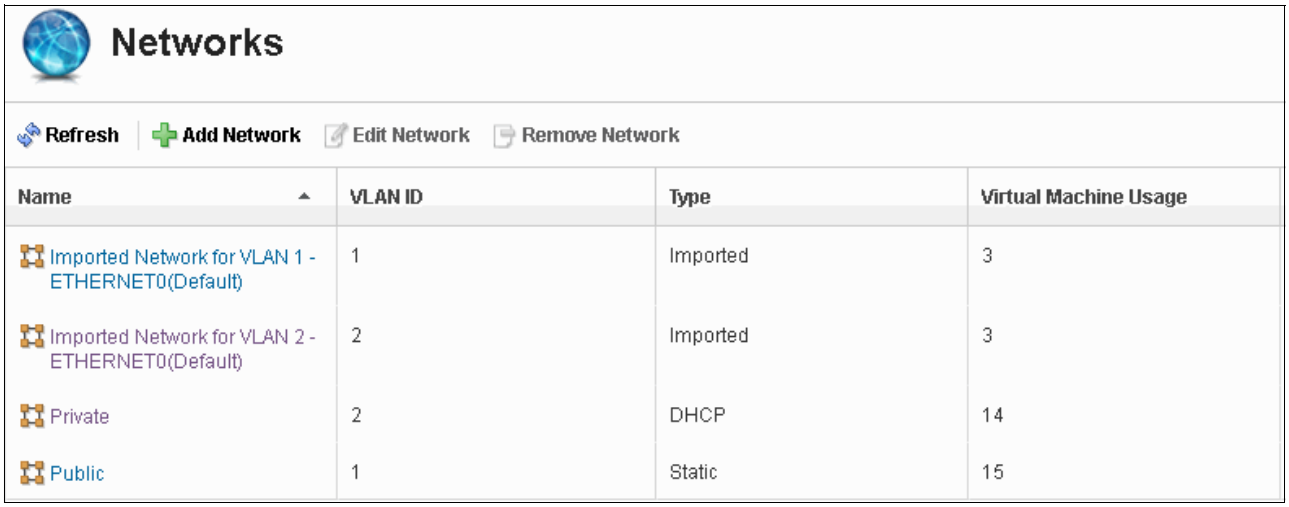


Figure 4-63 Networks in PowerVC server

3. Log in to the Cloud Management dashboard and click **ADMIN > System Panel > Networks**. You should see the two networks which you manually created in PowerVC, as shown in Figure 4-64.

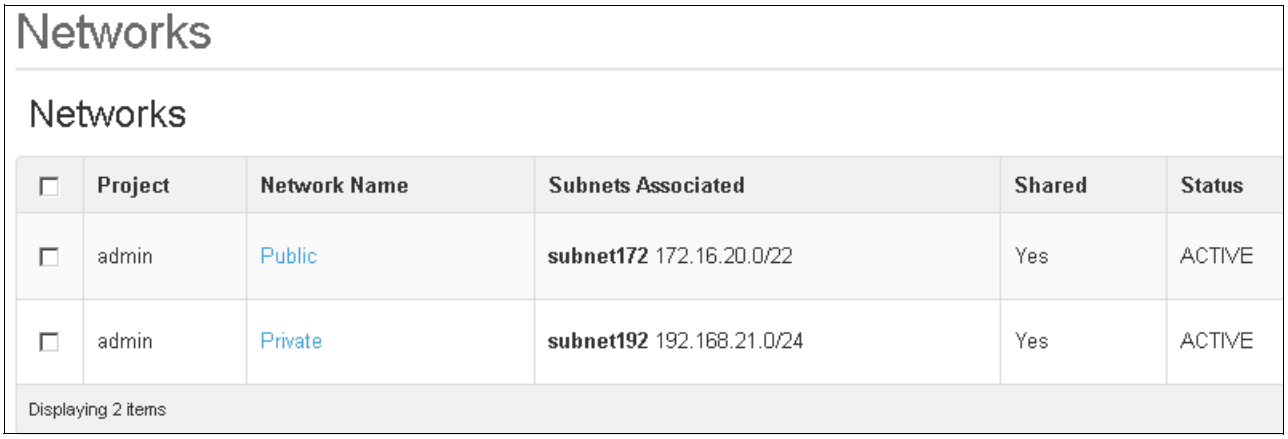


Figure 4-64 Networks in IBM Cloud Manager

4.10.4 Create networks in IBM Cloud Manager

Network setup is complete in the Controller and Compute nodes (hypervisors), but now you need to create a network in the Cloud Management dashboard to allow your instances to receive a virtual network interface card when you deploy an instance. During this step, you will create a network in Cloud Management dashboard and link it to the Open vSwitch network bridge available in the Compute nodes.

Note: This book provides instructions for flat and VLAN networks. For information about local and GRE networks, see the IBM Cloud Manager with OpenStack 4.1 Administrator Guide:

`http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html`

The next step will depend on the hypervisors available in your OpenStack cloud and the network type you plan to use, use the following topics to determine which kind of configuration you need to implement in the environment:

- For an OpenStack cloud with KVM or PowerKVM Compute nodes and Flat network, see “Create flat networks” on page 339.
- For an OpenStack cloud with KVM or PowerKVM Compute nodes and VLAN network, see “Create VLAN networks” on page 344.
- For an OpenStack hybrid cloud with PowerVM Compute nodes (PowerVC Driver), KVM, or PowerKVM Compute nodes, see “Modify networks imported from PowerVC server” on page 347.
- For an OpenStack cloud exclusively with PowerVM Compute nodes (PowerVC driver) where networks were already imported to Cloud Manager, the setup is complete and no additional configuration is required.

Create flat networks

In flat networks, a subnet is created, providing an IP address range. instances deployed in the OpenStack cloud using this network will receive an IP from this range. The IP address can be configured using DHCP server or fixed IP address:

- DHCP server. Cloud Manager will start a DHCP server (called dnsmasq) in the Controller node to allocate IP addresses to instances from the specified subnet.
- Fixed IP address. An IP address from the subnet range is injected to the instance during start process. The image from which the instance was deployed must have the `cloud-init` software installed to receive this IP address and automatically configure it as a fixed IP address.

The following steps will guide you in Cloud Manager to create a flat network:

1. Log in to the Cloud Management dashboard as the `admin` user.
2. Select **Admin > System Panel > Networks**, and click **Create Network**.
3. Provide the following information for your network:
   - Name. Provide a name for your network.
   - Project. You can create multiple projects in Cloud Manager (intended for multiple tenants). Select **Public** to allow this network to be used by multiple tenants.
   - Provider Network Type. Select **Flat**.

- Physical Network. The Cloud Manager will automatically fill this field with the `default` word. Keep the default name in this field, because this is the link between the network you are creating in the Cloud Management dashboard and the Open vSwitch bridge created during the deployment process (remember that you have specified **`default:`**`br-eth1` in the `openstack.network.openvswitch.bridge_mappings` entry in the `env.json` file, or **`default:`**`br-ext` in the PowerKVM Compute nodes).
- Admin State. Keep this check box selected. It means that the network is up.
- Shared. Select this check box to allow this network to be used by multiple projects.
- External Network. Select this check box to allow instances that use this network to have access to the external network. Optional: Use a floating IP addresses.

Figure 4-65 shows an example of a flat network being created. After providing all of the required information, click **Create Network**.

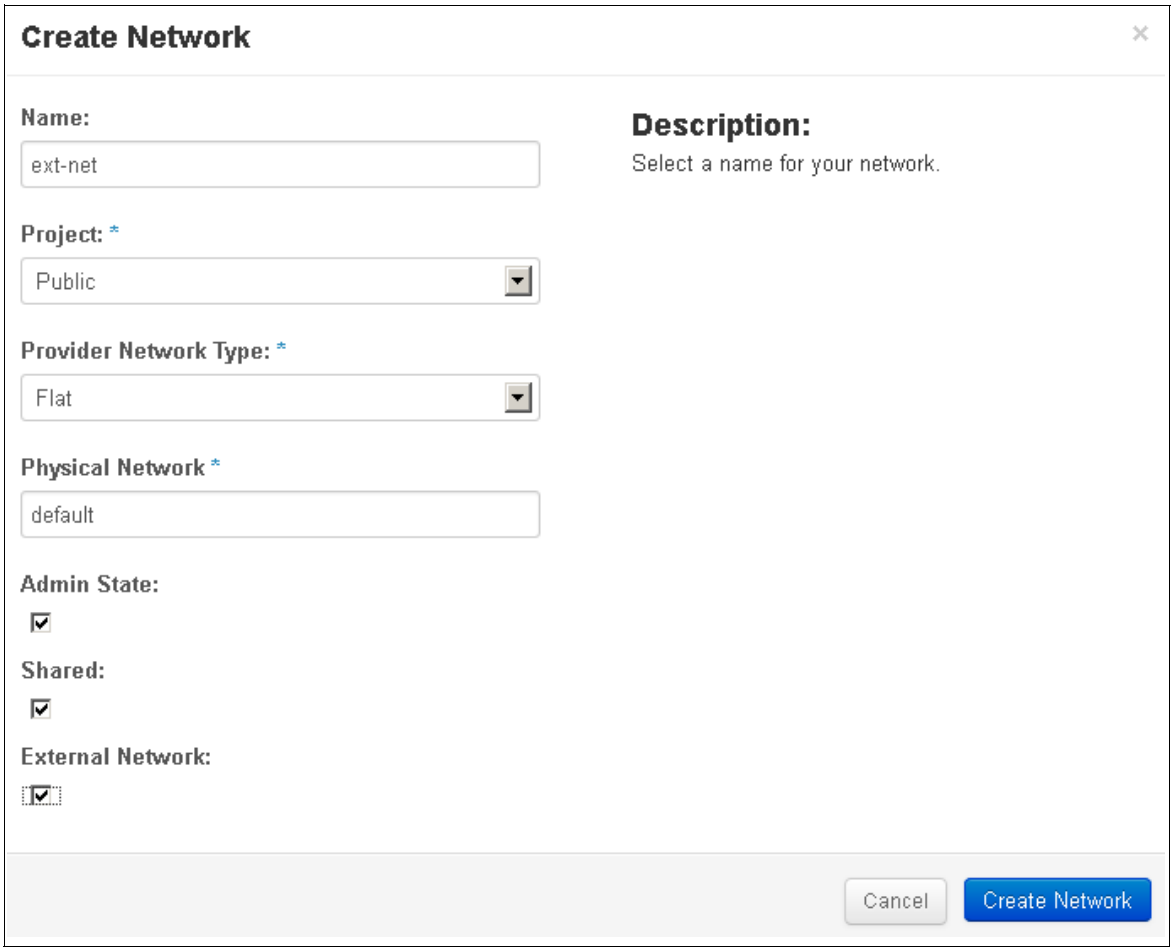


Figure 4-65 Creating a flat network in the Cloud Management dashboard

Optional: You can assign non-valid IP addresses for your instances, and define a range of valid IP addresses (floating IP addresses) and assign it temporarily to your instance when it needs to access the network. For example, if you have many instances in your cloud, but not enough IP addresses from your network to attend all the instances, you can create a different network and subnet with a non-valid range of IP addresses and assign them to your instances. Then, you can create a range of floating IP addresses that are valid in your network and temporarily assign them to the instances when it needs to access other hosts from your network.

This kind of configuration is not covered in this publication.

4. Select your network and click its name in the Network Name column, as shown in Figure 4-66.

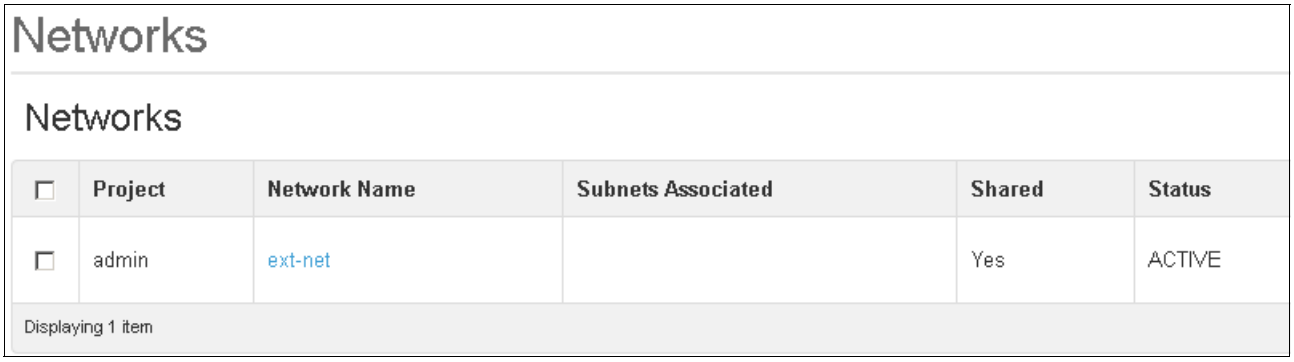


Figure 4-66 Accessing Flat network definition

5. Click **Create Subnet**.
6. Provide the following subnet information:
   - Subnet Name. A name to identify your Subnet.
   - Network Address. The network address and netmask in CIDR (Classless Inter-Domain Routing) format (for example, 192.168.21.0/24).
   - IP Version. Either IPv4 or IPv6.
   - Gateway IP address. Your gateway IP address.

   After providing this information, as shown in Figure 4-67 on page 342, click **Next**.

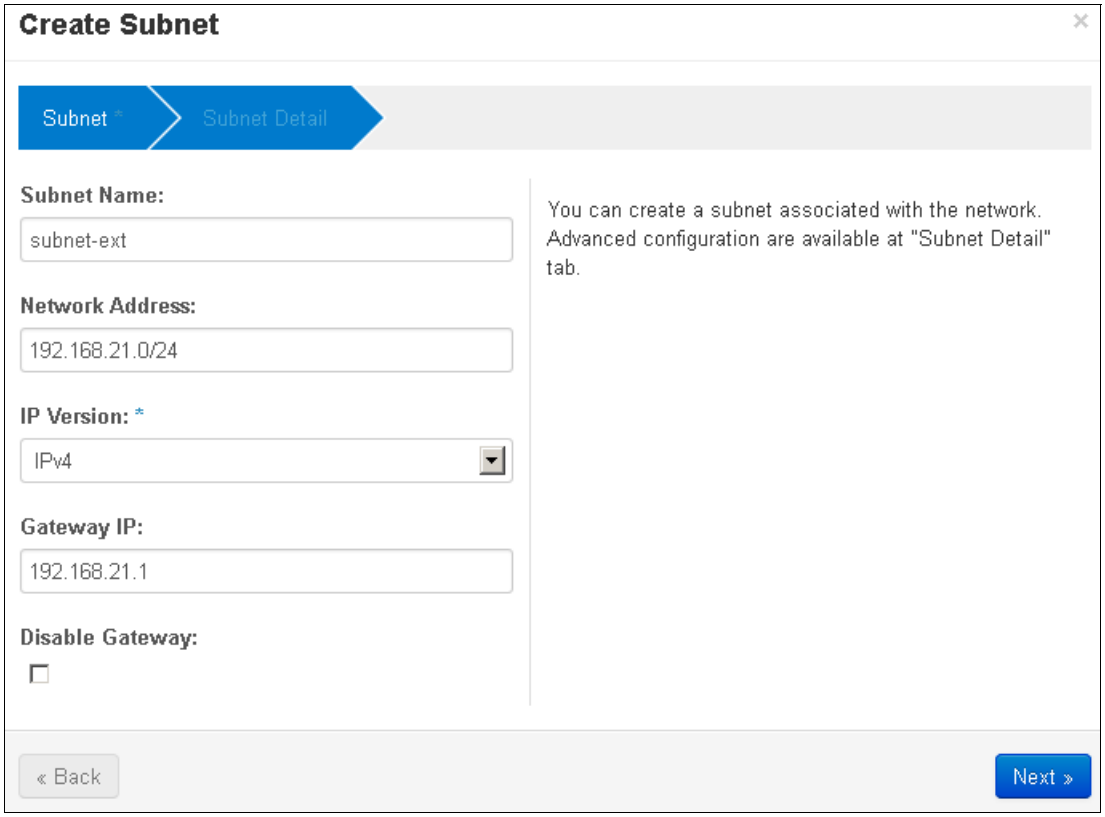


Figure 4-67 Providing subnet information

7. Select the **Enable DHCP** check box if you want to activate the DHCP server (`dnsmasq`) in the Controller node for this network, to automatically provide IP addresses to the instances. Leave this option unchecked if you want to assign fixed IP addresses to the instances (requires `cloud-init` to be installed in the image). Provide the following additional information.
   - Allocation Pools. Pool of IP addresses available to be allocated to the instances deployed using this network. The pool provided should be in the `<initial_ip_address>`, `<end_ip_address>` format, for example:

     `192.168.21.100,192.168.21.200`
   - DNS Name Servers. DNS Servers (if available). The IP addresses must be provided one per line.
   - Host Routes. Additional routes you may want to configure in the instances deployed using this network.

Figure 4-68 on page 343 shows an example of how this information should be provided. After completing this step click Create.

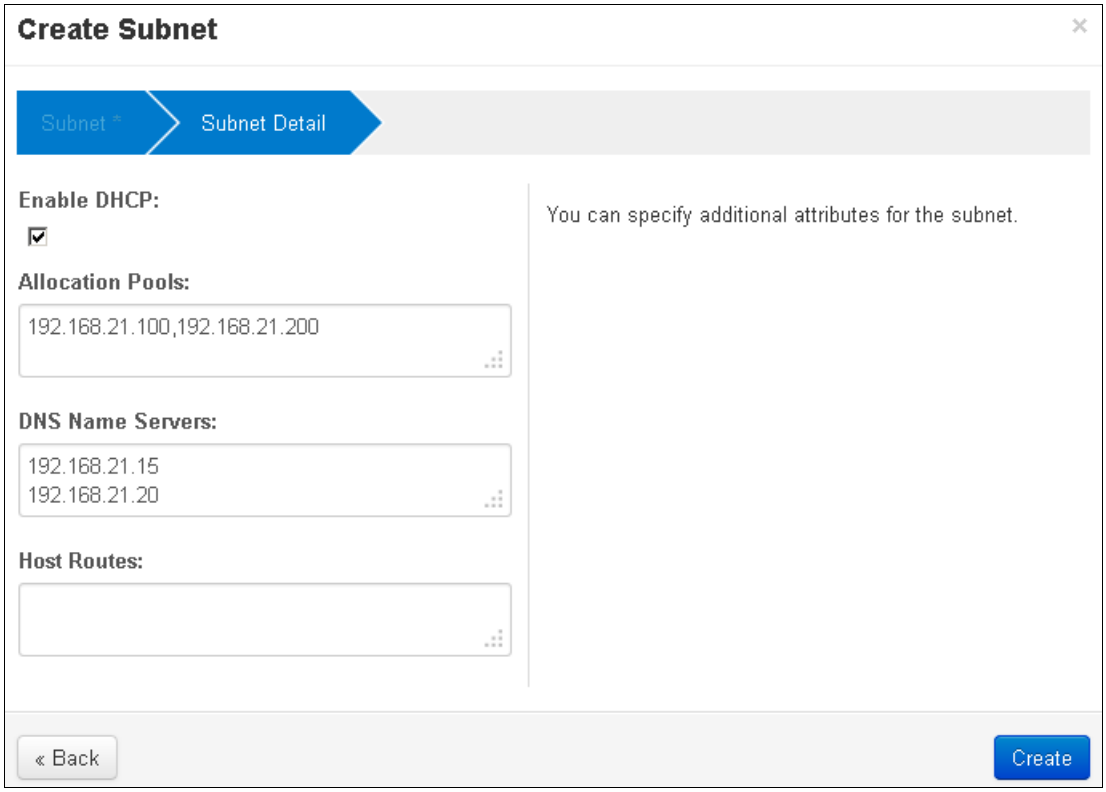


Figure 4-68 Providing subnet details

After completing these steps, the network you created will be available when deploying an instance, as shown in Figure 4-69.

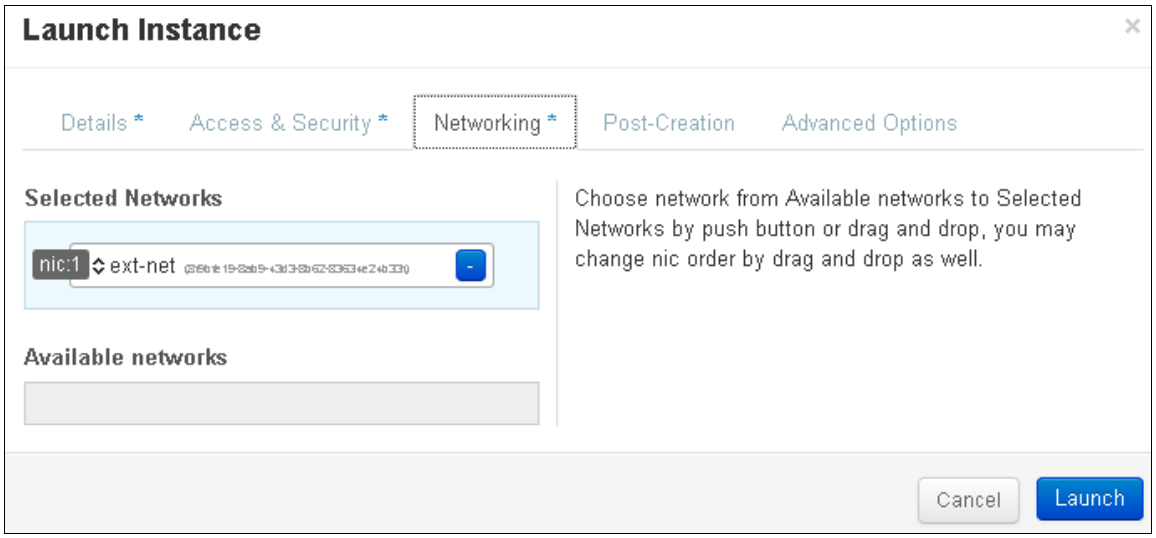


Figure 4-69 Deploying an instance using the network create in Cloud Management dashboard

The instances deployed using the network you created using these steps will be connected to the **`default`** `neutron` network, which is connected to the `br-eth1` (or `br-ext`) Open vSwitch bridge, allowing them to access the external network.

Create VLAN networks

In VLAN networks, a network is created with IEEE 802.1Q enabled. You need to provide a VLAN ID for this network, and the physical Ethernet switch where the data network is connected must be compatible and corectly configured with IEEE 802.1Q and the appropriate VLAN ID you want to use for your network.

Note: You can create multiple networks. Each network has its own VLAN ID. The physical port in the Ethernet switch connected to the data network interface must be configured with IEEE 802.1Q and the VLAN IDs must match the VLAN IDs that you used in the networks that were created in the Cloud Management dashboard.

This network configuration is valuable when multiple tenants have instances in the same physical Compute node because then you can segregate tenants using different networks and VLAN IDs.

You must specify a subnet definition for your network, providing a range of IP addresses that can be assigned to the instances using this network. The IP addresses can be assigned using DHCP server or fixed IP addresses:

- DHCP server

  Cloud Manager will start a DHCP server (called `dnsmasq`) in the Controller node to allocate IP addresses to instances from the specified subnet.

- Fixed IP address

  An IP address from the subnet range is injected to the instance during start process. The image from which the instance was deployed must have the `cloud-init` software installed to receive this IP address and automatically configure it as a Fixed IP address.

The following steps will guide you in using Cloud Manager to creat a VLAN network:

1. Log in to the Cloud Management dashboard as the `admin` user.
2. Click **Admin > System Panel > Networks** and click **Create Network**.
3. Provide the following information to your network:
   - Name. Provide a name to identify your network.
   - Project. You can create multiple `projects` in Cloud Manager (intended for multiple tenants). Select **Public** to allow this network to be used by multiple tenants.
   - Provider Network Type. Select **VLAN**.
   - Physical Network. The Cloud Manager will automatically fill this field with the word `default`. Keep the `default` name in this field, because this is the link between the network you are creating in the Cloud Management dashboard and the Open vSwitch bridge created during the deployment process (remember that you have specified **`default:`**`br-eth1` in the `openstack.network.openvswitch.bridge_mappings` entry in the `env.json` file, or **`default:`**`br-ext` in the PowerKVM Compute nodes).
   - Segmentation ID. The VLAN ID for your VLAN network.
   - Admin State. Keep this check box selected. It means that the network is up.
   - Shared. Select this check box to allow this network to be used by multiple projects.
   - External Network. Select this check box to allow instances using this network to have access to external network and, optionally, use floating IP addresses.

   Figure 4-70 on page 345 shows an example of a VLAN network being created by using VLAN ID 2. After providing all of the required information, click **Create Network**.

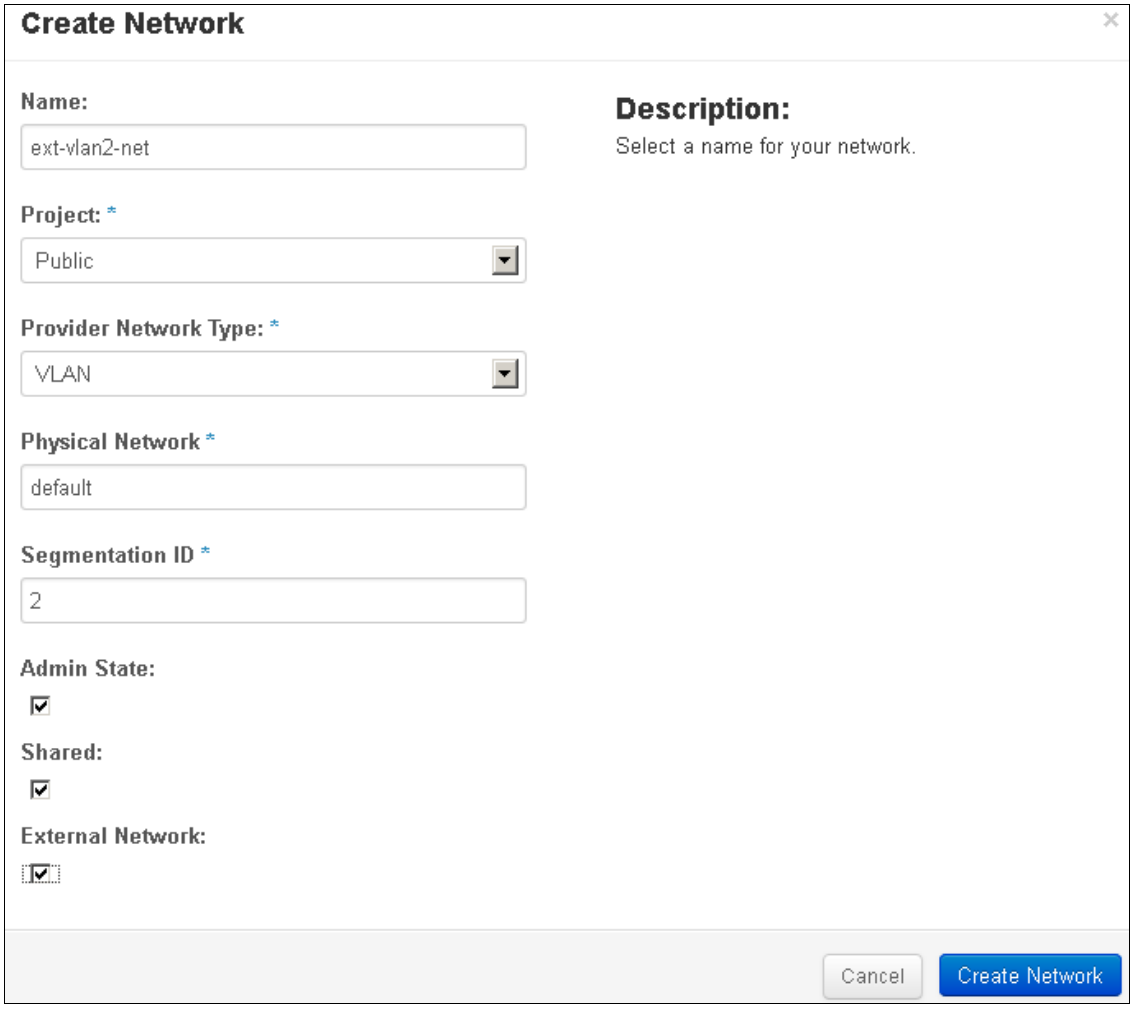


Figure 4-70 VLAN network definition

4. Select your network and click its name in the Network Name column, as shown in Figure 4-71.

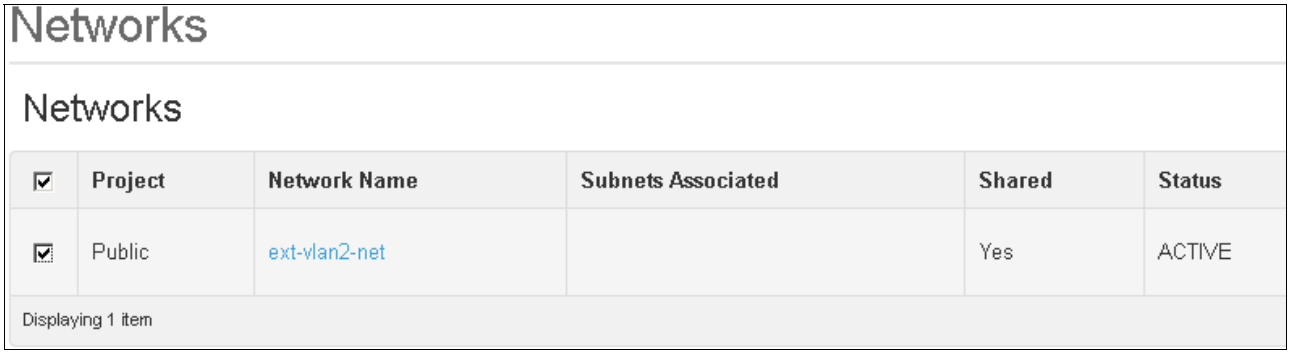


Figure 4-71 Accessing the VLAN network created

5. Click **Create Subnet**.
6. Provide the following information:
   - Subnet name. A name to identify your subnet.
   - Network Address. The Network Address and Netmask in CIDR (Classless Inter-Domain Routing) format (for example 192.168.21.0/24).
   - IP Version. Select either IPv4 or IPv6.
   - Gateway IP Address. Your gateway IP address.

   After providing this information, click **Next**. Figure 4-67 on page 342 shows an example of how you should provide information in this screen.
7. Enable DHCP check box:
   - Select the **Enable DHCP** check box if you want to activate the DHCP server (`dnsmasq`) in the Controller node for this network to automatically provide IP addresses to the instances.
   - Leave this option unchecked if you want to assign Fixed IP addresses to the instances (requires cloud-init to be installed in the image). Provide the following additional information.
   - Allocation Pools. Pool of IP addresses available to be allocated to the instances deployed using this network. The pool provided should be in this format:

     `<initial_ip_address>,<end_ip_address>`

     For example: `192.168.21.100,192.168.21.200`
   - Domain Name Servers (DNSes) (if available). Provide IP addresses, only one per line.
   - Host Routes. Additional routes that you might want to configure in the instances deployed using this network.

   Figure 4-68 on page 343 shows an example of how this information should be provided. After completing this step, click **Create**.

   After completing these steps, the network you created will be available when deploying an instance, as shown in Figure 4-72.

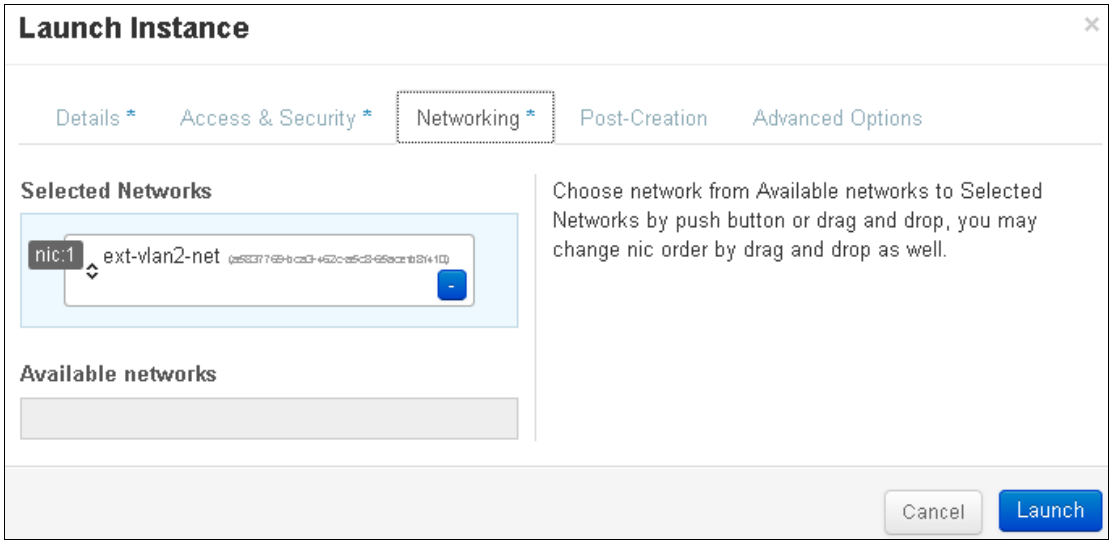


Figure 4-72 Deploying an instance using the VLAN network created in Cloud Management dashboard

The instances deployed using this network will be connected to the **default** neutron network, which is connected to the br-eth1 (or br-ext) Open vSwitch bridge, using VLAN ID 2 (IEEE 802.1Q tagging), allowing them to access the external network (requires appropriate configuration in the physical switch port).

Modify networks imported from PowerVC server

As mentioned in “Networks imported from PowerVC” on page 338, the networks available in the PowerVC server were automatically imported to Cloud Manager during the deployment process (when you connected the PowerVC driver in the Compute node to the PowerVC server. This enables you to manage the PowerVM Compute nodes.

These imported networks are already available for you to deploy instances in the PowerVM Compute nodes. When you select such networks during deployment of PowerVM instances, the Compute node will tell the PowerVC server to use these networks while creating a logical partition in the Power Systems. Figure 4-73 shows the networks imported from PowerVC available for use in Cloud Manager.



Figure 4-73 Networks imported from PowerVC inside Cloud Management dashboard

If the instances to be deployed in KVM Compute nodes and PowerKVM Compute nodes need to use the same network, you need to make some modifications to the subnet, considering the following topics:

- The Compute nodes have only one Open vSwitch network bridge, so all communication to the external network will go through the same physical adapter from the Compute node. Given this information, this adapter and the physical port in the switch where it is connected to, must be prepared to allow communication from all networks that you want to provide to your instances.
- You need to determine if DHCP or Fixed IP address will be used for the instances.

The following steps will enable you to modify the imported networks to **Enable external access** and **Enable DHCP Server** (optional). In this book, we used the network named *Private* to provide external access to the instances:

1. Log in to the Cloud Management dashboard as the admin user.
2. Click **Admin > System Panel > Networks**.
3. Select the network that you want to use for the KVM and PowerKVM instances and click **Edit Network**, as shown in Figure 4-74 on page 348.

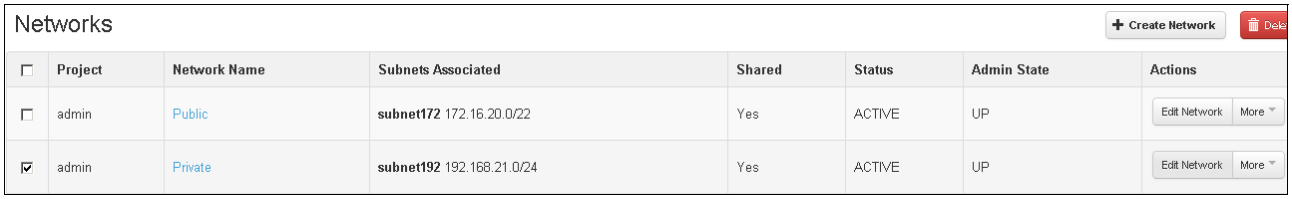


Figure 4-74 Edit network imported from PowerVC

4. Enable the **External Network** option and click **Save Changes**, as shown in Figure 4-75.

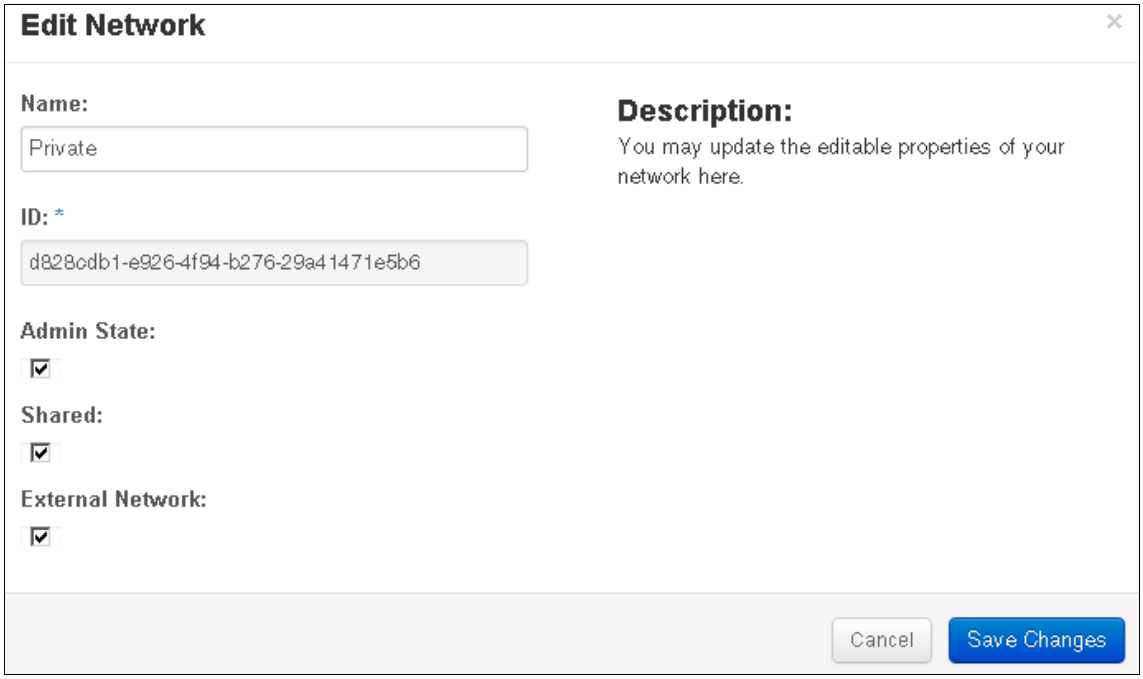


Figure 4-75 Enabling the external network in network imported from PowerVC

5. Select your network and click its name in the Network Name column, as shown in Figure 4-76.



Figure 4-76 Accessing the network imported from PowerVC

6. Select the subnet definition for this network and click **Edit Subnet** as shown in Figure 4-77.

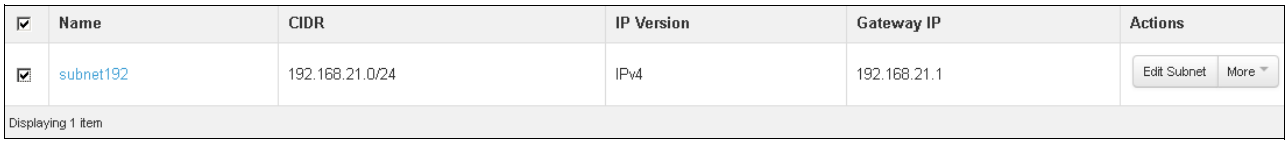


Figure 4-77 Editing the subnet imported from PowerVC

7. Provide a name for the subnet as shown in Figure 4-78.

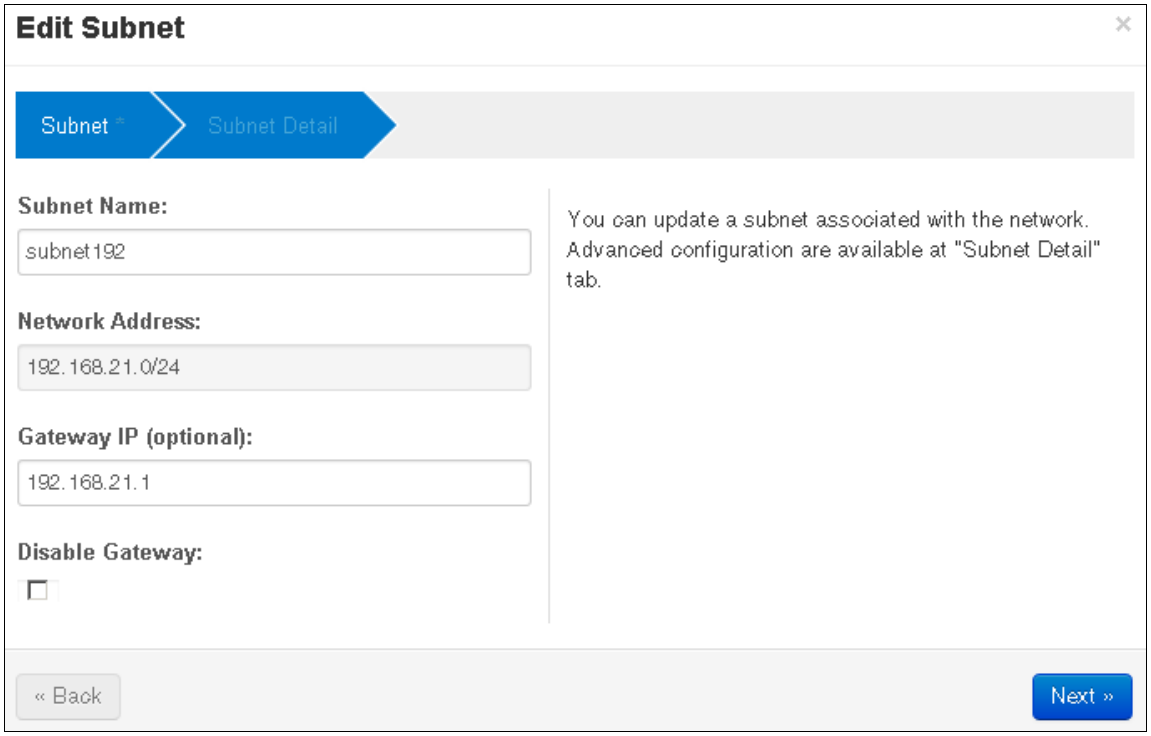


Figure 4-78 Providing a name for the Subnet imported from PowerVC

8. Click **Subnet Detail** and, optionally, click the **Enable DHCP** check box as shown in Figure 4-79 on page 350.

Important: If you do not enable the Enable DHCP check box, the instances deployed in this network will use Fixed IP addresses, and require the installation of cloud-init in the images. You can also specify DNS servers and additional routes to configure in the instances deployed in this network.

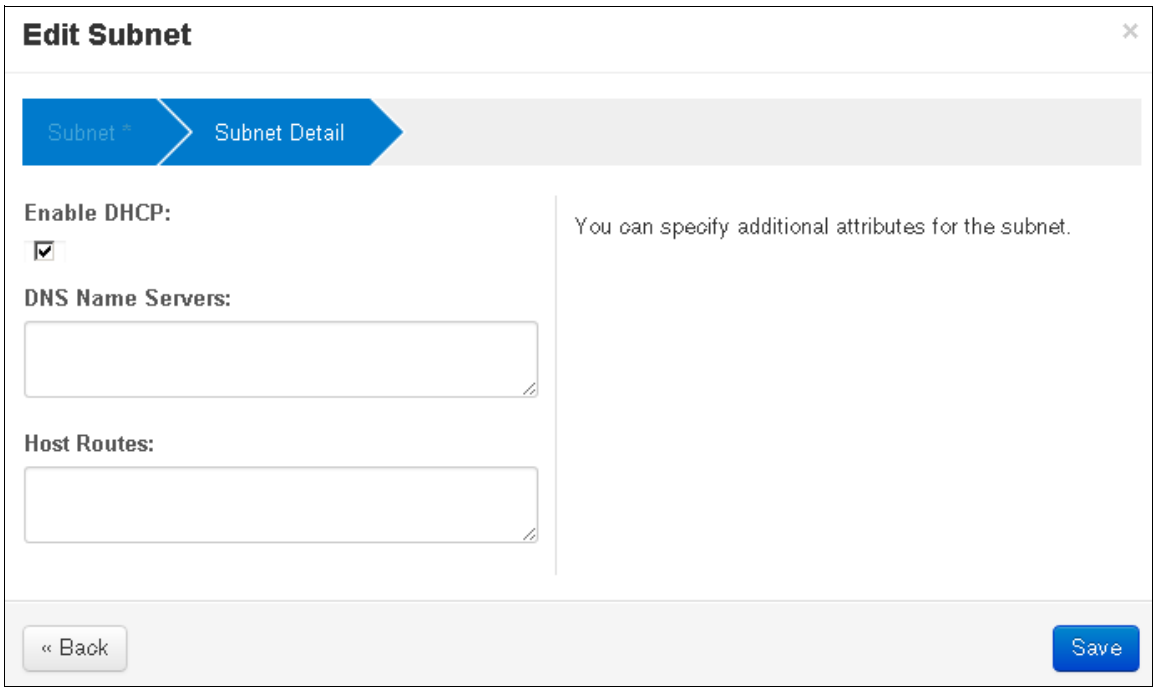


Figure 4-79 Enabling DHCP Server in the network imported from PowerVC

Note: At the time of writing this publication, you cannot modify the IP `Allocation Pools` after the network and subnet are created. Given that the network imported from PowerVC was already created, you cannot modify the IP `Allocation pools`.

After completing these steps, you can use the modified network during deployment of PowerKVM or KVM instances.

4.10.5 Network setup in the cloud prepared for this publication

During our initial stages of configuration, we had a parallel setup where our physical switch had a limitation and enabling VLAN Tagging (IEEE 802.1Q) was not possible. Some workarounds were required to enable network communication between instances deployed in all the hypervisors. This section describes these workarounds.

Network configuration in PowerVM Compute nodes

The following steps describes the network configuration performed in the PowerVM Compute nodes in the cloud setup for this publication:

1. In Virtual I/O Servers, the management network Virtual Ethernet Adapter was created without VLAN Tagging configuration and using VLAN ID 1, as shown in Figure 4-80 on page 351.

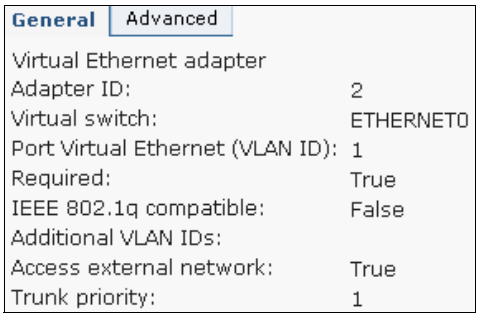


Figure 4-80 Virtual Ethernet Adapter for the management network

2. The Shared Ethernet Adapter was created using a physical Ethernet adapter (ent0) connected to a physical Ethernet switch (IEEE 802.1Q was not configured in the physical switch port) and a SEA was created as shown in Example 4-112.

Example 4-112 SEA configuration for management network in Virtual I/O Server

```
$ lsmap -all -net
SVEA   Physloc
------ --------------------------------------------
ent2   U8233.E8B.061AA6P-V1-C2-T1

SEA                  ent4
Backing device       ent0
Status               Available
Physloc              U78A0.001.DNWHZWR-P1-C4-T1

$ lsdev -dev ent4 -attr
attribute     value    description
user_settable

accounting    disabled Enable per-client accounting of network statistics                True
adapter_reset yes      Reset real adapter on HA takeover                                 True
ctl_chan      ent3     Control Channel adapter for SEA failover                          True
gvrp          no       Enable GARP VLAN Registration Protocol (GVRP)                     True
ha_mode       auto     High Availability Mode                                            True
hash_algo     0        Hash algorithm used to select a SEA thread                        True
jumbo_frames  no       Enable Gigabit Ethernet Jumbo Frames                              True
large_receive no       Enable receive TCP segment aggregation                            True
largesend     1        Enable Hardware Transmit TCP Resegmentation                       True
lldpsvc       no       Enable IEEE 802.1qbg services                                     True
netaddr       0        Address to ping                                                   True
nthreads      7        Number of SEA threads in Thread mode                              True
pvid          1        PVID to use for the SEA device                                    True
pvid_adapter  ent2     Default virtual adapter to use for non-VLAN-tagged packets        True
qos_mode      disabled N/A                                                               True
queue_size    8192     Queue size for a SEA thread                                       True
real_adapter  ent0     Physical adapter associated with the SEA                          True
send_RARP     yes      Transmit Reverse ARP after HA takeover                            True
thread        1        Thread mode enabled (1) or disabled (0)                           True
virt_adapters ent2     List of virtual adapters associated with the SEA (comma separated) True
```

3. The data network Virtual Ethernet Adapter was without VLAN tagging configuration and using VLAN ID 2, as shown in Figure 4-81.

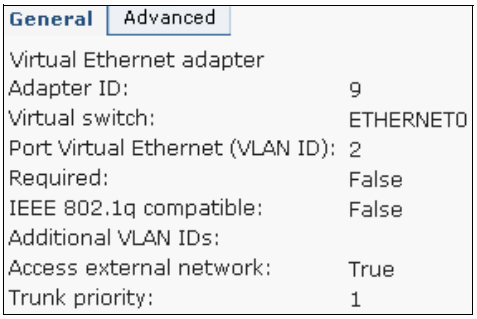


Figure 4-81 Virtual Ethernet Adapter for the data network

4. The Shared Ethernet Adapter was created using a physical Ethernet adapter (ent5) connected to a physical Ethernet switch (IEEE 802.1Q was not configured in the physical switch port), and a SEA was created as shown in Example 4-113.

Example 4-113 SEA configuration for the data network in the VIOS

```
$ lsmap -all -net
SVEA   Physloc
------ --------------------------------------------
ent9   U8233.E8B.061AA6P-V1-C9-T1

SEA                  ent10
Backing device       ent5
Status               Available
Physloc              U78A0.001.DNWHZWR-P1-C3-T1

$ lsdev -dev ent10 -attr
attribute     value    description
user_settable
accounting    disabled Enable per-client accounting of network statistics                True
adapter_reset yes      Reset real adapter on HA takeover                                 True
ctl_chan              Control Channel adapter for SEA failover                          True
gvrp          no       Enable GARP VLAN Registration Protocol (GVRP)                     True
ha_mode       disabled High Availability Mode                                            True
hash_algo     0        Hash algorithm used to select a SEA thread                        True
jumbo_frames  no       Enable Gigabit Ethernet Jumbo Frames                              True
large_receive no       Enable receive TCP segment aggregation                            True
largesend     1        Enable Hardware Transmit TCP Resegmentation                       True
lldpsvc       no       Enable IEEE 802.1qbg services                                     True
netaddr       0        Address to ping                                                   True
nthreads      7        Number of SEA threads in Thread mode                              True
pvid          2        PVID to use for the SEA device                                    True
pvid_adapter  ent9     Default virtual adapter to use for non-VLAN-tagged packets        True
qos_mode      disabled N/A                                                               True
queue_size    8192     Queue size for a SEA thread                                       True
real_adapter  ent5     Physical adapter associated with the SEA                          True
send_RARP     yes      Transmit Reverse ARP after HA takeover                            True
thread        1        Thread mode enabled (1) or disabled (0)                           True
virt_adapters ent9     List of virtual adapters associated with the SEA (comma separated) True
```

5. The Controller node, which in the cloud setup for this environment is also running in the Power System Server, uses a Virtual Ethernet Adapter for the management network configured with Port Virtual Ethernet (VLAN ID) 1, as shown in Figure 4-82.

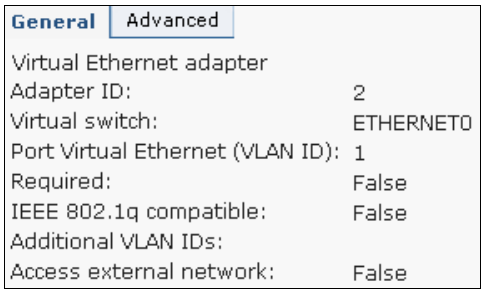


Figure 4-82 Virtual Ethernet Adapter for management network in Controller node

6. For the data network, the Controller node has a physical adapter, which is also connected to a port in the physical switch without VLAN tagging (IEEE 802.1Q) as shown in Figure 4-83.

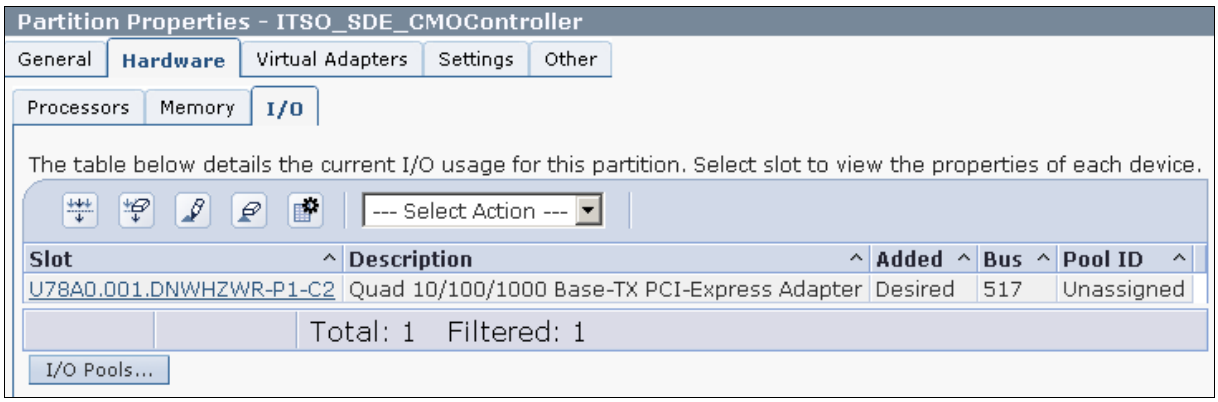


Figure 4-83 Physical adapter for data network in the Controller node

Network configuration in KVM and PowerKVM Compute nodes

Given that Cloud Manager is using VLAN networks, but the physical switch does not have VLAN tagging enabled, it was required to manually configure the Open vSwitch on KVM and PowerKVM Compute nodes to tag the packets going through the physical interface with the same VLAN ID as used in the private network. The command used for this configuration is the same for KVM and PowerKVM Compute nodes, but given that the network interface name convention is different, the following commands were used:

- For KVM Compute nodes:

```
# ovs-vsctl set port eth1 tag=2
```

- For PowerKVM Compute nodes:

```
# ovs-vsctl set port enP3p1s0f1 tag=2
```

After completing this step, the ovs-vsctl show command should show that the physical adapter is tagged with VLAN ID 2:

- For KVM Compute nodes:

```
Bridge "br-eth1"
        Port "br-eth1"
            Interface "br-eth1"
                type: internal
        Port "phy-br-eth1"
            Interface "phy-br-eth1"
        Port "eth1"
            tag: 2
            Interface "eth1"
```

- For PowerKVM Compute nodes:

```
Bridge br-ext
        Port "enP3p1s0f1"
            tag: 2
            Interface "enP3p1s0f1"
        Port br-ext
            Interface br-ext
                type: internal
        Port phy-br-ext
            Interface phy-br-ext
```

Network configuration in IBM Cloud Manager

After completing the cloud deployment, the two networks available in PowerVC were imported to the Cloud Manager. Figure 4-84 shows the networks configured in the PowerVC server in the cloud setup for this publication.

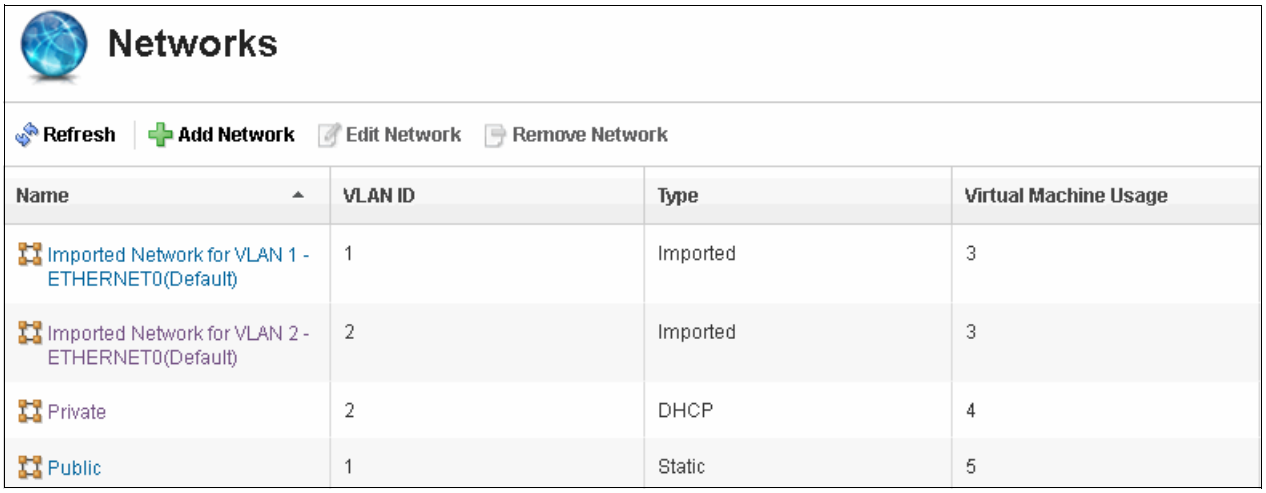


Figure 4-84 PowerVC network configuration

Figure 4-85 shows the networks from PowerVC imported in Cloud Manager.

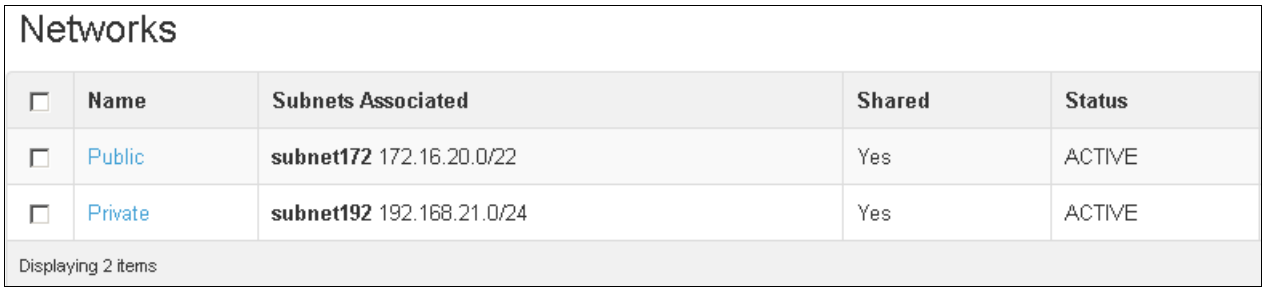


Figure 4-85 Networks from PowerVC imported in IBM Cloud Manager

The public network is only used for management purposes, while the private network is the data network. The private network was modified to enable DHCP server, as shown in Figure 4-86.

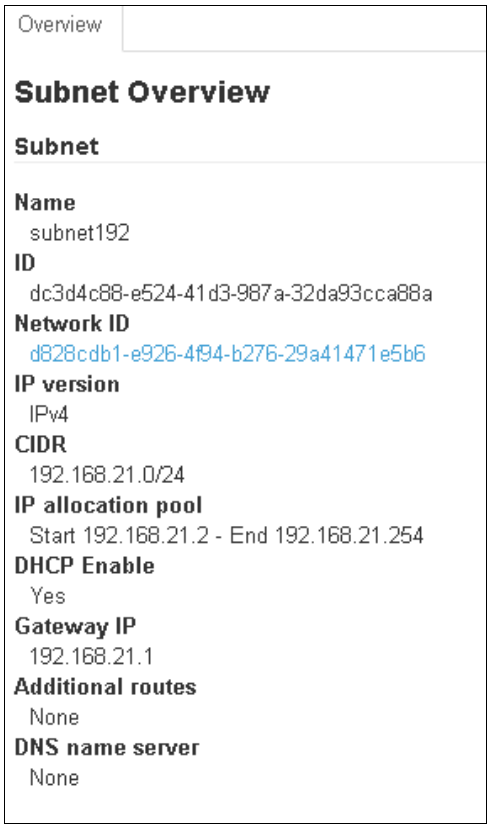


Figure 4-86 Private network modified to enable DHCP

When instances are deployed in the PowerVM Compute nodes, both networks are provided. The public network is used exclusively for communication with the Hardware Management Console (which is not available in the private network), so we can keep RMC communication and allow hardware migration using Live Partition Mobility. The private network is used for communication between instances from PowerVM, PowerKVM and KVM Compute nodes. Figure 4-87 on page 356 shows both networks being selected while deploying a PowerVM instance.

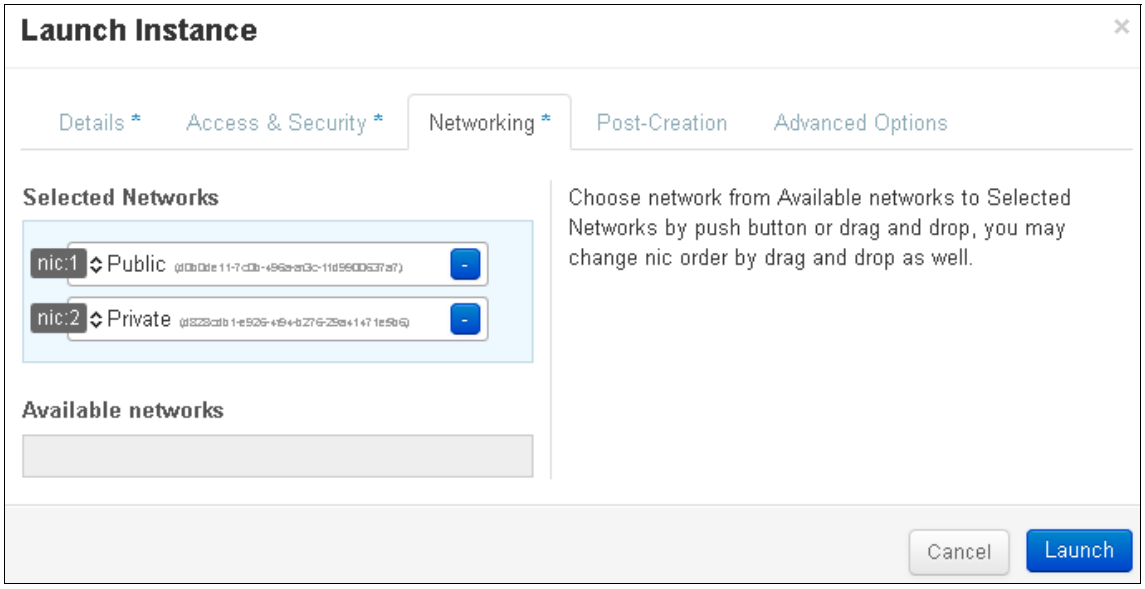


Figure 4-87 Selecting both networks for PowerVM instances

Given that only the data network (private network) is available for instances in the KVM and PowerKVM Compute nodes (because these nodes have only one Open vSwitch network bridge using the physical adapter connected to the private or data network), while deploying instances in KVM or PowerKVM Compute nodes, only the private network is selected, as shown in Figure 4-88.

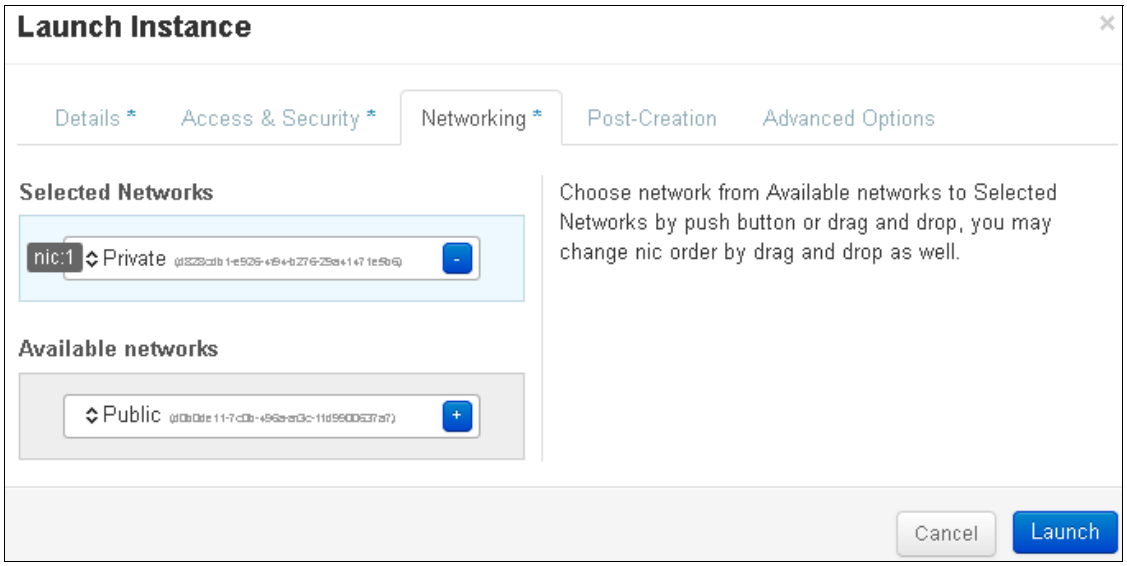


Figure 4-88 Selecting the private network for KVM and PowerKVM instances

4.11 Storage configuration in IBM Cloud Manager

The Cloud Manager component that handles Storage connection is called *Cinder*. It can be described as a block storage service for Cloud Manager, responsible for virtualizing storage devices that can be used by instances running in Compute nodes. Cinder integrates with several block storage components and provides an interface to the user to request and use these block storages transparently, without requiring any knowledge of where the storage is deployed, what type of device is being used or how to create the block storage volume.

The volumes created through Cinder are presented to the instances as additional disks. For example, if you deploy a Linux instance, it will have a single disk where the operating system is installed (usually `/dev/vda` for virtual disks). If the application that you need to install in this instance requires additional disk space, or if you want to separate the application installation path and data path from the operating system installation, you need to provide an additional disk (in this case, `/dev/vdb`) to complete this task. Cinder can transparently use several block storages as software-defined storage to complete this task, without requiring any knowledge on configuring these block storages from the cloud user.

In the following sections, you will be able to integrate Cinder with these elements:

- Logical Volume Manager (LVM)
- IBM SAN Volume Controller (SVC)
- IBM Power Virtualization Center (PowerVC)
- IBM General Parallel File System (GPFS)
- Network File System (NFS)

Important: This section will guide you to configure Cinder with multiple storage backends. To accomplish these configurations, the `/etc/cinder/cinder.conf` file in the Controller node will be modified multiple times. Consider keeping a backup of this configuration file, because if you use the Chef server (deployment server) to modify configuration in your Controller node, this file might be overwritten.

For more information about configuring Cinder with multiple storage backends, see the Chapter 6 from IBM Cloud Manager with OpenStack 4.1 Administrator Guide under the topic Configuring multiple block storage backends:

`http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html`

4.11.1 Integrate Cinder with Logical Volume Manager

To integrate Cinder with Logical Volume Manager (LVM), a volume group is created in the Controller node. When you create a new volume to provide to an instance, a logical volume is created inside this volume group. Then, the Controller node presents this logical volume as a block disk using iSCSI protocol to the Compute node where the instance is running, and the Compute node presents it as a raw block device to the instance. LVM volumes will be available to all Compute nodes which are able to access the Controller node using iSCSI (in the cloud setup for this publication, KVM and PowerKVM). The following steps will allow you to integrate Cinder with Logical Volume Manager:

1. Log in to your Controller node and choose which disk (or disks) that you will use to create the Cinder volume group.

Note: In the cloud setup for this publication, we created a volume in IBM SAN Volume Controller Storage and presented to the Controller node. After rescanning the SCSI bus with the `rescan-scsi-bus.sh` script and correctly configuring `multipath` to handle multiple paths, the volume name used is `/dev/mapper/mpathc`, but you can also use internal disks, such as `/dev/sdb` to accomplish this objective.

2. Use the **`pvcreate`** command to create the physical volume that will be used by the volume group, as shown in Example 4-114.

Example 4-114 Creating the physical volume

```
[root@cmocontroller ~]# pvcreate /dev/mapper/mpathc
Physical volume "/dev/mapper/mpathc" successfully created
```

3. Use the **`vgcreate`** command to create the volume group that will be used by Cinder to provide volumes for the instances, as shown in Example 4-115.

Example 4-115 Creating the Cinder volume group

```
[root@cmocontroller ~]# vgcreate cinder-volumes /dev/mapper/mpathc
Volume group "cinder-volumes" successfully created
```

4. Edit the `/etc/cinder/cinder.conf` file in the Controller node:

```
# vi /etc/cinder/cinder.conf
```

5. Comment the `iscsi_ip_address`, `volumes_dir` and `volume_driver` lines in this file using the `#` symbol, as shown in Example 4-116.

Example 4-116 Commenting lines in the cinder.conf file

```
#iscsi_ip_address=127.0.0.1
iscsi_port=3260
iscsi_helper=tgtadm
#volumes_dir=/var/lib/cinder/volumes
#volume_driver=powervc.volume.driver.powervc.PowerVCDriver
```

Note: To simplify the `cinder.conf` file used in this publication, all the pre-existing comments in the original file were deleted.

6. Now you must add or modify the `enabled_backends` line to include the backend which you plan to add to your Cinder configuration. The `enabled_backends` line is used to allow Cinder to connect with multiple backends. In the next topic, you will include the backend section. Example 4-117 shows a valid `enabled_backends` configuration.

Example 4-117 Valid enabled_backends line in cinder.conf

```
enabled_backends=lvm_driver
```

Important: The original `/etc/cinder/cinder.conf` file does not have an `enabled_backends` line. You must add this line below the `[DEFAULT]` section, usually at the end of the file, but before including the storage backends.

The name used in the `enabled_backends` line, such as `lvm_driver` is not a specific string. Consider using a name which informs you what kind of configuration is related to that backend. The same name must be used while creating the backend in step 7.

If you have already configured more than one backend in your /etc/cinder/cinder.conf file, the enabled_backends line should include all the backends and the new backend that you are adding. The enabled_backends allows you to specify multiple backends using comma-separated values, as shown in Example 4-118.

Example 4-118 Specifying multiple storage backends in the /etc/cinder/cinder.conf file

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,nfs-cmo
```

7. Now you need to create the LVM backend, using the same name you specified in step 6 on page 358. The backend name should be placed between square brackets [ ] and the parameters for that backend should be added in the following lines. For the LVM integration, the following parameters are required:

   volume_driver The Cinder driver name that will be used in this backend. For LVM, use the cinder.volume.drivers.lvm.LVMISCSIDriver.

   volume_backend_name The name for the volume backend that you will use later, when creating a Cinder volume type and associating it with the backend. This parameter can be any name, but it is suggested to specify something that can remind you of what kind of storage this backend is using (for example LVM_iSCSI).

   volume_group The name of the Cinder volume group created in step on page 358.

   iscsi_ip_address The IP address of the Controller node.

   Example 4-119 shows how you can add the LVM backend to the /etc/cinder/cinder.conf file.

   Example 4-119 Adding the LVM backend to the cinder.conf

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,nfs-cm
o

[lvm_driver]
volume_driver=cinder.volume.drivers.lvm.LVMISCSIDriver
volume_backend_name=LVM_iSCSI
volume_group=cinder-volumes
iscsi_ip_address=172.16.21.137
```

8. Restart the openstack-cinder-volume service to allow it to read the configuration file:

```
# service openstack-cinder-volume restart
```

9. Now you need to create a volume type and associate it with the LVM volume_backend_name parameter to allow the cloud user to create LVM volumes in the web interface. Run the following command to create a type called *lvm*:

```
# source /root/openrc
# cinder type-create lvm
```

10. Associate the volume type created in step 9 with the volume_backend_name specified in the LVM backend in step 7. Run the following command to associate the lvm volume type with the LVM_iSCSI volume_backend_name:

```
# cinder type-key lvm set volume_backend_name=LVM_iSCSI
```

Important: At the time of writing this publication, Cloud Manager has a defect where the **cinder type-key** command may fail with the following error if you use the volume-type name:

```
ERROR: list() got an unexpected keyword argument 'search_opts'
```

The `volume-type ID` should be used instead, as shown in the following example:

1. Use the **cinder type-list** command to obtain the volume-type ID:

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# cinder type-list
+--------------------------------------+-------------------------------+
|                  ID                  |              Name             |
+--------------------------------------+-------------------------------+
| 230e9aa5-c01d-49df-9d2e-55fb34983ec0 |          gpfs-fast            |
| 74193370-aa30-40e5-bcd4-a7f9b17582c1 | pvc:172_16_21_70:SVC default  |
| 76c8414a-d2d9-4191-b15e-3ea8025b7b66 |              lvm              |
| bc87ed7f-c2b7-44e3-b55f-66026ca6f6d6 |           nfs-cmo             |
| c833de5b-352d-4517-bbd5-fa02e5c5280e |          SVC_Cinder           |
| e659f0bc-5572-4b32-8179-ca7e7e3207ac |            lvmnew             |
+--------------------------------------+-------------------------------+
```

2. Run the **cinder type-key** command using the volume ID, as shown in the following example:

```
# cinder type-key 76c8414a-d2d9-4191-b15e-3ea8025b7b66 set
volume_backend_name=LVM_iSCSI
```

This problem is described in the following defect report:

https://bugs.launchpad.net/python-cinderclient/+bug/1248519

3. With this configuration, you are now able to create `lvm` volumes using the Cloud Management dashboard. Log in to the Cloud Management dashboard by using the `admin` user ID.
4. Click **Project > Compute > Volumes**, as shown in Figure 4-89 on page 361.

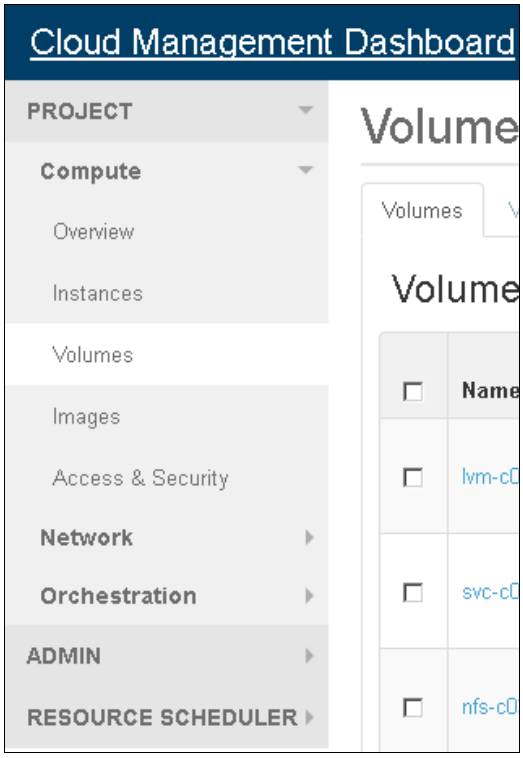


Figure 4-89 Accessing the Volumes interface in the Cloud Management dashboard

5. Click **Create Volume** and provide a volume name, a description (optional), select **lvm** as the `type`, and specify the size of the volume in Gigabytes, as shown in Figure 4-90 on page 362.

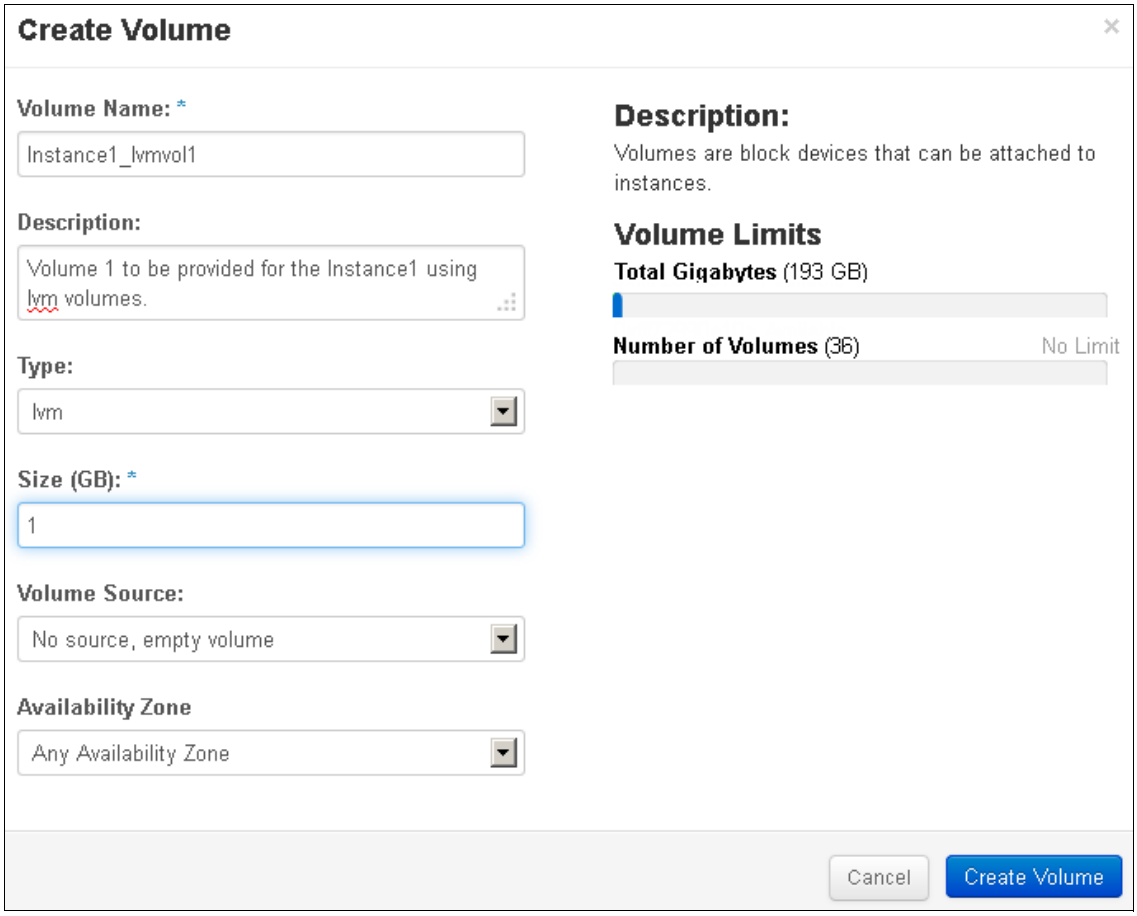


Figure 4-90 Creating an LVM volume

Note: In the Volume Source field, you can use existing images as source for the volumes (the volumes will be populated with the content of the image), but only the creation of empty volumes is covered by this publication.

In the Availability Zone, you can specify that the volume will be available for a limited number of Compute nodes. In this case you need to create host aggregates. This option is not covered in this book.

6. Now, select the volume you created in step 5 on page 361 and click **More > Edit Attachments**. Select the instance that will access this volume in the Attach to Instance field, provide the device name, and click **Attach Volume**, as shown in Figure 4-91 on page 363.

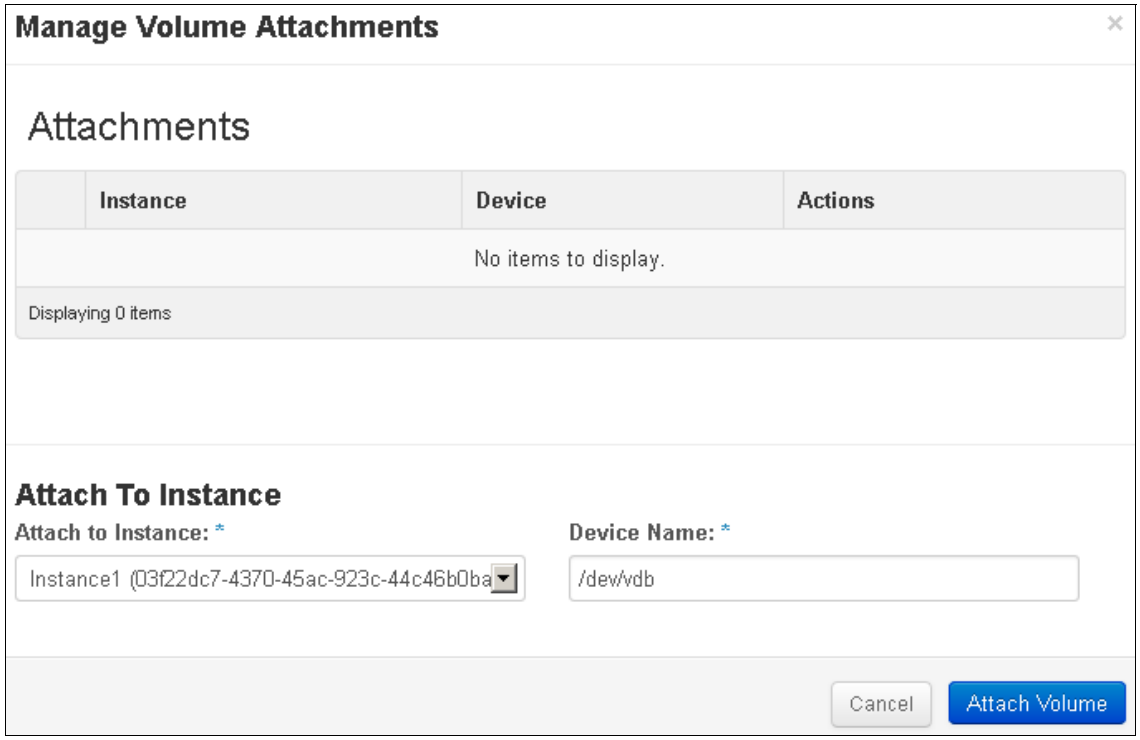


Figure 4-91 Attaching a volume to an instance

Note: Make sure that the Device Name specified in the Cloud Management dashboard is not being used by the instance. You can check this information using the **`fdisk -l`** command in the Linux instances. The device naming convention may be different according to the hypervisor where the instance is running.

7. When the attach operation is completed, you should see the appropriate information in the `Attached To` column as shown in Figure 4-92.

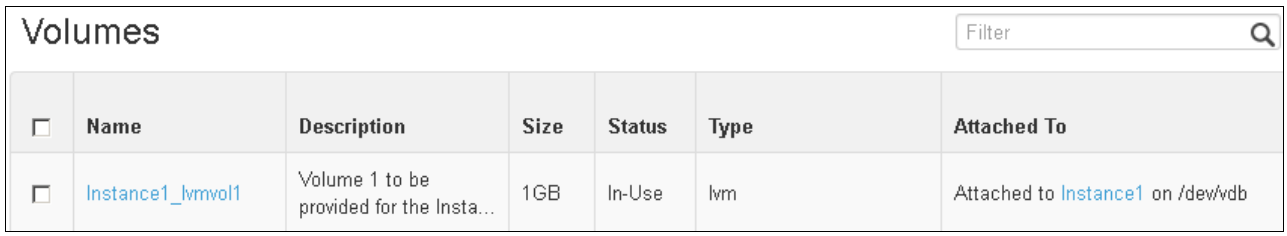


Figure 4-92 Volume attached to an instance

8. Log in to the instance which had the volume attached and use the **fdisk -l** command to identify the new volume (using the name you previously specified), as shown in Example 4-120.

Example 4-120 Using the fdisk -l command in the instance to identify the volume

```
[root@host-192-168-21-60 ~]# fdisk -l

Disk /dev/vda: 21.5 GB, 21474836480 bytes
16 heads, 63 sectors/track, 41610 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xd99d0900

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *           3          11        4096   41  PPC PReP Boot
Partition 1 does not end on cylinder boundary.
/dev/vda2              11        1027      512000   83  Linux
Partition 2 does not end on cylinder boundary.
/dev/vda3            1027       20806     9968640   8e  Linux LVM
Partition 3 does not end on cylinder boundary.

Disk /dev/vdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/mapper/VolGroup-lv_root: 6043 MB, 6043992064 bytes
255 heads, 63 sectors/track, 734 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/mapper/VolGroup-lv_swap: 4160 MB, 4160749568 bytes
255 heads, 63 sectors/track, 505 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

9. Back on the Cloud Management dashboard, click **Admin > System Panel > Instances** and note which Compute node your instance is running, as shown in Figure 4-93.

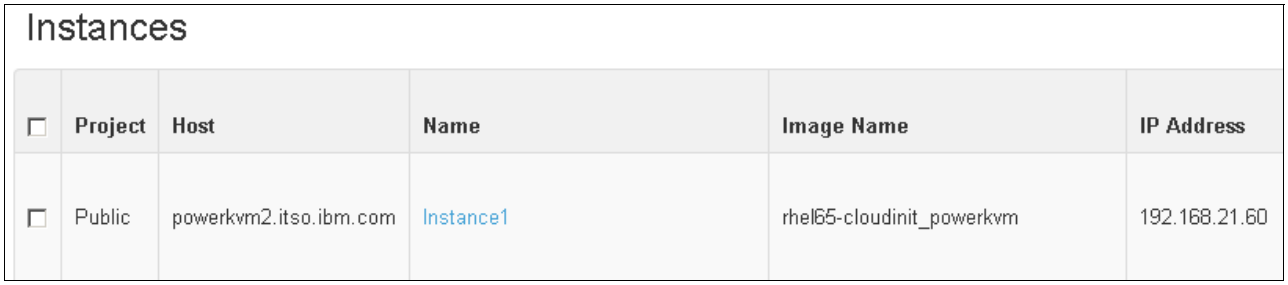


Figure 4-93 Finding the Compute node where the instance is running

10. Click the instance name and note its `ID`, as shown in Figure 4-94.

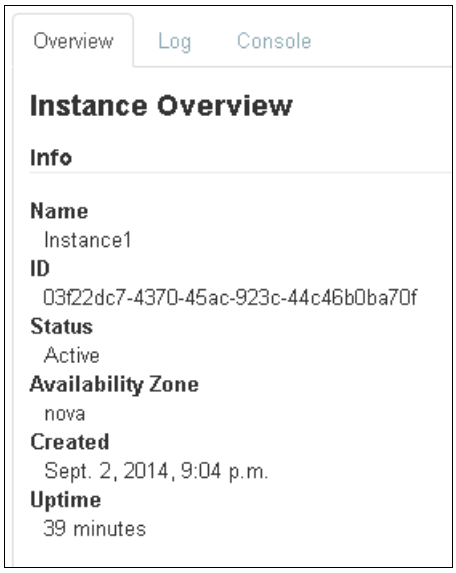


Figure 4-94 Note the instance ID

11. Now click **Admin > System Panel > Hypervisors** and click the Compute node where the instance is running (which you obtained in step 9). Use the instance ID to find the instance name in the Compute node where it is running. In the example used in this publication, instance ID is `03f22dc7-4370-45ac-923c-44c46b0ba70f`, which means instance name is `instance-00000280`, as shown in Figure 4-95 on page 366.

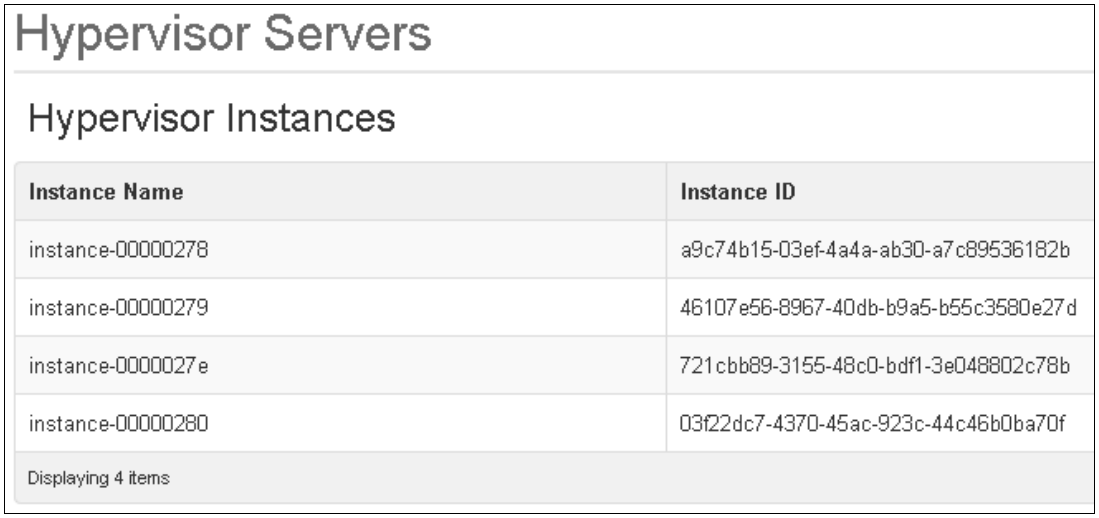


Figure 4-95 Obtaining the instance Name in the Compute node

12. Log in to the Compute node where the instance is running (in this example, `powerkvm2.itso.ibm.com`) and use the **`virsh dumpxml <instance_name>`** command to identify the volume that was created in this section:

    ```
    # virsh dumpxml instance-00000280
    ```

    Example 4-121 shows the section of the **`virsh dumpxml`** command where the lvm volume is shown.

Example 4-121 LVM volume in PowerKVM Compute node instance

```
<disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none'/>
      <source
dev='/dev/disk/by-path/ip-172.16.21.137:3260-iscsi-iqn.2010-10.org.openstack:volume-d5f34818-baa
9-4aae-ae0d-b9d64e43ba15-lun-1'/>
      <target dev='vdb' bus='virtio'/>
      <serial>d5f34818-baa9-4aae-ae0d-b9d64e43ba15</serial>
      <alias name='virtio-disk1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
</disk>
```

13. You can see that the volume was presented to the Compute node by using the **`fdisk -l`** command, as shown in Example 4-122.

Example 4-122 Checking the volume in the Compute node using the fdisk command

```
[root@powerkvm2 ~]# fdisk -l
/dev/disk/by-path/ip-172.16.21.137:3260-iscsi-iqn.2010-10.org.openstack:volume-d5f34818-baa9-4aa
e-ae0d-b9d64e43ba15-lun-1

Disk
/dev/disk/by-path/ip-172.16.21.137:3260-iscsi-iqn.2010-10.org.openstack:volume-d5f34818-baa9-4aa
e-ae0d-b9d64e43ba15-lun-1: 1073 MB, 1073741824 bytes, 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
[root@powerkvm2 ~]# ls -l
/dev/disk/by-path/ip-172.16.21.137:3260-iscsi-iqn.2010-10.org.openstack:volume-d5f34818-baa9-4aa
e-ae0d-b9d64e43ba15-lun-1
lrwxrwxrwx. 1 root root 9 Sep  2 17:10
/dev/disk/by-path/ip-172.16.21.137:3260-iscsi-iqn.2010-10.org.openstack:volume-d5f34818-baa9-4aa
e-ae0d-b9d64e43ba15-lun-1 -> ../../sdh

[root@powerkvm2 ~]# fdisk -l /dev/sdh

Disk /dev/sdh: 1073 MB, 1073741824 bytes, 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Now you can use the volume in the instance to create the file systems and populate it with application data.

4.11.2 Integrate Cinder with SAN Volume Controller or Storwize V7000

To integrate Cinder with IBM SAN Volume Controller (SVC) or Storwize V7000, the Compute node that hosts the instances that will access the SVC or V7000 volumes must have access to the storage area network (SAN) and to the storage subsystem. When a new volume is provided to an instance, a volume (LUN) is created in the SVC or V7000 and presented to the Compute node (physical host) using a Fibre Channel connection (SAN). The Compute node then presents this volume as a block device to the instance. SVC or V7000 volumes are available to instances running in all Compute nodes with access to the storage area network (in the cloud setup for this publication, only KVM Compute nodes). The steps that follow explain how to integrate Cinder with SVC or V7000 storage subsystem.

Important: Cinder integration with SVC or V7000 requires manual zoning configuration between the storage subsystem and the Compute nodes that will run instances using SVC or V7000 volumes. If the Controller is an `ibm-os-single-controller-node` (dedicated for the Controller role, where no instance will run), it does not need a Fibre Channel connection and zoning to the storage subsystem. But if the Controller node is an `ibm-os-allinone-kvm` (Controller + KVM Compute node role) and the instances that will run the Controller + Compute node need access to the SVC or V7000 volumes, zoning and a Fibre Channel connection to the storage subsystem are required.

In the cloud setup for this publication, the Controller node is performing the `ibm-os-single-controller-node`, so no zoning is required between the Controller and SVC. The KVM Compute nodes use SVC volumes, so this publication shows how to set up zoning between the KVM Compute node and the SVC.

Note: The following instructions for creating zones are based on an IBM SAN Switch 2498-B40 running FOS V7.1.1. If you have a different SAN Switch, use the instructions from your SAN Switch vendor to create zoning between Compute nodes and SVC or V7000.

1. Log in to your Compute node using the root user ID and run the **systool** command to identify the WWPN of the HBAs, as shown in Example 4-123.

Example 4-123 Using systool to identify the HBA WWPN

```
[root@cmo1 ~]# systool -c fc_host -v | grep port_name
    port_name           = "0x10000000c9809b6a"
    port_name           = "0x10000000c9809b6b"
```

Note: To use the **systool** command, you need to install the `sysfsutils` package. Run the following command to install it:

```
# yum install sysfsutils
```

2. Log in to your SAN switch web tool by using the SAN switch IP address. Use the `admin` user ID and its password:

   `http://<switch_ip_address>/switchExplorer.html`

3. Click **Configure> Zone Admin**.
4. Click **New Alias** and provide an alias name to identify the HBA WWPN of the Compute node, as shown in Figure 4-96.

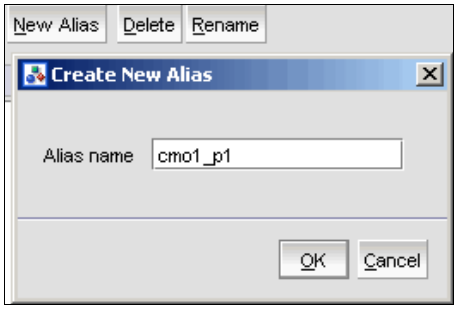


Figure 4-96 Creating an alias for the Compute node HBA WWPN

5. In the Member Selection List, select the HBA WWPN from the Compute node, which was identified in step 1, and click **Add member(s) to the Alias** (arrow pointing to the right) to add the WWPN to the alias, as shown in Figure 4-97.

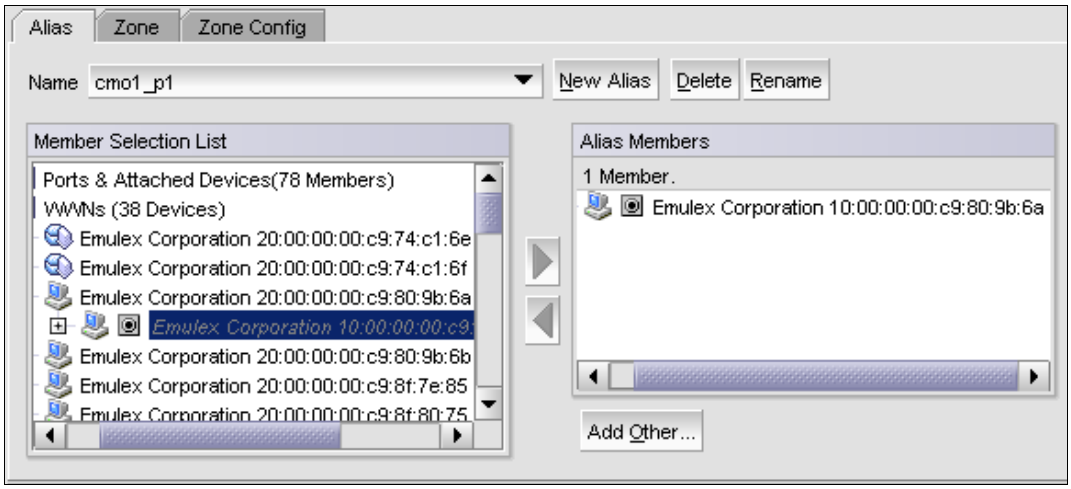


Figure 4-97 Adding WWPN to the alias

Important: Repeat the steps for all appropriate ports, including all ports from each Compute node and all ports from the SVC or V7000. Each alias should contain a single WWPN.

6. Click the **Zone** tab and click **New Zone**. Provide the Zone Name, as shown in Figure 4-98.

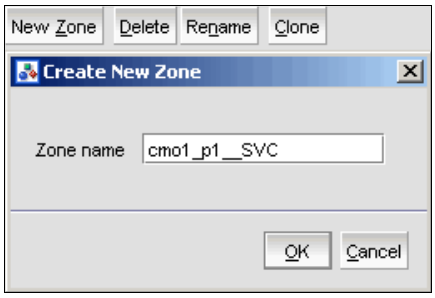


Figure 4-98 Creating a zone

7. In the Member Selection List area, select the aliases that will be part of the zone (one port from the Compute node and the number of ports that you want from the storage subsystem). Click **Add member(s) to the zone** (arrow points to the right) to add them to the **Zone members area**, as shown in Figure 4-99.

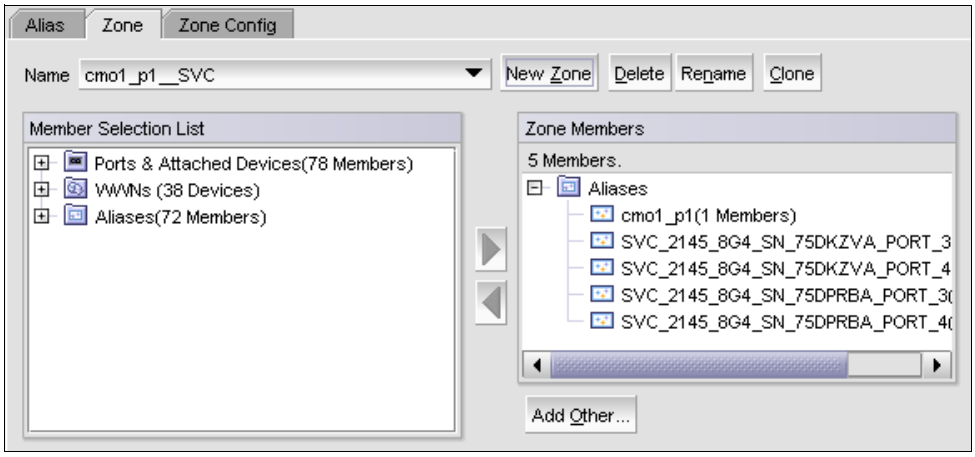


Figure 4-99 Adding aliases to the zone

Note: Review your storage subsystem documentation to determine the recommended number of ports to include in a single zone.

Repeat these steps to create all required zones for all Compute nodes that will run instances using the SVC or V7000 volumes.

8. In the Zone Config tab, select your zone in the Member Selection List area and add to your config by using **Add member(s) to the zone config** (arrow pointing to the right), as shown in Figure 4-100 on page 370.

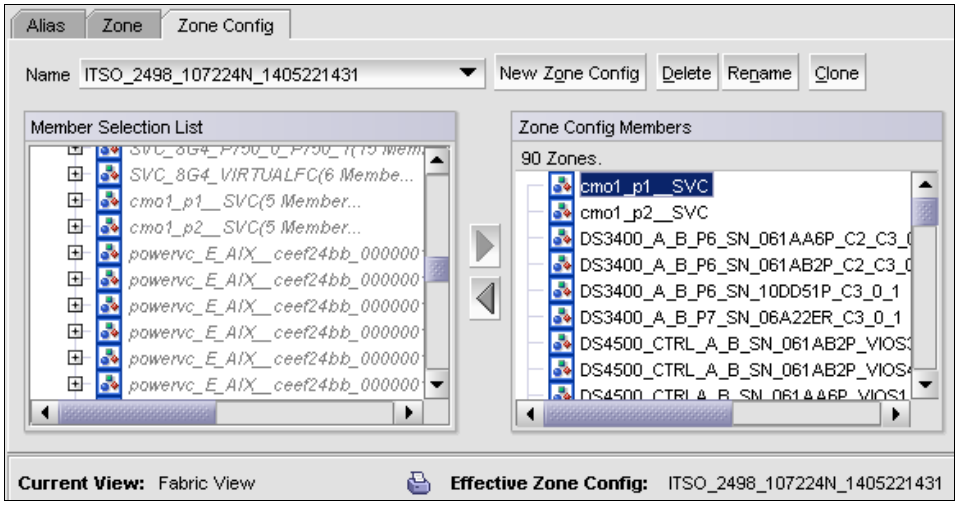


Figure 4-100 Adding the aone to the config

Note: Repeat these steps to add all of the appropriate zones to the Zone Config.

9. Click **Save Config**, and wait until that task is finished.
10. Click **Enable Config**, and wait until the task is complete.
11. Log in to the SVC or V7000 storage subsystem using its IP address. Use the superuser user name and its password:

    `https://<svc_ip_address>/login`

12. Click **Hosts > Hosts** and click **New Host**.
13. Select **Fibre Channel Host** and provide a Host Name to identify your Compute node. In the Fibre Channel Ports area, select all your Compute node WWPNs from the list (that you want to use for the SVC or V7000 volumes) and click **Add Port to List**. After completing that, click **Create Host**, as shown in Figure 4-101 on page 371.

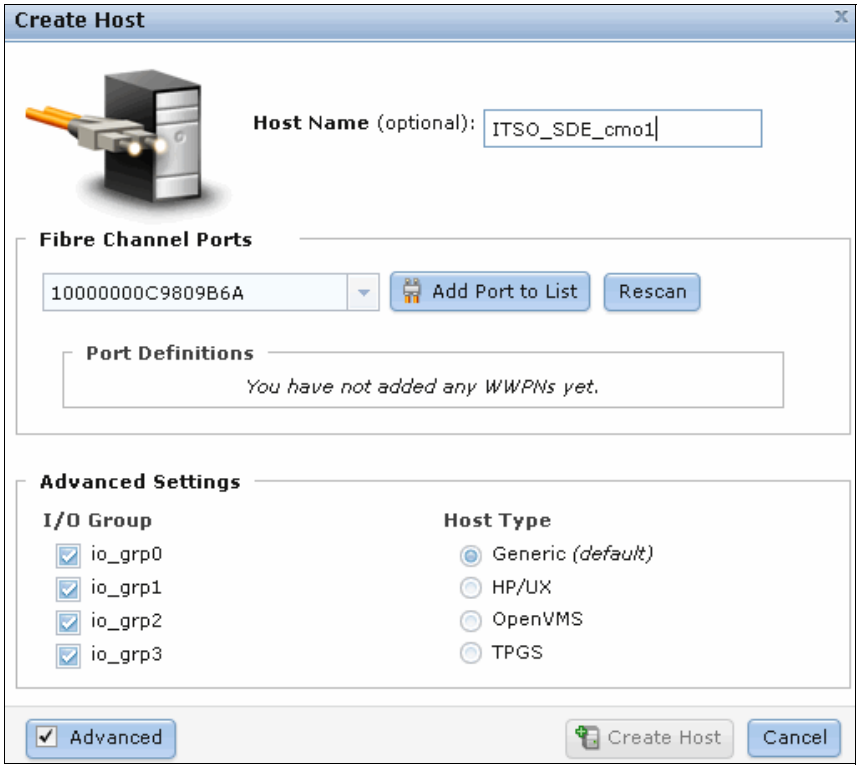


Figure 4-101 Creating a host in SAN Volume Controller

> **Important:** Repeat these steps for all Compute nodes which will have instances using the SVC or V7000 volumes.

14. Click **Pool > Volumes by Pool** and identify the storage pool that you plan to use to create Cinder volumes. Make a note of the pool name, because you will need to provide this information in the `/etc/cinder/cinder.conf` configuration file in the Controller node.

> **Important:** Before proceeding, make sure your Compute nodes which will access the SVC or V7000 volumes have appropriate configuration in `multipathd` to handle multiple paths. For instructions on how to configure multipathd with SVC and V7000 disks, see the following web pages:
>
> Configuring Device Mapper Multipath (DMMP) tool for hosts running the Linux operating system
>
> http://ibm.co/1ERPItO
>
> Settings for Linux hosts
>
> http://ibm.co/1GGdtSl

15. Now that the SAN Switch and SVC requirements are met, you can configure the Cinder device in the Controller node. Log in to your Controller node and edit the `/etc/cinder/cinder.conf` file:

```
# vi /etc/cinder/cinder.conf
```

16. Comment the `iscsi_ip_address`, `volumes_dir` and `volume_driver` lines in this file using the # symbol, as shown in Example 4-124.

Example 4-124 Commenting appropriate lines in the cinder.conf file

```
#iscsi_ip_address=127.0.0.1
iscsi_port=3260
iscsi_helper=tgtadm
#volumes_dir=/var/lib/cinder/volumes
#volume_driver=powervc.volume.driver.powervc.PowerVCDriver
```

Note: To simplify the cinder.conf file used in this publication, all the pre-existing comments in the original file were deleted.

17. Add or modify the `enabled_backends` line to include the backend which you plan to add to your Cinder configuration. The `enabled_backends` line is used to allow Cinder to connect with multiple backends. In the next topic, you will include the backend section. Example 4-125 shows a valid `enabled_backends` configuration.

Example 4-125 Valid enabled_backends line in cinder.conf

```
enabled_backends=storwize_svc_driver
```

Important: The original `/etc/cinder/cinder.conf` file does not have an `enabled_backends` line. You must add this line below the `[DEFAULT]` section, usually at the end of the file, but before including the storage backends.

The name used in the `enabled_backends` line, such as `storwize_svc_driver`, does not have to be any specific word. Consider using a name that informs you about what kind of configuration is related to that backend. The same name must be used while creating the backend in step 18.

If you have already configured more than one backend in your `/etc/cinder/cinder.conf` file, the `enabled_backends` line must include all the backends and the new backend that your are adding. The `enabled_backends` allow you to specify multiple backends using comma-separated values, as shown in Example 4-126.

Example 4-126 Valid enabled_backends line with multiple backends in cinder.conf file

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,nfs-cmo
```

18. Now you need to create the SVC or V7000 backend, using the same name specified in step 17. The backend name should be placed between square brackets `[ ]` and the parameters for that backend should be added in the following lines. For the SVC or V7000 integration using FC connection, the following parameters are required:
    - `volume_driver`: The Cinder driver name that will be used in this backend. For SVC or V7000, use the `cinder.volume.drivers.ibm.storwize_svc.StorwizeSVCDriver`.
    - `volume_backend_name`: The name for the volume backend that you will use later when creating a Cinder volume type and associating it with the backend. This parameter can be any name, but consider using your storage subsystem name to help you identify this backend (in the cloud setup for this publication, the storage name was used: `ITSO_SVC_8G4`).
    - `san_ip`: The SVC or V7000 management IP or host name.
    - `san_login`: The SVC or V7000 management login username (usually `superuser`).

- `san_password`: The SVC or V7000 management login password.
- `storwize_svc_volpool_name`: The SVC or V7000 pool that will be used for creating volumes (which you obtained in step 14 on page 371).
- `storwize_svc_connection_protocol`: Connection protocol to use to provide the volumes for the Compute nodes (at the time of writing, FC or iSCSI is supported, but only FC is covered in this publication).
- `storwize_svc_iscsi_chap_enabled`: CHAP authentication for iSCSI connections (in this publication, this configuration was set to false, given that iSCSI is not being used).
- `storwize_svc_multipath_enabled`: Enable multipath for FC connections.

Example 4-127 shows how you can add SVC or V7000 backend to the `/etc/cinder/cinder.conf` file.

Example 4-127 Adding the storwize_svc_driver backend to the cinder.conf file

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,nfs-cm
o
.
. <some lines from other backends were removed>
.
[storwize_svc_driver]
volume_driver=cinder.volume.drivers.ibm.storwize_svc.StorwizeSVCDriver
volume_backend_name=ITSO_SVC_8G4
san_ip=172.16.21.70
san_login=superuser
san_password=itso13sj
storwize_svc_volpool_name=ITSO_SDE_ninja_pool
storwize_svc_connection_protocol=fc
storwize_svc_iscsi_chap_enabled=false
storwize_svc_multipath_enabled=true
```

Note: There are additional parameters that you can provide for SVC or V7000 connections in the backend section. For more information of available parameters, see the OpenStack Documentation:

http://bit.ly/1ERZWJS

19. Restart the `openstack-cinder-volume` service to allow it to read the configuration file:

```
# service openstack-cinder-volume restart
```

20. Now you need to create a volume type and associate it with the SVC or V7000 `volume_backend_name` parameter to enable the cloud user to create SVC or V7000 volumes in the web interface. Run the following command to create a volume type called `SVC_Cinder`:

```
# source /root/openrc
# cinder type-create SVC_Cinder
```

21. Associate the `SVC_Cinder` volume type with the `volume_backend_name` specified for the SVC in step 18 on page 372. Run the following command to associate the `SVC_Cinder` volume type with the `ITSO_SVC_8G4 volume_backend_name`:

```
# cinder type-key SVC_Cinder set volume_backend_name=ITSO_SVC_8G4
```

Important: Review the problem described in step 10 on page 359.

22. With this configuration, you can now create `SVC_Cinder` volumes by using the Cloud Management dashboard. Log in to the Cloud Management dashboard by using the `admin` user ID, and then complete these steps:

    a. Click **Project > Compute > Volumes**, and then click **Create Volume**.
    b. Provide a volume name and (optionaly) a description.
    c. Select **SVC_Cinder** as the type.
    d. Specify the size of the volume in Gigabytes.
    e. Click **Create Volume**, as shown in Figure 4-102.

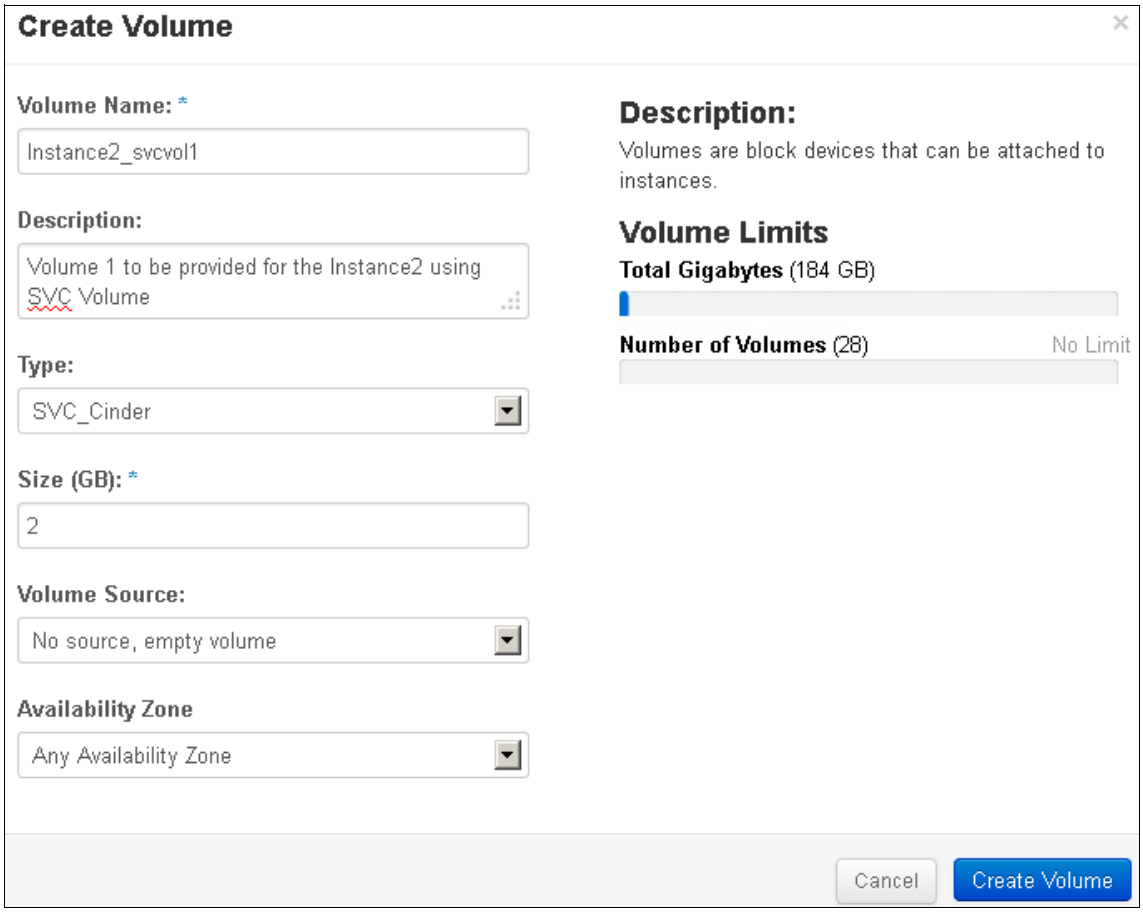


Figure 4-102 Creating the SVC volume

> **Note:** In the Volume Source field, you can use existing images as source for the volumes (the volumes will be populated with the content of the image), but only the creation of empty volumes is covered by this publication.
>
> In the Availability Zone, you can specify that the volume will be available for a limited number of Compute nodes. In this case, you need to create Host Aggregates. This option is not covered by this publication.

23. Now, select the volume you just created and click **More > Edit Attachments**. In the Attach to Instance field, select the instance that will access this volume, provide the device name, and click **Attach Volume**, as shown in Figure 4-103 on page 375.

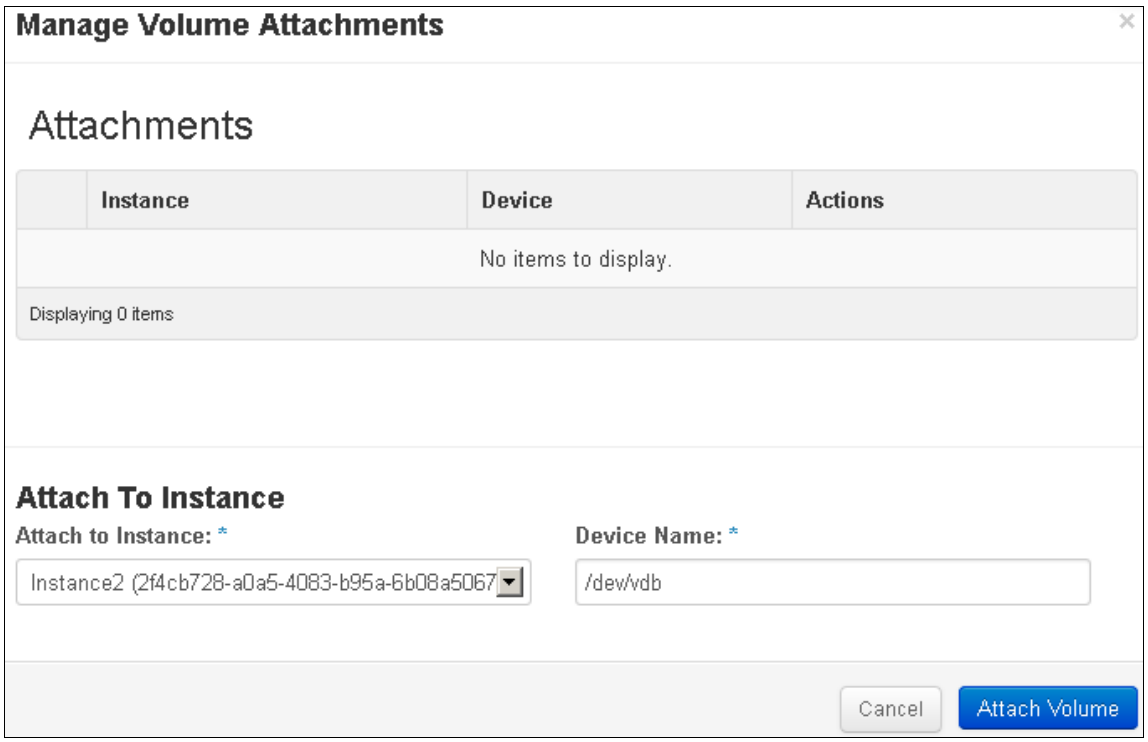


Figure 4-103 Attaching the SVC_Cinder volume to an instance

Note: Make sure that the device name specified in the Cloud Management dashboard is not being used by the instance. You can check this information by using the **`fdisk -l`** command in the Linux instances. The device naming convention might be different according to the hypervisor where the instance is running.

24. When the attach operation is finished, you should see the appropriate information in the Attached To column, as shown in Figure 4-104.

	Name	Description	Size	Status	Type	Attached To
☐	Instance2_svcvol1	Volume 1 to be provided for the Insta...	2GB	In-Use	SVC_Cinder	Attached to Instance2 on /dev/vdb

Figure 4-104 SVC_Cinder volume attached to an instance

25. Log in to the instance that has received the volume and use the **`fdisk -l`** command to identify the new volume, as shown in Example 4-128.

Example 4-128 Using the fdisk -l command in the instance to identify the new volume

```
# fdisk -l

Disk /dev/vda: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

```
   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *       16065     2088449     1036192+  83  Linux

Disk /dev/vdb: 2147 MB, 2147483648 bytes
16 heads, 63 sectors/track, 4161 cylinders, total 4194304 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/vdb doesn't contain a valid partition table
```

26. Back on the Cloud Management dashboard, click **Admin > System Panel > Instances** and note which Compute node your instance is running, as shown in Figure 4-105.

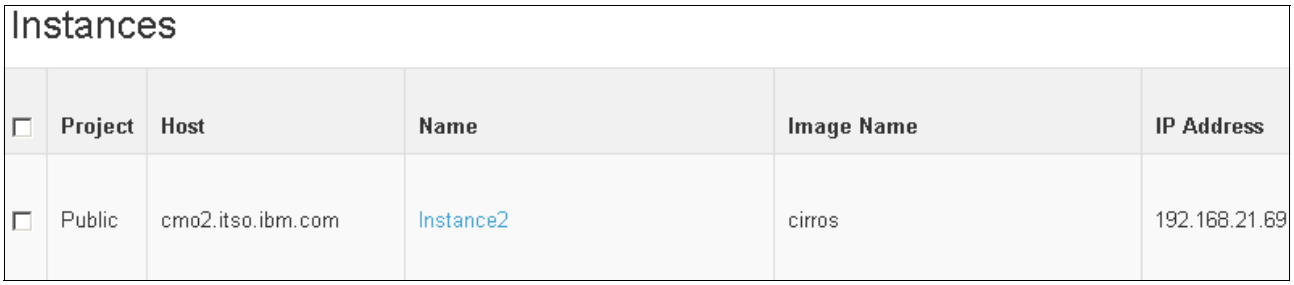


Figure 4-105 Finding the Compute node where the instance is running

27. Click the instance name and note its ID, as shown in Figure 4-106.

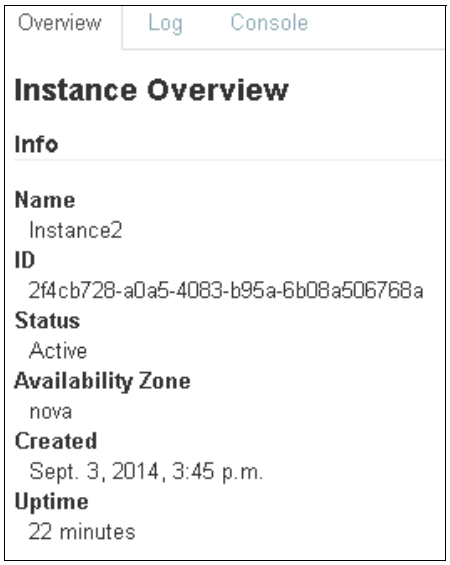


Figure 4-106 Note the instance ID

28. Now select **Admin > System Panel > Hypervisors**, and click the Compute node where the instance is running. Use the instance ID to find the instance name in the Compute node where it is running. In the example used in this publication, the instance ID is `2f4cb728-a0a5-4083-b95a-6b08a506768a`, which means that the instance name is `instance-00000289`, as shown in Figure 4-107 on page 377.

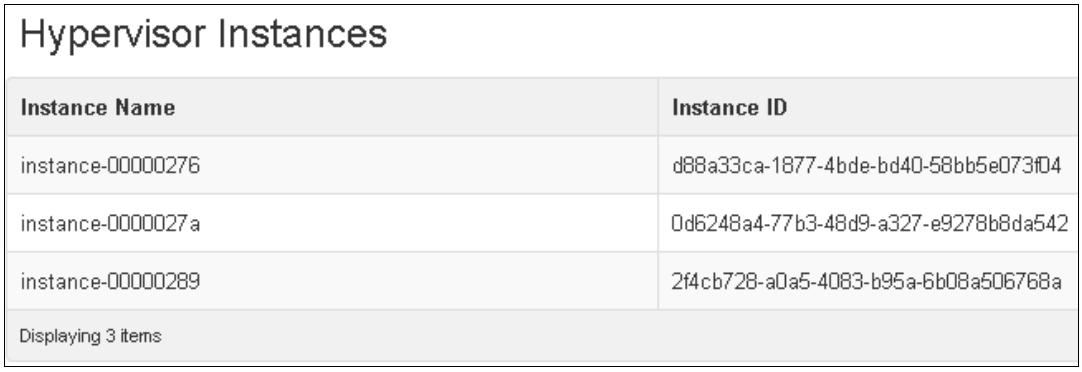


Figure 4-107 Obtaining the instance Name in the Compute node

29. Log in to the Compute node where the instance is running (in this example, `cmo2.itso.ibm.com`), and use the **`virsh dumpxml <instance_name>`** command to identify the volume that was created in this section:

    ```
    # virsh dumpxml instance-00000289
    ```

    Example 4-129 shows the section of the **`virsh dumpxml`** command where the SVC volume is shown.

    Example 4-129 Output of the virsh dumpxml with SVC volume attached

    ```
    <disk type='block' device='disk'>
          <driver name='qemu' type='raw' cache='none'/>
          <source dev='/dev/dm-7'/>
          <target dev='vdb' bus='virtio'/>
          <serial>b3dc825a-4049-48a8-a1db-068d47b87d79</serial>
          <alias name='virtio-disk1'/>
          <address type='pci' domain='0x0000' bus='0x00' slot='0x06'
    function='0x0'/>
    </disk>
    ```

30. You can use the **`fdisk -l`** and **`multipath -l`** commands to see that the SVC volume was presented to the Compute node as a block device, as shown in Example 4-130.

    Example 4-130 Checking the volume in the Compute node using fdisk and multipath commands

    ```
    [root@cmo2 ~]# fdisk -l /dev/dm-7

    Disk /dev/dm-7: 2147 MB, 2147483648 bytes
    255 heads, 63 sectors/track, 261 cylinders
    Units = cylinders of 16065 * 512 = 8225280 bytes
    Sector size (logical/physical): 512 bytes / 512 bytes
    I/O size (minimum/optimal): 512 bytes / 512 bytes
    Disk identifier: 0x00000000

    [root@cmo2 ~]# ls -l /dev/dm-7
    brw-rw---- 1 qemu qemu 253, 7 Sep  3 11:56 /dev/dm-7

    [root@cmo2 ~]# multipath -l
    mpathe (3600507680191026c4000000000000136) dm-7 IBM,2145
    size=2.0G features='1 queue_if_no_path' hwhandler='0' wp=rw
    |-+- policy='round-robin 0' prio=0 status=active
    | |- 3:0:0:3 sdz  65:144 active undef running
    ```

```
| |- 3:0:1:3 sdaa 65:160 active undef running
| |- 4:0:0:3 sdad 65:208 active undef running
| `- 4:0:1:3 sdae 65:224 active undef running
`-+- policy='round-robin 0' prio=0 status=enabled
  |- 3:0:2:3 sdab 65:176 active undef running
  |- 3:0:3:3 sdac 65:192 active undef running
  |- 4:0:2:3 sdaf 65:240 active undef running
  `- 4:0:3:3 sdag 66:0   active undef running
```

31. The `serial` parameter information that you obtained in step 29 on page 377 is the volume ID that you can find in the Cloud Management dashboard by clicking **Project > Compute Volumes**, and click the volume name, as shown in Figure 4-108.

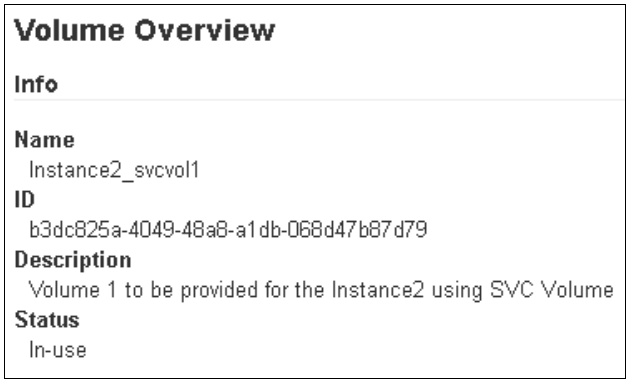


Figure 4-108 Identifying the volume ID

32. This ID will also be used in the SVC or V7000 as the volume name. Log in to the SVC or V7000, click V**olumes > Volumes by Pool**, and click the pool used for the Cinder volumes. You should see a volume named `volume-<volume_ID>`, as shown in Figure 4-109 on page 379.

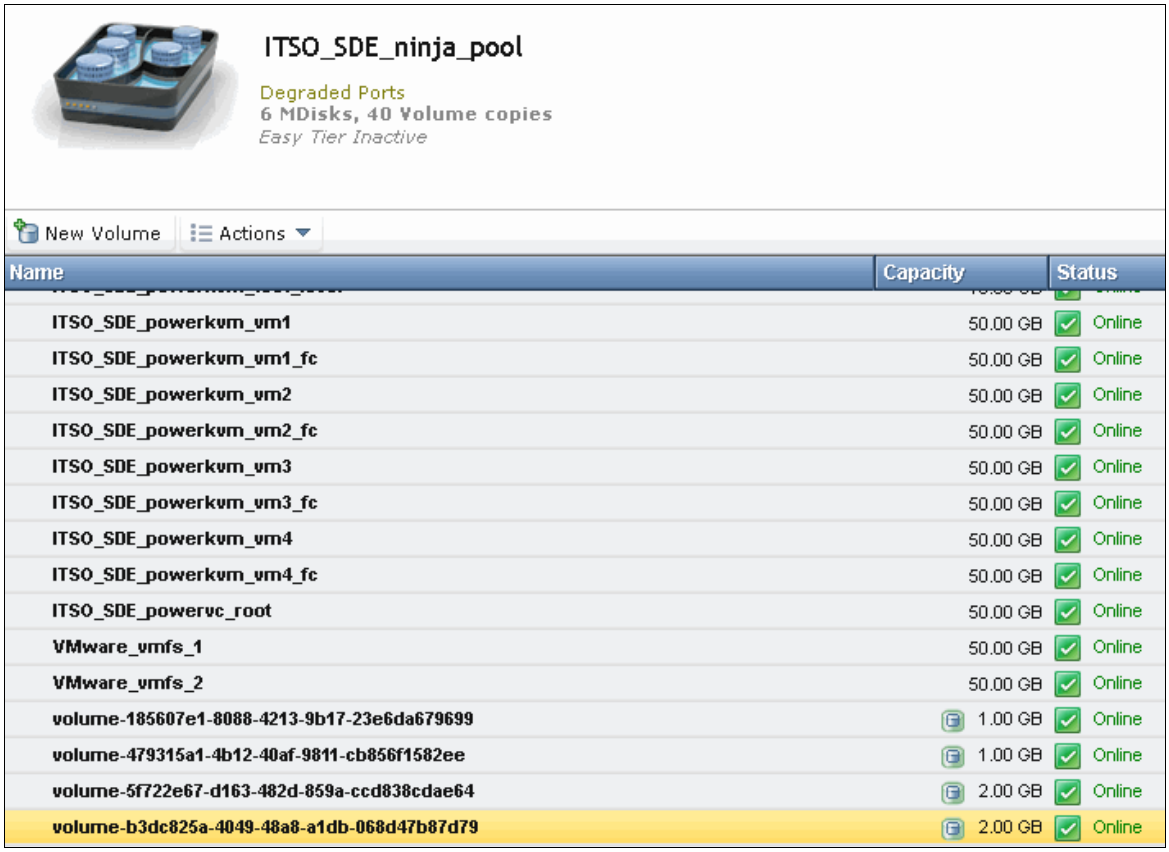


Figure 4-109 Identifying your volume in the SVC or V7000

33. Double-click this volume, and click the Host Maps tab to see that the volume is mapped to your Compute node, as shown in Figure 4-110.



Figure 4-110 Identifying the host maps of the Cinder volume

Now you can use the volume in the instance to create the file systems and populate it with application data.

4.11.3 Integrate Cinder with Power Virtualization Center (PowerVC)

By integrating Cinder with PowerVC, using the PowerVC driver, you enable the users to create volumes in Cloud Manager by using the Cloud Management dashboard and to allocate the volumes for instances running in the PowerVM Compute nodes. When a volume is created by using the Cloud Management dashboard, the Cloud Manager PowerVC driver communicates with the PowerVC server and requests the creation of the volume. The PowerVC then creates a volume in the SAN Volume Controller or V7000 that is under its management.

When you attach the volume to an instance on Cloud Management dashboard, the same process happens: Cloud Manager requests PowerVC server to attach the volume to an instance. The volume is then assigned from the SVC or V7000 directly to the instance by using NPORT_ID Virtualization (NPIV).

Important: To integrate Cinder with PowerVC, the PowerVC server setup must be complete and SVC or V7000 should already be under PowerVC management. For more information about how to complete this setup, see “Configure IBM Power Virtualization Center (PowerVC)” on page 289.

When PowerVM Compute nodes are added to the OpenStack cloud by connecting the Controller node to the PowerVC server using the PowerVC driver, the Cinder integration with SVC or V7000 through PowerVC is automatically performed. If you use Cloud Manager exclusively for PowerVC administration, no additional steps are required. The following steps are only required if you have multiple Compute nodes and you are configuring Cinder with multiple storage backends for the Compute nodes, customizations are required with PowerVC driver to allow it to continue working.

1. Log in to your Controller node using `root` user ID, and edit the `/etc/cinder/cinder.conf` file:

   ```
   # vi /etc/cinder/cinder.conf
   ```

2. Comment the lines `iscsi_ip_address`, `volumes_dir`, and `volume_driver` by using the `#` symbol, as shown in Example 4-131.

   Example 4-131 Commenting appropriate lines in the cinder.conf file

   ```
   #iscsi_ip_address=127.0.0.1
   iscsi_port=3260
   iscsi_helper=tgtadm
   #volumes_dir=/var/lib/cinder/volumes
   #volume_driver=powervc.volume.driver.powervc.PowerVCDriver
   ```

   Note: To simplify the `cinder.conf` file used in this publication, all of the pre-existing comments in the original file were deleted.

3. Add or modify the `enabled_backends` line to include the backend that you plan to add to your Cinder configuration. The `enabled_backends` line is used to allow Cinder to connect with multiple backends. In the next topic, you will include the backend section. Example 4-132 shows a valid `enabled_backends` configuration for PowerVC integration with Cinder.

   Example 4-132 Valid enabled_backends line for PowerVC integration in cinder.conf file

   ```
   enabled_backends=powervc_driver
   ```

   Note: The original `/etc/cinder/cinder.conf` file does not have a `enabled_backends` line. You must add this line below the `[DEFAULT]` section, usually at the end of the file, but before including the storage backends.

   The name used in the `enabled_backends` line, such as `powervc_driver` does not have to be any specific word. Consider using a name which informs you what kind of configuration is related to that backend. The same name must be used while creating the backend in step 4 on page 381.

If you have already configured more than one backend in your /etc/cinder/cinder.conf file, the enabled_backends line must include all the existing backends and the new backend that you are adding. The enabled_backends allows you to specify multiple backends using comma-separated values, as shown in Example 4-133.

Example 4-133 Valid enabled_backends line with multiple backends in cinder.conf file

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,nfs-cm
o
```

4. Now you need to create the PowerVC backend using the same name specified in step 3 on page 380. The backend name should be placed between square brackets [ ] and the parameters for the backend should be added in the following lines. For the PowerVC integration with Cinder, the following parameters are required:
   - volume_driver: The Cinder driver name that will be used in this backend. For PowerVC integration, use the powervc.volume.driver.powervc.PowerVCDriver.
   - volume_backend_name: The name for the volume backend that you will use later when creating a Cinder volume type and associating it with the backend. This parameter can be any name, but consider using something that tells you what kind of backend it is related to.

   Example 4-134 shows how you can add PowerVC backend to the /etc/cinder/cinder.conf file.

   Example 4-134 PowerVC backend in the cinder.conf file

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,nfs-cm
o
.
. <some lines from other backends were removed>
.
[powervc_driver]
volume_driver=powervc.volume.driver.powervc.PowerVCDriver
volume_backend_name=PowerVC
```

5. Restart the following services to allow Cinder to read the configuration file and to refresh the communication with the PowerVC server:

```
# service openstack-cinder-volume restart
# service openstack-cinder-powervc restart
```

6. In this step, you do not need to create a volume type (as you do with other storage backends), because this was automatically completed when PowerVC driver was connected to the PowerVC server. Use the following cinder command to identify the name of the volume previously created:

```
# source /root/openrc
# cinder type-list
```

Example 4-135 shows the output of the cinder type-list command in a Controller node integrated with PowerVC server. Notice that the volume type pvc:172_16_21_70:SVC default was automatically created and associated with the PowerVC driver. No additional configuration is required for PowerVC integration with Cinder at this point.

Example 4-135 cinder type-list output in a Controller node integrated with PowerVC

```
[root@cmocontroller ~]# cinder type-list
+--------------------------------------+------------------------------+
|                  ID                  |             Name             |
+--------------------------------------+------------------------------+
| 230e9aa5-c01d-49df-9d2e-55fb34983ec0 |          gpfs-fast           |
| 74193370-aa30-40e5-bcd4-a7f9b17582c1 | pvc:172_16_21_70:SVC default |
| 76c8414a-d2d9-4191-b15e-3ea8025b7b66 |             lvm              |
| bc87ed7f-c2b7-44e3-b55f-66026ca6f6d6 |           nfs-cmo            |
| c833de5b-352d-4517-bbd5-fa02e5c5280e |          SVC_Cinder          |
| e659f0bc-5572-4b32-8179-ca7e7e3207ac |            lvmnew            |
+--------------------------------------+------------------------------+
```

7. Use the **cinder extra-specs-list** command to make sure that the volume type created by the PowerVC driver for the PowerVC volumes is associated with the PowerVC backend. Example 4-136 shows the output of the command for the volume type that is associated with the PowerVC backend.

Example 4-136 Check if PowerVC volume type is associate with PowerVC/SVC information

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# cinder extra-specs-list
+--------------------------------------+------------------------------+-------------------------
|                  ID                  |             Name             | extra_specs
| 74193370-aa30-40e5-bcd4-a7f9b17582c1 | pvc:172_16_21_70:SVC default |
{u'drivers:display_name': u'SVC default', u'drivers:easytier': u'1', u'drivers:multipath': u'1',
u'drivers:compression': u'0', u'drivers:rsize': u'2', u'capabilities:volume_backend_name':
u'powervc', u'drivers:storage_pool': u'ITSO_SDE_ninja_pool', u'drivers:protocol': u'FC',
u'drivers:iogrp': u'0', u'drivers:warning': u'0', u'drivers:grainsize': u'256',
u'drivers:autoexpand': u'1'} |
```

8. With this configuration, you can create PowerVC volumes by using the Cloud Management dashboard. Log in to the Cloud Management dashboard by using the `admin` user ID.
9. Click **Project > Compute > Volumes**, and then click **Create Volume**.
   a. Provide a volume name.
   b. (Optional) Provide a description.
   c. For the Type field, select your PowerVC volume type (in this example, `pvc:172_16_21_70:SVC default`).

d. Specify the size of the volume in Gigabytes, and click **Create Volume**, as shown in Figure 4-111.

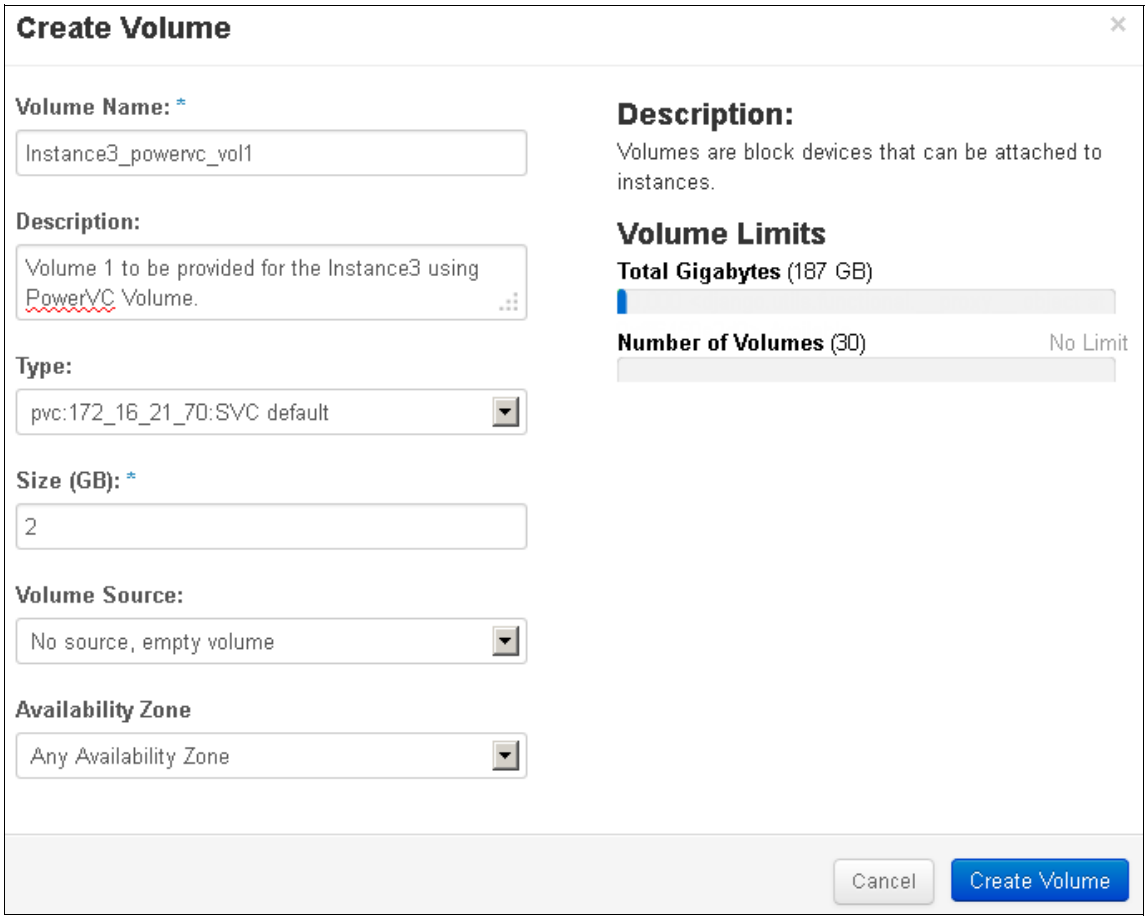


Figure 4-111 Creating PowerVC volume

10. Now, select the volume that you just created and click **More > Edit Attachments**. Select the instance that will access this volume in the Attach to Instance field, provide the device name and click **Attach Volume**, as shown in Figure 4-112.

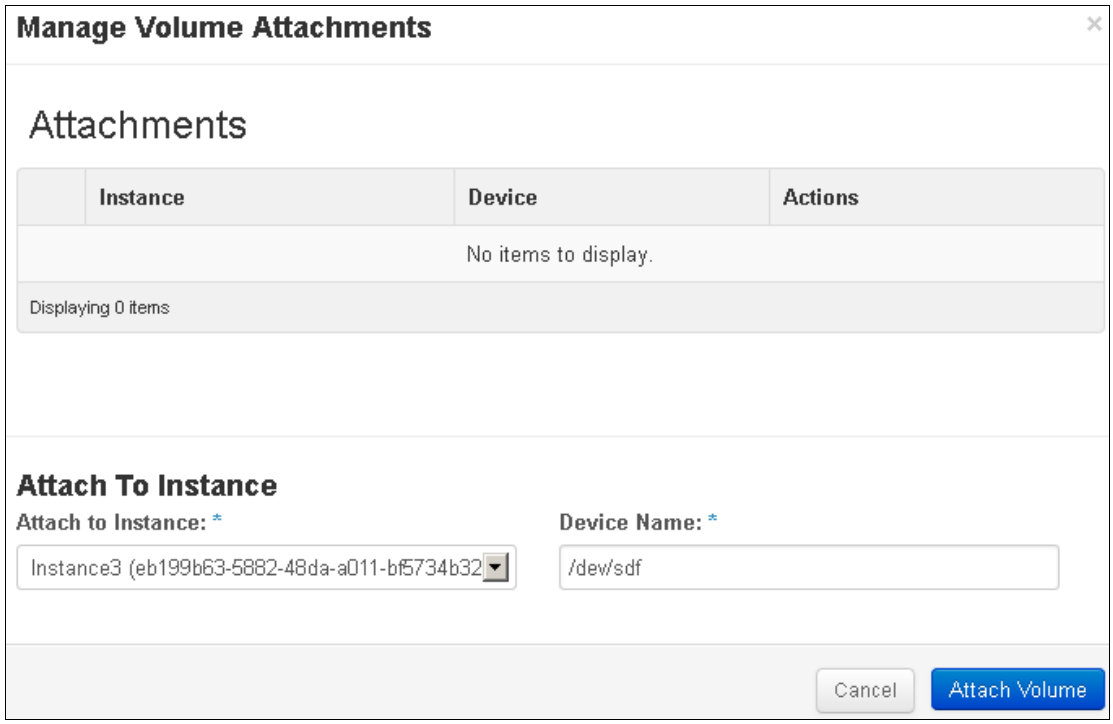


Figure 4-112 Attaching the PowerVC volume to an instance

Note: Make sure that the device name specified in the Cloud Management dashboard is not being used by the instance. You can check this information by using the `fdisk -l` command in the Linux instances. The device naming convention might be different according to the hypervisor where the instance is running.

11. When the attach operation is completed, you should see the appropriate information in the Attached To column, as shown in Figure 4-113.

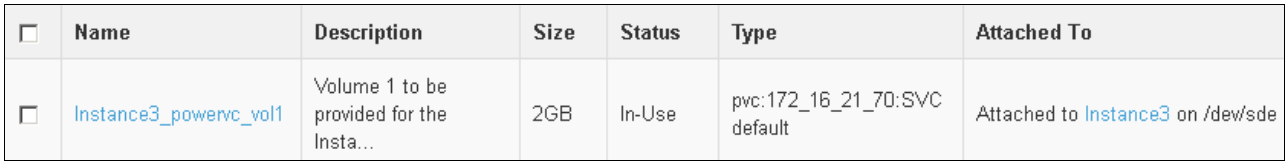


Figure 4-113 PowerVC volume attached to an instance

12. Log in to the instance that has received the volume, and use the **fdisk -l** and **multipath -l** commands to identify the new volume, as shown in Example 4-137.

Example 4-137 Checking the instance by using the fdisk and multipath commands

```
[root@instance3 ~]# fdisk -l /dev/sdf

Disk /dev/sdf: 2147 MB, 2147483648 bytes
67 heads, 62 sectors/track, 1009 cylinders
Units = cylinders of 4154 * 512 = 2126848 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

[root@instance3 ~]# multipath -l
mpathc (3600507680191026c400000000000013b) dm-2 IBM,2145
size=2.0G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=0 status=active
| |- 1:0:0:1 sdb 8:16  active undef running
| `- 2:0:0:1 sdf 8:80  active undef running
`-+- policy='round-robin 0' prio=0 status=enabled
  |- 1:0:1:1 sdd 8:48  active undef running
  `- 2:0:1:1 sdh 8:112 active undef running
```

13. Log in to the PowerVC server and click **Storage Volumes**. The volume you have just created will be displayed in the Volumes tab, and the appropriate instance should be displayed in the Attached Virtual Machine column, as shown in Figure 4-114.

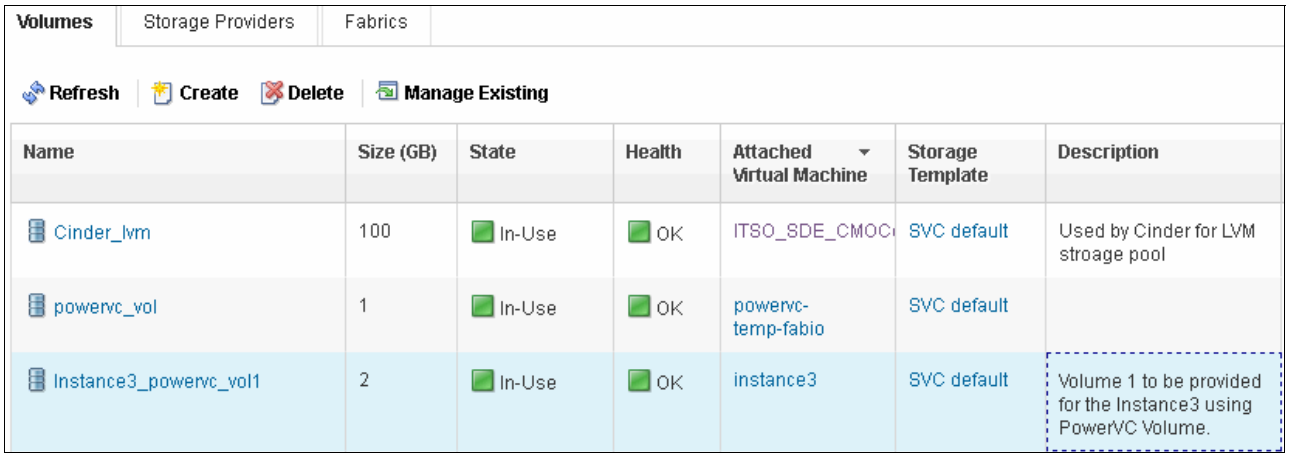


Figure 4-114 PowerVC volume displayed in the PowerVC server

14. You can find the name of the volume created in the SAN Volume Controller or V7000 by clicking the volume name on PowerVC. The SVC volume name will be displayed in the “Storage provider volume name” field, as shown in Figure 4-115.

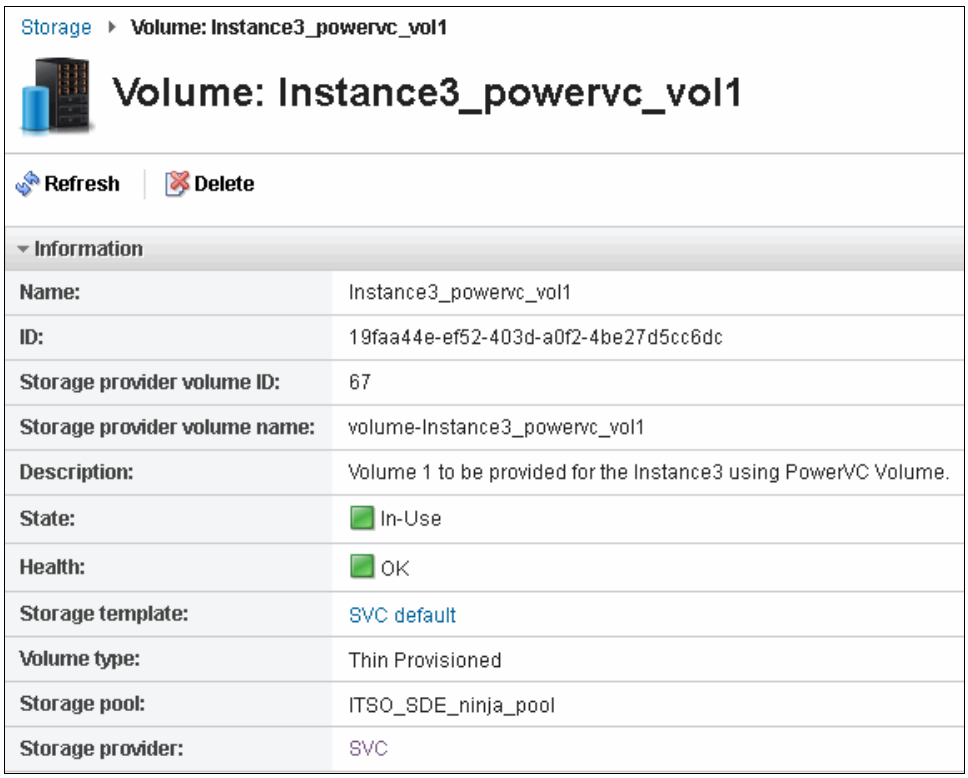


Figure 4-115 Volume name in the storage provider

15. Log in to the SVC or V7000 web interface using the `superuser` user name, click **Volumes > Volumes by Pool**, and click the pool that you use for the PowerVC volumes (specified in 9 on page 292). You should be able to see your volume, as shown in Figure 4-116.



Figure 4-116 Identifying your volume in the SVC or V7000

16. Double-click this volume and click the Host Maps tab to see that the volume is mapped directly to the instance, as shown in Figure 4-117.



Figure 4-117 Identifying the Host Maps of the PowerVC Cinder volume

Now you can use the volume in the instance to create the file systems and populate it with application data.

4.11.4 Integrate Cinder with IBM Spectrum Scale

To integrate Cinder, the Spectrum Scale cluster must be manually created and configured. The Controller node must be part of the Spectrum Scale cluster, even if it is an `ibm-os-single-controller-node`, dedicated for the Controller role, where no instance is running in the Controller physical host. All of the Compute nodes that run instances that will use Spectrum Scale volumes must also be part of the Spectrum Scale cluster and access the Spectrum Scale file system where the volumes will be created.

When you create a volume to provide to an instance, a file is created in the Spectrum Scale file system. Given that the Spectrum Scale file system is available at the Compute node, this file is presented as a file-backed storage device to the instance, which access this file as a disk. Spectrum Scalevolumes are available to all Compute nodes that can access this file system (in the cloud setup for this publication, all x86_64 KVM Compute nodes).

Note: This publication provides brief instructions on how to install and set up a Spectrum Scale (GPFS) cluster. For more information and options, see the following documentation:

Deploying a big data solution using IBM GPFS-FPO:

`http://public.dhe.ibm.com/common/ssi/ecm/en/dcw03051usen/DCW03051USEN.PDF`

Implementing the IBM General Parallel File System (GPFS) in a Cross Platform Environment, SG24-7844:

`http://www.redbooks.ibm.com/abstracts/sg247844.html`

The following steps are described in this section:

1. Creating a GPFS (Spectrum Scale) cluster on the KVM Compute nodes.
2. Adding the Controller node to the GPFS cluster.
3. Integrating Cinder with GPFS file system.

Note: This publication will guide you to complete the GPFS cluster in two steps: creating the cluster on the Compute nodes and then adding the Controller node to the cluster. You can complete this task in a single step, by creating the GPFS cluster with the Compute nodes and the Controller node. The reason for using two steps, is to show how to add the Controller node in an existing cluster, in case you already have a running GPFS cluster and just need to add the Controller, to allow Cinder to integrate with GPFS.

Create a Spectrum Scale cluster on the KVM Compute nodes

Use the following instructions to create a Spectrum Scale cluster running GPFS 4.1 in two Compute nodes running Red Hat Enterprise Linux Server release 6.5 on x86_64 servers. The same instructions can be used for an enterprise cluster with more nodes:

1. Obtain the GPFS installation binary files at the following IBM website:

   https://www.ibm.com/servers/eserver/ess/

   Note: IBM General Parallel File System is a licensed software and you must be entitled to download it from IBM website. You must register your IBM ID with your IBM Customer Number before downloading the GPFS software.

   a. Log in with your IBM ID.
   b. Select Software downloads.
   c. Select the Power (AIX) brand.
   d. Select your IBM Customer Number (ICN).
   e. Under 5692-LOP, select the GPFS software that you purchased and click continue.
   f. Download the ppc64 or x86_64 tgz file, according to the hardware where the GPFS cluster will be set up.

   Note: In the cloud setup for this publication, GPFS 4.1 for Linux running on x86_64 was used for the Compute nodes and GPFS 4.1 for Linux running on ppc64 was used for the Controller node.

2. Log in to the Compute nodes where Spectrum Scale (GPFS) cluster will be created and install the following prerequisite packages: `kernel-devel`, `cpp`, `gcc`, `gcc-c++`, `binutils` and `ksh`:

   ```
   # yum install kernel-devel
   # yum install cpp
   # yum install gcc
   # yum install gcc-c++
   # yum install binutils
   # yum install ksh
   ```

   Note: To use Yum to install the prerequisite packages, you must have a Yum repository with the Linux installation media. For instructions on setting up this Yum repository, see 4.6.6, “Set up a Yum repository” on page 242.

3. Add the GPFS commands path to the root user *PATH* variable:

   ```
   # vi /root/.bash_profile
   ```

   Modify the *PATH* variable to add `/usr/lpp/mmfs/bin` to it:

   ```
   PATH=$PATH:$HOME/bin:/usr/lpp/mmfs/bin
   export PATH
   ```

   Log out and log in to the system to read the *PATH* variable.

4. Extract the GPFS installation package and run the auto-extract utility:

   ```
   # tar -zxvf GPFS_4.1_ADV_LSX.tar.gz
   # ./gpfs_install-4.1.0-0_x86_64
   ```

Read and accept the License Agreements, as shown in Figure 4-118.

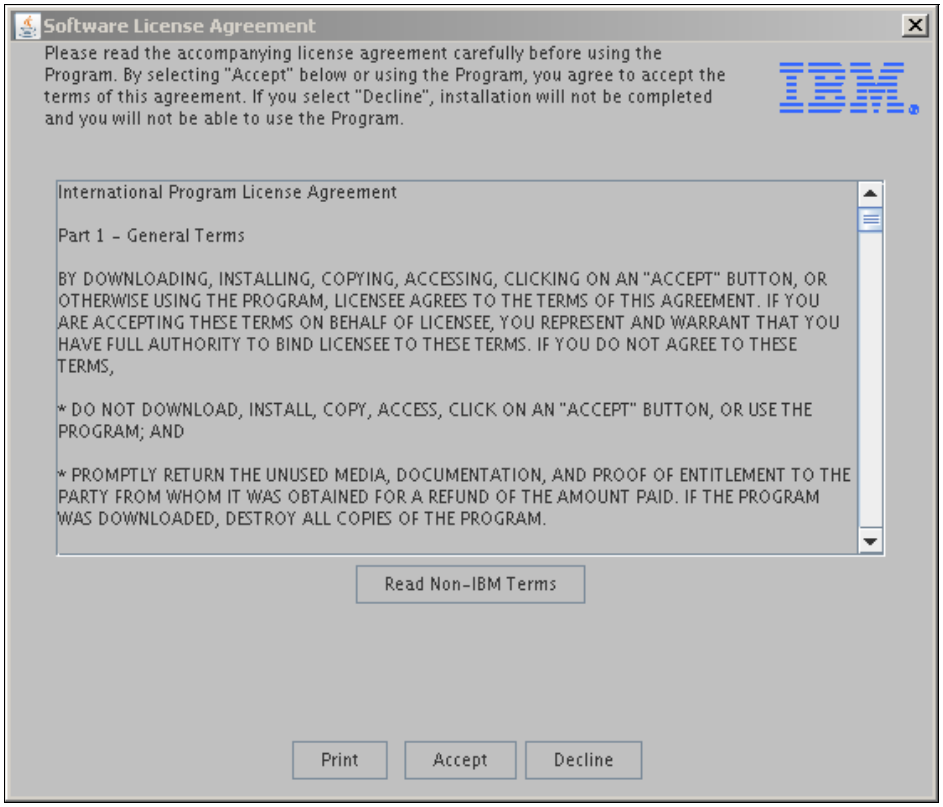


Figure 4-118 Reading and Accepting the GPFS license agreements

5. Install the GPFS software using the **rpm** command, as shown in Example 4-138.

Example 4-138 Installing the GPFS software

```
[root@cmo1 4.1]# rpm -ivh gpfs.*.rpm
Preparing...                ########################################### [100%]
   1:gpfs.base              ########################################### [ 14%]
   2:gpfs.ext               ########################################### [ 29%]
   3:gpfs.crypto            ########################################### [ 43%]
   4:gpfs.gpl               ########################################### [ 57%]
   5:gpfs.msg.en_US         ########################################### [ 71%]
   6:gpfs.gskit             ########################################### [ 86%]
   7:gpfs.docs              ########################################### [100%]
```

6. Verify if GPFS was corectly installed, as shown in Example 4-139.

Example 4-139 Verifying if GPFS was corectly installed

```
[root@cmo1 4.1]# rpm -qa | grep gpfs
gpfs.gskit-8.0.50-16.x86_64
gpfs.crypto-4.1.0-0.x86_64
gpfs.gpl-4.1.0-0.noarch
gpfs.base-4.1.0-0.x86_64
gpfs.msg.en_US-4.1.0-0.noarch
gpfs.ext-4.1.0-0.x86_64
gpfs.docs-4.1.0-0.noarch
```

7. Compile the GPFS Portability Layer using the following instructions:

```
# cd /usr/lpp/mmfs/src
# make Autoconfig
# make World
# make InstallImages
```

Note: Repeat these steps in all Compute nodes that will be part of the GPFS cluster.

8. To create the GPFS cluster, all cluster nodes must be able to authenticate using **rsh** or **ssh** and copy files by using **rcp** or **scp** without prompting for password. For security reasons, **ssh** and **scp** are preferred. The following steps will guide you to configure **ssh** to allow communication between hosts without prompting for password.

   a. Generate an SSH key by using the **ssh-keygen** command, as shown in Example 4-140.

   Example 4-140 Generating the ssh key

```
[root@cmo1 src]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
75:f9:de:28:b8:6e:79:28:83:89:f1:f7:d2:0e:b5:e0 root@cmo1.itso.ibm.com
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|              .  |
|          . o    |
|         . . .   |
|        S .   .  |
|    .  . o o . o |
|     + oE.oo. o .|
|    . + =.=...   |
|       . O=.     |
+-----------------+
```

   Note: Repeat this step in all cluster nodes.

   b. Using one of the cluster nodes, copy the `/root/.ssh/id_rsa.pub` from all nodes (including the node where you are running the commands) and append the content to a single file called `/root/.ssh/authorized_keys`. To complete this task, you can create a file containing the host name of all cluster nodes and use a script to automate the steps, as shown in Example 4-141 on page 391.

Example 4-141 Copy the id_rsa.pub from all nodes and create the authorized_keys file

```
[root@cmo1 src]# cat /etc/ibm.wcoll
cmo1
cmo2

[root@cmo1 src]# for i in `cat /etc/ibm.wcoll`; do scp
$i:/root/.ssh/id_rsa.pub /tmp/id_rsa.pub; cat /tmp/id_rsa.pub >>
/root/.ssh/authorized_keys; done
```

This step should create a single `/root/.ssh/authorized_keys` file with the `/root/.ssh/id_rsa.pub` content from all cluster nodes, as shown in Example 4-142.

Example 4-142 File authorized_keys created to allow ssh communication without password

```
[root@cmo1 src]# cat /root/.ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA3PLQUF2tSoaIuFsxK8cLv5WMNiiF4FzuqL1JrecZcax2Ds8j
7bZpQDMLV8oaPhf1QRernZUpTMHvafH4B5Ejis/QHA1XqRaQNb9qMRO9Exf2EePF48ZiHf1Yqf9c
Va1WgWwt1KzxdOvNHcBbIO7P42v/LfuMCWR9XPt5WljMdF68/lqqb5I5hkxblUhhQm4vrcQzEv18
P53+IDJsumXL3DptSjCBpfJNNgWYE8YAJTnB7Yg28SSsx2K2c+OcZACoFWi5Upt/iI51l6yX5i6L
8/Cig5n3FWsQDEBmYu69B3kTtm/j+yb476gQJBrM5vaPTWDrZDHXk3lQhCOkL7DRqQ==
root@cmo1.itso.ibm.com
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA39ZJONhh1CzPSSuhVxe8Kba7jyroytndk5dj9W+IOhY51ekH
feCW/p+iN2vabF792pf9b339dlhKknuBvfzrwiJLKkryHQ+ZCObGzBvknTizv+XZ8dIaN6g/+buv
T3PykUiTAVQt/Z2m2t1yZRVjyOwoZo7hfD5a8SSzoJbpfbRO+eU/c83VvQGdW21rHtUiJR+BAVQ9
8REcKeYAgWYqlyi/XFCOE3GHrNpyv1xGtApZKOdzXVaZd2PFhSxjcf3gSO+tgKNmnAFiz9ed+OrP
Eee1sxP8RKFJWQiK4fLOkjstQ2TCR7TRhpEUZpnbvPO+QD9eumJSPmgT/A9LGa1c/Q==
root@cmo2.itso.ibm.com
```

c. Copy the `/root/.ssh/authorized_keys` file from the node used in step b on page 390 to all cluster nodes. You can use the following command to complete this task:

```
# for i in `cat /etc/ibm.wcoll`; do scp /root/.ssh/authorized_keys
$i:/root/.ssh; scp /etc/ibm.wcoll $i:/etc; done
```

d. Try to log in using `ssh` between hosts to make sure the steps were successful before creating the GPFS cluster.

9. Make sure that SELinux is disabled by using the **`sestatus`** command, as shown in Example 4-143.

Example 4-143 Checking if SELinux is disabled

```
[root@cmo1 ~]# sestatus
SELinux status:                 disabled
```

Note: If SELinux is not disabled, use the instructions from “SELinux setting configuration” on page 287 to disable it.

10. Create the GPFS node descriptor file with the host names, using FQDN and roles for each of the nodes, as shown in Example 4-144.

Example 4-144 Creating the GPFS node descriptor file

```
[root@cmo1 GPFS]# cat gpfs-nodefile
cmo1.itso.ibm.com:quorum-manager:
cmo2.itso.ibm.com:quorum-manager:
```

11. Create the GPFS cluster using the **mmcrcluster** command, as shown in Example 4-145.

Example 4-145 Creating the GPFS cluster using the mmcrcluster command

```
[root@cmo1 GPFS]# mmcrcluster -N gpfs-nodefile -p cmo1.itso.ibm.com -s
cmo2.itso.ibm.com -C itso-sde-x86 -A -r /usr/bin/ssh -R /usr/bin/scp
mmcrcluster: Performing preliminary node verification ...
mmcrcluster: Processing quorum and other critical nodes ...
mmcrcluster: Finalizing the cluster data structures ...
mmcrcluster: Command successfully completed
mmcrcluster: Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmcrcluster: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

12. Assign the GPFS license to the cluster nodes, as shown in Example 4-146.

Example 4-146 Assigning the GPFS license to the cluster nodes

```
[root@cmo1 GPFS]# mmchlicense server --accept -N
cmo1.itso.ibm.com,cmo2.itso.ibm.com

The following nodes will be designated as possessing GPFS server licenses:
        cmo2.itso.ibm.com
        cmo1.itso.ibm.com
mmchlicense: Command successfully completed
mmchlicense: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

13. Start the GPFS cluster on all cluster nodes using the **mmstartup** command, as shown in Example 4-147.

Example 4-147 Start the GPFS cluster

```
[root@cmo1 ~]# mmstartup -a
Mon Sep  1 08:01:25 EDT 2014: mmstartup: Starting GPFS ...
cmo2.itso.ibm.com:  The GPFS subsystem is already active.
cmo1.itso.ibm.com:  The GPFS subsystem is already active.
```

14. Select the block disks which will be used for creating the GPFS Network Shared Disks (NSDs) and GPFS File Systems. In the cloud used for this publication, two SVC disks were used as shown in Example 4-148.

Example 4-148 Selecting the disks to use in the GPFS cluster

```
[root@cmo1 GPFS]# multipath -l
mpathc (3600507680191026c4000000000000086) dm-3 IBM,2145
size=100G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=0 status=active
| |- 5:0:0:1 sdc 8:32  active undef running
| |- 5:0:1:1 sde 8:64  active undef running
| |- 6:0:1:1 sdm 8:192 active undef running
| `- 6:0:2:1 sdo 8:224 active undef running
`-+- policy='round-robin 0' prio=0 status=enabled
  |- 5:0:2:1 sdg 8:96  active undef running
  |- 5:0:3:1 sdi 8:128 active undef running
  |- 6:0:0:1 sdk 8:160 active undef running
  `- 6:0:3:1 sdq 65:0  active undef running
mpathb (3600507680191026c4000000000000085) dm-2 IBM,2145
size=100G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=0 status=active
| |- 5:0:2:0 sdf 8:80  active undef running
| |- 5:0:3:0 sdh 8:112 active undef running
| |- 6:0:0:0 sdj 8:144 active undef running
| `- 6:0:3:0 sdp 8:240 active undef running
`-+- policy='round-robin 0' prio=0 status=enabled
  |- 5:0:0:0 sdb 8:16  active undef running
  |- 5:0:1:0 sdd 8:48  active undef running
  |- 6:0:1:0 sdl 8:176 active undef running
  `- 6:0:2:0 sdn 8:208 active undef running
```

15. Create the NSD descriptor file, providing the disks that will be used for the NSD creation, as shown in Example 4-149.

Example 4-149 Creating the NSD descriptor file

```
[root@cmo1 GPFS]# cat nsd1
%nsd: device=/dev/dm-2
nsd=sdensd1
servers=cmo1.itso.ibm.com,cmo2.itso.ibm.com
usage=dataAndMetadata

%nsd: device=/dev/dm-3
nsd=sdensd2
servers=cmo1.itso.ibm.com,cmo2.itso.ibm.com
usage=dataAndMetadata
```

16. Create the NSD using the **mmcrnsd** command, as shown in Example 4-150.

Example 4-150 Creating the NSD

```
[root@cmo1 GPFS]# mmcrnsd -F nsd1
mmcrnsd: Processing disk dm-2
mmcrnsd: Processing disk dm-3
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

[root@cmo1 GPFS]# mmlsnsd
File system   Disk name    NSD servers
---------------------------------------------------------------------------
 (free disk)   sdensd1      cmo1.itso.ibm.com,cmo2.itso.ibm.com
 (free disk)   sdensd2      cmo1.itso.ibm.com,cmo2.itso.ibm.com
```

17. Create the GPFS File System using the **mmcrfs** command, as shown in Example 4-151.

Example 4-151 Creating the GPFS file system

```
[root@cmo1 GPFS]# mmcrfs sdegpfs1 -F nsd1 -A yes

The following disks of sdegpfs1 will be formatted on node cmo2:
    sdensd1: size 102400 MB
    sdensd2: size 102400 MB
Formatting file system ...
Disks up to size 877 GB can be added to storage pool system.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Completed creation of file system /dev/sdegpfs1.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

18. Mount the GPFS File System on all Compute nodes using the **mmmount** command, as shown in Example 4-152.

Example 4-152 Mounting the GPFS file system

```
[root@cmo1 GPFS]# mmmount /dev/sdegpfs1 -a
Wed Jul 23 15:58:32 EDT 2014: mmmount: Mounting file systems ...
```

19. Configure GPFS to start automatically after every reboot by using the **mmchconfig** command, as shown in Example 4-153.

Example 4-153 Configuring the GPFS to start automatically

```
[root@cmo1 ~]# mmchconfig autoload=yes
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

20. Assign appropriate permission to the GPFS file system so the QEMU user can read and write to the file system.

Add the Controller node to the GPFS cluster

Use the following instructions to add the Controller node to the GPFS cluster, in case it was not already added during cluster creation in section “Create a Spectrum Scale cluster on the KVM Compute nodes” on page 388.

1. Follow the instructions from step 1 on page 388 to 9 on page 391 to install GPFS, compile GPFS portability layer, configure SSH authentication between hosts, and disable SELinux.
2. After completing these steps, run the **mmaddnode** command on one of the current cluster nodes to add the Controller node to the cluster, as shown in Example 4-154.

Example 4-154 Using the mmaddnode command to add the Controller node to the cluster

```
[root@cmo1 .ssh]# mmaddnode -N cmocontroller.itso.ibm.com:quorum-manager:
Mon Sep  1 02:21:03 EDT 2014: mmaddnode: Processing node
cmocontroller.itso.ibm.com
The authenticity of host 'cmocontroller.itso.ibm.com (172.16.21.137)' can't be
established.
RSA key fingerprint is 9c:c5:48:a2:39:d7:6d:c2:5b:8d:84:93:43:80:d9:40.
Are you sure you want to continue connecting (yes/no)? yes
mmaddnode: Command successfully completed
mmaddnode: Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmaddnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

3. Add the GPFS license to the Controller node using the **mmchlicense** command, as shown in Example 4-155.

Example 4-155 Adding the license to the Controller node

```
[root@cmo1 ~]mmchlicense client --accept -N cmocontroller.itso.ibm.com
```

4. Log in to the Controller node, start the GPFS cluster using the **mmstartup** command and mount the GPFS file system using the **mmmount** command, as shown in Example 4-156.

Example 4-156 Startup GPFS cluster on Controller node and mount the file system

```
[root@cmocontroller ~]# mmstartup
hu Sep  4 13:49:15 EDT 2014: mmstartup: Starting GPFS ...
cmocontroller.itso.ibm.com:  The GPFS subsystem is already active.

[root@cmocontroller .ssh]# mmmount /dev/sdegpfs1
Thu Sep  4 13:51:03 EDT 2014: mmmount: Mounting file systems ...
```

Note: The Controller node does not need Fibre Channel connection to the disks being used for the GPFS cluster, given that they can be accessed through the network using the Network Shared Disks.

Integrate Cinder with the Spectrum Scale (GPFS) file system

The following instructions integrate the GPFS file system created in the previous sections with the Cinder component from Cloud Manager, allowing you to use the Cloud Management dashboard to create Cinder volumes in the GPFS file system and allocate them as block disks to the instances.

1. Log in to the Controller node and run the **mmlspool** command to obtain the name of the GPFS storage pool, which will be used later, when configuring the `cinder.conf` file.

Example 4-157 shows how to obtain the GPFS Storage pool name from the /dev/sdegpfs1 file system.

Example 4-157 Obtaining the GPFS Storage pool name

```
[root@cmo1 ~]# mmlspool /dev/sdegpfs1
Storage pools in file system at '/gpfs/sdegpfs1':
Name    Id   BlkSize Data Meta Total Data in (KB) Free Data in (KB) Total Meta in
(KB)    Free Meta in (KB)
system  0    256 KB  yes  yes     209715200     205753600 ( 98%)     209715200
205755392 ( 98%)
```

2. Edit the /etc/cinder/cinder.conf file:

   ```
   # vi /etc/cinder/cinder.conf
   ```

3. Comment the iscsi_ip_address, volumes_dir and volume_driver lines in this file using the # symbol, as shown in Example 4-158.

   Example 4-158 Commenting lines in the cinder.conf file

   ```
   #iscsi_ip_address=127.0.0.1
   iscsi_port=3260
   iscsi_helper=tgtadm
   #volumes_dir=/var/lib/cinder/volumes
   #volume_driver=powervc.volume.driver.powervc.PowerVCDriver
   ```

   Note: To simplify the cinder.conf file used in this publication, all the pre-existing comments in the original file were deleted.

4. Add or modify the enabled_backends line to include the backend which you plan to add to your Cinder configuration. The enabled_backends line is used to allow Cinder to connect with multiple storage backends. In the next topic, you will include the backend section. Example 4-159 shows a valid enabled_backends configuration.

   Example 4-159 Valid enabled_backends line in cinder.conf

   ```
   enabled_backends=gpfs-fast
   ```

   Important: The original /etc/cinder/cinder.conf file does not have an enabled_backends line. You must add this line below the [DEFAULT] section, usually at the end of the file, but before including the storage backends.

   The name used in the enabled_backends line, such as gpfs-fast is not a specific string. Consider using a name which informs you what kind of configuration is related to that backend. The same name must be used while creating the backend in step on page 396.

   If you have already configured more than one backend in your /etc/cinder/cinder.conf file, the enabled_backends line should include all the backends and the new backend that you are adding. The enabled_backends allows you to specify multiple backends using comma-separated values, as shown in Example 4-160.

   Example 4-160 Specifying multiple backends in the /etc/cinder/cinder.conf file

   ```
   enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,nfs-cmo
   ```

5. Now you need to create the GPFS backend, using the same name you specified in step 4 on page 396. The backend name should be placed between square brackets [ ] and the parameter for that backend should be added in the following lines. For the GPFS integration with Cinder, the following parameters are required:
   - volume_driver: The Cinder driver name that will be used in this backend. For GPFS, use the cinder.volume.drivers.ibm.gpfs.GPFSDriver.
   - gpfs_mount_point_base: Path of directory in the GPFS file system where the volumes will be stored.
   - gpfs_storage_pool: Specify the GPFS storage pool name obtained in step 1 on page 395.
   - gpfs_sparse_volumes: If set to true, the volumes are created as sparse files, which initially use no space. If set to false, the volume is created as fully allocated file.
   - volume_group: Provide the same name used to define the backend.
   - volume_backend_name: The name for the volume backend that you will use later, when creating a Cinder volume type and associating it with the backend. Consider using the same name that was used to define the backend.

   Example 4-161 shows how you can add the GPFS backend to the /etc/cinder/cinder.conf file.

   Example 4-161 Adding the GPFS backend to the cinder.conf file

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,nfs-cm
o

[gpfs-fast]
volume_driver=cinder.volume.drivers.ibm.gpfs.GPFSDriver
gpfs_mount_point_base=/gpfs/sdegpfs1/fastpool
gpfs_storage_pool=system
gpfs_sparse_volumes=true
volume_group=gpfs-fast
volume_backend_name=gpfs-fast
```

6. Restart the openstack-cinder-volume service to allow it to read the configuration file:

```
# service openstack-cinder-volume restart
```

7. Create a volume type and associate it with the GPFS volume_backend_name parameter to allow the cloud user to create GPFS volumes in the Cloud Management dashboard. Run the following commands to create a type called gpfs-fast:

```
# source /root/openrc
# cinder type-create gpfs-fast
```

8. Associate the volume type previously created with the volume_backend_name specified in the /etc/cinder/cinder.conf file for the GPFS backend. Run the following command to associate the gpfs-fast volume type with the gpfs-fast volume_backend_name:

```
# cinder type-key gpfs-fast set volume_backend_name=gpfs-fast
```

Important: Consider reviewing the problem described in step 10 on page 359.

9. With this configuration, you are now able to create `gpfs-fast` volumes using the Cloud Management dashboard. Log in to the Cloud Management dashboard by using the `admin` user ID. Click **Project > Compute > Volumes**, and then click **Create Volume**. Provide the following information:
   - Provide a volume name.
   - (Optional) Provide a description.
   - For type, select **gpfs-fast**.
   - Specify the size in Gigabytes, as shown in Figure 4-119.

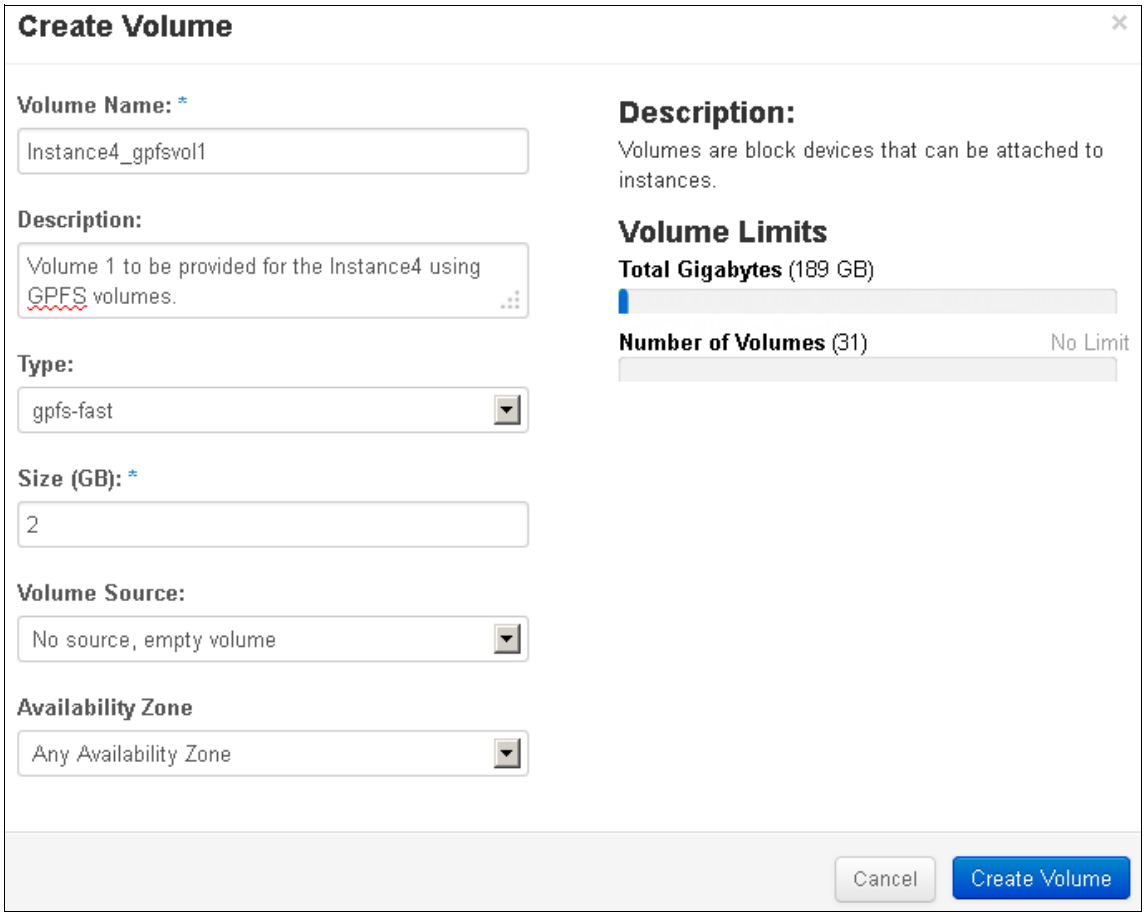


Figure 4-119 Creating a GPFS volume

Note: In the Volume Source field, you can use existing images as source for the volumes (the volumes will be populated with the content of the image), but only the creation of empty volumes is covered by this publication.

In the Availability Zone, you can specify that the volume will be available for a limited number of Compute nodes. In this case, you need to create *host aggregates*, which are not covered by this publication.

10. Now, select the volume that you created in step 9 on page 398 and click **More > Edit Attachments**. Select the instance that will access this volume in the Attach to Instance field, provide the device name, and click **Attach Volume**, as shown in Figure 4-120.

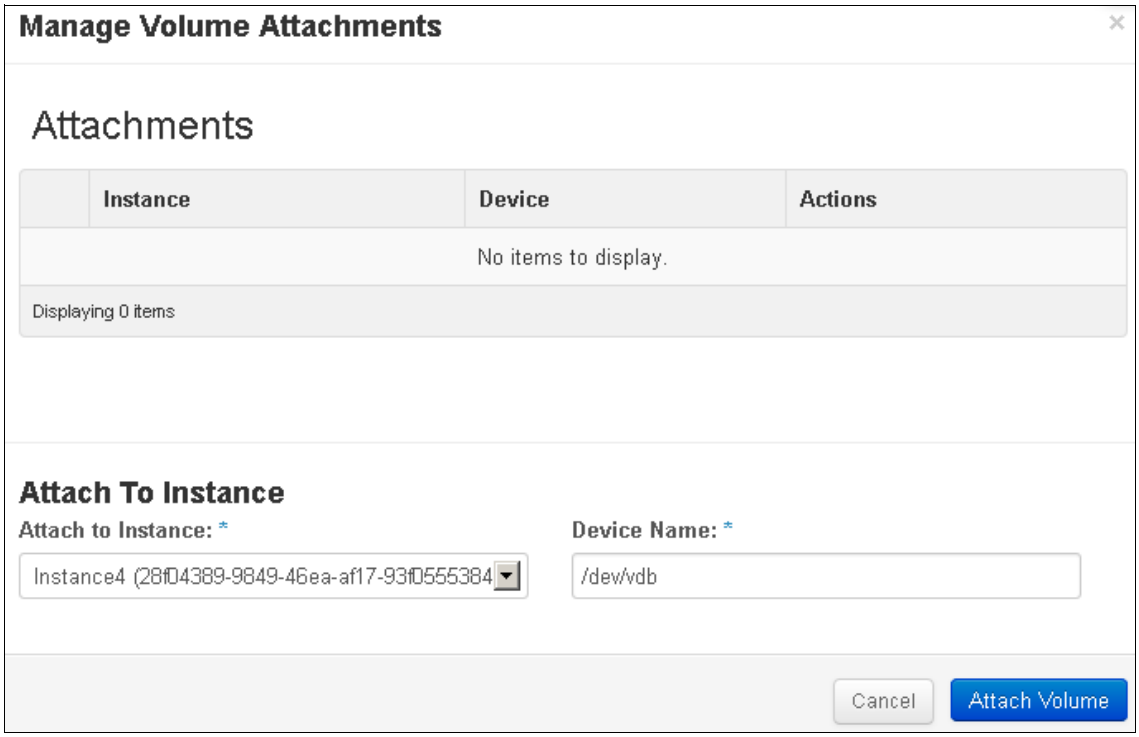


Figure 4-120 Attaching a volume to an instance

Note: Make sure that the Device Name specified in the Cloud Management dashboard is not being used by the instance. You can check this information using the **`fdisk -l`** command in the Linux instances. The device naming convention may be different according to the hypervisor where the instance is running.

11. Log in to the instance that received the volume and run the **`fdisk -l`** command to identify the new volume (use the name you previously specified), as shown in Example 4-162.

Example 4-162 Using the fdisk -l command in the instance to identify the volume

```
# fdisk -l

Disk /dev/vda: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *       16065     2088449     1036192+  83  Linux

Disk /dev/vdb: 2147 MB, 2147483648 bytes
16 heads, 63 sectors/track, 4161 cylinders, total 4194304 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
```

```
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
Disk /dev/vdb doesn't contain a valid partition table
```

12. Back on the Cloud Management dashboard, click **Admin > System Panel > Instances** and note which Compute node your instance is running, as shown in Figure 4-121.

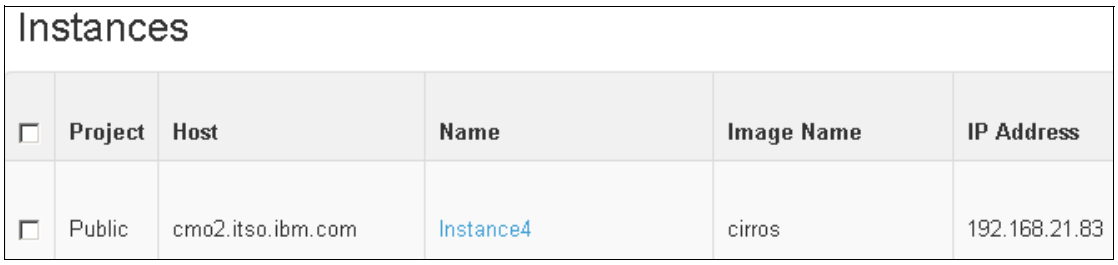


Figure 4-121 Finding the Compute node where the instance is running

13. Click the instance name and note its ID, as shown in Figure 4-122.

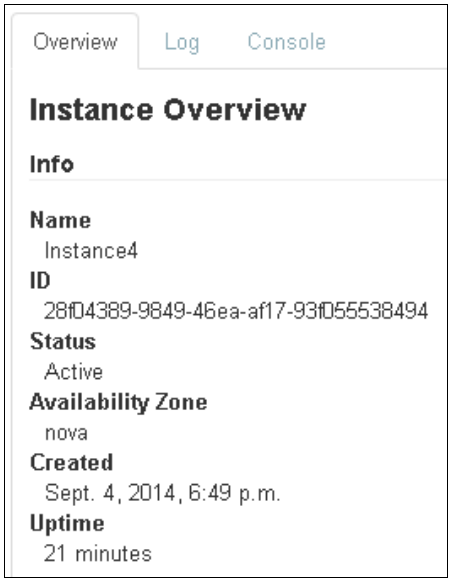


Figure 4-122 Obtaining the instance ID

14. Now click **Admin > System Panel > Hypervisors** and click the Compute node where the instance is running (which you obtained in step). Use the instance ID to find the instance name in the Compute node where it is running.

In this example, the instance ID is `28f04389-9849-46ea-af17-93f055538494`, which means the instance name `instance-00000297` is shown in Figure 4-123.

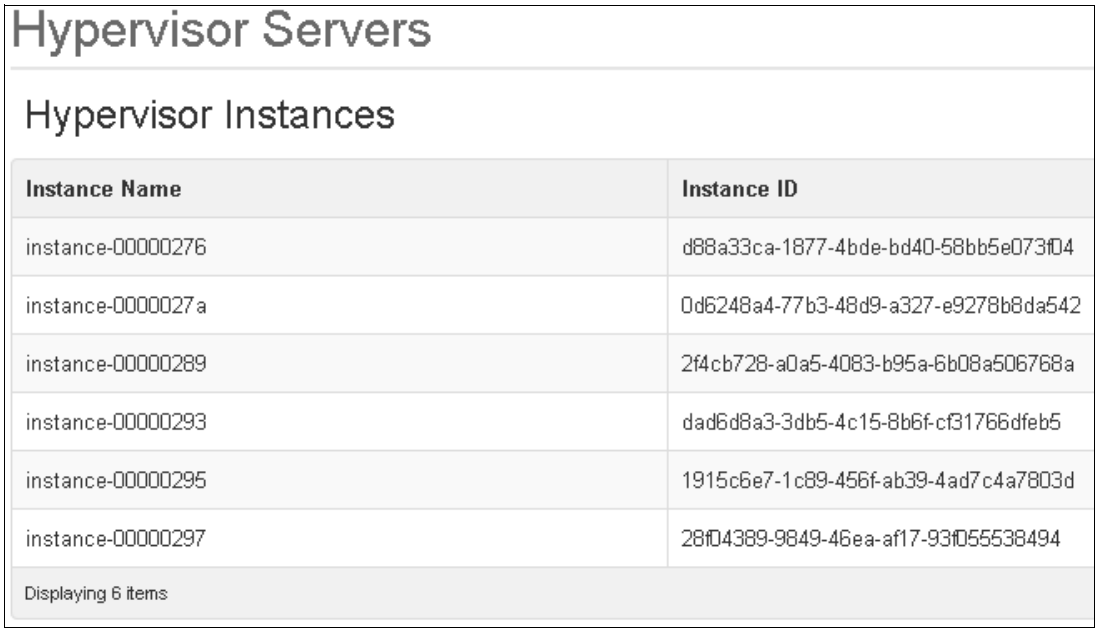


Figure 4-123 Obtaining the instance Name in the Compute node

15. Log in to the Compute node where the instance is running (in this example, cmo2.itso.ibm.com) and use the **`virsh dumpxml <instance_name>`** command to identify the volume that was created in this section:

    ```
    # virsh dumpxml instance-00000297
    ```

    Example 4-163 shows the section of the **`virsh dumpxml`** command where the GPFS volume is shown.

Example 4-163 GPFS volume in KVM Compute node instance

```
<disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none'/>
      <source dev='/gpfs/sdegpfs1/fastpool/volume-5e358f9c-4225-419a-97e1-fc899264a413'/>
      <target dev='vdb' bus='virtio'/>
      <serial>5e358f9c-4225-419a-97e1-fc899264a413</serial>
      <alias name='virtio-disk1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</disk>
```

16. In the KVM Compute node where the instance is running, you can see that the volume is a file-backed block device created in the GPFS file system, as shown in Example 4-164.

Example 4-164 GPFS volume used as file backed block device

```
[root@cmo2 ~]# ls -l /gpfs/sdegpfs1/fastpool/volume-5e358f9c-4225-419a-97e1-fc899264a413
-rw-rw---- 1 qemu qemu 2147483648 Sep  4 14:57
/gpfs/sdegpfs1/fastpool/volume-5e358f9c-4225-419a-97e1-fc899264a413
```

17. Notice that the name of the volume created in the GPFS file system contains the volume ID, which you can get from the Cloud Management dashboard by clicking **Project > Compute > Volumes**. Then, click the volume name, as shown in Figure 4-124.



Figure 4-124 Verifying the volume ID

Now you can use the volume in the instance to create the file systems and populate it with application data.

4.11.5 Integrate Cinder with Network File System

To integrate Cinder with Network File Systems (NFS), the NFS file system must be manually configured and exported in the NFS server, but it should not be manually mounted in the Compute nodes or Controller node. It must be exported and accessible for the Controller and Compute nodes that will use the NFS file system (even if the Controller is an `ibm-os-single-controller-node`, dedicated for the Controller role, where no instance is running in the Controller).

When you integrate the NFS file system with Cinder configuration, the file system is mounted in the Controller node, so it can create the files to use as file backed block device. When the volume is created and attached to an instance, the NFS file system is automatically mounted in the Compute node where the instance is running, and provided to the instance. NFS volumes will be accessible to all Compute nodes that can access the NFS file system (in the cloud setup for this publication, KVM and PowerKVM Compute nodes).

The steps that follow provide brief instructions on setting up an NFS server and exporting a file system, followed by integrating NFS with Cinder. You can also use other NFS technologies, such as clustered NFS to create a reliable NFS server. For more information about IBM Clustered Network File System, see the A Guide to the IBM Clustered Network File System, REDP-4400:

`http://www.redbooks.ibm.com/abstracts/redp4400.html`

1. Log in to the NFS server. It can be a Compute node from your cloud or other host running an NFS server. In this publication, the NFS server is the KVM Compute node cmo1.itso.ibm.com. Select the file system that will be exported using NFS. Example 4-165 shows the /sdenfs file system.

Example 4-165 Select the file system which will be exported using NFS

```
[root@cmo1 ~]# df -h
Filesystem                   Size  Used Avail Use% Mounted on
/dev/mapper/vg_cmo1-lv_root  912G   25G  841G   3% /
tmpfs                         64G     0   64G   0% /dev/shm
/dev/sda2                    485M   40M  421M   9% /boot
/dev/sda1                    200M  260K  200M   1% /boot/efi
/dev/sdegpfs1                200G  3.8G  197G   2% /gpfs/sdegpfs1
/dev/mapper/vgnfs-lvnfs      196G  188M  186G   1% /sdenfs
```

2. Modify the /etc/exports file to add the file system to the exports list:

```
# vi /etc/exports
```

Add the file system which will be exported, as shown in Example 4-166.

Example 4-166 Adding the file system to the exports list

```
[root@cmo1 ~]# cat /etc/exports
/sdenfs         *.itso.ibm.com(rw,async,no_root_squash)
```

In Example 4-166, the file system /sdenfs is being exported to any host with the .itso.ibm.com domain. The rw option is required to allow Cinder to create volumes in the NFS file system in the Controller and Compute nodes (NFS clients). The async option was used to allow asynchronous writes and improve performance. The no_root_squash option is required to allow the root user from the NFS clients (Controller and Compute nodes) to create volumes in the file system as root.

3. Make sure you set appropriate permissions to allow files to be created in the NFS clients using qemu user and group.
4. Configure the NFS server to be started after every reboot:

```
# chkconfig --level 345 nfs on
# chkconfig --level 345 nfslock on
```

5. Start the NFS server:

```
# service nfs start
# service nfslock start
```

6. Export the NFS file system:

```
# exportfs -va
```

7. Verify that the file system was exported by using the **showmount -e** command, as shown in Example 4-167.

Example 4-167 Verifying if the file system is exported

```
[root@cmo1 ~]# showmount -e
Export list for cmo1.itso.ibm.com:
/sdenfs *.itso.ibm.com
```

8. You can temporarily mount the NFS file system in the Compute nodes and Controller node to make sure you can mount it, but remember about unmounting the file system before proceeding to the next step:

```
# mount cmo1.itso.ibm.com:/sdenfs /mnt
# umount /mnt
```

9. Now that the NFS server is set up, log in to the Controller node and create a text file called /etc/cinder/nfsshares:

```
# vi /etc/cinder/nfsshares
```

The content of this file will provide the NFS server name (FQDN) and the exported file system, in the following format:

```
fqdn_nfs_server_name:/exported_file_system
```

Example 4-168 shows the `/etc/cinder/nfsshares` used in the cloud setup for this publication as a reference.

Example 4-168 File /etc/cinder/nfsshares used in the cloud setup for this publication

```
[root@cmocontroller ~]# cat /etc/cinder/nfsshares
cmo1.itso.ibm.com:/sdenfs
```

10. Set the /etc/cinder/nfsshares file to be owned by the root user and the cinder group. Set the permissions to be readable by members of the cinder group:

```
# chown root:cinder /etc/cinder/nfsshares
# chmod 0640 /etc/cinder/nfsshares
```

Example 4-169 shows the permissions of the file used in the cloud setup for this publication.

Example 4-169 File /etc/cinder/nfsshares permission

```
[root@cmocontroller ~]# ls -l /etc/cinder/nfsshares
-rw-r----- 1 root cinder 27 Sep  1 07:02 /etc/cinder/nfsshares
```

11. Modify the `/etc/cinder/cinder.conf` file using the **`vi`** command:

```
# vi /etc/cinder/cinder.conf
```

12. Comment the `iscsi_ip_address`, `volumes_dir` and `volume_driver` lines in this file using the `#` symbol, as shown in Example 4-170.

Example 4-170 Commenting the lines in cinder.conf file

```
#iscsi_ip_address=127.0.0.1
iscsi_port=3260
iscsi_helper=tgtadm
#volumes_dir=/var/lib/cinder/volumes
#volume_driver=powervc.volume.driver.powervc.PowerVCDriver
```

Note: To simplify the cinder.conf file used in this publication, all the pre-existing comments in the original file were deleted.

13. Add or modify the `enabled_backends` line to include the backend which you plan to add to your Cinder configuration. The `enabled_backends` line is used to allow Cinder to connect with multiple storage backends. In the next topic, you will include the backend section. Example 4-171 shows a valid `enabled_backends` configuration.

Example 4-171 Adding NFS to the enabled_backends section

```
enabled_backends=nfs-cmo
```

Important: The original `/etc/cinder/cinder.conf` file does not have a `enabled_backends` line. You must add this line below the `[DEFAULT]` section, usually at the end of the file, but before including the storage backends.

The name used in the `enabled_backends` line, such as `nfs-cmo` does not have to be any specific word. Consider using a name which informs you what kind of configuration is related to that backend. The same name must be used while creating the backend in step 14.

If you have already configured more than one backend in your /etc/cinder/cinder.conf file, the enabled_backends line should include all existing backends and the new backend that you are creating. The enabled_backends allows you to specify multiple backends using comma-separated values, as shown in Example 4-172.

Example 4-172 Specifying multiple backends in the /etc/cinder/cinder.conf file

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,nfs-cm
o
```

14. Now you need to create the NFS backend, using the same name you specified in step 13. The backend name should be placed between square brackets `[ ]` and the parameters for that backend should be added in the following lines. For the NFS integration with Cinder, the following parameters are required:
   - `volume_driver`: The Cinder driver name that will be used in this backend. For NFS, use `cinder.volume.drivers.nfs.NfsDriver`.
   - `nfs_shares_config`: The configuration file that contains the NFS server and exported directory. This should point to the `/etc/cinder/nfsshares` file created in 9 on page 404.
   - `nfs_mount_point_base`: Directory where the NFS file system will be mounted in the Controller node. The Cinder driver will create a subdirectory inside this directory to mount the NFS file system, so the `cinder` user and group should be able to read, write and execute inside this directory.
   - `volume_backend_name`: The name for the volume backend that you will use later, when creating a Cinder volume type and associating it with the backend. Consider using the same name used to define the backend.

Example 4-173 shows how to add the GPFS backend to the /etc/cinder/cinder.conf file.

Example 4-173 Adding the NFS backend to the /etc/cinder/cinder.conf file

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,nfs-cm
o

[nfs-cmo]
volume_driver=cinder.volume.drivers.nfs.NfsDriver
nfs_shares_config=/etc/cinder/nfsshares
nfs_mount_point_base=/var/lib/cinder/nfs
volume_backend_name=nfs-cmo
```

Important: Make sure that the directory specified in nfs_mount_point_base exists and cinder user and group have rwx permission on it, as shown in the following example:

```
[root@cmocontroller ~]# ls -ld /var/lib/cinder/nfs
drwxrwxr-x 4 cinder cinder 4096 Sep  1 15:54 /var/lib/cinder/nfs
```

15. Restart the openstack-cinder-volume service to allow it to read the configuration file:

    ```
    # service openstack-cinder-volume restart
    ```

16. Create a volume type and associate it with the NFS *volume_backend_name* parameter, to allow the cloud user to create NFS volumes using the Cloud Management dashboard.

    Run the following commands to create a type called nfs-cmo:

    ```
    # source /root/openrc
    # cinder type-create nfs-cmo
    ```

17. Associate the volume type previously created with the volume_backend_name specified in the /etc/cinder/cinder.conf file for the NFS backend. Run the following command to associate the nfs-cmo volume type with the nfs-cmo volume_backend_name:

    ```
    # cinder type-key nfs-cmo set volume_backend_name=nfs-cmo
    ```

    Important: Consider reviewing the problem described in 10 on page 359.

18. With this configuration, you can now create nfs-cmo volumes using the Cloud Management dashboard. Log in to the Cloud Management dashboard by using the admin user ID.

19. Click **Project > Compute > Volumes**, and then click **Create Volume**. Provide a volume name, provide a description (optional), select **nfs-cmo** as the type, and specify the size in Gigabytes, as shown in Figure 4-125.

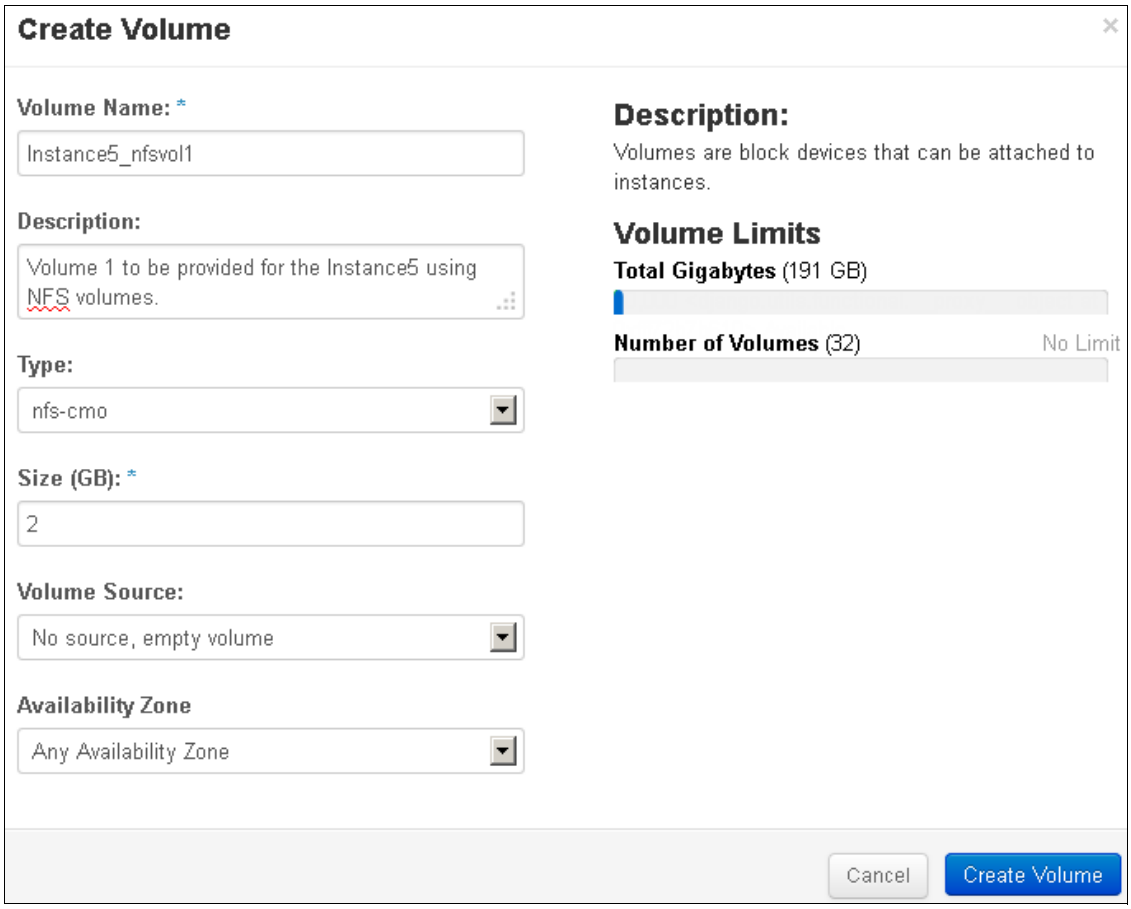


Figure 4-125 Creating an NFS volume

Note: In the Volume Source field, you can use existing images as source for the volumes (the volumes will be populated with the content of the image), but only the creation of empty volumes is covered in this book.

In the Availability Zone, you can specify that the volume will be available for a limited number of Compute nodes. In this case, you need to create *host aggregates*, which are not covered in this book.

20. Now, select the volume that you created and click **More > Edit Attachments**. Select the instance that will receive the volume in the Attach to Instance field, provide the device name and click **Attach Volume**, as shown in Figure 4-126.

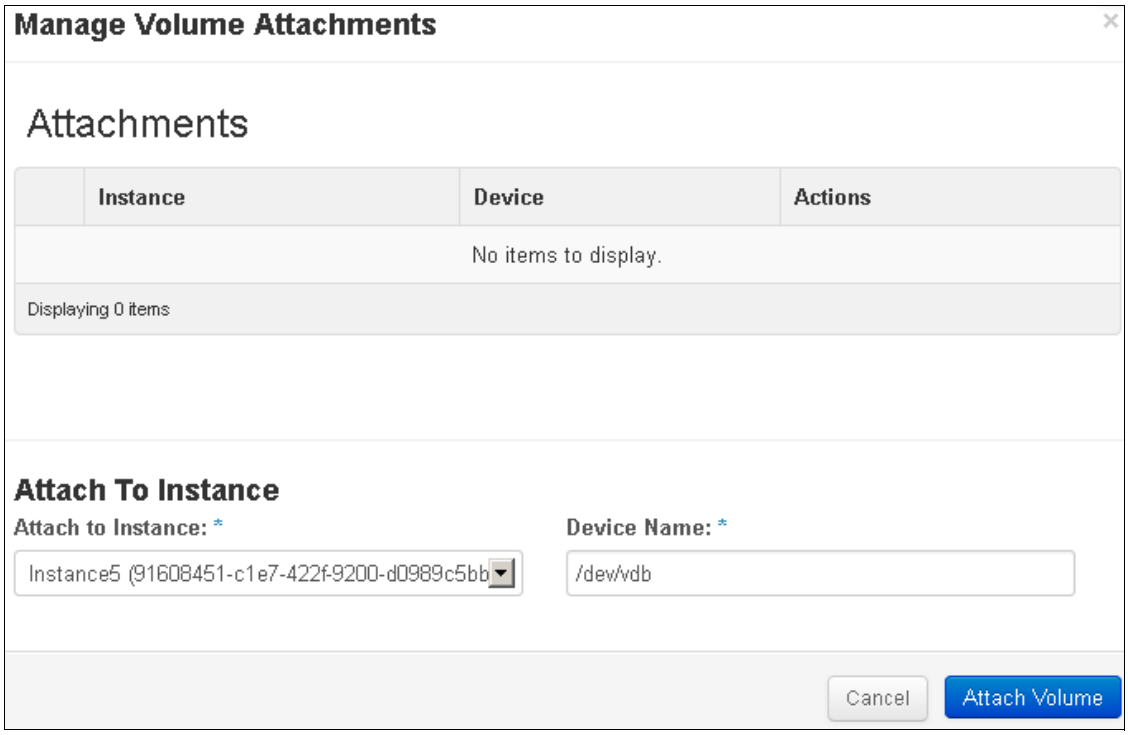


Figure 4-126 Attaching a volume to an instance

Note: Make sure that the Device Name specified in the Cloud Management dashboard is not being used by the instance. You can check this information using the **`fdisk -l`** command in the Linux instances. The device naming convention may be different according to the hypervisor where the instance is running.

21. Log in to the instance which received the volume and run the **`fdisk -l`** command to identify the new volume (using the name you previously specified), as shown in Example 4-174.

Example 4-174 Using the fdisk -l command in the instance to identify the new volume

```
[root@host-192-168-21-84 ~]# fdisk -l

Disk /dev/vda: 21.5 GB, 21474836480 bytes
16 heads, 63 sectors/track, 41610 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xd99d0900

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *           3          11        4096   41  PPC PReP Boot
Partition 1 does not end on cylinder boundary.
/dev/vda2              11        1027      512000   83  Linux
Partition 2 does not end on cylinder boundary.
```

```
/dev/vda3            1027       20806     9968640   8e  Linux LVM
Partition 3 does not end on cylinder boundary.

Disk /dev/vdb: 2147 MB, 2147483648 bytes
16 heads, 63 sectors/track, 4161 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/mapper/VolGroup-lv_root: 6043 MB, 6043992064 bytes
255 heads, 63 sectors/track, 734 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/mapper/VolGroup-lv_swap: 4160 MB, 4160749568 bytes
255 heads, 63 sectors/track, 505 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

22. In the Controller node, you will be able to see that a directory was created inside the directory specified in `nfs_mount_point_base` and the NFS file system was mounted, as shown in Example 4-175. The NFS volume above is a file created inside this file system.

Example 4-175 Checking the NFS mounted in the Controller node

```
[root@cmocontroller ~]# grep nfs_mount_point_base /etc/cinder/cinder.conf
nfs_mount_point_base=/var/lib/cinder/nfs

[root@cmocontroller ~]# ls -l /var/lib/cinder/nfs
total 4
drwxrwxrwx 2 root root 4096 Sep  4 18:08 e26a938bf321e24cd460d27db77b779a

[root@cmocontroller ~]# ls -l /var/lib/cinder/nfs/e26a938bf321e24cd460d27db77b779a
total 0
-rw-rw-rw- 1 qemu qemu 2147483648 Sep  4 18:08 volume-678eee68-0024-4b6b-b151-6c23b27e097e
-rw-rw-rw- 1 qemu qemu 1073741824 Sep  1 22:46 volume-ef131710-fe1a-4672-8308-9caee8a4e051

[root@cmocontroller ~]# df -h
Filesystem                                      Size  Used Avail Use% Mounted on
/dev/mapper/vg_cmocontroller-lv_root             94G   35G   54G  40% /
tmpfs                                            16G  576K   16G   1% /dev/shm
/dev/mapper/mpathap2                            485M   53M  407M  12% /boot
/root/RHEL6.5-20131111.0-Server-ppc64-DVD1.iso  3.2G  3.2G     0 100% /media/cdrom
/root/IBM_Installation_Toolkit_56.iso           787M  787M     0 100% /media/toolkit
/dev/mapper/glancevg-glancelv                    98G   20G   74G  21% /var/lib/glance
/dev/sdegpfs1                                   200G  3.9G  197G   2% /gpfs/sdegpfs1
cmo1.itso.ibm.com:/sdenfs                       196G  187M  186G   1%
/var/lib/cinder/nfs/e26a938bf321e24cd460d27db77b779a
/dev/mapper/vg_cmocontroller-lv_repo             99G  5.4G   89G   6% /repo
```

23. You can obtain the Volume ID of the volume that you have just created on the Cloud Management dashboard by clicking **Project > Compute > Volumes** and clicking the volume name. You will be able to see the Volume ID, and it should match the file name created in the NFS file system, as shown in Example 4-175 on page 409. Figure 4-127 shows the volume ID for the NFS volume.

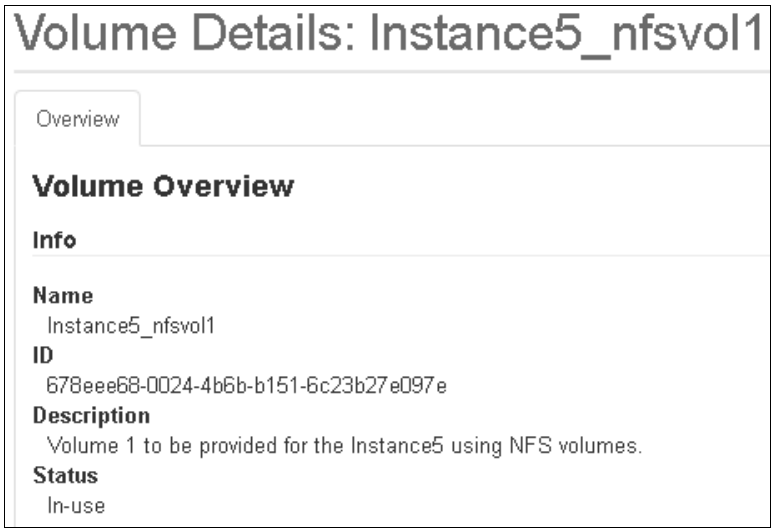


Figure 4-127 Verifying the volume ID of the NFS volume

24. Back on the Cloud Management dashboard, click **Admin > System Panel > Instances** and note which Compute node where your instance is running, as shown in Figure 4-128.

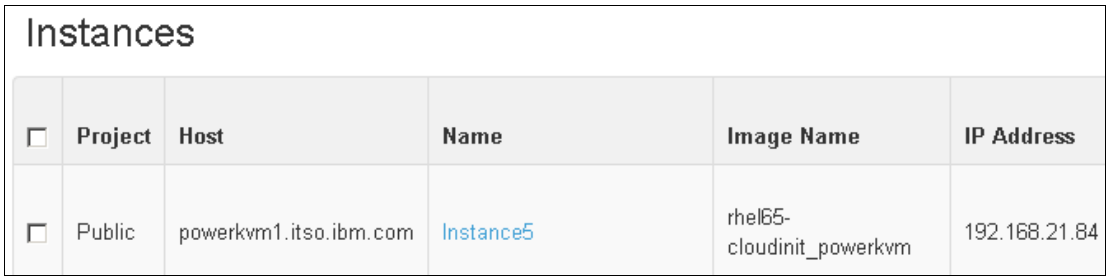


Figure 4-128 Finding the Compute node where the instance is running

25. Click the instance name and note its ID, as shown in Figure 4-129.

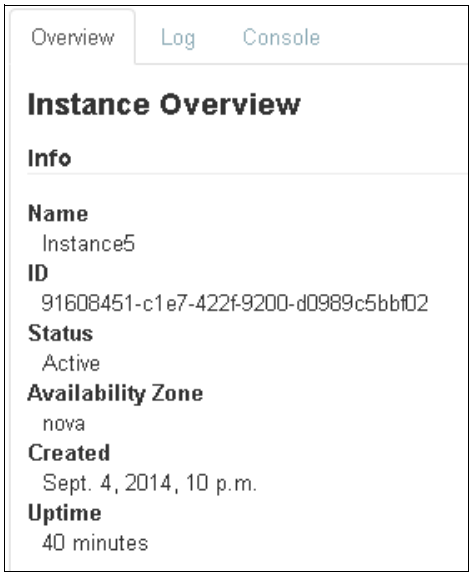


Figure 4-129 Obtaining the instance ID

26. Now click **Admin > System Panel > Hypervisors** and click the Compute node where the instance is running. Use the instance ID to find the instance Name in the Compute node where it is running. In the example used in this publication, the instance ID is `91608451-c1e7-422f-9200-d0989c5bbf02`, which means that the instance name is `instance-00000298`, as shown in Figure 4-130.

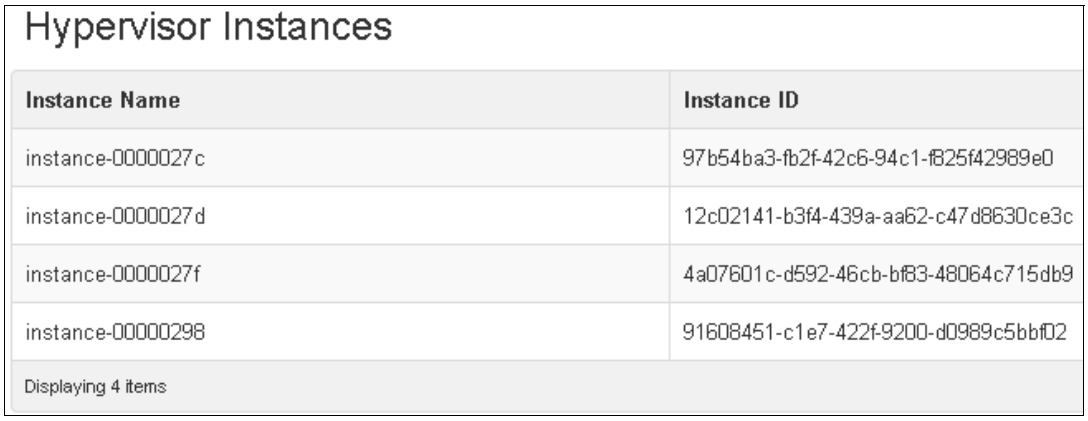


Figure 4-130 Obtaining the instance name in the Compute node

27. Log in to the Compute node where the instance is running (in this example, `powerkvm1.itso.ibm.com`) and use the virsh **`dumpxml <instance_name> command`** to identify the volume that was created in this section:

```
# virsh dumpxml instance-00000298
```

Example 4-176 shows the section of the **virsh dumpxml** command where the NFS volume is shown.

Example 4-176 NFS volume in PowerKVM Compute node instance

```
<disk type='file' device='disk'>
      <driver name='qemu' type='raw' cache='none'/>
      <source
file='/var/lib/nova/mnt/e26a938bf321e24cd460d27db77b779a/volume-678eee68-0024-4
b6b-b151-6c23b27e097e'>
        <seclabel model='selinux' labelskip='yes'/>
      </source>
      <target dev='vdb' bus='virtio'/>
      <serial>678eee68-0024-4b6b-b151-6c23b27e097e</serial>
      <alias name='virtio-disk1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04'
function='0x0'/>
</disk>
```

28. In the PowerKVM Compute node, the NFS file system was automatically mounted and the file (used as file backed block device) is available, as shown in Example 4-177.

Example 4-177 NFS volume mounted in the PowerKVM Compute node

```
[root@powerkvm1 ~]# df -h
Filesystem                                      Size  Used Avail Use% Mounted on
/dev/mapper/ibmpkvm_vg_root-ibmpkvm_lv_root      20G  8.0G   12G  41% /
devtmpfs                                         62G     0   62G   0% /dev
tmpfs                                            64G     0   64G   0% /dev/shm
tmpfs                                            64G  9.9M   64G   1% /run
tmpfs                                            64G     0   64G   0% /sys/fs/cgroup
tmpfs                                            64G  256K   64G   1% /tmp
/dev/sda2                                       488M  121M  358M  26% /boot
/dev/mapper/ibmpkvm_vg_log-ibmpkvm_lv_log       9.8G  314M  9.4G   4% /var/log
/dev/mapper/ibmpkvm_vg_data-ibmpkvm_lv_data     513G   70M  513G   1% /var/lib/libvirt/images
cmo1.itso.ibm.com:/sdenfs                       196G  187M  186G   1%
/var/lib/nova/mnt/e26a938bf321e24cd460d27db77b779a

[root@powerkvm1 ~]# ls -l
/var/lib/nova/mnt/e26a938bf321e24cd460d27db77b779a/volume-678eee68-0024-4b6b-b151-6c23b27e097e
-rw-rw-rw-. 1 qemu qemu 2147483648 Sep  4 18:08
/var/lib/nova/mnt/e26a938bf321e24cd460d27db77b779a/volume-678eee68-0024-4b6b-b151-6c23b27e097e
```

Now you can use the volume in the instance to create the file system and populate it with application data.

4.12 Manage images in IBM Cloud Manager

In Cloud Manager, you use images to deploy your instances (virtual machines). The images are virtual machines previously installed and setup with appropriate operating system, additional software and specific configurations for your environment. The OpenStack component that manages images is called *Glance*. This section will describe how to create images for deployment on KVM, PowerKVM and PowerVM Compute nodes in your OpenStack cloud.

Most of the Linux distributions allow you to download images ready for deployment in your OpenStack cloud, so you do not need to manually install, prepare and setup your image. You can also set up your own image, installing it from scratch. This allows you a better customization of your image, given that you can install specific software required by your Company and used in your environment (such as monitoring components, backup agents, and so on). If you prepare your own image, you can also set up images with Enterprise Software pre-loaded, for example, setting up a Red Hat Enterprise Linux + DB2 image or a SUSE Linux Enterprise Server + IBM WebSphere Application Server image.

After corectly preparing and creating your images, with a few steps you can deploy as many instances you need, and all the manual steps you would have to do to install and configure additional software on the instances will be already done. Using images, you can have your IBM DB2 and WebSphere servers (and many other software) deployed and running in a few minutes, by deploying an instance based on a previously prepared image with such software.

The following topics are covered in this section:

- "Manage images in KVM and PowerKVM Compute nodes" on page 413
  - "Prepare your image" on page 413
  - "Obtain cloud-ready images from the Linux distributions" on page 424
  - "Create the image in IBM Cloud Manager" on page 426
- "Manage images in PowerVM Compute nodes" on page 431
  - "Create the logical partition and installing the operating system" on page 432
  - "Install Activation Engine" on page 434
  - "Create the image in IBM Power Virtualization Center (PowerVC)" on page 437
  - "Import the PowerVC images in IBM Cloud Manager" on page 442

4.12.1 Manage images in KVM and PowerKVM Compute nodes

The procedure to create KVM and PowerKVM images in Cloud Manager is similar. This section will describe how you can prepare your own image and add this image to the Cloud Manager and also show how you can download cloud-ready images from your Linux Distribution and add it to the Cloud Manager. Although most of the steps are the same for KVM and PowerKVM Compute nodes, there are some specific modifications that need to be made to inform the architecture of the image (x86_64 or ppc64) to Cloud Manager. These steps are also described in this section.

Prepare your image

To prepare your image to use in the OpenStack cloud, besides creating the virtual machine and installing the operating system, these requirements must be met to allow full functions:

- Disk partitions and resizing the root partition on boot
- No hardcoded MAC address for the network adapters
- SSH server installed and running
- Firewall disabled
- SSH key-pair injection allowed to log in to the deployed instance without a password

Most of the requirements for an image can be accomplished by using the cloud-init software. The cloud-init is an open source package that allows early initialization of a cloud instance. If cloud-init is installed in an image, when the image is deployed, it can handle resizing the disk size (according to the flavor selected during the image deployment), injecting a fixed IP address (in case dhcp is not being used), injecting public ssh keys to allow login without password, modifying the instance hostname, and so on. The cloud-init project is maintained by Canonical (the Company that runs the Ubuntu project), but it is not exclusively for Ubuntu and has successfully been ported to several other Linux distributions, including Red Hat Enterprise Linux.

The following steps will allow you to install a virtual machine and setup all required steps to prepare it to be fully functional for OpenStack cloud deployment:

Important: The next steps will use a Red Hat Enterprise Linux 6.5 running on x86_64 KVM Compute node as an example to show how to install and prepare the virtual machine for deployment in the OpenStack cloud. Some of these steps might be different (including the cloud-init installation) in other Linux distributions or architectures. Please check your Linux distribution documentation for more information about how to install cloud-init and setup a virtual machine to be fully functional in an OpenStack cloud. Several examples from different distributions can be found at OpenStack website:

http://docs.openstack.org/image-guide/content/ch_creating_images_manually.html

1. Log in to a KVM host (preferably log in to a KVM host which is not part of your OpenStack cloud, to avoid network restrictions due to the use of Open vSwitch bridges rather than Linux bridges).
2. Run the following command to create a qcow2 disk for your virtual machine:

```
# qemu-img create -f qcow2 /tmp/rhel65.qcow2 10G
```

Note: Disk images using qcow2 format are qemu copy on write, a thin provisioning disk format for KVM virtual machines.

3. Run the virt-install command to create the KVM virtual machine using the qcow2 disk you created in step 2, as shown in Example 4-178.

Example 4-178 Creating the KVM virtual machine using virt-install command

```
[root@cmokvm ~]# virt-install --virt-type kvm --name rhel65 --ram 1024 \
--disk /tmp/rhel65.qcow2,format=qcow2 --graphics vnc,listen=0.0.0.0 \
--noautoconsole --os-type=linux --os-variant=rhel6 \
--extra-args="console=tty0 console=ttyS0,115200n8 serial" \
--location=/root/RHEL6.5-20131111.0-Server-x86_64-DVD1.iso \
--network network=default

Starting install...
Retrieving file .treeinfo...
| 3.2 kB     00:00 ...
Retrieving file vmlinuz...
| 7.9 MB     00:00 ...
Retrieving file initrd.img...
|  64 MB     00:00 ...
Creating domain...
|    0 B     00:00
Domain installation still in progress. You can reconnect to
the console to complete the installation process.
```

Note: Replace the appropriate **virt-install** parameters according to your KVM host configuration.

(Optional) You can create the virtual machine by using the **virt-manager** command to open the virtual machine manager GUI.

4. Run the **virt-manager** command to open the virtual machine Manager graphical user interface, so you can install the Red Hat Linux operating system (you need to export the DISPLAY to a valid X11 server, in case you are not running from the KVM host graphical console):

   ```
   # virt-manager
   ```

   In the virtual machine Manager, right click your virtual machine and click **Open**, as shown in Figure 4-131.

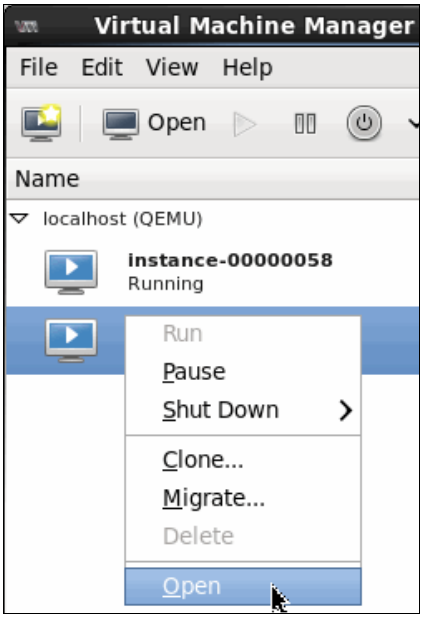


Figure 4-131 Opening the virtual machine

5. The virtual machine console will be opened and the Red Hat Enterprise Linux installation will be in progress. Follow the regular steps for installing the Red Hat Enterprise Linux on your virtual machine. For guidance in installation, see the Red Hat Enterprise Linux Installation Guide:

   http://red.ht/1xEzlge

6. After completing the installation steps, log in to your virtual machine and run the **setup** command to disable the Linux firewall:

   ```
   # setup
   ```

   Go to **Firewall configuration**, and select **Run Tool**, as shown in Figure 4-132 on page 416.

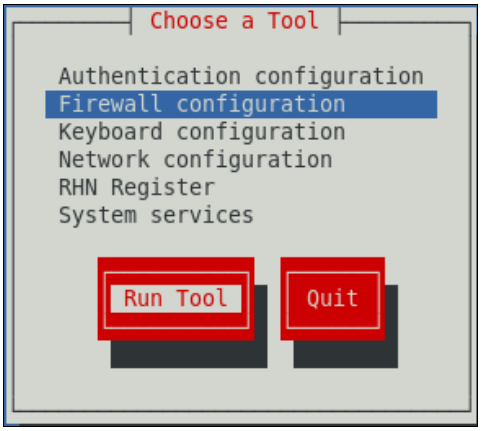


Figure 4-132 Running the Firewall configuration tool

Disable the firewall, go **OK**, and press Enter, as shown in Figure 4-133.

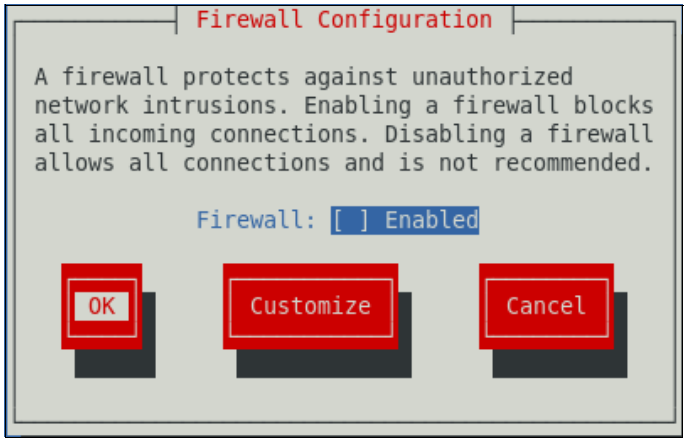


Figure 4-133 Disabling the Linux firewall

7. Now disable the SELinux in your virtual machine. Modify the `/etc/selinux/config` configuration file:

   ```
   # vi /etc/selinux/config
   ```

   Change the `SELINUX` parameter to `disabled`, as shown in Example 4-179.

Example 4-179 Changing SELinux to disabled

```
[root@rhel65 ~]# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

8. Reboot your virtual machine to disable SELinux. Verify whether it was corectly disabled with the **sestatus** command, as shown in Example 4-180.

Example 4-180 Verifying that SELinux is disabled

```
[root@rhel65 ~]# sestatus
SELinux status:                 disabled
```

9. Copy the Red Hat Enterprise Linux installation media ISO file to your virtual machine and set up a Yum repository, which is required for installing the `cloud-init` software:

   a. Add the Red Hat Installation Media ISO file to the `/etc/fstab`:

   ```
   /root/RHEL6.5.iso /media/cdrom    iso9660 loop,ro,auto    0 0
   ```

   b. Create the mount point:

   ```
   # mkdir /media/cdrom
   ```

   c. Mount the installation media:

   ```
   # mount -a
   # df -h /media/cdrom
   Filesystem          Size  Used Avail Use% Mounted on
   /root/RHEL6.5.iso  3.6G  3.6G     0 100% /media/cdrom
   ```

   d. Copy the `media.repo` to the `/etc/yum.repos.d`:

   ```
   # cp /media/cdrom/media.repo /etc/yum.repos.d/media.repo
   ```

   e. Modify the `/etc/yum.repos.d/media.repo` to add the `enabled` and `baseurl` lines:

   ```
   # cat /etc/yum.repos.d/media.repo
   [InstallMedia]
   name=Red Hat Enterprise Linux 6.5
   mediaid=1384196515.415715
   metadata_expire=-1
   gpgcheck=0
   cost=500
   enabled=1
   baseurl=file:///media/cdrom/Server
   ```

   f. Verify with the **yum repolist** command if the Yum repository was successfully setup:

   ```
   # yum repolist
   ```

10. Install and configure any additional packages and software that you want to use in this image (for example, Monitoring tools, backup agents, and so on. You can also install enterprise software to have an image to attend a specific purpose, such as setting up an RHEL 6.5 + DB2 image).

11. To install `cloud-init` software on your virtual machine, in addition to the Red Hat Installation Media Yum repository, it must also have access to the Internet and be registered in the Red Hat Satellite Channel, “Red Hat Enterprise Server (v. 6 for 64-bit x86_64)”, from which additional packages will be installed to attend `cloud-init` dependencies. For instructions on how to subscribe your virtual machine to the Red Hat Satellite, see Chapter 1. Channel Management, in the documentation on the Red Hat website (Support > Product Documentation > Red Hat Satellite > 5.6 > Getting Started Guide):

    http://red.ht/1DWVvqD

    Important: The channel name might change according to the architecture being used in your virtual machines (x86_64 or ppc64).

12. Additional packages are also required from the EPEL repository (Extra Packages for Enterprise Linux), to install the `cloud-init` package. Download and install the EPEL package to set up the EPEL repository:

```
# wget http://fedora.mirror.nexicom.net/epel/6/i386/epel-release-6-8.noarch.rpm
# rpm -ivh /root/epel-release-6-8.noarch.rpm
```

13. Remove any unnecessary RPM package from your virtual machine. According to the experience during the cloud setup for this publication, you should consider removing the following packages, to avoid problems with cloud-init (in case you do not need these packages for some specific reason):

```
# yum remove -y biosdevname plymouth iprutils kbd firewalld \
dracut-config-rescue authconfig linux-firmware
```

14. Log in to the Red Hat Network Customer portal using your Red Hat Network ID, and click **Downloads > Software Components> RPM Package**s. Then, search for `cloud-init`. Download the latest `cloud-init` package as shown in Figure 4-134 (at the time of writing this publication, the latest `cloud-init` version was `cloud-init-0.7.2-2.el6.noarch`).

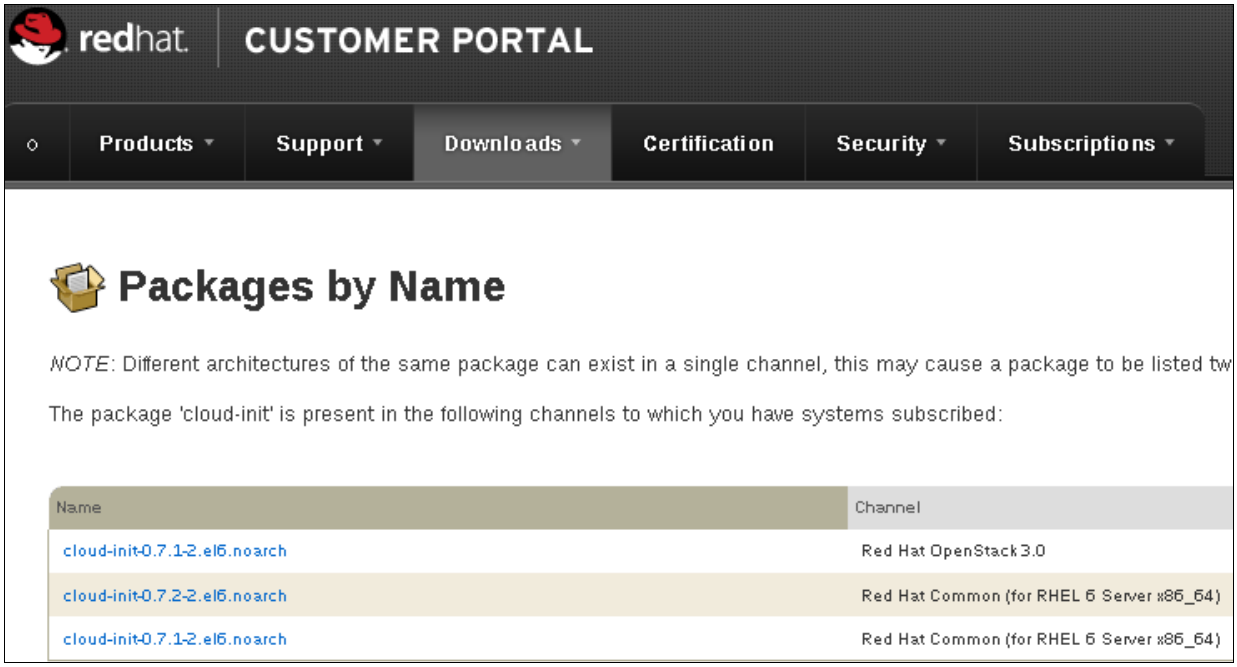


Figure 4-134 Downloading cloud-init from Red Hat Customer portal

15. Install the cloud-init package using Yum, so it can automatically install all dependencies from the Yum repositories:

```
# yum install cloud-init-0.7.2-2.el6.noarch.rpm
```

In a virtual machine created using Red Hat Enterprise Linux 6.5, selecting the **Basic Server** installation, the following dependencies will be automatically installed as shown in Example 4-181.

Example 4-181 Dependencies installed during cloud-init installation

```
Installed:
  cloud-init.noarch 0:0.7.2-2.el6

Dependency Installed:
  PyYAML.x86_64 0:3.10-3.el6
audit-libs-python.x86_6
  libsemanage-python.x86_64 0:2.0.43-4.2.el6             libyaml.x86_64
0:0.1.6-
  python-argparse.noarch 0:1.2.1-2.el6
python-backports.x86_64
  python-boto.noarch 0:2.27.0-1.el6                      python-chardet.noarch
0
  python-configobj.noarch 0:4.6.0-3.el6                  python-markdown.noarch
  python-prettytable.noarch 0:0.7.2-1.el6                python-pygments.noarch
  python-six.noarch 0:1.7.3-1.el6                        python-urllib3.noarch
0
  setools-libs-python.x86_64 0:3.3.7-4.el6

Dependency Updated:
  audit.x86_64 0:2.2-4.el6_5
```

Verify that the cloud-init package was corectly installed using the **rpm** command:

```
# rpm -qa | grep cloud-init
cloud-init-0.7.2-2.el6.noarch
```

16. Install the `net-tools`, `NetworkManager`, `cloud-utils-growpart`, and `git` packages to allow additional functionalities for the `cloud-init` software:

```
# yum install -y net-tools cloud-utils-growpart git NetworkManager
```

17. Install the acpid package to allow the hypervisor to reboot the virtual machine:

```
# yum install -y acpid
```

18. Replace the `/etc/udev/rules.d/70-persistent-net.rules` with an empty file. It contains network persistence rules, including MAC address, which must be removed because the multiple instances which will be deployed based on this image will have different MAC addresses):

```
# > /etc/udev/rules.d/70-persistent-net.rules
# touch /etc/udev/rules.d/70-persistent-net.rules
```

19. Replace the `/lib/udev/rules.d/75-persistent-net-generator.rules` with an empty file. It contains rules to generate the file replaced in step 18:

```
# > /lib/udev/rules.d/75-persistent-net-generator.rules
# touch /lib/udev/rules.d/75-persistent-net-generator.rules
```

Important: The recommended action is to replace the files with empty files rather than just remove them. If you delete the files, you might receive a `udev` warning at boot time.

The reason for removing the network persistence rules in the image is that they can cause network interface in the deployed instance to come up as an interface other than `eth0`, given that the MAC address will be different.

20. Now, modify the network interface configuration file to remove the `IPADDR`, `NETMASK` (or `PREFIX`), and `HWADDR`, change the `BOOTPROTO` to `dhcp`, `ONBOOT` to `yes`, and the `NM_CONTROLLED` parameter to `no`:

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

Note: In other Linux distributions, the location of the network interface configuration file might be different.

Example 4-182 shows a modified network configuration file according to these recommendations. Notice that the `HWADDR` line was removed and so were the network configuration parameters, such as `IPADDR`, `NETMASK`, `GATEWAY`, and `DNS`.

Example 4-182 Modified ifcfg-eth0 file for image creation

```
# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
TYPE=Ethernet
UUID=7bf45025-e869-4110-a579-b5c6b7b26131
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="System eth0"
```

21. Add the `NOZEROCONF` parameter set to `yes` and remove the `GATEWAY` parameter in the `/etc/sysconfig/network` configuration file:

```
# echo "NOZEROCONF=yes" >> /etc/sysconfig/network
# cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=tempvm
NOZEROCONF=yes
```

22. Now, modify the `/etc/cloud/cloud.cfg` file. This is the `cloud-init` configuration file:

```
# vi /etc/cloud/cloud.cfg
```

   a. Change the `disable_root` to `0` (required to allow root login to the deployed instances - root login is disabled by default).

   b. Change the `ssh_pwauth` to `1` (required to allow password authentication - password authentication is disabled by default, requiring ssh key pair injection to log in to the instance).

   c. Change the `ssh_deletekeys` to `1` (for security reasons, remove pre-existing ssh keys during virtual machine deployment).

   d. Add the following lines after the `syslog_fix_perms` parameter (cloud-init uses metadata API to inject key-pairs, modify IP address, hostname and other customizations in the instance. At the time of writing this publication, metadata API is not supported by Cloud Manager, so you need to add the following lines to use a Config Drive to allow cloud-init to inject these modifications in the instance during the deployment):

```
disable_ec2_metadata: True
datasource_list: ['ConfigDrive']
```

e. Note the `default_user name` parameter in the `system_info` section. You can create this user in the instance to allow key-pair injection with this user:

```
system_info:
  distro: rhel
  default_user:
    name: cloud-user
  paths:
    cloud_dir: /var/lib/cloud
    templates_dir: /etc/cloud/templates
  ssh_svcname: sshd
```

Example 4-183 shows the `/etc/cloud/cloud.cfg` file customized in the environment setup for this publication.

Example 4-183 File cloud.cfg customized in the environment setup for this publication

```
# cat /etc/cloud/cloud.cfg
users:
 - default

disable_root: 0
ssh_pwauth:   1

locale_configfile: /etc/sysconfig/i18n
mount_default_fields: [~, ~, 'auto', 'defaults,nofail', '0', '2']
resize_rootfs_tmp: /dev
ssh_deletekeys:   1
ssh_genkeytypes:  ~
syslog_fix_perms: ~
disable_ec2_metadata: True
datasource_list: ['ConfigDrive']

cloud_init_modules:
 - bootcmd
 - write-files
 - resizefs
 - set_hostname
 - update_hostname
 - update_etc_hosts
 - rsyslog
 - users-groups
 - ssh

cloud_config_modules:
 - mounts
 - locale
 - set-passwords
 - timezone
 - puppet
 - chef
 - salt-minion
 - mcollective
 - disable-ec2-metadata
 - runcmd

cloud_final_modules:
```

```
 - rightscale_userdata
 - scripts-per-once
 - scripts-per-boot
 - scripts-per-instance
 - scripts-user
 - ssh-authkey-fingerprints
 - keys-to-console
 - phone-home
 - final-message

system_info:
  distro: rhel
  default_user:
    name: cloud-user
  paths:
    cloud_dir: /var/lib/cloud
    templates_dir: /etc/cloud/templates
  ssh_svcname: sshd

# vim:syntax=yaml
```

23. Create and define a password for the cloud user to allow key pair injection with this user:

```
# useradd cloud-user
# passwd cloud-user
```

24. Make sure that the following `cloud-init` services are set up to start on boot on run level 3 or 5 (depending on what is configured in the `initdefault` entry in the `/etc/inittab` for your image):

```
# chkconfig --list | grep cloud
cloud-config            0:off   1:off   2:on    3:on    4:on    5:on    6:off
cloud-final             0:off   1:off   2:on    3:on    4:on    5:on    6:off
cloud-init              0:off   1:off   2:on    3:on    4:on    5:on    6:off
cloud-init-local        0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

25. Modify the `/boot/grub/menu.lst` configuration file to allow the console to continue working after the installation of cloud-init software:

    a. Open the `/boot/grub/menu.lst` configuration file:

    ```
    # vi /boot/grub/menu.lst
    ```

    b. Add the following lines after the `default` parameter:

    ```
    serial --unit=0 --speed=115200
    terminal --timeout=10 console serial
    ```

    c. Modify the `console` parameter in the `kernel` line to use ttyS0 as follows:

    ```
    kernel /vmlinuz-2.6.32-431.el6.x86_64 ro root=/dev/mapper/vg_tempvm-lv_root
    rd_LVM_LV=vg_tempvm/lv_swap rd_NO_LUKS LANG=en_US.UTF-8 rd_NO_MD
    rd_LVM_LV=vg_tempvm/lv_root SYSFONT=latarcyrheb-sun16 crashkernel=auto
    KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet console=ttyS0
    ```

26. Log in to the KVM host where the virtual machine is running and eject the CDROM drive. In KVM hosts, you need to attach an empty disk at the same target that the CDROM was previously attached (usually hdc):

    a. Check which target device is associated with the CDROM:

    ```
    # virsh dumpxml <instance_name>
    ```

    Look for the following section:

    ```
    <disk type='file' device='cdrom'>
          <driver name='qemu' type='raw'/>
          <source file='/root/RHEL6.5-20131111.0-Server-x86_64-DVD1.iso'/>
          <target dev='hdc' bus='ide' tray='open'/>
          <readonly/>
          <alias name='ide0-1-0'/>
          <address type='drive' controller='0' bus='1' target='0' unit='0'/>
    </disk>
    ```

    b. Run the following command to attach an empty disk at the same target device obtained in step a:

    ```
    # virsh attach-disk --type cdrom --mode readonly rhel65 "" hdc
    Disk attached successfully
    ```

    c. Verify with the **virsh dumpxml <instance_name>** if the cdrom tray is open:

    ```
    <disk type='block' device='cdrom'>
          <driver name='qemu' type='raw'/>
          <target dev='hdc' bus='ide' tray='open'/>
          <readonly/>
          <alias name='ide0-1-0'/>
          <address type='drive' controller='0' bus='1' target='0' unit='0'/>
    </disk>
    ```

27. Shut down the virtual machine. You can use the following command from the KVM host to shutdown the virtual machine:

    ```
    # virsh destroy <instance_name>
    ```

28. Run the **virt-sysprep** command to prepare the virtual machine for being captured as an image:

    ```
    [root@cmokvm ~]# virt-sysprep -d rhel65
    Examining the guest ...
    Performing "yum-uuid" ...
    Performing "utmp" ...
    Performing "udev-persistent-net" ...
    Performing "sssd-db-log" ...
    Performing "ssh-userdir" ...
    Performing "ssh-hostkeys" ...
    Performing "smolt-uuid" ...
    Performing "script" ...
    Performing "samba-db-log" ...
    Performing "rpm-db" ...
    Performing "rhn-systemid" ...
    Performing "random-seed" ...
    Performing "puppet-data-log" ...
    Performing "pam-data" ...
    Performing "package-manager-cache" ...
    Performing "pacct-log" ...
    Performing "net-hwaddr" ...
    ```

```
Performing "net-hostname" ...
Performing "mail-spool" ...
Performing "machine-id" ...
Performing "logfiles" ...
Performing "hostname" ...
Performing "firstboot" ...
Performing "dovecot-data" ...
Performing "dhcp-server-state" ...
Performing "dhcp-client-state" ...
Performing "cron-spool" ...
Performing "crash-data" ...
Performing "blkid-tab" ...
Performing "bash-history" ...
Performing "abrt-data" ...
Performing "lvm-uuids" ...
```

29. Run the **virsh undefine** command to remove the virtual machine definition from the KVM host, preserving its disk:

```
# virsh undefine rhel65
Domain rhel65 has been undefined
```

After completing these steps, the virtual machine disk created in step 2 on page 414 can be used to create an image with the qcow2 format in Cloud Manager. See “Create the image in IBM Cloud Manager” for detailed instructions.

Obtain cloud-ready images from the Linux distributions

Some Linux distributions provide cloud-ready images for download from the Internet. The images already have the `cloud-init` software installed and configured, and you can download these images and add it to your OpenStack cloud for instances deployment. Although it is the easiest way of setting up images, it does not allow customization before you start using it.

CirrOS images

CirrOS is a minimal Linux system that can be used as a test image for KVM Compute nodes. The CirrOS images are available for download at the following website:

`https://launchpad.net/cirros/+download`

The latest CirrOS version in qcow2 format that was available at the time of writing this publication is available at this web address:

`http://download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64-disk.img`

Note: The login account in CirrOS image is `cirros` and the password is `cubswin:)`.

This account has sudo authority to run commands as root. Run the **sudo su -** command to become root.

Ubuntu images

Ubuntu images for x86_64 and ppc64 for KVM and PowerKVM Compute nodes are maintained by Canonical at the following website:

`http://cloud-images.ubuntu.com/`

Note: In Ubuntu images, the login account is `ubuntu`. There is no password defined in the Ubuntu account, and SSH login with password is disabled for the root user. You need to use SSH key pair injection to be able to log in using the Ubuntu account.

The `ubuntu` user has authority to run commands as `root` user using the **`sudo`** command.

Red Hat Enterprise Linux images

You can download Red Hat Enterprise Linux images (which requires valid Red Hat Enterprise Linux subscription) at the following websites:

- RHEL 7

  `https://access.redhat.com/downloads/content/69/ver=/rhel---7/7.0/x86_64/product-downloads`

- RHEL 6

  `https://rhn.redhat.com/rhn/software/channel/downloads/Download.do?cid=16952`

Note: The login account to Red Hat images is `cloud-user`. The `cloud-user` does not have a password defined and SSH login with password is disabled for the `root` user. You must use SSH key pair injection to be able to log in using the `cloud-user` account.

The `cloud-user` has authority to run commands as `root` using the **`sudo`** command.

Fedora images

You can obtain cloud-ready Fedora images from the following website:

`http://cloud.fedoraproject.org/`

Note: The login account for Fedora images is `fedora`. The `fedora` user has no password defined and SSH login with `root` user is disabled. You must use key-pair injection in Cloud Manager to be able to log in to this image using the `fedora` user.

The `fedora` user has authority to run commands as `root` using the **`sudo`** command.

SUSE Linux Enterprise Server (SLES) and openSUSE images

You can use the following tool to create your SLES or openSUSE image:

`http://susestudio.com/`

For information about other sources of cloud-ready images that you can download, see these websites:

- Chapter 2. Get Images, OpenStack Virtual Machine Image Guide

  `http://docs.openstack.org/image-guide/content/ch_obtaining_images.html`

- Downloading Pre-Built Images for OpenStack on the RDOproject.org website:

  `https://www.rdoproject.org/Image_resources`

For more information about how to use SSH key-pair injection to log in to the instances deployed using these images, see 4.15, “Use SSH key pairs in IBM Cloud Manager” on page 485.

Create the image in IBM Cloud Manager

After preparing your image or downloading a cloud-ready image from your Linux distribution, you should have a `qcow2` file which can be used in Cloud Manager to create an image. This image can later be used for deploying instances in your OpenStack cloud. The `qcow2` file can be used for KVM or PowerKVM Compute nodes, depending on the architecture of the Linux installed inside this disk (inside this `qcow2` file). The next topics will allow you to create appropriate x86_64 or ppc64 `qcow2` images, according to the architecture of the Linux installed in this disk.

Create Glance images for KVM Compute nodes

If you have prepared an image using x86_64 platform, use the following steps to add the image in Cloud Manager for instances deployment in KVM Compute nodes:

1. Log in to the Cloud Management dashboard by using the `admin` user ID:

   `https://<controller_node_ip_address>/`

2. Click **Admin > System Panel > Images**, and click **Create Image**.

   a. Provide a name, description (optional), select **Image File** as the Image Source, click **Browse**, and select the **`.qcow2`** (or **`.img`**) file that you prepared in section “Prepare your image” on page 413, or that you downloaded in section “Obtain cloud-ready images from the Linux distributions” on page 424.

   b. In the Architecture field, select **QCOW2** - **QEMU Emulator**.

   c. In the Format field and type `x86_64`. Optionally, you can provide an amount in Gigabytes for Minimum Disk and an amount in megabytes in the Minimum RAM fields.

   d. Click the **Public** check box if you want to allow multiple projects (multiple tenants) to use this image, and click **Create Image**, as shown in Figure 4-135 on page 427.

   Important: You must type `x86_64` in the Architecture field, so that Cloud Manager knows that this image can be deployed only in KVM Compute nodes. If you do not do it, Cloud Manager maight try to deploy this image in PowerKVM Compute nodes and it will fail, given that x86_64 images have a different architecture than ppc64.

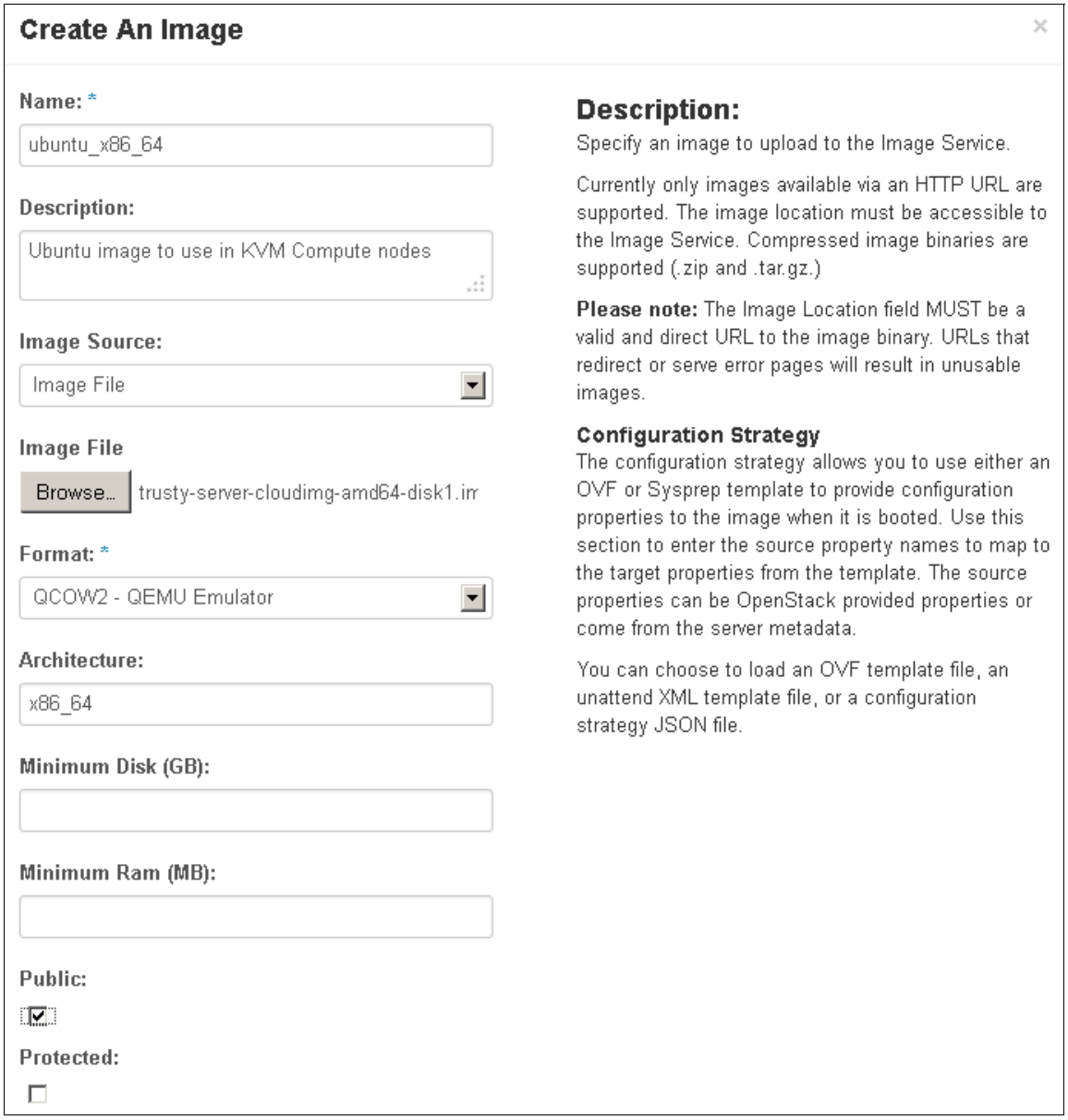


Figure 4-135 Creating an image for KVM Compute nodes

Note: If you provide the Minimum Disk or Minimum RAM information, the deployment of instances using this image and a flavor that does not attend these minimum requirements will fail.

3. You can verify that the image was successfully created in the Cloud Management dashboard by clicking on **Admin > System Panel > Images**, as shown in Figure 4-136.

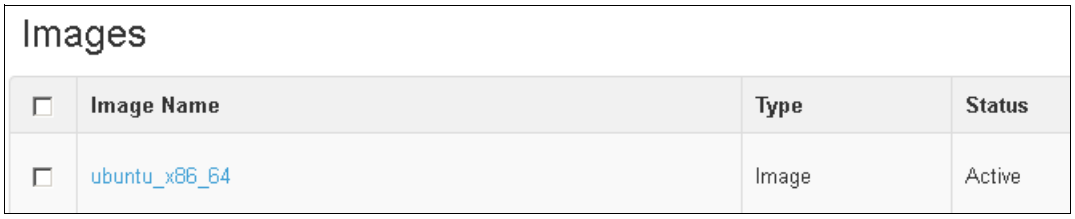


Figure 4-136 Image created in Cloud Management dashboard

4. You can also list the available images using the **`glance image-list`** command in the Controller node, as shown in Example 4-184.

Example 4-184 Output of the glance image-list command

```
[root@cmocontroller ~]# glance image-list
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
| ID                                   | Name                            | Disk Format | Container Format | Size        | Status |
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
| 850fa378-1dc2-4b34-b710-583f301dd12e | Fedora_20_x86_64                | qcow2       | bare             | 210829312   | active |
| 4e06e2c7-389c-4751-93a8-1e47d60b0c5b | ITSO_SDE_AIX_Template_capture_1 | raw         | bare             | 53687091200 | active |
| 63e08c8c-9334-4215-879e-c40e0f681406 | ITSO_SDE_capture_1              | raw         | bare             | 53687091200 | active |
| 354a2183-cdb7-45f6-bfd0-4ce3a4c1025c | PowerVC Default Image           | raw         | ovf              |             | queued |
| df5f6a15-8c21-413c-a577-7f020bc96e2b | RHEL65_x86_64                   | qcow2       | bare             | 338427904   | active |
| 88d74f68-a0b5-499e-bc31-ba39d5aa839b | cirros                          | qcow2       | bare             | 9761280     | active |
| 9af3d32a-80c7-4865-8173-4756cd70e5b6 | rhel65-cloudinit_powerkvm       | qcow2       | bare             | 1893203968  | active |
| 094ce631-2814-42a5-9a78-f202ab7690ca | rhel65ppc64_powerkvm            | qcow2       | bare             | 10737418240 | active |
| f98aead8-ebbc-4cea-901c-a72b97366487 | rhel_x86_cloud_init             | qcow2       | bare             | 6552420352  | active |
| 4daa0c00-3bf9-42aa-85ba-d5e67ba70cce | ubuntu_x86_64                   | qcow2       | bare             | 255394304   | active |
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
```

5. Use the **`glance image-show <image_name>`** command to obtain more details of the image, as shown in Example 4-185.

Example 4-185 Output of the glance image-show <image_name> command

```
[root@cmocontroller ~]# glance image-show ubuntu_x86_64
+-------------------------+--------------------------------------+
| Property                | Value                                |
+-------------------------+--------------------------------------+
| Property 'architecture' | x86_64                               |
| checksum                | 224053ebd52cfa25fefed1c44778c3ac     |
| container_format        | bare                                 |
| created_at              | 2014-09-09T16:10:48.315483           |
| deleted                 | False                                |
| disk_format             | qcow2                                |
| id                      | 4daa0c00-3bf9-42aa-85ba-d5e67ba70cce |
| is_public               | True                                 |
| min_disk                | 0                                    |
| min_ram                 | 0                                    |
| name                    | ubuntu_x86_64                        |
| owner                   | 64de5a3d843c4e9d8173b1f0c2d54f59     |
| protected               | False                                |
| size                    | 255394304                            |
| status                  | active                               |
| updated_at              | 2014-09-09T16:10:50.629240           |
+-------------------------+--------------------------------------+
```

Note: Run **`source /root/openrc`** before running the **`glance`** commands.

After completing these steps, you can deploy instances in your KVM Compute nodes using this x86_64 image. For more information about deploying virtual machines see 4.13, “Manage instances in IBM Cloud Manager” on page 443.

Create Glance images for PowerKVM Compute nodes

If you have prepared an image using ppc64 platform, use the following steps to add the image in Cloud Manager for instances deployment in PowerKVM Compute nodes:

1. Log in to the Cloud Management dashboard by using the `admin` user ID:

   `https://<controller_node_ip_address>/`

2. Click **Admin > System Panel > Images**, and the click **Create Image**.
3. Provide a name, description (optional), and select **Image File** in the Image Source field, click **Browse**, and select the **`.qcow2`** (or **`.img`**) file that you prepared or downloaded previously. Select **QCOW2 - QEMU Emulator** in the Format field and type `ppc64` in the Architecture field. Optionally, you can provide an amount in Gigabytes in the Minimum Disk and an amount in megabytes in the Minimum RAM fields.

   Click the **Public** check box in case you want to allow multiple projects (multiple tenants) to use this image, and click **Create Image**, as shown in Figure 4-137.

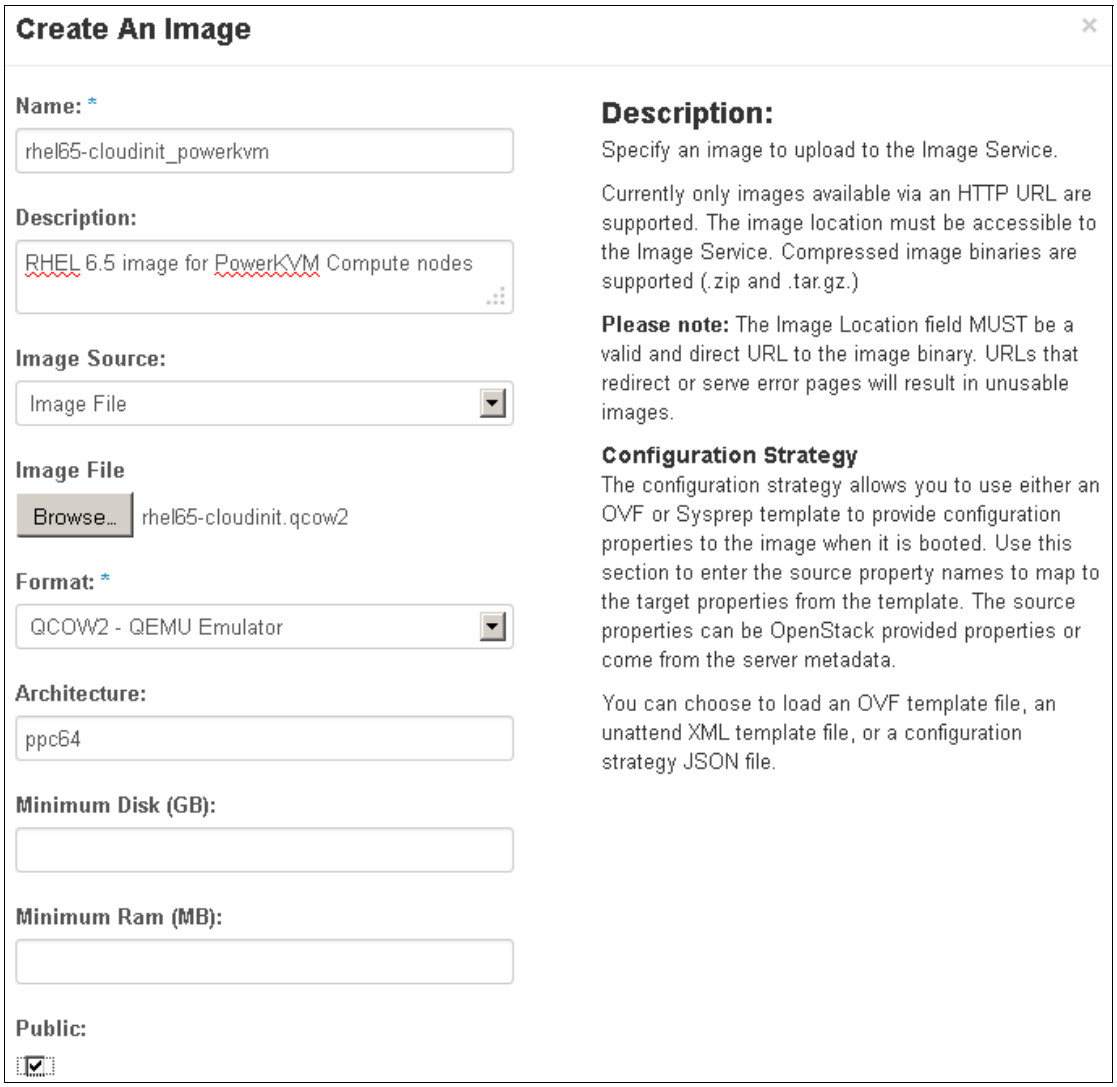


Figure 4-137 Creating images for PowerKVM Compute nodes

Important: You must type `ppc64` in the Architecture field so that Cloud Manager knows that this image can only be deployed in ppc64 hypervisors. If you do not provide this information in this field, Cloud Manager might try to deploy this image in KVM Compute nodes. That will fail because they are x86_64 hypervisors.

Note: If you provide the Minimum Disk or the Minimum RAM information, the deployment of instances using this image and a flavor that does not attend these minimum requirements will fail.

4. You can verify that the image was successfully created in the Cloud Management dashboard by clicking **Admin > System Panel > Images**, as shown in Figure 4-138.



Figure 4-138 Image for PowerKVM Compute nodes

Important: If you only have KVM and PowerKVM Compute nodes in your OpenStack cloud, these steps are enough for allowing Cloud Manager to determine which Compute nodes can be used during the image deployment. If you also have PowerVM Compute nodes in your OpenStack cloud, additional steps are required, given that both PowerKVM and PowerVM Compute nodes have the ppc64 Architecture.

5. Log in as `root` user to the Controller node using SSH.
6. Run the following commands to add the `hypervisor_type` property to your PowerKVM Glance image with the `qemu` value:
   a. Use the **`source`** command to read the variables from the `/root/openrc` file:

      ```
      # source /root/openrc
      ```

   b. Run the **`glance image-list`** command to obtain the image ID for your Glance image, as shown in Example 4-186.

Example 4-186 Output of the glance image-list command

```
[root@cmocontroller ~]# glance image-list
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
| ID                                   | Name                            | Disk Format | Container Format | Size        | Status |
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
| 850fa378-1dc2-4b34-b710-583f301dd12e | Fedora_20_x86_64                | qcow2       | bare             | 210829312   | active |
| 4e06e2c7-389c-4751-93a8-1e47d60b0c5b | ITSO_SDE_AIX_Template_capture_1 | raw         | bare             | 53687091200 | active |
| 63e08c8c-9334-4215-879e-c40e0f681406 | ITSO_SDE_capture_1              | raw         | bare             | 53687091200 | active |
| 354a2183-cdb7-45f6-bfd0-4ce3a4c1025c | PowerVC Default Image           | raw         | ovf              |             | queued |
| df5f6a15-8c21-413c-a577-7f020bc96e2b | RHEL65_x86_64                   | qcow2       | bare             | 338427904   | active |
| 88d74f68-a0b5-499e-bc31-ba39d5aa839b | cirros                          | qcow2       | bare             | 9761280     | active |
| 9af3d32a-80c7-4865-8173-4756cd70e5b6 | rhel65-cloudinit_powerkvm       | qcow2       | bare             | 1893203968  | active |
| 094ce631-2814-42a5-9a78-f202ab7690ca | rhel65ppc64_powerkvm            | qcow2       | bare             | 10737418240 | active |
| f98aead8-ebbc-4cea-901c-a72b97366487 | rhel_x86_cloud_init             | qcow2       | bare             | 6552420352  | active |
| 4daa0c00-3bf9-42aa-85ba-d5e67ba70cce | ubuntu_x86_64                   | qcow2       | bare             | 255394304   | active |
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
```

c. Run the **glance image-update** command to add the hypervisor_type property to the PowerKVM image, as shown in Example 4-187.

Example 4-187 Adding the hypervisor_type property to the image

```
[root@cmocontroller init.d]# glance image-update
094ce631-2814-42a5-9a78-f202ab7690ca --property hypervisor_type=qemu
+----------------------------+--------------------------------------+
| Property                   | Value                                |
+----------------------------+--------------------------------------+
| Property 'architecture'    | ppc64                                |
| Property 'hypervisor_type' | qemu                                 |
| checksum                   | 94050357c61c497c67808373861a94d4     |
| container_format           | bare                                 |
| created_at                 | 2014-08-04T15:04:05.619495           |
| deleted                    | False                                |
| deleted_at                 | None                                 |
| disk_format                | qcow2                                |
| id                         | 094ce631-2814-42a5-9a78-f202ab7690ca |
| is_public                  | True                                 |
| min_disk                   | 10                                   |
| min_ram                    | 1                                    |
| name                       | rhel65ppc64_powerkvm                 |
| owner                      | 64de5a3d843c4e9d8173b1f0c2d54f59     |
| protected                  | False                                |
| size                       | 10737418240                          |
| status                     | active                               |
   | updated_at                 | 2014-08-05T02:15:03.684569           |
| virtual_size               | None                                 |
+----------------------------+--------------------------------------+
```

With this modification, Cloud Manager will look for hypervisors using the `ppc64 Architecture` (valid for PowerKVM and PowerVM) AND also with the `hypervisor_type qemu` (only PowerKVM Compute nodes), so it will be able to deploy the image in the correct Compute nodes. After completing these steps, you can deploy instances in your PowerKVM Compute nodes using this ppc64 image. For more information about deploying virtual machines, see 4.13, “Manage instances in IBM Cloud Manager” on page 443.

4.12.2 Manage images in PowerVM Compute nodes

Creating images for deployment in PowerVM Compute nodes involve creating the images in IBM Power Virtualization Center (PowerVC) and then refreshing Cloud Manager communication services to import these into Cloud Management dashboard. To complete this action, the following steps are required:

1. Create the logical partition and install the operating system (AIX, RHEL, or SLES).
2. Install the Activation Engine (set of scripts to prepare the logical partition for being captured).
3. Capture the image in the IBM Power Virtualization Center (PowerVC).
4. Refresh Cloud Manager services to obtain from PowerVC the information about new images.

Create the logical partition and installing the operating system

The following steps will allow you to create a logical partition (LPAR) and install the operating system, which you will capture later to use as an image in Cloud Manager:

1. Log in to the Hardware Management Console (HMC) and create a logical partition for running AIX or Linux. Select the managed system where the LPAR will run, and click **Configuration > Create Partition > AIX or Linux**. Follow the instructions in the Create LPAR wizard to complete this task.

   Notes: As mentioned in section 4.8, “Add PowerVM Compute nodes to the cloud” on page 274, this publication uses SAN Volume Controller (SVC) in the PowerVC environment. If you use the IBM Storwize V7000 storage subsystem, the instructions are the same. If you use shared storage pools, some steps might be different. The Shared storage pools steps are not covered in this book.

   To capture the logical partition as a PowerVC image, all resources must be virtual (no physical adapter is allowed). Remember to create the Virtual Ethernet Adapters to allow access to the network and Virtual Fibre Channel to connect to the SAN Switch and SVC or V7000.

2. Use dynamic logical partitioning to add the Virtual Fibre Channel Server Adapter to the Virtual I/O Servers. Select the VIOS, click **Dynamic Partitioning > Virtual Adapters**, and add the Virtual Fibre Channel Server Adapter that matches the Virtual Fibre Channel Client Adapter slot number from the logical partition that you just created.
3. Save the VIO profile. Select the VIO Server and click **Configuration > Save Current Configuration**.
4. Log in to the VIOS and run the following commands to map the Virtual Fibre Channel Server Adapter to a physical HBA:

   ```
   $ cfgdev
   $ vfcmap -vadapter vfchostX -fcp fcsX
   ```

   Where `vfchostX` is the Virtual Fibre Channel Server Adapter that you added to the VIO Server and `fcsX` is the physical HBA used to connect to the SAN Switch and SVC or V7000.
5. Obtain the WWPN of the Virtual Fibre Channel Client Adapters in the Properties of the logical partition you created. Select the LPAR, click **Properties**. Go to the Virtual Adapters tab and click the Virtual Fibre Channel Client Adapter. Note the WWPNs, as shown in Figure 4-139.

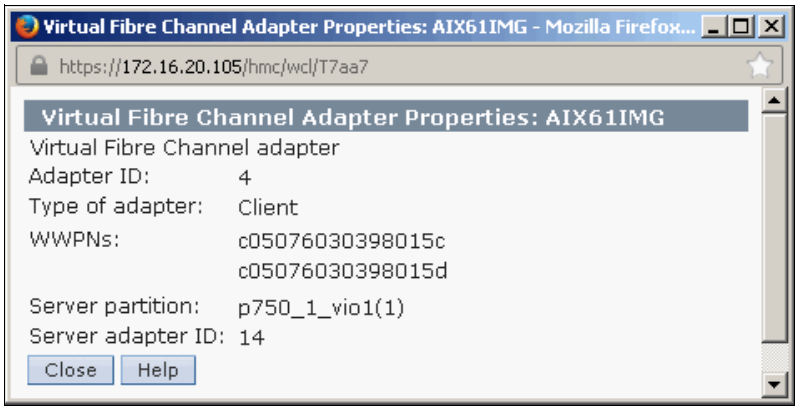


Figure 4-139 Obtaining the Virtual Fibre Channel WWPNs

Note: Each Virtual Fibre Channel Client Adapter has two WWPNs for live partition mobility purposes (moving logical partition between different physical servers).

6. Create zone configuration in your SAN Switch to allow communication between the logical partition and the SVC or V7000 storage subsystem, as shown in Example 4-188.

Example 4-188 Zone configuration between the LPAR and SVC

```
zone:  ITSO_SDE_AIX61IMG_SVC
                c0:50:76:03:03:98:01:5c
                c0:50:76:03:03:98:01:5d
                c0:50:76:03:03:98:01:5e
                c0:50:76:03:03:98:01:5f
                50:05:07:68:01:10:4d:88
                50:05:07:68:01:20:4d:88
                50:05:07:68:01:10:4f:45
                50:05:07:68:01:20:4f:45
```

Note: Check the SAN Switch documentation for instructions and best practices on setting up the zone configuration.

7. Log in to the SVC or V7000 storage subsystem, create the host, and add the port definitions, as shown in Figure 4-140.

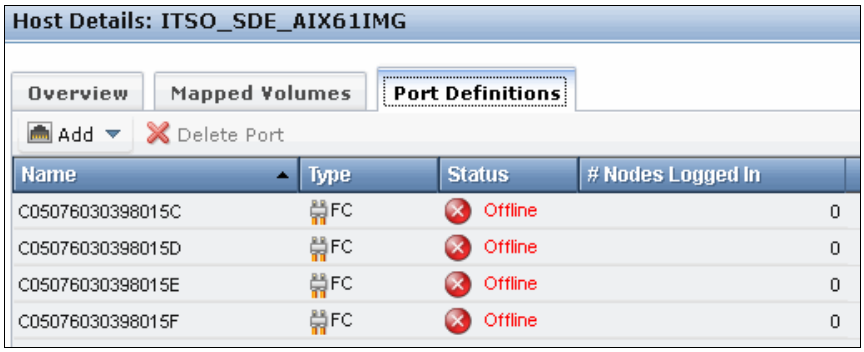


Figure 4-140 Host and Port Definitions in SVC or V7000

8. Create the volume that will be used to install the operating system and map to the host, as shown in Figure 4-141.

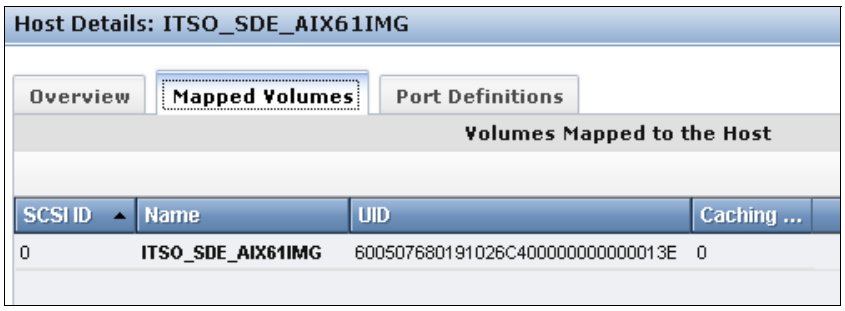


Figure 4-141 Volumes mapped to the host

Note: For more information about SVC configuration and use, see *IBM SAN Volume Controller 2145-DH8 Introduction and Implementation*, SG24-8229:

http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg248229.html

9. Map the DVD drive to the LPAR and insert the operating system installation disk to start the installation. You can use a VIOS Media Repository (Virtual Media Library) to install the operating system using an ISO file.

Note: For instructions on using a VIOS Media Repository / Virtual Media Library, see the IBM Technote titled “How to configure a VIOS Media Repository/Virtual Media Library (ex. AIX Install/Restore):”

http://www.ibm.com/support/docview.wss?uid=isg3T1013047

10. Install the operating system in the logical partition.
    - For AIX installation instructions, see “Installing new and complete overwrite BOS from media” in the IBM Knowledge Center:

      http://ibm.co/1OFtq2n
    - For Red Hat Enterprise Linux installation instructions, see the RHEL Installation Guide:

      http://red.ht/1xEzlge
    - For SUSE Linux Enterprise Server installation instructions, see the SUSE Deployment Guide:

      http://bit.ly/1KuOXoy

    After completing this step, you can do all the customization you need for your environment, for example installing monitoring tools, backup agents, specific configurations, and so on. You can also install applications if you plan to have an image specific to an application, for example an image running AIX + DB2. After you complete the customization tasks, you can install the Activation Engine.

Note: For Linux systems accessing SAN volumes, remember to set up the multipathing configuration. For more information, see the following websites:

- “Configuring Device Mapper Multipath (DMMP) tool for hosts running the Linux operating system” in the IBM Knowledge Center:

  http://ibm.co/1ERPItO
- “Settings for Linux hosts” in the IBM Knowledge Center:

  http://ibm.co/1GGdtSl

Install Activation Engine

Activation Engine is a set of scripts provided in PowerVC for preparing the virtual machine for being captured. It will execute specific actions to avoid duplicated information about the virtual machines created based on this image, such as a duplicated MAC aAddress, wrong network interface naming (preparing Linux `udev` and AIX ODM), wrong disk naming, and so on.

Important: In Linux images, consider installing the IBM Installation Toolkit to add the RMC daemon so the virtual machine can communicate with the HMC for Dynamic logical partitioning and Live Partition Mobility purposes. For more information about installing IBM Installation Toolkit, see “Install IBM Installation Toolkit for PowerLinux” on page 203.

For SUSE Linux Enterprise Server, additional steps are required before installing the Activation Engine.

For both RHEL and SLES, you have the option to use `cloud-init` rather than the Activation Engine to prepare the image.

For more information about SLES additional steps and use of cloud-init in the PowerVC images, see *IBM PowerVC Version 1.2.1 Introduction and Configuration*, SG24-8199:

`http://www.redbooks.ibm.com/abstracts/sg248199.html`

Follow these steps to install and enable Activation Engine:

1. Log in to the PowerVC server and look for the Activation Engine package `vmc.vsae.tar` under `/opt/ibm/powervc/activation-engine`.
2. Copy this file to the virtual machine that you want to capture. There is no specific location that this file should be placed.
3. Log in to the virtual machine that you will capture and extract the `vmc.vsae.tar` file:

   ```
   # tar -xvf vmc.vsae.tar
   ```

4. Run the Activation Engine installation:
   a. For AIX virtual machines, ensure that the `JAVA_HOME` variable is set and points to a valid Java Runtime Environment path, and then run the `aix-install.sh` script:

      ```
      # export JAVA_HOME=/usr/java5/jre
      # ./aix-install.sh
      ```

   b. For Linux virtual machines, run the `linux-install.sh` script:

      ```
      # ./linux-install.sh
      ```

5. Execute the Activation Engine to prepare the virtual machine for being captured:

   Important: This step will shut down the virtual machine. Make sure that there are no active users or applications before executing this command.

   There are also additional steps required in case this virtual machine has been captured previously. Please see IBM Redbooks *IBM PowerVC Version 1.2.1 Introduction and Configuration*, SG24-8199 for more information.

   ```
   # /opt/ibm/ae/AE.sh -R
   ```

   Example 4-189 shows an AIX 6100-09 virtual machine being prepared for capture:

Example 4-189 Executing Activation Engine in an AIX virtual machine

```
# /opt/ibm/ae/AE.sh -R
JAVA_HOME=/usr/java5/jre
[2014-09-11 10:48:16,461] INFO: Looking for platform initialization commands
[2014-09-11 10:48:16,470] INFO: OS: AIX Version: 6
.1

[2014-09-11 10:48:16,967] INFO: No initialization commands found....continuing
```

```
[2014-09-11 10:48:16,971] INFO: Base PA: /opt/ibm/ae/ovf-env-base.xml
[2014-09-11 10:48:16,973] INFO: VSAE Encryption Level:  Disabled
[2014-09-11 10:48:16,976] INFO: CLI parameters are '['AE/ae.py', '-R']'
[2014-09-11 10:48:16,976] INFO: AE base directory is /opt/ibm/ae/
[2014-09-11 10:48:16,996] INFO: Resetting system. AP file: None. Interactive:
False
[2014-09-11 10:48:17,148] INFO: In reset
[2014-09-11 10:48:17,148] INFO: Resetting products
[2014-09-11 10:48:17,150] INFO: Start to reset com.ibm.ovf.vmcontrol.system
[2014-09-11 10:48:17,510] INFO: Start to reset
com.ibm.ovf.vmcontrol.restore.network
[2014-09-11 10:48:20,878] INFO: Resetting the operating system
[2014-09-11 10:48:20,880] INFO: Cleaning AR and AP directories
[2014-09-11 10:48:20,887] INFO: Shutting down the system

SHUTDOWN PROGRAM
Thu Sep 11 10:48:22 CDT 2014

Broadcast message from root@AIX61IMG (tty) at 10:48:22 ...

shutdown: PLEASE LOG OFF NOW !!!
System maintenance is in progress.
All processes will be killed now.

Broadcast message from root@AIX61IMG (tty) at 10:48:22 ...

shutdown: THE SYSTEM IS BEING SHUT DOWN NOW

JAVA_HOME=/usr/java5/jre
[2014-09-11 10:48:31,744] INFO: Looking for platform initialization commands
[2014-09-11 10:48:31,753] INFO: OS: AIX Version: 6
.1

[2014-09-11 10:48:32,299] INFO: No initialization commands found....continuing
[2014-09-11 10:48:32,305] INFO: Base PA: /opt/ibm/ae/ovf-env-base.xml
[2014-09-11 10:48:32,309] INFO: VSAE Encryption Level:  Disabled
[2014-09-11 10:48:32,311] INFO: CLI parameters are '['AE/ae.py', '-d', 'stop']'
[2014-09-11 10:48:32,315] INFO: AE base directory is /opt/ibm/ae/
[2014-09-11 10:48:32,344] INFO: Stopping AE daemon.
[2014-09-11 10:48:32,359] INFO: AE daemon was not running.
0513-044 The sshd Subsystem was requested to stop.
JAVA_HOME=/usr/java5/jre
[2014-09-11 10:48:37,401] INFO: Looking for platform initialization commands
[2014-09-11 10:48:37,410] INFO: OS: AIX Version: 6
.1

[2014-09-11 10:48:37,927] INFO: No initialization commands found....continuing
[2014-09-11 10:48:37,933] INFO: Base PA: /opt/ibm/ae/ovf-env-base.xml
[2014-09-11 10:48:37,936] INFO: VSAE Encryption Level:  Disabled
[2014-09-11 10:48:37,937] INFO: CLI parameters are '['AE/ae.py', '-d', 'stop']'
[2014-09-11 10:48:37,938] INFO: AE base directory is /opt/ibm/ae/
[2014-09-11 10:48:37,971] INFO: Stopping AE daemon.
[2014-09-11 10:48:37,977] INFO: AE daemon was not running.
JAVA_HOME=/usr/java5/jre
[2014-09-11 10:48:42,711] INFO: Looking for platform initialization commands
```

```
[2014-09-11 10:48:42,721] INFO: OS: AIX Version: 6
.1

[2014-09-11 10:48:43,244] INFO: No initialization commands found....continuing
[2014-09-11 10:48:43,250] INFO: Base PA: /opt/ibm/ae/ovf-env-base.xml
[2014-09-11 10:48:43,253] INFO: VSAE Encryption Level:  Disabled
[2014-09-11 10:48:43,256] INFO: CLI parameters are '['AE/ae.py', '-d', 'stop']'
[2014-09-11 10:48:43,256] INFO: AE base directory is /opt/ibm/ae/
[2014-09-11 10:48:43,276] INFO: Stopping AE daemon.
[2014-09-11 10:48:43,288] INFO: AE daemon was not running.

Wait for '....Halt completed....' before stopping.
Error reporting has stopped.
```

After completing this step, you are ready to capture the image in PowerVC.

Create the image in IBM Power Virtualization Center (PowerVC)

The following topics will allow you to capture the virtual machine you have just prepared as an image in the PowerVC:

1. Log in to the PowerVC server and click virtual machines, as shown in Figure 4-142.

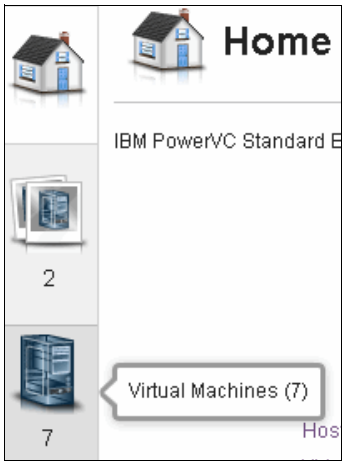


Figure 4-142 Accessing virtual machines information in PowerVC

2. Click **Manage Existing**.
3. Select the Managed System where the virtual machine was created and click **Continue**, as shown in Figure 4-143 on page 438.

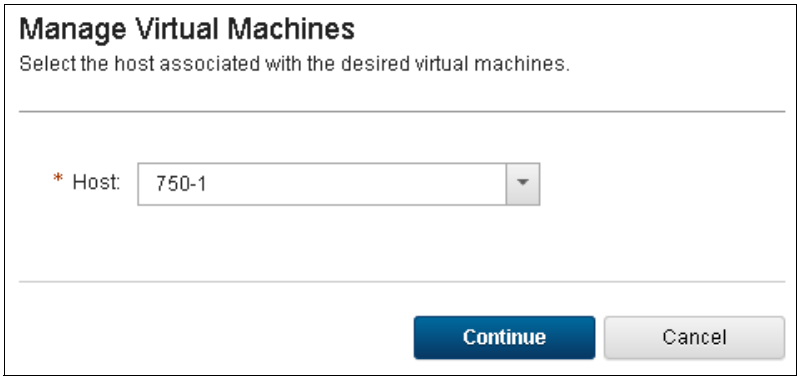


Figure 4-143 Managing existing virtual machines

4. Click **Select specific virtual machines**, and then select the virtual machine you have just created and prepared and click `Manage`, as shown in Figure 4-144.

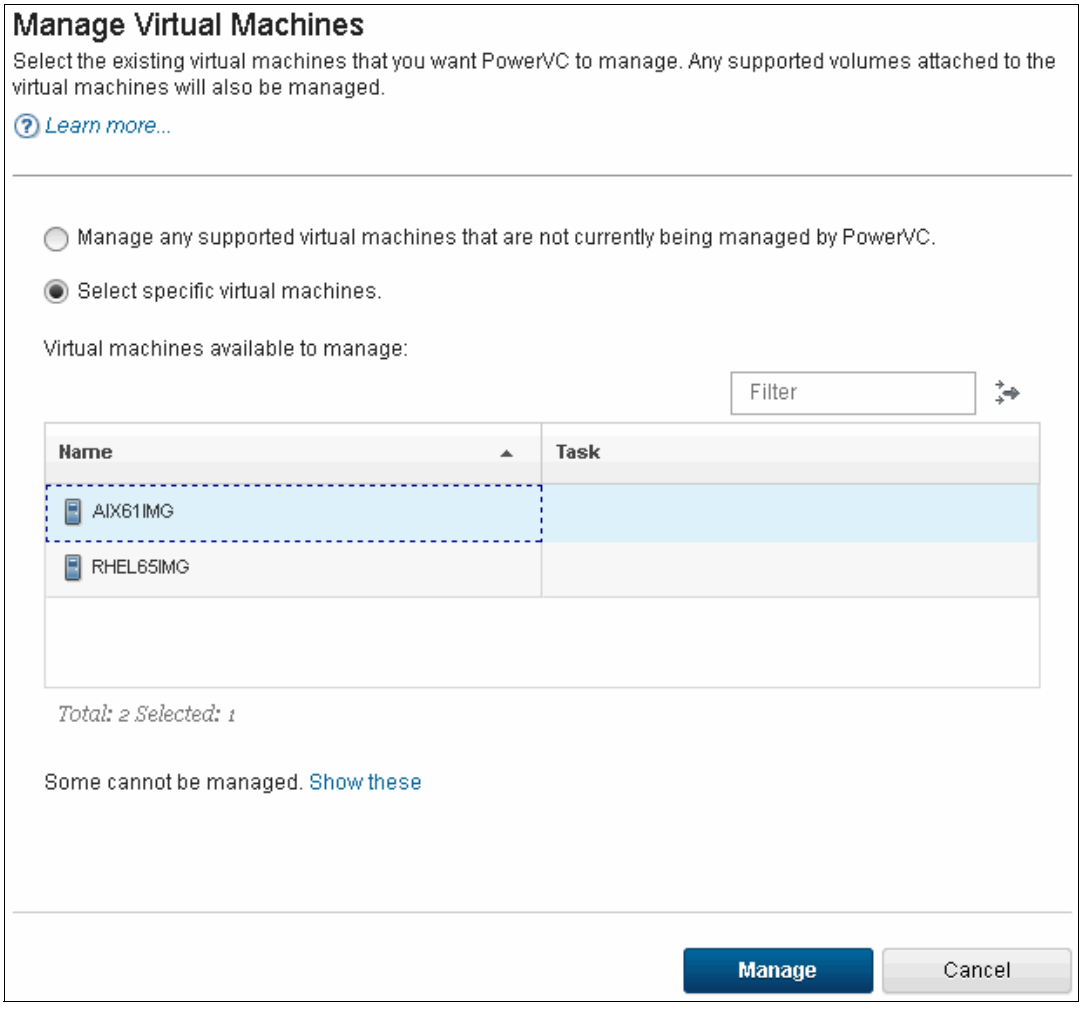


Figure 4-144 Managing a specific virtual machine on PowerVC

5. The action will continue running in background. Click **Messages** at the top of the window and wait until you see the message confirming that the virtual machines were managed by PowerVC.

6. Now click **Virtual Machines**, select the virtual machine, and click **Capture**, as shown in Figure 4-145.

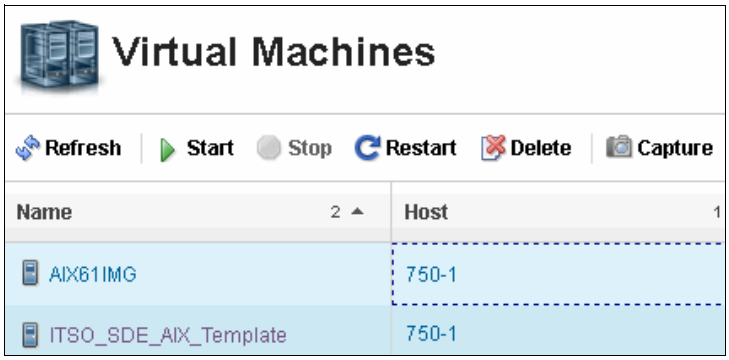


Figure 4-145 Selecting the virtual machine and capturing it

7. In case you have more than one disk associated with the virtual machine, select the disk where the operating system is installed, which contains the Boot volume. Select the operating system type that is running in this virtual machine (AIX, RHEL, or SLES) and click **Continue**, as shown in Figure 4-146.

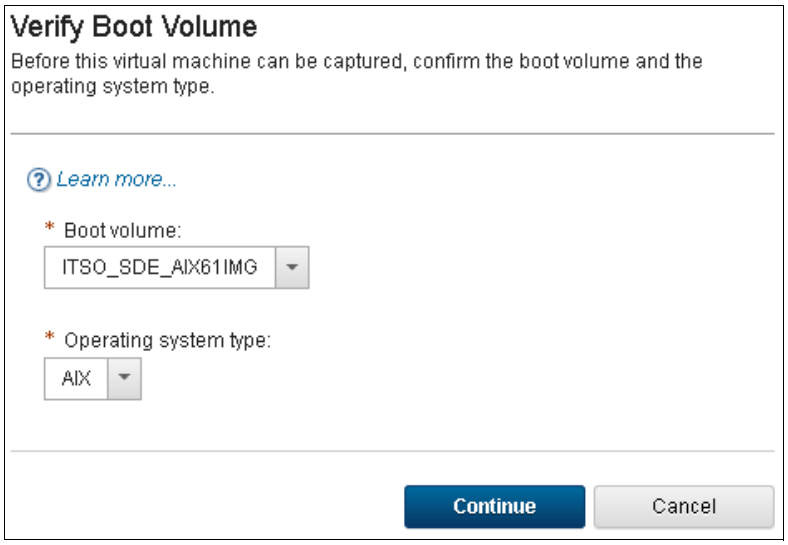


Figure 4-146 Verifying the boot volume

8. Click **Continue** to show that the virtual machine was prepared before capturing, as shown in Figure 4-147.

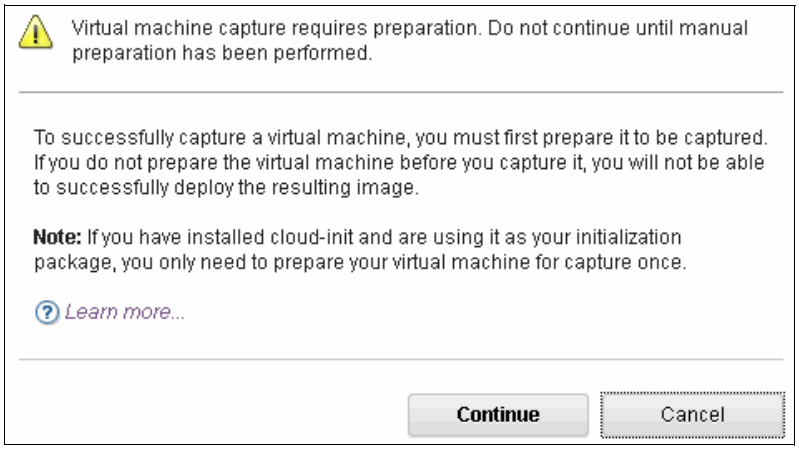


Figure 4-147 Confirming that the virtual machine was previously prepared

9. (Optional) Provide a name for your image and click **Capture**, as shown in Figure 4-148.

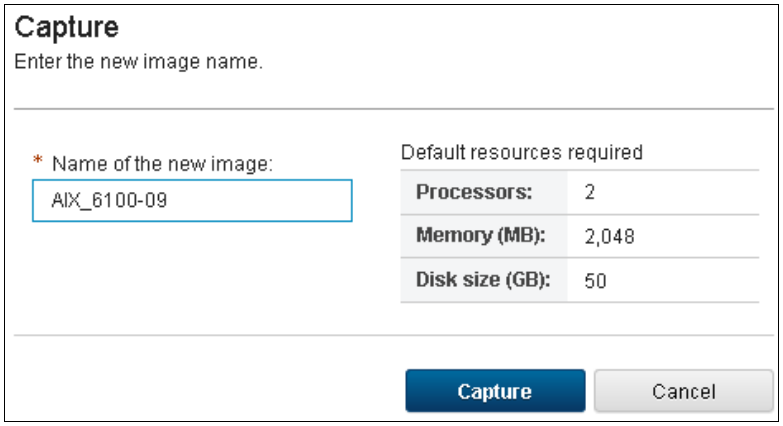


Figure 4-148 Providing a name for the image

10. Click **Images** at the left, and wait until your Image shows the Active state, as shown in Figure 4-149.

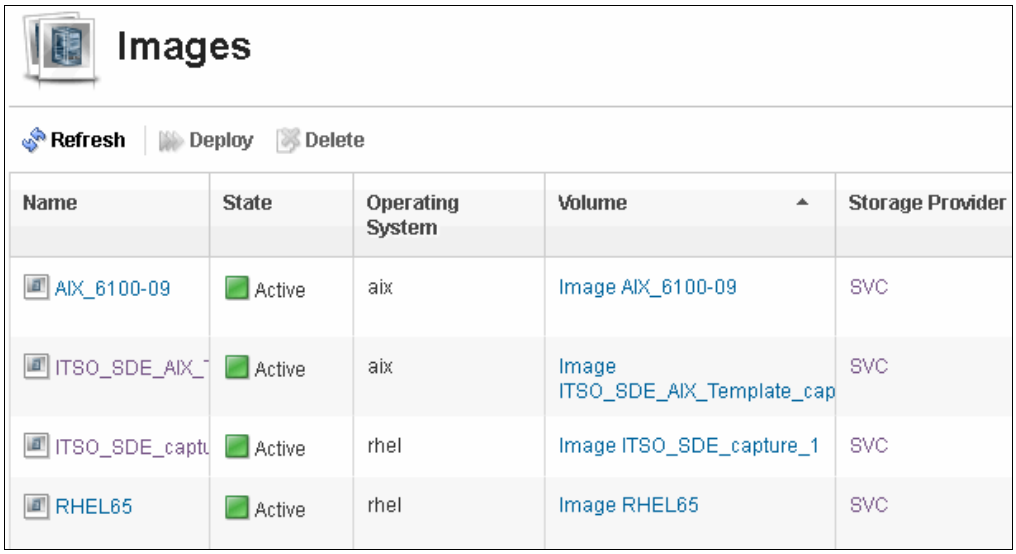


Figure 4-149 Images defined in PowerVC

11. In this process, the Cinder component from PowerVC, which is integrated with SVC storage subsystem will create a snapshot in the SVC. You can see the snapshot volume in the Storage Volumes section of the PowerVC, as shown in Figure 4-150.

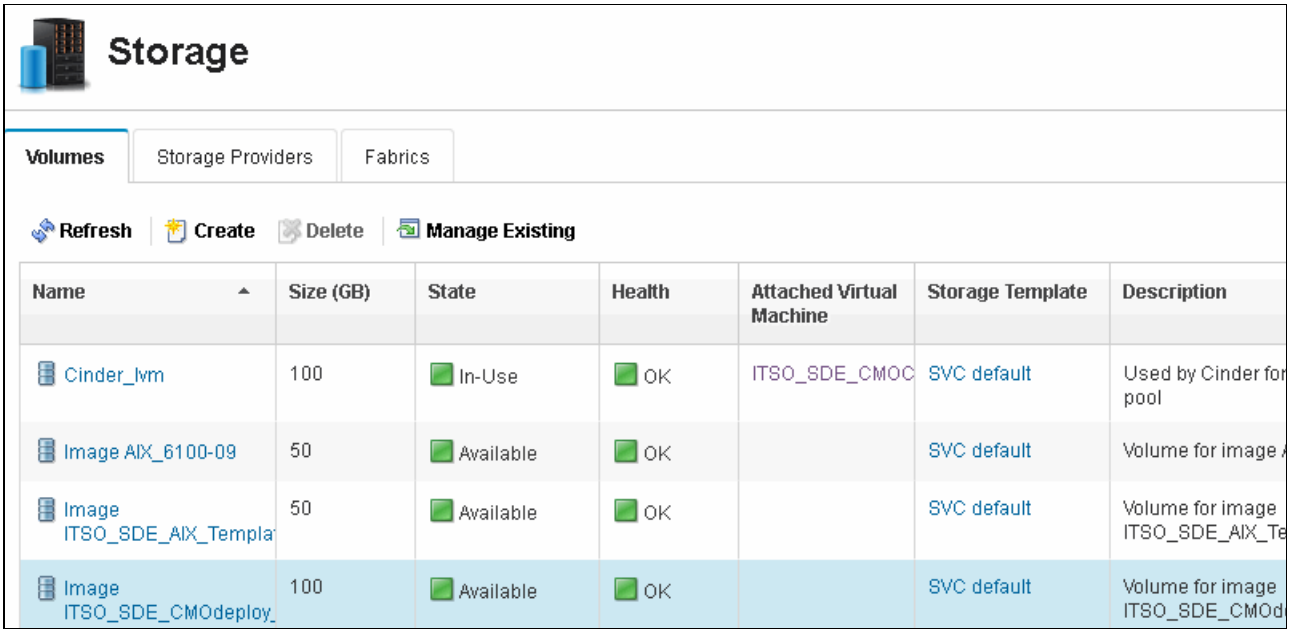


Figure 4-150 Snapshot volume for the image captured in PowerVC

12. A FlashCopy volume is created in SVC. It will be used to deploy new virtual machines based on this image. The FlashCopy volume can be seen in SVC under **Copy Services > FlashCopy**, as shown in Figure 4-151.

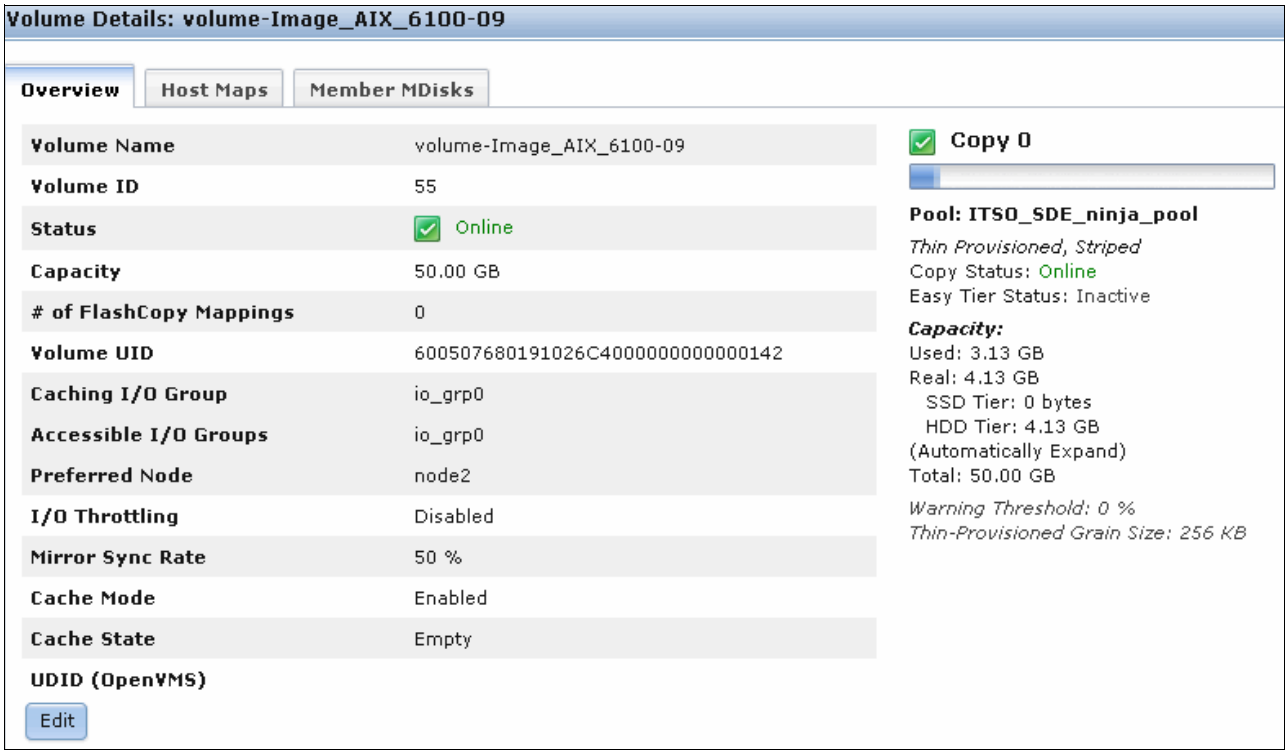


Figure 4-151 FlashCopy created in SVC during image creation in PowerVC

After completing these steps, the image is ready for being added to the Cloud Manager.

Import the PowerVC images in IBM Cloud Manager

When the image is creating in PowerVC server, the PowerVC driver from the Controller node automatically synchronize the images information between PowerVC and Cloud Manager. The image synchronization action is performed every five minutes. When the new image is detected by the PowerVC driver, you should see the following message in the `/var/log/powervc/glance-powervc.log` from the Controller node, as shown in Example 4-190.

Example 4-190 Image synchronization by PowerVC driver

```
2014-09-11 14:33:39.909 24029 INFO powervc.glance.manager.manager [-] Performing
periodic image synchronization...
.
.<some lines were removed>
.
2014-09-11 14:33:50.121 24029 INFO powervc.glance.manager.manager [-] Creating
image 'AIX_6100-09' on the local hosting OS
2014-09-11 14:33:51.241 24029 INFO powervc.glance.manager.manager [-] Periodic
image synchronization is complete. Elapsed time: 11.33 seconds (local:1/2/0,
powervc:0/2/0)
```

If you cannot wait until the next synchronization cycle is completed, you can restart the `openstack-glance-powervc` service in the Controller node:

```
# service openstack-glance-powervc restart
```

When the synchronization is completed, you will be able to see the new image in the Cloud Management dashboard under **Admin > System Panel > Images**, as shown in Figure 4-152.

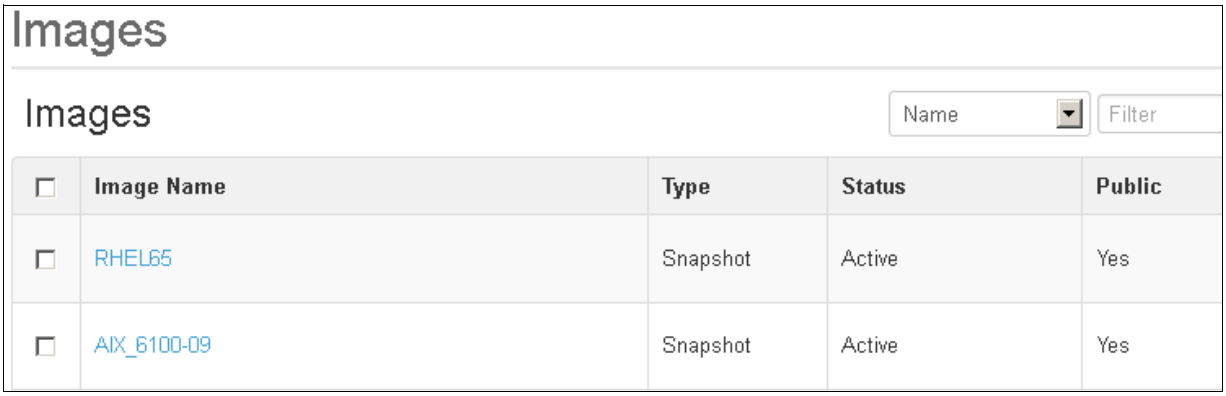


Figure 4-152 Images from PowerVC synchronized with IBM Cloud Manager

4.13 Manage instances in IBM Cloud Manager

In Cloud Manager, the virtual machines deployed in your cloud are called instances. In case the instructions on section 4.9, “Enable the self-service portal” on page 311 were followed, there are three interfaces that can be used to manage instances in your Cloud. Some basic operations that can be performed in these interfaces are covered in this section under the following topics:

- Managing instances by using Cloud Management dashboard
- Managing instances by using the self-service portal
- Managing instances by using the command-line interface

4.13.1 Manage instances by using Cloud Management dashboard

This section allows you to perform basic instance management using the Cloud Management dashboard, which is the Cloud Manager interface based on the OpenStack Horizon project. This interface can be accessed using the Controller node fully qualified domain name:

```
https://<controller_node_fqdn>
```

After you log in to the Cloud Management dashboard, most of the operations can be performed after selecting **Project > Compute > Instances**, as shown in Figure 4-153.

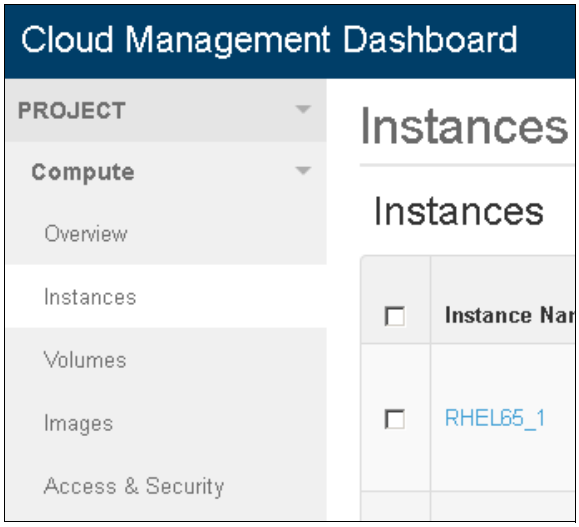


Figure 4-153 Using Cloud Management dashboard

Deploy instances

You can use IBM Cloud Management dashboard to deploy instances on your cloud, which means creating virtual machines based on the images you have created using instructions from section 4.12, “Manage images in IBM Cloud Manager” on page 412. The following instructions are provided in these topics:

- “Deploy KVM instances using Cloud Management dashboard” on page 445
- “Deploy PowerKVM instances using Cloud Management dashboard” on page 450

Note: The instances deployed in KVM and PowerKVM Compute nodes will be placed in the /var/lib/nova/instances directory by default. If you want to place your instances in a different file system (for example a GPFS shared file system), you should modify the `instances_path` variable in the `/etc/nova/nova.conf` file in the *Compute node*:

```
[DEFAULT]

# LOGS/STATE
debug=false
verbose=true
auth_strategy=keystone
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
log_dir=/var/log/nova
state_path=/var/lib/nova
instances_path=/gpfs/sdegpfs1/instances
instance_name_template=instance-%08x
network_allocate_retries=0
lock_path=/var/lib/nova/lock
```

After completing this modification, restart the openstack-nova-compute service:

```
# service openstack-nova-compute restart
```

Deploy KVM instances using Cloud Management dashboard

The following steps allow you to deploy instances in KVM Compute nodes using the Cloud Management dashboard:

1. Log in to the Controller node and click **Project > Compute > Instances**.
2. Click **Launch Instance**.
3. In the Details tab, provide the following information:
   - Availability Zone. Select the **Availability Zone** if you have created Host Aggregates.
   - Instance Name. Provide a name for your instance (if you request more than one instance to be deployed at the same time, this will be used as a prefix in the instance names).
   - Flavor. Select the flavor that will be used by this instance. The flavor will determine the amount of memory and CPU and the disk size of the new instance.
   - Instance Count. Inform how many instances you want to deploy at the same time.
   - Instance Boot Source. Select **Boot from Image** to use the qcow2 images prepared using the KVM Compute nodes.
   - Image Name. Select the image that will be used to deploy this virtual machine (this image will be copied from the Glance repository to the Compute node and used as the disk for the instance).

Figure 4-154 shows an example of deploying two Red Hat Enterprise Linux 6.5 images in KVM Compute nodes using the `m1.small` flavor. The image used in this action was prepared in section 4.12.1, “Manage images in KVM and PowerKVM Compute nodes” on page 413.

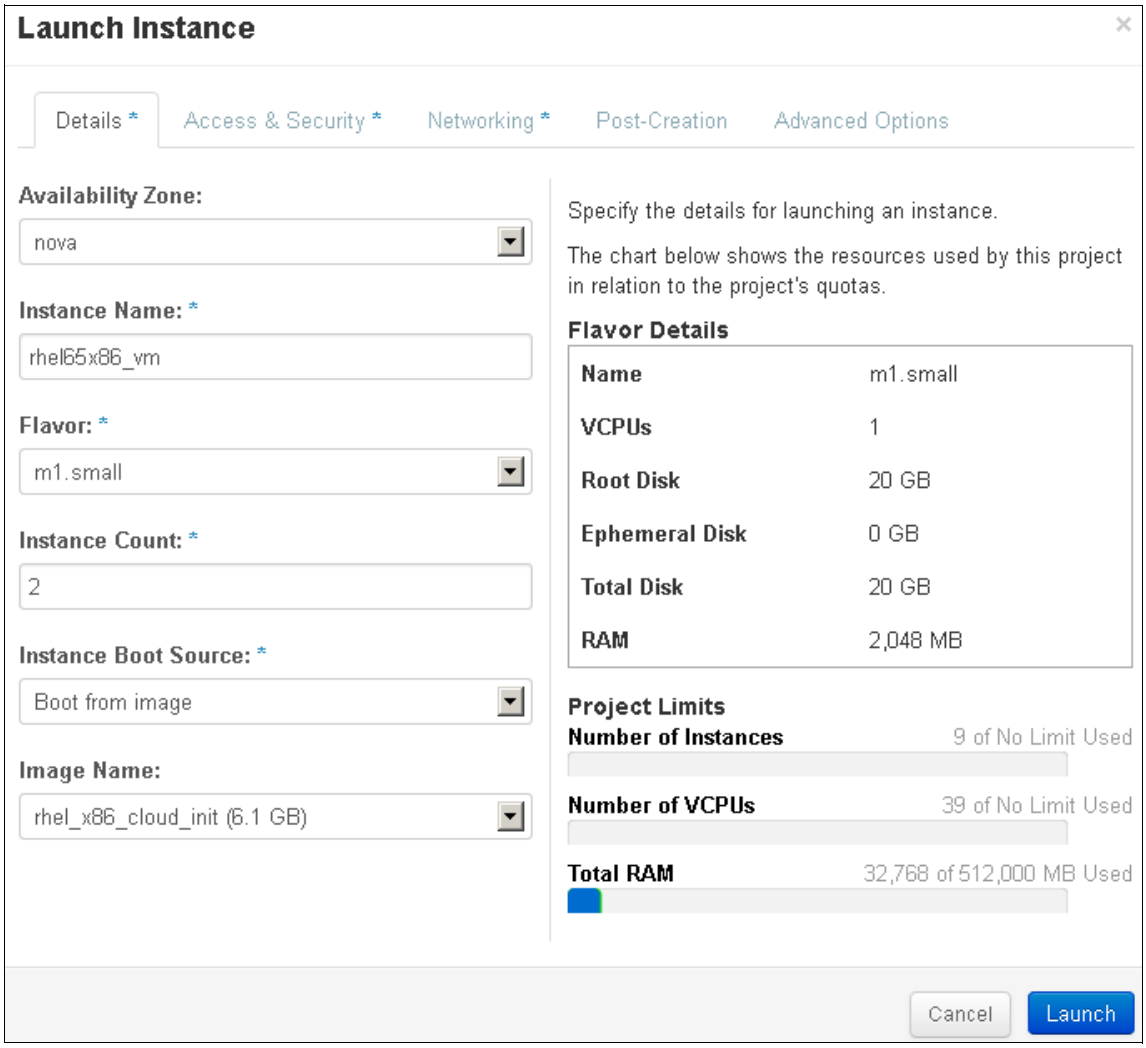


Figure 4-154 Deploying two instances in KVM Compute nodes

4. Click the **Access and Security** tab and (optionally) select an SSH key pair to inject in the instance, so you can access it without password authentication. Keep the `default Security Group` enabled (security groups are set of iptables rules applied to an instance network), as shown in Figure 4-155.

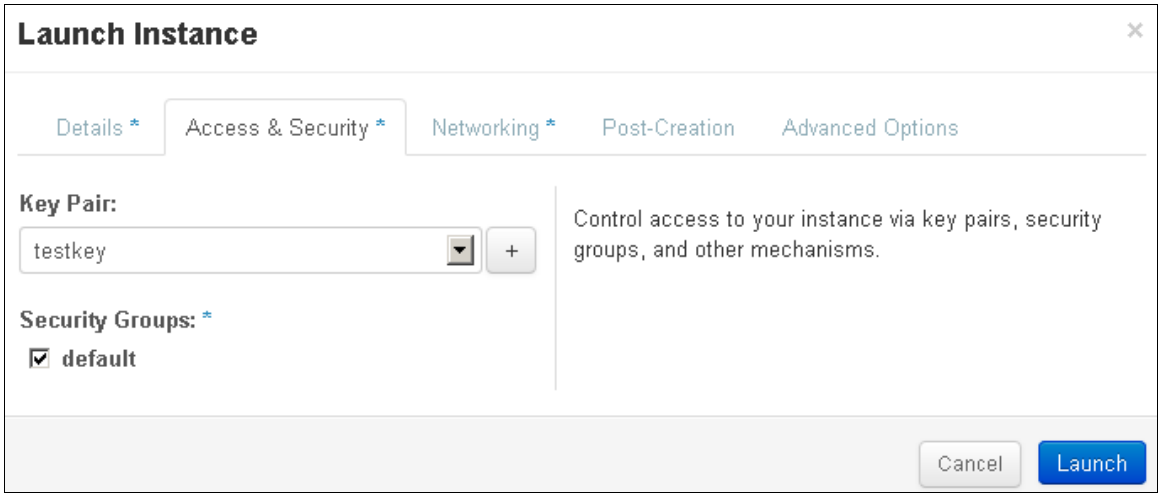


Figure 4-155 Selecting an SSH key pair for injection on the instances

5. Click the **Networking** tab and drag networks that you want to provide to your instances from the Available Networks section to the Selected Networks section, as shown in Figure 4-156.

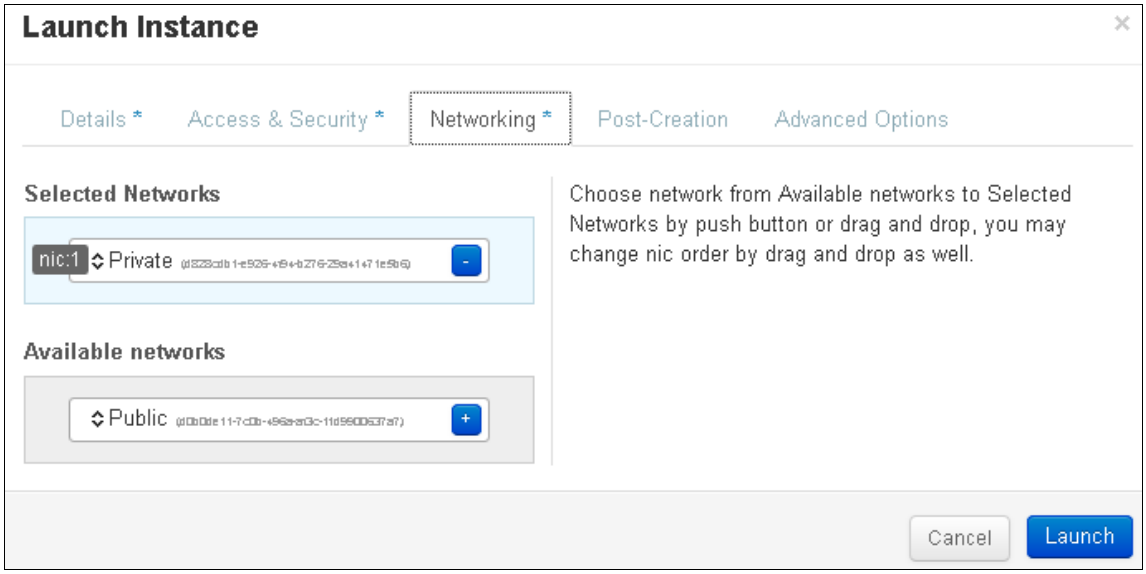


Figure 4-156 Selecting the networks to provide to the instances

6. On the Post-Creation tab, you can optionally provide a Customization Script to execute in the instance after it is deployed, as shown in Figure 4-157. The image must have been prepared using cloud-init to allow you the use of a customization script.

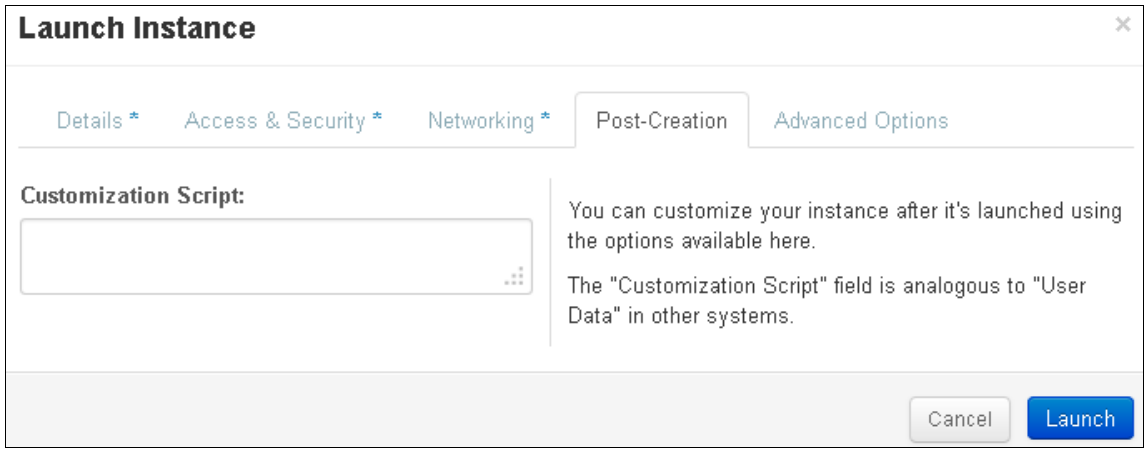


Figure 4-157 Customization Script during instance deployment

7. In the `Advanced Options` tab, you can optionally modify the disk partitioning schema of the instance being deployed (valid only for Linux instances), as shown in Figure 4-158.

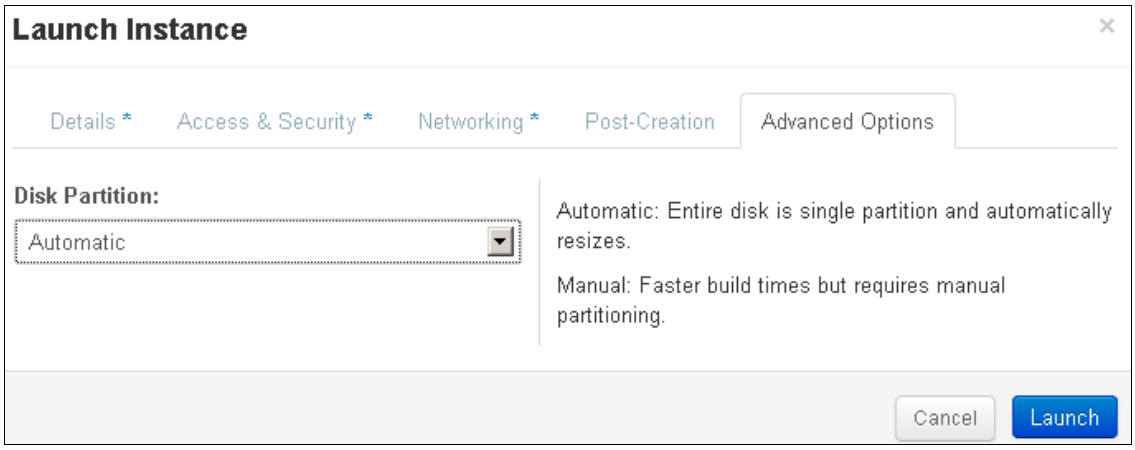


Figure 4-158 Advanced Options during instances deployment

8. Click `Launch` to start the instances deployment.

9. You can see that during the deployment, the instances will appear in the Cloud Management dashboard with the `Spawning` information in the `Task` field, as shown in Figure 4-159. The Spawning task means that the image is being copied to the Compute node.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task
rhel65x86_vm-8faa62b5-0437-4ecc-973c-0cef719f3aaa	rhel_x86_cloud_init		m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk	testkey	Build	nova	Spawning
rhel65x86_vm-44882946-09c4-4db8-873e-4713f6f06f86	rhel_x86_cloud_init		m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk	testkey	Build	nova	Spawning

Figure 4-159 Instances Spawning during deployment

10. When the deployment is complete, you should be able to see the instance IP address from the Cloud Management dashboard and it will be available on the network, as shown in Figure 4-160.

Instance Name	Image Name	IP Address	Size
rhel65x86_vm-8faa62b5-0437-4ecc-973c-0cef719f3aaa	rhel_x86_cloud_init	192.168.21.27	m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk
rhel65x86_vm-44882946-09c4-4db8-873e-4713f6f06f86	rhel_x86_cloud_init	192.168.21.28	m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk

Figure 4-160 Instance IP address after deployment

You may notice that you have not selected the KVM Compute node where the instance is going to run. This was automatically selected by ego according to the policy configured in Platform Resource Scheduler and the current workload in the KVM Compute nodes. In this case only KVM Compute nodes are considered, based on the image properties ('`architecture`' property from the Glance image).

You can see where the instance was deployed in the Host column by clicking on **Admin > System Panel > Instances**, as shown in Figure 4-161.

Project	Host	Name	Image Name	IP Address
Public	cmo1.itso.ibm.com	rhel65x86_vm-8faa62b5-0437-4ecc-973c-0cef719f3aaa	rhel_x86_cloud_init	192.168.21.27
Public	cmo2.itso.ibm.com	rhel65x86_vm-44882946-09c4-4db8-873e-4713f6f06f86	rhel_x86_cloud_init	192.168.21.28

Figure 4-161 Checking where the instance is running

Deploy PowerKVM instances using Cloud Management dashboard

The following steps allow you to deploy instances in PowerKVM Compute nodes using the Cloud Management dashboard:

1. Log in to the Controller node. Select **Project > Compute > Instances**, and then click **Launch Instance**.
2. Under the Details tab, provide the following information:
   - *Availability Zone*. Select the Availability Zone in case you have created `Host Aggregates`.
   - *Instance Name*. Provide a name for your instance (in case you request more than one instance to be deployed at the same time, this will be used as a prefix in the instance names).
   - *Flavor*. Select the flavor that will be used by this instance. The flavor will determine the amount of memory and CPU and the disk size of the new instance.
   - *Instance Count*. Enter the number of instances that you want to deploy at the same time.
   - *Instance Boot Source*. Select **Boot from Image** to use the `qcow2` images that you prepared by using the PowerKVM Compute nodes.
   - *Image Name*. Select the image that will be used to deploy this virtual machine (this image will be copied from the Glance repository to the Compute node and used as the disk for the instance).

Figure 4-162 shows an example of deploying two Red Hat Enterprise Linux 6.5 images in PowerKVM Compute nodes using the `m1.small` flavor.

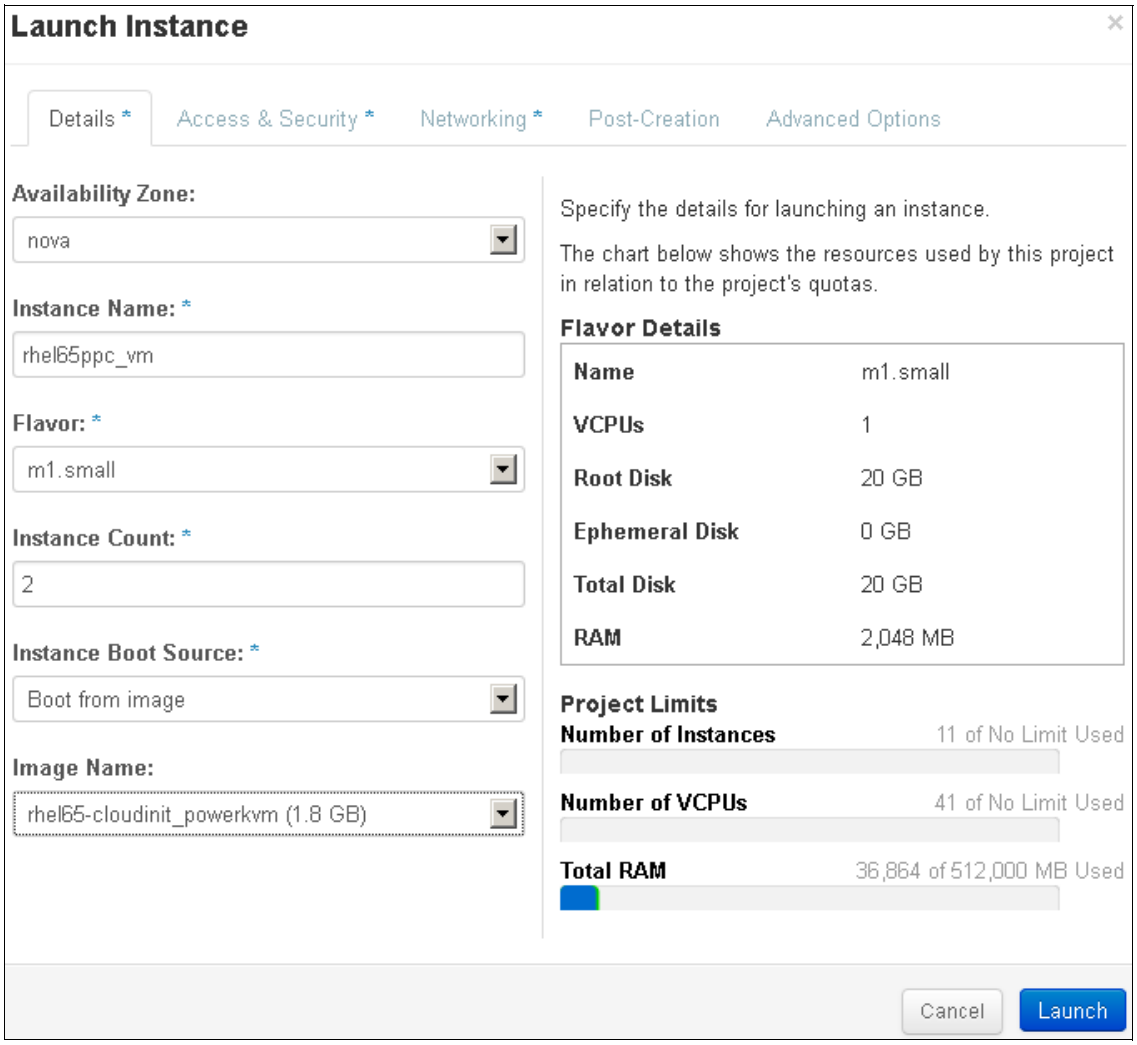


Figure 4-162 Deploying instances in PowerKVM Compute nodes

3. Click the `Access and Security` tab and optionally provide an SSH `key pair` to be injected in the instances after deployment (requires `cloud-init` in the image), so you can log in to the instance without providing a password.

Keep the `default Security Group` enabled (security groups are set of iptables rules applied to an instance network), as shown in Figure 4-163.

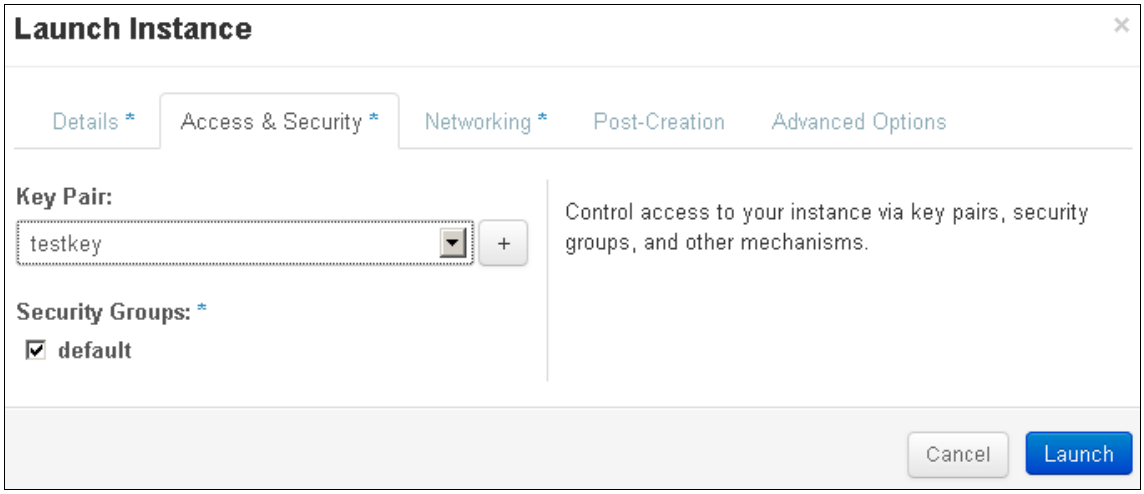


Figure 4-163 Selecting the SSH key pair for injection in the instance

4. Click the **Networking** tab and drag the networks that you want to provide for your instance. Drag the appropriate network from the Available Networks section to the Selected Networks section, as shown in Figure 4-164.

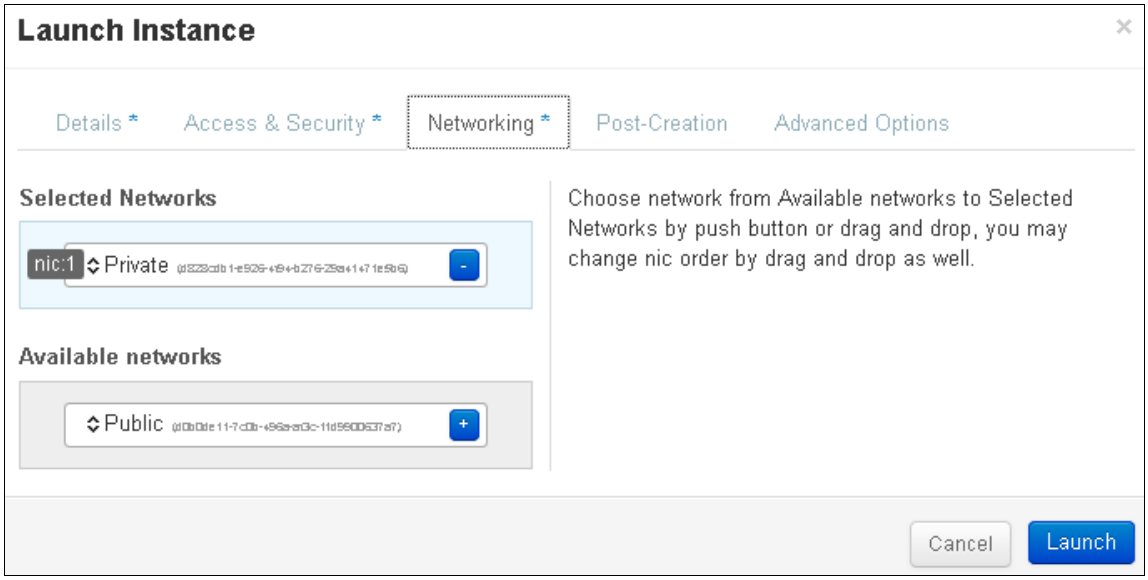


Figure 4-164 Selecting networks for your instances

5. In the `Post-Creation` tab, you can optionally provide a `customization script` to execute in the instance after its deployment. In the `Advanced Options` tab, you can modify the partitioning layout of the disk. These options are not covered in this publication.
6. Click `Launch` to start deploying the instance.

During the deployment, you can see in the Task column that the instance is `Spawning`, as shown in Figure 4-165. This means that the image disk is being copied to the hypervisor where the instance will run.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task
rhel65ppc_vm-7b1e7e3c-c516-495d-87f0-dbe97c68f0fe	rhel65-cloudinit_powerkvm		m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk	testkey	Build	nova	Spawning
rhel65ppc_vm-6b6288cc-4109-4607-826c-9adac803e85a	rhel65-cloudinit_powerkvm		m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk	testkey	Build	nova	Spawning

Figure 4-165 Instances Spawning during deployment

7. When the deployment is completed, you will be able to see the instance IP address from the Cloud Management dashboard and it will be available on the network, as shown in Figure 4-166.

Instance Name	Image Name	IP Address	Size
rhel65ppc_vm-7b1e7e3c-c516-495d-87f0-dbe97c68f0fe	rhel65-cloudinit_powerkvm	192.168.21.30	m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk
rhel65ppc_vm-6b6288cc-4109-4607-826c-9adac803e85a	rhel65-cloudinit_powerkvm	192.168.21.29	m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk

Figure 4-166 Instance IP address after deployment in PowerKVM Compute nodes

You may notice that you have not selected the PowerKVM Compute node where the instance is going to run. This was automatically selected by `EGO` according to the policy configured in Platform Resource Scheduler and the current workload in the PowerKVM Compute nodes. In this case, only PowerKVM Compute nodes are considered, based on the image properties (Properties `architecture` and `hypervisor_type` of the Glance image), as shown in Example 4-191.

Example 4-191 Verifying the Glance image properties

```
[root@cmocontroller ~]# glance image-show 9af3d32a-80c7-4865-8173-4756cd70e5b6
+---------------------------+--------------------------------------+
| Property                  | Value                                |
+---------------------------+--------------------------------------+
| Property 'architecture'   | ppc64                                |
| Property 'hypervisor_type'| qemu                                 |
| checksum                  | f04f4dd0c07afa18ee1a71be65e26585     |
| container_format          | bare                                 |
| created_at                | 2014-08-05T14:47:30.928399           |
| deleted                   | False                                |
| disk_format               | qcow2                                |
| id                        | 9af3d32a-80c7-4865-8173-4756cd70e5b6 |
| is_public                 | True                                 |
| min_disk                  | 0                                    |
```

```
| min_ram                    | 0                                    |
| name                       | rhel65-cloudinit_powerkvm            |
| owner                      | 64de5a3d843c4e9d8173b1f0c2d54f59     |
| protected                  | False                                |
| size                       | 1893203968                           |
| status                     | active                               |
| updated_at                 | 2014-08-05T14:49:00.926767           |
+----------------------------+--------------------------------------+
```

You can see where the instance was deployed in the Host column by clicking **Admin > System Panel > Instances**, as shown in Figure 4-167.

Project	Host	Name	Image Name	IP Address
Public	powerkvm2.itso.ibm.com	rhel65ppc_vm-7b1e7e3c-c516-495d-87f0-dbe97c68f0fe	rhel65-cloudinit_powerkvm	192.168.21.30
Public	powerkvm1.itso.ibm.com	rhel65ppc_vm-6b6288cc-4109-4607-826c-9adac803e85a	rhel65-cloudinit_powerkvm	192.168.21.29

Figure 4-167 Checking the hypervisor where the instance was deployed

Deploy PowerVM instances using Cloud Management dashboard

The following steps allow you to deploy images in PowerVM Compute nodes using the Cloud Management dashboard. This action requires an image previously prepared and captured using the instructions from 4.12.2, “Manage images in PowerVM Compute nodes” on page 431:

1. Log in to the Controller node and click **Project > Compute > Instances** and click **Launch Instance**.
2. In the `Details` tab, provide the following information:
   - *Availability Zone*. Select the Availability Zone in case you have created Host Aggregates.
   - *Instance Name*. Provide a name for your instance (in case you request more than one instance to be deployed at the same time, this will be used as a prefix in the instance names).
   - *Flavor*. Select the flavor that will be used by this instance. The flavor will determine the amount of memory and CPU and the disk size of the new instance. For PowerVM Compute node, you must select a PowerVC flavor. The PowerVC flavors are automatically synchronized between the PowerVC server and the Controller node through the PowerVC driver. The PowerVC flavors usually start with the *PVC* prefix.
   - *Instance Count*. Indicate how many instances you want to deploy at the same time.
   - *Instance Boot Source*. Select **Boot from snapsho**t to use the snapshot images prepared using the PowerVC.
   - *Image Name*. Select the image that will be used to deploy this virtual machine (a new snapshot will be created for the instance, based on the image volume).

Figure 4-168 shows an example of deploying two AIX 6.1 instances in the PowerVM Compute nodes using the PVC-m1.small flavor.

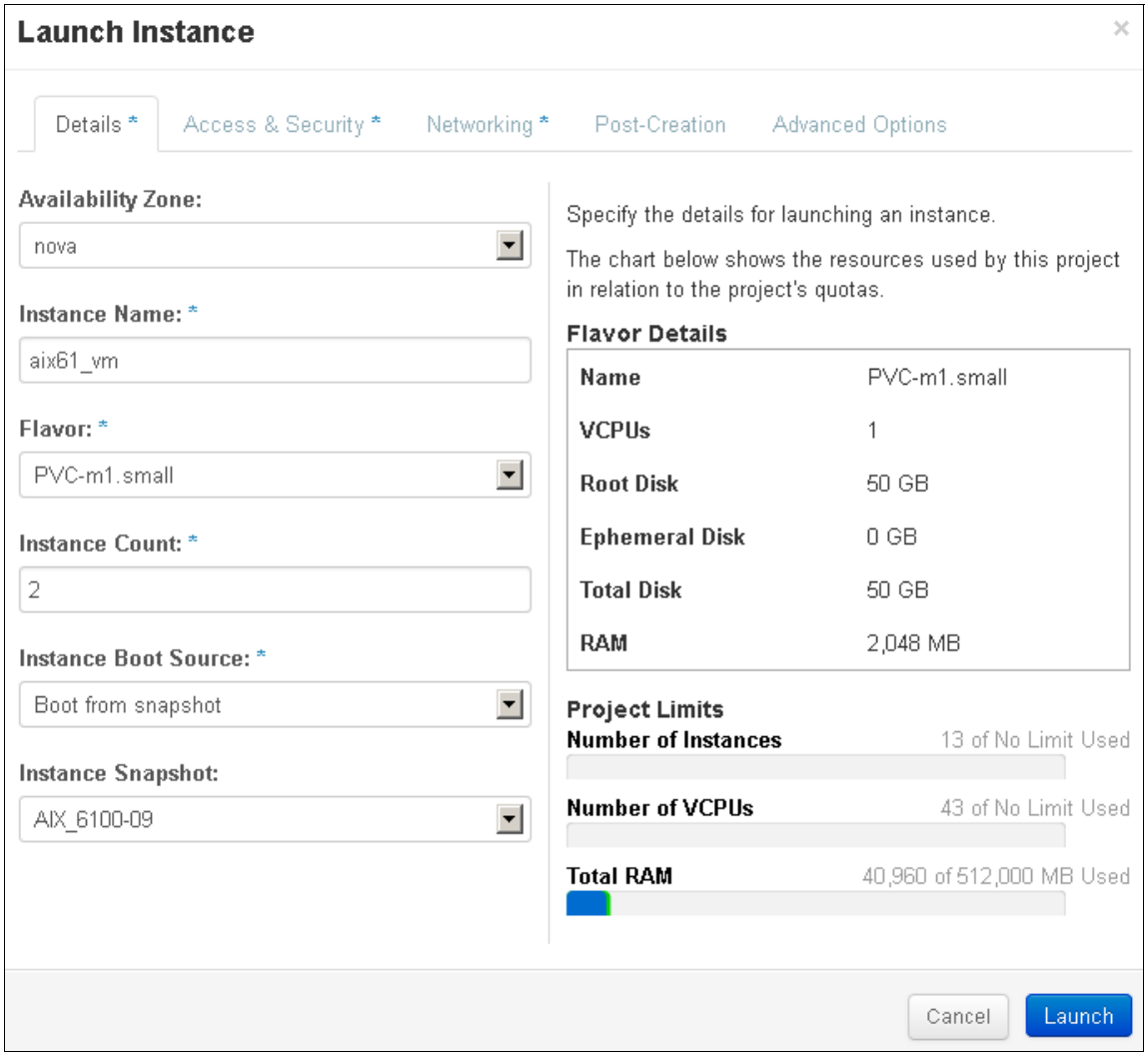


Figure 4-168 Deploying AIX instances in PowerVM Compute nodes

3. Click the **Access and Security** tab and *do not* select an SSH key pair (requires `cloud-init` in the image, which is not valid for AIX images, so only use the key pair if you have a Linux image prepared with cloud-init in the PowerVC server).

Keep the default Security Group enabled (security groups are set of `iptables` rules applied to an instance network), as shown in Figure 4-169.

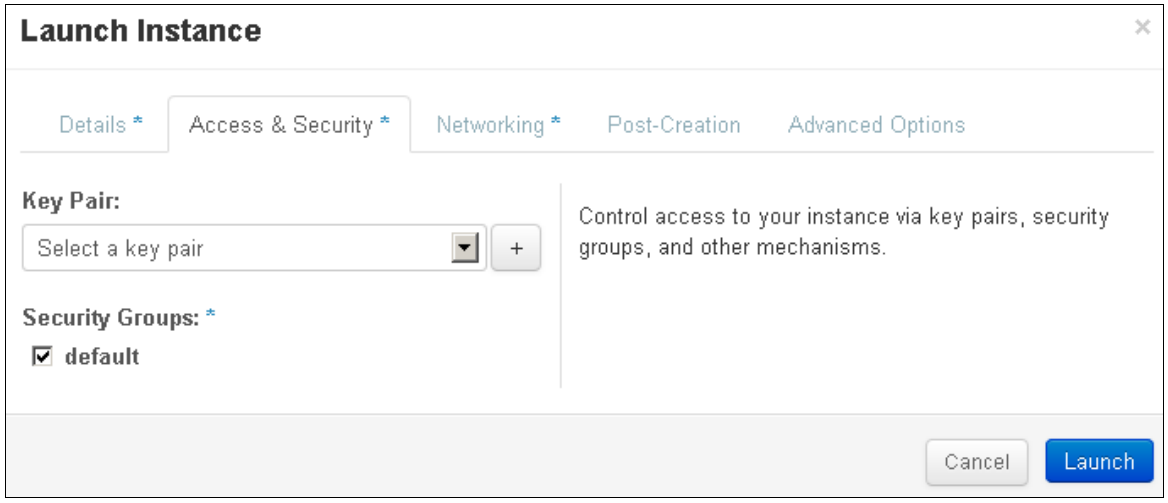


Figure 4-169 Access and Security tab deploying instances in PowerVM node

4. Click the **Networking** tab and drag the networks that you will to provide for your instance. Drag the appropriate network from the Available Networks section to the Selected Networks section, as shown in Figure 4-170.

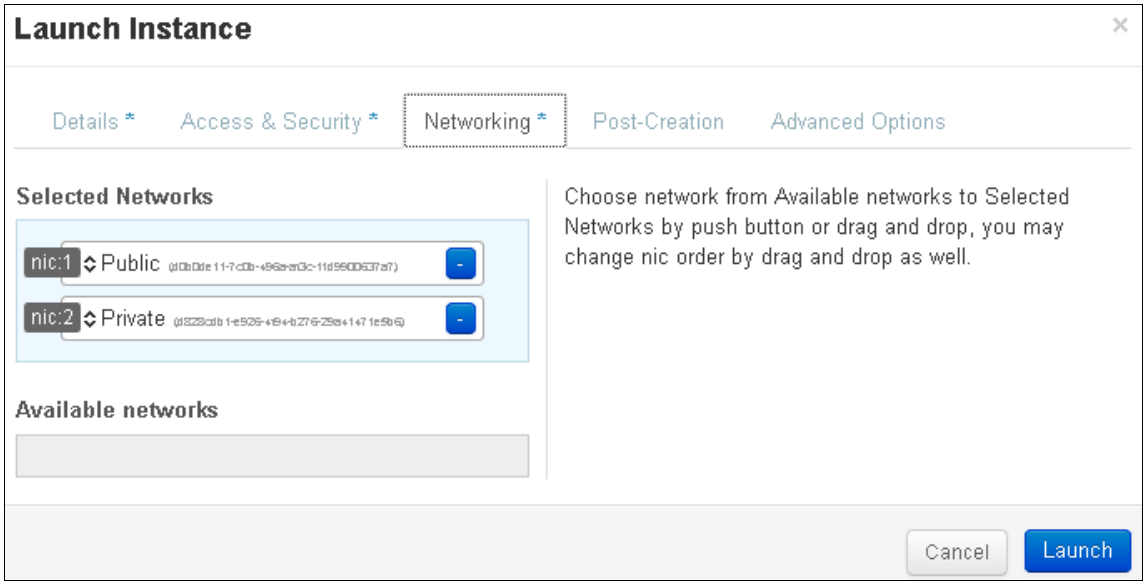


Figure 4-170 Providing networks for instances deployed in PowerVM Compute nodes

5. In the Post-Creation tab, you can optionally provide a customization script to execute in the instance after its deployment (only valid for Linux images with `cloud-init`). In the Advanced Options tab, you can modify the partitioning layout of the disk (only valid for Linux images with `cloud-init`). These options are not covered in this publication.

6. Click **Launch** to start deploying the instance. During the deployment, you can see in the Task column that the instance is s*pawning*, as shown in Figure 4-171. This means that the image is being deployed in the PowerVC server.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task
aix61_vm-38b1ac04-5c4b-4939-8070-6d1a7ce3c8ae	AIX_6100-09		PVC-m1.small | 2GB RAM | 1 VCPU | 50.0GB Disk	-	Build	nova	Spawning
aix61_vm-19d60eb5-3736-4893-ab27-f31bbbf4b5e6	AIX_6100-09		PVC-m1.small | 2GB RAM | 1 VCPU | 50.0GB Disk	-	Build	nova	Spawning

Figure 4-171 Instances Spawning while being deployed in the PowerVM Compute nodes

Important: You will notice that the instances deployed in PowerVM Compute nodes will be in the spawning task for longer than the KVM or PowerKVM Compute nodes. This is because more tasks are involved in this deployment (including creating zones in the SAN switch, hosts in the SVC and V7000, volumes and mapping them accordingly), and the PowerVC server (and, consequently, the Cloud Manager) wait for communication to be established between the instance and HMC using RMC connection.

7. If you log in to the PowerVC server and click virtual machines during this step, you will notice the same instances being deployed, as shown in Figure 4-172.

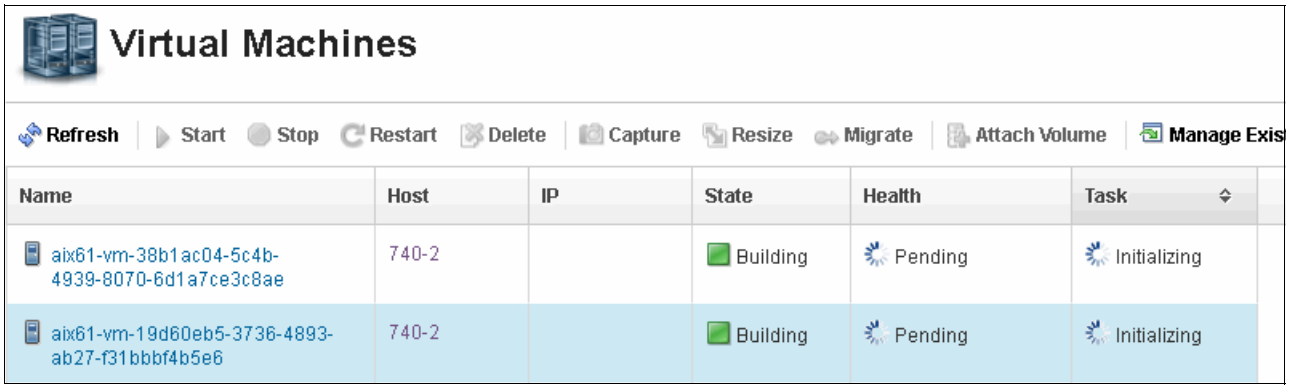


Figure 4-172 Instances being deployed in the PowerVC server

8. When the deployment is complete, you will be able to see the instance addresses from the Cloud Management dashboard, as shown in Figure 4-173.

Instance Name	Image Name	IP Address	Size
aix61_vm-38b1ac04-5c4b-4939-8070-6d1a7ce3c8ae	AIX_6100-09	**Public** 172.16.20.172 **Private** 192.168.21.32	PVC-m1.small | 2GB RAM | 1 VCPU | 50.0GB Disk
aix61_vm-19d60eb5-3736-4893-ab27-f31bbbf4b5e6	AIX_6100-09	**Public** 172.16.20.171 **Private** 192.168.21.31	PVC-m1.small | 2GB RAM | 1 VCPU | 50.0GB Disk

Figure 4-173 Instances deployed in PowerVM Compute nodes - Cloud Management dashboard

9. You will also be able to see the instance in the PowerVC server by clicking **Virtual Machines**, as shown in Figure 4-174.

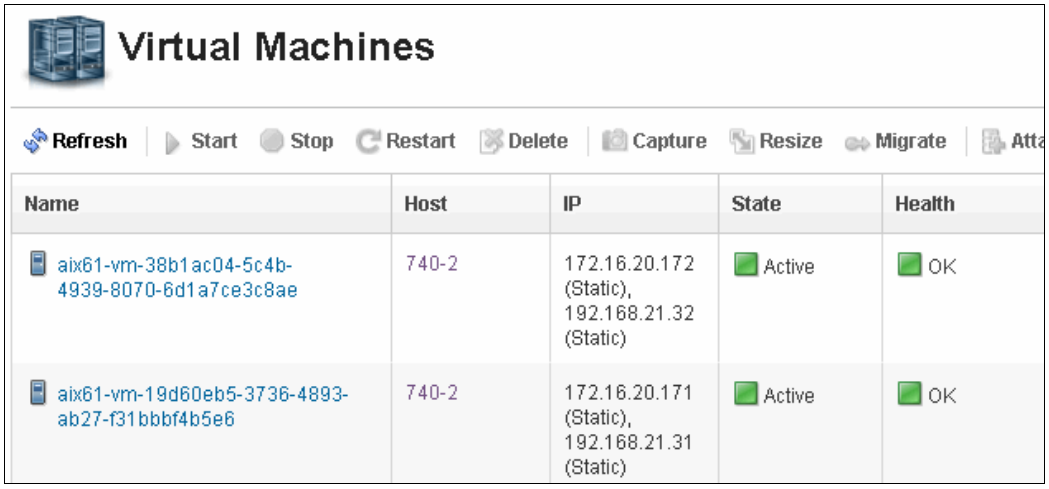


Figure 4-174 Instances Deployed in PowerVM Compute nodes (PowerVC screen capture)

You might notice that you have not selected the PowerVM Compute node where the instance is going to run. In this case, EGO has detected that the only hypervisor capable of hosting the image is the PowerVM Compute nodes (given that this is a snapshot image imported from PowerVC server), so it deploys the image using the PowerVC Driver. The placement policy configured in the PowerVC server is considered to determine which PowerVM Compute node will receive the instance.

You can see which placement policy is configured in PowerVC server, log in to the PowerVC interface and click **Configuration** at the top of the display, as shown in Figure 4-175.

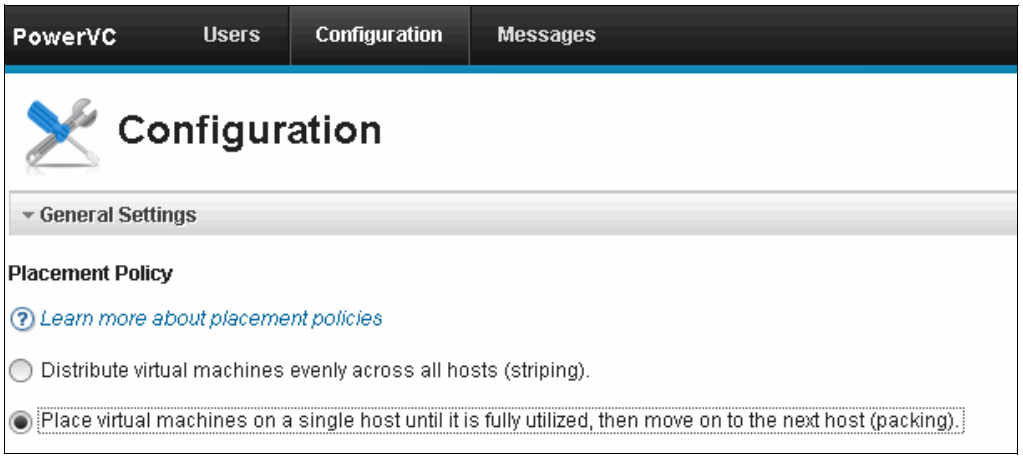


Figure 4-175 Selecting the placement policy in PowerVC

Stopp, restart, and start instances

To stop (shut down) an instance, log in to the Cloud Management dashboard, click **Project > Compute > Instances**, select the instance that you want to shutdown and click **More > Shut Off Instance.** You may also want to use **Soft Reboot Instance** or **Hard Reboot Instance** to restart the instance rather than shutting down, as shown in Figure 4-176.

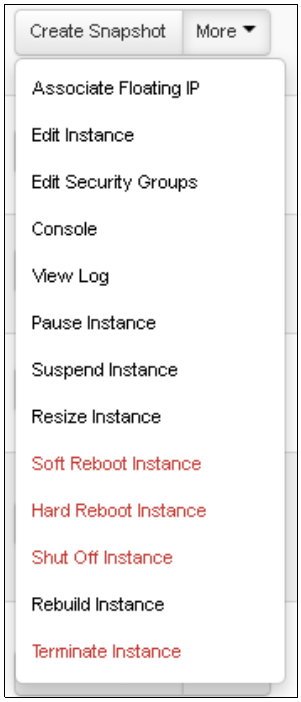


Figure 4-176 Shutting Down or Restarting an instance

If the instance is shut down and you want to start the instance (power on the instance), you can select it and click **Start Instance** (click **More**), as shown in Figure 4-177.

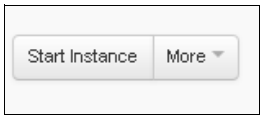


Figure 4-177 Starting an instance in the Cloud Management dashboard

Delete an instance

To delete an instance, log in to the Cloud Management dashboard and click **Project > Compute > Instances**. Select the instance that you want to delete and, in the upper-right corner of the screen, click **Terminate Instances**. Then, confirm the deletion, as shown in Figure 4-178.



Figure 4-178 Terminating instances in Cloud Management dashboard

The instance is immediately deleted from KVM and PowerKVM Compute nodes. In PowerVM Compute nodes, the request for deletion is forwarded to the PowerVC server, as shown in Figure 4-179.

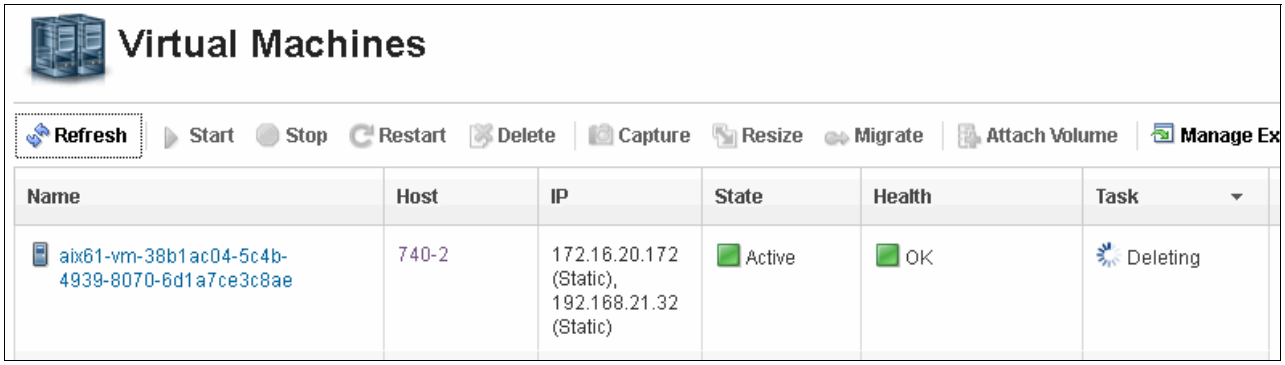


Figure 4-179 Image being deleted from PowerVC

4.13.2 Manage instances using self-service portal

This section explains how to perform basic instance management operations by using the self-service portal, which is an additional management interface that based on IBM Cloud Manager that you can optionally enable in your Cloud Manager. Follow the instructions from section 4.9, “Enable the self-service portal” on page 311. This interface can be accessed using the Controller node fully qualified domain name or IP address:

```
http://<controller_node_fqdn>:18080/cloud/web/login.html
```

Deploy instances

You can use the self-service portal to deploy instances on your cloud, which means creating virtual machines based on the images you have previously defined using instructions from section 4.12, “Manage images in IBM Cloud Manager” on page 412.

Use the following instructions to deploy instances using the self-service portal:

1. Log in to the self-service portal and click **Images**. Click the name of the image that you want to deploy, and then click **More > Advanced deploy**, as shown in Figure 4-180.

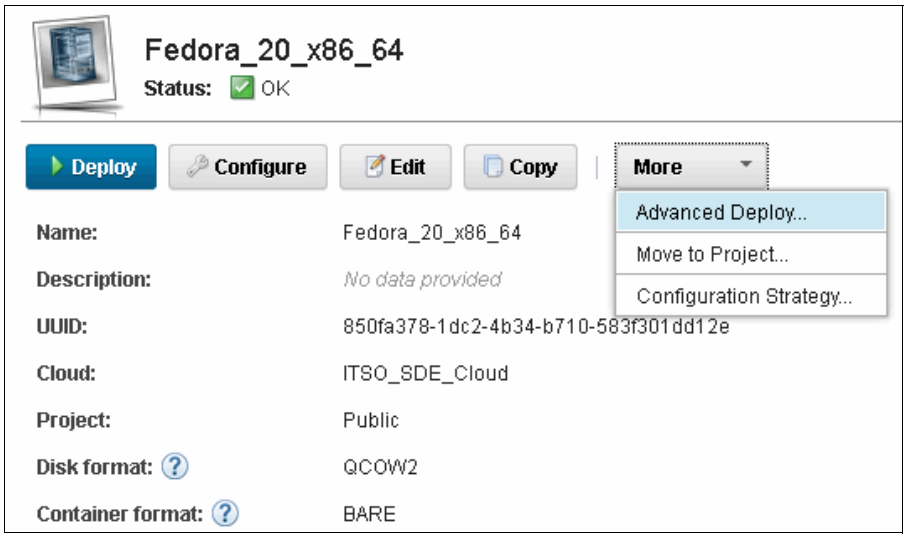


Figure 4-180 Initiating an instance deploy in KVM Compute node from self-service portal

2. Provide a name and (option) description for the instance, select the project where the instance will be deployed (in case you have more than one project or tenant), and provide the number of instances that you want to deploy at the same time based on this image.

If you have more than one OpenStack cloud connected to your self-service portal, you must select in which cloud (Deployment Target) you want to deploy your instance, as shown in Figure 4-181.

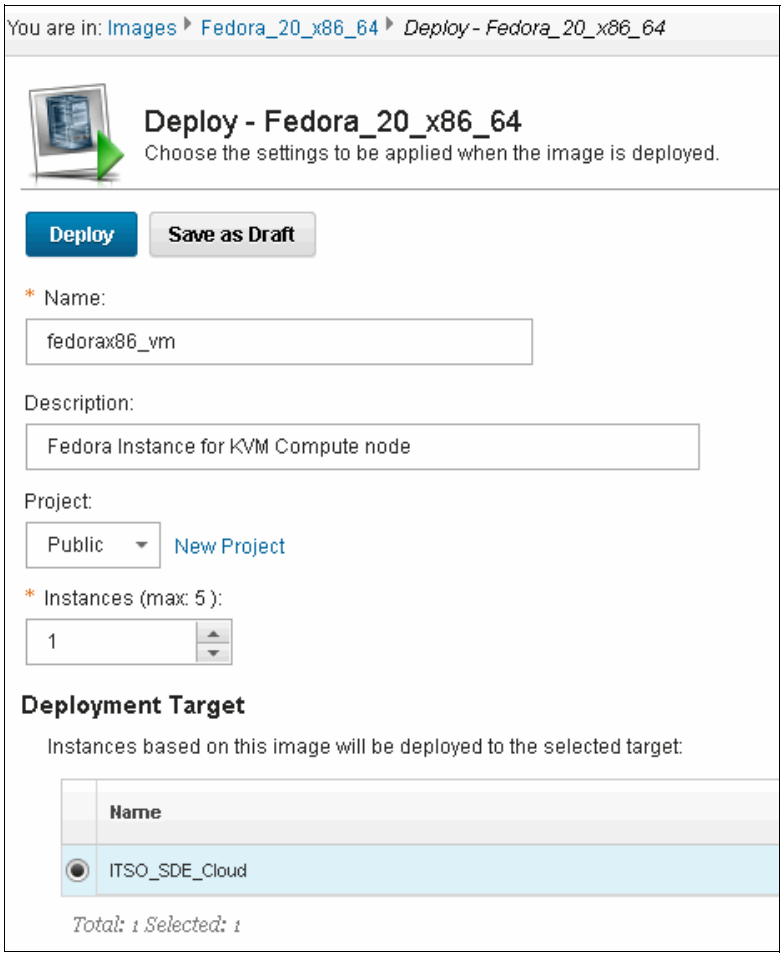


Figure 4-181 Initial customization of the instance for deployment

3. Scroll down and, in the Hardware section, select the flavor that will be used in the instance deployment, as shown in Figure 4-182 on page 463.

Important: If you are deploying an image from PowerVC, you must select a PowerVC flavor. The PowerVC flavors are automatically synchronized between PowerVC server and the Controller node through the PowerVC driver. The PowerVC flavors usually start with the `PVC` prefix.

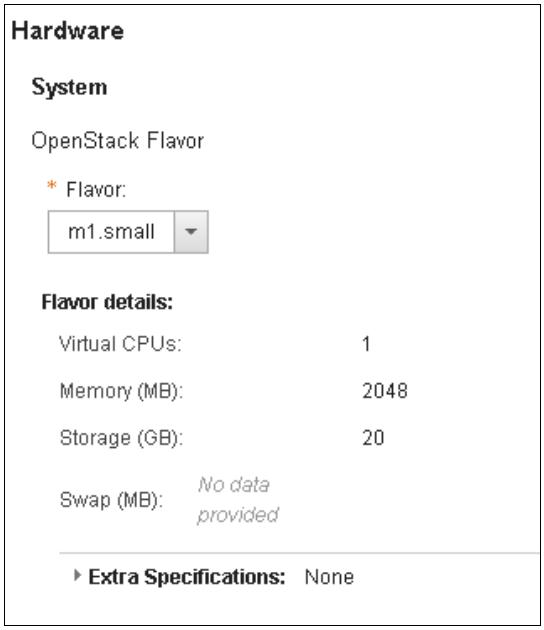


Figure 4-182 Selecting the flavor for the instance in self-service portal

4. Scroll down and select the **Enable config drive** option if you want to use SSH key pair injection. Select which SSH key pair to use in this instance and, optionally, provide Customization scripts (to execute in the instance after its deployment) and personality files (files to create in the instance after its deployment). Config drive, SSH key pair injection, customization scripts, and personality files require an image created with cloud-init (not valid for AIX images). Use of customization scripts and personality files is not covered in this publication. Figure 4-183 shows the use of the SSH key pair in the self-service portal during instance deployment.

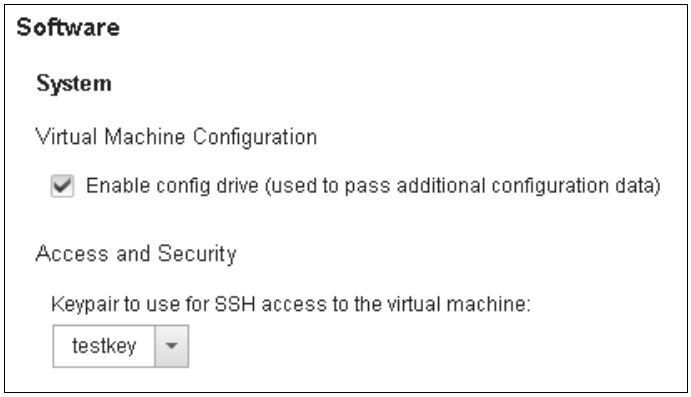


Figure 4-183 Selecting the SSH key pair to inject in the instance during deployment

5. In the Network section, select the network that you want to provide for the instance. In case you want to provide more than one network, click **Add a network**, and select the second network.

Click **Deploy** to start deploying the instance, as shown in Figure 4-184.

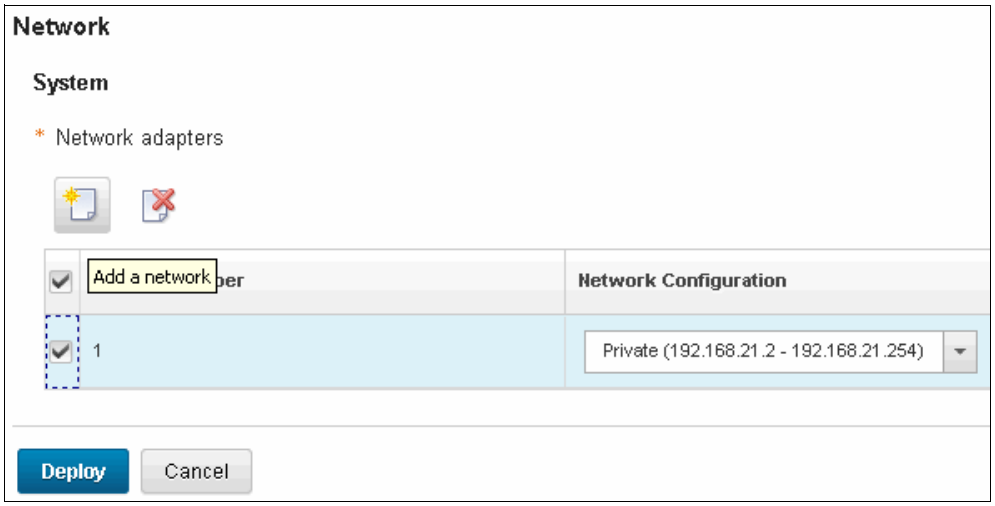


Figure 4-184 Selecting the networks for the instance

6. The Status column will show that the image is deploying, as you can see in Figure 4-185.

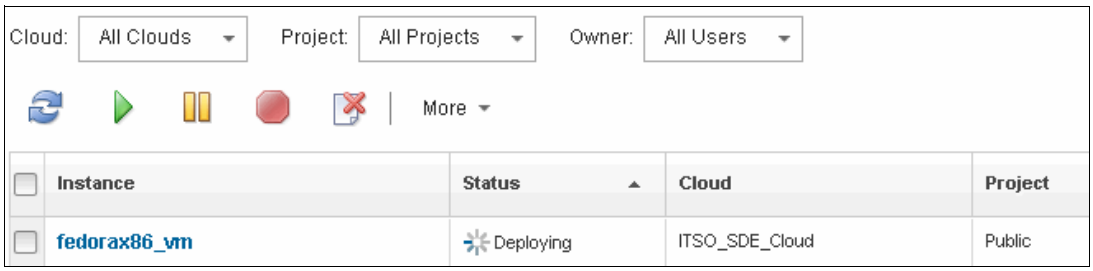


Figure 4-185 Instance deploying in the self-service portal

7. Click **Refresh** to update the instance status, as shown in Figure 4-186.

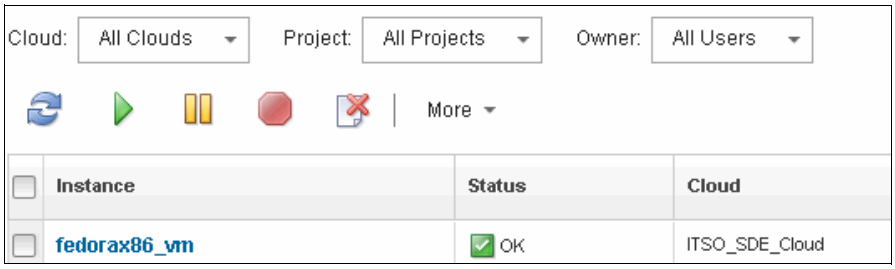


Figure 4-186 Instance deployed in self-service portal

8. Click the instance to view its information, including its name on the hypervisor where it is running, the hypervisor type and its IP address. Expand the virtual machine Properties to obtain detailed information, including exactly which Compute node the instance is running.

Figure 4-187 shows the instance that was just deployed using the self-service portal.

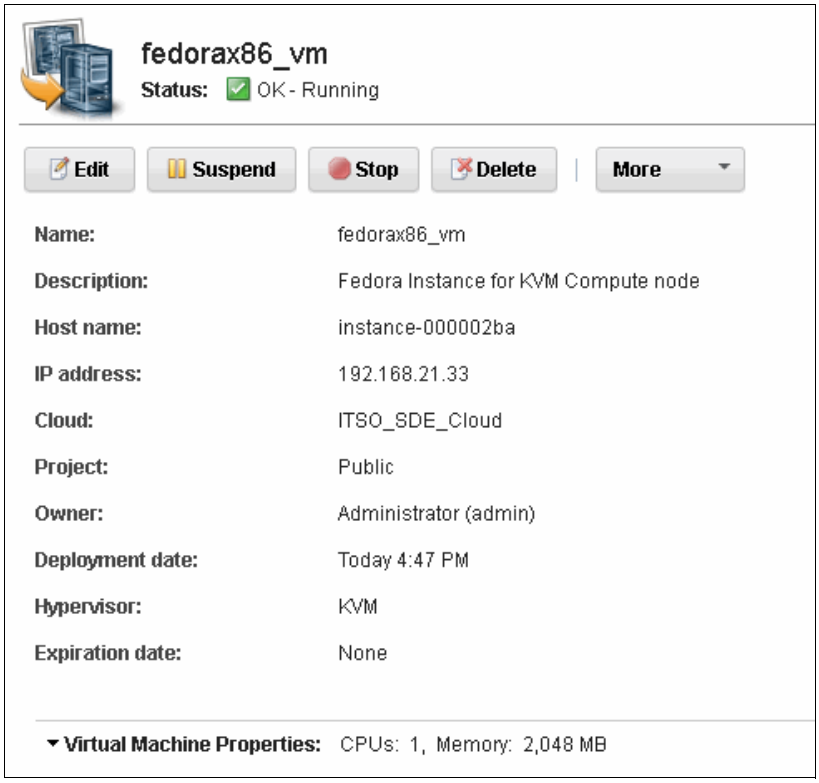


Figure 4-187 Virtual machine in the self-service portal

You will also be able to see the same instance in the Cloud Management dashboard, after its deployment (and in PowerVC in case it is a PowerVC image).

Stop and start instances

To stop (shutdown) an instance, log in to the self-service portal, click Instances and click the instance name which you want to shut down, and then click `Stop`, as shown in Figure 4-188 on page 466.

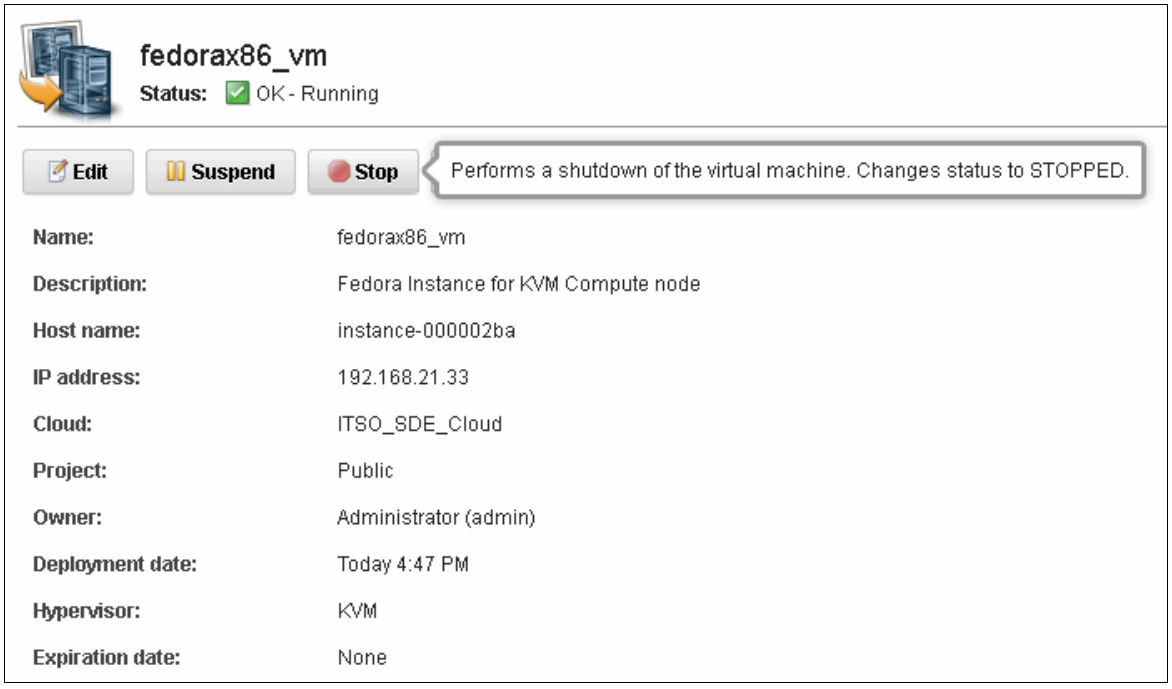


Figure 4-188 Stopping an instance using the self-service portal

Click the Refresh icon in the upper-right corner of the screen to refresh the status of the instance.

If the instance is stopped, you will see a Start button, as shown in Figure 4-189. Click this button to start the instance.

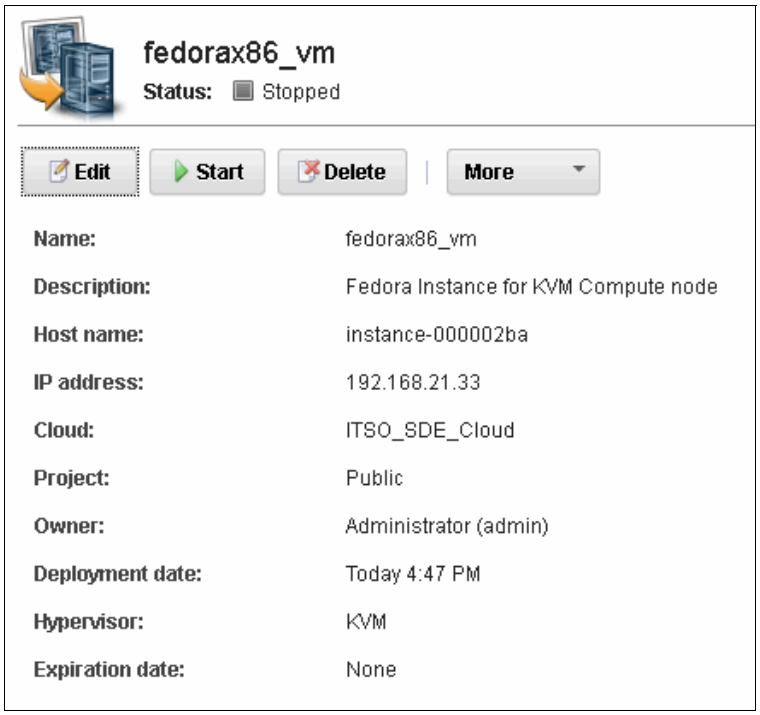


Figure 4-189 Starting the instance using self-service portal

Delete an instance

To delete an instance using self-service portal, log in to the portal and click instances, and then click the name of the instance that you want to delete and click the Delete button, as shown in Figure 4-190.

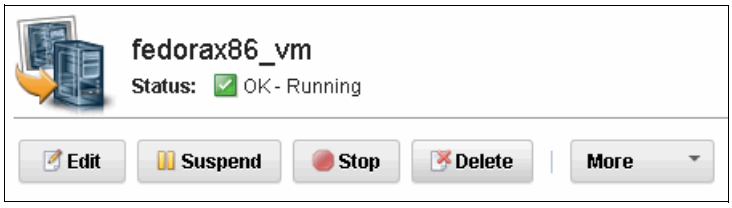


Figure 4-190 Deleting an instance using self-service portal

Click Delete to confirm the deletion, as shown in Figure 4-191.

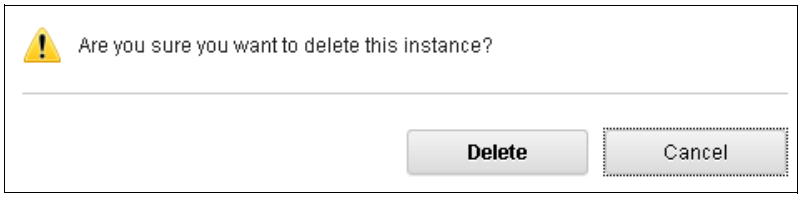


Figure 4-191 Confirming the instance deletion

4.13.3 Manage instances using the command-line interface

This section allows you to perform basic instance operations using the command-line interface (CLI). To use the command-line interface, log in to the Controller node using SSH. Most of the instance management activity can be accomplished using the **nova** command.

You can obtain more information about the **nova** command and its syntax using the following commands:

```
# man nova
# nova --help
```

Deploy instances

You can use the command-line interface to deploy instances in your cloud, which means creating virtual machines based on the images you have previously prepared and defined in section 4.12, “Manage images in IBM Cloud Manager” on page 412. To deploy instances, you need to use the **nova boot** subcommand:

```
# nova help boot
usage: nova boot [--flavor <flavor>] [--image <image>]
                 [--image-with <key=value>] [--boot-volume <volume_id>]
                 [--snapshot <snapshot_id>] [--num-instances <number>]
                 [--meta <key=value>] [--file <dst-path=src-path>]
                 [--key-name <key-name>] [--user-data <user-data>]
                 [--availability-zone <availability-zone>]
                 [--security-groups <security-groups>]
                 [--block-device-mapping <dev-name=mapping>]
                 [--block-device key1=value1[,key2=value2...]]
                 [--swap <swap_size>]
                 [--ephemeral size=<size>[,format=<format>]]
                 [--hint <key=value>]
```

```
[--nic <net-id=net-uuid,v4-fixed-ip=ip-addr,port-id=port-uuid>]
[--config-drive <value>] [--poll]
<name>
```

To get more information about a subcommand, you can use the **nova help <subcommand>**:

```
# nova help boot
```

Note: You will also need to use the **neutron** and **glance** commands to obtain, respectively, the network ID and the image name.

1. Log in to the Controller node and run the **source** command to read the variables from /root/openrc file:

   ```
   # source /root/openrc
   ```

2. Run the **glance image-list** command to show the available images and note which image you plan to use in this deployment, as shown in Example 4-192.

Example 4-192 Obtaining the image to use in the instance deployment

```
[root@cmocontroller ~]# glance image-list
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
| ID                                   | Name                            | Disk Format | Container Format | Size        | Status |
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
| b3064622-6215-4c1b-b2a7-fdbb3e48e9fa | AIX_6100-09                     | raw         | bare             | 53687091200 | active |
| 850fa378-1dc2-4b34-b710-583f301dd12e | Fedora_20_x86_64                | qcow2       | bare             | 210829312   | active |
| 4e06e2c7-389c-4751-93a8-1e47d60b0c5b | ITSO_SDE_AIX_Template_capture_1 | raw         | bare             | 53687091200 | active |
| 63e08c8c-9334-4215-879e-c40e0f681406 | ITSO_SDE_capture_1              | raw         | bare             | 53687091200 | active |
| 354a2183-cdb7-45f6-bfd0-4ce3a4c1025c | PowerVC Default Image           | raw         | ovf              |             | queued |
| c2e714ca-37e9-4a99-8ad7-d7bf10cad4fb | RHEL65                          | raw         | bare             | 53687091200 | active |
| df5f6a15-8c21-413c-a577-7f020bc96e2b | RHEL65_x86_64                   | qcow2       | bare             | 338427904   | active |
| 88d74f68-a0b5-499e-bc31-ba39d5aa839b | cirros                          | qcow2       | bare             | 9761280     | active |
| 9af3d32a-80c7-4865-8173-4756cd70e5b6 | rhel65-cloudinit_powerkvm       | qcow2       | bare             | 1893203968  | active |
| 094ce631-2814-42a5-9a78-f202ab7690ca | rhel65ppc64_powerkvm            | qcow2       | bare             | 10737418240 | active |
| f98aead8-ebbc-4cea-901c-a72b97366487 | rhel_x86_cloud_init             | qcow2       | bare             | 6552420352  | active |
| 4daa0c00-3bf9-42aa-85ba-d5e67ba70cce | ubuntu_x86_64                   | qcow2       | bare             | 255394304   | active |
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
```

3. Use the **nova flavor-list** command to obtain the available flavors, so you can choose which you want to use during the deployment, as shown in Example 4-193.

Example 4-193 Obtaining the flavor to use during instance deployment

```
[root@cmocontroller ~]# nova flavor-list
+--------------------------------------+---------------+-----------+------+-----------+------+-------+-------------+-----------+
| ID                                   | Name          | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+--------------------------------------+---------------+-----------+------+-----------+------+-------+-------------+-----------+
| 1                                    | m1.tiny       | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 1f6596de-2708-42b2-81fc-f324fbb3cde6 | PVC-m1.small  | 2048      | 50   | 0         |      | 1     | 1.0         | True      |
| 2                                    | m1.small      | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3                                    | m1.medium     | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4                                    | m1.large      | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5                                    | m1.xlarge     | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
| 714ebbee-c0ed-4a55-9bcd-8a2a7bea944c | PVC-m1.large  | 8192      | 50   | 0         |      | 4     | 1.0         | True      |
| d9599491-cc9c-435f-a363-852f1c1e6c94 | PVC-m1.medium | 4096      | 50   | 0         |      | 2     | 1.0         | True      |
| e21d6835-cd1d-4fc1-8922-b441be11bf39 | PVC-m1.tiny   | 1024      | 50   | 0         |      | 1     | 1.0         | True      |
| e7c1f9e3-a725-460a-9fe5-d65b084030f5 | PVC-m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
+--------------------------------------+---------------+-----------+------+-----------+------+-------+-------------+-----------+
```

Note: If you are deploying an instance in PowerVM Compute node, you must use a PowerVC flavor, which usually starts with PVC prefix.

4. Run the **nova keypair-list** command to obtain the available list of SSH key pair to inject in the instance during deployment (requires an image prepared with cloud-init), as shown in Example 4-194.

Example 4-194 Listing the Available SSH key pairs

```
[root@cmocontroller ~]# nova keypair-list
+---------+-------------------------------------------------+
| Name    | Fingerprint                                     |
+---------+-------------------------------------------------+
| fabio   | d2:5a:41:7b:6b:2e:c2:76:6f:01:39:34:aa:11:23:2f |
| seckey  | 29:51:4a:f8:a9:f8:2a:12:cd:0f:89:cc:4f:18:89:96 |
| testkey | a9:76:37:58:b1:97:f1:67:3a:1b:8d:57:29:63:11:10 |
+---------+-------------------------------------------------+
```

5. Run the **neutron net-list** command to list the available networks, so you can choose which network you want to provide for the instance, as shown in Example 4-195.

Example 4-195 Listing the available networks

```
[root@cmocontroller ~]# neutron net-list
+--------------------------------------+---------+-------------------------------------------------------+
| id                                   | name    | subnets                                               |
+--------------------------------------+---------+-------------------------------------------------------+
| d0b0de11-7c0b-496a-af3c-11d9900637a7 | Public  | a31a550c-d26e-4283-a908-fc0c388e845e 172.16.20.0/22   |
| d828cdb1-e926-4f94-b276-29a41471e5b6 | Private | dc3d4c88-e524-41d3-987a-32da93cca88a 192.168.21.0/24  |
+--------------------------------------+---------+-------------------------------------------------------+
```

Note: You can export the EDITOR variable to avoid the following messages when running **neutron** commands:

```
which: no gedit in
(/usr/lib64/qt-3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/
usr/bin:/root/bin:/usr/lpp/mmfs/bin)
which: no kate in
(/usr/lib64/qt-3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/
usr/bin:/root/bin:/usr/lpp/mmfs/bin)
```

Export the EDITOR variable to use vi to edit the output:

```
# export EDITOR=vi
```

6. Run the **nova boot** command to deploy the instance by using the information that you have obtained in the previous steps (image name, flavor, SSH key pair, and network ID), as shown in Example 4-196.

Example 4-196 Deploying an instance using command-line interface

```
[root@cmocontroller ~]# nova boot --image ubuntu_x86_64 --flavor m1.medium --key-name testkey
--nic net-id=d828cdb1-e926-4f94-b276-29a41471e5b6 ubuntux86vm --config-drive True
+--------------------------------------+-------------------------------------------------------+
| Property                             | Value                                                 |
+--------------------------------------+-------------------------------------------------------+
| IBM-PVM:health_status                | {}                                                    |
| OS-DCF:diskConfig                    | MANUAL                                                |
| OS-EXT-AZ:availability_zone          | nova                                                  |
| OS-EXT-SRV-ATTR:host                 | -                                                     |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                                     |
```

```
| OS-EXT-SRV-ATTR:instance_name        | instance-000002bb                                    |
| OS-EXT-STS:power_state               | 0                                                    |
| OS-EXT-STS:task_state                | scheduling                                           |
| OS-EXT-STS:vm_state                  | building                                             |
| OS-SRV-USG:launched_at               | -                                                    |
| OS-SRV-USG:terminated_at             | -                                                    |
| accessIPv4                           |                                                      |
| accessIPv6                           |                                                      |
| adminPass                            | qymK3pE79Tsz                                         |
| config_drive                         | True                                                 |
| created                              | 2014-09-14T14:30:52Z                                 |
| flavor                               | m1.medium (3)                                        |
| hostId                               |                                                      |
| id                                   | 017fae17-22d0-4ccd-a533-e94c26493cf4                 |
| image                                | ubuntu_x86_64 (4daa0c00-3bf9-42aa-85ba-d5e67ba70cce) |
| key_name                             | testkey                                              |
| metadata                             | {}                                                   |
| name                                 | ubuntux86vm                                          |
| os-extended-volumes:volumes_attached | []                                                   |
| progress                             | 0                                                    |
| security_groups                      | default                                              |
| status                               | BUILD                                                |
| tenant_id                            | 64de5a3d843c4e9d8173b1f0c2d54f59                     |
| updated                              | 2014-09-14T14:30:52Z                                 |
| user_id                              | 13f4fcb07a5f447e9b3fb5555a0eeabc                     |
+--------------------------------------+------------------------------------------------------+
```

In this example, the following parameters were provided during the deployment:

`--image ubuntu_x86_64`

Use the `ubuntu_x86_64` image for this instance.

`--flavor m1.medium`

Use the `m1.medium` flavor (4096 MB of RAM, 40 GB of disk storage, and 2 virtual CPUs).

`--key-name testkey`

Use the `testkey` SSH key pair to inject in the instance.

`--nic net-id=d828cdb1-e926-4f94-b276-29a41471e5b6`

Use the `private network` in this instance (obtained by using the **`neutron net-list`** command).

`--config-drive True`

Enable the config drive to allow SSH key pair injection in the instance.

7. After a few seconds (or minutes, depending on the image size), use the **`nova show`** command to check the instance information, including its IP address and the Compute node where it was deployed, See Example 4-197.

Example 4-197 Obtaining instance information using the nova show command

```
[root@cmocontroller ~]# nova show ubuntux86vm
+--------------------------------------+------------------------------------------------------+
| Property                             | Value                                                |
+--------------------------------------+------------------------------------------------------+
| IBM-PVM:health_status                | {}                                                   |
| OS-DCF:diskConfig                    | MANUAL                                               |
```

```
| OS-EXT-AZ:availability_zone          | nova                                                             |
| OS-EXT-SRV-ATTR:host                 | cmo2.itso.ibm.com                                                |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | cmo2.itso.ibm.com                                                |
| OS-EXT-SRV-ATTR:instance_name        | instance-000002bb                                                |
| OS-EXT-STS:power_state               | 1                                                                |
| OS-EXT-STS:task_state                | -                                                                |
| OS-EXT-STS:vm_state                  | active                                                           |
| OS-SRV-USG:launched_at               | 2014-09-14T14:30:34.000000                                       |
| OS-SRV-USG:terminated_at             | -                                                                |
| Private network                      | 192.168.21.34                                                    |
| accessIPv4                           |                                                                  |
| accessIPv6                           |                                                                  |
| config_drive                         | True                                                             |
| created                              | 2014-09-14T14:30:52Z                                             |
| flavor                               | m1.medium (3)                                                    |
| hostId                               | 1cae6e996f96b1b60a2f57e188b1b398e756e24e3e2bd8feda45f3ad          |
| id                                   | 017fae17-22d0-4ccd-a533-e94c26493cf4                             |
| image                                | ubuntu_x86_64 (4daa0c00-3bf9-42aa-85ba-d5e67ba70cce)             |
| key_name                             | testkey                                                          |
| metadata                             | {}                                                               |
| name                                 | ubuntux86vm                                                      |
| os-extended-volumes:volumes_attached | []                                                               |
| progress                             | 0                                                                |
| security_groups                      | default                                                          |
| status                               | ACTIVE                                                           |
| tenant_id                            | 64de5a3d843c4e9d8173b1f0c2d54f59                                 |
| updated                              | 2014-09-14T14:31:15Z                                             |
| user_id                              | 13f4fcb07a5f447e9b3fb5555a0eeabc                                 |
+--------------------------------------+------------------------------------------------------------------+
```

8. Given that the image used in Example 4-196 on page 469 was the Ubuntu cloud-ready image downloaded from Ubuntu website following the instructions from “Obtain cloud-ready images from the Linux distributions” on page 424, the SSH login using root user is disabled and the only way to login is using Key-Pair injection using the `ubuntu` user (configured in the `/etc/cloud/cloud.cfg` file in the image), as shown in Example 4-198.

Example 4-198 Accessing the instance deployed using command-line interface

```
[root@cmocontroller ~]# ssh ubuntu@192.168.21.34
The authenticity of host '192.168.21.34 (192.168.21.34)' can't be established.
RSA key fingerprint is c2:41:70:42:6d:b3:d3:90:fd:4f:0a:9e:5f:6b:51:3c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.21.34' (RSA) to the list of known hosts.
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-35-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sun Sep 14 14:30:43 UTC 2014

  System load: 1.04              Memory usage: 2%   Processes:       93
  Usage of /:  56.0% of 1.32GB   Swap usage:   0%   Users logged in: 0

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud
```

```
0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@ubuntux86vm:~$
```

Important: You can provide multiple network adapters by using more than one --nic argument. Remember to provide a PowerVC flavor if you are deploying instances in PowerVM Compute nodes by using PowerVC images:

```
# nova boot --image AIX_6100-09 --flavor PVC-m1.large --nic
net-id=d0b0de11-7c0b-496a-af3c-11d9900637a7 --nic
net-id=d828cdb1-e926-4f94-b276-29a41471e5b6  aix6100vm
```

Stop and start an instance

To stop (shutdown) an instance, log in to the Controller node using SSH and run the **nova stop <instance_name>** command:

```
# nova stop ubuntux86vm
```

Use the **nova start <instance_name>** command to turn an instance on:

```
# nova start ubuntux86vm
```

Note: You may use the **nova list** command to view the instance names.

Delete an instance

To delete an instance by using the CLI, log in to the Controller node using SSH and run the **nova delete <instance_name>** command:

```
# nova delete ubuntux86vm
```

Note: If you have more than one instance using the same name, you can use the instance ID rather than the instance name. Use the **nova list** command to obtain the instance ID of the instance you want to delete:

```
# nova list
# nova delete <ID>
```

4.13.4 Identify an instance in the Compute node

The instance name in the hypervisor (the Compute node where it is running) is different than the name provided in the Cloud Manager. You can find exactly the Compute node where an instance is running and its exact name using either the Cloud Management dashboard, self-service portal or the command-line interface.

Identify an instance using the Cloud Management dashboard

To use the Cloud Management dashboard to identify the Compute node where an instance is running and its exact name in the Compute node, complete the tasks that follow. The steps are different for PowerVM Compute nodes as compared to KVM or PowerKVM Compute nodes.

Identify instances in KVM and PowerKVM Compute nodes

To identify your instances in KVM or PowerKVM Compute nodes, follow these steps:

1. Log in to the Cloud Management dashboard and click `ADMIN` → `System Panel` → `instances`.
2. Obtain the name of the Compute node where the instance is running (`Host` column), as shown in Figure 4-192.



Figure 4-192 Obtaining the Compute node where the instance is running

3. Click the instance `Name` to obtain the instance `ID`, as shown in Figure 4-193.



Figure 4-193 Obtaining the instance ID in Cloud Management dashboard

4. Click **ADMIN > System Panel > Hypervisors**, and click the hypervisor host name where the instance is running, which you obtained in step 2 on page 473. Use the instance ID obtained in step 3 to find the instance name, as shown in Figure 4-194.

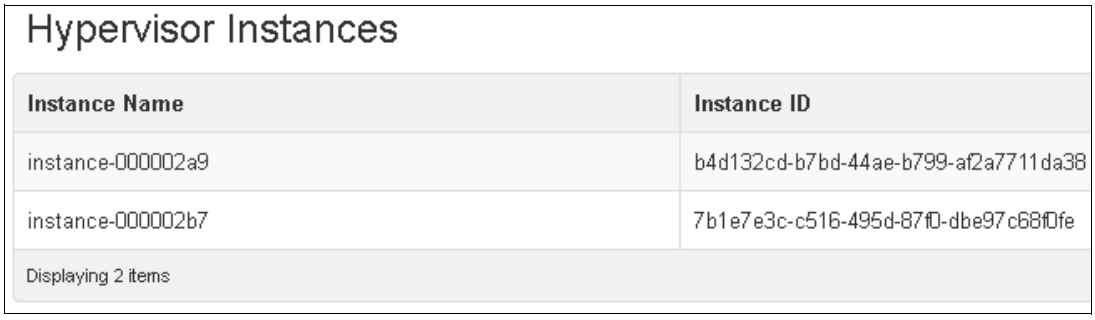


Figure 4-194 Identifying the instance name

5. Log in to the Compute node where the instance is running and use the **`virsh list`** command to see your instance, as shown in Example 4-199.

Example 4-199 Using the virsh list command

```
[root@powerkvm2 ~]# virsh list
 Id    Name                           State
----------------------------------------------------
 41    instance-000002a9              running
 44    instance-000002b7              running
```

Identify instances in PowerVM Compute nodes

To identify your instances in PowerVM Compute nodes, follow these steps:

1. Log in to the Cloud Management dashboard and click **ADMIN > System Panel > instances**.

2. Obtain the name of the Compute node where the instance is running (`Host` column), as shown in Figure 4-195.

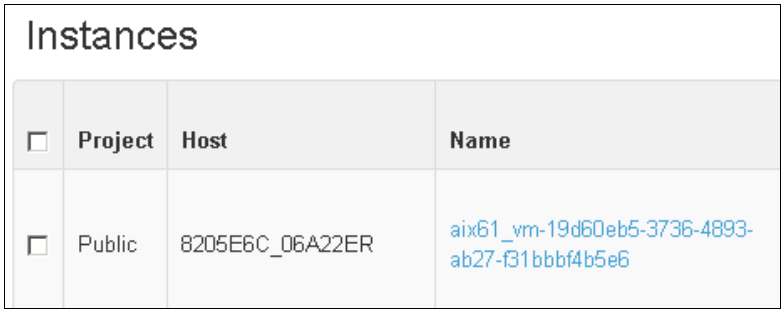


Figure 4-195 Obtaining the PowerVM Compute node where the instance is running

3. Click the instance `Name` and obtain its instance `ID`, as shown in Figure 4-196.

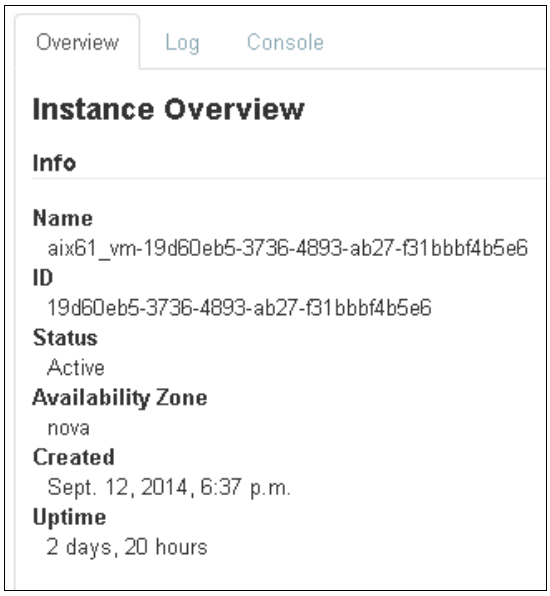


Figure 4-196 Obtaining the instance ID in Cloud Management dashboard

4. Log in to the PowerVC server and look for a virtual machine with the same ID, as shown in Figure 4-197.

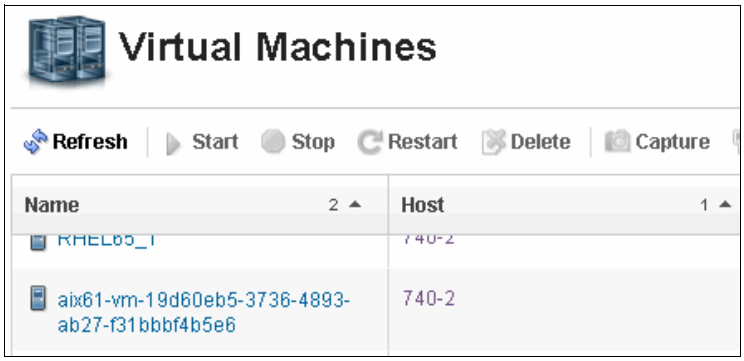


Figure 4-197 Finding the instance in PowerVC server

5. Still in PowerVC server, click the virtual machine name and go to the **Details** section. Obtain the hypervisor partition name, as shown in Figure 4-198.

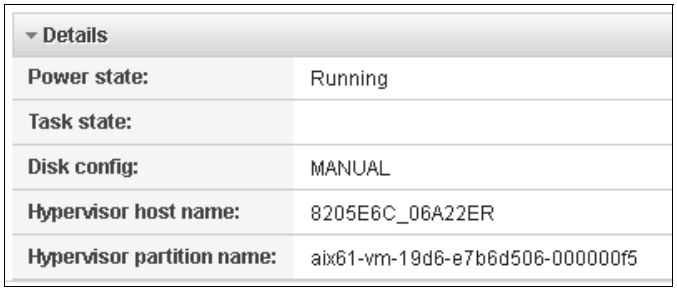


Figure 4-198 Obtaining the hypervisor partition name from PowerVC

6. Log in to the Hardware Management Console and click the Managed System where the instance is running (which you can see on hypervisor host name in step 5, and you should see the logical partition using the name you obtained in step 5, as shown in Figure 4-199.

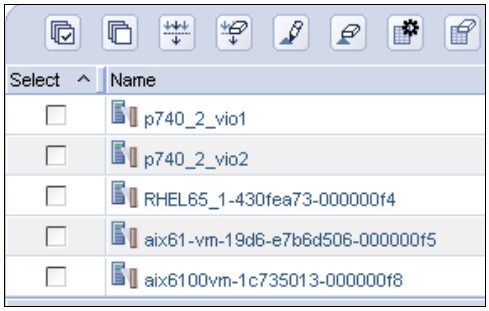


Figure 4-199 Finding the instance in PowerVM Compute node using the HMC

Identify an instance using the self-service portal

If you are using the self-service portal, use the following steps to identify your instance in the Compute node:

1. Log in to the self-service portal and click `instances`, and then click the instance name that you want to identify. Click **virtual machine Properties**, as shown in Figure 4-200 on page 477.

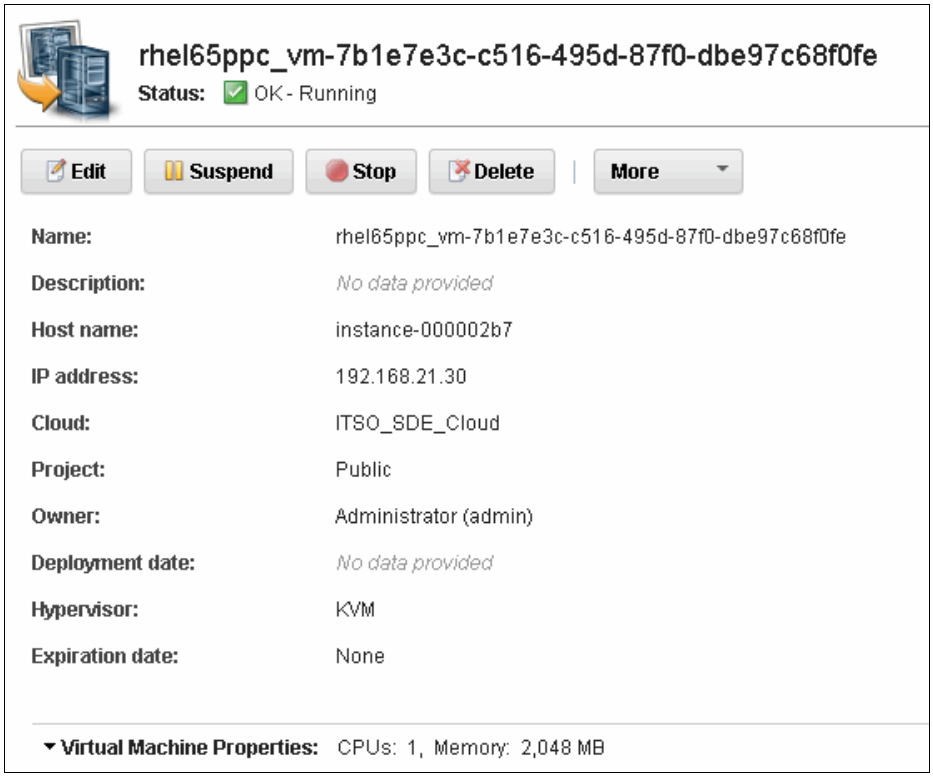


Figure 4-200 Accessing the virtual machine Properties

2. You can get the following information in this section:

Name	Instance name in Cloud Manager
UUID	Instance ID in Cloud Manager
Hypervisor Hostname	Host name of the Compute node where the instance is running
Instance Name	Instance name in the Compute node

Figure 4-201 on page 478 shows the virtual machine properties in self-service portal of an instance running in PowerKVM Compute node.

Property	Value
Flavor ID	2
Image Architecture	Power
Image ID	9af3d32a-80c7-4865-8173-4756cd70e5b6
Image Name	rhel65-cloudinit_powerkvm
Key Name	testkey
Memory (MB)	2048
Name	rhel65ppc_vm-7b1e7e3c-c516-495d-87f0-dbe97c68f0fe
CPUs	1
Tenant ID	64de5a3d843c4e9d8173b1f0c2d54f59
Tenant Name	Public
UUID	7b1e7e3c-c516-495d-87f0-dbe97c68f0fe
Volumes Attached	
Guest Hostname	
Guest IP Addresses	192.168.21.30
Access IPv4	
Access IPv6	
Config Drive	
Created	Friday, September 12, 2014 1:11:13 PM GMT
Disk Config	AUTO
Host	powerkvm2.itso.ibm.com
Host ID	e531b677b93b3edfd16ce81c7ad0e4204855c7c145255df5885b0
Hypervisor Hostname	powerkvm2.itso.ibm.com
Hypervisor Type	qemu
Instance Name	instance-000002b7

Figure 4-201 Virtual machine Properties in self-service portal

Now you can log in to your Compute node and find your instance by using the `Name instance`.

Note: In PowerVM Compute nodes, after obtaining the instance ID (`UUID`), use the steps from “Identify instances in PowerVM Compute nodes” on page 474 to find your instance using the PowerVC server.

Identify an instance by using the command-line interface

To identify your instances using command-line interface (CLI), use the following steps:

1. Log in to the Controller node by using SSH.
2. Use the **source** command to read the environment variables from the `/root/openrc` file:

   ```
   # source /root/openrc
   ```

3. Use the **nova list** command to obtain a list of the instances running in your OpenStack cloud:

   ```
   # nova list
   ```

4. Use **nova show <instance_name>** to obtain details about your instance, as shown in Example 4-200.

Example 4-200 Obtaining instance details using the nova show command

```
[root@cmocontroller ~]# nova show rhel65ppc_vm-7b1e7e3c-c516-495d-87f0-dbe97c68f0fe
+--------------------------------------+----------------------------------------------------------------------+
| Property                             | Value                                                                |
+--------------------------------------+----------------------------------------------------------------------+
| IBM-PVM:health_status                | {}                                                                   |
| OS-DCF:diskConfig                    | AUTO                                                                 |
| OS-EXT-AZ:availability_zone          | nova                                                                 |
| OS-EXT-SRV-ATTR:host                 | powerkvm2.itso.ibm.com                                               |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | powerkvm2.itso.ibm.com                                               |
| OS-EXT-SRV-ATTR:instance_name        | instance-000002b7                                                    |
| OS-EXT-STS:power_state               | 1                                                                    |
| OS-EXT-STS:task_state                | -                                                                    |
| OS-EXT-STS:vm_state                  | active                                                               |
| OS-SRV-USG:launched_at               | 2014-09-12T13:00:30.000000                                           |
| OS-SRV-USG:terminated_at             | -                                                                    |
| Private network                      | 192.168.21.30                                                        |
| accessIPv4                           |                                                                      |
| accessIPv6                           |                                                                      |
| config_drive                         |                                                                      |
| created                              | 2014-09-12T13:11:13Z                                                 |
| flavor                               | m1.small (2)                                                         |
| hostId                               | e531b677b93b3edfd16ce81c7ad0e4204855c7c145255df5885b0aee             |
| id                                   | 7b1e7e3c-c516-495d-87f0-dbe97c68f0fe                                 |
| image                                | rhel65-cloudinit_powerkvm (9af3d32a-80c7-4865-8173-4756cd70e5b6)     |
| key_name                             | testkey                                                              |
| metadata                             | {}                                                                   |
| name                                 | rhel65ppc_vm-7b1e7e3c-c516-495d-87f0-dbe97c68f0fe                    |
| os-extended-volumes:volumes_attached | []                                                                   |
| progress                             | 0                                                                    |
| security_groups                      | default                                                              |
| status                               | ACTIVE                                                               |
| tenant_id                            | 64de5a3d843c4e9d8173b1f0c2d54f59                                     |
| updated                              | 2014-09-12T19:13:27Z                                                 |
| user_id                              | 13f4fcb07a5f447e9b3fb5555a0eeabc                                     |
+--------------------------------------+----------------------------------------------------------------------+
```

From this output, you can see the following information:

- `OS-EXT-SRV-ATTR:hypervisor_hostname`: Host name of the Compute node where the instance is running
- `OS-EXT-SRV-ATTR:instance_name`: Instance name in the Compute node
- `name`: Instance name in Cloud Manager

- `id`: Instance ID in Cloud Manager

You can now log in to your Compute node and find your instance using the instance name.

Note: In PowerVM Compute nodes, after obtaining the instance ID, use the steps from “Identify instances in PowerVM Compute nodes” on page 474 to find your instance using the PowerVC server.

4.14 Manage flavors in IBM Cloud Manager

During the deployment of instances, you must select the flavor that the instance will use. The selected flavor determines the disk size, amount of RAM, and number of CPUs. This section covers the following topics that help you create additional flavors to use in your OpenStack cloud:

- “Create flavors for KVM and PowerKVM Compute nodes” on page 480
- “Create flavors for PowerVM Compute nodes” on page 482

4.14.1 Create flavors for KVM and PowerKVM Compute nodes

Log in to the Cloud Management dashboard and click **ADMIN > System Panel > Flavors**, and then click **Create Flavor**. You must provide the following information to create a flavor:

Name	Provide a name to identify your flavor.
ID (Optional)	Specify an ID. However, it is best to leave this as `auto` so that Cloud Manager can generate the ID.
VCPUs	Number of virtual CPUs to provide for the instance.
RAM MB	Amount of RAM memory to use (in megabytes).
Root Disk GB	Amount of disk space (in Gigabytes) for the disk where the operating system will be installed.
Ephemeral Disk GB	Amount of disk space (in Gigabytes) to provide for an instance as an additional disk. If you do not want to specify an additional disk, use `0` (zero) in this value.
Swap Disk MB	Amount of swap space to provide for the instance (in megabytes). If you do not want to specify the amount, leave this field with `0` (zero).

Figure 4-202 shows an example of creating a flavor.

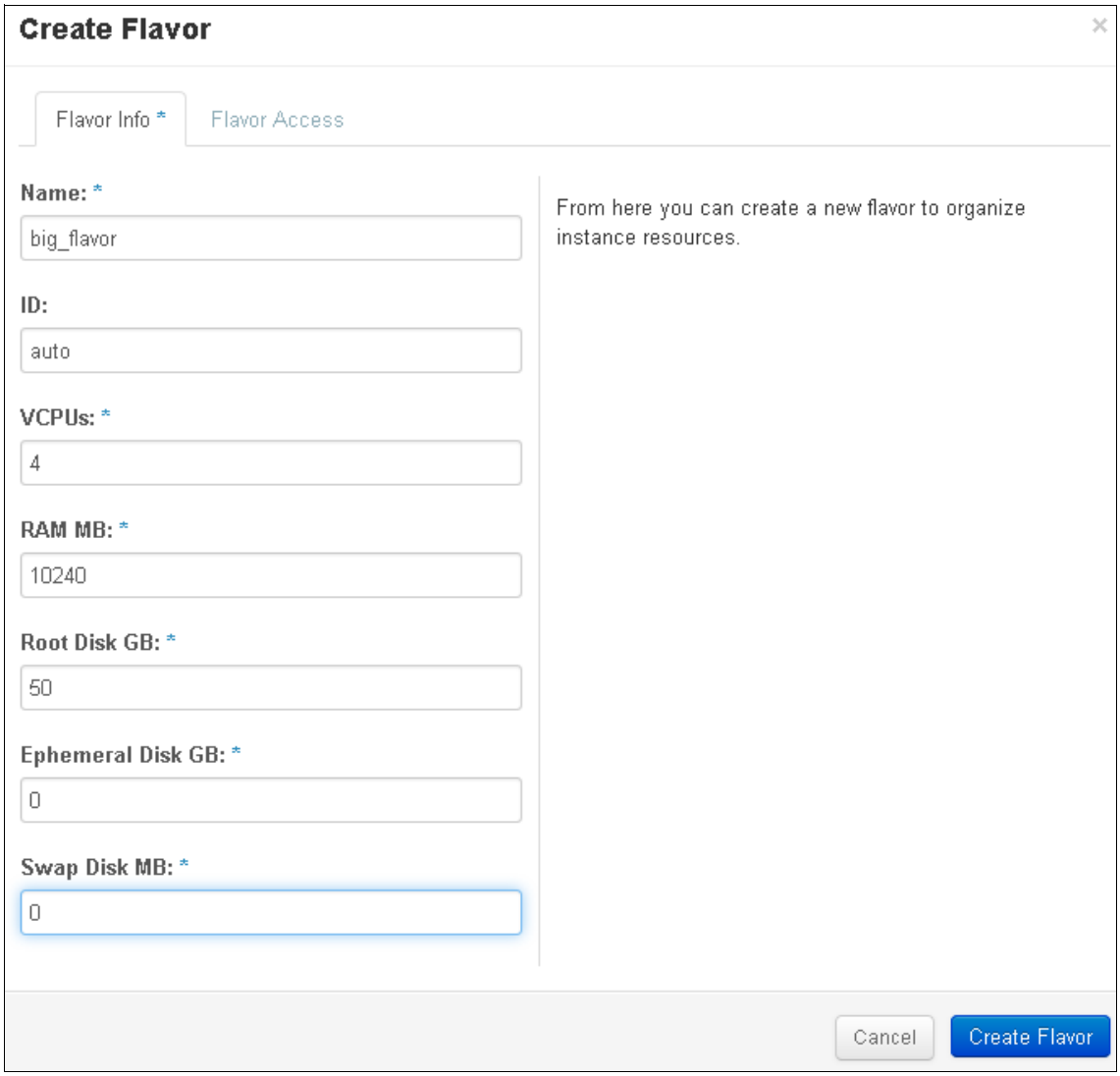


Figure 4-202 Creating a flavor for your OpenStack cloud

After completing this step, you can use the flavor during instances deployment, as shown in Figure 4-203.

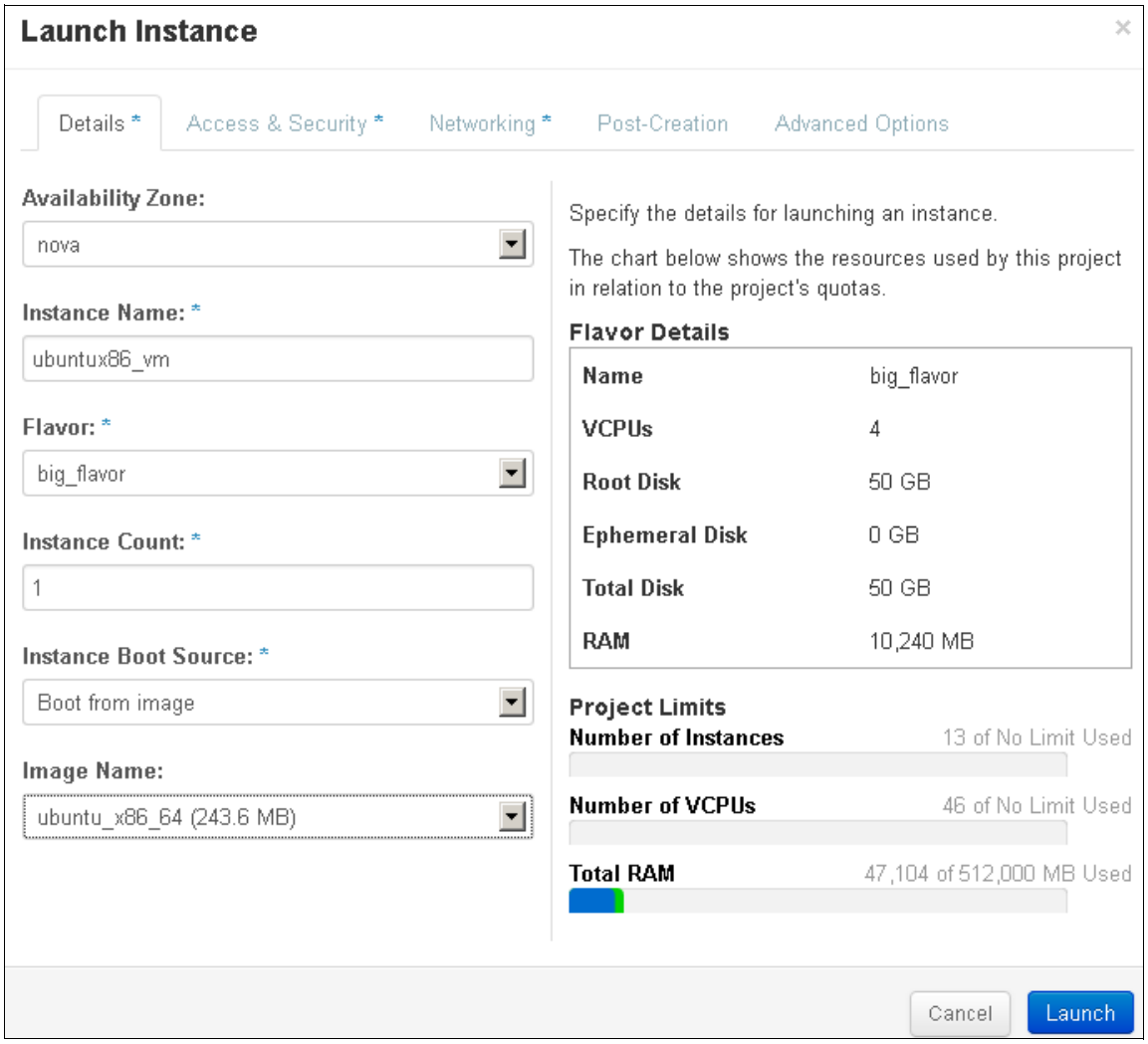


Figure 4-203 Using the flavor during instance deployment

4.14.2 Create flavors for PowerVM Compute nodes

The instances deployed in PowerVM Compute nodes, must use flavors available in the PowerVC server, given that the Controller node will interact with the PowerVC server to deploy the instance.

1. To create additional flavors to use in the PowerVM Compute nodes, log in to the PowerVC server and click **Configuration > Compute Templates**, as shown in Figure 4-204 on page 483.

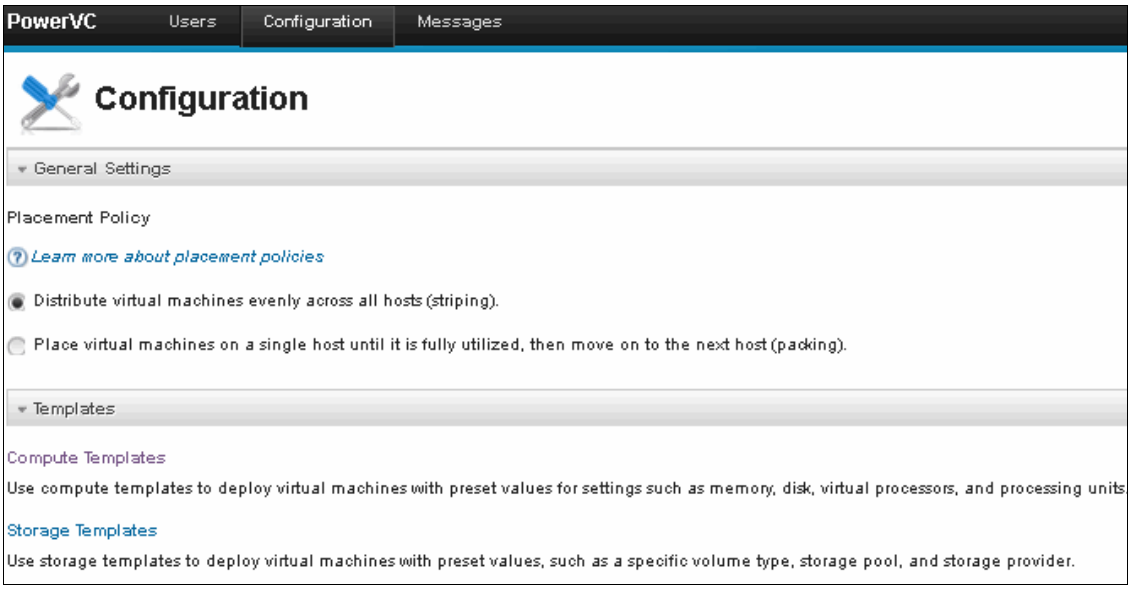


Figure 4-204 Creating flavors in PowerVC server

2. Click **Create** and provide the following information:

Template Name	The name to identify your flavor.
Virtual Processors	Number of Virtual CPUs to provide for the instance.
Memory (MB)	Amount of Memory to provide for the instance (in megabytes).
Processing Units	Number of processing units to use.
Disk (GB)	Amount of disk for the operating system (in Gigabytes).
Compatibility Mode	Optionally, select the processor compatibility mode (POWER6, POWER6+, or POWER7).

Figure 4-205 on page 482 shows a PowerVC flavor being created.

Note: You can also modify from Basic to Advanced to customize more details in the flavor or template.

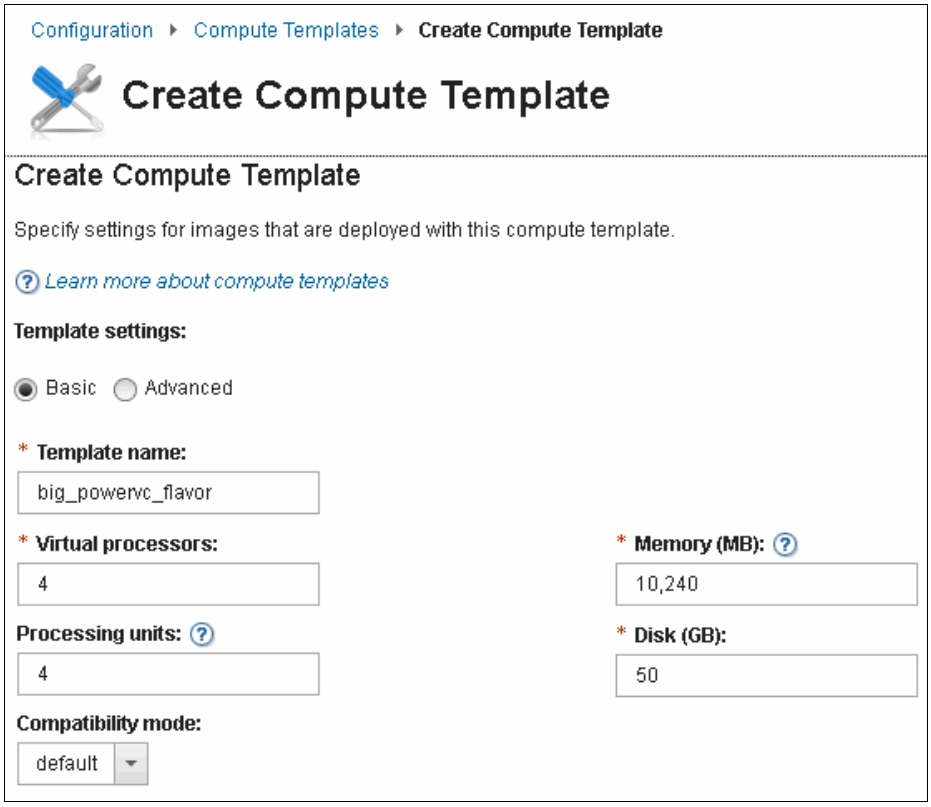


Figure 4-205 Creating a flavor in the PowerVC server

3. When all of the information is provided, click **Create Compute Template** to complete this task. The PowerVC driver in Controller node synchronizes the information with PowerVC server every five minutes. After this synchronization cycle is complete, look into the `/var/log/powervc/nova-powervc.log`. You should be able to see the flavor synchronization, as shown in Example 4-201.

Example 4-201 Flavor synchronization between PowerVC server and Controller node

```
2014-09-15 15:48:29.227 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavors synchronization
starts.
2014-09-15 15:48:29.408 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor:
7a0233222c4d43fa0d3a6bbfe45ca150>
2014-09-15 15:48:29.409 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor:
ba3579f80e5cd0db6a685649112c4d4b>
2014-09-15 15:48:29.409 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor: m1.tiny>
2014-09-15 15:48:29.557 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor:
803f92d54ee8e073c958241a378c4e98>
2014-09-15 15:48:29.557 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor: m1.small>
2014-09-15 15:48:29.713 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor:
dbdb6fef22b6db950b6e5661ffa01c5d>
2014-09-15 15:48:29.713 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor: m1.medium>
2014-09-15 15:48:29.788 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor: m1.large>
2014-09-15 15:48:29.853 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor:
382d184d5f298847679dc97da290699f>
2014-09-15 15:48:29.853 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor: m1.xlarge>
2014-09-15 15:48:30.100 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor:
big_powervc_flavor>
2014-09-15 15:48:30.193 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor:
1ad421559fd25720151c071221e693a3>
2014-09-15 15:48:30.193 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor:
26ff143a47099868a7a844271ebf64d5>
2014-09-15 15:48:30.194 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor:
3126ebb87c568516751314e7b298511a>
2014-09-15 15:48:30.194 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavor:<Flavor:
cb8513cbf6befc71bb43daa73957e9e9>
2014-09-15 15:48:30.194 38447 INFO powervc.nova.driver.virt.powervc.sync.flavorsync [-] Flavors synchronization ends.
```

When this task is completed, you should be able to deploy instances in PowerVM Compute nodes using the flavor created in PowerVC server, as shown in Figure 4-206.

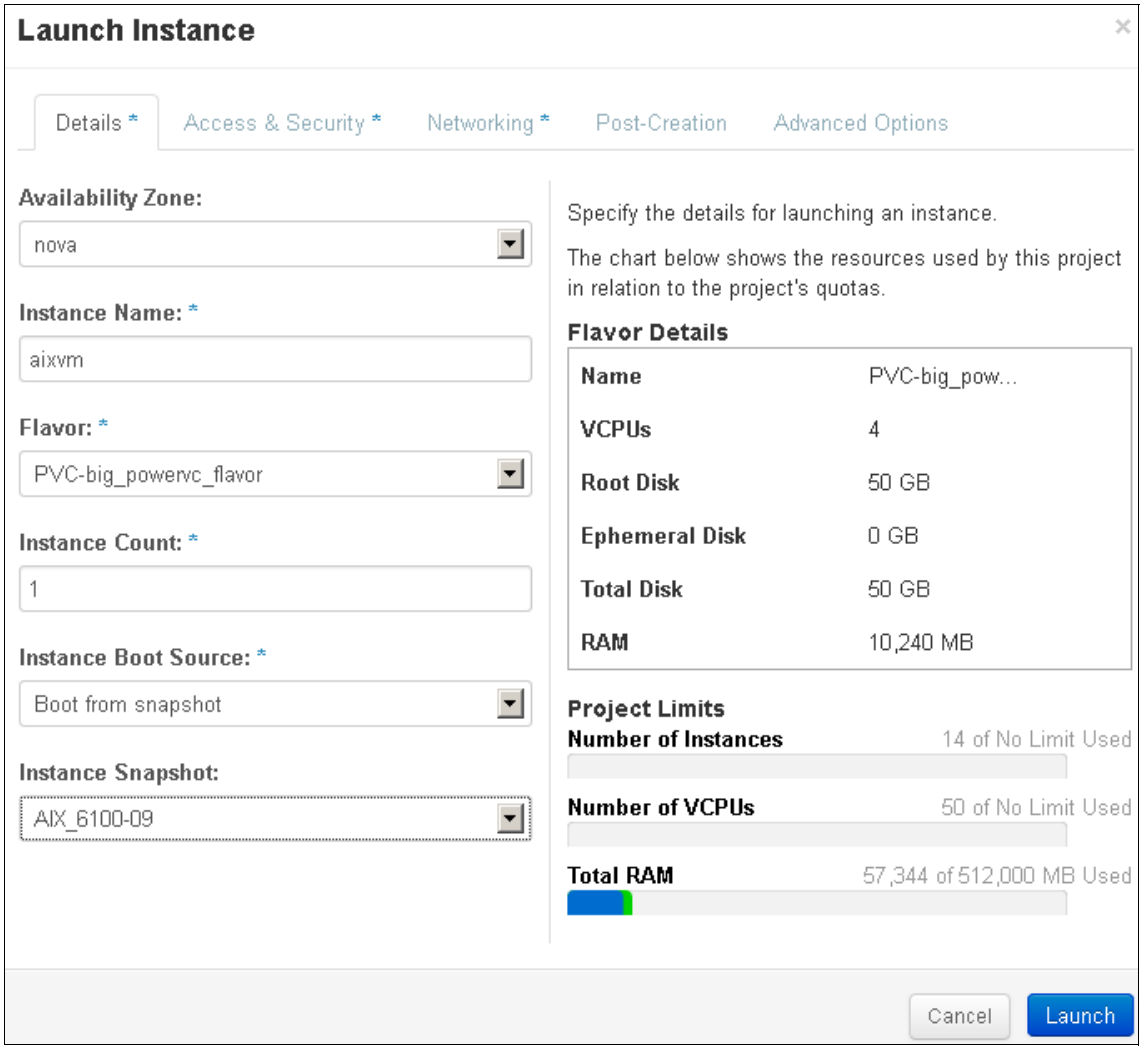


Figure 4-206 Using the PowerVC flavor in the Cloud Management dashboard

4.15 Use SSH key pairs in IBM Cloud Manager

You can create SSH key pairs to inject in instances deployed in KVM or PowerKVM Compute nodes, so you can log in to the instances without providing regular user and password authentication. To use key pairs, the images must have been prepared with cloud-init.

Note: For cloud-ready images downloaded from the Linux distributions websites, usually the only way to log in is by using key pairs, given that SSH login using root user is generally disabled and the cloud user created for the image does not have a password. For more information about obtaining and using cloud-ready images, see “Obtain cloud-ready images from the Linux distributions” on page 424.

4.15.1 Create key pairs

1. To create key pairs, log in to the Cloud Management dashboard and click **Project > Compute > Access & Security**, and then click the **Key Pairs** tab, as shown in Figure 4-207.

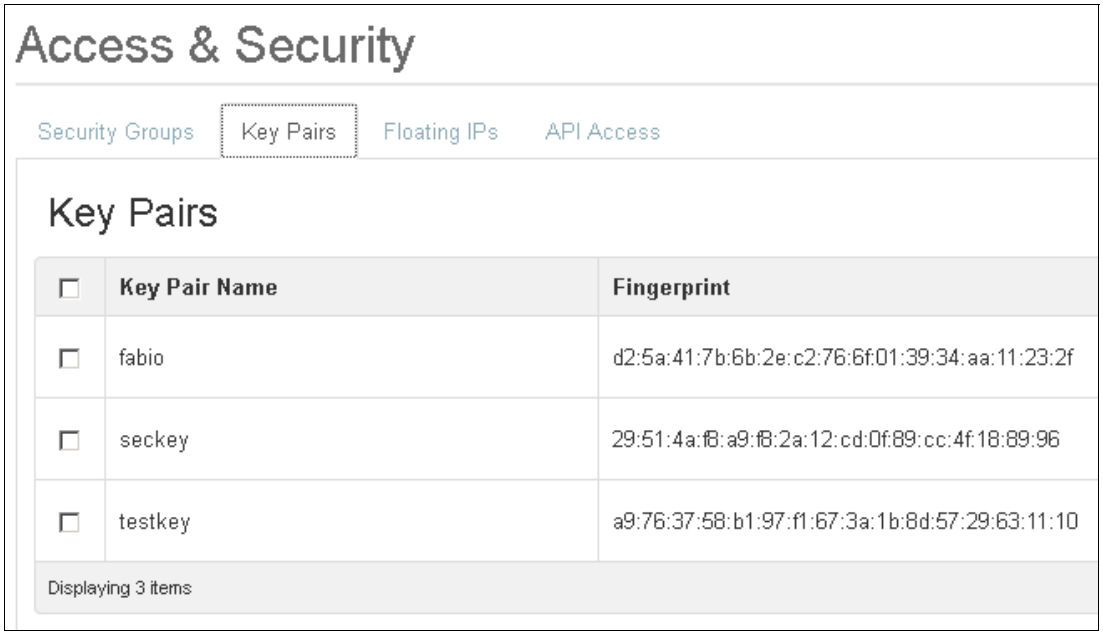


Figure 4-207 Key pairs tab in Cloud Management dashboard

2. Click **Create Key Pair,** and provide a name for the key pair, as shown in Figure 4-208.

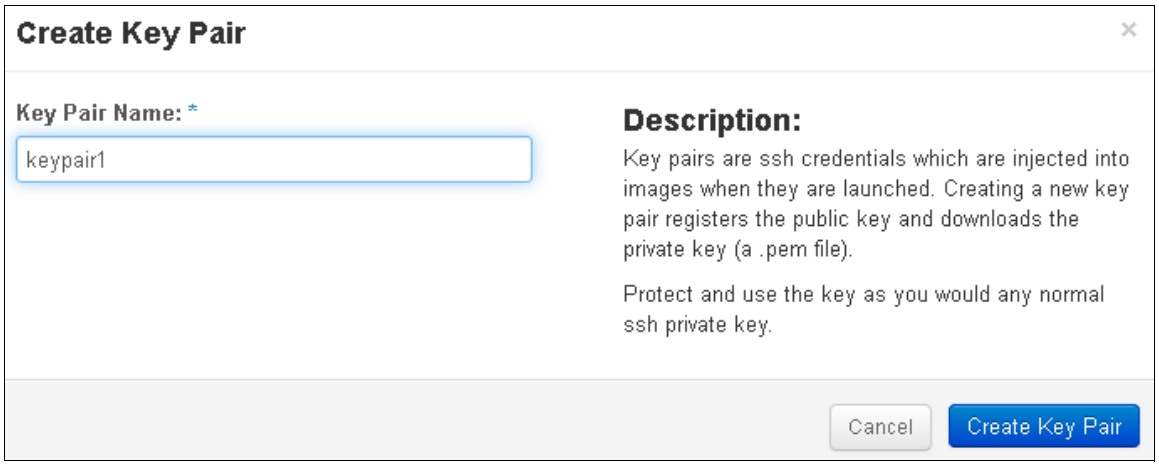


Figure 4-208 Creating a key pair

3. After creating the key pair inside Cloud Manager, the Cloud Management dashboard will automatically prompt you to download the key pair file, as shown in Figure 4-209 on page 487.

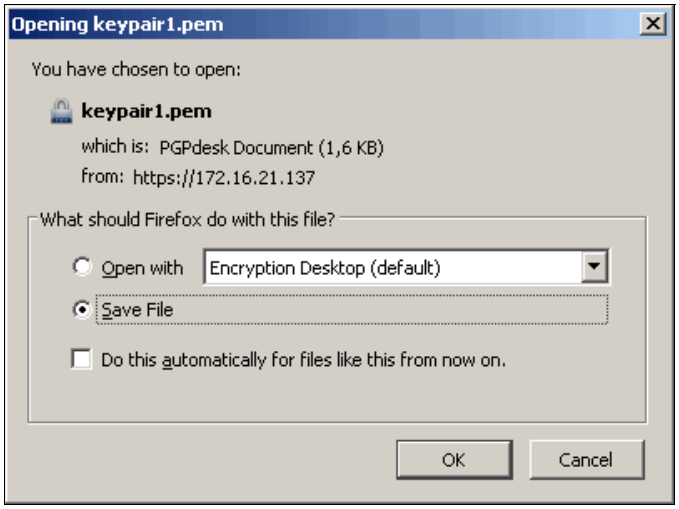


Figure 4-209 Downloading key pairs from Cloud Management dashboard

Important: Save this file in a safe place, because it will allow you to log in to the instances deployed in your cloud without providing a password.

You can also create the key pair using the CLI rather than the Cloud Management dashboard. Example 4-202 shows the creation of a key-pair using command line with the **`nova keypair-add`** command.

Example 4-202 Creating a key pair using command-line interface (CLI)

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# nova keypair-add keypair2 > keypair2.pem
[root@cmocontroller ~]# cat keypair2.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAzVPrnRNELhrSbOeBBISO1XfQiuOajPJ78GmUpeZIBaFOOxZl
e81GJ/VTidWw1KNhcu2A8eo+8rJa11r/C7lnq5WI+C9wOMHv4W5fD2JzUfXa/bH6
pKsTmxbThxZR+HeCuuErBhpE2f+bBSEICY+q2VhY25Of+nFgNlEBaIz9gTUNl8VA
+/5NCU4dzcq9Jxh6X74hL2i4+aRTQM5b2cXWlU1lRvFpj2yBkjpRze6uVahR+O7z
sovh3aksQtuyyW3Ls1ToTqFdEwIo7CGtH1Qg9j/D1hXLyHIw4o7zxw5tbf4uTRvg
LtVi1umMvO2mQ4t9Tx1CEyGG7APlz4hBefOs9QIBIwKCAQEAnmVPW+pKifdvIqTf
3unqh2sOlwdZ97O+siWP7bGlRiwX1c9y29jeWVbaCz5yeCZD3FDRL6YTTYJGE9h7
m1SDLJftUb4x/+YQOnm3C947tEFRKhuVd7csYbKF694EuFwxphtqVSo1HTLlTRl7
OpNmimi5k3Hhz9PN4MIlqGzDj43wRsIGDxcN+iOBjnfVVyjubiYU3JFKJXOoiZHf
x9D5WRrYLrgtl7Eft7WQGJkPVnSphXSru3rbOcVVDkxw2xJBGXC5nNd5ZqDqKnGH
gZvgCfPBiPBM+4tO2DqfW81qgx7+nv5QuPwxkfKKpqfn+Ju1RbmqhkE1NL6O4g6E
IyONFwKBgQD8UTZjkoy9mh9xdek35bmwBIwxaZBncODr5r7V1QMwpQW5QsGWxP6i
iHez+GHFoE65zEolPFa89WxBx6W3yOvOObpyEhqkrnpguWvBxMD4jJdGmDKqDxvf
gsK/n1vsLmSOvKMRT+kTXRVuHxbEcIm4yMOHn74nqaHd/ywi8rjDGwKBgQDQUyGT
L/TcIWYcGnWWoPpS6NqlUpc9QtQf1yMFHi8D5RHpHPU4392Zem72AYQgrZL1UDz5
U8VLGJvuAGEc2KtckTe+hCOIDuaYpluj/mpIXMHXskzQ5QnWWna66ZVDB2fQr2gH
amMFvKMyARFfyaA4W1YRjHcjxR8oq1LIko7BLwKBgA5rCmwXALpglBUcrj21WxFf
WHfaJYJBGacUf+72OrJSkpzfPkMhMx86/4agep2UIcF5Y1KVu9BIidffhc/8/Qas
5hUlmx9abWSc4ZYLPjoWqY8BYfsWzmSLIRJDntpLyz2HHOLRXcaXmtMXuCh7dZWH
OTOiumiqm4kHRFnEuhnHAoGAF878hdmYU6wLq3C8+OWY84hTgJu5g1gYPic3M8jv
bijO2NAcBn/8EYpVOvjb5no8rlJQHH6aNHfXTmZxf6O7OBCYqA8bFtOhqwuys6gM
JYbxlP5vLdEIcETaXoEYX26PhZBjqRN5B/hNKklDOG7Qe3gnF/LLyZLe/VVoi/N+
B3MCgYEAv8UQxPZnX8w2Ul3kAgZ+zBrU6RKaXreOdKHGA1uDnNt/qgfqpNZmlJIX
```

```
6MHPQM5vJLxoBPAnHuU+yTxGiOqDRRtoUoK5zirYvOk4fdCA3jaLOXhPKn7uhieq
DIPUJUiUOZFugTva72kXPvl3pgjSvM19A2QCwoXJtWW43t3GUfA=
-----END RSA PRIVATE KEY-----

[root@cmocontroller ~]# nova keypair-list
+----------+-------------------------------------------------+
| Name     | Fingerprint                                     |
+----------+-------------------------------------------------+
| fabio    | d2:5a:41:7b:6b:2e:c2:76:6f:01:39:34:aa:11:23:2f |
| keypair1 | fb:28:ff:0f:9b:f0:c3:4f:a1:4e:7f:10:e2:95:f7:3d |
| keypair2 | c8:97:c4:55:a1:54:12:a2:27:67:23:e7:1c:07:5d:73 |
| seckey   | 29:51:4a:f8:a9:f8:2a:12:cd:0f:89:cc:4f:18:89:96 |
| testkey  | a9:76:37:58:b1:97:f1:67:3a:1b:8d:57:29:63:11:10 |
+----------+-------------------------------------------------+
```

Optionally, you can use your own SSH key in the Cloud Manager key pair, as shown in Example 4-203.

Example 4-203 Creating key pair by using an existing SSH key

```
[root@cmocontroller ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
0e:a2:84:14:2d:76:d7:68:66:a2:e9:36:06:9a:2e:fe root@pw4303-l1
The key's randomart image is:
+--[ RSA 2048]----+
| ..   o          |
| o.+ * .         |
|..= *            |
|o+               |
|=.. . . S        |
|o* . . o         |
|+ o     .        |
|..               |
|o..E             |
+-----------------+

[root@cmocontroller ~]# nova keypair-add --pub-key /root/.ssh/id_rsa.pub testkey
```

4.15.2 Inject key pairs in instances

You can provide the key pairs to the deployed instances by using a config drive or metadata service. Natively, only the config drive method is available in Cloud Manager, but some customizations can be performed to configure the metadata service.

Inject key pairs by using Config Drive

During its deployment, the instance can obtain some information from the Controller node to automate some tasks, including configuring hostname and injecting the SSH key pair. You can provide this information through a special *Configuration Drive* that will be attached to the instance when it boots. The instance can obtain the information from this drive by mounting this disk and reading the files from it. This option is enabled by default in Cloud Manager and it requires a cloud-init package in the image configured with the `ConfigDrive` option, as shown in Example 4-204.

Example 4-204 Enabling ConfigDrive in cloud.cfg cloud-init configuration file

```
# cat /etc/cloud/cloud.cfg
users:
 - default

disable_root: 0
ssh_pwauth:   1

locale_configfile: /etc/sysconfig/i18n
mount_default_fields: [~, ~, 'auto', 'defaults,nofail', '0', '2']
resize_rootfs_tmp: /dev
ssh_deletekeys:   1
ssh_genkeytypes:  ~
syslog_fix_perms: ~
disable_ec2_metadata: True
datasource_list: ['ConfigDrive']
.
. <some lines were removed>
.
```

Note: The cloud-ready images obtained from the Linux distributions websites following the instructions from “Obtain cloud-ready images from the Linux distributions” on page 424 are usually prepared with the ConfigDrive option.

If the image used in the deployment was prepared with the ConfigDrive option, you can enable the Config Drive during the instance deployment, so it can obtain the key pair. You can use the CLI or the self-service portal to deploy an instance using a Configuration Drive.

Note: At the time of writing this documentation, you cannot enable the Configuration Drive while deploying instances using the Cloud Management dashboard.

Example 4-205 shows how you can deploy an instance with Configuration Drive by using the CLI by providing the **`--config-drive True`** option with the **`nova boot`** command.

Example 4-205 Deploying instances by using the Configuration Drive

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# nova boot --image ubuntu_x86_64 --flavor m1.medium --key-name keypair1
--nic net-id=d828cdb1-e926-4f94-b276-29a41471e5b6 ubuntux86vmkp1 --config-drive True
+--------------------------------------+------------------------------------------------------+
| Property                             | Value                                                |
+--------------------------------------+------------------------------------------------------+
| IBM-PVM:health_status                | {}                                                   |
| OS-DCF:diskConfig                    | MANUAL                                               |
| OS-EXT-AZ:availability_zone          | nova                                                 |
```

```
| OS-EXT-SRV-ATTR:host                 | -                                                    |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                                    |
| OS-EXT-SRV-ATTR:instance_name        | instance-000002c1                                    |
| OS-EXT-STS:power_state               | 0                                                    |
| OS-EXT-STS:task_state                | scheduling                                           |
| OS-EXT-STS:vm_state                  | building                                             |
| OS-SRV-USG:launched_at               | -                                                    |
| OS-SRV-USG:terminated_at             | -                                                    |
| accessIPv4                           |                                                      |
| accessIPv6                           |                                                      |
| adminPass                            | aL29d4P9afJU                                         |
| config_drive                         | True                                                 |
| created                              | 2014-09-16T00:41:29Z                                 |
| flavor                               | m1.medium (3)                                        |
| hostId                               |                                                      |
| id                                   | 2d23a935-7bdf-4436-9b2a-36bd98b6e78b                 |
| image                                | ubuntu_x86_64 (4daa0c00-3bf9-42aa-85ba-d5e67ba70cce) |
| key_name                             | keypair1                                             |
| metadata                             | {}                                                   |
| name                                 | ubuntux86vmkp1                                       |
| os-extended-volumes:volumes_attached | []                                                   |
| progress                             | 0                                                    |
| security_groups                      | default                                              |
| status                               | BUILD                                                |
| tenant_id                            | 64de5a3d843c4e9d8173b1f0c2d54f59                     |
| updated                              | 2014-09-16T00:41:30Z                                 |
| user_id                              | 13f4fcb07a5f447e9b3fb5555a0eeabc                     |
+--------------------------------------+------------------------------------------------------+
```

If you are deploying your instance using the self-service portal, you select the **Enable config drive (used to pass additional configuration data)** option, as shown in Figure 4-210.

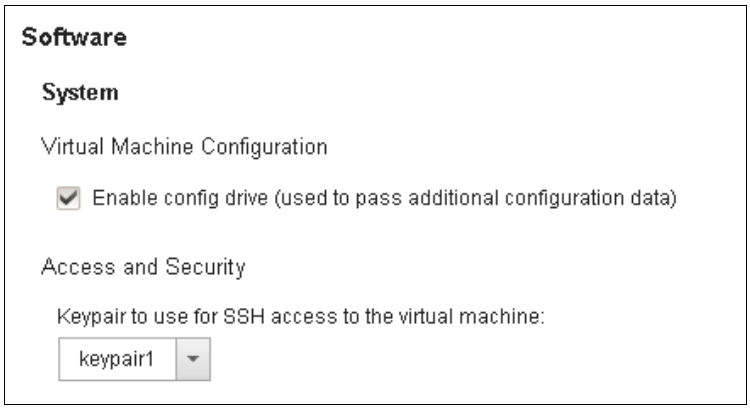


Figure 4-210 Deploying an instance with Config Drive in self-service portal

After completing this task, you can log in to the instance by using the key pair file (`.pem` file) that you obtained while creating the key pair in section 4.15.1, “Create key pairs” on page 486. If the key pair you are using is the default SSH key, `/root/.ssh/id_rsa.pub`, and you can log in by providing the appropriate cloud user name for Ubuntu images, for example:

```
# ssh ubuntu@<instance_ip_address>
```

If you are using a non-regular key pair (usually a `.pem` file), you can use the `-i` option in the SSH command to provide an identity file for reading the RSA or DSA authentication key:

```
# ssh -i keypair1.pem ubuntu@<instance_ip_address>
```

Inject key pairs using metadata service

During its deployment, the instance can also obtain the SSH key pair if the Controller node has the metadata service enabled. This is disabled by default in Cloud Manager.

The metadata service is implemented by either the nova-api service or the nova-api-metadata service (the nova-api-metadata service is generally only used when running in multihost mode). If you are running the nova-api service, you must have metadata as one of the elements of the list of the enabled_apis configuration option in `/etc/nova/nova.conf`.

Use the following steps to enable metadata service in the Controller node:

1. Log in to the Controller node and edit the `/etc/nova/nova.conf` file. Append metadata to `enabled_apis` field separated by comma:

   ```
   # a list of APIs to enable by default (list value)
   enabled_apis=osapi_compute,metadata
   ```

2. Add the metadata setting in /etc/nova/nova.conf under the `osapi_compute_listen_port` field. You should add the following entries:

   ```
   osapi_compute_workers = 2
   metadata_workers = 2
   metadata_host=192.168.21.176
   metadata_listen=0.0.0.0
   metadata_listen_port=8775
   metadata_manager=nova.api.manager.MetadataManager
   metadata_port=8775
   ```

   Note: Replace the IP address from `metadata_host` to your Controller node IP address that is accessible by your instances, which means the IP address from the data network.

3. Restart the openstack-nova-api service:

   ```
   # service openstack-nova-api restart
   ```

4. Check if the openstack-nova-api service is running:

   ```
   # service openstack-nova-api status
   openstack-nova-api (pid  21433) is running...
   ```

5. Check if the Firewall chain was updated for port 8775:

   ```
   # service iptables status
   ... <some lines were removed>
   Chain nova-api-INPUT (1 references)
   num target prot opt source destination
   1 ACCEPT tcp -- 0.0.0.0/0 192.168.21.176 tcp dpt:8775
   ... <some lines were removed>
   ```

After completing this step, you can use the Cloud Management dashboard, self-service portal or CLI to deploy instances by using a key pair. In this case, you do not need to provide the Configuration Drive option. Use regular instructions from section 4.13, “Manage instances in IBM Cloud Manager” on page 443 for more information about deploying instances and providing a key pair.

4.16 Developer tool installation: UrbanCode Deploy with Patterns

(Optional) You can implement and use IBM UrbanCode Deploy with Patterns in your OpenStack cloud to automate deployment and customization of instances which improves management features. For more information about UrbanCode Deploy with Patterns, see section 1.6, “Supporting technologies” on page 9.

IBM UrbanCode Deploy with Patterns uses HOT (Heat Orchestration Templates) to deploy instances and volumes in your OpenStack cloud, and customize the instances using scripts. For more information about HOT, see Chapter Chapter 7, “Cloud user deployment and use scenarios” on page 563.

The IBM UrbanCode Deploy with Patterns installation has the following requirements:

- Red Hat Enterprise Linux 6.5 on x86_64 platform (can be a physical server or a virtual machine).
- IBM UrbanCode Deploy server installation image: At the cloud created for this publication, the file `URBANCODE_DEPLOY_V6.1.0.1_EN_EVAL.zip` was used. Please, contact your IBM sales representative to obtain the appropriate license and installation media.
- IBM UrbanCode Deploy with Patterns Installation image: At the cloud created for this publication, the files `ibm-ucd-patterns-engine-6.1.0.1.544477.zip` and `ibm-ucd-patterns-web-6.1.0.1.545068.zip` were used. Please, contact your IBM sales representative to obtain the appropriate license and installation media.
- Device driver for MySQL. At the cloud used for this publication, the driver used was `mysql-connector-java-5.1.30-bin.jar`. You can obtain this driver from the following website:

  `https://dev.mysql.com/downloads/connector/j/`
- Use a fully qualified domain name and appropriate name resolution with `/etc/hosts` or `DNS` configuration in the UrbanCode Deploy server. For more information about configuring fully qualified domain name and name resolution, see “Use a fully qualified domain name” on page 204 and “Configure host name resolution” on page 204.
- Prepare a Yum repository with the Red Hat Enterprise Linux 6.5 installation media. For more information about preparing this Yum repository, see 4.6.6, “Set up a Yum repository” on page 242.
- The server where the UrbanCode Deploy and UrbanCode Deploy with Patterns software will be installed, must have access to the Internet, because required packages will be automatically downloaded and installed in the installation process.

After configuring your system to meet these requisites, use the following steps to install the UrbanCode Deploy and UrbanCode Deploy with Patterns in your server:

Important: Given that both applications will run in the same server in this example, customization of some port numbers to avoid conflicts is required. These customizations are considered and explained in this section.

1. Log in to the Urban Code server and install the following requisite packages from the Yum repository:

```
# yum -y install java-1.7.0-openjdk.x86_64
# yum -y install bind-utils
# yum -y install unzip
```

2. Extract the contents of the UrbanCode Deploy with Patterns Engine:

```
# unzip ibm-ucd-patterns-engine-6.1.0.1.544477.zip
```

   In this process, the directory `ibm-ucd-patterns-install/engine-install/` is created and the contents of the compressed file are extracted inside this directory.

3. Access the directory with the contents of the UrbanCode Deploy with Patterns Engine installation and execute the **`install.sh`** script:

```
# cd ibm-ucd-patterns-install/engine-install/
# ./install.sh
```

4. The license agreement is displayed and you are prompted to accept it. Read and accept the license agreement to continue the installation process:

```
If you accept the license agreement, enter 'y' [ n ] : y
```

5. The UrbanCode Deploy with Patterns Engine installation will prompt you to provide the URL of the OpenStack keystone server. This is your Cloud Manager authentication URL. You can obtain this information from the `OS_AUTH_URL` variable in the `/root/openrc` file in the Controller node, as shown in Example 4-206.

Example 4-206 Obtaining the authentication URL in the Controller node

```
[root@cmocontroller ~]# cat /root/openrc

# This file autogenerated by Chef
# Do not edit, changes will be overwritten

# COMMON OPENSTACK ENVS
export OS_USERNAME=admin
export OS_PASSWORD=openstack2
#export OS_TENANT_NAME=admin
export OS_TENANT_NAME=Public
export OS_AUTH_URL=http://172.16.21.137:5000/v2.0
export OS_REGION_NAME=RegionOne
```

   Provide this information to the UrbanCode Deploy with Patterns Engine installation:

```
Enter the url of the OpenStack Keystone server. In most cases, specify the url
of the OpenStack server. [ http://localhost:5000/v2.0/ ] :
http://172.16.21.137:5000/v2.0
```

6. Provide the host name or IP address of the UrbanCode Deploy server (where you are installing this software):

```
Enter the public IP address or host name of this system. This field forms the
URL that the designer uses to access the engine. [ ucdp ] : 172.16.21.151
```

7. The installation will proceed automatically. The installation output is too long to be displayed in this publication, but Example 4-207 shows partial output.

Example 4-207 UrbanCode Deploy with Patterns Engine installation

```
detected RHEL 6
service qpidd stop
qpidd: unrecognized service
stopping qpidd done
yum remove akonadi -y
Loaded plug-ins: product-id, refresh-packagekit, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Setting up Remove Process
No Match for argument: akonadi
Repository InstallMedia is listed more than once in the configuration
Package(s) akonadi available, but not installed.
No Packages marked for removal
mkdir -p /opt/ibm/ibm-ucd-p
tar xzf /root/Downloads/ibm-ucd-patterns-install/engine-install/ibm-yum.tar.gz -C
/opt/ibm/ibm-ucd-p
yum makecache
.
. <many lines were removed>
.
Recipe: rabbitmq::default
  * service[rabbitmq-server] action restart
    - restart service service[rabbitmq-server]

Recipe: openstack-orchestration::engine
  * service[heat_engine] action restart
    - restart service service[heat_engine]

Recipe: openstack-orchestration::api
  * service[heat-api] action restart
    - restart service service[heat-api]

Recipe: openstack-orchestration::api-cfn
  * service[heat-api-cfn] action restart
    - restart service service[heat-api-cfn]

Recipe: openstack-orchestration::api-cloudwatch
  * service[heat-api-cloudwatch] action restart
    - restart service service[heat-api-cloudwatch]

Chef Client finished, 127 resources updated
UrbanCode Deploy with Patterns engine installation complete
```

8. Now, create a text file which will be used to create the databases in MySQL server. Example 4-208 shows the file used in the cloud prepared for this publication to create the MySQL databases.

Example 4-208 Creating the MySQL databases

```
[root@pw4303-l1 ~]# cat create_databases.sql
CREATE USER 'ibm_ucd'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE ibm_ucd;
CREATE DATABASE ibm_ucdp;
GRANT ALL ON ibm_ucd.* TO 'ibm_ucd'@'%'IDENTIFIED BY 'password' WITH GRANT OPTION;
GRANT ALL ON ibm_ucdp.* TO 'ibm_ucd'@'%'IDENTIFIED BY 'password' WITH GRANT OPTION;
```

9. Run the following command to read this file and create the databases:

```
# /usr/bin/mysql --user=root -ppassw0rd < create_databases.sql
```

Note: Do not replace -ppassw0rd for a different value. The MySQL admin password was set to passw0rd during the UrbanCode Deploy with Patterns Engine installation by a Chef recipe:

```
* service[mysql-start] action nothing (skipped due to action :nothing)
  * execute[/usr/bin/mysql_install_db] action run (skipped due to
only_if)
  * execute[assign-root-password] action run
    - execute /usr/bin/mysqladmin -u root password passw0rd
```

10. Extract the UrbanCode Deploy installation media:

```
# unzip URBANCODE_DEPLOY_V6.1.0.1_EN_EVAL.zip
```

The content of this file is extracted to the `ibm-ucd-install/` directory.

11. Copy the MySQL Java driver to the IBM UrbanCode Deploy installation directory:

```
# cp mysql-connector-java-5.1.30-bin.jar ibm-ucd-install/lib/ext/
```

12. Access the UrbanCode Deploy installation directory and execute the **`install-server.sh`** script:

```
# cd ibm-ucd-install/
# ./install-server.sh
```

13. The license agreement is displayed and you are prompted to accept it. Read and accept the license agreement to continue the installation process:

```
[echo] Do you accept the license? [y,n]
y
```

14. Just press Enter to accept the default installation path and the *JAVA_HOME* variable as shown in Example 4-209.

Example 4-209 Accepting installation location and JAVA_HOME in UCDP installation

```
[echo] Installing IBM UrbanCode Deploy version 6.1.0.1.532639
[echo] Enter the directory where the IBM UrbanCode Deploy should be installed.
[Default: /opt/ibm-ucd/server]

[echo] The specified directory does not exist. Do you want to create it? Y,n
[Default: Y]

[echo]
[echo] Installing IBM UrbanCode Deploy to: /opt/ibm-ucd/server
```

```
[echo] Please enter the home directory of the JRE/JDK used to run the server.
[Default: /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86_64/jre]

[echo] JVM Version detected: 1.7.0_45
[echo] JAVA_HOME: /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86_64/jre
```

15. If you have attended the requirements of hostname resolution and FQDN use, the installation process will continue and suggest you to use your fully qualified domain name to provide remote web access to the UrbanCode Deploy server. Just press Enter to accept the hostname and press Enter again to proceed by using secure connections, as shown in Example 4-210.

Example 4-210 Accepting the host name and secure connections

```
[echo] What host name will users access the Web UI at? [Default:
ucdp.itso.ibm.com]

[echo] Do you want the Web UI to always use secure connections using SSL? Y,n
[Default: Y]
```

16. To avoid port conflict between UrbanCode Deploy and UrbanCode Deploy with Patterns, change the HTTPS port from 8443 to 7443 and the HTTP port from 8080 to 7080, as shown in Example 4-211.

Example 4-211 Changing the HTTP and HTTPS ports for UrbanCode Deploy server

```
[echo] Enter the port on which the Web UI should listen for secure HTTPS requests.
[Default: 8443]
7443
[echo] Enter the port on which the Web UI should redirect unsecured HTTP requests
from. [Default: 8080]
7080
```

17. Accept the default values for the options shown in Example 4-212.

Example 4-212 Accepting default values during part of the UrbanCode Deploy installation

```
[echo] Enter the port to use for agent communication. [Default: 7918]

[echo] Do you want the Server and Agent communication to require mutual
authentication?  This requires a manual key exchange between the server and each
agent. See the documentation for more details. y,N [Default: N]

[echo] Enter the port and hostname of a Rational License Key Server containing
product licenses for IBM UrbanCode Deploy, in the form of port@hostname. (e.g.
27000@licenses.example.com) Alternatively, you may leave this blank to begin a
60-day evaluation period. [Default: none]

[echo] When installing a server as a part of an existing cluster or when using a
pre-populated database, it is not necessary to create the database schema. This
step must be performed when installing a stand-alone server to a fresh database or
when installing the first server in a cluster.
[echo] Create database schema? Y,n [Default: Y]
```

18. Use the `mysql` database type, accept the default value for the database driver, provide the database connection string, and accept the default values for the database user name and password, as shown in Example 4-213.

Example 4-213 Database properties during the UrbanCode Deploy installation process

```
[echo] The following database types are supported: derby, mysql, oracle, sqlserver,
postgres, db2.
[echo] Enter the database type to use. [Default: derby]
mysql
[echo] Enter the database driver. [Default: com.mysql.jdbc.Driver]

[echo] Enter the database connection string. Eg. jdbc:mysql://localhost:3306/ibm_ucd
jdbc:mysql://localhost:3306/ibm_ucd
[echo] Enter the database username. [Default: ibm_ucd]

[echo] Enter the database password. [Default: password]
```

19. The UrbanCode Deploy installation will proceed. The complete output is too long to show in this publication, but a partial output is shown in Example 4-214.

Example 4-214 UrbanCode Deploy installation output

```
[mkdir] Created dir: /opt/ibm-ucd/server/bin
[mkdir] Created dir: /opt/ibm-ucd/server/conf/server
[mkdir] Created dir: /opt/ibm-ucd/server/endorsed
[mkdir] Created dir: /opt/ibm-ucd/server/lib
[mkdir] Created dir: /opt/ibm-ucd/server/licenses
[mkdir] Created dir: /opt/ibm-ucd/server/native
[mkdir] Created dir: /opt/ibm-ucd/server/patches
[mkdir] Created dir: /opt/ibm-ucd/server/extensions
[mkdir] Created dir: /opt/ibm-ucd/server/var/log
[mkdir] Created dir: /opt/ibm-ucd/server/var/temp
[copy] Copying 1 file to /opt/ibm-ucd/server/conf
[sync] Copying 195 files to /opt/ibm-ucd/server/lib
.
. <many lines were removed>
.
[propertyfile] Updating property file:
/opt/ibm-ucd/server/conf/server/installed.properties
[copy] Copying 1 file to /opt/ibm-ucd/server
[propertyfile] Updating property file:
/opt/ibm-ucd/server/conf/server/installed.properties
[propertyfile] Updating property file:
/opt/ibm-ucd/server/conf/server/installed.properties
[propertyfile] Creating new property file:
/opt/ibm-ucd/server/conf/installed.version
[echo] After starting the server, you may access the web UI by pointing your
web-browser at
[echo] https://ucdp.itso.ibm.com:7443 to complete the Installation.
[echo] Installer Complete. (press return to exit installer)

[delete] Deleting directory /tmp/install-2411760977605916198.tmp

BUILD SUCCESSFUL
Total time: 3 minutes 52 seconds
```

20. Now **source** the /root/clientrc file in the UrbanCode Deploy server and run the **nova endpoints** command, as shown in Example 4-215.

Example 4-215 Using the nova endpoints command

```
[root@ucdp ibm-ucd-install]# source /root/clientrc
[root@ucdp ibm-ucd-install]# nova endpoints
+-------------+----------------------------------------------------------------+
| nova        | Value                                                          |
+-------------+----------------------------------------------------------------+
| adminURL    | http://172.16.21.137:8774/v2/78329f1240f9483b892f1dfd9b31a1f0 |
| id          | 3942df56eac34c5c997fb4cbd5d83921                               |
| internalURL | http://172.16.21.137:8774/v2/78329f1240f9483b892f1dfd9b31a1f0 |
| publicURL   | http://172.16.21.137:8774/v2/78329f1240f9483b892f1dfd9b31a1f0 |
| region      | RegionOne                                                      |
| serviceName | nova                                                           |
+-------------+----------------------------------------------------------------+
+-------------+----------------------------------+
| neutron     | Value                            |
+-------------+----------------------------------+
| adminURL    | http://172.16.21.137:9696        |
| id          | 17a6bb5e48794c35a5765a344dd10ffc |
| internalURL | http://172.16.21.137:9696        |
| publicURL   | http://172.16.21.137:9696        |
| region      | RegionOne                        |
+-------------+----------------------------------+
+-------------+----------------------------------+
| glance      | Value                            |
+-------------+----------------------------------+
| adminURL    | http://172.16.21.137:9292        |
| id          | afb0b713f8a3403e87cd2cd14d751348 |
| internalURL | http://172.16.21.137:9292        |
| publicURL   | http://172.16.21.137:9292        |
| region      | RegionOne                        |
+-------------+----------------------------------+
+-------------+----------------------------------+
| ceilometer  | Value                            |
+-------------+----------------------------------+
| adminURL    | http://172.16.21.137:8777        |
| id          | 2709787306cb4339b3da725de2f24727 |
| internalURL | http://172.16.21.137:8777        |
| publicURL   | http://172.16.21.137:8777        |
| region      | RegionOne                        |
+-------------+----------------------------------+
+-------------+----------------------------------+
| heat-cfn    | Value                            |
+-------------+----------------------------------+
| adminURL    | http://172.16.21.137:8000/v1     |
| id          | 6968e62204704a4593082f3e1ce5c08e |
| internalURL | http://172.16.21.137:8000/v1     |
| publicURL   | http://172.16.21.137:8000/v1     |
| region      | RegionOne                        |
+-------------+----------------------------------+
+-------------+----------------------------------------------------------------+
| cinder      | Value                                                          |
+-------------+----------------------------------------------------------------+
```

```
| adminURL    | http://172.16.21.137:8776/v1/78329f1240f9483b892f1dfd9b31a1f0 |
| id          | 5a8cb20a770c49258ac2410701b8091a                             |
| internalURL | http://172.16.21.137:8776/v1/78329f1240f9483b892f1dfd9b31a1f0 |
| publicURL   | http://172.16.21.137:8776/v1/78329f1240f9483b892f1dfd9b31a1f0 |
| region      | RegionOne                                                    |
+-------------+--------------------------------------------------------------+
+-------------+--------------------------------------------------------------+
| heat        | Value                                                        |
+-------------+--------------------------------------------------------------+
| adminURL    | http://172.16.21.137:8004/v1/78329f1240f9483b892f1dfd9b31a1f0 |
| id          | 0304fd6efba24b67ae24cb380a6124ad                             |
| internalURL | http://172.16.21.137:8004/v1/78329f1240f9483b892f1dfd9b31a1f0 |
| publicURL   | http://172.16.21.137:8004/v1/78329f1240f9483b892f1dfd9b31a1f0 |
| region      | RegionOne                                                    |
+-------------+--------------------------------------------------------------+
+-------------+----------------------------------+
| keystone    | Value                            |
+-------------+----------------------------------+
| adminURL    | http://172.16.21.137:35357/v2.0  |
| id          | f7c7d0a96cb144cca44d8b92ee93a295 |
| internalURL | http://172.16.21.137:35357/v2.0  |
| publicURL   | http://172.16.21.137:5000/v2.0   |
| region      | RegionOne                        |
+-------------+----------------------------------+
```

Note: The `OS_USERNAME` and `OS_PASSWORD` variables must match the Controller node admin user and password:

```
[root@pw4303-l1 ~]# cat /root/clientrc

export HEAT_URL=http://172.16.21.151:8004/v1/CHANGEME
export OS_NO_CLIENT_AUTH=True
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin
export OS_AUTH_URL=http://172.16.21.137:5000/v2.0
export OS_REGION_NAME=RegionOne
```

21. Look for the heat section of the **`nova endpoints`** output, and extract the authentication token from the `PublicURL`, which is highlighted in Example 4-216.

Example 4-216 Obtaining the heat authentication token

```
+-------------+--------------------------------------------------------------+
| heat        | Value                                                        |
+-------------+--------------------------------------------------------------+
| adminURL    | http://172.16.21.137:8004/v1/78329f1240f9483b892f1dfd9b31a1f0 |
| id          | 0304fd6efba24b67ae24cb380a6124ad                             |
| internalURL | http://172.16.21.137:8004/v1/78329f1240f9483b892f1dfd9b31a1f0 |
| publicURL   | http://172.16.21.137:8004/v1/78329f1240f9483b892f1dfd9b31a1f0 |
| region      | RegionOne                                                    |
+-------------+--------------------------------------------------------------+
```

You can also get this information by using the following command:

```
# nova endpoints | grep 8004 | grep publicURL | cut -f 5 -d "/" | awk '{print $1}'
```

22. Replace the `CHANGEME` word in the `/root/clientrc` file for this authentication token:

```
# sed -i "s/CHANGEME/<authentication_token>/g" /root/clientrc
```

Important: Use the authentication token obtained in step 21 in this **sed** command, as shown in the example below:

```
# sed -i "s/CHANGEME/78329f1240f9483b892f1dfd9b31a1f0/g" /root/clientrc

# cat /root/clientrc
export
HEAT_URL=http://172.16.21.151:8004/v1/78329f1240f9483b892f1dfd9b31a1f0
export OS_NO_CLIENT_AUTH=True
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin
export OS_AUTH_URL=http://172.16.21.137:5000/v2.0
export OS_REGION_NAME=RegionOne
```

23. Add the following entries to your `.bashrc` file and **source** this file and the modified `/root/clientrc`:

```
# echo "set -o vi" >> ~/.bashrc
# echo "export JAVA_HOME=/usr/lib/jvm/jre-1.7.0-openjdk.x86_64" >> ~/.bashrc
# source ~/.bashrc
# source /root/clientrc
```

24. Run the following command to start the UrbanCode Deploy server and wait two minutes before proceeding, to allow appropriate time for the daemon to start:

```
# /opt/ibm-ucd/server/bin/server start
# sleep 120
```

25. Create a new authentication token for use by the UrbanCode Deploy with Patterns web-designer using the steps from Example 4-217.

Example 4-217 Creating a new authentication token

```
# cd /root/ibm-ucd-install/overlay/opt/tomcat/webapps/ROOT/tools/
# unzip udclient.zip -d /tmp
Archive:  udclient.zip
   creating: /tmp/udclient/
  inflating: /tmp/udclient/udclient.jar
  inflating: /tmp/udclient/udclient
  inflating: /tmp/udclient/udclient.cmd
# cd /tmp/udclient/
# source ~/.bashrc
# source /root/clientrc
# /tmp/udclient/udclient --verbose -weburl https://172.16.21.151:7443 -username
admin -password admin createAuthToken -user admin -expireDate 12-31-2020-01:00
-description "Used by UCDP web designer application for authentication with this
UCD server"
{"token": "8f9951af-466e-4d31-9d20-91bce7b06034"}
```

Note: The information in bold is the output of the command udclient, providing the token created for the UrbanCode Deploy with Patterns web-designer.

26. Create a temporary file with the token value (only the token value, *do not include* the word “token”), as shown in Example 4-218.

Example 4-218 Creating a temporary file with the authentication token

```
# echo 8f9951af-466e-4d31-9d20-91bce7b06034 > /tmp/token
```

27. Now, extract the installation file for the UrbanCode Deploy with Patterns Web Engine and access the installation directory (files are extracted to `ibm-ucd-patterns-install/web-install/` directory), and then run the **`install.sh`** installation script:

```
# unzip ibm-ucd-patterns-web-6.1.0.1.545068.zip
# cd ibm-ucd-patterns-install/web-install/
# ./install.sh
```

28. The license agreement is displayed and you are prompted to accept it. Read and accept the license agreement to continue the installation process:

```
If you accept the license agreement, enter 'y' [ n ] : y
```

29. Accept the default value when prompted if you want to configure a connection to an IBM UrbanCode Deploy server:

```
Would you like to configure a connection to an IBM UrbanCode Deploy server? [ n ] :
```

30. Provide the authentication token created in step 25 on page 500:

```
Enter the authentication token [  ] : 8f9951af-466e-4d31-9d20-91bce7b06034
```

31. Press Enter to accept the default value for the URI for the cloud discovery service:

```
Enter the URI for the cloud discovery service [ http://localhost:7575 ] :
```

32. Provide the IP address of the UrbanCode Deploy server (this server), type `mysql` as the type of database to use and press Enter to accept the default value for the URL of the MySQL database:

```
Enter the public IP address or host name of this system [ ucdp ] : 172.16.21.151
Enter the type of database to use (supported types are derby, mysql) [ derby ] : mysql
Enter the URL of the MySQL database [ jdbc:mysql://localhost:3306/ibm_ucdp ] :
```

33. Provide the user name and password for the MySQL database:

```
Enter the user name for the MySQL database [ ibm_ucdp ] : ibm_ucd
Enter the password for the MySQL database [ ibm_ucdp ] : password
```

Important: You should not accept the default values for the user name and password in this step. The user name and password for the MySQL database was created by Chef during step 7 on page 494.

34. Press Enter to accept the default value for the driver to use to connect to the MySQL database:

```
Enter the driver to use to connect to the MySQL database [ com.mysql.jdbc.Driver ] :
```

35. Provide the IBM Rational® Common Licensing (RCL) server path:

```
Please enter the RCL server path [  ] : 27000@sus-nc1comlic.rtp.raleigh.ibm.com
```

Note: The RCL server is the license server for the UrbanCode Deploy with Patterns installation. For more information about this product and how to set up this server in your environment, see “Rational Common Licensing overview” in the IBM Knowledge Center:

http://ibm.co/1JacolW

36. The installation will proceed. The installation output is too long to be displayed in this publication, but a partial output is shown in Example 4-219.

Example 4-219 UrbanCode Deploy with Patterns web engine installation

```
detected RHEL 6
service qpidd stop
qpidd: unrecognized service
stopping qpidd done
Configuring chef...
mkdir -p /opt/ibm/ibm-ucd-p
tar xzf /root/Downloads/ibm-ucd-patterns-install/web-install/ibm-yum.tar.gz -C
/opt/ibm/ibm-ucd-p
yum makecache
.
. <many lines were removed>
.
Recipe: tomcat::default
  * service[tomcat] action restart
    - restart service service[tomcat]

  * execute[wait for tomcat] action run
    - execute sleep 5

Chef Client finished, 32 resources updated
UrbanCode Deploy with Patterns designer installation complete
```

37. Copy the MySQL connector driver to the tomcat6 library directory:

```
# cp mysql-connector-java-5.1.30-bin.jar /usr/share/tomcat6/lib/
```

38. Modify the `/var/lib/tomcat6/webapps/landscaper/WEB-INF/classes/config.properties` file and replace the `localhost:8443` entry for the `<ip_address>:8443` entry, where <ip_address> is the IP address of the UrbanCode Deploy with Patterns server (this server). You can use the **sed** command to replace the appropriate entry:

```
# sed -i "s/localhost:8443/172.16.21.151:7443/g"
/var/lib/tomcat6/webapps/landscaper/WEB-INF/classes/config.properties
```

Note: There is no line break in the command above.

39. Modify the `/var/lib/tomcat6/webapps/landscaper/WEB-INF/classes/config.properties` file and append the following lines after the `com.ibm.ucd-patterns.user_data.dir` entry:

```
com.ibm.landscaper.ucdeploy.username=admin
com.ibm.landscaper.ucdeploy.password=admin
```

You can use the following sed commands to append these lines in the appropriate place:

```
# sed -i "/com.ibm.ucd-patterns.user_data.dir/a
com.ibm.landscaper.ucdeploy.password=admin"
/var/lib/tomcat6/webapps/landscaper/WEB-INF/classes/config.properties

# sed -i "/com.ibm.ucd-patterns.user_data.dir/a
com.ibm.landscaper.ucdeploy.username=admin"
/var/lib/tomcat6/webapps/landscaper/WEB-INF/classes/config.properties
```

Note: There are no line breaks in the commands above.

The file /var/lib/tomcat6/webapps/landscaper/WEB-INF/classes/config.properties should be similar to Example 4-220 after these modifications.

Example 4-220 config.properties file from UrbanCode Deploy with Patterns after modification

```
[root@pw4303-l1 ~]# cat /var/lib/tomcat6/webapps/landscaper/WEB-INF/classes/config.properties
################################################################################
# Licensed Materials - Property of IBM Copyright IBM Corporation 2014. All Rights Reserved.
# U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP
# Schedule Contract with IBM Corp.
#
# Contributors:
#  IBM Corporation - initial API and implementation
################################################################################
# You must provide any specific packages that must be scanned for karyon for finding Application and
Component classes.
# By default karyon only searches com.netflix package
# The package specified here is the root package, any subpackages under the root will also be scanned.
com.netflix.karyon.server.base.packages=com.urbancode.landscape

com.netflix.karyon.server.bootstrap.class=com.urbancode.landscape.web.server.LandscaperBootstrap

# The health check handler for this application.
com.netflix.karyon.health.check.handler.classname=com.urbancode.landscape.web.health.LandscaperHealthCheck

# Comment this property if you need Eureka integration and populate eureka-client.properties with your
environment details.
com.netflix.karyon.eureka.disable=true

com.ibm.landscaper.ucdeploy.url=https://172.16.21.151:7443
com.ibm.landscaper.ucdeploy.token=8f9951af-466e-4d31-9d20-91bce7b06034

com.ibm.ucd-patterns.user_data.dir=/var/lib/ibm-ucd-patterns
com.ibm.landscaper.ucdeploy.username=admin
com.ibm.landscaper.ucdeploy.password=admin

hystrix.command.GetComponents.execution.isolation.thread.timeoutInMilliseconds=5000
hystrix.command.GetComponentVersions.execution.isolation.thread.timeoutInMilliseconds=5000
hystrix.command.GetComponentProcesses.execution.isolation.thread.timeoutInMilliseconds=5000
hystrix.command.GetApplications.execution.isolation.thread.timeoutInMilliseconds=5000

hystrix.command.GetAccessToken.execution.isolation.thread.timeoutInMilliseconds=5000
hystrix.command.OSGetImages.execution.isolation.thread.timeoutInMilliseconds=8000
hystrix.command.GetStorageVolume.execution.isolation.thread.timeoutInMilliseconds=8000
hystrix.command.GetNetworks.execution.isolation.thread.timeoutInMilliseconds=8000

hystrix.command.EC2GetImages.execution.isolation.thread.timeoutInMilliseconds=8000
hystrix.command.EC2GetStorageVolume.execution.isolation.thread.timeoutInMilliseconds=8000
hystrix.command.EC2GetNetworks.execution.isolation.thread.timeoutInMilliseconds=8000

hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds=8000
hystrix.command.ValidateBlueprint.execution.isolation.thread.timeoutInMilliseconds=180000
hystrix.command.FindAllEnvironments.execution.isolation.thread.timeoutInMilliseconds=180000

hystrix.command.LoadStackUsageCommand.execution.isolation.thread.timeoutInMilliseconds=180000
hystrix.command.LicenseManageCommand.execution.isolation.thread.timeoutInMilliseconds=180000
hystrix.command.LoadTenantUsageCommand.execution.isolation.thread.timeoutInMilliseconds=180000
hystrix.command.RCLTokenCheckoutCommand.execution.isolation.thread.timeoutInMilliseconds=180000
```

40. Make sure that the `/etc/ssh/sshd_config` file has `PasswordAuthentication` set to `yes`:

```
# grep PasswordAuthentication /etc/ssh/sshd_config
PasswordAuthentication yes
```

41. Give at least 30 seconds to allow the installation to complete before executing the next step:

```
# sleep 30
```

42. Restart the SSH and tomcat6 services:

```
# /sbin/service tomcat6 restart
# /sbin/service sshd restart
```

43. Stop the Linux firewall to allow the web interfaces to work:

```
# /sbin/service iptables stop
# chkconfig iptables off
```

44. Now that the installation is completed, you can log in to the UrbanCode Deploy web interface at this address:

    `https://<ucdp_server_ip_address>:7443/`

    Use the `admin` user name and `admin` password to log in to the UrbanCode Deploy web interface as shown in Figure 4-211.

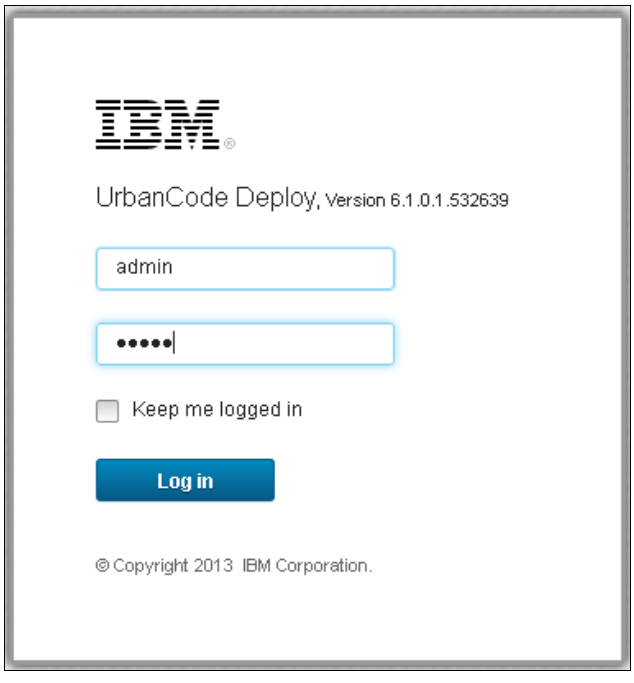


Figure 4-211 UrbanCode Deploy login interface

You can also log in to the UrbanCode Deploy with Patterns web interface using the following URL:

`http://<ucdp_server_ip_address>:8080/landscaper`

Use the `ucdpadmin` user name and `ucdpadmin` password to log in to the UrbanCode Deploy with Patterns web interface, as shown in Figure 4-212.

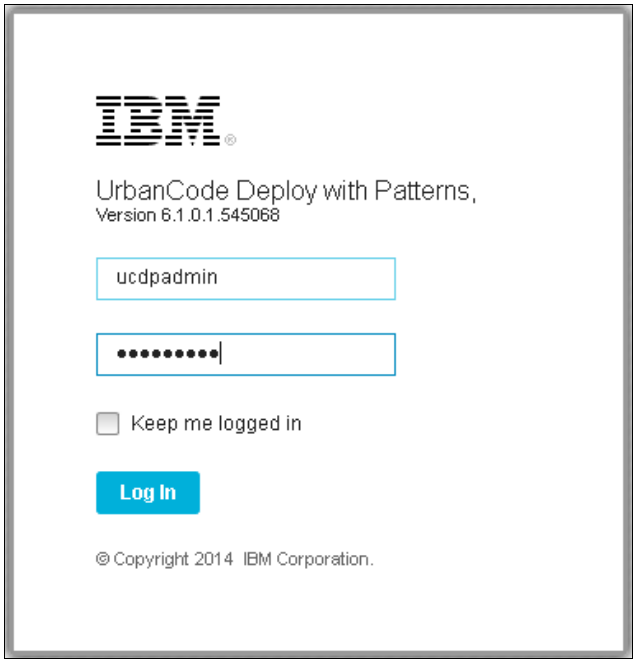


Figure 4-212 UrbanCode Deploy with Patterns login interface

For more information about connecting the UrbanCode Deploy and UrbanCode Deploy with Patterns applications to your OpenStack cloud and how to use these applications, see Chapter 7, “Cloud user deployment and use scenarios” on page 563.

5

IBM big data and analytics capabilities

Big data requires a new view on business intelligence, data management, governance, and delivery. The following topics in this chapter explain how IBM addresses those concerns:

- 5.1, “Logical architecture” on page 508
- 5.2, “IBM big data and analytics portfolio” on page 509
- 5.3, “Big data use cases” on page 523

© Copyright IBM Corp. 2015. All rights reserved.

5.1 Logical architecture

The big data and analytics reference architecture logical view, shown in Figure 5-1, depicts a set of capabilities that a business must consider as they enter the big data space, which is explained below. It includes capabilities around data integration, management, security, and analytics.

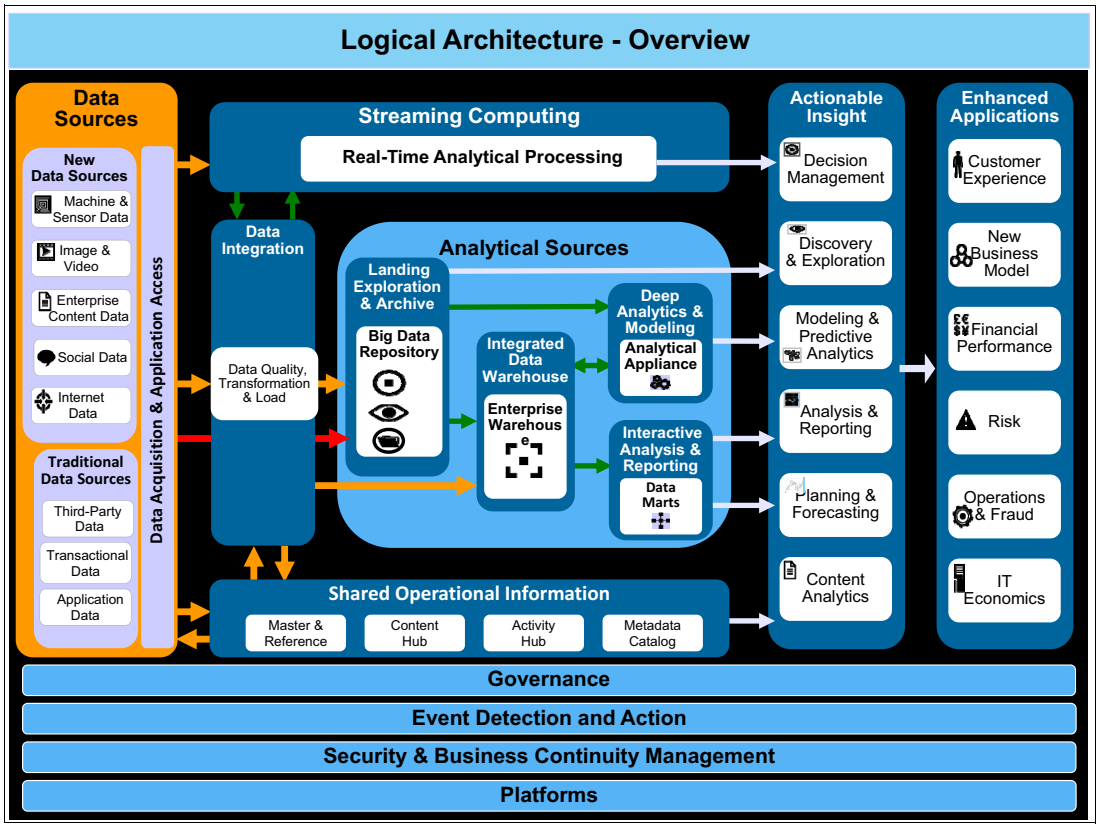


Figure 5-1 Big data and analytics capability logical architecture overview

5.2 IBM big data and analytics portfolio

Figure 5-2 illustrates a high-level view of the IBM big data and analytics portfolio.

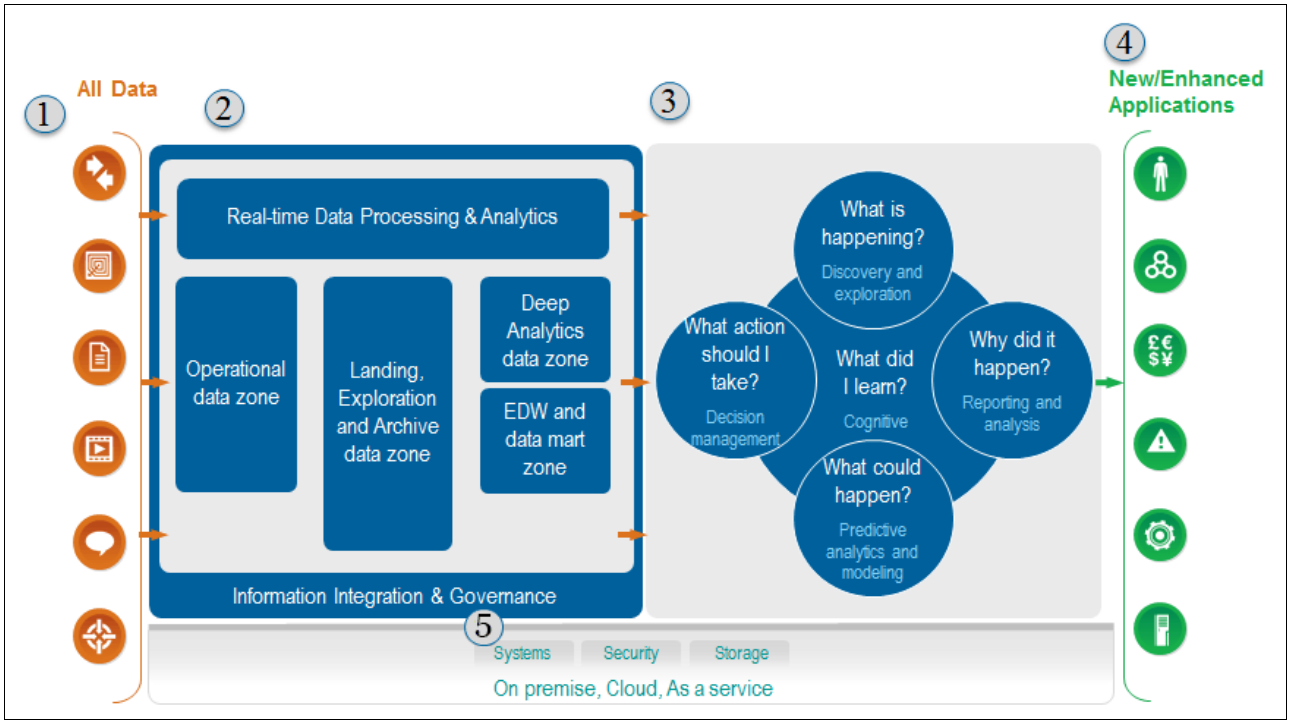


Figure 5-2 IBM big data and analytics portfolio

The following keys provide explanations of the IBM big data and analytics capability logical architecture[1] shown in Figure 5-1 on page 508 to support various workloads. The numbers cited in the subheads refer to annotations in the IBM big data and analytics portfolio illustrated in Figure 5-2.

5.2.1 Data sources (1)

IBM offers many different solutions based upon a client's needs and requirements in terms of the business strategy and drivers to make the optimal use of structured and unstructured data for extraction, ingestion, transformation, and loading. The availability of new sources of data opens up opportunities for gaining deeper, more accurate insights on operations, customers, weather, traffic, and so on.

This component of the architecture highlights the importance of acquiring both new data sources and traditional data sources. A truly cohesive solution considers traditional data as well as this emerging data, often with an objective of having each enhance the understanding of the other. The data sources can be processed in batch or real-time mode. The data volume and data generation speed are significantly higher than it has been before.

New sources of data open opportunities for gaining deeper, more accurate insights on operations, customers, weather, traffic, and so on. The data sources can be processed in batch or real-time mode. The data volume and data generation speed are significantly higher than before.

[1] See *IBM Big Data and Analytics Reference Architecture: Architecture Overview*, Copyright 2014, IBM Corp.

Traditional sources

These represent the data that is available in the transactional systems, typically stored in relational databases and flat file formats. The information contained in these data sources is mostly structured. Examples of traditional data sources are credit card transactions and sales transactions. An example of third-party data is the census information that contains demographic information that can be used to enhance customer profiles.

New sources

In today's world, to use the power of analytics fully within an organization, there is a need to use any kind of data that is available to the business. About 80% of this data is unstructured. This unstructured data is obtained from many new sources, such as video, images, documents, free format text files, web logs, social media data, machine data, and climate data. There are also other varieties of data available, from sensor devices and smart meters to click streams. All of these new sources of data require new technologies to store, process, and make sense of it.

As explained earlier, big data is typically identified as having one or more of the following attributes: volume, variety, velocity, and veracity. Types of source data can be broken into the categories in Table 5-1, each of which may demonstrate any of the Four Vs.

Table 5-1 Categories that demonstrate the Four Vs

Data type	Description	Examples
Machine and sensor data	An increasingly sensor-enabled, instrumented, and connected world generates huge volumes of data with machine speed characteristics. Machine and sensor data covers all data generated by machines.	► Radio frequency identification (RFID) ► Data generated from servers, applications, networks, and so on
Image and video	Digital images and videos.	► Security surveillance cameras ► Smartphones ► Web images and videos
Enterprise content	An organization's documents and other content that relate to the organization's processes.	► Documents ► Forms ► Checks
Transaction and application data	Typically, structured data that describes an event. This is generally recognized as a traditional source.	► Point of sale transaction data ► Data entered by user via a web form ► Data entered into a CRM application
Social data	An expression of social media (user-generated content on the Internet) in a computer-readable format that also includes metadata that provides context (location, engagement, links). Focused strictly on publicly available data.	► Twitter ► Facebook ► YouTube ► Tumble ► Disqus
Third-party data	Data obtained under license from third-party organizations. Typically structured and recognized as a traditional source.	► Experian ► Dun and Bradstreet

Organizations that use these varied sources of big data can accomplish several key things:

- Uncover new insights and understand why certain events happen.
- Predict based in previous learning what is likely to happen and get recommendations for the next best action.
- Make analytics pervasive across the enterprise and empower many more employees to make better, informed decisions and make the entire business more effective and competitive.
- Proactively identify, understand, and manage financial and operational risks to make more confident and informed decisions and protect against security and privacy violations.

5.2.2 Analytical sources (2)

The analytical sources are the data repositories that store information to be analyzed. In the big data and analytics reference architecture, these sources are used by various technologies for many forms of business analytics.

Data can be separated into different zones for exploration, analysis, and archived.

Real-time data processing and analytics

Applying analytics to data in motion transforms burgeoning data into actionable information and business insights. Streaming computing, a critical component of the big data and analytics reference architecture, performs in-motion analytics on a wide variety of data at unprecedented volumes and speeds. This component ingests data at varying volumes and velocities, classifies and filters that data, applies appropriate analytics, and produces insights immediately so that correct actions may be taken (either by a person or automatically). In this way, real-time analytical processing works in coordination with the broader big data and analytics architecture. It acts as a method of ingesting in motion data to persist in the environment and to apply analytics in the moment for immediate results.

There are many applications of real-time analytical processing (RTAP) for many industries. These are just few examples to illustrate the benefits of real-time processing and analytics:

- Alerts

  The RTAP application notifies the users that the analysis has identified a situation (based on a set of rules or process models) has occurred and provides options and recommendations for appropriate actions. Alerts are useful in situations where the application should not automatically modify the process or automatically take action. They are also effective in situations where the action to be taken is outside the scope of influence of the RTAP application.

  Example: A patient monitoring application would alert a nurse to take a particular action, such as administering additional medicine.

- Feedback applications
  - The RTAP application identifies that a situation (based on a set of rules or process models) has occurred and makes the appropriate modifications to the processes to prevent further problems or to correct the problems that have already occurred.
  - Feedback analysis is useful in manufacturing scenarios where the application determines that defective items have been produced and takes action to modify components to prevent further defects.

Example: A manufacturer of plastic containers might run an application that uses the data from sensors on the production line to check the quality of the items through the manufacturing cycle. If defective items are sensed, the application generates instructions to the blending devices to adjust the ingredients to prevent further defects from occurring.

- Detecting system or application failures:
  - The RTAP application is designed to notice when a data source does not respond or generate data in a prescribed time.
  - Failure detection is useful in determining system failure in remote locations or problems in communication networks.
  - *Example:* An administrator for a critical communications network deploys an application to test continuously that the network is delivering an adequate response time. When the application determines that the speed drops below certain level or is not responding at all, it alerts the administrator.

Note: IBM InfoSphere Streams software processes petabytes of data in a highly parallel and cost-effective manner and enables the development and execution of applications that process information in motion. It enables continuous, fast analysis of massive volumes of moving data to help improve the speed of business insight and decision making. It also provides integration with IBM SPSS products to enable predictive models to be deployed in real time on a parallel framework for fast scoring of transactions for instant actions.

Operational data zone, shared operational information

The s*hared operational information* consists of a set of capabilities to managed operational information that is shared by many systems. Application independent operational data is stored in managed repositories that serve to integrate operational and analytical processes via a series of specialized services. This includes master and reference data, content hub, activity hub, and metadata catalog.

Master and reference data

Master data is key business data elements that may include information about customers, products, employees, suppliers, and vendors. This is shared as a single source of basic business data across systems, applications, and processes**.** Master Data enables organizations to develop applications that incorporate trusted data into an enhanced 360-degree view of the customer to become more responsive to their needs. the IBM Master Data component provides real-time matching of master records at big data volumes in Hadoop.

Reference data is the data that defines the standard data domain values used within an organization. Examples of Reference Data are units of measure, country codes, corporate codes, conversion rates (currency, weight, temperature, and so on) and calendar dates.

The Master and Reference Data component is applied to different industry use cases as shown in the following examples:

- Government: Enable government organizations to create a single view of citizens to improve citizen service, verify benefits and prevent and detect threats and fraud.
- Banking: Enable banking institutions to increase wallet share, provide better customer service and reduce costs through operational efficiencies.
- Retail: Help to better understand the full customer relationship and introduce new products quicker.

- Insurance: Enable organizations to shift from policy-centric views to a customer-centric view.
- Healthcare: Enable healthcare organizations to connect disparate pieces of patient, provider and member information into a single, actionable view.
- Telecommunications: Enable telco organizations to connect disparate pieces of customer information across lines of business, service locations, billing systems and more.

Content hub

The Content Hub is the processes and technologies to collect, manage, and publish any format of digital content and enable content analytics. The digital content may take the form of text (such as electronic documents), multimedia files (such as audio or video files), or any other file type requiring a content management lifecycle. Content analytics uses sophisticated natural language processing to enable the organization to understand the meaning and context of human language within textual information found in sources such as documents, reports, email, and web content.

Activity hub

The Activity Data Hub manages consolidated information about recent activity, including analytical decisions that are related to the entities in the master data and content hubs. This type of data is normally managed by applications. However, because the organization is processing information about activity from outside of the scope of its applications, a new type of hub is needed to manage the dynamic nature of the information. Activity Hub sources can range from social media interaction, customer interaction from multiple channels, and information from the integrated data warehouse and exploration repositories. This information is used to enhance customer views.

Metadata catalog

In the big data and analytics reference architecture unstructured data has moved from being on the sideline to being front and center. Unstructured data should not be confused with a lack of metadata. All data has metadata (data that describes it). Metadata and lineage remain central to big data governance and quality assurance. Note that the reference architecture supports having a landing area for data in its raw form. This provides the opportunity to be able to track and show data lineage back to its original form.

Metadata can be broken down into two forms: business and technical. Technical metadata describes the tables and columns, the files, the message types, and so on. Business metadata provides a reference for terminology. The two, of course, can be linked to one another in a business glossary. The metadata catalog contains the semantic definitions for business and IT terms, data models, types, and repositories. It provides functions to browse, discovery, and search metadata assets.

The InfoSphere Master Data Management product family enables powerful MDM solution with collaborative and operational capabilities. InfoSphere Metadata Workbench provides metadata management and lineage.

Landing, exploration, and archive data zones

Landing, exploration, and archive zones support data provisioning and analytics. These three capabilities should be considered separately. However, they are clustered together because they share some similar characteristics and system requirements. This makes them candidates for residing in a shared services multitenant environment, such as Hadoop.

They include these functions and characteristics:

- Raw data
- Structured and unstructured
- Pre-discovery
- Search and indexing
- Text analytics
- Self-provisioning source
- Discovery

In all cases, a cost effective and multi-processing platform is ideal for handling both variety and volume of data. The reference architecture uses Hadoop to capture operational data, use additional data types and enable exploration of data before normalization.

Landing zone

The Landing zone refers to a landing or provisioning area to capture and store the raw data for further processing. In the case of big data and analytics this landing area may support data that is not yet modeled and unstructured. A key consideration is to use a cost efficient platform. Data landed in its original raw state format is used both for a historical record and to support the exploration activities described below.

The open source Hadoop platform and commodity hardware is a preferred choice of technology for the Landing zone. Hadoop can scale as needed and offers a cost effective solution to store and process large amounts of structure and unstructured data.

A Hadoop platform provides a mechanism for storage and retrieval of data that is not necessary modeled in traditional tabular relations as used in relational databases. This facilitates rapid ingestion of the data into a Hadoop File System (HDFS) before a schema is defined. The Landing zone simplifies the ability to collect any kind of data (structured and unstructured) and land it into a Hadoop system for further investigation and processing. After the data is loaded, a schema can be applied to provide SQL-like capabilities and allow data interaction and exploration by using traditional BI and reporting tools.

Hadoop is based on massively parallel processing (MPP) architecture. It is possible to scale a Hadoop cluster to hundreds and even thousands of nodes. Data in a Hadoop cluster is broken down into smaller pieces called blocks and distributed throughout the cluster. In this way, the map and reduce functions can be executed concurrently on smaller subsets of the large data sets. This provides the scalability and speed needed for big data processing. In addition to speed and scalability, Hadoop also brings other benefits such as high availability and data redundancy. When combined with a SDE-based environment, further optimization can occur to prevent adding servers in terms of horizontal scalability.

Exploration zone

The exploration repository is a key component for delivering big data information and insights. The Exploration zone refers to the component supporting exploratory querying and analytics of detailed data (not aggregated), often in raw form. Although the Exploration zone may have a SQL-based interface, it is not restricted to structured data. Activities supported by this component include clustering, segmentation, forecasting, predictive analytics, and other analytics that require large amounts of data. Typical Exploration zone users include business analytic users and data scientists.

The Exploration zone provides more flexibility and agility so the users quickly get insights about new sets of information previously unavailable. New big data and analytics tools enable users to access structured and unstructured data for exploratory analytics to identify interesting correlations and insights. Exploration is an iterative process with multiples passes through the data. Specialized tools such as R, SAS, SPSS, and other discovery tools are used. The data in this repository may include replicated data pulled from operational systems, third party, and public sources.

Exploration implies that the answer or the question is not typically known. Therefore, a modeled environment is ill-suited for these activities. Rather, a *schema-less* environment enables users to access data in its original form quickly, manipulate it in various ways, and use all available data. Exploratory analysis directly executes the processing in this zone, without duplicating and moving the data elsewhere.

In some organizations, the amount of data in the exploration repository can be bigger than the data stored in the enterprise warehouse. The scale and flexibility of the Hadoop platform is ideal for the Exploration zone and fulfills the needs of the data scientists to perform exploratory analytics on detailed raw data and multiple years of historical data.

The exploration zone includes these functions and characteristics:

- Discovery
- Sandboxes
- Forecasting and predictive modeling
- Deep analysis

Archive zone

Archiving “cold” data from data warehousing environments is still necessary as way of reducing warehouse costs and improving performance. Although this cold data may be of no interest for operational reporting or business intelligence, it is increasingly of relevance to users performing exploratory or deep analytics. Taking advantage of a cost effective Hadoop infrastructure within the Archive zone to store the cold historic data provides data for deeper analytics.

The Hadoop component for the landing, exploration and archive repositories is offered in both IBM InfoSphere BigInsights and IBM PureData System for Hadoop. These products provide a complete set of enterprise capabilities such as enhanced security, text analytics and high availablity. These capabilities enable users to extract value from unstructured data sets and use the contents for analytics.

Deep Analytics data zone

Deep analytics supports meaningful analysis of data to drive more effective decision making. As opposed to Exploration zone where the question is unknown, this component supports the need to answer a known question. Thus, it requires a highly tuned environment that can run the necessary analysis and queries to support that question or set of questions. From a modeling perspective, SPSS, SAS, or R. are able to run or fine-tune models based on the appropriate data, and to do it in a cost effective manner.

The deep analytics and modeling process needs to use all data available at the detail level. The Data Warehouse contains the trusted data, but not necessary all the data to support deep analytics and predictive modeling processes. The advanced users and data scientist need to quickly explore and investigate new types of data to enhance their analysis. With this component of the architecture, advanced users can easily use both their traditional tools along with the new big data tools such as text analytics and explore new types of data (structured and unstructured) to identify new insights and to develop predictive models.

Rather than attempting to have a single enterprise level warehouse support all manner of reporting and analytics, organizations are now realizing that by using fit-for-purpose modeled environments they can realize far better cost savings and performance tuning. Although it may seem that introducing additional appliances to an environment will require additional maintenance and complexity, the opposite is true.

There are purpose-built analytical appliances such as IBM PureData for Analytics designed to deliver extremely fast performance and perform deep queries against large data sets, and these appliances can be integrated into the Software Defined Environment architecture. Advanced users can use their data mining and statistical tools to push their workloads into these appliances and drastically increase their productivity.

Enterprise data warehouse and data mart zone

This zone includes the trusted data repository and the interactive analysis and reporting repository, or *data mart*.

- The trusted data repository

  The enterprise data warehouse continues to have an important function in the reference architecture for big data and analytics. Trusted data is stored in this traditional enterprise data warehouse. Data for this repository is modeled to support interactive business intelligence activities. Warehouse data is normalized, matched, cleansed and validated. This repository typically requires high availability and disaster recovery. It is also the most expensive repository.

  The EDW keeps detailed data for the most current months and aggregates yearly data as opposed to maintaining years of detailed data in raw form. It has the following characteristics:

  - Structured
  - Validated (trusted)
  - Consolidated
  - Aggregated
  - Historical

- The interactive analysis and reporting repository (data mart)

  The interactive analysis and reporting repository supports business and interactive users. Driving capabilities here include performance, analytics at the speed of thought and availability for a broad spectrum of users and workloads. The data mart is a subset of the data warehouse that is usually oriented to a specific business line or team. It enables users to create reports quickly or to use predefined reports that use trusted data and promotes better decision making.

  Analytical and reporting repositories provide the best class of analytics services to the information users. Data is stored in star or snowflake schemas or a combination of both and is optimized for performance based on application SLA requirements. The goal is to use the best class of technology such as appliances and in-memory databases to provide flexibility so that data can be rapidly prepared to meet needs of a wide range of applications. For instance, an application that requires extremely high performance with extremely high level of concurrency can use in-memory data store.

  Note: An example of in-memory database is IBM DB2 BLU Acceleration. It enables faster, less complex analytic queries and reporting using dynamic in-memory columnar technologies for business intelligence and data warehouse applications.

Information integration and governance

The IBM PureData for Analytics and DB2 Analytics Accelerator for z/OS solutions integrate hardware and software capabilities specifically designed and tuned for high-performance analytics.

Information integration

Data integration is the combination of technical and business processes used to combine data from disparate sources into meaningful, valuable information. As in any architecture with multiple systems, data must move from place to place. How this is performed depends on service level agreements, acceptable latency and analytical data requirements. Information integration may involve changed data capture, batch ETL, federation and real-time data services.

A complete data integration solution encompasses discovery, cleansing, monitoring, transforming, and delivering data from a variety of sources. Integration of trusted traditional data coupled with unstructured and semistructured new data sources improves the success rate of critical projects and key analytical initiatives. Organizations of all sizes need data integration capabilities that fully support their business requirements. They need to integrate data wherever it resides, whether on the system of record or within big data sources, while applying governance and data-quality best practices.

Note: IBM provides a full set of products and capabilities to enable organizations to integrate big data information, including these IBM InfoSphere products:

- IBM InfoSphere Information Server
- IBM InfoSphere Federation Server
- IBM InfoSphere Data Replication

Information governance

Governance is the capability to ensure that regulations and policies are defined, enabled, and followed. The concepts of information governance apply as much within a big data landscape as in a more traditional one. However, the challenges are different and, thus, the implementation is often different. The Four Vs (volume, velocity, variety, and veracity) of big data affect how integration and governance can be performed, and what are realistic outputs and expectations.

Governance includes corporate governance, information governance, and information technology governance:

- *Corporate governance* defines the principles and guidelines by which corporations are directed and controlled. It is the mechanism for monitoring the actions, policies and decisions of corporations. The information assets need to be governed and protected to ensure that all principles of the corporate governance are followed.
- *Information governance* consists of series of disciplines, policies, procedures, processes and controls that are implemented to manage and protect information at the enterprise level. It includes regulatory and legal requirements, data privacy aspects of the information and the risk associated with protecting information.
- *Information technology (IT) governance* is a subset of the corporate governance that consists of managing all performance and risk associated with the IT assets and the impact on the entire organization. An example of this would be a disaster and recovery contingency plan.

Note: The IBM Business Information Exchange products provide a full set of capabilities to help organizations to implement information governance solutions. These are the key components:

- InfoSphere Blueprint Director: Creating blueprints and using best practices
- InfoSphere Discovery: Data discovery
- InfoSphere Metadata Workbench: Metadata management and lineage
- InfoSphere Data Architect: Logical and physical data modeling
- InfoSphere Business Glossary: Business glossary and workflow

5.2.3 IBM actionable insights (3)

In a world of rapidly growing information, key decision makers require an environment where they can discover, analyze, predict and derive timely, actionable insight from a wide domain of continuously changing structured, unstructured, streaming, or static data. As more and more data is collected from the marketplace, from new data sources and from every part of the organization it is of utmost importance to embed analytics and actionable insights into daily transactions to drive better business outcomes. In the section that follows, we explain how the IBM big data and analtyics portfolio supports insights that are actionable.

Discovery and exploration: What is happening?

Discovery and exploration answers the question: “What is happening?” The Discovery and Exploration component interactively analyzes, explores and discovers insights within data available in any repository, regardless of the format and technology being used to store and manage the data. This capability provides insights into areas that were previously untapped by an organization. In big data and analytics, this activity consists of enabling users with tools and capabilities to manipulate data in new ways, to access any and all the data (structured and unstructured) available to the organization and to perform exploratory analytics and discovery of new insights.

A key part of actionable insight is finding the information when it is needed. Discovery and exploration with intelligent search capabilities discovers information that resides in big data repositories or isolated systems and delivers it to enable business users to make insightful, fact-based decisions.

Note: The IBM solutions for Discovery and Exploration include these components:

- IBM Watson™ Explorer
- IBM SPSS Analytic Catalyst
- IBM Watson Analytics

Decision management: What action should I take?

Decision management answers the question: “What action should I take?” Decision Management is a component that enables organizations to automate, optimize, and govern repeatable business decisions based on insight gleamed from data. Decision Management automates and optimizes transactional decisions by combining predictive analytics, rules and scoring to deliver recommended actions in real time. This component enables the integration of predictive analytics and business rules into an organization’s processes to optimize and automate high-volume decisions at the point of impact.

Organizations manage their interactions with customers, employees and suppliers with information-based decision systems. The decisions are based on analysis of historical behavioral data, insights from prior decisions, and analysis of outcomes. Decision Management improves business decision agility while reducing cost and improving speed.

Note: These are among the IBM offerings for decision management:

- IBM Decision Optimization Center
- IBM SPSS Modeler Gold
- IBM ILOG CPLEX® Optimizer
- IBM Operational Decision Manager

Predictive analytics and modeling: What could happen?

Predictive analytics and modeling answer the question “What could happen?” Predictive analytics is a discipline that uses advanced analytical algorithms such as Linear Regression or Decision Tree to process current and historical data and create models that can make predictions about future outcomes. This enables organizations to act proactively rather than reactively, make better decisions, solve issues, and foster better outcomes.

Predictive analytic and modeling tools use deep analytics and exploratory zones to create models that can then be applied either for batch reporting or in a real-time process to enable organizations to act on their insights. Predictive analytics and modeling are typically performed by data scientists and statisticians. Data mining and statistical tools such as these are used to build predictive models:

- IBM SPSS products, which address the entire analytical process, from planning, to data collection, to analysis, reporting, and deployment
- Open Source R
- SAS

Note: The IBM solutions for modeling and predictive analytics include these products:

- IBM SPSS Statistics

  http://www.ibm.com/software/analytics/spss/products/statistics/
- IBM SPSS Modeler

  http://www.ibm.com/software/analytics/spss/products/modeler/

Reporting and analysis: Why did it happen?

Reporting and analysis answers the question: “Why did it happen?” Reporting and analytics are still applicable within a big data and analytics architecture. Reporting provides a visualization of analyzed information in a form that enables business users to understand and act. The capabilities in this component include pre-defined batch reports delivered on an hourly, daily, weekly, monthly, quarterly, or yearly basis. The reports can also be interactive reports and dashboards that enable users to view metrics based on currently available data.

Additionally, the component includes ad hoc reporting capabilities able to support concurrent user requests with little latency to ensure that they are getting accurate and timely information.

Business users use business intelligence tools to interact with the data and execute queries to answer predefined business questions. The result of these queries may be displayed in visualization tools in tabular or graphical format.

Note: The IBM solutions for analysis and reporting include these applications:

- IBM Cognos® Business Intelligence
- IBM Cognos Express
- IBM Business Intelligence Pattern
- IBM Business Intelligence Pattern with BLU Acceleration

Cognitive capabilities: What did I learn?

The Content and Planning and Forecasting components answer the questions “What did I learn?” and “How can I use this information to better prepare for the future?”

Content analytics

Analysis of textual content enables businesses to anticipate, shape and transform business outcomes by surfacing new actionable insights through the identification of trends, patterns, and relationships in enterprise content. The Content analytics component uses sophisticated natural language processing that enables the organization to understand the meaning and context of human language within textual information. Content analytics enables accurate and consistent categorization of unstructured information through the application of sophisticated text analytics techniques, as well as content-specific rules, to extract the meaning buried within content.

Note: IBM solutions for content analytics include these examples:

- IBM Watson Content Analytics
- IBM Social Media Analytics
- IBM SPSS Data Collection

Planning and forecasting

The Planning and Forecasting component can transform the entire planning cycle, from target setting and budgeting to reporting, creating score cards, analysis, and forecasting. It uses historical data to determine future trends. Integrated scorecards and strategy management capabilities help monitor performance metrics and align resources and initiatives with corporate objectives and market events.

Forecasting is used by organizations to determine how to allocate their budgets for an upcoming period. Forecasting analysis helps determine if events affecting a company, such as sales expectations, will increase or decrease the price of shares in that company. Forecasting provides a benchmark for organizations that have a long-term perspective of operations.

Note: The IBM solutions for Planning and Forecasting include these products:

- IBM Cognos TM1®
- IBM Cognos Insight™
- IBM Concert™
- IBM Cognos Express

5.2.4 New and enhanced applications (4)

The IBM big data and analytics portfolio also provides new applications and results in enhanced applications for the following business needs:

- Customer experience
- New business models
- Financial performance
- Risk
- Operations and fraud
- IT economics

5.2.5 On-premises or hybrid clouds and as-a-service capabilities (5)

These are enhanced through SDE with OpenStack and the additional capabilities described in the subsections that follow.

Event detection and action

The Event Detection and Action component triggers business processes and other activity when significant events are detected in the applications, analytic sources, and information interaction solutions. Automated actions can be implemented based on insights uncovered by real-time analytical processing.

Security and business continuity management

Security and business continuity management are processes and policies that are put in place to ensure that the access to the information is only for authorized purposes and that there is the ability to recover from unforeseen circumstances.

Security management

An important aspect to be considered when designing solutions for big data and analytics is the security and privacy of the data assets. Within a big data solution, many aspects of data privacy and government regulation need special considerations. For many organizations, losing the control of data assets could put their business at risk. In addition, there are industry standards that force organizations to be compliant with data privacy rules.

Inherent in the various capabilities shown in this architecture overview are various levels of security and privacy requirements. These are based on access types, data content, threat of risk and other factors. Security management may be controlled through user IDs and monitored through automated systems. The logs and activity records generated from the monitoring activity can be a source for operational security analytics. As in any system that may be shared among various lines of business, projects, and people it is important to ensure that appropriate security and privacy measures are followed.

Security management consists of implementation of policies and procedures to protect the information management assets and avoid unauthorized access. There are two key aspects of security:

- *Encryption*, which is a set of services and tools that encrypt information, based on corporate and external regulations
- *Access control*, which are the services and tools needed to control access to the information assets.

Note: The IBM InfoSphere Data Privacy for Hadoop and InfoSphere Data Privacy Product family provides a full set of capabilities to help organizations to implement security for the entire big data and analytics platform.

Business and continuity management

Business continuity management focuses on maintaining business continuity and recovering critical business applications no matter the circumstances. It consists of all the activities related to the continuous operation of critical business functions. This includes disaster recovery with the ability to recover to an operational state within a reasonably short period.

IBM provides services, reference architecture, business and technology solutions, along with best practices for business and continuity management, including high availability and disaster recovery optimized for the demands of critical big data and analytics:

- *Platforms* are the different approaches to managing the infrastructure to support the components for the big data and analytics solution. There are four common approaches; however, a customer might decide to deploy their big data Solutions using a combination of two or more:
- *Custom hardware solutions* are infrastructure platforms purchased and installed independently by a customer. (For example, our SDE solution fits here.)
- *Appliances* are infrastructure purchased with software installed, ready to use.
- *Private cloud* is a shared infrastructure for the organization's sole use. (Our SDE solution fits here.)
- *Public cloud* is an infrastructure and software available on a pay-as-you-use basis and usually multitenant. (Our SDE solution can also fit here.)

Note: See the following IBM service offerings:

- IBM Global Technology Services: Resiliency Services

  http://www.ibm.com/services/us/en/it-services/business-continuity/
- IBM Lab Services and Training: High Availability Center of Competency

  http://www.ibm.com/systems/services/labservices/solutions/hacoc.html

For more information about the BAO reference architecture and solutions, read this IBM Redpaper publication:

Smarter Analytics: Making Better Decisions Faster with IBM Business Analytics and Optimization Solutions, REDP-4886

http://www.redbooks.ibm.com/abstracts/redp4886.html

5.3 Big data use cases

Because much of the big data activity in the market in the past has focused on learning about big data technologies, vendors haven't made a concerted effort to help organizations understand which problems big data can address. IBM has been the exception.

Through conducting surveys, studying analyst findings, talking with more than 300 customers and prospects and implementing hundreds of big data solutions, IBM has identified two top high-value use cases that can be your first step into big data.

5.3.1 Big data exploration use case

Find, visualize, understand all big data to improve decision making. Big data exploration addresses the challenge that every large organization faces: Information is stored in many different systems and silos and people need access to that data to do their day-to-day work and make important decisions. This is one of the use cases that the MyBank developers use big data for in 1.12.1, "Enable line-of-business cloud users" on page 39.

This use case addresses the need to explore and mine big data to find what is interesting and relevant to the business for better decision making. These requirements are included:

- Exploring new data sources for potential value
- Mining for what is relevant for a business imperative
- Assessing the business value of unstructured content
- Uncovering patterns with visualization and algorithms
- Preventing exposure of sensitive information

The big data exploration use case is shown in Figure 5-3. The MyBank developers can explore big data in their development environment to build the customer retention application they will deploy in the production environment.

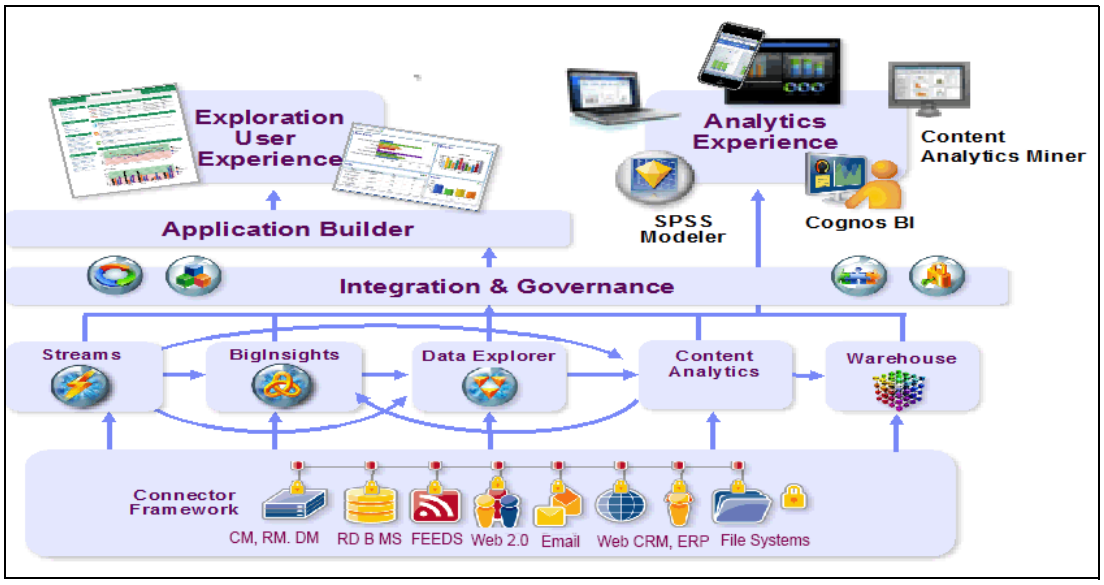


Figure 5-3 Big data exploration use case

Example of a customer's implementation

A global aerospace manufacturer increases knowledge worker efficiency and saves $36M annually with the help of big data solution implementation.

Requirements

- Delays in fixing maintenance issues are expensive and potentially incur financial penalties for out-of-service equipment
- Increase the efficiency of its maintenance and support technicians, support staff, and engineers

Results

- Supported 5,000 service representatives
- Eliminated use of paper manuals that were previously used for research
- Placed more than 40 additional airplanes into service without adding more support staff
- Reduced call time by 70% (from 50 minutes to 15 minutes)

5.3.2 Enhanced 360° View of the Customer use case

The purpose of this use case is to use big data to extend existing customer views by incorporating additional internal and external information sources. Gain a full understanding of customers—what makes them tick, why they buy, how they prefer to shop, why they switch, what they'll buy next, and what factors lead them to recommend a company to others.

This use case addresses the need to optimize every customer interaction by gathering comprehensive sets of data about the customers. These requirements are included:

- Create a connected picture of the customer
- Mine all existing and new sources of information about the customer
- Analyze social media to uncover sentiment about products
- Add value by optimizing every client interaction

The Enhanced 360º View of the Customer use case is shown in Figure 5-4 on page 525 and Figure 5-5 on page 525. The MyBank developers can explore big data further in their development environment, and add additional new sources of data for ingestion to build the customer retention application they will deploy in the production environment.

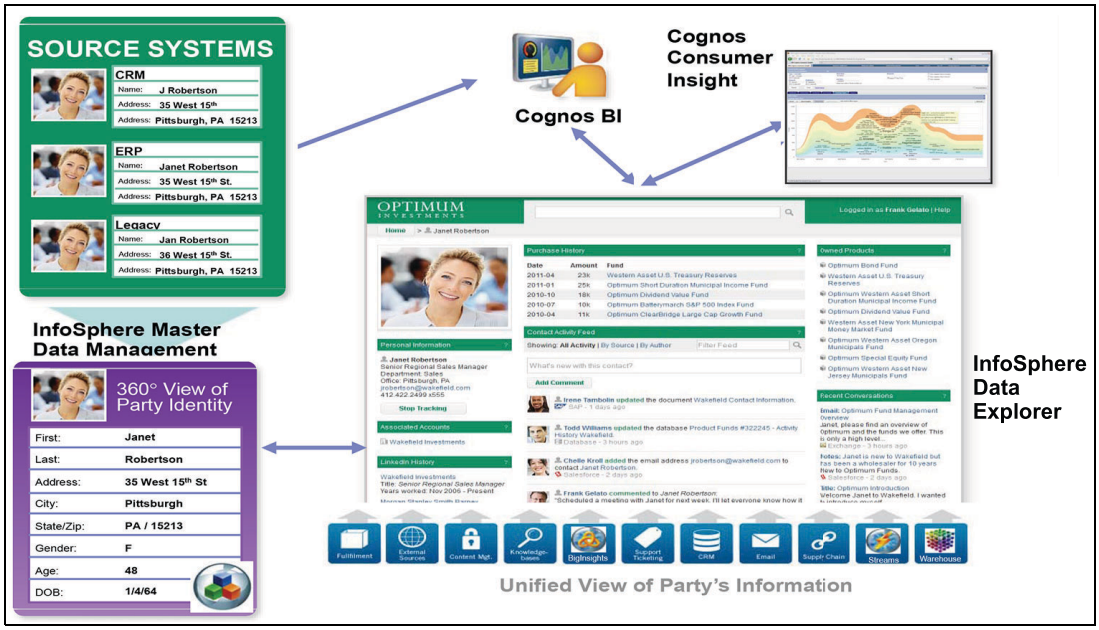


Figure 5-4 Enhanced 360 Degree View of the Customer use case

Figure 5-5 provides a high-level solution architecture representation of the Enhanced 360º View of the Customer use case.

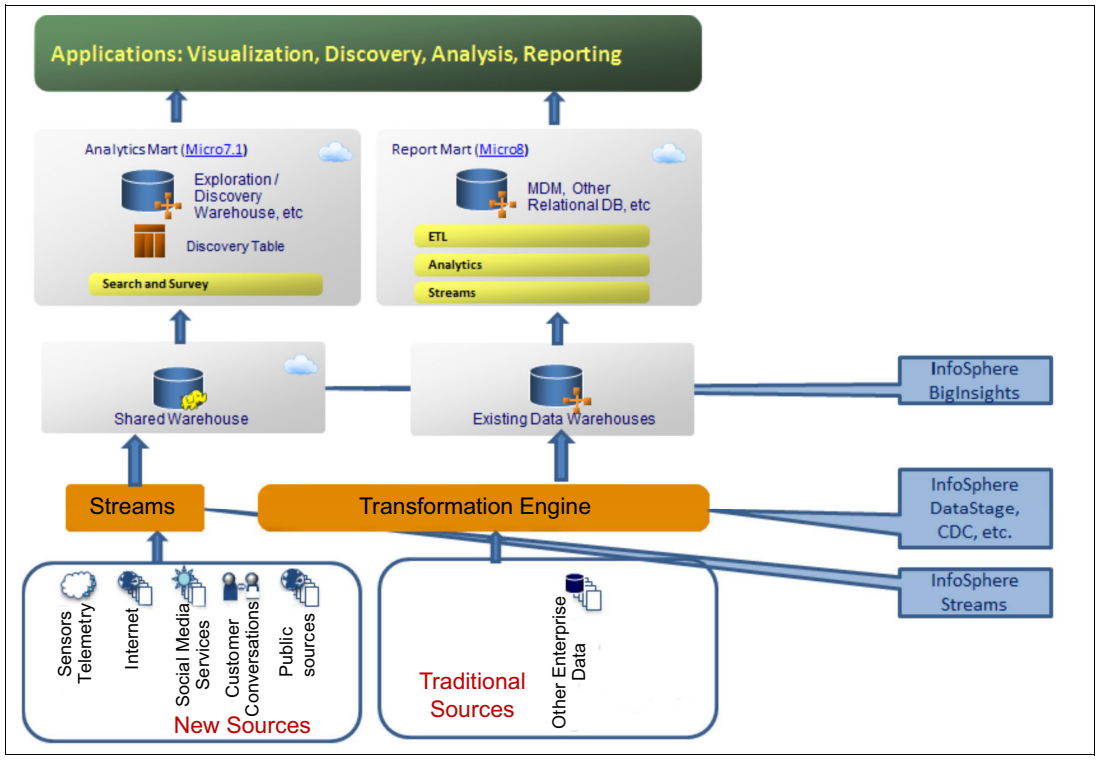


Figure 5-5 Enhanced 360º View of the Customer solution architecture

Example of an IBM customer's implementation

A product company improved information access across 30 different repositories.

Requirements

- Intuitive user interface for exploration and discovery across 30 repositories
- Encompass all global offices and be deployed quickly for a lower total cost of ownership.
- Provide secure search capabilities across Microsoft Sharepoint sites, intranet pages, wikis, blogs, and databases.

Results

- Achieved ability to identify experts across all global offices and 125,000 users worldwide
- Eliminated duplicate work effort being performed across all employees
- Improved discovery and "find-ability" across global organization
- Provided a federated and integrated system of knowledge and information that improved internal decision making

5.3.3 For more examples

For further use case detail for BigInsights, see the following web pages:

- The top five ways to get started with big data, an IBM Software Thought Leadership white paper

  http://ibm.co/1GqMXtz
- Top 5 Big Data Use Cases, IBM Big Data and Analytics Hub

  http://www.ibmbigdatahub.com/podcast/top-5-big-data-use-cases
- The 5 game changing big data use casesThe 5 game changing big data use cases

  http://www.ibm.com/software/data/bigdata/use-cases.html
- Societe Generale: Leading industry practices for real-time identification of risk-efficient trades with a centralized CVA desk

  http://www.ibm.com/software/businesscasestudies/us/en/corp?synkey=E657056F59617Y16
- *An Edge 2014 Highlight: Building a Multitenant Big Data Infrastructure at USAA*

  https://www.ibm.com/connections/blogs/platformcomputing/entry/multitenantbigdata
- Building a Multitenant Analytics Infrastructure: SPSS Analytic Catalyst, InfoSphere BigInsights, and IBM Platform Symphony

  https://www.youtube.com/watch?v=pGfYuMEGVPQ&list=UU8XvRSWNYZtzoZwGTt2v85w
- *IBM Platform Computing Solutions Reference Architectures and Best Practices*, SG24-8169

  http://www.redbooks.ibm.com/abstracts/sg248169.html
- Low-latency Hadoop for Risk Analytics with Platform Symphony, featuring Gord Sissons, IBM Platform Computing Senior Product Marketing Manager

  http://www.youtube.com/watch?v=JtfaXycBOsM

6

Advanced cloud provider scenarios

This chapter describes tasks and products that are not part of default IBM Cloud Manager installation or administration. In some cases, especially when cloud environment grows, there is a need to more sophisticated management to keep the whole environment in good shape. Sometimes, there might be a need to manually expand storage resources, for example when your physical storage is becoming full and you need to add additional storage expansions or volumes to existing storage systems or hosts. In the other cases, you might need to deploy virtual machines with compliance to quality-of-service policies or required performance. These tasks cannot be done or are hard to accomplish with a basic cloud infrastructure.

To assist an administrator or cloud provider with such tasks, this chapters describes the following methods and offerings:

- 6.1, “IBM Platform Resource Scheduler” on page 528
- 6.2, “Manual compute and storage expansion” on page 552

© Copyright IBM Corp. 2015. All rights reserved.

6.1 IBM Platform Resource Scheduler

In the cloud environments, with dozens or even hundreds of physical hosts running hundreds or thousands of virtual machines, new challenges emerge in terms of management, performance, electrical efficiency, and quality of service. Cloud providers are faced with questions similar to these:

- How should the workloads be placed into the infrastructure to achieve a high quality of service?
- How should the workloads be migrated as the infrastructure conditions change to maintain that high quality of service?
- How should the workloads be placed to reduce power consumption?
- How should the infrastructure resources be allocated to different competing tenants (such as projects, departments, applications) according to business priorities? Who should get what, when, and how much?
- How can the cloud service ensure that capacity is available for planned, typically recurring workloads (such as every Friday, 8 - 10 pm)?
- How can the cloud infrastructure team use site-specific resources to define allocation policies that are tailored to their specific needs?
- How can application license investments be optimally used?
- How can we enable or improve automation to reduce administrative work?
- How can we assure continued high availability, in case a physical server fails?

To address these problems, IBM enhances OpenStack with IBM Platform Resource Scheduler (PRS), which provides enterprise-class dynamic resource management for OpenStack cloud environments to optimize business agility and reduce the cost of cloud operations and management.

6.1.1 IBM Platform Resource Scheduler overview

IBM Platform Resource Scheduler (PRS) is a software component that originates from IBM's high-performance computing (HPC) Platform Computing portfolio. Platform Computing was acquired by IBM in 2012 and has enriched IBM software offerings since then. PRS is an intelligent resource scheduler that enables dynamic, policy-based resource allocation and migration. It is workload- and resource-aware, which means it can deploy or move virtual machines to the physical hosts that best fit the chosen policy. PRS determines where to create virtual machines during initial deployment, but also checks whether VMs are optimally placed during their run times. It replaces the OpenStack default Nova scheduler.

PRS supports several default policies and two modes of operation: global and host aggregate. You can choose one global policy for your whole cloud or several policies for different host aggregates. PRS also supports user-defined policies if none of the default ones fit the user's specific environment.

PRS introduces an enterprise-level solution for high availability (HA) within an OpenStack cloud. It supports default and also user-defined HA policies for physical server malfunction. If a physical host fails, PRS triggers the default or user-defined policy to move virtual machines to remaining hosts.

PRS is an optional component included in other IBM products, such as IBM Cloud Manager and IBM Cloud Orchestrator or PowerVC. It can be used with x86 and IBM Power Systems. For more information about PRS, see these web pages:

- IBM Platform Resource Scheduler on IBM developerWorks

  http://www.ibm.com/developerworks/servicemanagement/sds/prs/index.html

- IBM Platform Resource Scheduler in the IBM Knowledge Center

  http://ibm.co/1D7JAMr

6.1.2 The Enterprise Grid Orchestrator

PRS is based on Enterprise Grid Orchestrator (EGO), which originates from Platform Computing and is the foundation for PRS and several other IBM products. It is a resource management and distribution system. It controls underlying infrastructure and dynamically provides managed resources. PRS integrates OpenStack with EGO. OpenStack provisions virtual machines, and EGO schedules resources for OpenStack, based on criteria delivered through PRS.

PRS, EGO, and OpenStack integration is illustrated in Figure 6-1.

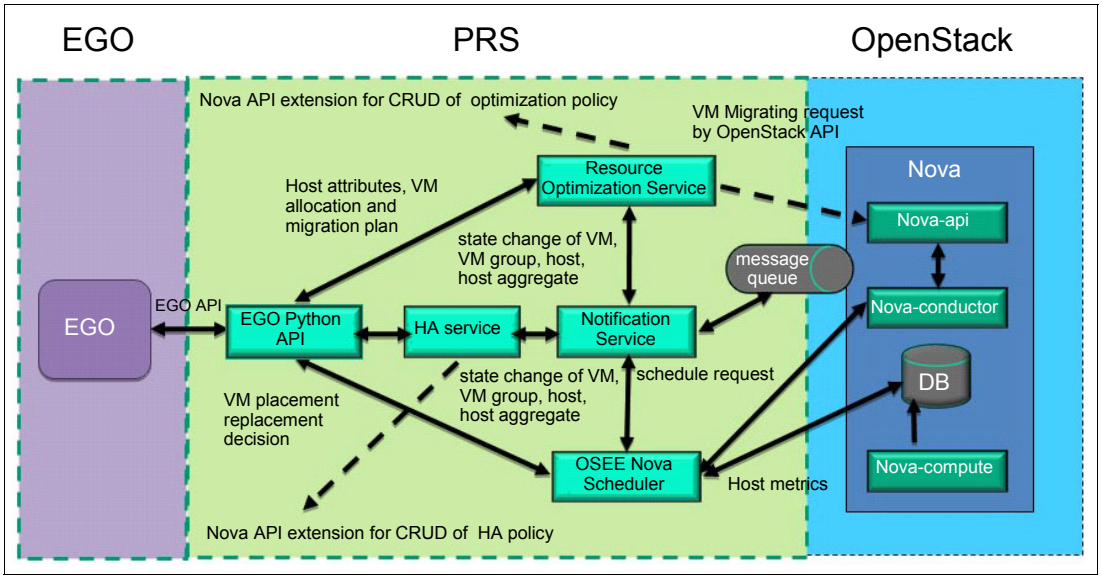


Figure 6-1 PRS architecture

PRS has the following 10 main components and functions:

1. EGO, which is used for resource scheduling for OpenStack.
2. An OpenStack Nova EGO scheduler plug-in, which is a resource scheduler add-on component that functions as the bridge between EGO and OpenStack.
3. A Nova notification service, which allows Nova to make notifications available to the Nova scheduler service, resource optimization service, and HA service in as close to real-time as possible.
4. A Nova resource optimization service, which communicates with EGO and the Nova scheduler to optimize resources according to a preconfigured goal.
5. A client to configure resource optimization policies.

6. A Nova high availability (HA) service, which provides HA support for a VM instance according to a preconfigured goal.
7. A client to configure HA policies.
8. A client to show hypervisor HA and maintenance status.
9. An ability to remove a host from scheduling so that the host is switched to maintenance mode during HA. You can later disable maintenance mode to allow Platform Resource Scheduler to use the host again.
10. A graphical user interface (the OpenStack dashboard) to customize resource optimization policies for both initial and runtime placement policies.

6.1.3 Platform Resource Scheduler installation

There are two ways to install PRS, either by installing RPMs or by using the Chef cookbook method. After the default Cloud Manager installation Chef deployment server is ready to use, so the easiest way to install PRS is by using Chef cookbooks.

Prerequisites for installation

The following are the PRS installation prerequisites:

- Supported servers
  - IBM POWER
  - x86_64
- Supported operating systems
  - Red Hat Enterprise Linux 6.4 or 6.5 on x86_64
  - Red Hat Enterprise Linux (RHEL) 6.4 or 6.5 on IBM PowerLinux 64
- Supported cloud management
  - IBM Cloud Manager 4.1
- Supported hypervisors and virtualization managers
  - IBM PowerVM
  - IBM PowerVC
  - PowerKVM
  - KVM
  - Hyper-V
  - VMware

To install PRS, you will need at least one physical host for all-in-one Cloud Manager installation or several hosts if you have controller plus n-compute Cloud Manager installation. You need to install PRS components on particular hosts:

EGO master host This can be the same host as the Cloud Manager Nova controller. It can also be a separate host. It installs the EGO binary package and runs the EGO service.

Nova controller Installs PRS optimization service, notification service, HA service and OpenStack graphical dashboard on the Nova controller host.

Nova compute Installs PRS library package and configures Nova to cooperate with PRS on the controller node. You must install this package on every Compute node that is managed by Cloud Manager. This can be the same host as the Nova controller if you have an all-in-one Cloud Manager installation. It can also be another Compute node if you have a controller + n-compute Cloud Manager installation.

EGO master candidate (Optional) If you want to make EGO master highly available. This is not mandatory and you must dedicate separate host or hosts for EGO master candidate.

Preparing the Chef environment

If you followed the steps in Chapter 4, “Cloud provider installation guidelines and implementation scenarios” on page 195, you should have Cloud Manager installed, configured, and running. You should also have compute KVM or PowerKVM nodes installed, with several virtual machines up and running. With this done, there is no need for special Chef environment configuration. The only thing that needs to be done is to install PRS on the controller and Compute nodes.

If you do not have the Chef environment prepared yet, you can find all necessary steps in *Platform Resource Scheduler v2.2 - Installing and configuring, SC27-6131-00* on page 21. “Installing PRS.”

In this example, we show how to install EGO master and PRS components on one controller node and PRS libraries on several KVM and PowerKVM Compute nodes. To do this using Chef, follow these steps:

1. Edit your topology JSON file to add the following roles for the controller node:

   ```
   "role[ibm-os-prs-ego-master]",
   "role[ibm-os-prs-controller-node]"
   ```

2. Edit your topology file to add the following role to the Compute nodes:

   ```
   "role[ibm-os-prs-compute-node]"
   ```

3. After the changes, your file should look similar to the one shown in Example 6-1 (added roles are in bold).

   Example 6-1 Topology file (JSON)

```
[root@cmodeploy itso-sde-deployment]# cat topo.json
{
  "name":"topo-ITSO-SDE-CLOUD",
  "description":"topo-ITSO-SDE-CLOUD",
  "environment":"ITSO-SDE-CLOUD",
  "secret_file":"/root/cmo/xhh/your-secret-key-name",
  "run_sequentially":false,
  "nodes": [
    {
      "fqdn":"cmocontroller.itso.ibm.com",
      "password":"password",
      "attribute_file":"/root/cmo/itso-sde-deployment/env-powervc.json",
      "quit_on_error":true,
      "run_order_number":1,
      "runlist": [
        "role[ibm-os-single-controller-node]",
        "role[ibm-os-powervc-driver-node]",
        "role[ibm-sce-node]",
        "role[ibm-os-prs-ego-master]",
        "role[ibm-os-prs-controller-node]"
      ]
    },
    {
      "fqdn":"cmo1.itso.ibm.com",
      "password":"password",
      "quit_on_error":true,
      "run_order_number":2,
      "runlist": [
        "role[ibm-os-compute-node-kvm]",
```

```
        "role[ibm-os-prs-compute-node]"
      ]
    },
    {
      "fqdn":"cmo2.itso.ibm.com",
      "password":"password",
      "quit_on_error":true,
      "run_order_number":2,
      "runlist": [
        "role[ibm-os-compute-node-kvm]",
        "role[ibm-os-prs-compute-node]"
      ]
    },
    {
      "fqdn":"powerkvm1.itso.ibm.com",
      "password":"password",
      "attribute_file":"/root/cmo/itso-sde-deployment/env-powekvm1.json",
      "quit_on_error":true,
      "run_order_number":1,
      "runlist": [
        "role[ibm-os-compute-node-powerkvm]",
        "role[ibm-os-prs-compute-node]"
      ]
    },
    {
      "fqdn":"powerkvm2.itso.ibm.com",
      "password":"password",
      "attribute_file":"/root/cmo/itso-sde-deployment/env-powekvm2.json",
      "quit_on_error":true,
      "run_order_number":1,
      "runlist": [
        "role[ibm-os-compute-node-powerkvm]",
        "role[ibm-os-prs-compute-node]"
      ]
    }
  ]
}
```

4. Update the topology file as shown in Example 6-2.

Example 6-2 Updating the topology file

```
[root@cmodeploy itso-sde-deployment]# knife os manage update topology topo.json
Updating topology 'topo-ITSO-SDE-CLOUD' ...
```

5. The output of the last knife command (Example 6-2) is very long, but at the end, you should see something similar to Example 6-3.

Example 6-3 Output from the topology file updates

```
Results for update of topology 'topo-ITSO-SDE-CLOUD'
Results for nodes with run_order_number '1'
Update of node at cmocontroller.itso.ibm.com was successful.
Update of node at powerkvm1.itso.ibm.com was successful.
Update of node at powerkvm2.itso.ibm.com was successful.
Results for nodes with run_order_number '2'
```

```
Update of node at cmo1.itso.ibm.com was successful.
Update of node at cmo2.itso.ibm.com was successful.
Update of topology 'topo.json' completed in 368 seconds.
```

6. After the installation, log in to the Cloud Manager web GUI. You should see the new *Resource Scheduler* option on the left side of the screen, as shown in Figure 6-2.

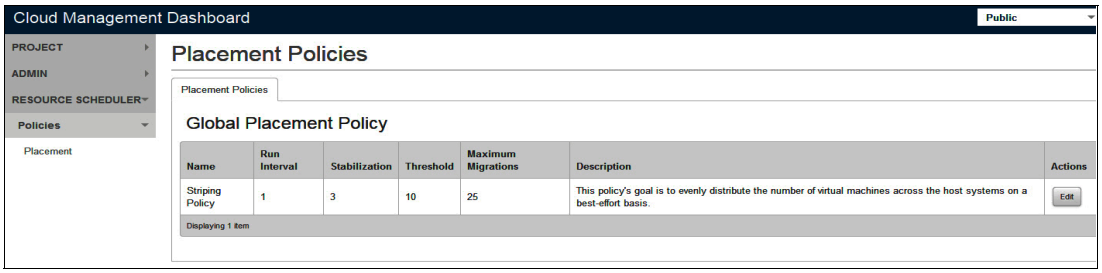


Figure 6-2 PRS options in Cloud Manager dashboard

7. After the installation, you should also see EGO processes running on controller node as shown in Example 6-4.

Example 6-4 Checking for EGO processes on the controller node

```
[root@cmocontroller ~]# service --status-all | grep -i ego
ego service is running.
openstack-nova-ibm-ego-ha-service (pid  4842) is running...
openstack-nova-ibm-ego-resource-optimization (pid 9615) is running...
```

6.1.4 Using PRS

In this section we will describe how to configure and use PRS, especially how to use different PRS policies, placement strategies and modes of operation. We will also show how to configure and use PRS HA policies.

Placement strategies

PRS can use two different strategies: *initial placement* or *runtime optimization*.

The initial placement strategy makes decisions about virtual machines placement, resizing, cold migration and live migration.

The runtime optimization strategy continuously monitors and optimizes resource use by live migrating virtual machines between physical hosts.

Policy modes

PRS can work in two modes: *global* or *host aggregate*.

In global mode, resource optimization policy affects whole cluster, both at initial placement and at a runtime. To set the global resource optimization policy type following command:

```
openstack-config --set /etc/nova/nova.conf DEFAULT policy_per_host_aggregate False
```

In host aggregate mode, each host aggregate can have its own policy or policies because it can have a policy for initial placement and different policy at run time. To set the host aggregate resource optimization policy, issue the following command:

```
openstack-config --set /etc/nova/nova.conf DEFAULT policy_per_host_aggregate True
```

In our lab, we used global placement policy, as shown in Figure 6-2 on page 533.

Resource policies

PRS supports the following resource policies:

Packing policy	Packs VM instances as densely as possible to maximize the number and density of running VM instances per physical host.
Striping policy	Packs the workload as widely as possible to provide best application performance.
CPU load balance policy	Balances the CPU load as evenly as possible among hypervisors.
Memory load balance policy	Balances the memory load as evenly as possible among hypervisors.
CPU utilization policy	Balances the CPU utilization as evenly as possible among hypervisors.

The policy definitions are placed in the `/etc/nova/rtpolicy_conf.xml` file, and you can view them by using the nova command:

```
nova rtpolicy-list
```

Or use **`cat`** to list the file. In our test environment, the policy file looks like Example 6-5.

Example 6-5 CPU utilization policy file

```
[root@cmocontroller nova]# cat rtpolicy_conf.xml
<?xml version="1.0" encoding="utf-8"?><placement_policies>
    <placement_policy>
        <id>1</id>
        <name>CPU Load Balance</name>
        <description>This policy's goal is to balance the CPU load as evenly as
possible among the host systems on a best-effort basis.</description>
        <state>disabled</state>
        <run_interval>1</run_interval>
        <stabilization>3</stabilization>
        <goal>(vcpuConsumed+vcpuRequest)*100/vcpu</goal>
        <threshold>70</threshold>
        <max_parallel>25</max_parallel>
        <action>migrate_vm</action>
    </placement_policy>
    <placement_policy>
        <id>2</id>
        <name>Memory Load Balance</name>
       <description>This policy's goal is to balance the memory load as evenly as
possible among the host systems on a best-effort basis.</description>
        <state>disabled</state>
        <run_interval>1</run_interval>
        <stabilization>3</stabilization>
        <goal>(memSizeConsumed+memSizeRequest)*100/memSize</goal>
        <threshold>70</threshold>
        <max_parallel>25</max_parallel>
        <action>migrate_vm</action>
    </placement_policy>
    <placement_policy>
```

```
        <id>3</id>
        <name>Striping Policy</name>
        <description>This policy's goal is to evenly distribute the number of
virtual machines across the host systems on a best-effort basis.</description>
        <state>enabled</state>
        <run_interval>1</run_interval>
        <stabilization>3</stabilization>
        <goal>running_vms+1</goal>
        <threshold>10</threshold>
        <max_parallel>25</max_parallel>
        <action>migrate_vm</action>
    </placement_policy>
    <placement_policy>
        <id>4</id>
        <name>Packing Policy</name>
        <description>This policy's goal is to pack host systems as densely as
possible with virtual machines on a best-effort basis.</description>
        <state>disabled</state>
        <run_interval>1</run_interval>
        <stabilization>3</stabilization>
        <goal>0-(running_vms+1)</goal>
        <threshold>-10</threshold>
        <max_parallel>25</max_parallel>
        <action>migrate_vm</action>
    </placement_policy>
    <placement_policy>
        <id>5</id>
        <name>CPU Utilization Policy</name>
        <description>This policy's goal is to balance the CPU utilization as
evenly as possible among the host systems on a best-effort basis.</description>
        <state>disabled</state>
        <run_interval>1</run_interval>
        <stabilization>3</stabilization>
        <goal>cpu.percent</goal>
        <threshold>70</threshold>
        <max_parallel>25</max_parallel>
        <action>migrate_vm</action>
    </placement_policy>
```

These are the attributes in the policy file shown in Example 6-5 on page 534:

- `<id>` The policy ID.
- `<name>` The policy name.
- `<description>` A description of the policy.
- `<state>` The current state of the policy (such as “enabled” or “disabled”).
- `<run_interval>` A runtime interval, in minutes, when the policy will run. For example, a value of **1** indicates that the policy will run every one minute.
- `<stabilization>` The number of consecutive intervals, in which a policy goal must exceed the policy threshold value before triggering the policy action.

- <goal> The resource requirement string, which can be one of five kinds:
  - CPU sensitive: (vcpuConsumed+vcpuRequest) * 100/vcpu
    - vcpuConsumed: Host vcpus that has been consumed by the instance
    - vcpuRequested: vcpus requested by the instance
    - vcpu: Total vcpu for the host
  - Memory sensitive: (memSizeConsumed+memSizeRequest) * 100/memSize
    - memSizeConsumed: Host memory that has been consumed by the instance
    - memSizeRequest: memory requested by the instance
    - memSize: Total memory size for the host
  - Striping policy: (running_vms + 1)
    - running_vms: The number of running instances on the host
  - Packing policy: (0-(running_vms + 1))
    - running_vms: The number of running instances on the host
  - CPU utilization policy: cpu.percent. The percent of CPUs that the host uses
- <threshold> Starting point for live migration. This is a percentage of either the CPU utilization or memory utilization based on the configuration of the <goal>.
- <max_parallel> The maximum number of live migration operations in the cluster. This is used to control the number of migration operations in the whole system. If the number of operations (triggered by the load balance policy) has met the <max_parallel> value, the load balance policy will not trigger more migration operations.
- <action> Indicates when the policy has met all conditions. The policy engine will execute this action to balance the hypervisor load among the whole system. Currently, only the migrate_vm action is supported.

You can also see them in the Cloud Manager web GUI by clicking **Resource Scheduler > Policies** > **Placement > Edit**. See Figure 6-3 on page 537.

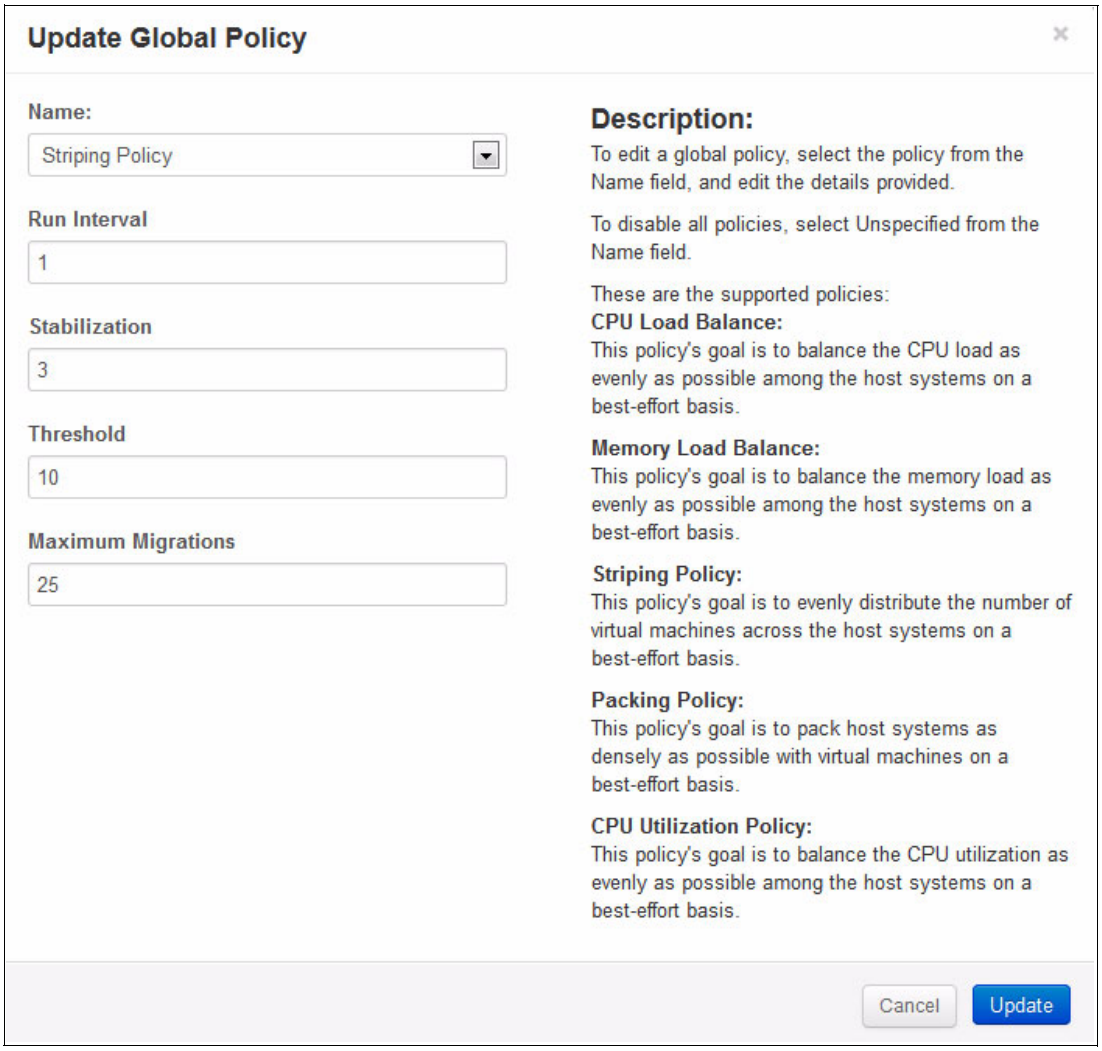


Figure 6-3 PRS placement policies

For descriptions of particular attributes, see “Resource optimization global based policy” in the IBM Knowledge Center:

http://ibm.co/1CK68AN

Platform Resource Scheduler HA policy

PRS introduces a high availability feature to OpenStack. You can configure PRS to move virtual machines to other physical hosts in the cluster in case the host they run on fails. PRS supports global operating mode or host aggregate mode. In global mode, PRS supports one HA policy per whole cluster. In host aggregate mode, PRS uses one HA policy per each host aggregate. The configuration of HA policy can be found in the `/etc/nova/hapolicy_conf.xml` file in the Cloud Manager controller. You can list it by using this Nova command:

```
nova hapolicy-list
```

Or use **`cat`** to list the file.

In our test environment, the `hapolicy` file looks like Example 6-6.

Example 6-6 PRS HA policy file

```
[root@cmocontroller nova]# cat hapolicy_conf.xml
<?xml version="1.0" encoding="utf-8"?>
<hapolicies>
    <hapolicy>
        <id>1</id>
        <name>HA service policy</name>
        <description>This policy's goal is to support HA so that when a node
fails, workloads can be redistributed.</description>
        <state>disabled</state>
        <run_interval>1</run_interval>
        <stabilization>3</stabilization>
    </hapolicy>
</hapolicies>
```

These are the attributes of the HA policy file shown in Example 6-6:

<id>	The policy ID.
<name>	The policy name.
<description>	A description of the policy.
<state>	The current state of the policy (such as “enabled” or “disabled”).
<run_interval>	A runtime interval, in minutes, when the policy will run. For example, a value of 1 indicates that the policy will run every one minute.
<stabilization>	The number of consecutive intervals, in which a policy goal must exceed the policy threshold value, before triggering the policy action.

PRS maintenance mode

In some cases, you want to disable PRS for particular host, such as when the host is malfunctioning or you want to perform software upgrade. PRS has special task to accomplish this. It can switch physical hosts to maintenance mode without shutting down running virtual machines. It can also evacuate all VMs or only running VMs form that host if there is a need to do so. In maintenance mode, the host is disabled for scheduling.

Note: If you want to use maintenance mode with migration of virtual machines, you need to have those virtual machines running on shared storage. Otherwise, migration fails.

On the Controller or Compute node, you can use this Nova command:

```
nova maintenance-enable [--migrate all | active only] hostname
```

To enable maintenance mode on a particular host, use the `--migrate` option to migrate either all or only running virtual machines from this host. Similarly, if you want to disable PRS maintenance mode on a particular host, use the following command:

```
nova maintenance-disable hostname
```

Note: The PRS host maintenance feature is not installed by Chef. The RPM package with maintenance features must be installed manually by using this command:

```
rpm -ivh python-novaclient-ibm-ego-prs-hypervisor-maintenance-version.ibm.el6.noarch.rpm
```

Other PRS features

PRS has more features, such as user-defined optimization policies, user-defined HA policies, and using EGO failover to another physical host. Those features are beyond the scope of this book, but you can find instructions how to configure and use them in the Platform Resource Scheduler section of the IBM Knowledge Center:

http://ibm.co/1D7JAMr

6.1.5 Platform Resource Scheduler use example

This section shows you how to perform the following tasks:

- "Enable PRS"
- "Select policy"
- "Deploy instances" on page 541
- "Modify the policy" on page 544
- "Deploy instances with modified policy" on page 545
- "Disable PRS" on page 546
- "Enable the HA policy" on page 547
- "Modifythe HA policy" on page 549
- "Disable the HA policy" on page 549
- "Enable maintenance mode" on page 550
- "Disable maintenance mode" on page 551

Enable PRS

In this example scenario, we use PRS global policy, so all optimization policies will affect whole cluster. To enable the global policy, we use the command shown in Example 6-7.

Example 6-7 Enabling the PRS global policy

```
[root@cmocontroller nova]# openstack-config --set /etc/nova/nova.conf DEFAULT
policy_per_host_aggregate False

[root@cmocontroller nova]# cat nova.conf | grep -i policy_per_host_aggregate
policy_per_host_aggregate = False
```

Select policy

To select the policy, you can use either the CLI or GUI. We show an example using the command line. First, list all available policies, as shown in Example 6-8.

Example 6-8 Listing all available policies

```
[root@cmocontroller nova]# nova rtpolicy-list
+----+---------------------------+-------------------------------------------+
| id | name              | state   | description | goal
|
+----+---------------------------+-------------------------------------------+
| 1  | CPU Load Balance       | disabled | This policy's goal is to balance the
CPU load as evenly as possible among the host systems on a best-effort basis.
| (vcpuConsumed+vcpuRequest)*100/vcpu          |

| 2  | Memory Load Balance    | disabled | This policy's goal is to balance the
memory load as evenly as possible among the host systems on a best-effort basis.
| (memSizeConsumed+memSizeRequest)*100/memSize |
```

```
| 3  | Striping Policy        | disabled | This policy's goal is to evenly
distribute the number of virtual machines across the host systems on a best-effort
basis. | running_vms+1                                  |

| 4  | Packing Policy         | disabled | This policy's goal is to pack host
systems as densely as possible with virtual machines on a best-effort basis.
| 0-(running_vms+1)                             |

| 5  | CPU Utilization Policy | disabled | This policy's goal is to balance the
CPU utilization as evenly as possible among the host systems on a best-effort
basis. | cpu.percent                                    |
+----+------------------------+----------------------------------------------+
```

As you can see in Example 6-8 on page 539, all policies are disabled. Now you can choose one of the policies that you want PRS to use to manage virtual machine deployment. We choose the striping policy. First, we list its parameters, as shown in Example 6-9.

Example 6-9 Listing the striping policy parameters

```
[root@cmocontroller nova]# nova rtpolicy-show 3
+---------------+------------------------------------------------------------------+
| Property    | Value
|
+---------------+------------------------------------------------------------------+
| action         | migrate_vm
|
| description   | This policy's goal is to evenly distribute the number of virtual
machines across the host systems on a best-effort basis. |
| goal           | running_vms+1
|
| id             | 3
|
| max_parallel  | 25
|
| name          | Striping Policy
|
| run_interval  | 1
|
| stabilization | 3
|
| state         | disabled
|
| threshold    | 10
|
+---------------+------------------------------------------------------------------+
```

Then, we enable the policy, as shown in Example 6-10.

Example 6-10 Enabling the policy

```
[root@cmocontroller nova]# nova rtpolicy-update --state enabled 3
+---------------+------------------------------------------------------------------+
| Property    | Value
|
+---------------+------------------------------------------------------------------+
```

```
| action      | migrate_vm
|
| description   | This policy's goal is to evenly distribute the number of virtual
machines across the host systems on a best-effort basis. |
| goal          | running_vms+1
|
| id            | 3
|
| max_parallel | 25
|
| name          | Striping Policy
|
| run_interval | 1
|
| stabilization | 3
|
| state         | enabled
|
| threshold     | 10
|
+---------------+-------------------------------------------------------------+
```

Deploy instances

Now that PRS is using the striping policy, we can deploy several virtual machines and watch how PRS places them on all available servers. In our environment, we have two PowerKVM servers, and we create four virtual machines with the Cloud Manager GUI. In the Cloud Manager GUI, select **Project > Instances**, and then click **Launch Instance**, as shown on Figure 6-4 on page 542.

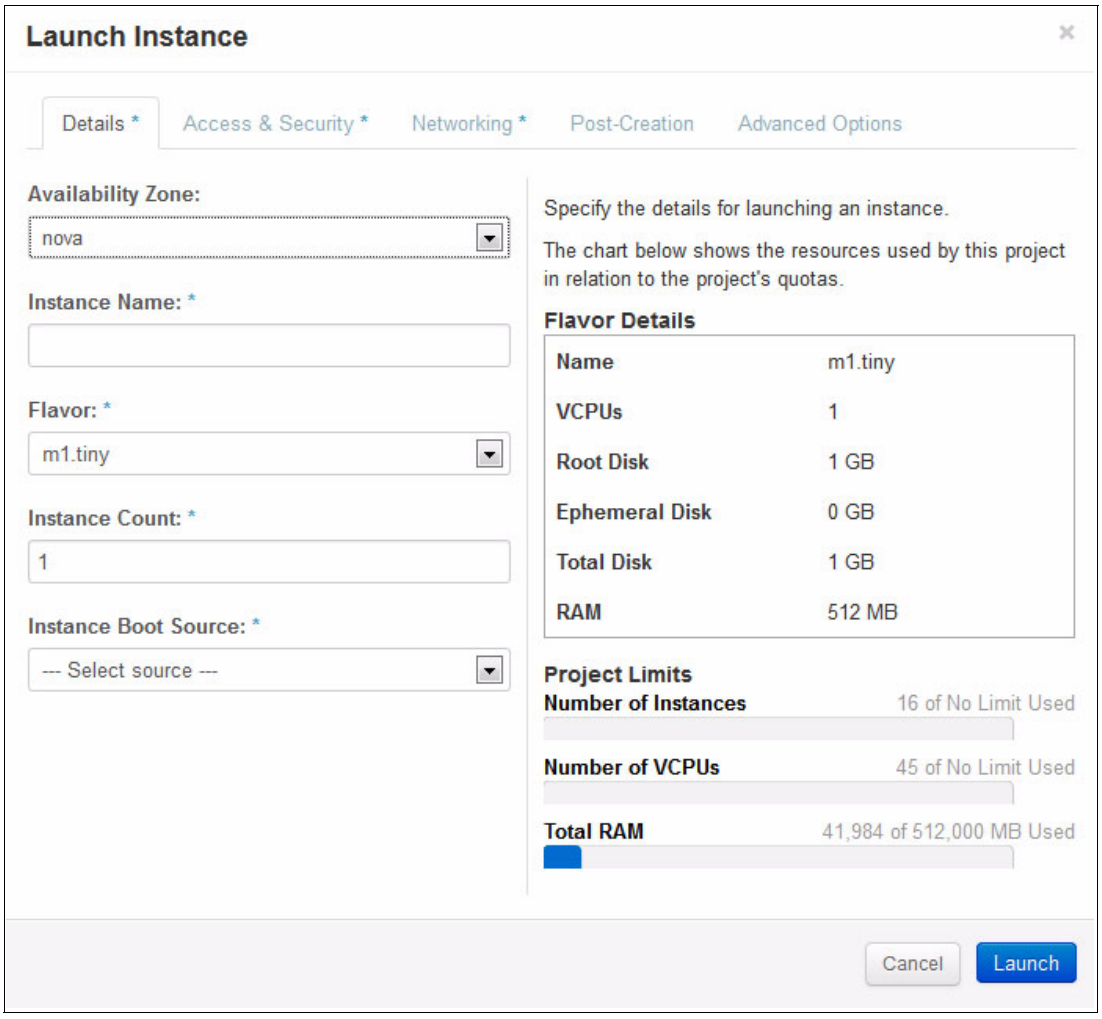


Figure 6-4 Launch instance menu

The parameters of the virtual machine do not matter, because we want to demonstrate how PRS works, and we won't be deploying any workload on top of the instance. We create four instances for the PowerLinux image with prefix type `prs_power_test and m1.tiny`. Figure 6-5 shows four instances deployed and running. Notice that all instances have the `prs_power_test` prefix plus the instance ID as suffix. We need this suffix to identify instances on a particular host.

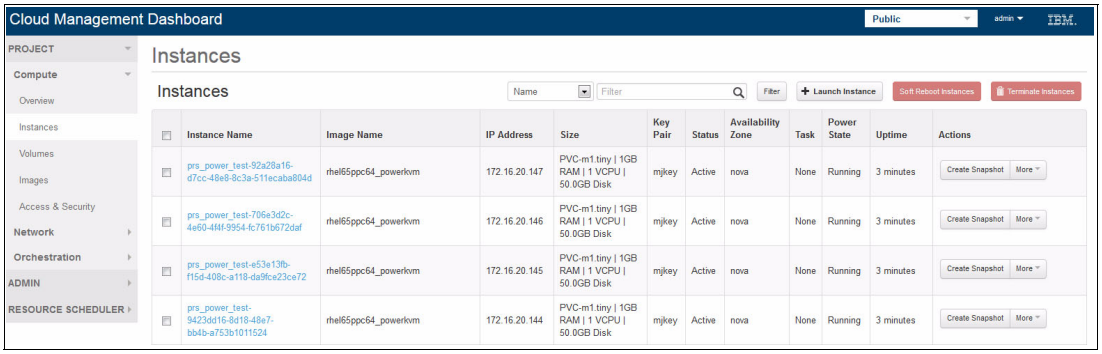


Figure 6-5 Four PowerLinux instances running on PowerKVM hosts

Now we can check the instances distribution in the GUI or in the CLI. In the GUI, select **Admin > Hypervisors** and click one of the hypervisors to list which virtual machines are running on it. In our scenario, we have two PowerKVM hypervisors and we can check whether our four virtual machines are deployed according to the chosen policy. Figure 6-6 shows instances running on the first PowerKVM hypervisor.

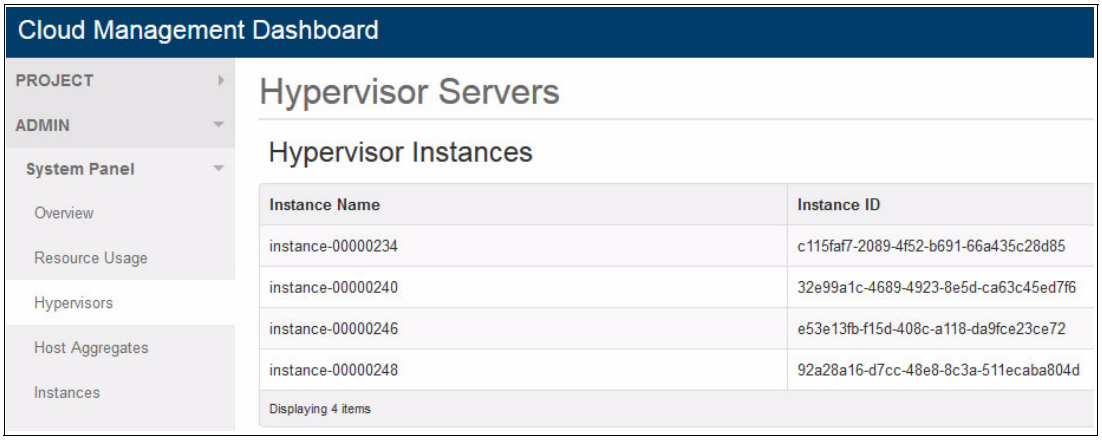


Figure 6-6 Instances running on the first PowerKVM

Figure 6-7 shows instances running on the second PowerKVM hypervisor.

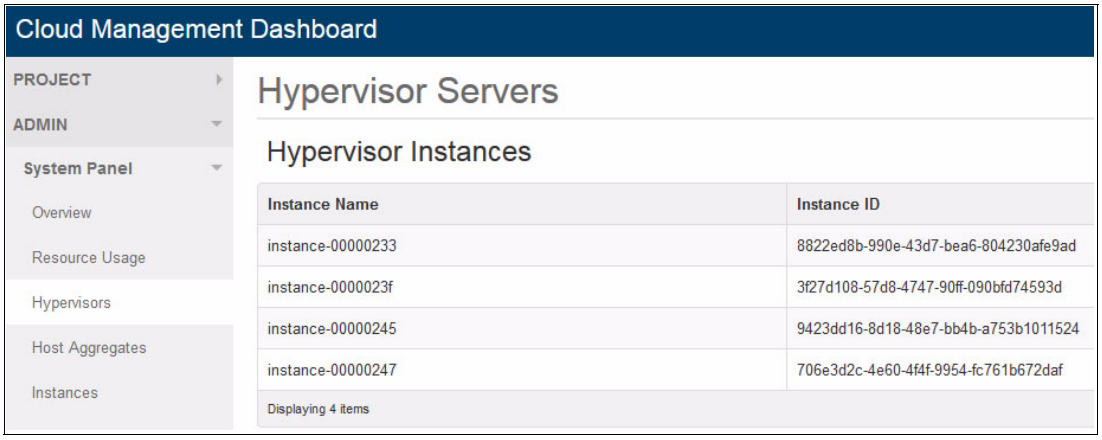


Figure 6-7 Instances running on the second PowerKVM

When you compare the instances IDs from the Instances view with the instances IDs from the hypervisor view, you can notice that two of the instances are deployed on the first PowerKVM host and two of instances are deployed on the second PowerKVM host. This is expected behavior because our placement policy is set to striping, so PRS stripes the virtual machines on all available hosts.

Table 6-1 aligns instances names from the Instances view with instance IDs from the Hypervisors view.

Table 6-1 Corresponding instances' names and IDs

Instance Name (Hypervisors view)	Instance Name (Instances view)	Instance ID (Hypervisors view)	Host name
instance-00000246	prs_power_test-e53e13fb-f15d-408c-a118-da9fce23ce72	e53e13fb-f15d-408c-a118-da9fce23ce72	powerkvm1
instance-00000248	prs_power_test-92a28a16-d7cc-48e8-8c3a-511ecaba804d	92a28a16-d7cc-48e8-8c3a-511ecaba804d	powerkvm1
instance-00000247	prs_power_test-706e3d2c-4e60-4f4f-9954-fc761b672daf	706e3d2c-4e60-4f4f-9954-fc761b672daf	powerkvm2
instance-00000245	prs_power_test-94233dd16-8d18-48e7-bb4b-a753b1011524	94233dd16-8d18-48e7-bb4b-a753b1011524	powerkvm2

Modify the policy

You can change the placement policy either via GUI or CLI. In this section, we show how to do it in the Cloud Manager GUI.

After logging in to Cloud Manager, go to **Resource Scheduler > Policies > Placement** and click **Edit** in the current policy row. From the drop-down list, chose the policy that you want to use. As Figure 6-8 on page 545 shows, we use the packing policy, which means that PRS will try to place as many instances as it can on the fewest hosts.

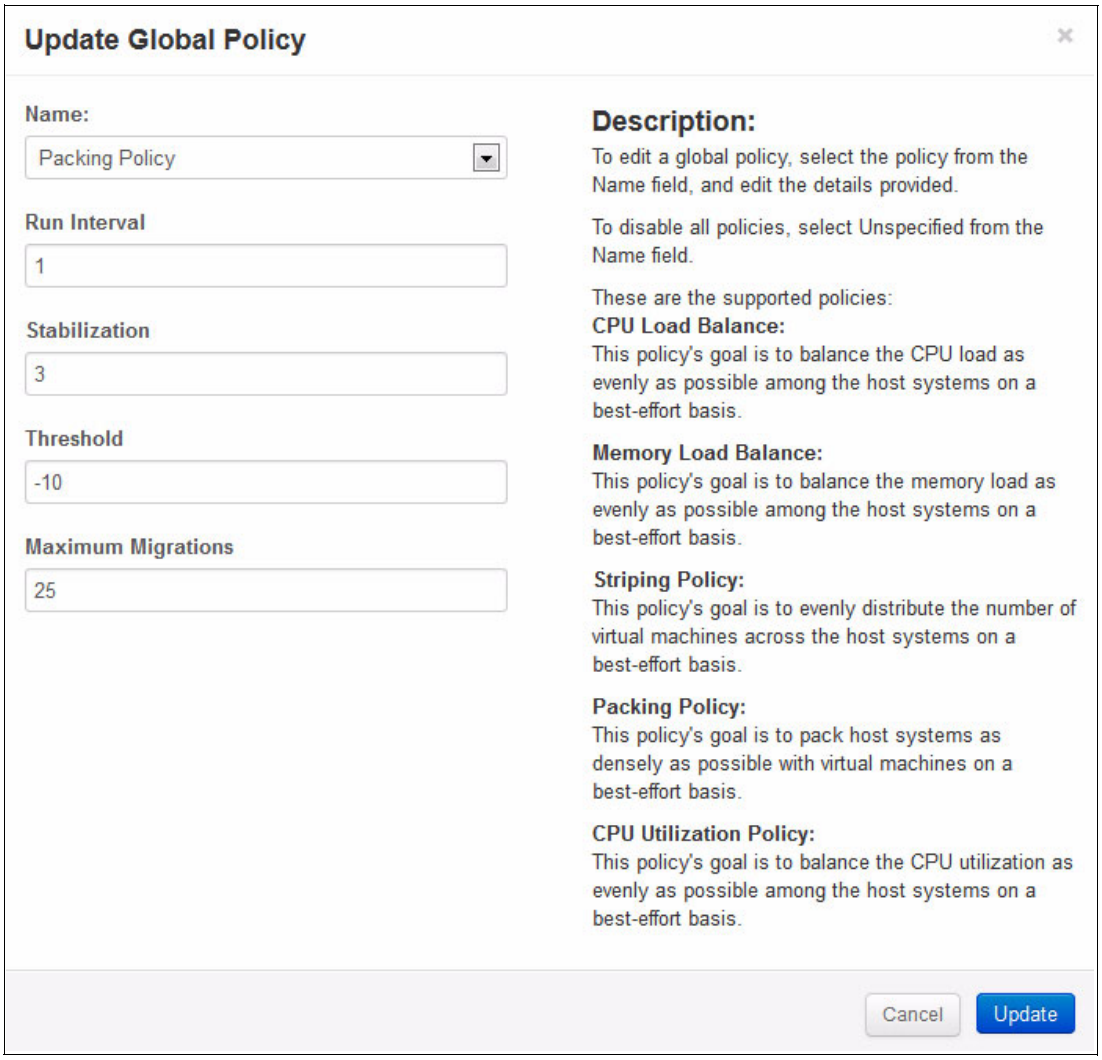


Figure 6-8 Changing PRS policy

Deploy instances with modified policy

After modifying the policy, we can create several instances in the same way that the we did in "Deploy instances" on page 541. Figure 6-9 shows deployment of additional four PowerLinux instances with the `prs_power_packing_test` prefix.

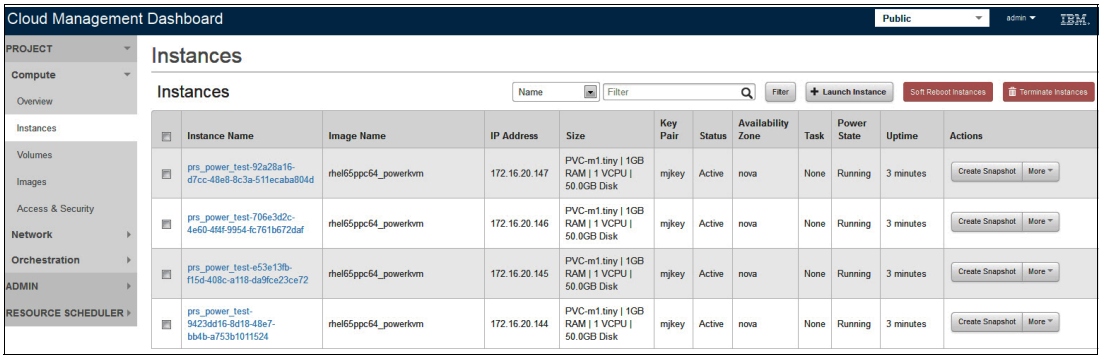


Figure 6-9 Four PowerLinuxinstances on PowerKVM hosts after changing the policy

Now we can check how instances were deployed on the PowerKVM hypervisors. We can do this the same way that we did in “Deploy instances” on page 541. Figure 6-10 shows instances running on the first PowerKVM hypervisor.

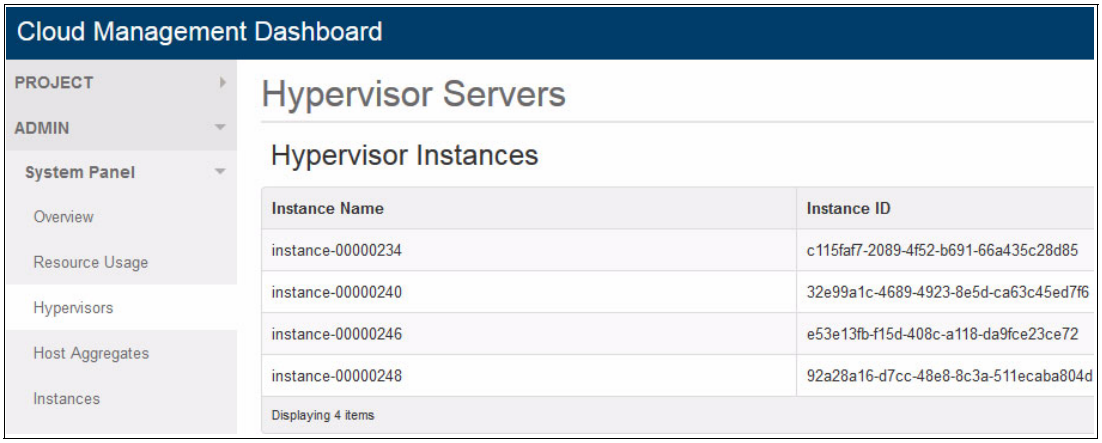


Figure 6-10 Instances running on the first PowerKVM

Figure 6-11 shows instances running on the second PowerKVM hypervisor.

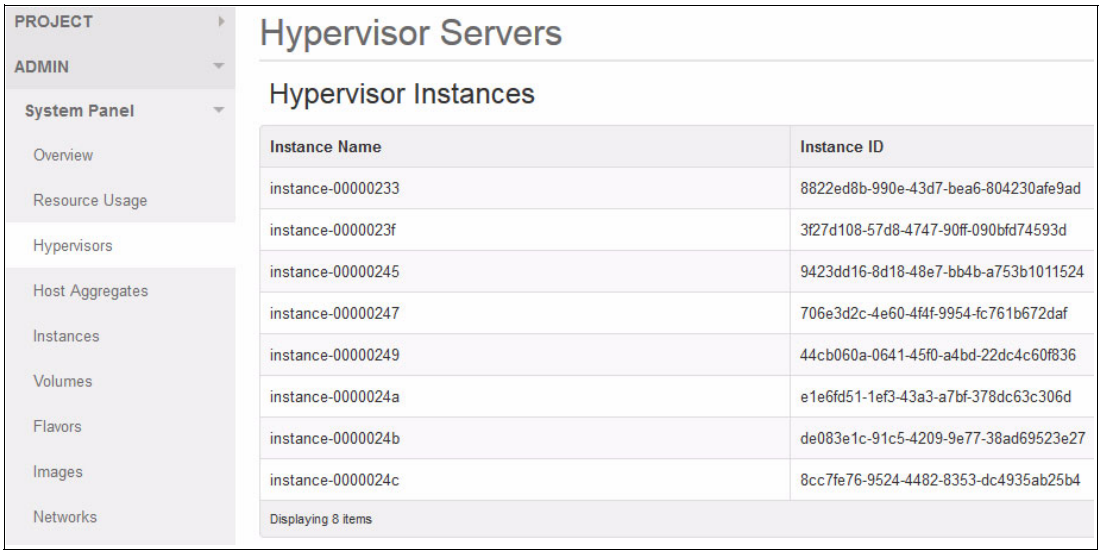


Figure 6-11 Instances running on the second PowerKVM

As you can see, after changing the policy from striping to packing, PRS deployed all instances on one of the PowerKVMs, which is correct and expected behavior. With the PRS packing policy, we try to put as many instances as possible on the fewest hosts.

Disable PRS

If you want to disable PRS, you can do it in the GUI or with the CLI. In the GUI, after logging to Cloud Manager, go to **Resource Scheduler > Policies** > **Placement** and click **Edit** in the current policy row.

From the drop-down list, choose **Unspecified** to disable PRS, as shown on Figure 6-12.

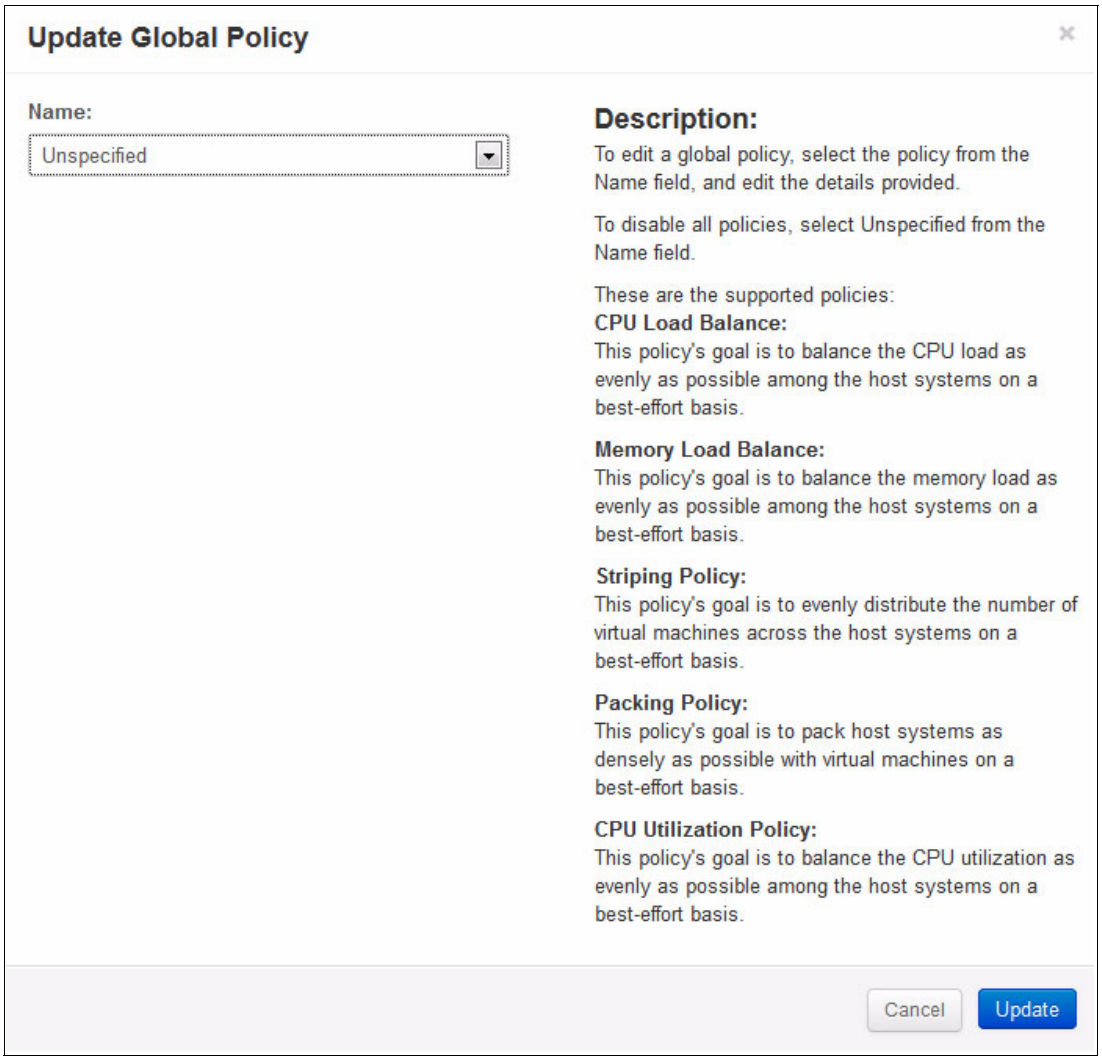


Figure 6-12 Disabling PRS

Enable the HA policy

If you want to use the high availability feature of PRS, you must enable it from the command line. As described in the “Platform Resource Scheduler HA policy” on page 537, PRS supports only one policy, either default or user-defined. First, list the current status of HA policy as shown in Example 6-11.

Example 6-11 Listing the current status of the HA policy

```
[root@cmocontroller nova]# nova hapolicy-list
+----+------------------+----------+-----------------------------------------+
| id | name        | state   | description
|
+----+------------------+----------+-----------------------------------------+
| 1  | HA service policy | disabled | This policy's goal is to support HA so that
when a node fails, workloads can be redistributed. |
+----+------------------+----------+-----------------------------------------+
```

You can also list the details of the HA policy using the command as shown in Example 6-12.

Example 6-12 Listing the details of the HA policy via the command line

```
[root@cmocontroller nova]# nova hapolicy-show 1
+---------------+-----------------------------------------------------------------+
| Property      | Value
|
+---------------+-----------------------------------------------------------------+
| description   | This policy's goal is to support HA so that when a node fails,
workloads can be redistributed. |
| id            | 1
|
| name          | HA service policy
|
| run_interval  | 1
|
| stabilization | 3
|
| state         | disabled
|
+---------------+-----------------------------------------------------------------+
```

In our environment, the current status of the HA policy is disabled. To enable it, use the command shown in Example 6-13.

Example 6-13 Enabling the HA policy

```
[root@cmocontroller nova]# nova hapolicy-update --state enabled 1
+---------------+-----------------------------------------------------------------+
| Property       | Value
|
+---------------+-----------------------------------------------------------------+
| description   | This policy's goal is to support HA so that when a node fails,
workloads can be redistributed. |
| id            | 1
|
| name          | HA service policy
|
| run_interval  | 1
|
| stabilization | 3
|
| state         | enabled
|
+---------------+-----------------------------------------------------------------+
```

Now the HA policy is enabled and PRS periodically checks the hosts' status. If PRS finds a host down, it moves all virtual machines to the remaining hosts according to the resource optimization policy.

Modifythe HA policy

You can modify several parameters of the HA policy. For example, to modify the HA policy run interval, use the command as shown in Example 6-14.

Example 6-14 Modifying the HA policy

```
[root@cmocontroller nova]# nova hapolicy-update --run_interval 3 1
+---------------+------------------------------------------------------------------+
| Property      | Value
|
+---------------+------------------------------------------------------------------+
| description   | This policy's goal is to support HA so that when a node fails,
workloads can be redistributed. |
| id            | 1
|
| name          | HA service policy
|
| run_interval  | 3
|
| stabilization | 3
|
| state         | enabled
|
+---------------+------------------------------------------------------------------+
```

After changing the run interval from 1 to 3, PRS checks for hosts status changes every 3 minutes. You can also change several other parameters, as shown in Example 6-15.

Example 6-15 Checking for status changes

```
[root@cmocontroller nova]# nova hapolicy-update
usage: nova hapolicy-update [--state <enabled|disabled>]
                            [--description <description>]
                            [--run_interval <run_interval>]
                            [--stabilization <stabilization>] [--name <name>]
                            <id>
```

Disable the HA policy

If, for any reason, you want to disable the PRS HA policy, you can do this by using the command as shown in Example 6-16.

Example 6-16 Disabling the PRS HA policy

```
[root@cmocontroller nova]# nova hapolicy-update --state disabled 1
+---------------+------------------------------------------------------------------+
| Property      | Value
|
+---------------+------------------------------------------------------------------+
| description   | This policy's goal is to support HA so that when a node fails,
workloads can be redistributed. |
| id            | 1
|
| name          | HA service policy
|
| run_interval | 3
|
```

```
| stabilization| 3
|
| state        | disabled
|
+--------------+-------------------------------------------------------------+
```

Example 6-16 on page 549 disables the HA policy, PRS does not move any virtual machines from failed physical hosts.

Enable maintenance mode

Before enabling maintenance mode, check the state of your hypervisors, as shown in Example 6-17.

Example 6-17 Checking the hypervisor status

```
[root@cmocontroller ~]# nova hypervisor-status-list
+----+-----------------------+-----------------+-----------+--------------------+
| id | hypervisor_hostname   | hypervisor_type | ha_status | maintenance_status |
+----+-----------------------+-----------------+-----------+--------------------+
| 1  | cmo1.itso.ibm.com     | QEMU            | ok        | ok                 |
| 3  | cmo2.itso.ibm.com     | QEMU            | ok        | ok                 |
| 21 | powerkvm1.itso.ibm.com| QEMU            | ok        | ok                 |
| 22 | powerkvm2.itso.ibm.com| QEMU            | ok        | ok                 |
| 41 | 8205E6C_06A22ER       | powervm         | ok        | ok                 |
| 61 | 8233E8B_061AA6P       | powervm         | ok        | ok                 |
+----+-----------------------+-----------------+-----------+--------------------+
```

We enable maintenance on the `cmo1` host. The `cmo1` status is shown in Example 6-18.

Example 6-18 cmo1 status

```
[root@cmocontroller ~]# nova hypervisor-status-show 1
+---------------------+--------------------+
| Property            | Value              |
+---------------------+--------------------+
| ha_status           | ok                 |
| hypervisor_hostname | cmo1.itso.ibm.com  |
| hypervisor_type     | QEMU               |
| id                  | 1                  |
| maintenance_status  | ok                 |
| migrate             |                    |
+---------------------+--------------------+
```

Then, enable the policy on the chosen hypervisor. In our environment, we do not use the option to migrate hosts (see Example 6-19).

Example 6-19 Enabling the policy on the selected hypervisor

```
[root@cmocontroller ~]# nova maintenance-enable cmo1.itso.ibm.com
+---------------------+--------+
| hypervisor_hostname | status |
+---------------------+--------+
| cmo1.itso.ibm.com   | enable |
+---------------------+--------+
```

To check the hypervisor status, use the command in Example 6-20.

Example 6-20 Checking the hypervisor status

```
[root@cmocontroller ~]# nova hypervisor-status-list
+----+-----------------------+-----------------+-----------+--------------------+
| id | hypervisor_hostname   | hypervisor_type | ha_status | maintenance_status |
+----+-----------------------+-----------------+-----------+--------------------+
| 1  | cmo1.itso.ibm.com     | QEMU            | ok        | on                 |
| 3  | cmo2.itso.ibm.com     | QEMU            | ok        | ok                 |
| 21 | powerkvm1.itso.ibm.com | QEMU           | ok        | ok                 |
| 22 | powerkvm2.itso.ibm.com | QEMU           | ok        | ok                 |
| 41 | 8205E6C_06A22ER       | powervm         | ok        | ok                 |
| 61 | 8233E8B_061AA6P       | powervm         | ok        | ok                 |
+----+-----------------------+-----------------+-----------+--------------------+

[root@cmocontroller ~]# nova hypervisor-status-show 1
+---------------------+-------------------+
| Property            | Value             |
+---------------------+-------------------+
| ha_status           | ok                |
| hypervisor_hostname | cmo1.itso.ibm.com |
| hypervisor_type     | QEMU              |
| id                  | 1                 |
| maintenance_status  | on                |
| migrate             | none              |
+---------------------+-------------------+
```

The `cmo1` host is now ready for maintenance, because it is no longer considered a source or target for VM migration by PRS.

Disable maintenance mode

After maintenance is finished, you can turn PRS back on. To do so, use the command in Example 6-21.

Example 6-21 Disabling maintenance mode

```
[root@cmocontroller ~]# nova maintenance-disable cmo1.itso.ibm.com
+---------------------+---------+
| hypervisor_hostname | status  |
+---------------------+---------+
| cmo1.itso.ibm.com   | disable |
+---------------------+---------+
```

Then, check the status of the hypervisor to make sure that the PRS maintenance mode is disabled, as shown in Example 6-22.

Example 6-22 Checking PRS status

```
[root@cmocontroller ~]# nova hypervisor-status-list
+----+-----------------------+-----------------+-----------+--------------------+
| id | hypervisor_hostname   | hypervisor_type | ha_status | maintenance_status |
+----+-----------------------+-----------------+-----------+--------------------+
| 1  | cmo1.itso.ibm.com     | QEMU            | ok        | ok                 |
| 3  | cmo2.itso.ibm.com     | QEMU            | ok        | ok                 |
| 21 | powerkvm1.itso.ibm.com | QEMU           | ok        | ok                 |
```

```
| 22 | powerkvm2.itso.ibm.com | QEMU            | ok        | ok                |
| 41 | 8205E6C_06A22ER        | powervm         | ok        | ok|
| 61 | 8233E8B_061AA6P        | powervm         | ok        | ok|
+----+------------------------+-----------------+-----------+-------------------+

[root@cmocontroller ~]# nova hypervisor-status-show cmo1.itso.ibm.com
+---------------------+-------------------+
| Property            | Value             |
+---------------------+-------------------+
| ha_status           | ok                |
| hypervisor_hostname | cmo1.itso.ibm.com |
| hypervisor_type     | QEMU              |
| id                  | 1                 |
| maintenance_status  | ok                |
| migrate             |                   |
+---------------------+-------------------+
```

After disabling maintenance mode, PRS can use this hypervisor as a source or target for live VM migration.

6.2 Manual compute and storage expansion

As the time passes, most cloud environments grow and, at some point, are likely to lack adequate capacity or resources. You can run out of storage space or compute hosts. You may try to deal with it by using features such as storage compression or thin provisioning, removing unused volumes or virtual machines, trying to pack more virtual machines on single host, and so on. But this might result in poor performance or violation of the SLA. When there are no other options, you will must physically expand your cloud environment. In this section, we explain how to accomplish this task in terms of compute and storage expansion.

6.2.1 Manual compute expansion

At the time of writing of this book, OpenStack Nova has an obsolete bare metal driver. *Bare metal* deployment means to put several new servers in the rack, boot them via PXE protocol, install OpenStack components on them, and integrate them into the cloud. There is an ongoing OpenStack project called Ironic that will provide such functions, but it is not supported in the current version of Cloud Manager. For more information about Ironic, see the official OpenStack Ironic project wiki:

https://wiki.openstack.org/wiki/Ironic

Without OpenStack Ironic support, for now, the best way to expand your Cloud Manager environment is to add Compute nodes manually or use software such as IBM Extreme Cloud Administration Toolkit (xCAT). xCAT is beyond the scope of this book, so we focus on manual expansion. Depending on what type of hypervisor you have, the manual expansion is done in a different way. The only common step is physical installation of a new server which includes these steps:

1. Physical installation of new servers in the rack cabinet
2. Connecting new servers to power
3. Connecting new servers to LAN network
4. Connecting new servers to SAN network, if needed

After this, the next step includes installation of hypervisors and components that are needed for integration with a cloud. These can be KVM on Intel servers, PowerKVM on PowerLinux servers, or VIOS on POWER servers with PowerVM. All steps for those three virtualization methods are covered in Chapter 4, “Cloud provider installation guidelines and implementation scenarios” on page 195:

- For help with manual addition of Intel KVM Compute nodes, see 4.6, “Add KVM Compute nodes to the cloud” on page 236.
- For help with manual addition of PowerKVM Compute nodes, see 4.7, “Add PowerKVM Compute nodes to the cloud” on page 253.
- For help with manual addition of PowerVM Compute nodes, see 4.8, “Add PowerVM Compute nodes to the cloud” on page 274.

6.2.2 Manual storage expansion

In real-world scenarios, there is no such thing as too much storage. No matter how much storage you provide, there is never enough space. Even with tricks such as compression, deduplication, or thin provisioning, at some point there is an urgent need for physical storage expansion. This can be achieved in several ways, depending on storage software and hardware you use in your environment and on the Compute nodes that are using provisioned storage. In the Cloud Manager environment, the following storage provisioning methods are supported, depending on virtualization type:

- Intel KVM:
  - Local Compute node LVM
  - GPFS on local disks or IBM SAN Volume Controller (SVC) or IBM Storwize volumes (via the Cinder driver)
  - SVC or Storwize through Fibre Channel (via Cinder driver)
  - Network File System (NFS)
- PowerKVM:
  - Local Compute node LVM
  - NFS
- PowerVM:
  - SVC or Storwize through Fibre Channel (via Cinder driver)
  - Shared storage pool on local disks

With all of these possibilities, some of them make more sense than others, of course. For example, using local LVM makes no sense if you want to use PRS and provide high availability or balance loads for your virtual machines, because with virtual machines using the Compute node’s local storage, it is impossible to move them to another Compute node.

Similarly, if you care about performance, using NFS makes no sense, especially on 1Gbit links.

Choosing the correct storage layer for your Cloud Manager environment is a part of initial planning and isn’t discussed in this chapter. Instead, we focus on ways to expand storage for the three most popular storage provisioning methods:

- Local LVM/NFS on KVM or PowerKVM
- GPFS on KVM
- SVC or Storwize on PowerVM or KVM

Manually expand local LVM resources

In this scenario, we assume that your environment uses Compute node local disks or NFS exported from Compute or Controller node local disks. To expand local storage, follow these steps:

1. Add new physical hard disks to the Compute node.
2. Create physical volumes on newly added physical drives by using **`pvcreate`** command.
3. Add previously created physical volume to volume group with your virtual machines instances by using the **`vgextend`** command.
4. Expand logical volume in the extended volume group with the **`lvextend`** command.
5. Expand the file system using expanded logical volume with the **`resize2fs`** command.

After these steps, you should have more space in the file system where your virtual machines instances are located.

Manually expand GPFS file system

The initial installation and configuration of the GPFS cluster, as well as the test lab environment setup, is covered in detail in “Integrate Cinder with the Spectrum Scale (GPFS) file system” on page 395. In this chapter, we focus on the situation when your GPFS clustered file system is full and needs to be expanded. In our test lab, we used SVC volumes as NSD for the GPFS cluster, but you can also use local Compute node hard disk drives. To expand the GPFS clustered file system online, follow these steps:

1. Create and map additional volumes from SVC or Storwize storage to all nodes in the GPFS cluster, as shown in Figure 6-13. You can add physical hard drives to the Compute nodes if you do not use SVC or Storwize storage systems.

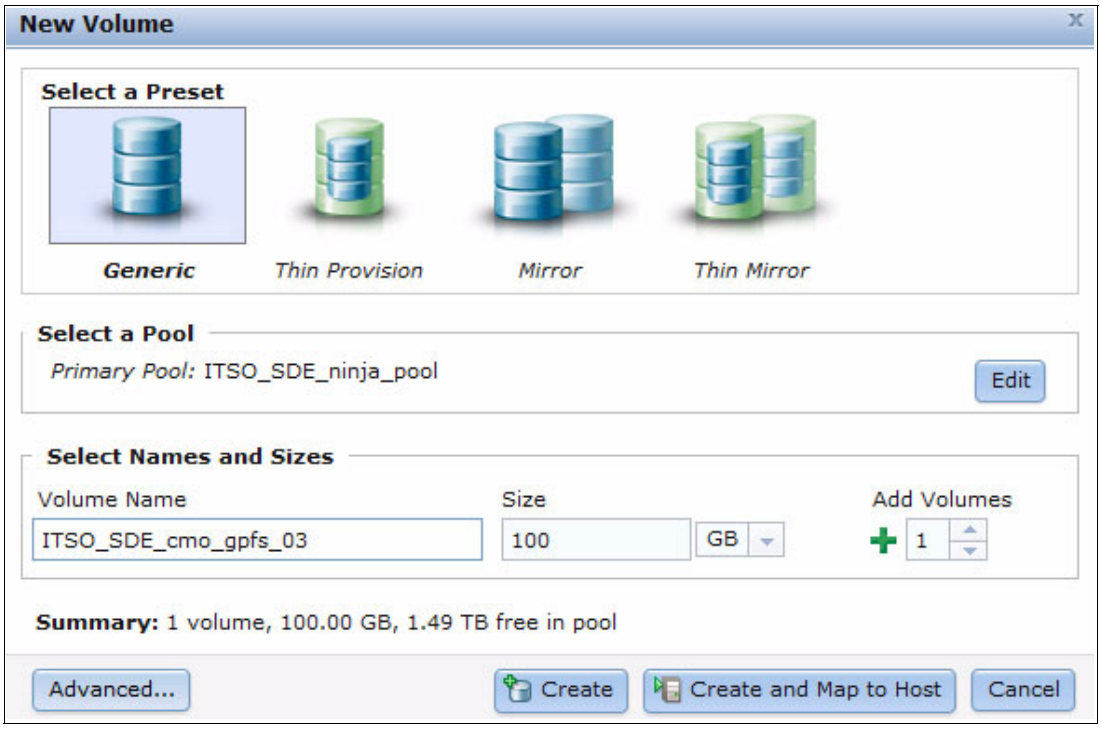


Figure 6-13 Creating new GPFS volume in SVC

Figure 6-14 shows the mapping of the new GPFS volumes to all GPFS cluster nodes.

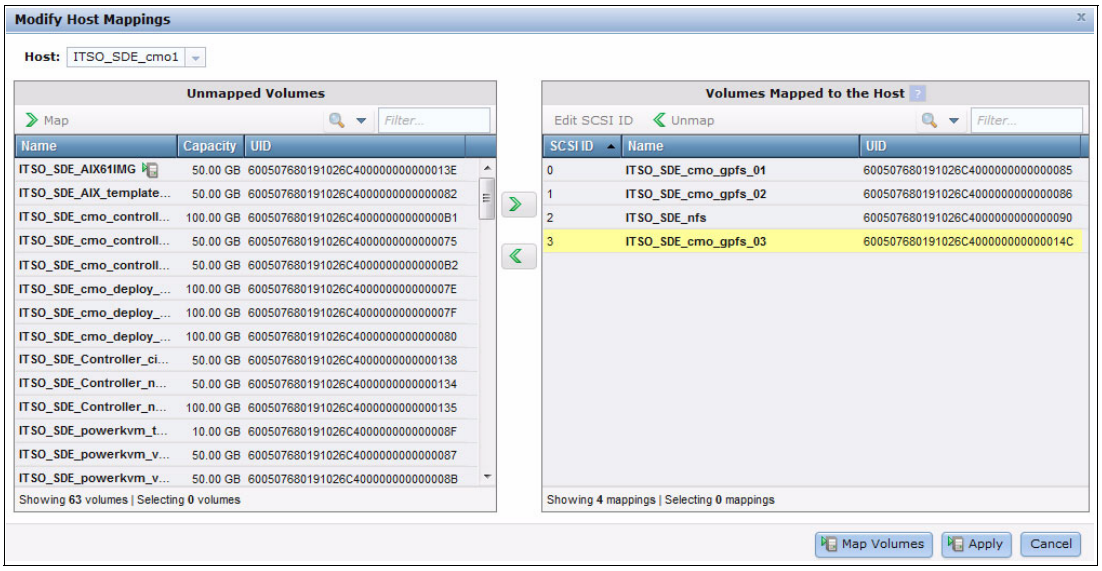


Figure 6-14 Mapping new GPFS volume to all GPFS cluster nodes

2. Detect new volumes on all GPFS cluster nodes by using the **rescan-scsi-bus.sh** command. Check the new volume with the **multipath -ll** command, as shown in Example 6-23.

Example 6-23 Checking the new volumes

```
[root@cmo1 nova]# multipath -ll
mpathd (3600507680191026c4000000000000090) dm-3 IBM,2145
size=200G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| |- 1:0:0:2 sdd  8:48   active ready running
| |- 1:0:1:2 sdg  8:96   active ready running
| |- 2:0:1:2 sds  65:32  active ready running
| `- 2:0:0:2 sdp  8:240  active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 2:0:3:2 sdy  65:128 active ready running
  |- 1:0:3:2 sdm  8:192  active ready running
  |- 1:0:2:2 sdj  8:144  active ready running
  `- 2:0:2:2 sdv  65:80  active ready running
mpathc (3600507680191026c4000000000000086) dm-4 IBM,2145
size=100G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| |- 1:0:0:1 sdc  8:32   active ready running
| |- 1:0:1:1 sdf  8:80   active ready running
| |- 2:0:0:1 sdo  8:224  active ready running
| `- 2:0:1:1 sdr  65:16  active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 1:0:3:1 sdl  8:176  active ready running
  |- 2:0:3:1 sdx  65:112 active ready running
  |- 1:0:2:1 sdi  8:128  active ready running
  `- 2:0:2:1 sdu  65:64  active ready running
mpathb (3600507680191026c4000000000000085) dm-2 IBM,2145
size=100G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
```

```
| |- 1:0:3:0 sdk  8:160  active ready running
| |- 2:0:3:0 sdw  65:96  active ready running
| |- 1:0:2:0 sdh  8:112  active ready running
| `- 2:0:2:0 sdt  65:48  active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 1:0:0:0 sdb  8:16   active ready running
  |- 1:0:1:0 sde  8:64   active ready running
  |- 2:0:0:0 sdn  8:208  active ready running
  `- 2:0:1:0 sdq  65:0   active ready running
mpathg (3600507680191026c400000000000014c) dm-8 IBM,2145
size=100G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| |- 1:0:1:3 sdaa 65:160 active ready running
| |- 1:0:0:3 sdz  65:144 active ready running
| |- 2:0:1:3 sdae 65:224 active ready running
| `- 2:0:0:3 sdad 65:208 active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 1:0:3:3 sdac 65:192 active ready running
  |- 1:0:2:3 sdab 65:176 active ready running
  |- 2:0:3:3 sdag 66:0   active ready running
  `- 2:0:2:3 sdaf 65:240 active ready running
```

3. As you can see, a third 100 GB volume appeared with the name *dm-8*. This is the volume that we have just created and mapped to the GPFS cluster nodes from SVC. Check the current GPFS cluster with several commands, as shown in Example 6-24.

Example 6-24 Checking the new GPFS cluster

```
[root@cmo1 nova]# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         itso-sde-x86.itso.ibm.com
  GPFS cluster id:           6303607005052398228
  GPFS UID domain:           itso-sde-x86.itso.ibm.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name            IP address     Admin node name
Designation
--------------------------------------------------------------------------------
-----------
   1   cmo1.itso.ibm.com           172.16.21.131  cmo1.itso.ibm.com
quorum-manager
   2   cmo2.itso.ibm.com           172.16.21.130  cmo2.itso.ibm.com
quorum-manager
   3   cmocontroller.itso.ibm.com  172.16.21.137  cmocontroller.itso.ibm.com


[root@cmo1 nova]# mmlsnsd

 File system   Disk name    NSD servers
------------------------------------------------------------------------
 sdegpfs1      sdensd1      cmo1.itso.ibm.com,cmo2.itso.ibm.com
 sdegpfs1      sdensd2      cmo1.itso.ibm.com,cmo2.itso.ibm.com
```

```
[root@cmo1 nova]# df -h /gpfs/sdegpfs1
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdegpfs1   200G  3.8G  197G   2% /gpfs/sdegpfs1
```

4. Modify the NSD descriptor file by adding the new disk, dm-8, as shown in Example 6-25.

Example 6-25 Modifying the NSD descriptor file

```
[root@cmo1 GPFS]# cat nsd1
%nsd: device=/dev/dm-2
nsd=sdensd1
servers=cmo1.itso.ibm.com,cmo2.itso.ibm.com
usage=dataAndMetadata

%nsd: device=/dev/dm-4
nsd=sdensd2
servers=cmo1.itso.ibm.com,cmo2.itso.ibm.com
usage=dataAndMetadata

%nsd: device=/dev/dm-8
nsd=sdensd3
servers=cmo1.itso.ibm.com,cmo2.itso.ibm.com
usage=dataAndMetadata
```

5. Create new NSD on the new volume by using the modified NSD descriptor file as shown in Example 6-26. Note that GPFS omits the devices that are already part of the cluster.

Example 6-26 Creating the new NSD on the new volume

```
[root@cmo1 GPFS]# mmcrnsd -F nsd1
mmcrnsd: Processing disk dm-2
mmcrnsd: Disk name sdensd1 is already registered for use by GPFS.
mmcrnsd: Processing disk dm-4
mmcrnsd: Disk name sdensd2 is already registered for use by GPFS.
mmcrnsd: Processing disk dm-8
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

6. Check whether the new NSD was added to the cluster, as shown in Example 6-27.

Example 6-27 Checking if the NSD was added to the cluster

```
[root@cmo1 GPFS]# mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 sdegpfs1      sdensd1      cmo1.itso.ibm.com,cmo2.itso.ibm.com
 sdegpfs1      sdensd2      cmo1.itso.ibm.com,cmo2.itso.ibm.com
 (free disk)   sdensd3      cmo1.itso.ibm.com,cmo2.itso.ibm.com
```

7. Add the new NSD to GPFS file system, as shown in Example 6-28.

Example 6-28 Adding the new NSD to the GPFS file system

```
[root@cmo1 GPFS]# mmadddisk sdegpfs1 sdensd3

The following disks of sdegpfs1 will be formatted on node cmo2.itso.ibm.com:
    sdensd3: size 102400 MB
Extending Allocation Map
Checking Allocation Map for storage pool system
Completed adding disks to file system sdegpfs1.
mmadddisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

[root@cmo1 GPFS]# mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 sdegpfs1      sdensd1      cmo1.itso.ibm.com,cmo2.itso.ibm.com
 sdegpfs1      sdensd2      cmo1.itso.ibm.com,cmo2.itso.ibm.com
 sdegpfs1      sdensd3      cmo1.itso.ibm.com,cmo2.itso.ibm.com
```

8. Check the new size of file system, as shown in Example 6-29.

Example 6-29 Checking the new file system size

```
[root@cmo1 GPFS]# df -h /gpfs/sdegpfs1
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdegpfs1   300G  3.9G  297G   2% /gpfs/sdegpfs1
```

9. You can also use the **mmrestripefs** command on the GPFS file system to equally restripe file system data to all disks, including the new NSDs. This should be used mostly for file systems with large and invariant files.

The procedure shown in Example 6-28 allows the GPFS file system to be expanded online. If you want to create a new file system on new NSDs, you need to add another cinder enabled backend in the `/etc/cinder/cinder.conf` file in the Controller node to point to the new file system. Having two GPFS file systems forces two enabled backend definitions and should look similar to Example 6-30.

Example 6-30 Enabling the backend definition

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,gpfs-fast,
gpfs-fast2

[gpfs-fast]
volume_driver=cinder.volume.drivers.ibm.gpfs.GPFSDriver
gpfs_mount_point_base=/gpfs/sdegpfs1/fastpool
gpfs_storage_pool=system
gpfs_sparse_volumes=true
volume_group=gpfs-fast
volume_backend_name=gpfs-fast

[gpfs-fast2]
volume_driver=cinder.volume.drivers.ibm.gpfs.GPFSDriver
gpfs_mount_point_base=/gpfs/sdegpfs2/fastpool
gpfs_storage_pool=system
gpfs_sparse_volumes=true
```

```
volume_group=gpfs-fast2
volume_backend_name=gpfs-fast2
```

Next, you must create a new volume type, associate this type with new enabled backend, and restart cinder services as described in “Integrate Cinder with the Spectrum Scale (GPFS) file system” on page 395. Then, follow the procedure in “Create a Spectrum Scale cluster on the KVM Compute nodes” on page 388. If you already have a GPFS cluster, omit the GPFS cluster creation steps and start with the NSD configuration in Example 6-26 on page 557.

Manually increase SVC or Storwize capacity

The initial configuration and integration of the SVC or Storwize with Cloud Manager through cinder driver is covered in 4.11.2, “Integrate Cinder with SAN Volume Controller or Storwize V7000” on page 367. In this section, we describe steps to increase SVC or Storwize storage space in case its capacity runs out.

SVC or Storwize provisions volumes to Cloud Manager from a logical construct called *storage pool*. The storage pool consists of *mdisks*, which can be RAID arrays or volumes mapped from external storage systems. When you run out of storage for Cloud Manager, that means the storage pool Cloud Manager is using runs low on space. It not necessarily means all SVC/Stories runs out of storage, it just mean this particular storage pool is almost full. The storage pool you use in your Cloud Manager is defined in the `/etc/cinder/cinder.conf` file in the Controller node. In our environment, it looks like Example 6-31. The storage pool our Cloud Manager is using is highlighted.

Example 6-31 Showing the cinder.conf file

```
[storwize_svc_driver]
# Sample cinder.conf stanza for an IBM Storwize SVC volume driver
# To enable, add the stanza name (storwize_svc_driver) to the DEFAULT
# enabled_backends property
volume_driver=cinder.volume.drivers.ibm.storwize_svc.StorwizeSVCDriver
volume_backend_name=ITSO_SVC_8G4
san_ip=172.16.21.70
san_login=superuser
#san_private_key=path-to-private-key
san_password=yourpassword
storwize_svc_volpool_name=ITSO_SDE_ninja_pool
storwize_svc_connection_protocol=fc
storwize_svc_iscsi_chap_enabled=false
storwize_svc_multipath_enabled=true
```

We can check parameters of this storage pool in SVC as shown in Example 6-32. The view is formatted to make it easier to view.

Example 6-32 Checking the SVC storage pool

```
IBM_2145:ITSO_SVC_8G4:superuser>lsmdiskgrp
id name          status mdisk_count vdisk_count capacity extent_size free_capacity
0 SSP_powervc        online      5           4          100.00GB        256
0 1  DS8300_site2_p01 online    8     6        158.50GB         256    122.00GB
2 DS4800_site2_p02   offline    2     4       484.00GB           256    472.00GB
3 MST_mixed          offline    2     0          262.00GB         256    262.00GB
4 ITSO_SDE_ninja_pool degraded_ports 6 43         2.93TB      256         1.49TB
IBM_2145:ITSO_SVC_8G4:superuser>
```

To expand the capacity of a storage pool, the only thing that you need to do is to add new mdisks to it. In case you use SVC and have several external storage systems, you need to map several volumes from those storage systems to SVC. Then, SVC will recognize them as mdisks and you will be able to add them to a particular storage pool.

If you use Storwize, the mdisks can be added in similar way. But because Storwize is also a disk array, you can create RAIDs on internal Storwize disks. These raids are mdisks of an array type and can be added to storage pools. So, whether you use external volumes as mdisks or internal RAIDs as mdisks, both can be added to storage pool to increase its capacity.

Preferred practice: We recommend using SVC or Storwize with the latest firmware (V7.3 at time of writing). This version introduces the EasyTier V3 feature, which automatically rebalances storage pools. The feature moves extents from busy mdisks to the new, empty or not so busy mdisks and makes sure that whole storage pool is balanced. EasyTier V3 also works with three different tiers of disks in the same pool: SDD, SAS, and NL-SAS. Extents that require better performance are moved to faster disks, and extents that do not require such performance are moved to slower disks automatically and online.

If you do not want to expand an existing storage pool but want to create new one, you can do that by creating a new pool in SVC or Storwize and adding several mdisks to it. Then, you must add a Cinder-enabled backend entry in the `/etc/cinder/cinder.conf` file in the Controller node. This will point to the new storage pool and can look similar to Example 6-33.

Example 6-33 Enabling storage backend

```
enabled_backends=lvm_driver,powervc_driver,storwize_svc_driver,
storwize_svc_driver2

[storwize_svc_driver]
# Sample cinder.conf stanza for an IBM Storwize SVC volume driver
# To enable, add the stanza name (storwize_svc_driver) to the DEFAULT
# enabled_backends property
volume_driver=cinder.volume.drivers.ibm.storwize_svc.StorwizeSVCDriver
volume_backend_name=ITSO_SVC_8G4
san_ip=172.16.21.70
san_login=superuser
#san_private_key=path-to-private-key
san_password=yourpassword
storwize_svc_volpool_name=ITSO_SDE_ninja_pool
storwize_svc_connection_protocol=fc
storwize_svc_iscsi_chap_enabled=false
storwize_svc_multipath_enabled=true

[storwize_svc_driver2]
# Sample cinder.conf stanza for an IBM Storwize SVC volume driver
# To enable, add the stanza name (storwize_svc_driver) to the DEFAULT
# enabled_backends property
volume_driver=cinder.volume.drivers.ibm.storwize_svc.StorwizeSVCDriver
volume_backend_name=ITSO_SVC_8G4
san_ip=172.16.21.70
san_login=superuser
#san_private_key=path-to-private-key
san_password=yourpassword
storwize_svc_volpool_name=ITSO_SDE_new_pool
```

```
storwize_svc_connection_protocol=fc
storwize_svc_iscsi_chap_enabled=false
storwize_svc_multipath_enabled=true
```

The next step is to create a new volume type, associate it with the new enabled backend, and restart Cinder services on the Controller node to apply the changes in the Cinder configuration file. This is described in 4.11.2, “Integrate Cinder with SAN Volume Controller or Storwize V7000” on page 367.

Real-time Compression

The other way to gain some storage space is by using SVC or Storwize feature called *Real-time Compression*. This might be the option if you don’t have much spare physical storage. IBM Real-time Compression is a great feature and can reduce space used by up to 80%. It works in real time, so it looks very tempting, but there are several things you must know before you consider enabling Real-time Compression:

- Real-time Compression is a licensed feature
- After enabling Real-time Compression,some portion of CPU and RAM is reserved exclusively for Real-time Compression. This means that performance of SVC or Storwize can drop. However, this is true only for SVC models before the latest DH8 and for the first generation of Storwize V7000. The most recent models of SVC and Storwize V7000 can have additional hardware resources to support Real-time Compression, so performance is not hurt and is even much better.
- Real-time Compression works on a volume basis, which means you enable compression for the volumes and all data in that volume is being compressed. But you cannot enable it on existing volumes. You can create new, compressed volumes. If you want to compress existing volumes, you need to add a compressed copy for that volume. After adding a compressed copy, SVC or Storwize will mirror data form first, original copy to second, new copy. While mirroring, it also compresses the data online, so the new copy will be fully compressed. After both copies are in sync, you can remove the original copy and leave the new, compressed copy. Therefore, a small amount of additional storage will be needed for mirroring the uncompressed volumes to compressed volumes.
- Before using Real-time Compression, the IBM Comprestimator utility should be used to estimate how much savings compression can add. You can find Comprestimator on the following web page:

  http://www.ibm.com/webapp/set2/sas/f/comprestimator/home.html
- To understand Real-time Compression better and to find out whether it fits your needs, see these IBM Redpapers:
  - *IBM Real-time Compression in IBM SAN Volume Controller and IBM Storwize V7000*, REDP-4859

    http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/redp4859.html
  - *Implementing IBM Real-time Compression in SAN Volume Controller and IBM Storwize V7000*, TIPS1083

    http://www.redbooks.ibm.com/abstracts/tips1083.html

7

Cloud user deployment and use scenarios

This chapter provides an explanation and demonstration of cloud user scenarios in the IBM Software Defined Environment lab. The scenarios focus on the cloud user, who is typically in the line of business division as a developer. This chapter also explains and illustrates working IBM BigInsights scenario transactions for a sample companies called MyBank and Acme Air.

In this chapter, we also introduce SDE capabilities for autoscaling, high availability for applications, and service chaining in Distributed Overlay Virtual Etherne (DOVE) networks for security and protection.

This chapter includes the following sections:

- 7.1, “Heat engine and HOT templates” on page 564
- 7.2, “IBM UrbanCode Deploy with Patterns” on page 583
- 7.3, “IBM BigInsights big data analytics software” on page 655
- 7.4, “Acme Air scenario” on page 668
- 7.5, “High availability for applications” on page 695

© Copyright IBM Corp. 2015. All rights reserved.

7.1 Heat engine and HOT templates

The OpenStack project implements a program to manage the lifecycle of infrastructure and applications in an OpenStack cloud, called the *OpenStack Orchestration* program. The main component of this program is the *Heat*, which implements an orchestration engine to deploy multiple components to your cloud based on templates. These components deployed by the Heat engine can be instances, network connections, IP addresses, volumes, and applications (using scripts to execute in the instance to download, install and configure applications). The templates used by Heat are created as text files that can be treated as source codes. These templates are called *Heat Orchestration Templates (HOT)*.

IBM Cloud Manager (formerly IBM Cloud Manager with OpenStack) includes a Heat engine, which allows you to use HOT templates to deploy a *stack*. A stack is a set of instructions that will be deployed in your cloud, including deploying the instances (virtual machines) using a certain flavor and image, creating volumes and attaching them to the instances, connecting the instances to networks, injecting SSH key pairs to the instances, downloading software and executing scripts to install them, and so on. We can summarize with these definitions:

HOT	Human-readable and writable template as a text file to provide instructions to the Heat engine
Heat	OpenStack engine that deploys your stack
Stack	Set of actions that you want to complete in your cloud

For more information about Heat, see the OpenStack wiki:

https://wiki.openstack.org/wiki/Heat

You can use the instructions from Chapter 4, “Cloud provider installation guidelines and implementation scenarios” on page 195, to create instances and volumes, connect the instances to networks, Attach volumes to the instances, inject SSH key pairs, but the HOT templates and Heat engine will allow you to automate these steps. This section describes the HOT template structure and how you can create a HOT template and use the Heat engine in the IBM Cloud Manager to deploy your stack.

7.1.1 HOT template structure

HOT templates are defined as YAML files. YAML is a human readable data serialization format inspired in XML, Python, C, and Perl programming languages. The acronym YAML means “YAML Ain’t Markup Language”. Example 7-1 shows the main HOT template structure components in YAML format.

Example 7-1 HOT template structure

```
heat_template_version: 2013-05-23

description:
  # a description of the template

parameters:
  # declaration of input parameters

resources:
  # declaration of template resources
```

Note: At the time of writing this publication, the only `heat_template_version` that can be used in IBM Cloud Manager is 2013-05-23.

heat_template_version You must provide the HOT template version, so the Heat engine knows how to interpret the template file.

description You can optionally provide a description of what your HOT template will deploy in your cloud. Although the description is optional, you must still add the `description` field in the HOT template, even if you do not want to provide a description.

parameters You can create a `parameters` section to specify parameters that will be used in the `resources` section when deploying your HOT template. This section is optional and can also be omitted.

resources You must provide the resources that you want to deploy in your cloud in this section. This includes the instances that will be deployed and information such as the flavor it will use, network that it will be connected to, volumes to attach to the instance and SSH key-pair to inject. Each of these parameters can be obtained from the `parameters` section. You must specify at least one resource in this section to make it a valid HOT template.

Each parameter specified in the `parameters` section should be added in a separate block, starting with the name of the parameter, followed by the additional attributes, as shown in Example 7-2.

Example 7-2 Adding parameters to the HOT template

```
parameters:
  <param name>:
    type: <string | number | json | comma_delimited_list | boolean>
    label: <human-readable name of the parameter>
    description: <description of the parameter>
    default: <default value for parameter>
    hidden: <true | false>
    constraints:
      <parameter constraints>
```

Example 7-3 shows a valid example of parameters section in a HOT template. Four parameters are defined in this section, including the SSH key name to inject in the instances during deployment, the image name to use to deploy instances, the flavor to use for each instance, and the network that the instance will be connected to.

Example 7-3 Sample parameters section in a HOT template

```
parameters:
  key_name:
    type: string
    description: Name of keypair to assign to servers
    default: testkey
  image:
    type: string
    description: Name of image to use for servers
    default: rhel65-cloudinit_powerkvm
  flavor:
    type: string
```

```
    description: Flavor to use for servers
    default: m1.small
  private_net_id:
    type: string
    default: d828cdb1-e926-4f94-b276-29a41471e5b6
```

In the `resources` section, you should specify each resource to create in your cloud when the stack is deployed. The resources should also be placed in text blocks to specify each of their properties, as shown in Example 7-4.

Example 7-4 Specifying resources in a HOT template

```
resources:
  <resource ID>:
    type: <resource type>
    properties:
      <property name>: <property value>
    metadata:
      <resource specific metadata>
    depends_on: <resource ID or list of ID>
    update_policy: <update policy>
    deletion_policy: <deletion policy>
```

You can get the list of the valid resource types and their properties in the following website:

http://docs.openstack.org/developer/heat/template_guide/openstack.html

Example 7-5 shows a `resources` section where the parameters from Example 7-3 on page 565 are used to create an instance called *pkvm_vm1*.

Example 7-5 Sample resources section in a HOT template

```
resources:
  server1:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      name: pkvm_vm1
      image: { get_param: image }
      flavor: { get_param: flavor }
      networks:
        - network: { get_param: private_net_id }
```

Important: When indenting the HOT template, you are not allowed to use the Tab key. Use spaces to indent the YAML file. If you use a tab, you will get the following error message when trying to create a stack using this HOT template:

```
An unknown problem occurred validating the template.
```

Additional information and guidance for creating HOT templates are available at the following websites:

http://docs.openstack.org/developer/heat/template_guide/hot_guide.html
http://docs.openstack.org/developer/heat/template_guide/hot_spec.html

7.1.2 Creating a sample HOT template

Using a HOT template, you can create a stack to automate the deployment of multiple components in your cloud. This section explains how to create a HOT template, starting from a basic example and then increasing the details and components. It provides guidance to create your own HOT template with the most-used components, based on your needs. The following topics are covered:

- "Create an instance"
- "Specify the network connection" on page 569
- "Inject the SSH key pair" on page 570
- "Using the parameters section" on page 572
- "Creating Cinder volumes and attaching them to an instance" on page 574
- "Executing scripts in the instance after its deployment" on page 577

Create an instance

To use a HOT template to create an instance, use the following instructions:

1. Open a simple text editor, such as **`vi`** or **`notepad`**, and create a new document with the HOT template structure, as shown in Example 7-6. Save the file with the `.yaml` extension (for example, `sample_hot_template.yaml`).

   Example 7-6 Sample HOT template structure

   ```
   heat_template_version: 2013-05-23

   description: >

   resources:
   ```

2. Do not modify the `heat_template_version` section. As explained in 7.1.1, "HOT template structure" on page 564, this section provides to the Heat engine the information about the version of the template. At the time of writing this publication, the only version supported in IBM Cloud Manager is 2013-05-23.
3. Provide a description to your HOT template, as shown in Example 7-7.

   Example 7-7 Adding a description to your HOT template

   ```
   heat_template_version: 2013-05-23

   description: >
     Sample HOT Template

   resources:
   ```

4. In the `resources` section, add a component called `server1` (you can provide a different name) with the `type OS::Nova::Server` and provide the properties of the instance (`name`, `image` and `flavor` are required properties), as shown in Example 7-8.

   Example 7-8 Adding an instance to the HOT template

   ```
   heat_template_version: 2013-05-23

   description: >
     Sample HOT Template

   resources:
     server1:
   ```

```
type: OS::Nova::Server
properties:
  name: kvm_vm1
  image: rhel_x86_cloud_init
  flavor: m1.small
```

Important: If you have more than one network in your IBM Cloud Manager with OpenStack, you must insert `networks` in the `properties` section of the HOT template.

You can find the available images of the IBM Cloud Manager in the Cloud Management dashboard under **Project > Compute > Images**, as shown in Figure 7-1.

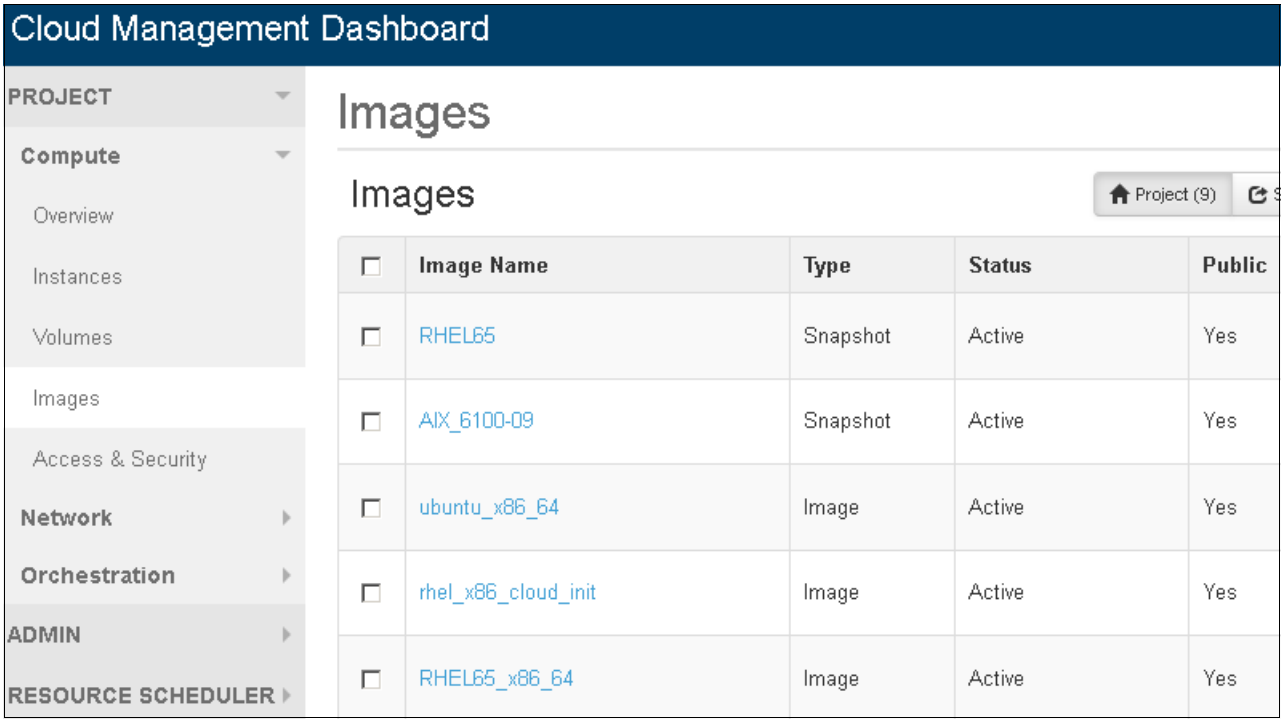


Figure 7-1 Obtaining image names in the Cloud Management dashboard

You can also get this information by using the command-line interface, as shown in Example 7-9.

Example 7-9 Obtaining image names using command-line interface

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# glance image-list
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
| ID                                   | Name                            | Disk Format | Container Format | Size        | Status |
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
| b3064622-6215-4c1b-b2a7-fdbb3e48e9fa | AIX_6100-09                     | raw         | bare             | 53687091200 | active |
| 850fa378-1dc2-4b34-b710-583f301dd12e | Fedora_20_x86_64                | qcow2       | bare             | 210829312   | active |
| 4e06e2c7-389c-4751-93a8-1e47d60b0c5b | ITSO_SDE_AIX_Template_capture_1 | raw         | bare             | 53687091200 | active |
| 63e08c8c-9334-4215-879e-c40e0f681406 | ITSO_SDE_capture_1              | raw         | bare             | 53687091200 | active |
| 354a2183-cdb7-45f6-bfd0-4ce3a4c1025c | PowerVC Default Image           | raw         | ovf              |             | queued |
| c2e714ca-37e9-4a99-8ad7-d7bf10cad4fb | RHEL65                          | raw         | bare             | 53687091200 | active |
| df5f6a15-8c21-413c-a577-7f020bc96e2b | RHEL65_x86_64                   | qcow2       | bare             | 338427904   | active |
| 88d74f68-a0b5-499e-bc31-ba39d5aa839b | cirros                          | qcow2       | bare             | 9761280     | active |
| 9af3d32a-80c7-4865-8173-4756cd70e5b6 | rhel65-cloudinit_powerkvm       | qcow2       | bare             | 1893203968  | active |
| 094ce631-2814-42a5-9a78-f202ab7690ca | rhel65ppc64_powerkvm            | qcow2       | bare             | 10737418240 | active |
| f98aead8-ebbc-4cea-901c-a72b97366487 | rhel_x86_cloud_init             | qcow2       | bare             | 6552420352  | active |
| 4daa0c00-3bf9-42aa-85ba-d5e67ba70cce | ubuntu_x86_64                   | qcow2       | bare             | 255394304   | active |
+--------------------------------------+---------------------------------+-------------+------------------+-------------+--------+
```

You can see the available flavors of the IBM Cloud Manager in the Cloud Management dashboard under **ADMIN > System Panel > Flavors**, as shown in Figure 7-2.

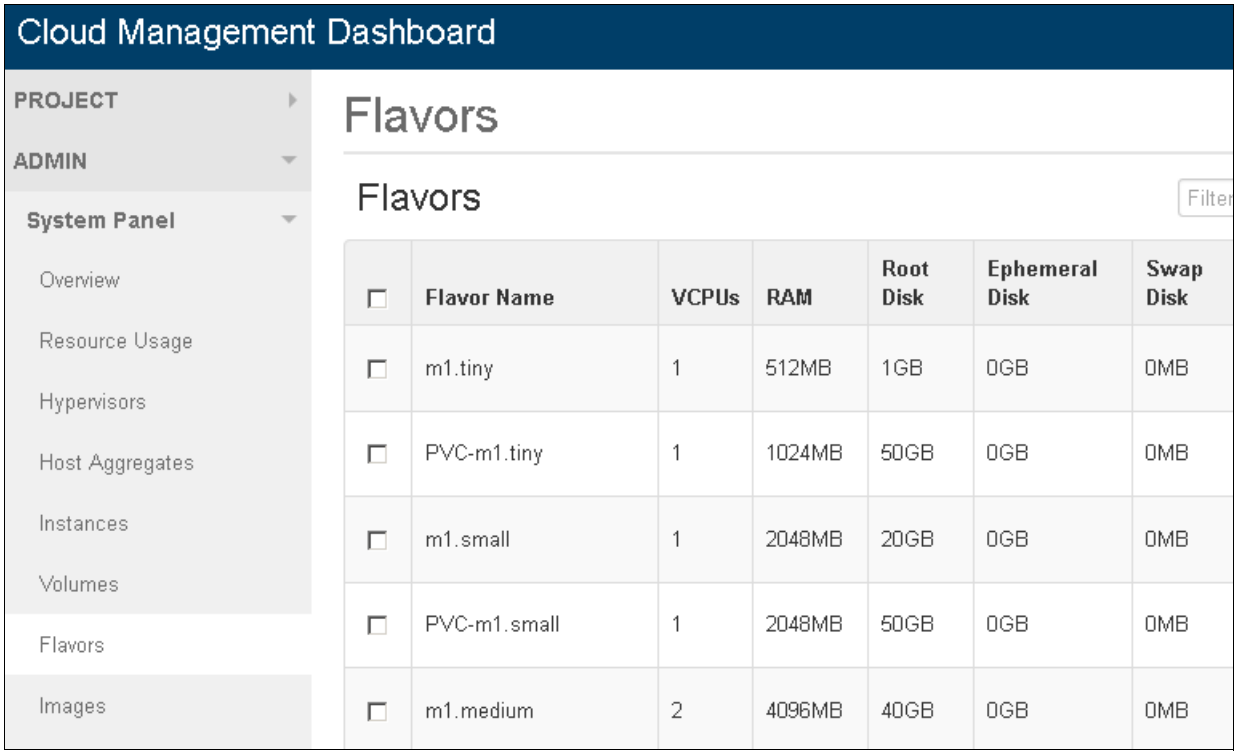


Figure 7-2 Obtaining flavors in Cloud Management dashboard

The same information can also be obtained using the command-line interface with the following commands:

```
# source /root/openrc
# nova flavor-list
```

Specify the network connection

If you have a single network in your IBM Cloud Manager with OpenStack, it can be automatically used by your instance when you create your stack (when you deploy the HOT template). If you have more than one network, you must specify which network the instance will use in the HOT template using the following instructions:

Note: If you have multiple networks and do not specify the network in the HOT template, the creation of the stack will fail and the following message is displayed in the `/var/log/heat/engine.log` file:

```
2014-10-02 18:43:39.107 3977 TRACE heat.engine.resource BadRequest: NV-7D573CB
Multiple possible networks found, use a Network ID to be more specific. (HTTP
400) (Request-ID: req-e836ee1b-4284-46df-a32c-620e4d802e94)
```

1. Log in to the Controller node using SSH, and run the following commands to obtain the network ID of the network that you plan to use:

   ```
   # source /root/openrc
   # export EDITOR=vi
   # neutron net-list
   ```

Example 7-10 shows the expected output for the **neutron net-list** command.

Example 7-10 Obtaining the network ID

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# export EDITOR=vi
[root@cmocontroller ~]# neutron net-list
+--------------------------------------+---------+-------------------------------------------------------+
| id                                   | name    | subnets                                               |
+--------------------------------------+---------+-------------------------------------------------------+
| d0b0de11-7c0b-496a-af3c-11d9900637a7 | Public  | a31a550c-d26e-4283-a908-fc0c388e845e 172.16.20.0/22   |
| d828cdb1-e926-4f94-b276-29a41471e5b6 | Private | dc3d4c88-e524-41d3-987a-32da93cca88a 192.168.21.0/24  |
+--------------------------------------+---------+-------------------------------------------------------+
```

2. Modify your HOT template to include `networks` in the `properties` section of your instance. In this property, provide your network ID as shown in Example 7-11.

   Example 7-11 Adding networks to your HOT template

   ```
   heat_template_version: 2013-05-23

   description: >
     Sample HOT Template

   resources:
     server1:
       type: OS::Nova::Server
       properties:
         name: kvm_vm1
         image: rhel_x86_cloud_init
         flavor: m1.small
         networks:
           - network: d828cdb1-e926-4f94-b276-29a41471e5b6
   ```

3. Save your modified file.

Inject the SSH key pair

Similarly to what 4.15, “Use SSH key pairs in IBM Cloud Manager” on page 485 shows, you can inject SSH key pair to your instance during its deployment in a HOT template. Follow these instructions:

1. Get the name of the key pair that you want to use in your instance.

   Log in to the Cloud Management dashboard, select **Project > Compute > Access > Security**, and click the **Key Pairs** tab, as shown in Figure 7-3 on page 571.

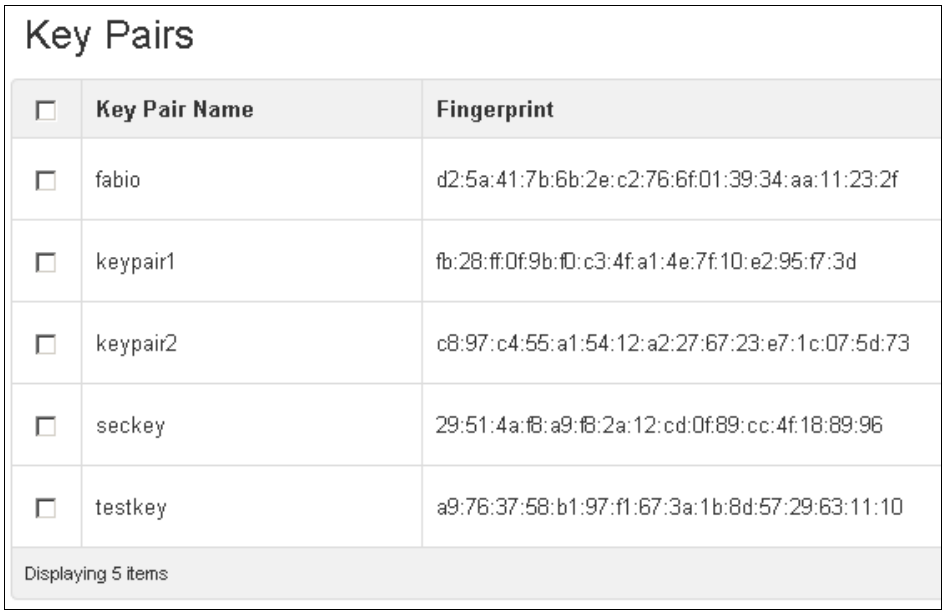


Figure 7-3 Obtaining the key pair name

(Optional) You can get this information by using the command-line interface on the Controller node, as shown in Example 7-12.

Example 7-12 Obtaining the key pair name in command-line interface

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# nova keypair-list
+----------+-------------------------------------------------+
| Name     | Fingerprint                                     |
+----------+-------------------------------------------------+
| fabio    | d2:5a:41:7b:6b:2e:c2:76:6f:01:39:34:aa:11:23:2f |
| keypair1 | fb:28:ff:0f:9b:f0:c3:4f:a1:4e:7f:10:e2:95:f7:3d |
| keypair2 | c8:97:c4:55:a1:54:12:a2:27:67:23:e7:1c:07:5d:73 |
| seckey   | 29:51:4a:f8:a9:f8:2a:12:cd:0f:89:cc:4f:18:89:96 |
| testkey  | a9:76:37:58:b1:97:f1:67:3a:1b:8d:57:29:63:11:10 |
+----------+-------------------------------------------------+
```

2. Modify your HOT template to include the `key_name` under the `properties` section of your instance. If you do not have the metadata service configured in your IBM Cloud Manager (as explained in section “Inject key pairs using metadata service” on page 491), you must use a Configuration Drive to provide the metadata for the instance, so you need to add the `config_drive` option under the `properties` section. Example 7-13 shows the HOT template modified to add the `key_name` and to enable the `config_drive`.

Example 7-13 Adding key-pair and enabling config_drive

```
heat_template_version: 2013-05-23

description: >
  Sample HOT Template

resources:
  server1:
    type: OS::Nova::Server
```

```
    properties:
      key_name: testkey
      name: kvm_vm1
      image: rhel_x86_cloud_init
      flavor: m1.small
      networks:
        - network: d828cdb1-e926-4f94-b276-29a41471e5b6
      config_drive: "true"
```

3. **Save** your modified HOT template file.

Using the parameters section

If you plan to use the HOT template to create multiple instances, you can use the `parameters` section to provide the parameters for the `resources` section. The properties can be obtained from the parameters section rather than typing each property in the instances, so you avoid extra work and possible typing mistakes. Use the following instructions to create the parameters section, where the key_name, image names, network ID, and flavor are specified:

1. Modify your HOT template and include the `parameters` section. Provide each parameter that you plan to use in your instances by using a text block. It must contain the `type` of the parameter, its `default` value, and, optionally, a `description`, as shown in Example 7-14.

Example 7-14 Creating the parameters section

```
heat_template_version: 2013-05-23

description: >
  Sample HOT Template

parameters:
  key_name:
    type: string
    description: Name of keypair to assign to servers
    default: testkey
  image:
    type: string
    description: Name of image to use for servers
    default: rhel65-cloudinit_powerkvm
  image_kvm:
    type: string
    description: Name of image to use for servers
    default: rhel_x86_cloud_init
  flavor:
    type: string
    description: Flavor to use for servers
    default: m1.small
  private_net_id:
    type: string
    default: d828cdb1-e926-4f94-b276-29a41471e5b6

resources:
```

2. Modify the `resources` section to obtain the properties from the `parameters` section by using the **`get_param`** command rather than manually specifying the parameters in the resources section, as shown in Example 7-15.

Example 7-15 Modifying the resources to use the get_param command

```
resources:
  server1:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      name: kvm_vm1
      image: { get_param: image_kvm }
      flavor: { get_param: flavor }
      networks:
        - network: { get_param: private_net_id }
      config_drive: "true"
```

3. If you need to specify more than one instance at the same HOT template, you can include another component in the following line, as shown in Example 7-16.

Example 7-16 HOT template using the parameters section and creating two instances

```
heat_template_version: 2013-05-23

description: >
  Sample HOT Template

parameters:
  key_name:
    type: string
    description: Name of keypair to assign to servers
    default: testkey
  image:
    type: string
    description: Name of image to use for servers
    default: rhel65-cloudinit_powerkvm
  image_kvm:
    type: string
    description: Name of image to use for servers
    default: rhel_x86_cloud_init
  flavor:
    type: string
    description: Flavor to use for servers
    default: m1.small
  private_net_id:
    type: string
    default: d828cdb1-e926-4f94-b276-29a41471e5b6

resources:
  server1:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      name: kvm_vm1
      image: { get_param: image_kvm }
      flavor: { get_param: flavor }
```

```
      networks:
        - network: { get_param: private_net_id }
      config_drive: "true"
  server2:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      name: pkvm_vm2
      image: { get_param: image }
      flavor: { get_param: flavor }
      networks:
        - network: { get_param: private_net_id }
      config_drive: "true"
```

4. Save your modified HOT template file.

Creating Cinder volumes and attaching them to an instance

You can also create Cinder volumes in the HOT template and attach these volumes to an instance:

1. Modify your HOT template. In the parameters section, include the appropriate parameters for the volume size and volume type (to specify which backend storage you want to use). Example 7-17 shows the volume size and type parameters added to the parameters section.

Example 7-17 Adding the volume size and type to the parameters section of the HOT template

```
nfs_volume_size:
    type: number
    description: Size of the volume to be created.
    default: 1
  nfs_vol_type:
    type: string
    default: bc87ed7f-c2b7-44e3-b55f-66026ca6f6d6
```

You can obtain the volume type ID using the **`cinder type-list`** command as shown in Example 7-18.

Example 7-18 Obtaining the volume type ID

```
[root@cmocontroller ~]# source /root/openrc
[root@cmocontroller ~]# cinder type-list
+--------------------------------------+-------------------------------+
|                  ID                  |              Name             |
+--------------------------------------+-------------------------------+
| 230e9aa5-c01d-49df-9d2e-55fb34983ec0 |          gpfs-fast            |
| 74193370-aa30-40e5-bcd4-a7f9b17582c1 | pvc:172_16_21_70:SVC default  |
| 76c8414a-d2d9-4191-b15e-3ea8025b7b66 |             lvm               |
| bc87ed7f-c2b7-44e3-b55f-66026ca6f6d6 |           nfs-cmo             |
| c833de5b-352d-4517-bbd5-fa02e5c5280e |          SVC_Cinder           |
| e659f0bc-5572-4b32-8179-ca7e7e3207ac |            lvmnew             |
+--------------------------------------+-------------------------------+
```

2. In the `resources` section, create a `cinder_volume` component (you can provide a different name) with the type `OS::Cinder::Volume`. You must specify the `size` and `volume_type` properties, as shown in Example 7-19.

Example 7-19 Creating the volume in the HOT template

```
cinder_volume:
    type: OS::Cinder::Volume
    properties:
      size: { get_param: nfs_volume_size }
      volume_type: { get_param: nfs_vol_type }
```

3. Still in the `resources` section, create a `volume_attachment` component (you can provide a different name) with the type `OS::Cinder::VolumeAttachment`. You must provide the properties `volume_id` (the volume that you created in step 2), `instance_uuid` (the instance you created in your HOT template that you want to attach the volume) and `mount point` (the device name that will be used in the instance), as shown in Example 7-20.

Example 7-20 Creating the volume attachment component

```
volume_attachment:
    type: OS::Cinder::VolumeAttachment
    properties:
      volume_id: { get_resource: cinder_volume }
      instance_uuid: { get_resource: server1 }
      mountpoint: /dev/vdc
```

In this example, the get_resource command will obtain the volume_id from the cinder volume created in step 2 and the get_resource command will obtain the instance_uuid of the instance created in 3 on page 573.

Example 7-21 shows the complete HOT template used to create one instance running RHEL 6.5 in KVM Compute node, one instance running RHEL 6.5 in PowerKVM Compute node, one volume in NFS backend, and attaching this volume to the instance from KVM Compute node.

Example 7-21 HOT template with Cinder volumes

```
heat_template_version: 2013-05-23

description: >
  Sample HOT Template with Cinder volumes

parameters:
  key_name:
    type: string
    description: Name of keypair to assign to servers
    default: testkey
  image:
    type: string
    description: Name of image to use for servers
    default: rhel65-cloudinit_powerkvm
  image_kvm:
    type: string
    description: Name of image to use for servers
    default: rhel_x86_cloud_init
  flavor:
    type: string
```

```
    description: Flavor to use for servers
    default: m1.small
  private_net_id:
    type: string
    default: d828cdb1-e926-4f94-b276-29a41471e5b6
  nfs_volume_size:
    type: number
    description: Size of the volume to be created.
    default: 1
  nfs_vol_type:
    type: string
    default: bc87ed7f-c2b7-44e3-b55f-66026ca6f6d6

resources:
  server1:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      name: kvm_vm1
      image: { get_param: image_kvm }
      flavor: { get_param: flavor }
      networks:
        - network: { get_param: private_net_id }
      config_drive: "true"
  server2:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      name: pkvm_vm2
      image: { get_param: image }
      flavor: { get_param: flavor }
      networks:
        - network: { get_param: private_net_id }
      config_drive: "true"
  cinder_volume:
    type: OS::Cinder::Volume
    properties:
      size: { get_param: nfs_volume_size }
      volume_type: { get_param: nfs_vol_type }
  volume_attachment:
    type: OS::Cinder::VolumeAttachment
    properties:
      volume_id: { get_resource: cinder_volume }
      instance_uuid: { get_resource: server1 }
      mountpoint: /dev/vdc
```

4. **Save** your modified HOT template.

Executing scripts in the instance after its deployment

You can also use your HOT templates to execute scripts in the instances after its deployment. These scripts can automate any task that you plan to complete, including the installation of applications. With previously prepared scripts you can deploy a complete environment within a few minutes and through few clicks in your Cloud Management dashboard (deploying instances, creating volumes, attaching them to the instances, creating the LVM and file systems structure, downloading and installing the application).

The scripts are passed to the instances using the cloud-init software, so the metadata service or a config_drive must be used to execute the scripts in the instance after its deployment. The HOT template must specify the `user_data` property in your instance resource section. The `user_data` must use the **`str_replace`** command to look for the `template` section of the cloud-init and pass the script. In summary, you must use the structure Example 7-22 in your instance resource section.

Example 7-22 Instance resource section structure

```
user_data:
        str_replace:
          template: |
            #!/bin/bash -v
             <commands>
          params:
<optional_parameters_to_use_in_the_template_section>
```

Use the following instructions to add the execution of scripts to your HOT template:

1. Modify your HOT template and add the `user_data` in the properties section of your instance. Include the commands that you want to execute in the instance after its deployment. Example 7-23 shows a script that will add a parameter in the `/etc/ssh/sshd_config` file, restart the SSH service, download a sample application from a software repository (HTTP Server), and install this sample application.

   Example 7-23 Executing scripts in the instances using a HOT template

   ```
   resources:
     server1:
       type: OS::Nova::Server
       properties:
         key_name: { get_param: key_name }
         name: kvm_vm1
         image: { get_param: image_kvm }
         flavor: { get_param: flavor }
         networks:
           - network: { get_param: private_net_id }
         user_data:
           str_replace:
             template: |
               #!/bin/bash -v
               LOGFILE=/var/tmp/user_data.log
               echo "Start..." > $LOGFILE
               sed -i "/AcceptEnv XMODIFIERS/a UseDNS no" /etc/ssh/sshd_config
               service sshd restart
               wget http://$software_repository/repos/install_images/app_inst.tgz
               tar -xvf app_inst.tgz
               chmod /root/application_installer.sh
               /root/application_installer.sh
   ```

```
          params:
            $software_repository: { get_param: software_repository }
      config_drive: "true"
```

2. Notice that the **get_param** command was used in the `params` section of the `user_data`, so this parameter needs to be obtained from the `parameters` section of the HOT template. Modify your `parameters` section to provide this information, as shown in Example 7-24.

Example 7-24 Software repository details

```
software_repository:
    type: string
    description: IP address of the software repository
    default: 172.16.21.137
```

Example 7-25 shows the complete HOT template that will create one RHEL 6.5 instance running on KVM, one RHEL 6.5 instance running on PowerKVM, one NFS Cinder volume and attach to the KVM instance, modify the SSH configuration on both instances, download and install the sample application on both instances.

Example 7-25 Sample HOT template with Cinder and user_data

```
heat_template_version: 2013-05-23

description: >
  Sample HOT Template with Cinder volumes

parameters:
  key_name:
    type: string
    description: Name of keypair to assign to servers
    default: testkey
  image:
    type: string
    description: Name of image to use for servers
    default: rhel65-cloudinit_powerkvm
  image_kvm:
    type: string
    description: Name of image to use for servers
    default: rhel_x86_cloud_init
  flavor:
    type: string
    description: Flavor to use for servers
    default: m1.small
  private_net_id:
    type: string
    default: d828cdb1-e926-4f94-b276-29a41471e5b6
  nfs_volume_size:
    type: number
    description: Size of the volume to be created.
    default: 1
  nfs_vol_type:
    type: string
    default: bc87ed7f-c2b7-44e3-b55f-66026ca6f6d6
  software_repository:
    type: string
    description: IP address of the software repository
```

```
    default: 172.16.21.137

resources:
  server1:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      name: kvm_vm1
      image: { get_param: image_kvm }
      flavor: { get_param: flavor }
      networks:
        - network: { get_param: private_net_id }
      user_data:
        str_replace:
          template: |
            #!/bin/bash -v
            LOGFILE=/var/tmp/user_data.log
            echo "Start..." > $LOGFILE
            sed -i "/AcceptEnv XMODIFIERS/a UseDNS no" /etc/ssh/sshd_config
            service sshd restart
            wget http://$software_repository/repos/install_images/app_inst.tgz
            tar -xvf app_inst.tgz
            chmod /root/application_installer.sh
            /root/application_installer.sh
          params:
            $software_repository: { get_param: software_repository }
      config_drive: "true"
  server2:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      name: pkvm_vm2
      image: { get_param: image }
      flavor: { get_param: flavor }
      networks:
        - network: { get_param: private_net_id }
      user_data:
        str_replace:
          template: |
            #!/bin/bash -v
          LOGFILE=/var/tmp/user_data.log
          echo "Start..." > $LOGFILE
          sed -i "/AcceptEnv XMODIFIERS/a UseDNS no" /etc/ssh/sshd_config
          service sshd restart
          wget http://$software_repository/repos/install_images/app_inst.tgz
          tar -xvf app_inst.tgz
          chmod /root/application_installer.sh
          /root/application_installer.sh
        params:
          $software_repository: { get_param: software_repository }
     config_drive: "true"
  cinder_volume:
    type: OS::Cinder::Volume
    properties:
      size: { get_param: nfs_volume_size }
```

```
    volume_type: { get_param: nfs_vol_type }
  volume_attachment:
    type: OS::Cinder::VolumeAttachment
    properties:
      volume_id: { get_resource: cinder_volume }
      instance_uuid: { get_resource: server1 }
      mountpoint: /dev/vdc
```

3. **Save** your modified HOT template.

7.1.3 Create the Stack in Cloud Management dashboard

After completing the creation of your HOT template, you can use the Cloud Management dashboard to create your `Stack`. Through the creation of a Stack, you use the HOT template to provide instructions for the Heat engine to deploy the resources in your Stack. Use the following instructions to create a Stack:

1. Log in to the Cloud Management dashboard, select **Project > Orchestration > Stacks**, and then click **Launch Stack**.
2. In the Template Source field, select **File**. Click **Browse** in the Template File field, select your HOT template file with the `yaml` extension, and then click **Next,** as shown in Figure 7-4.

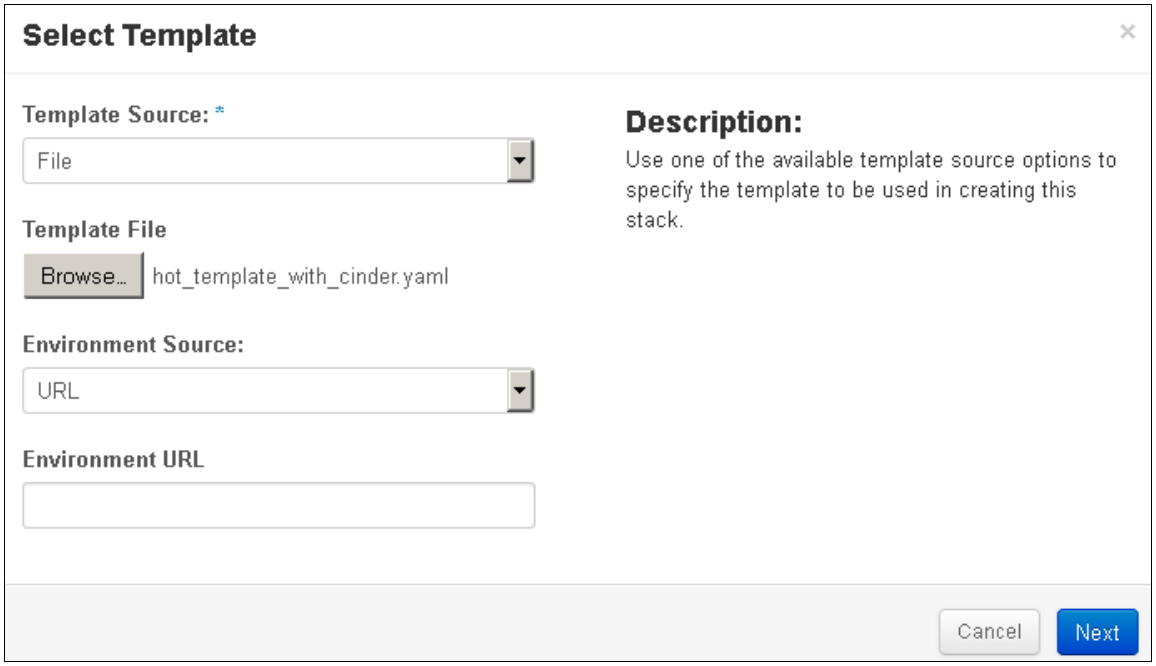


Figure 7-4 Selecting HOT template to create your stack

3. Provide a stack name and the password for the admin user. The remaining fields will be automatically read from your HOT template. Then, click **Launch**, as shown in Figure 7-5.

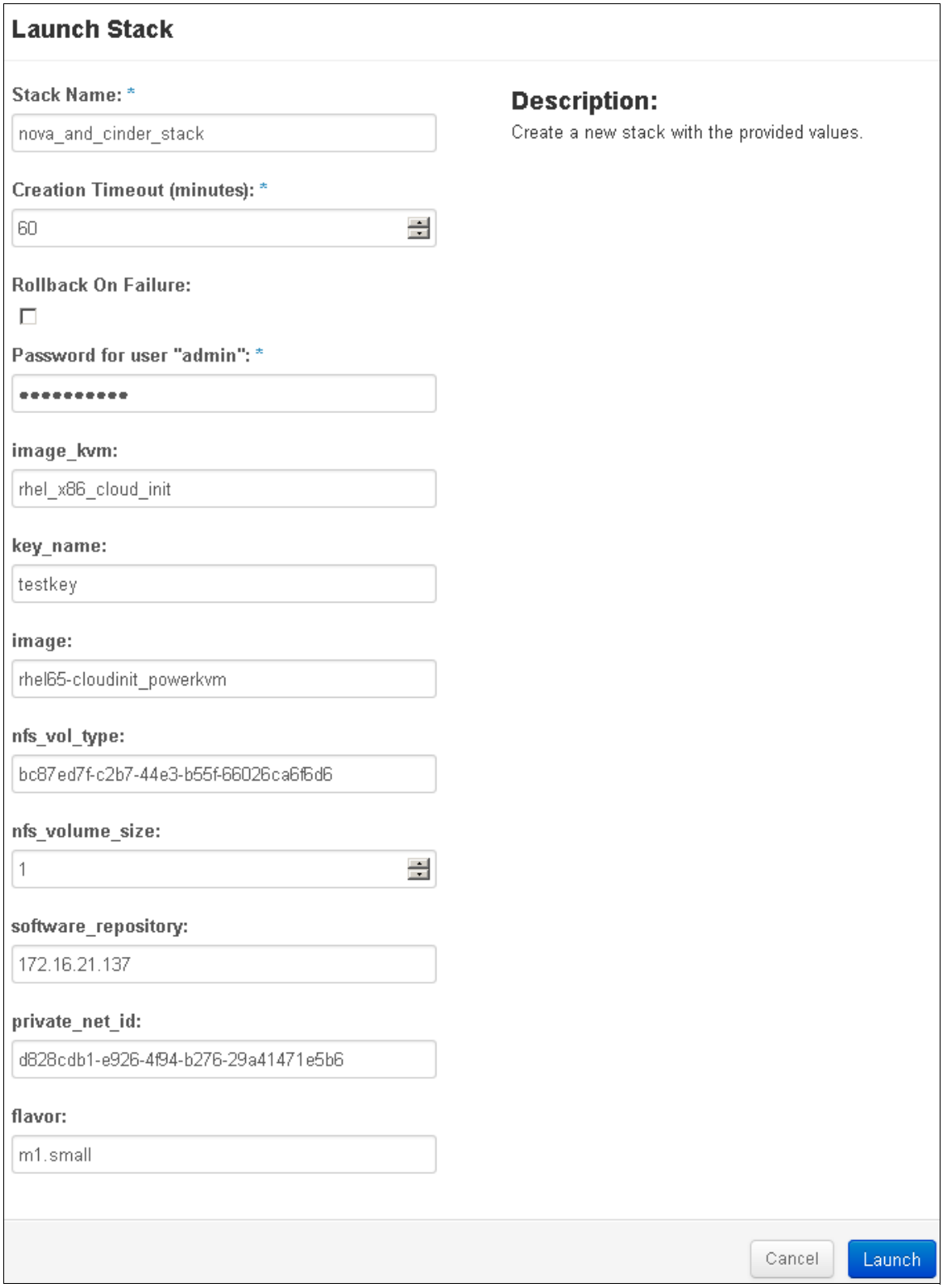


Figure 7-5 Launching a Stack based on a HOT template

Important: If you see the following error when attempting to create your stack, see “Error due to personality property” on page 653:

```
2014-10-07 19:51:23.029 1463 ERROR heat.engine.parser
[req-97d9d907-d451-430e-ba7b-bbea4fa08d37 None] HT-EA96E26 HT-7CD4351 The
personality property may not contain greater than -1 entries.
```

4. The Heat engine will deploy your Stack as shown in Figure 7-6.



Figure 7-6 Stack deployed by the Heat engine

5. Click **Project > Compute > Instances**, and verify that your two instances from the HOT template were deployed. Note the IP address so you can access the instances, as shown in Figure 7-7.

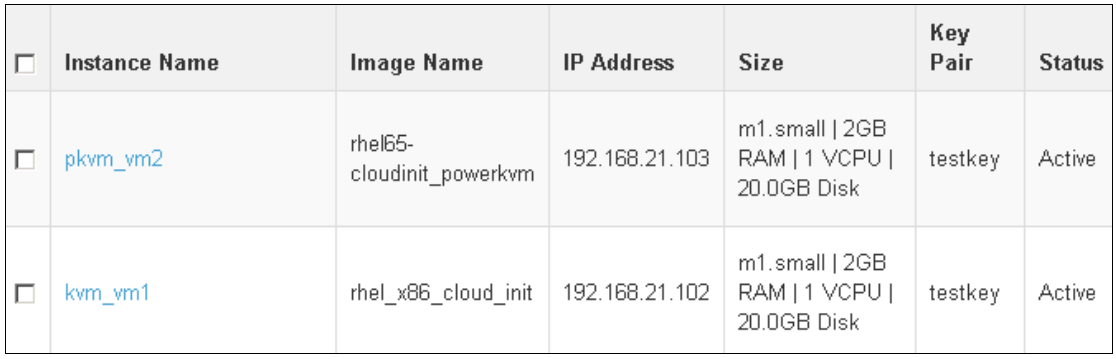


Figure 7-7 Instances from HOT template deployed in the cloud

6. Click **ADMIN > System Panel > Volumes** and verify that a volume was created according to the size and type specified in the HOT template. Verify that this volume is attached to your instance created using the HOT template, as shown in Figure 7-8.



Figure 7-8 Cinder volumes created using HOT template

7. Log in to the instance and verify that the script was executed. You can notice from Example 7-26 that the log was created and /etc/ssh/sshd_config was modified. You can also notice the SSH key pair injection.

Example 7-26 Log in to the instance created using the HOT template

```
[root@cmocontroller ~]# ssh 192.168.21.102
[root@kvm-vm1 ~]# cat /var/tmp/user_data.log
Start...
[root@kvm-vm1 ~]# grep UseDNS /etc/ssh/sshd_config
UseDNS no
#UseDNS yes
```

You can improve your HOT template to provide additional instructions for the instances during the deployment, accomplishing complex implementation and customization tasks through the deployment of a stack.

7.2 IBM UrbanCode Deploy with Patterns

IBM UrbanCode Deploy with Patterns (UCDP) allows you to design, deploy and update full-stack environments on multiple clouds. You can connect it to your OpenStack cloud and use IBM UrbanCode Deploy with Patterns to design your HOT templates and create stacks in your cloud, deploying all the components from the template.

This software also provides a Heat engine that can be used in connection to your OpenStack cloud to deploy your stacks. It provides the following capabilities for your cloud:

- Graphical and text editor to design full-stack environments and HOT templates
- Possibility to deploy full-stack application to multiple clouds
- Ability to update existing environments and complete infrastructure changes
- Integration with UrbanCode Deploy to support the end-to-end pipeline
- Uses open standards defined by OpenStack

With this tool, used with IBM UrbanCode Deploy, infrastructure specialists can prepare components, processes and applications and make it available for Architects and integrators to design and deploy a complete environment. For more information about IBM UrbanCode Deploy with Patterns, see the following website:

https://developer.ibm.com/urbancode/products/urbancode-deploy-with-patterns/

Together with IBM UrbanCode Deploy with Patterns, you can use IBM UrbanCode (UCD) for automating application deployments through your environments. In IBM UrbanCode you create components which are used by UrbanCode Deploy with Patterns to prepare and deploy environments (using HOT templates and Heat engine). With these tools (UCD and UCDP) you can:

- Integrate them to automatically deploy new builds
- IBM Orchestrate®, a complex production deployment of applications and configuration
- Grant to different teams or tenants the rights to deploy environments and applications

For more information about IBM UrbanCode Deploy, see the following website:

https://developer.ibm.com/urbancode/products/urbancode-deploy/

The next steps allow you will to connect the IBM UrbanCode Deploy with Patterns to your IBM Cloud Manager cloud, prepare components in the IBM UrbanCode Deploy and create and deploy environments using these tools. For instructions on how to install IBM UrbanCode

Deploy with Patterns and IBM UrbanCode Deploy, see 4.16, “Developer tool installation: UrbanCode Deploy with Patterns” on page 492.

Note: Remember that both UrbanCode Deploy with Patterns and UrbanCode Deploy are installed onto the same server, but using different Port Numbers, so you will need to login to both interfaces to complete the next tasks:

- IBM UrbanCode Deploy with Patterns web interface: Initial admin user is `ucdpadmin` and initial password is also `ucdpadmin`.

  `http://<ucdp_server_ip_address>:8080/landscaper/login`

- IBM UrbanCode Deploy web interface: Initial admin user is `admin` and initial password is also `admin`.

  `https://<ucdp_server_ip_address>:7443/`

These tasks are divided in the following topics:

- Creating the versioner agent in IBM UrbanCode Deploy
- Connecting IBM UrbanCode Deploy with Patterns to the OpenStack cloud
- Creating components in UrbanCode Deploy
- Deploying environments in UrbanCode Deploy with Patterns with UrbanCode Deploy components

7.2.1 Creating the versioner agent in IBM UrbanCode Deploy

IBM UrbanCode Deploy requires an agent, which is used to import component versions. The agent must have access to component artifacts. If you plan to create components in IBM UrbanCode Deploy and use it in IBM UrbanCode Deploy with Patterns to automate the deployment of applications in your cloud it is a required configuration. The following steps will guide you to create and configure an Agent for Version Imports in UrbanCode Deploy:

1. Log in to the UrbanCode Deploy with Patterns server using SSH. You must modify the SSH configuration file to allow the root user to log in using password Authentication. Edit the `/etc/ssh/sshd_config` file:

   ```
   # vi /etc/ssh/sshd_config
   ```

2. Uncomment the following lines:

   ```
   PermitRootLogin yes
   PasswordAuthentication yes
   ```

3. Restart the SSH daemon:

   ```
   # service sshd restart
   ```

4. Log in to the UrbanCode Deploy web interface as the admin user, as shown in Figure 7-9 on page 585.

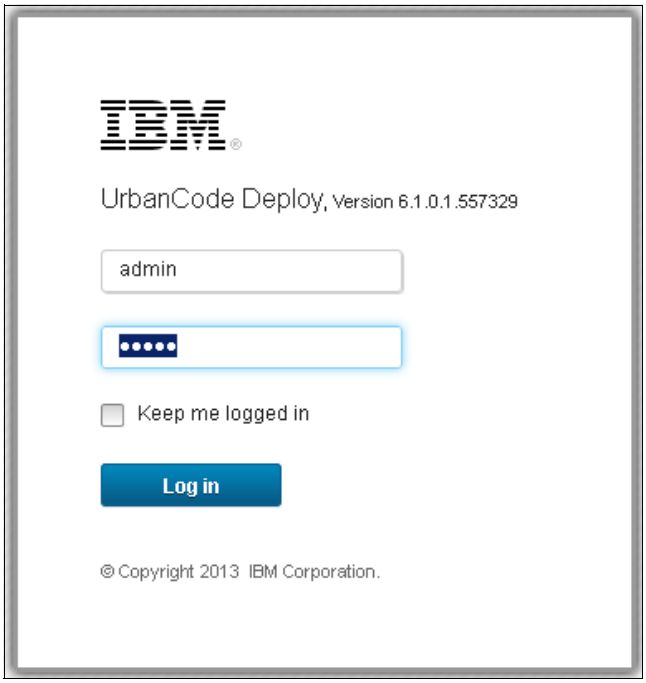


Figure 7-9 Log in to the UrbanCode Deploy web interface

5. You will notice a warning saying that “There is no agent or tag configured to import new component versions, so no new versions will be imported. Please set this on the Settings > System Settings page.” This is the first step required in the UrbanCode Deploy configuration. A Java agent must be created and configured to allow it to import components.
6. To create a new `agent`, select **Resources > Agents** and click **Install New Agent,** as shown in Figure 7-10.

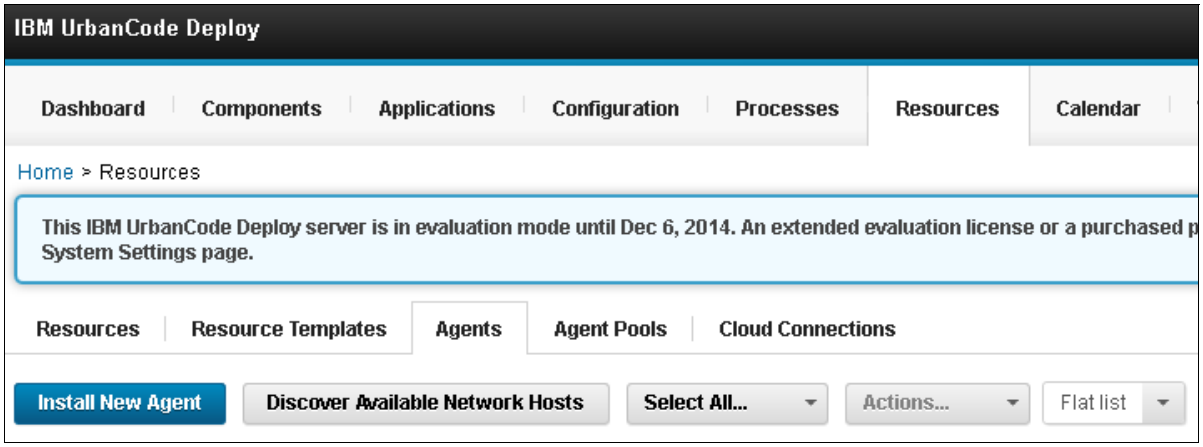


Figure 7-10 Creating Agent in UrbanCode Deploy

7. Select the **SSH in the Install New Agen**t screen, as shown in Figure 7-11.

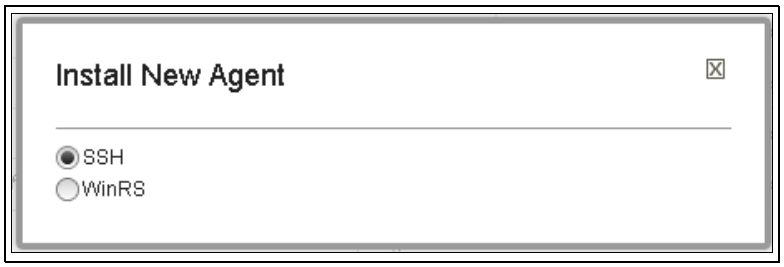


Figure 7-11 Selecting SSH in the New Agent definition

8. Provide the following information in the Install New Agent screen:
   - *Target Hosts.* Provide the UrbanCode Deploy IP address, which will be the same IP address used to access the UrbanCode web interface. This means that the components created in the UrbanCode Deploy are located in the UrbanCode Deploy Server itself.
   - *Username.* Provide the `root` username.
   - *Password.* Provide the `root` password (from the UrbanCode Deploy server).
   - *Agent Name.* Provide a name to identify your agent (in this example, the name used is `Example Versioner`, but it could be any name).
   - Figure 7-12 shows the new agent configuration, with appropriate information completed the mentioned field to create the agent.

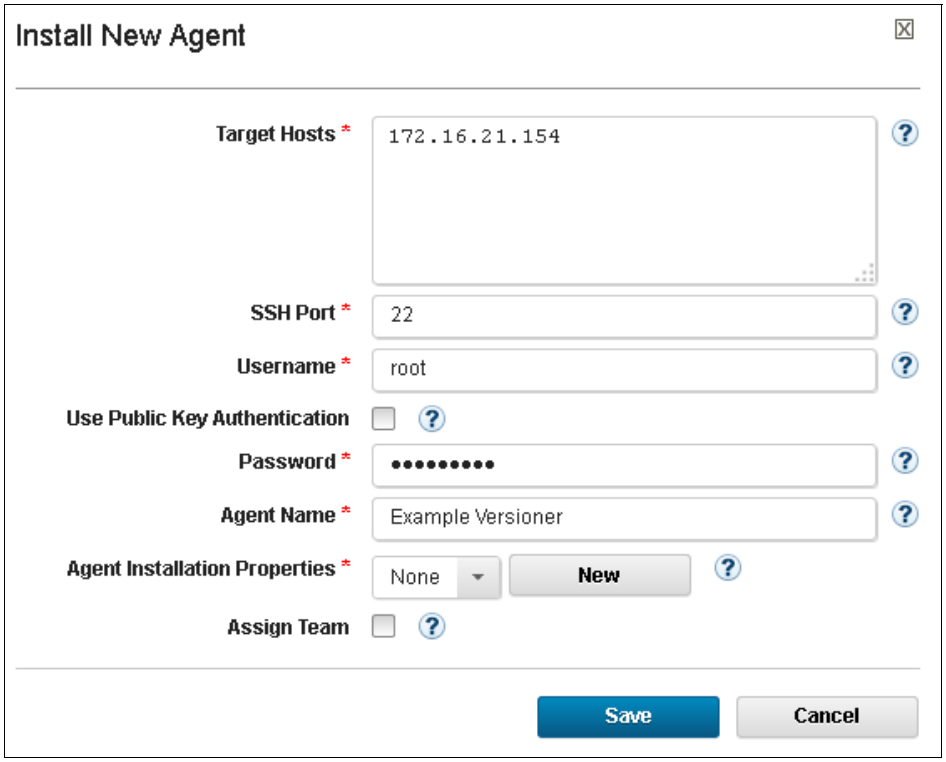


Figure 7-12 Providing the new agent configuration

9. Click the **New** button in the Agent Installation Properties field.
10. Using the command-line interface in the UrbanCode Deploy with Patterns server, run the following commands to get the value of your `JAVA_HOME` variable (this value will be used in the Agent configuration:

```
# source /root/.bashrc
# echo $JAVA_HOME
/usr/lib/jvm/jre-1.7.0-openjdk.x86_64
```

11. Back in the Edit Agent Installation Properties window, which was opened when you clicked New, provide a name for your agent properties in the Name field, provide the value of the `JAVA_HOME` variable in the Java Home Path field, and click **Save**, as shown in Figure 7-13.

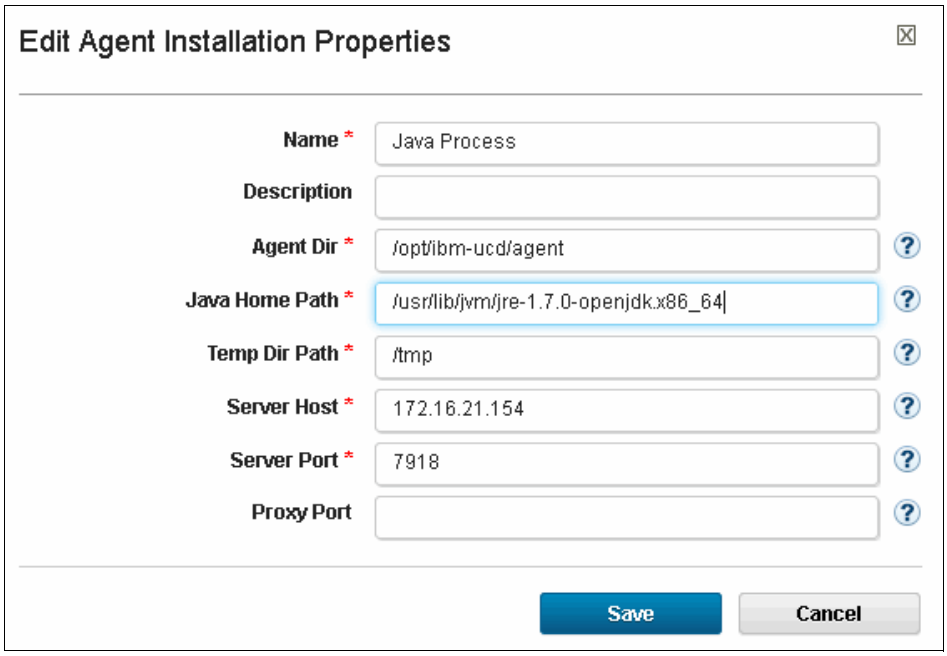


Figure 7-13 Modifying agent installation properties

Important: Make sure that you have completed steps 7.2.1, “Creating the versioner agent in IBM UrbanCode Deploy” on page 584 through 3 on page 584 before clicking the Save button, because the agent will try to log in to the UrbanCode Deploy with Patterns server to validate its functions.

12. Back in the Install New Agent screen, select the **Java Process in the Agent Installation Properties** field and click **Save**, as shown in Figure 7-14.

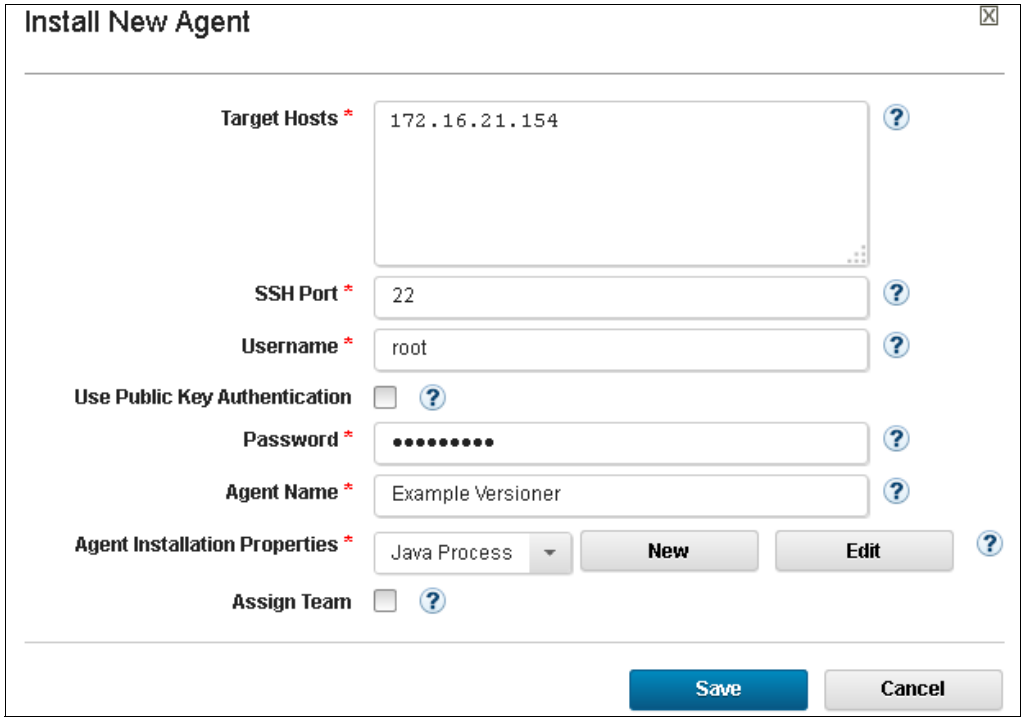


Figure 7-14 Selecting the Java Process in the Agent Installation Properties

13. Now that the Agent is created, you need to configure it on the UrbanCode Deploy system. Click **Settings > System > System Settings**, as shown in Figure 7-15 on page 589.

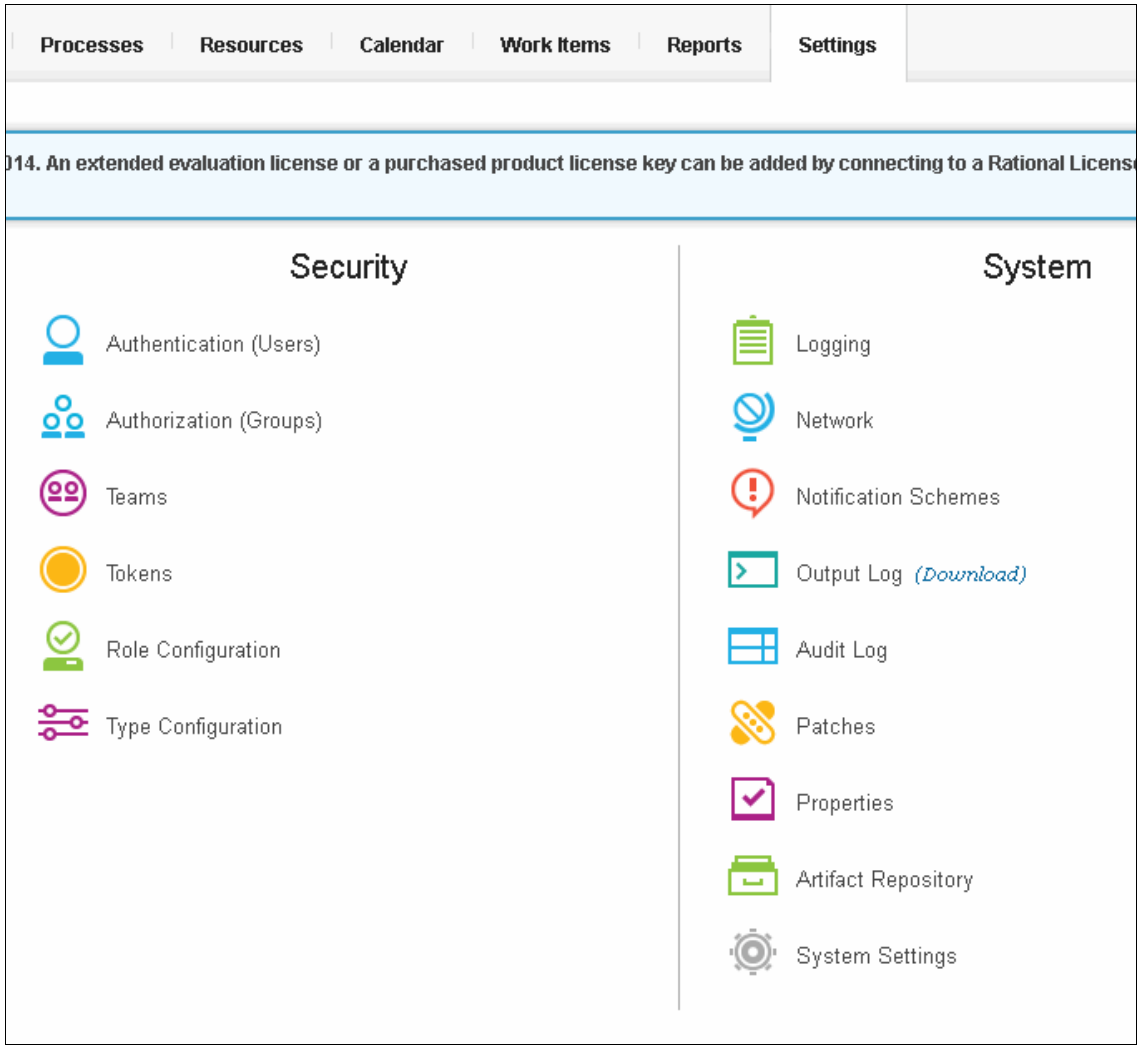


Figure 7-15 Accessing the System Settings configuration in the UrbanCode Deploy Server

14. In the General Settings section, select the **Example Versioner** (or the appropriate name, according to the agent that you have just created) in the Agent for Version Imports field and click **Save**, as shown in Figure 7-16.

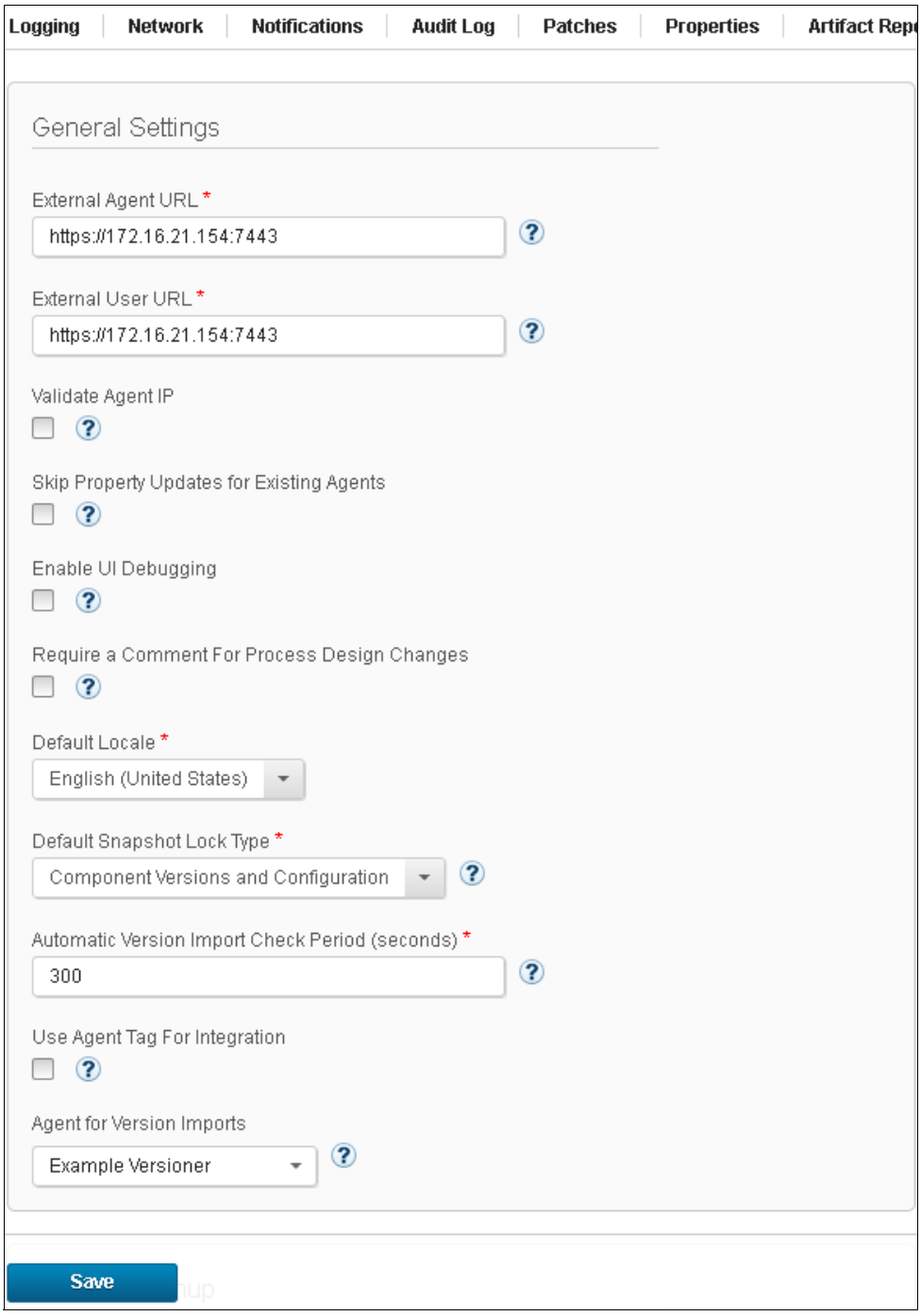


Figure 7-16 Selecting the Versioner Agent in the Agent for Version Imports field

After completing these steps, you should no longer see the warning mentioned in step 5 on page 585. Now IBM UrbanCode Deploy with Patterns will be able to use components created in UrbanCode Deploy to automate the deployment of applications in your OpenStack cloud.

7.2.2 Connect IBM UrbanCode Deploy with Patterns to the OpenStack cloud

To deploy environments in your OpenStack cloud, you must connect the UrbanCode Deploy with Patterns server to the cloud. The next steps will guide you to connect it to the IBM Cloud Manager cloud prepared on Chapter 4, “Cloud provider installation guidelines and implementation scenarios” on page 195:

1. Log in to the UrbanCode Deploy with Patterns web interface:

   `http://<ucdp_server_ip_address>:8080/landscaper/login`

2. On the upper-right corner of the screen, click **Settings** (engine button) and then click **Users** as shown in Figure 7-17.

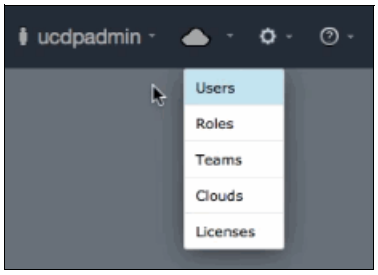


Figure 7-17 Clicking Settings > Users in UrbanCode Deploy with Patterns

3. Click **Create New Realm** and provide the following information:
   - *Name.* Provide a name that identifies your OpenStack cloud that is being added to UrbanCode Deploy with Patterns.
   - *Identity URL.* The OpenStack cloud authentication URL. This information can be obtained from the OS_AUTH_URL variable in the `/root/openrc` file in the Cloud Manager Controller node, as shown in Example 7-27.

Example 7-27 Obtaining the Identity URL from the /root/openrc file in the Controller node

```
[root@cmocontroller ~]# cat /root/openrc

# This file autogenerated by Chef
# Do not edit, changes will be overwritten

# COMMON OPENSTACK ENVS
export OS_USERNAME=admin
export OS_PASSWORD=openstack2
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://172.16.21.137:5000/v2.0
export OS_REGION_NAME=RegionOne
```

Important: The information provided in the Identity URL field must match exactly the information provided in step 5 on page 297 of section 4.16, “Developer tool installation: UrbanCode Deploy with Patterns” on page 492 (during installation time). It must be an exact match, so if you did not place a slash (/) in the end of the link during the installation time, you should also do not place a slash in the Identity URL field.

– *Timeout in Mins.* Increase the default timeout value from 10 minutes to 60 minutes. This timeout value is used during deployment of environments. In case you have a complex application which takes longer than 10 minutes to install, the deployment might fail due to this timeout, so increasing this value allows you to give more time for the application installation to complete.

– *Use default orchestration engine.* Clear this box.

– *Admin Username.* Provide the `admin` user name of the OpenStack cloud (in IBM Cloud Manager the admin user is `admin`). This is the user name to log in to the IBM Cloud Management dashboard.

– *Admin Password.* Provide the password of the `admin` user of the OpenStack cloud (the one used to log in to the Cloud Management dashboard).

– *Admin Tenant.* The admin tenant or project from your OpenStack cloud (in IBM Cloud Manager with OpenStack, the admin tenant is `admin`).

– *Orchestration Engine URL.* The Heat engine URL that will be used by UrbanCode Deploy with Patterns, which should be the UCDP Server IP address, given that IBM UrbanCode Deploy with Patterns provides its own Heat engine. You should only replace the localhost for the IP address and keep the same protocol and port number as specified in the default value:

  `http://<ucdp_server_ip_address>:8004`

Figure 7-18 shows a valid example of information provided to UrbanCode Deploy with Patterns to connect it to your OpenStack cloud. After providing all of the required information, click the **Save** button.

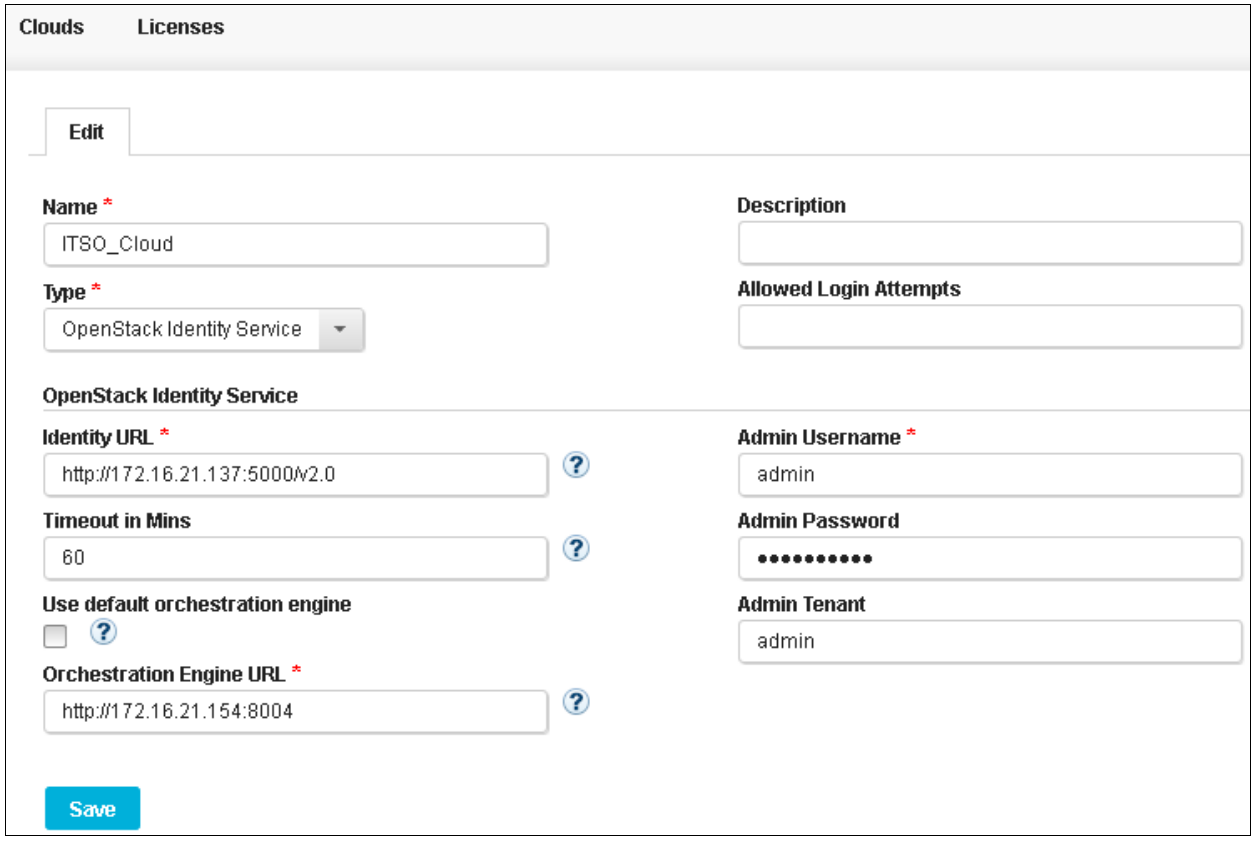


Figure 7-18 Creating the New Realm to connect UCDP to OpenStack cloud

4. Now click **Import User**, and then confirm this action by clicking **Import**, as shown in Figure 7-19.

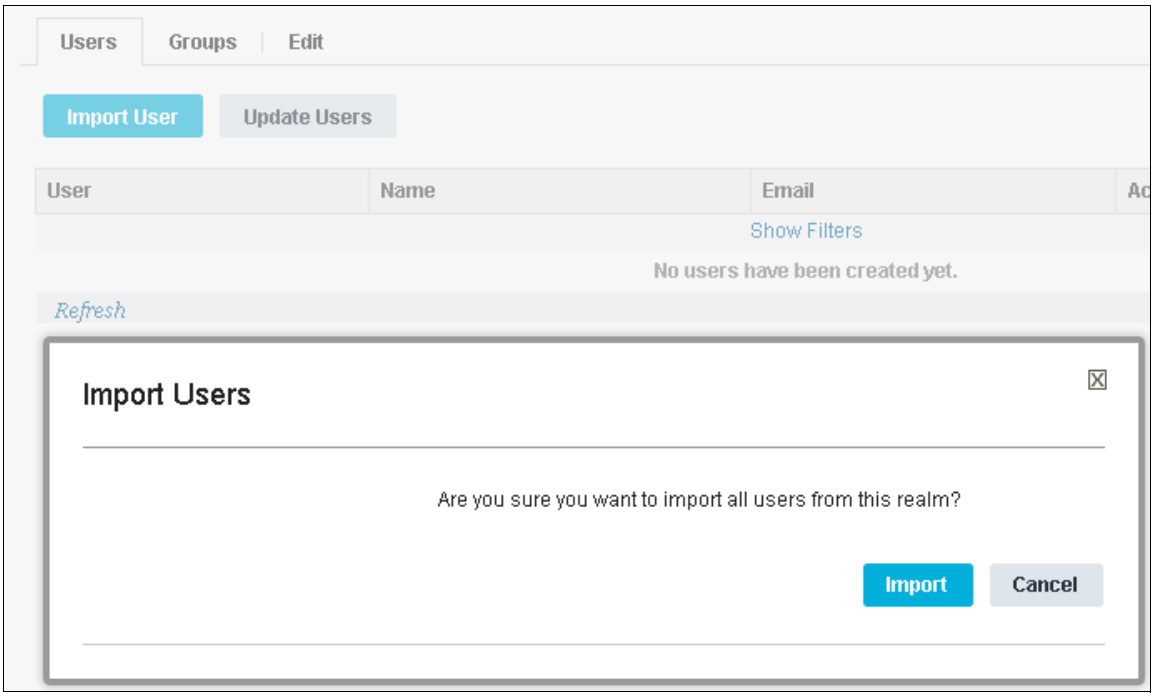


Figure 7-19 Importing users from OpenStack cloud to UrbanCode Deploy with Patterns

If you have corectly provided the correct admin username and admin password in step 3 on page 591, the Cloud Manager users should be displayed, as shown in Figure 7-20.

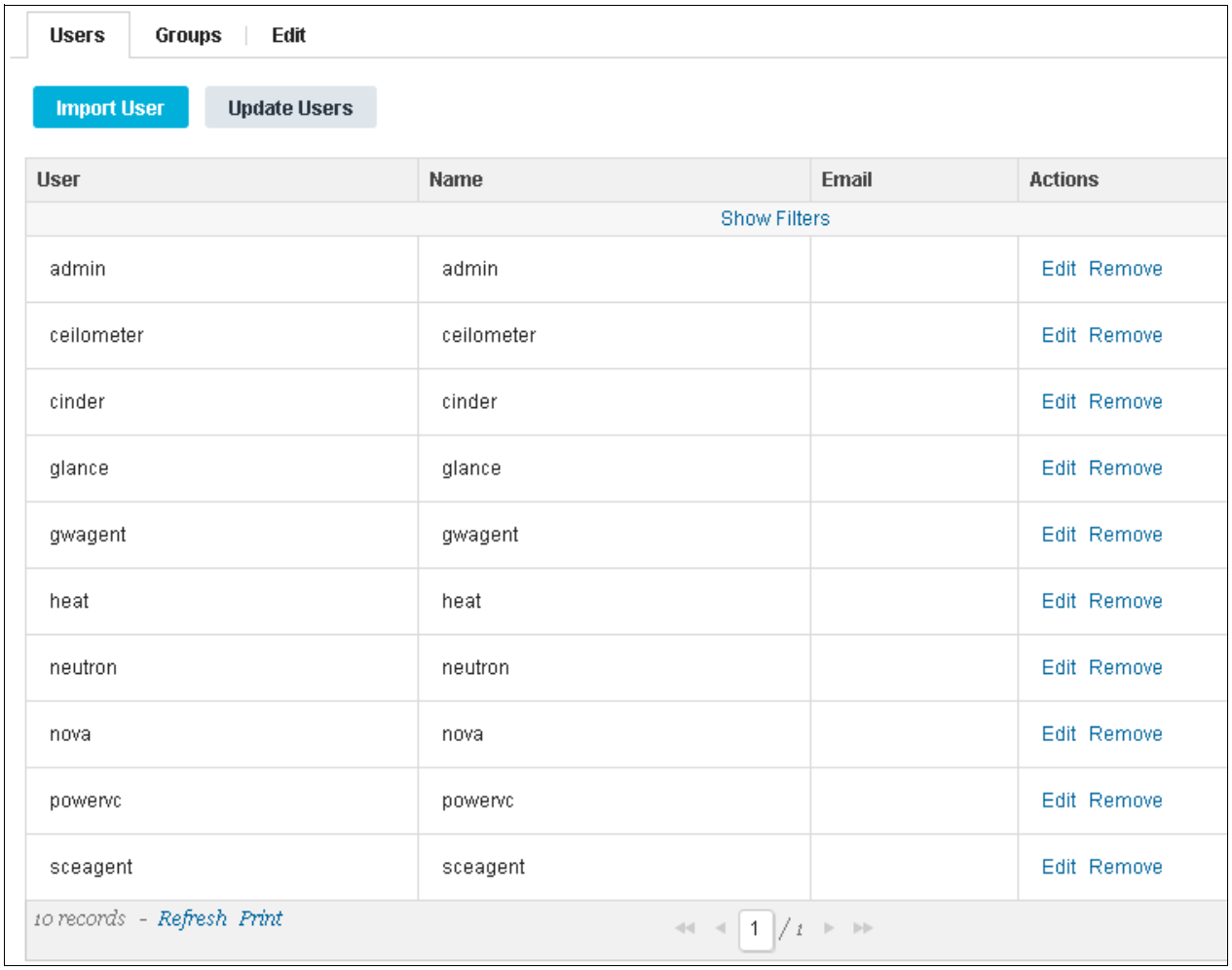


Figure 7-20 Users imported from IBM Cloud Manager onto UrbanCode Deploy with Patterns

5. Now click **Teams** and use the **Add User** drop-down list to select the **admin** user. Enable the **Admin** check box as the Role for the admin user and click **Save**, as shown in Figure 7-21.

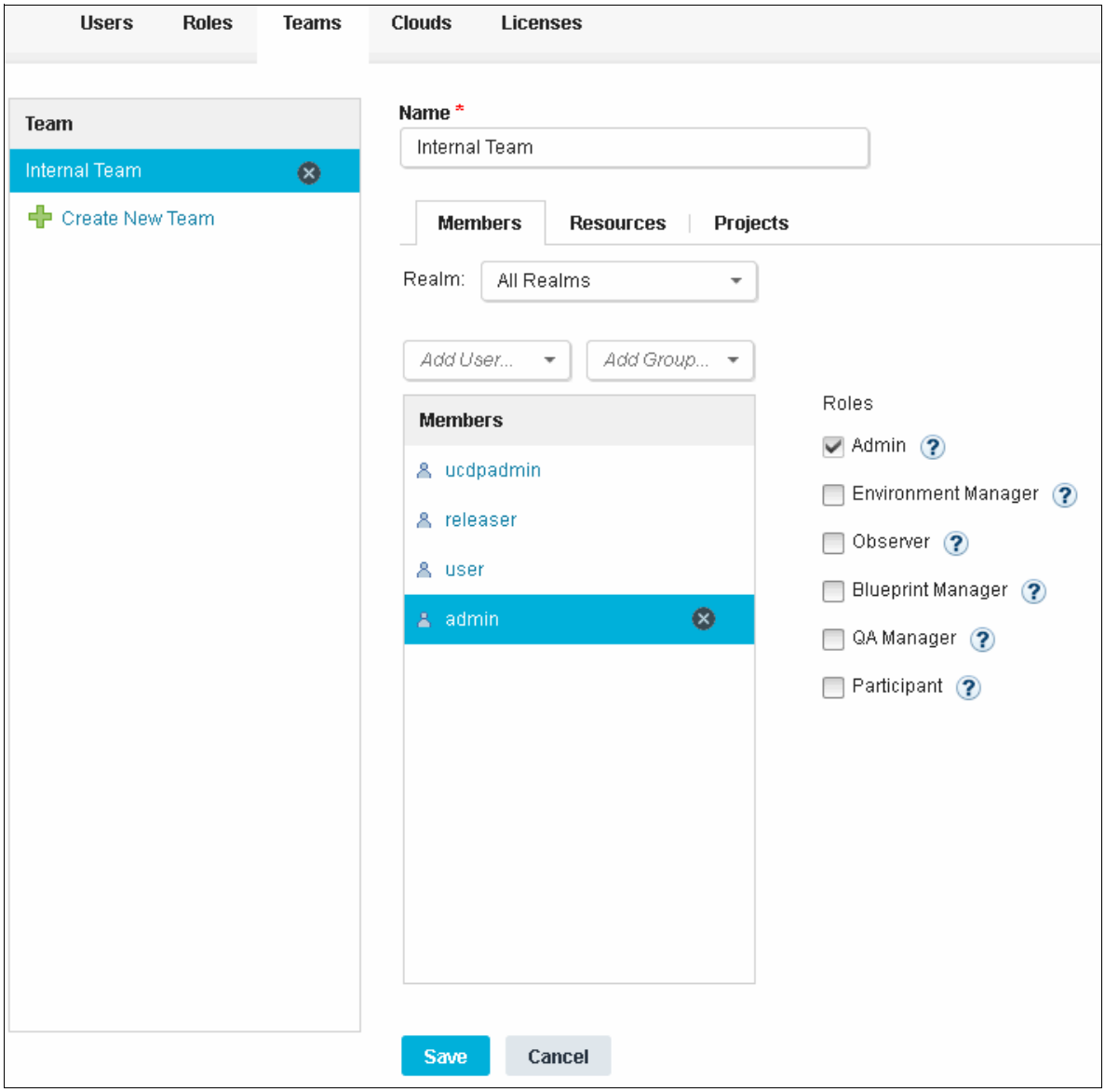


Figure 7-21 Adding admin user as a Team member and assigning the Admin role

6. Now log out and log back in to the UrbanCode Deploy with Patterns web interface, but this time, use the `admin` username and password (same password that you use to log in to the IBM Cloud Manager dashboard), as shown in Figure 7-22.

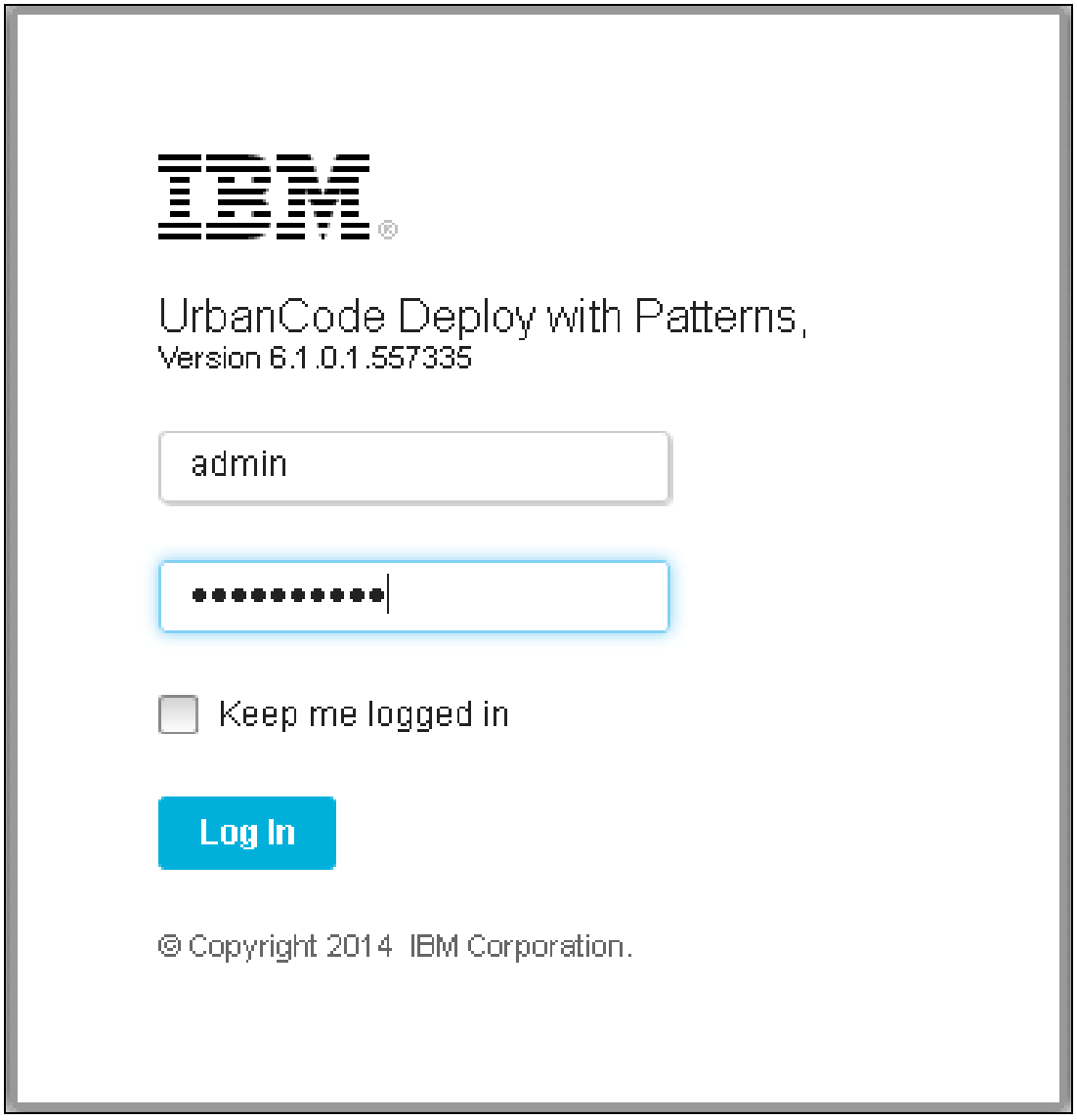


Figure 7-22 Logging in to UCDP using IBM Cloud Manager admin user

7. In the upper-right corner of the screen, click **Settings** (engine button) and then click **Teams**.

8. Now you need to connect the admin user from the IBM Cloud Manager to a project (also from IBM Cloud Manager with OpenStack). Select the **admin** user, click the **Projects** tab, and then click **Add**, as shown in Figure 7-23.

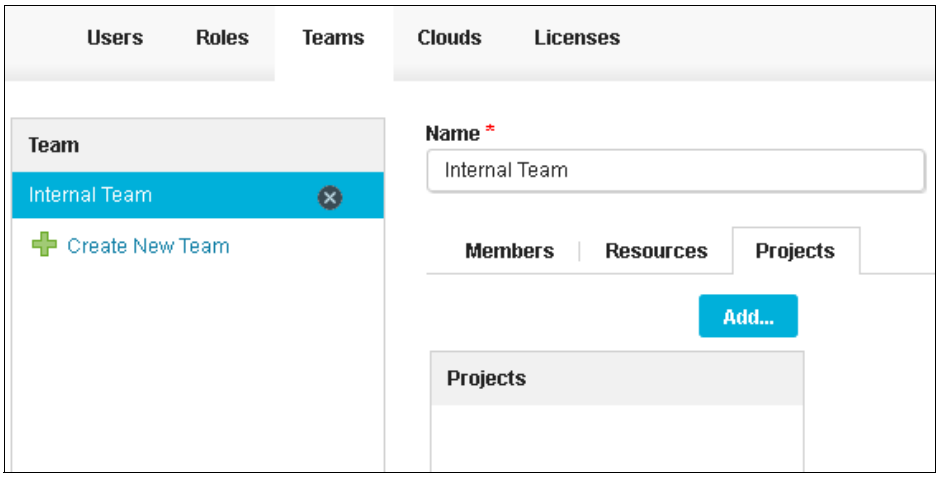


Figure 7-23 Adding the admin user to a project

9. Select the project (or tenant) to add the **admin** user, click **OK** and then **Save**. These are the projects that you can create in IBM Cloud Manager in the Cloud Management dashboard on ADMIN > Identity Panel > Projects. In this case, the **admin** project is selected, as shown in Figure 7-24.

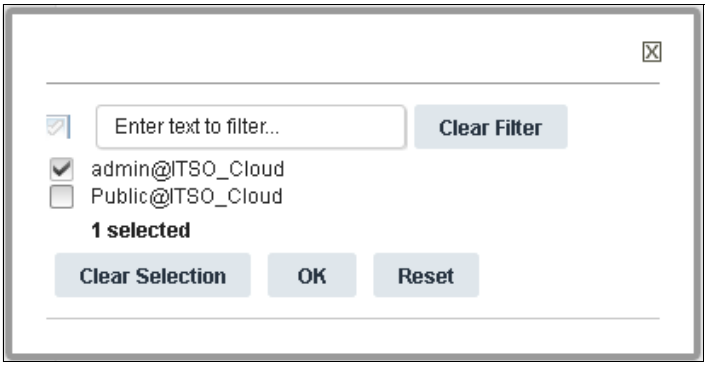


Figure 7-24 Selecting the project for the admin user

> **Note:** This is the point where the images, network configuration, volumes and other definitions from your OpenStack cloud became visible to the UrbanCode Deploy with Patterns users, so they can deploy environments in your cloud.

10. After saving, log out of UrbanCode Deploy with Patterns, and then log back in as the `admin` user.

11. You should see the cloud name on your upper-right corner of the screen, which means that you can see the OpenStack cloud components, as shown in Figure 7-25.



Figure 7-25 UrbanCode Deploy with Patterns connected to the OpenStack cloud

Now, you can deploy virtual machines in your OpenStack cloud by using UrbanCode Deploy with Patterns.

Note: At the cloud setup for this publication, two additional modifications were required to allow UrbanCode Deploy, UrbanCode Deploy with Patterns, and its Heat engine to work corectly. Please see section 7.2.8, “Required corrections in the cloud setup for this publication” on page 653 for additional information.

7.2.3 Deploying environments in UrbanCode Deploy with Patterns

After connecting the UrbanCode Deploy with Patterns server to your OpenStack cloud, you can deploy environments using the components created in your Cloud, during implementation and configuration in Chapter 4, “Cloud provider installation guidelines and implementation scenarios” on page 195. The following steps will show you how to create a simple virtual machine, connect it to a network and to a volume. We also show how to use HOT templates in the UrbanCode Deploy with Patterns.

1. Log in to UrbanCode Deploy with Patterns as the `admin` user or the user who was previously connected to your OpenStack cloud (following the instructions described in section 7.2.2, “Connect IBM UrbanCode Deploy with Patterns to the OpenStack cloud” on page 591).
2. Click **Blueprints**, as shown in Figure 7-26.

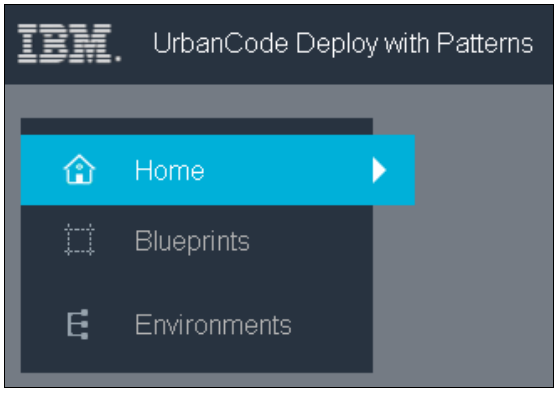


Figure 7-26 Clicking Blueprints

3. Click **New**, as shown in Figure 7-27.

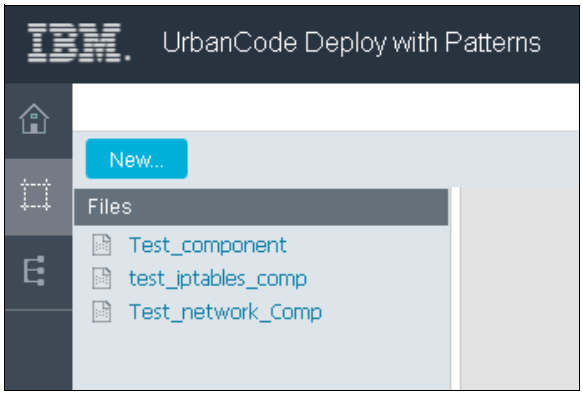


Figure 7-27 Creating a new blueprint

4. Provide a name for your blueprint and click **Save**, as shown in Figure 7-28.

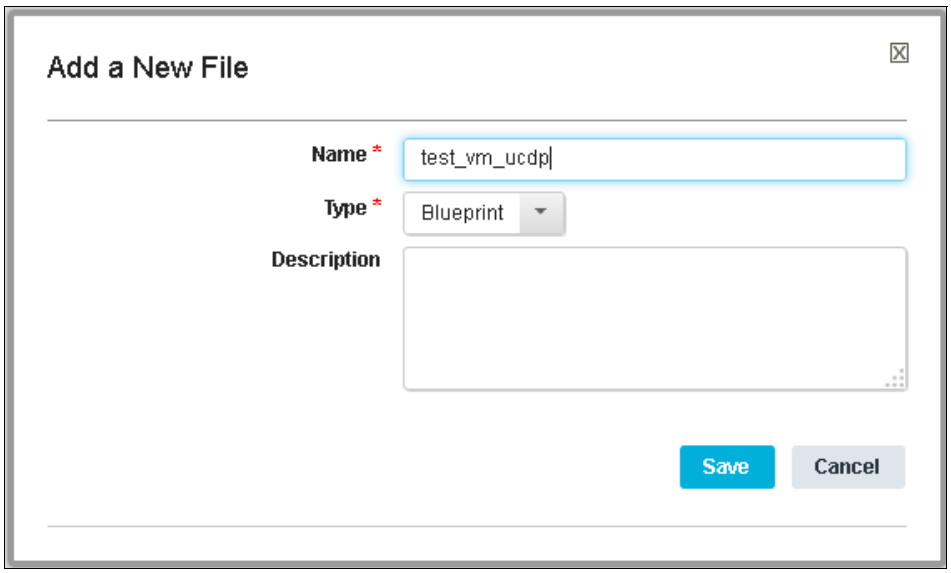


Figure 7-28 Giving your blueprint a name

5. The components from your OpenStack cloud are displayed in the left corner of your screen.

Drag the image to the center of the Diagram pane, as shown in Figure 7-29.

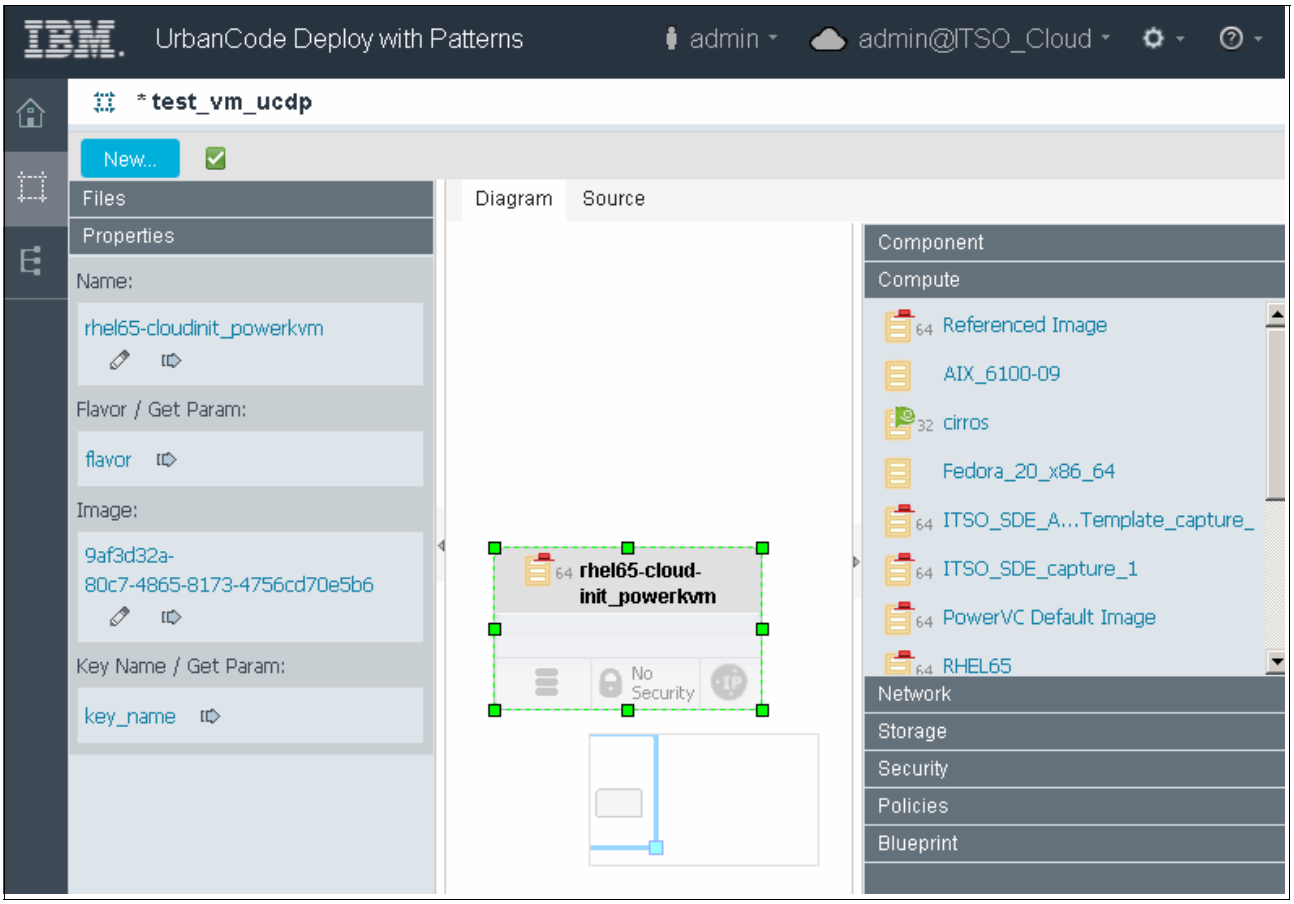


Figure 7-29 Adding a virtual machine to the diagram

6. Click **Network** and drag your network to the diagram. Move your cursor over the network until you see a blue arrow.

Drag this arrow to connect it to the virtual machine, as shown in Figure 7-30.

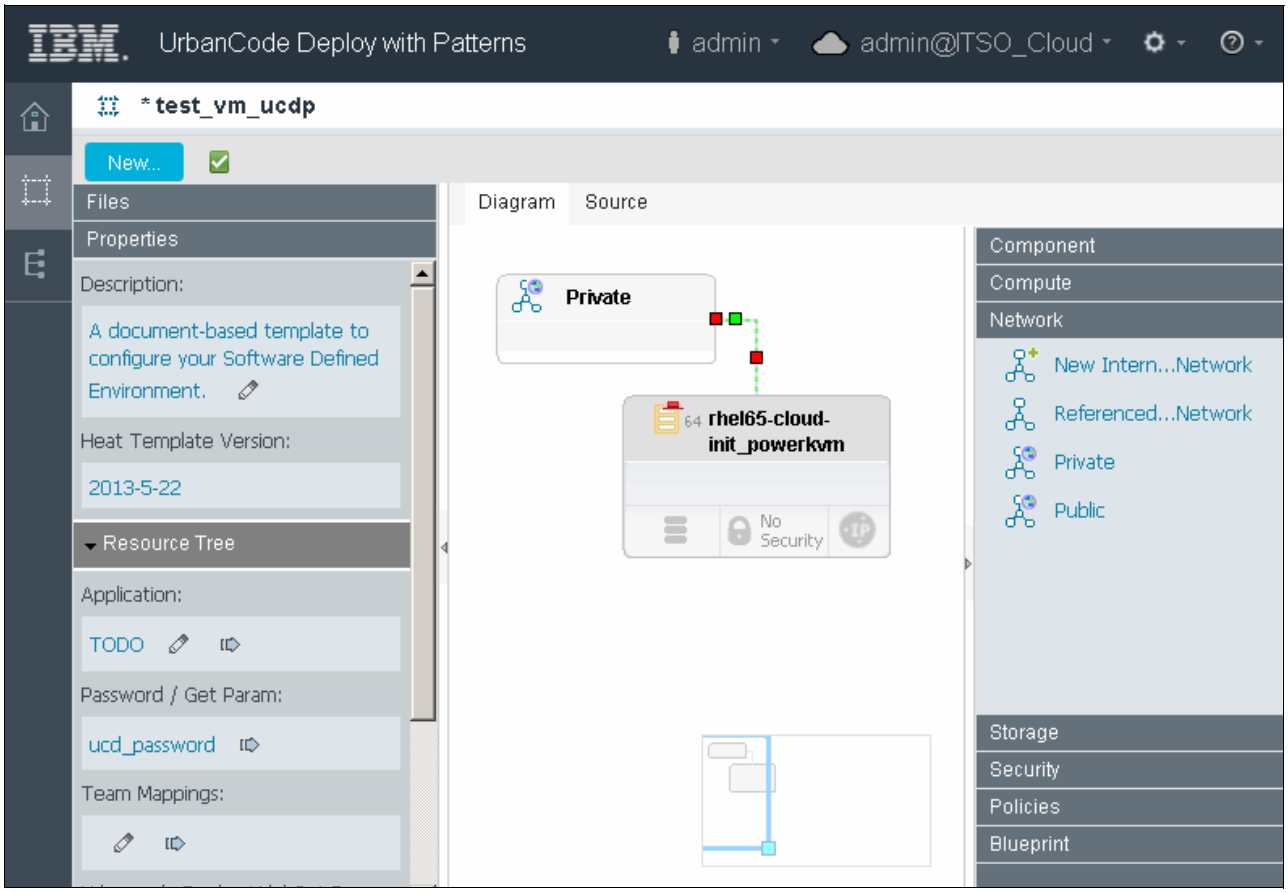


Figure 7-30 Adding a network and connecting to the virtual machine

7. You can also connect a volume to the virtual machine by clicking in Storage and dragging a volume over to the virtual machine.

You will see a blue icon with the Volume Size information in the lower-left corner of the virtual machine, as shown in Figure 7-31.

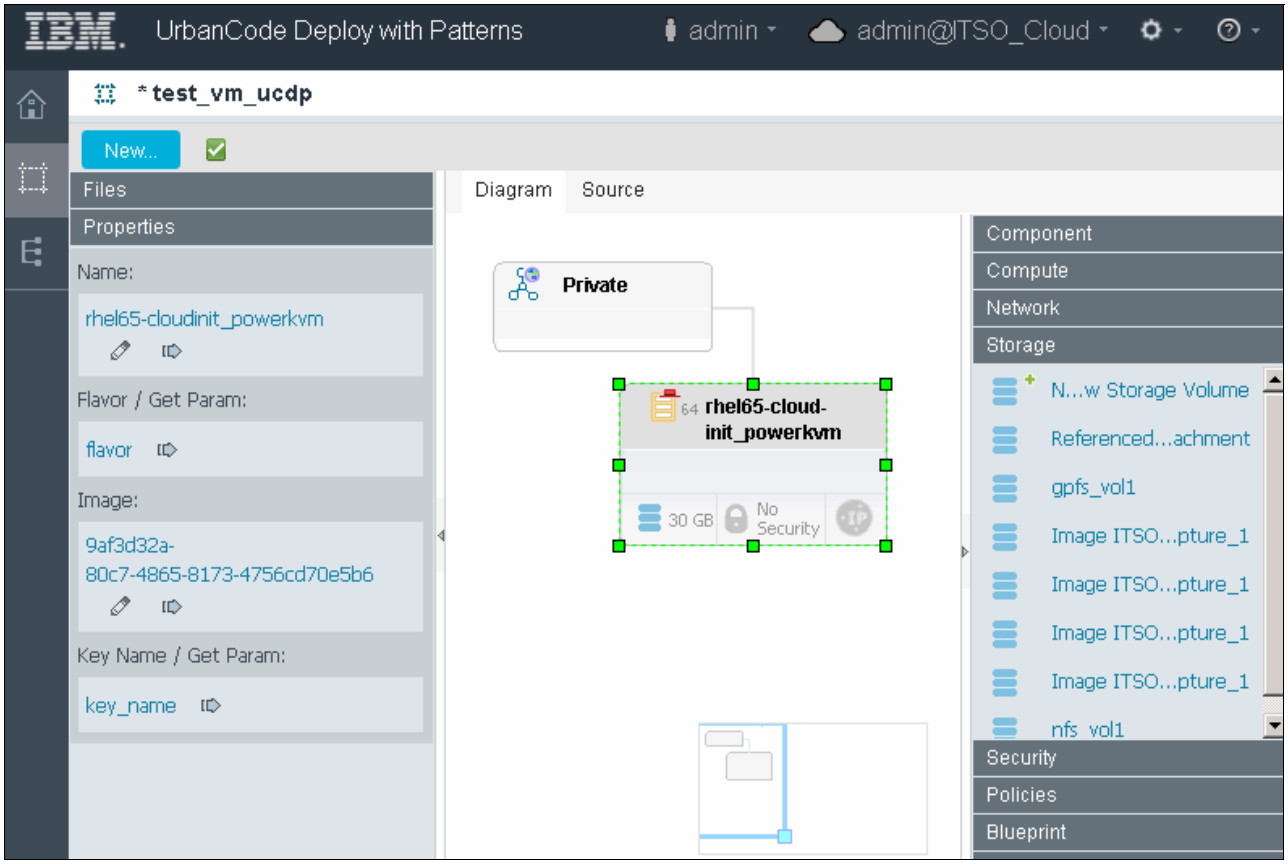


Figure 7-31 Connecting a Cinder volume to the virtual machine

8. At the top of your blueprint, you can see the Diagram tab (where you are currently drawing your environment using the OpenStack cloud components), and also a Source tab. Click the **Source** tab to see the HOT template. What UrbanCode Deploy with Patterns is doing, although you drag and connect the components, is creating the HOT template for you. It uses its own Heat engine, as configured on step 3 on page 591, to create a stack in your cloud, deploying all of these components from the HOT template.

Figure 7-32 shows the HOT template displayed when you click in the Source tab.

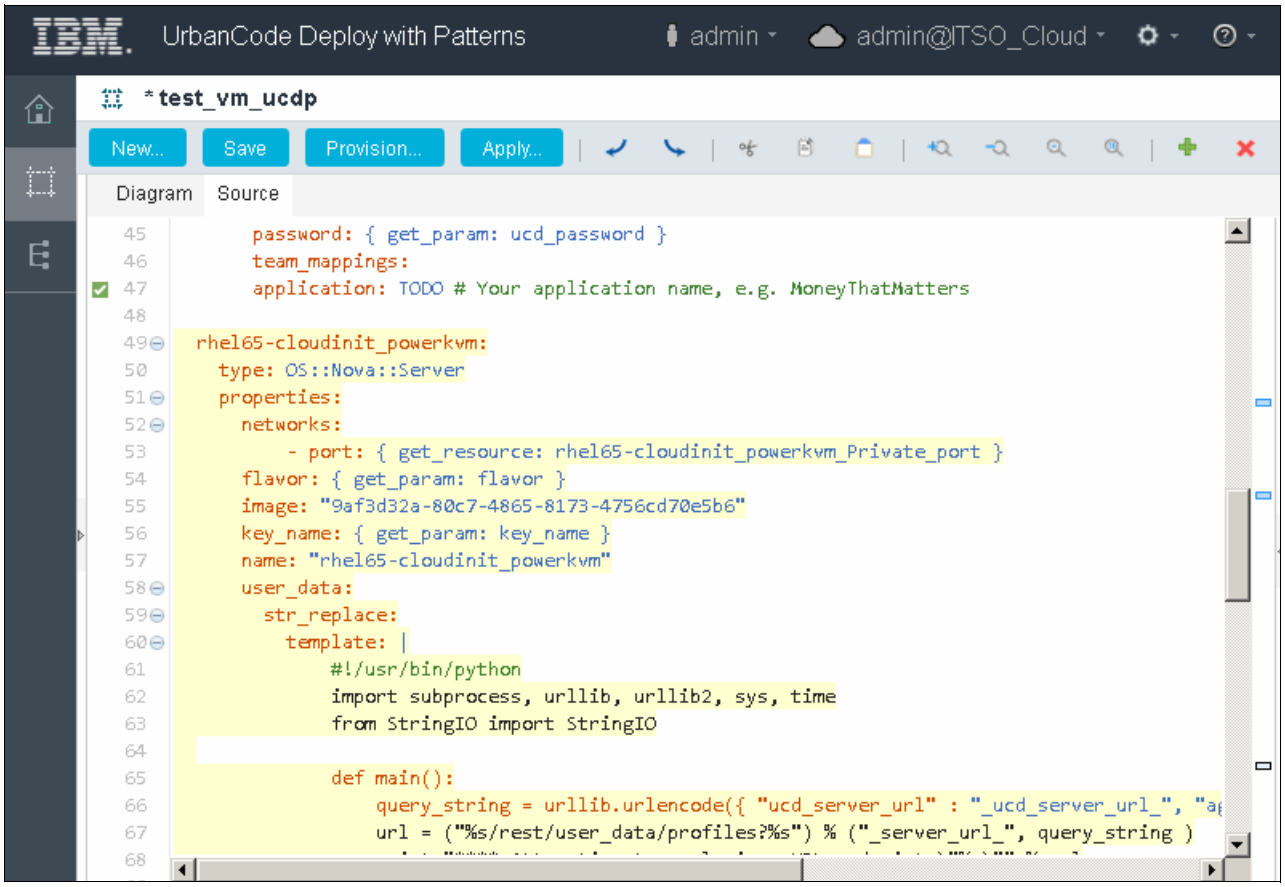


Figure 7-32 Checking the blueprint source code in UrbanCode Deploy with Patterns

9. You can modify the HOT template using instructions from section 7.1, “Heat engine and HOT templates” on page 564, directly in the Diagram tab of the blueprint that you are creating. Example 7-28 completely shows the HOT template created by in this section. In this example, the HOT template was manually modified to add a Configuration Drive to the instance by adding the following line:

   ```
   config_drive: "true"
   ```

Example 7-28 HOT template created by UrbanCode Deploy with Patterns

```
heat_template_version: 2013-05-23

description: >
  A document-based template to configure your Software Defined Environment.

parameters:
  server_url:
    type: string
    description: The base URL for the server which hosts files required by user data. Do *not*
add a trailing slash.
    default: http://172.16.21.154:8080/landscaper
  ucd_server_url:
    type: string
```

```
    description: The server URL for agent communication to UrbanCode Deploy. Do *not* add a
trailing slash.
    default: https://172.16.21.154:7443
  ucd_user:
    type: string
    description: The user credential for the UrbanCode Deploy server.
    default: admin
  ucd_password:
    type: string
    description: The user credential for the UrbanCode Deploy server.
    hidden: true
  flavor:
    type: string
    description: Flavor to be used for compute instance
    default: "m1.small"
  key_name:
    type: string
    description: Name of key-pair to be used for compute instance
    default: "TODO"
  network_id_Private:
    type: string
    description: Generated to reference 'Private' network
    default: "d828cdb1-e926-4f94-b276-29a41471e5b6"

resources:
## REFERENCE {"Private":{"type": "OS::Neutron::Net", "properties": {"name": "Private",
"network_id": "d828cdb1-e926-4f94-b276-29a41471e5b6"}}}

  # declaration of template resources
  resource_tree:
    type: IBM::UrbanCode::ResourceTree
    properties:
       urbancode_deploy_url: { get_param: ucd_server_url }
       username: { get_param: ucd_user }
       password: { get_param: ucd_password }
       team_mappings:
       application: TODO # Your application name, e.g. MoneyThatMatters

  rhel65-cloudinit_powerkvm:
    type: OS::Nova::Server
    properties:
      networks:
          - port: { get_resource: rhel65-cloudinit_powerkvm_Private_port }
      flavor: { get_param: flavor }
      image: "9af3d32a-80c7-4865-8173-4756cd70e5b6"
      key_name: { get_param: key_name }
      name: "rhel65-cloudinit_powerkvm"
      config_drive: "true"
      user_data:
        str_replace:
          template: |
              #!/usr/bin/python
              import subprocess, urllib, urllib2, sys, time
              from StringIO import StringIO
```

```
              def main():
                  query_string = urllib.urlencode({ "ucd_server_url" : "_ucd_server_url_",
"agent_name" : _agent_name_ })
                  url = ("%s/rest/user_data/profiles?%s") % ("_server_url_", query_string )
                  print "**** Attempting to curl given URL endpoint \"%s\"" % url
                  for x in range(0, 5):
                      try:
                          version = urllib2.urlopen("_server_url_/server/version",
timeout=5).read()
                      except urllib2.URLError, e:
                          print "***** Network access not yet available. Will try again ..."
                          time.sleep(5)
                  script = urllib2.urlopen(url, timeout=300).read()
                  print "****** SCRIPT RESULT: %s" % str(subprocess.call(script, shell=True))
              if __name__ == "__main__":
                  sys.exit(main())

          params:
            _agent_name_:
               str_replace:
                  template: |
                    "_application_name_._environment_name_._server_name_"
                  params:
                    _application_name_: { get_attr: [resource_tree, application_name ] }
                    _environment_name_: { get_param: "OS::stack_name" }
                    _server_name_: "rhel65-cloudinit_powerkvm"
            _ucd_server_url_: { get_param: ucd_server_url }
            _server_url_: { get_param: server_url }

  rhel65-cloudinit_powerkvm_gpfs_vol1_attachment:
    type: OS::Cinder::VolumeAttachment
    properties:
      volume_id: 37b5bbce-aebd-4bf8-bdc5-e29c63ccdd2e
      instance_uuid: { get_resource: rhel65-cloudinit_powerkvm }
      mountpoint: /dev/vdc

  rhel65-cloudinit_powerkvm_Private_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_param: network_id_Private }

outputs:
  blueprint_url:
    description: Blueprint Origin URL
    value:  http://172.16.21.154:8080/landscaper/view/projects?open=test_vm_ucdp
```

10. You can notice in the HOT template that some of the components have the `TODO` string. This means that UrbanCode Deploy with Patterns could not automatically select a value for this component. For some of the components you can select the available values when you click the Provision button, on step 13 on page 607, but for some components, there is no combo box to select the available values, so you need to solve these `TODO` strings directly in the HOT template.

Example 7-29 shows the `key_name` parameter from the HOT template with the `TODO` option (you can select the available values for this component when you click Provision, so you do not need to solve this value directly in the HOT template).

Example 7-29 Sample value with the TODO String

```
key_name:
    type: string
    description: Name of key-pair to be used for compute instance
    default: "TODO"
```

11. Given that at this time you have not created any component in UrbanCode Deploy, you need to remove the `resource_tree` component from the HOT template, otherwise the UrbanCode Deploy with Patterns will try to use an application called `TODO`, which does not exist in UrbanCode Deploy, and the provisioning might fail. Click the `Source` tab and remove the resource_tree shown in Example 7-30.

Example 7-30 Removing the resource_tree from HOT template

```
resource_tree:
    type: IBM::UrbanCode::ResourceTree
    properties:
       urbancode_deploy_url: { get_param: ucd_server_url }
       username: { get_param: ucd_user }
       password: { get_param: ucd_password }
       team_mappings:
       application: TODO # Your application name, e.g. MoneyThatMatters
```

12. You will notice that all entries in the HOT template that are broken will have a red X sign, as shown in Figure 7-33.

Diagram Source

```
65                          time.sleep(5)
66                  script = urllib2.urlopen(url, timeout=300).read()
67                  print "****** SCRIPT RESULT: %s" % str(subprocess.call(script, shell=True))
68              if __name__ == "__main__":
69                  sys.exit(main())
70
71          params:
72            _agent_name_:
73               str_replace:
74                  template: |
75                    "_application_name_._environment_name_._server_name_"
76                  params:
77                    _application_name_: { get_attr: [resource_tree, application_name ] }
78                    _environment_name_: { get_param: "OS::stack_name" }
79                    _server_name_: "rhel65-cloudinit_powerkvm"
80            _ucd_server_url_: { get_param: ucd_server_url }
81            _server_url_: { get_param: server_url }
```

Figure 7-33 Red X in a broken component from the HOT template

In this case, the `_application_name_` component is broken, because it is trying to reference the `application_name`, which was removed in step 11, so this line must also be removed from the HOT template.

13. After completing the modifications to your blueprint, click **Save**, and then click **Provision**, as shown in Figure 7-34.

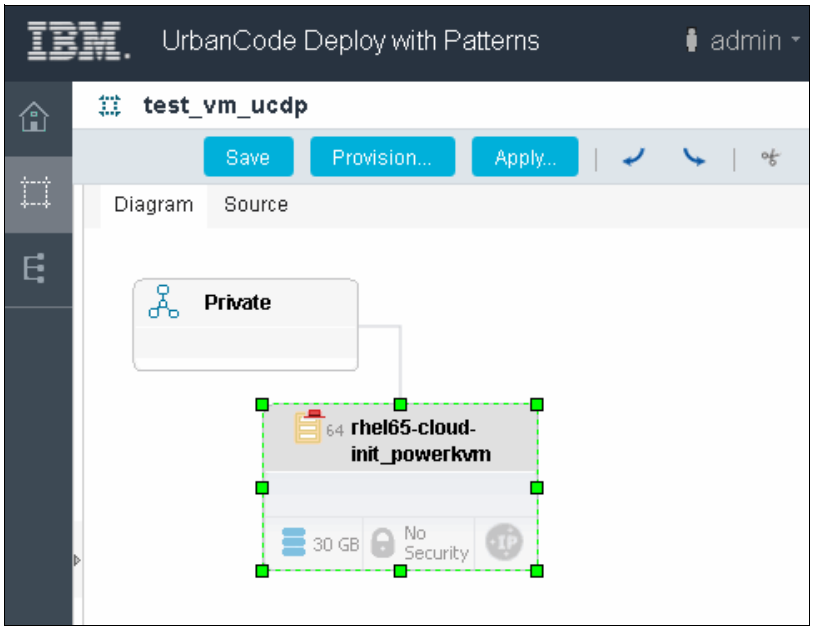
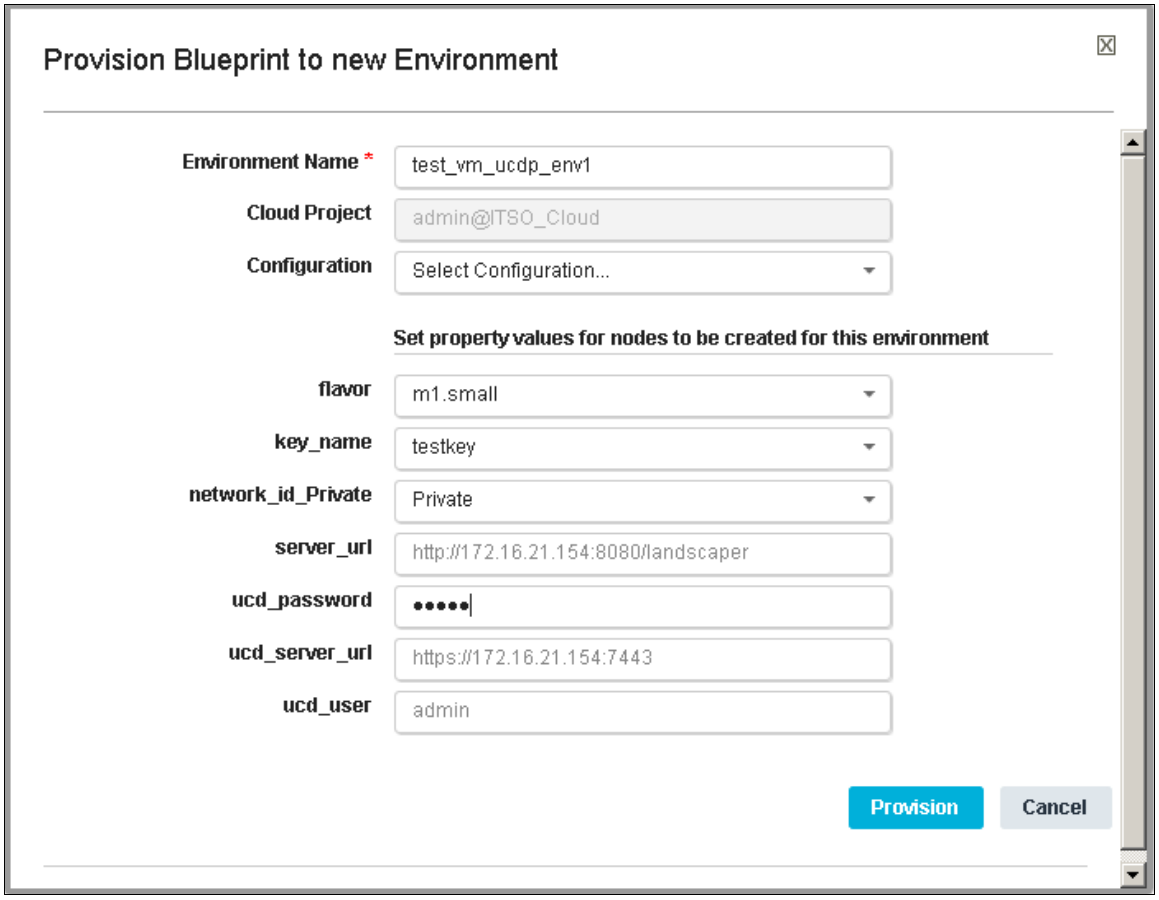


Figure 7-34 Clicking on Save and Provision to start deploying your blueprint

14. In the Provision Blueprint to New Environment screen, provide a name for your new environment, and then select the property values for the virtual machines, including the flavor, SSH-Key pair to inject, and the network to be used. In the `ucd_password` field, type the password of the `admin` user from the UrbanCode Deploy server, and click **Provision**, as shown in Figure 7-35 on page 608.

Important: The fields that UrbanCode Deploy with Patterns could not suggest a value according to the selections made in the diagram will be filled with the `TODO` string. All of the `TODO` entries should be modified before provisioning your environment.

The password in the `ucd_password` field is the `admin` password from the UrbanCode Deploy server. This is not your admin password from OpenStack cloud. The initial password for the admin user in UrbanCode Deploy is `admin`. This information is required to allow UrbanCode Deploy with Patterns to communicate with UrbanCode Deploy to use the components that you can create there. At this section no components were created yet, but this information is still required.



Important:

Figure 7-35 Providing the properties of the virtual machine or instance

15. Click the **Environments** tab on the left of the screen, and you should see the environment that has just been created, as shown in Figure 7-36.

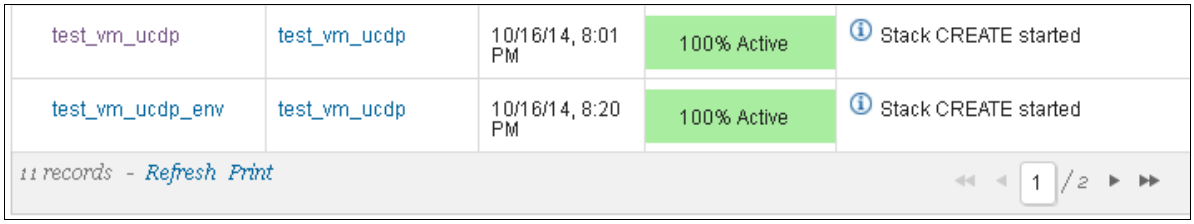


Figure 7-36 Environments created in UrbanCode Deploy with Patterns

16. Click the environment name and you should see each component deployed in your OpenStack cloud, including the instances and its assigned IP address, as shown in Figure 7-37.

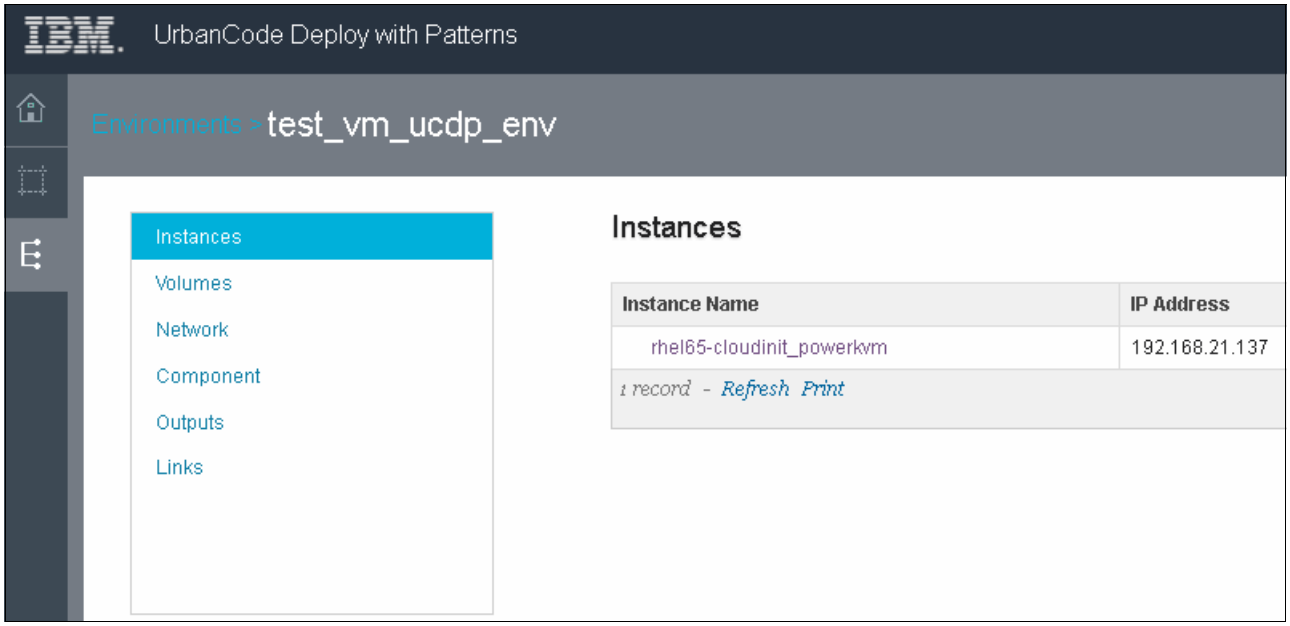


Figure 7-37 Checking the components of the environment

7.2.4 Creating components and applications in UrbanCode Deploy

Using IBM UrbanCode Deploy, you can enhance your cloud by creating components and applications that can be used by UrbanCode Deploy with Patterns, automating the deployment of applications. The user of the UrbanCode Deploy with Patterns can drag the component onto a virtual machine and the application will be automatically downloaded and installed, according to the instructions provided in the component created in UrbanCode Deploy.

When a component is created in UrbanCode Deploy and used in UrbanCode Deploy with Patterns, the HOT template is populated with instructions to allow the communication between these two applications, allowing the deployed instance to obtain the component from UrbanCode Deploy and install it.

The next steps allow you to create a component in UrbanCode Deploy that will download and install a MySQL Server in the instance. The same steps can be used to create additional components for any application that you plan to deploy in your OpenStack cloud:

1. Log in to the UrbanCode Deploy web interface as the `admin` user.
2. Click **Components** and then **Create New Component**, as shown in Figure 7-38 on page 610.

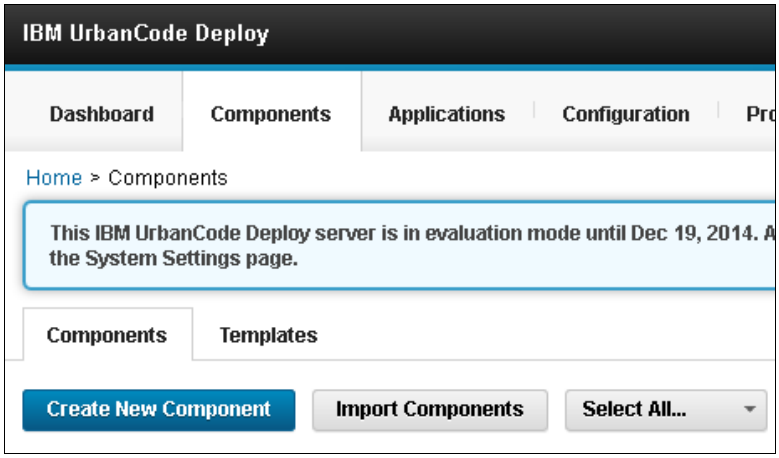


Figure 7-38 Creating a new component in UrbanCode Deploy

3. Provide a `Name` to your component, so you can identify it. Select **File System (Versioned)** in the Source Config Type and provide the Base Path, which is where your application files will be placed in the UrbanCode Deploy server (the instance will look for these files when it is deployed) and click `Save`, as shown in Figure 7-39 on page 611.

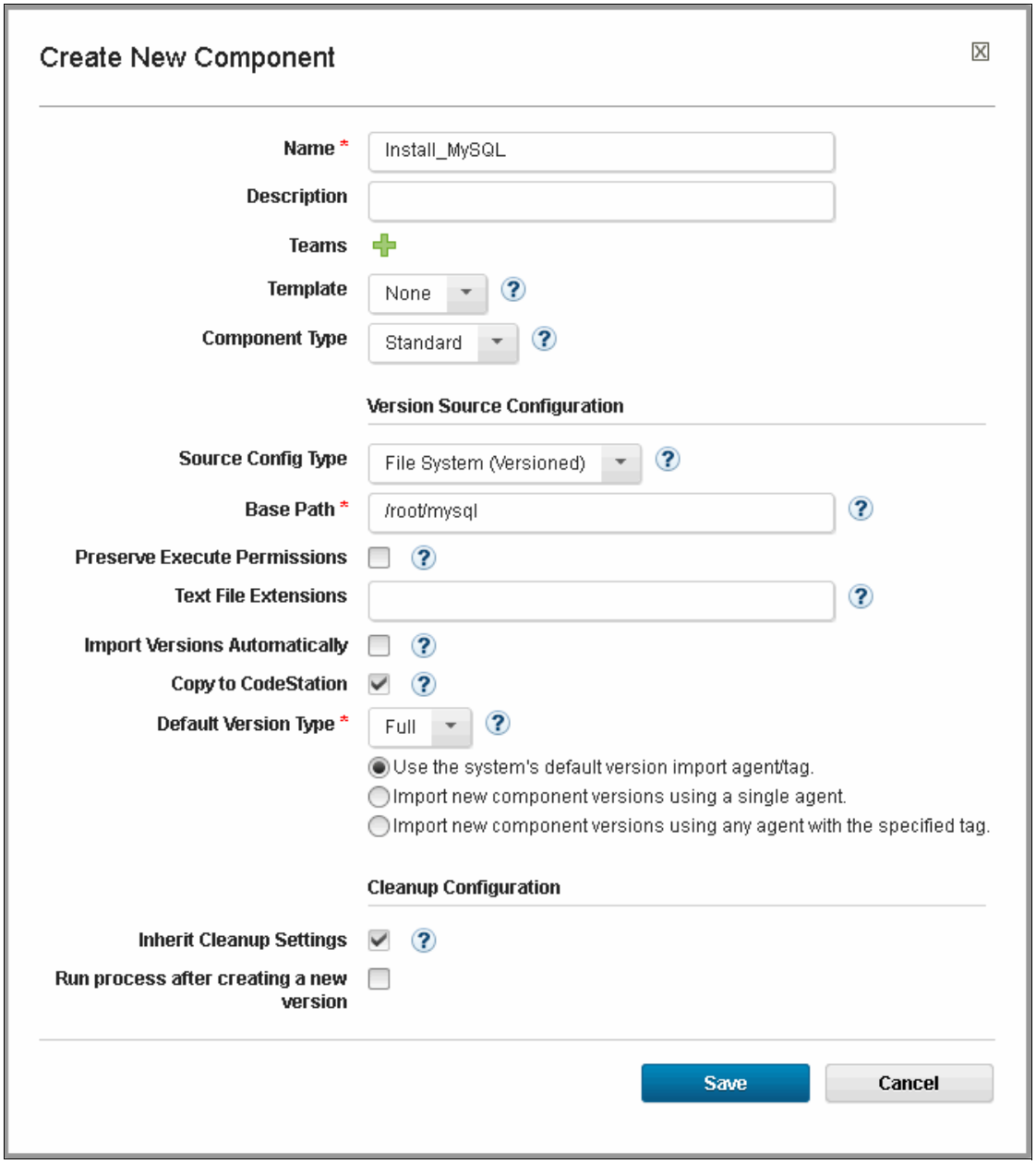


Figure 7-39 Creating a new component

4. Log in to the UrbanCode Deploy server shell using the root user. You need to create the structure of the component in the `Base Path` you have specified in step 3 on page 610. Inside the Base Path, you must create directories to place the versions of the application you will provide in this component. UrbanCode Deploy looks for directories named `1.0`, `2.0`, `3.0`, and so on, which represent the version of the application that you are creating. Example 7-31 creates a directory `/root/mysql`, which is the Base Path specified in step 3 on page 610 and then creates a directory called `1.0` inside this Base Path.

Example 7-31 Creating the directory structure for the new component

```
[root@ucdp ~]# mkdir /root/mysql
[root@ucdp ~]# mkdir /root/mysql/1.0
```

5. Place all of the files relevant to this component, which you want to provide for the instance, inside this directory. These files will be copied to the instance (or virtual machine) and used in the installation process of the application. These files are called *artifacts*. Example 7-32 shows the MySQL installer being placed inside the `/root/mysql/1.0` directory.

Example 7-32 Placing the component files in the base path versioned directory

```
[root@ucdp ~]# cp /root/mysql-5.1.71-1.el6.x86_64.rpm /root/mysql/1.0/
[root@ucdp mysql]# cd /root/mysql/1.0/
[root@ucdp 1.0]# ls -l
total 896
-r--r--r-- 1 root root 914896 Oct 21 09:31 mysql-5.1.71-1.el6.x86_64.rpm
```

6. Go back to the UrbanCode Deploy web interface and, inside your component, click **Versions** and then click **Import New Versions,** as shown in Figure 7-40.

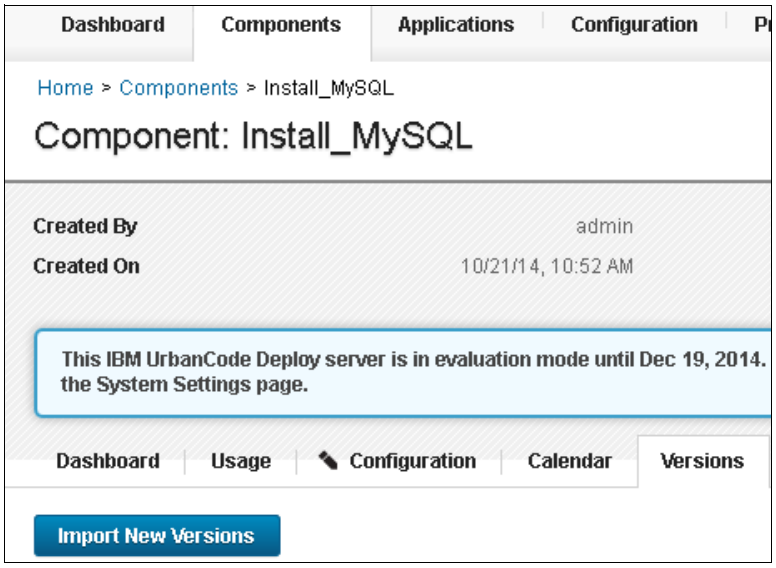


Figure 7-40 Importing new versions of a component

7. During the import process, you will be able to see the version being imported in the Currently Running Version Imports section, as shown in Figure 7-41.

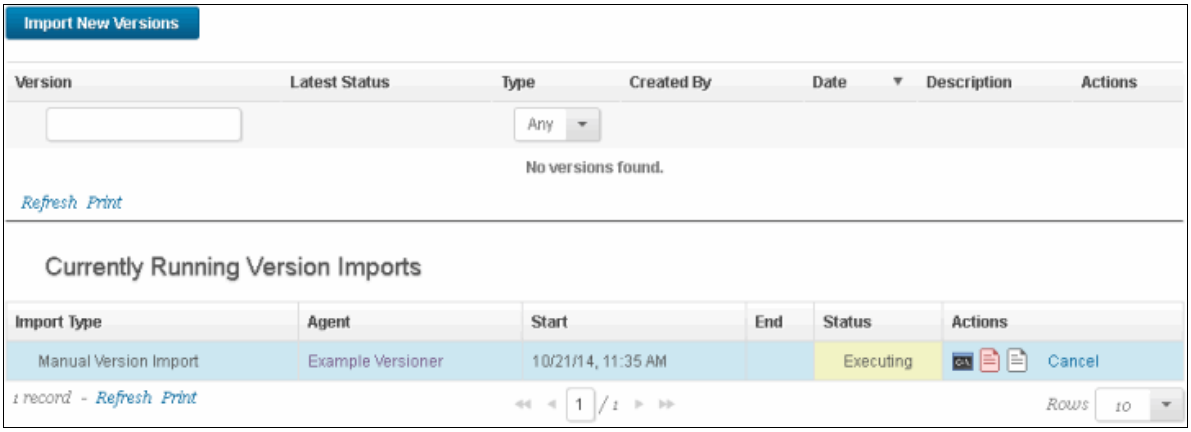


Figure 7-41 Importing the new version of the component

8. Click **Refresh**, and you should be able to see the version 1.0 of your component, as shown in Figure 7-42.

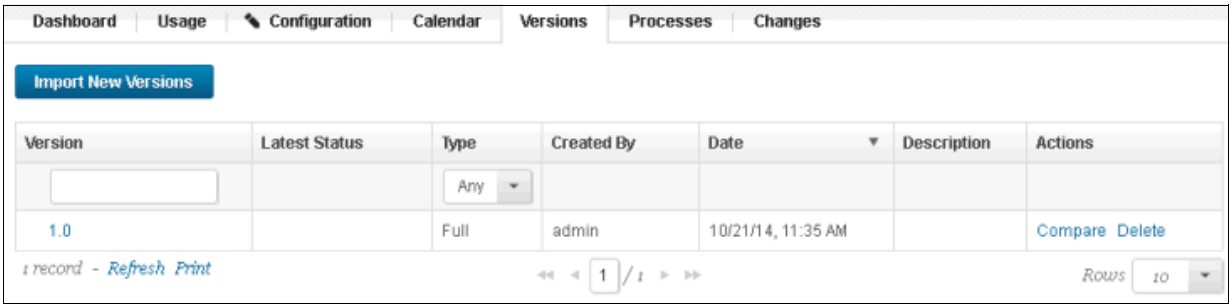


Figure 7-42 Version 1.0 of the component imported by UrbanCode Deploy

9. Click the **1.0** version of your component, and you should be able to see the `mysql-5.1.71-1.el6.x86_64.rpm` file in the Artifacts section, as shown in Figure 7-43.

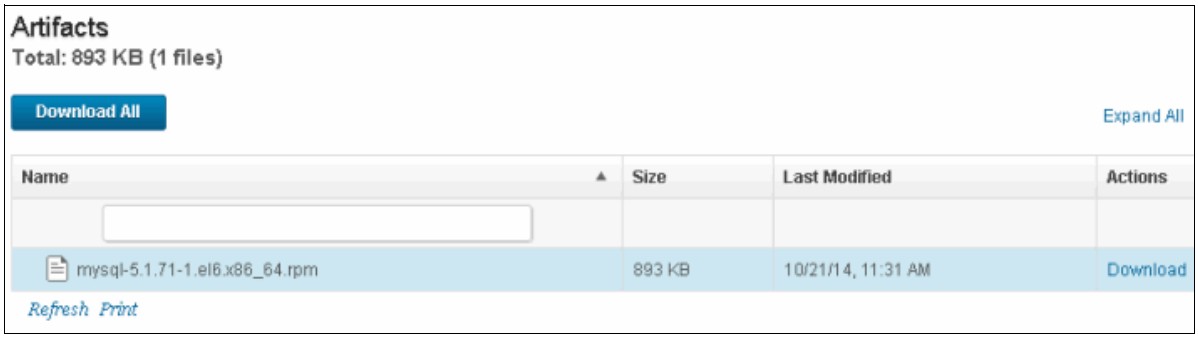


Figure 7-43 Artifacts from a component in UrbanCode Deploy

10. Now you need to create a process for your component. In this process, you will create a directory and download the artifacts (in this case, the MySQL installer) to the instance.

    Click the component name, as shown in Figure 7-44.



Figure 7-44 Click the component name

11. Now, click **Processes** and then **Create New Process**, as shown in Figure 7-45.

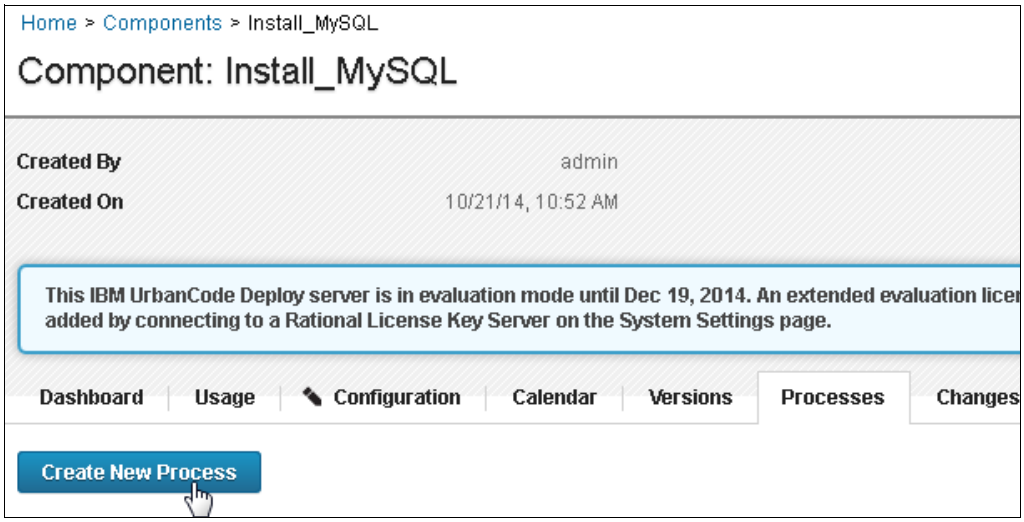


Figure 7-45 Creating a new Process

12. Provide a name for your process and click **Save**, as shown in Figure 7-46.

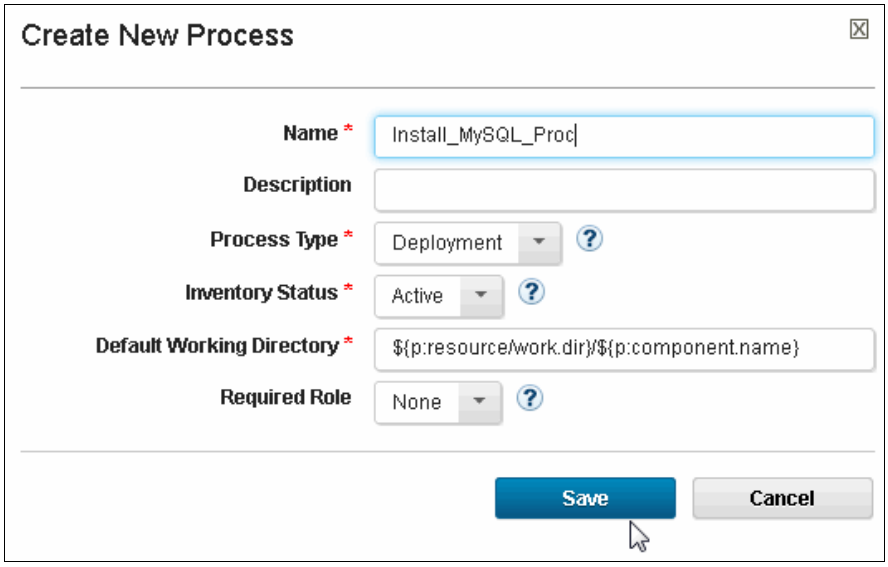


Figure 7-46 Providing a name for the new Process

13. In the line that contains the process name, click **Edit**, as shown in Figure 7-47 on page 615.

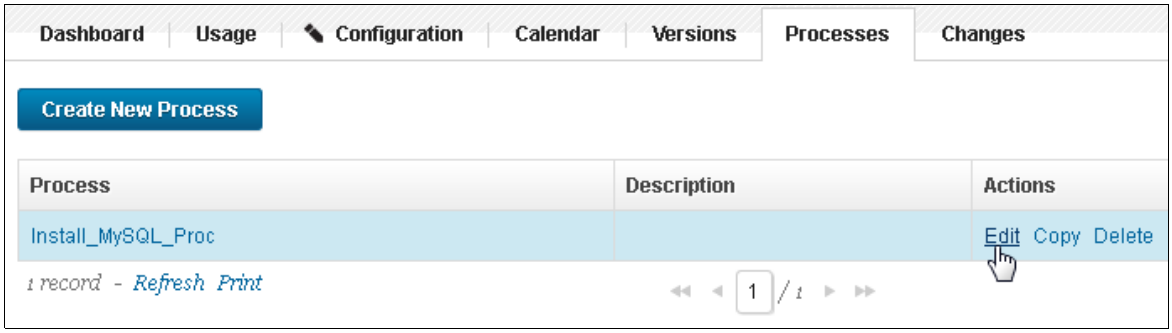


Figure 7-47 Editing the process

14. The Designer is now open. In this process, you will create a directory, download the artifacts, and execute a script to install the MySQL. To tell UrbanCode Deploy to create a directory when this process is executed, in the Step Palette, click **FileUtils > Create Directories**, and drag it to the blank Designer screen, as shown in Figure 7-48.

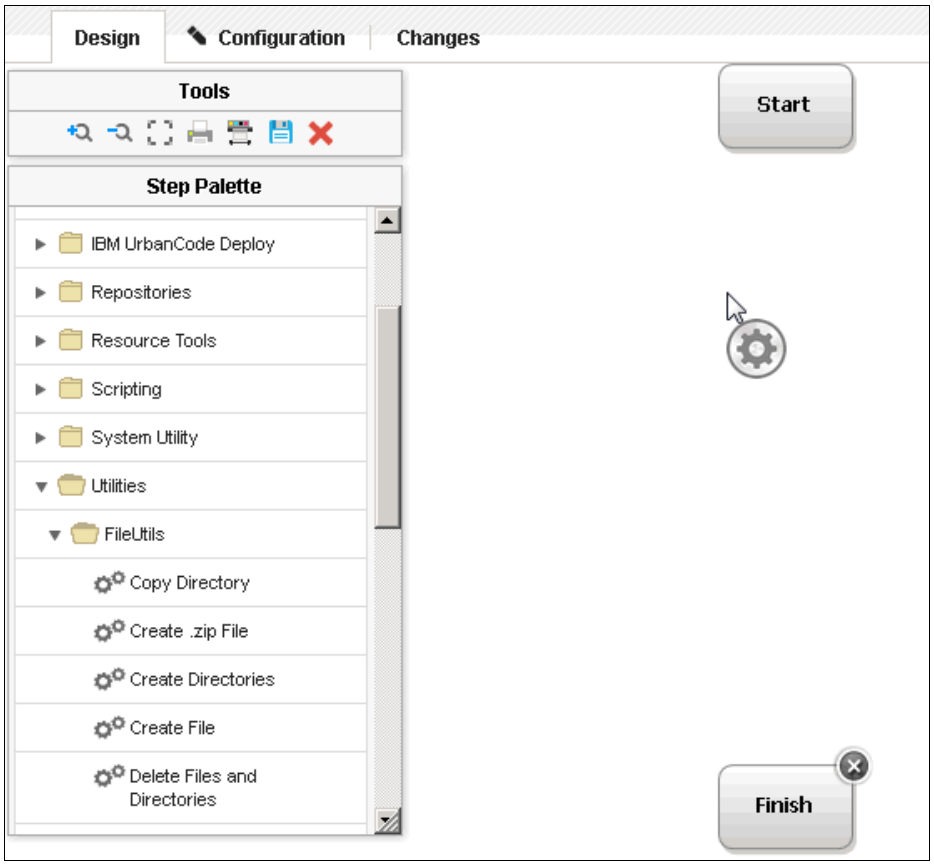


Figure 7-48 Adding the Create Directories step to the Process

15. Provide a Name to this step and provide the Working Directory information, and then click **Save**. The specified Working Directory will be created in the instance or virtual machine after this process is executed. Figure 7-49 on page 616 shows a process where the `/root/mysql-installer` is created.

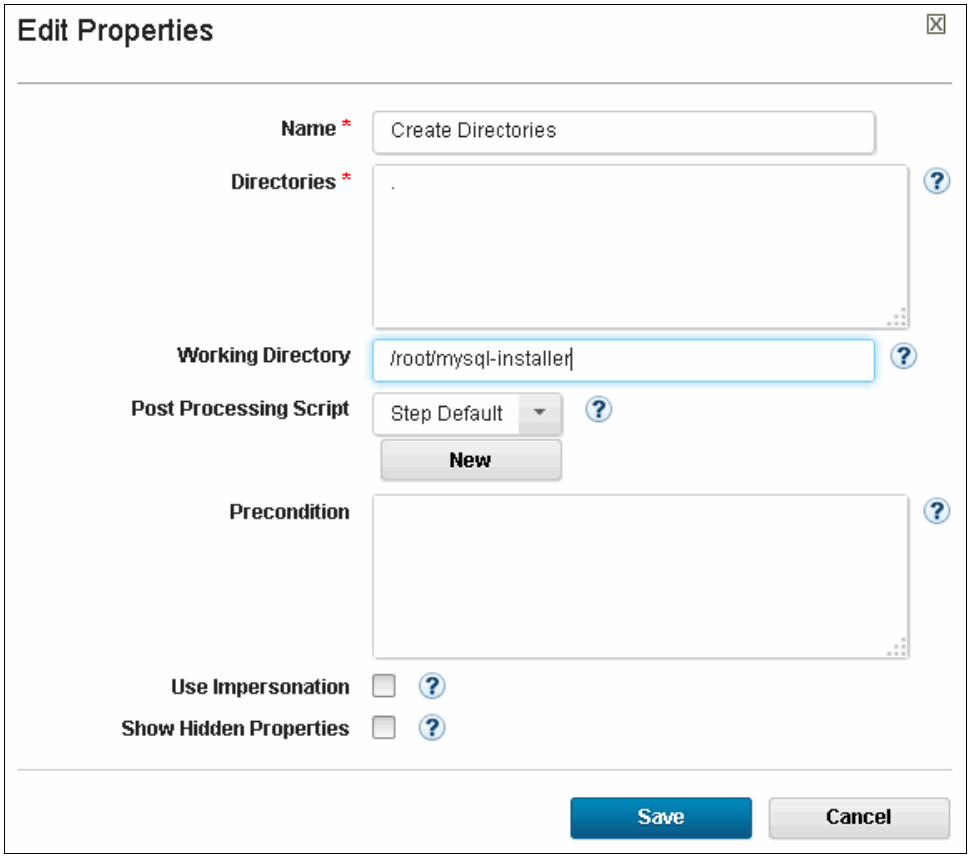


Figure 7-49 Shows the directory creation step in the process

16. Now add a step to download the artifacts from this component (in this case, download the MySQL installer `rpm` file). In the Step Palette, drag the **Repositories > Artifact > IBM UrbanCode Deploy > Download Artifacts** step to the blank screen in the Designer, as shown in Figure 7-50 on page 617.

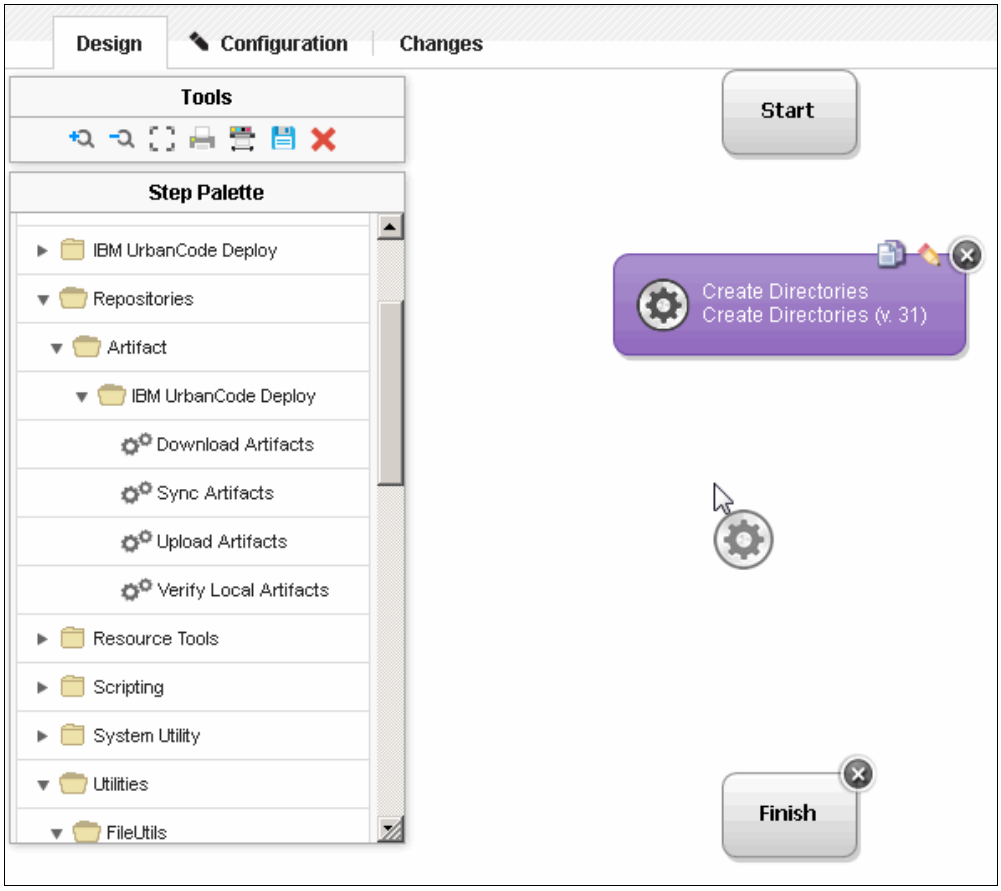


Figure 7-50 Adding the Download Artifacts step to the Process

17. Provide a name for this step. In the Working Directory field, provide the path where you want this artifact to be placed in the virtual machine when it is downloaded, and then click **Save**, as shown in Figure 7-51 on page 618.

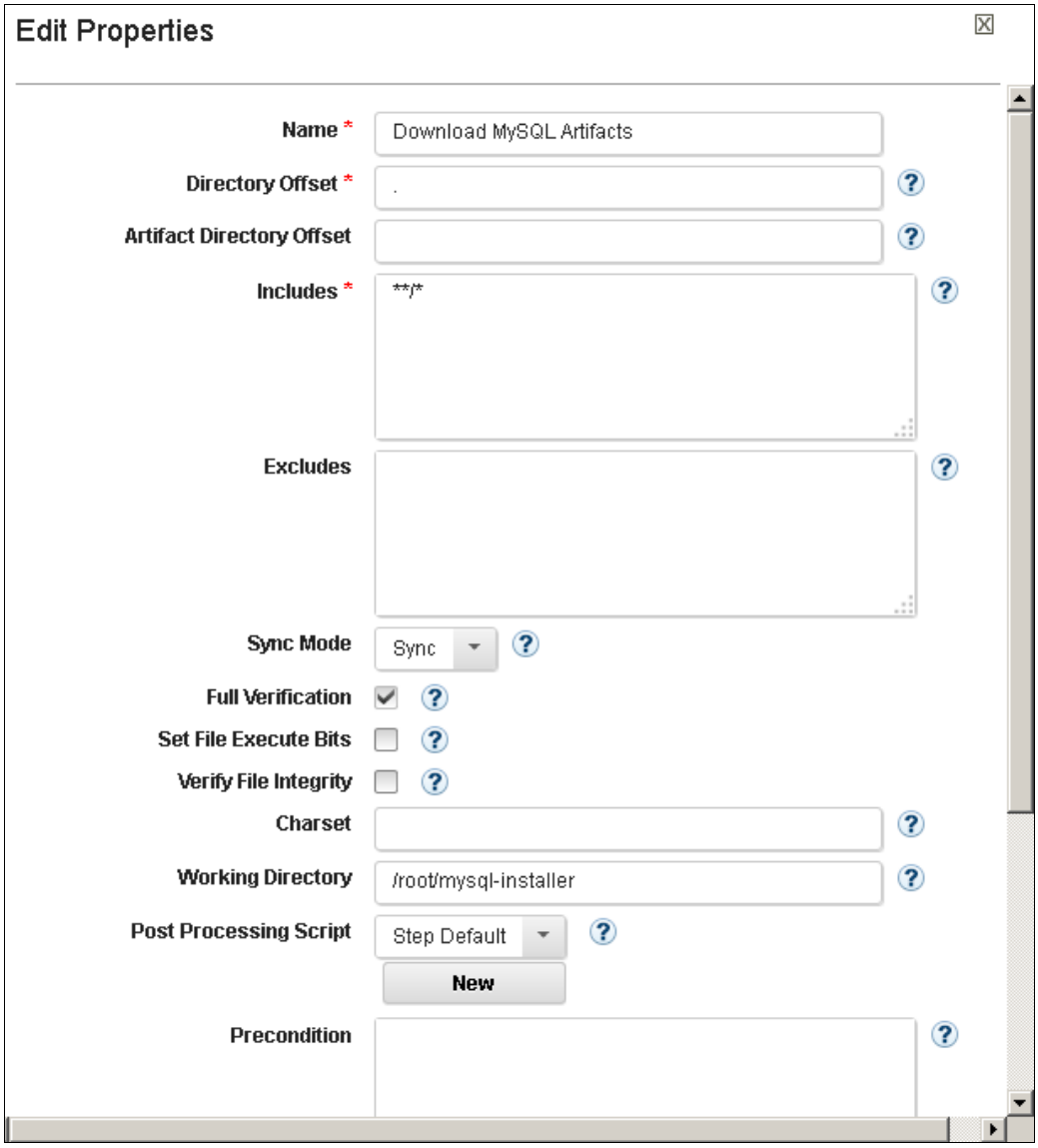


Figure 7-51 Creating the step to download the component artifacts

18. Now add a shell script step to execute the rpm command to install the artifact downloaded from the component. In the Step palette, select **Scripting > Shell > Shell**, and drag the shell script to the blank screen in the Designer, as shown in Figure 7-52 on page 619.

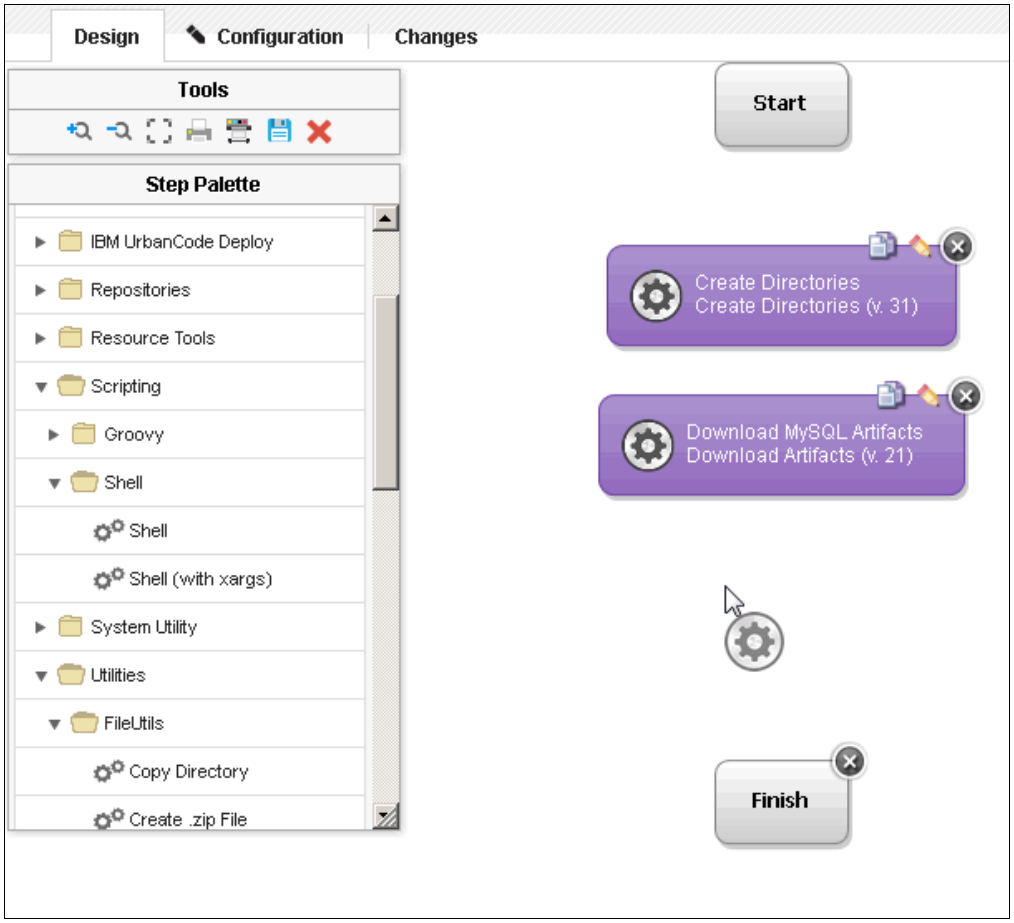


Figure 7-52 Adding a shell script to the process

19. Provide a name for your shell script in the Name field. In the Shell Script field, create the shell script, which will be executed in the virtual machine. In Figure 7-53 on page 620, a simple shell script is prepared to install the MySQL file by using the `rpm` command.

    Click **Save** when you finish creating the shell script.

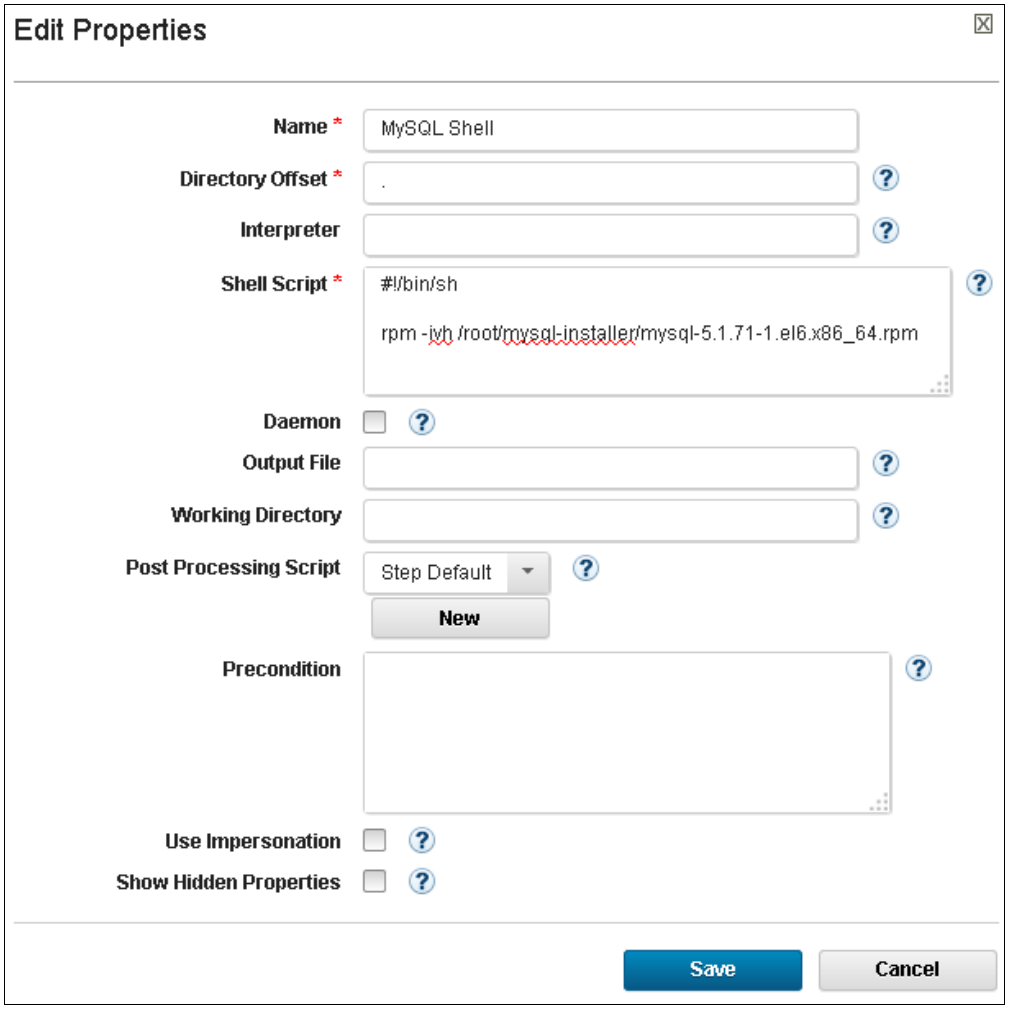


Figure 7-53 Creating the Shell Script step in your Process

20. Now you need to set up the connection between each of these steps. In the Designer, mode, hover your cursor over the **Start** box until you see a blue arrow, and then drag this arrow over the first step that you want to execute, which is the **Create Directories** box, and then move the cursor to Create Directories and drag the blue arrow over the **Download MySQL Artifacts** box.

    Then, move the cursor the **Download MySQL Artifacts** and drag the blue arrow over the MySQL shell. Do the same to connect the MySQL shell to the **Finish** box.

    When you are finished with connecting all the boxes, click the blue diskette icon, which is the **Save** icon, as shown in Figure 7-54 on page 621.

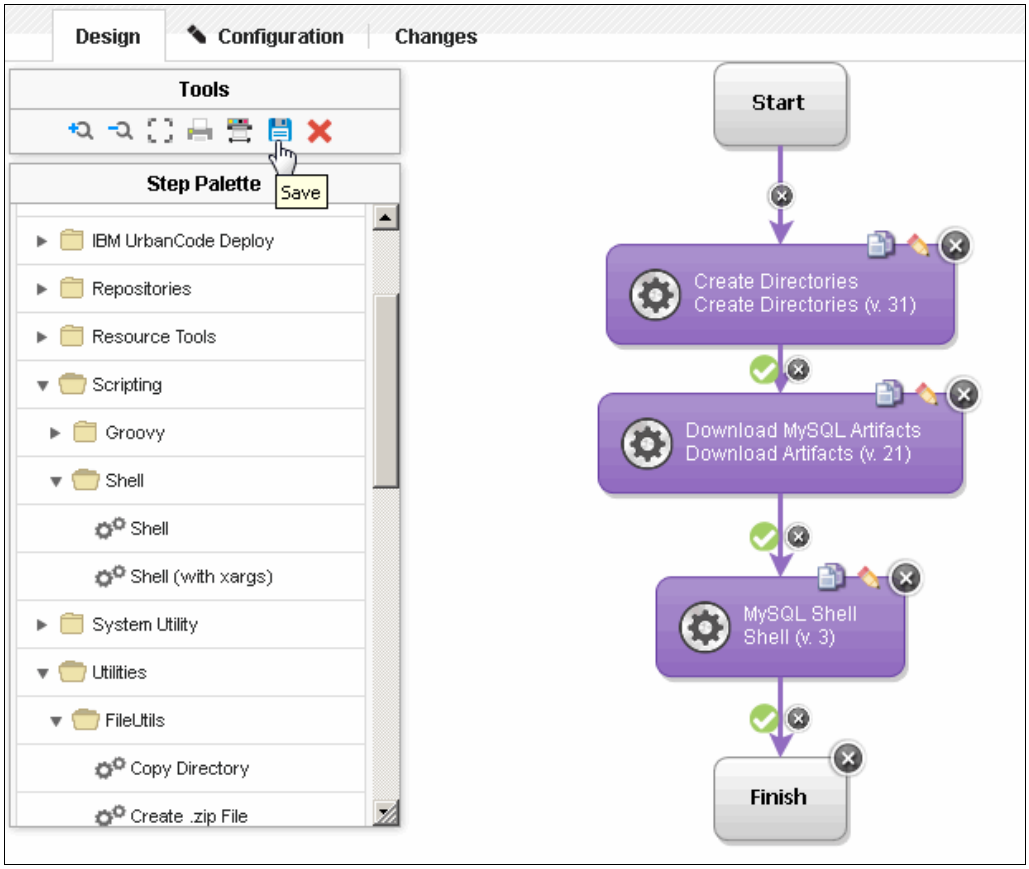


Figure 7-54 Connecting the Steps

With this configuration, when you run this component, it will create a directory, download the `rpm` file to the virtual machine, and install it using the **`rpm`** command through the shell script.

21. After saving the component, you need to create an application (which will use this component). Click **Applications > Create New Application**, as shown in Figure 7-55.

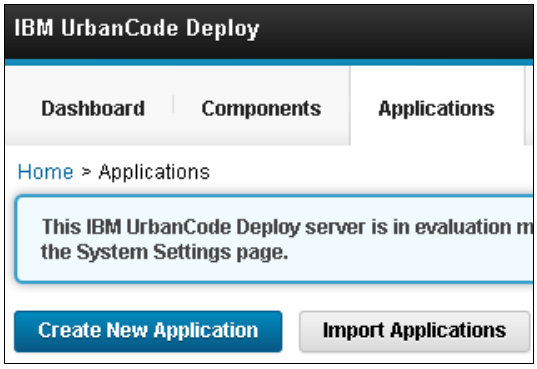


Figure 7-55 Creating a new application

22. Provide a name for your application and click **Save**, as shown in Figure 7-56.

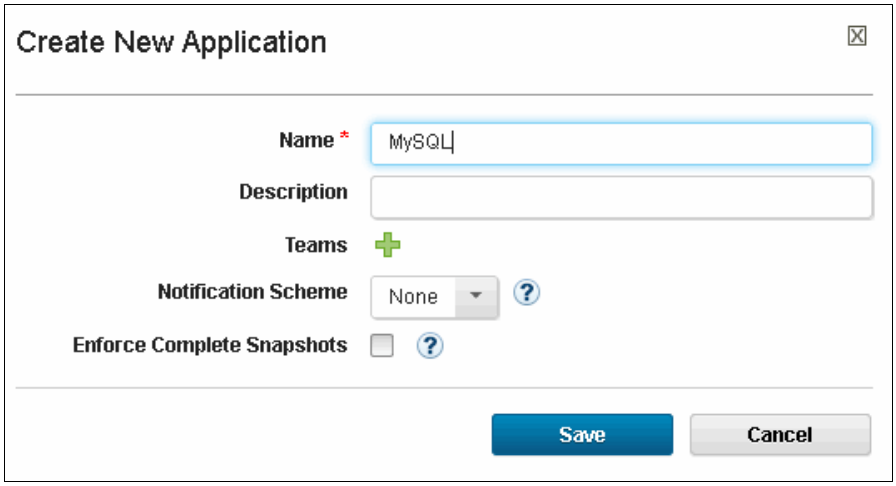


Figure 7-56 Creating an application in UrbanCode Deploy with Patterns

23. An application is made of components, so you need to add components to your application (in this case, the component you created in the previous steps from this section). Still inside the application, click **Components > Add Component**, as shown in Figure 7-57.

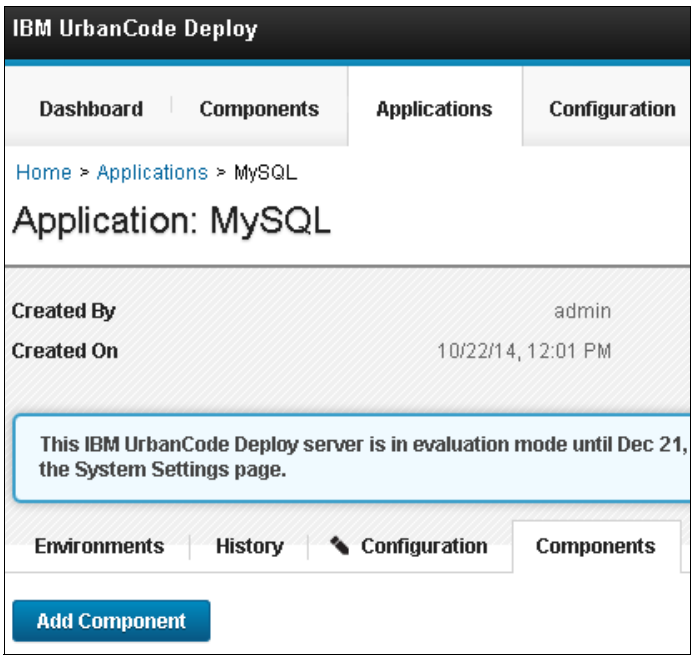


Figure 7-57 Adding components to your application

24. Select all of the desired components that you want for your application (in this example, only the **Install_MySQL** component is selected). Then, click **Save**, as shown in Figure 7-58 on page 623.

Note: You can have multiple components in the same application.

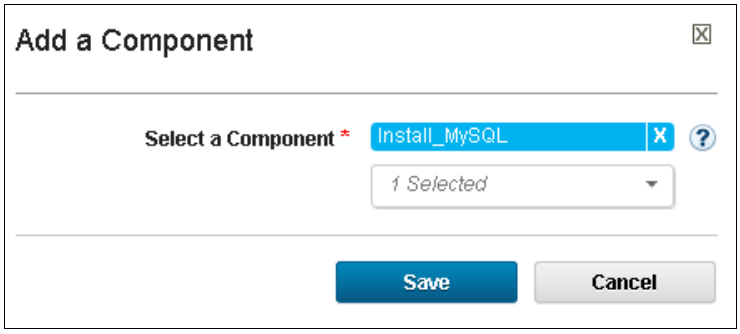


Figure 7-58 Selecting the components for the application

25. Now you need to create a Process inside the application, to tell it to install the components, when it is executed. Still inside the application, click `Processes` → `Create New Process`, as shown in Figure 7-59.

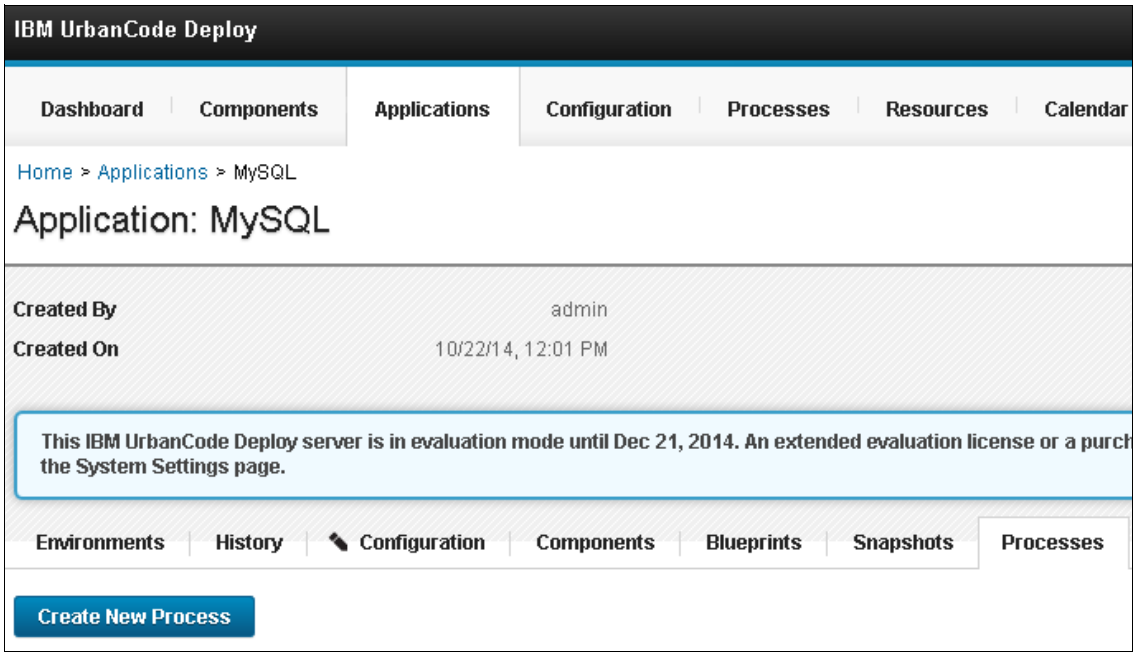


Figure 7-59 Creating a new process in the application

26. Provide a name for the process and click **Save**, as shown in Figure 7-60 on page 624.

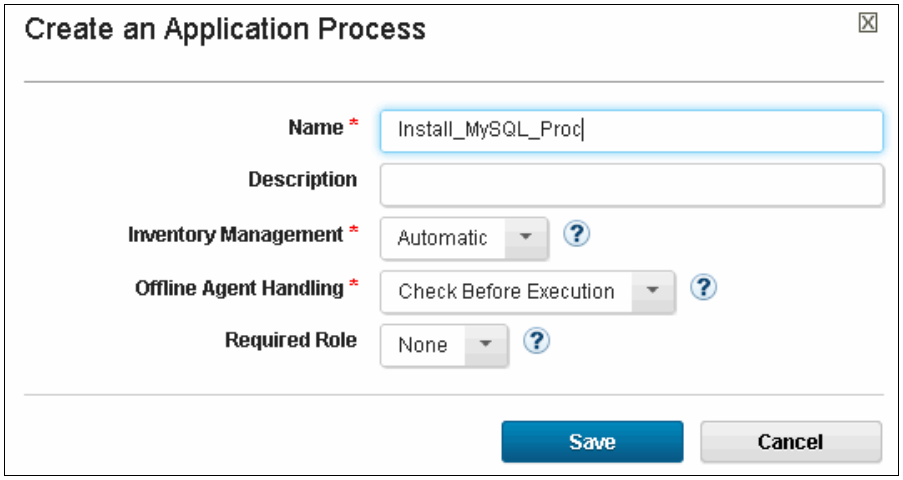


Figure 7-60 Providing a name for the application process

27. On the Process line, click **Edit**, as shown in Figure 7-61.

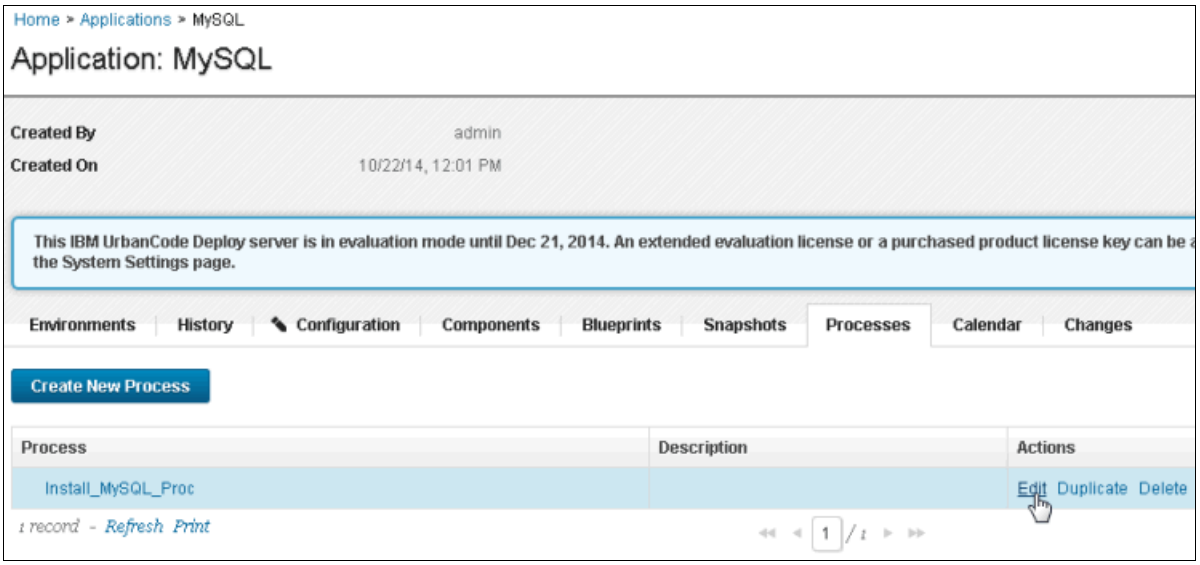


Figure 7-61 Modifying your application process

28. In the Step palette, drag **Install Component** to the Designer, as shown in Figure 7-62 on page 625.

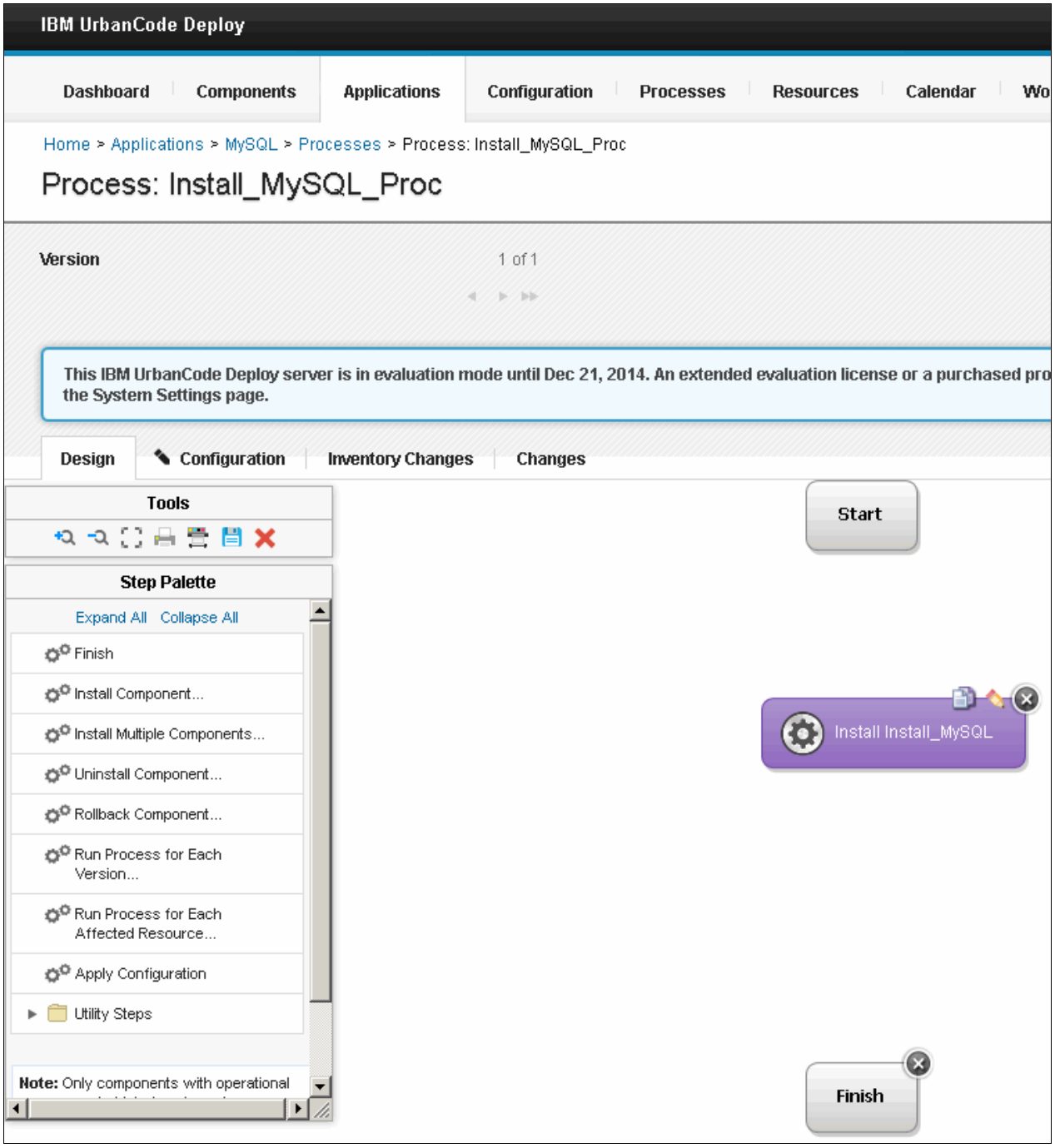


Figure 7-62 Adding Install Component to your application process

29. Provide a name for the step, select the component to install (in this example, there is only one component), select the component process (in this example, the component has only one process), and click **Save**, as shown in Figure 7-63 on page 626.

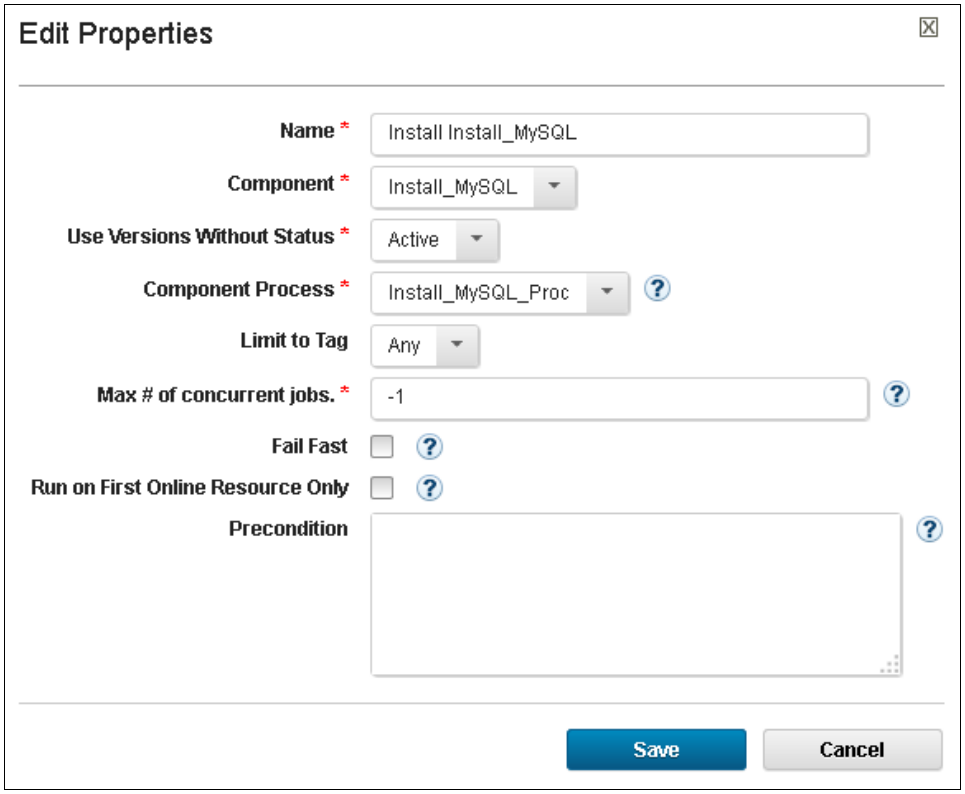


Figure 7-63 Modifying the properties of the Install Component step in the application process

30. Move your cursor onto the **Start** box so you can see the blue arrow. Drag this arrow to the **Install Component** box, and then do the same to link **Install Component** to the **Finish** boxes. Click the blue diskette icon (**Save** button), as shown in Figure 7-64 on page 627.

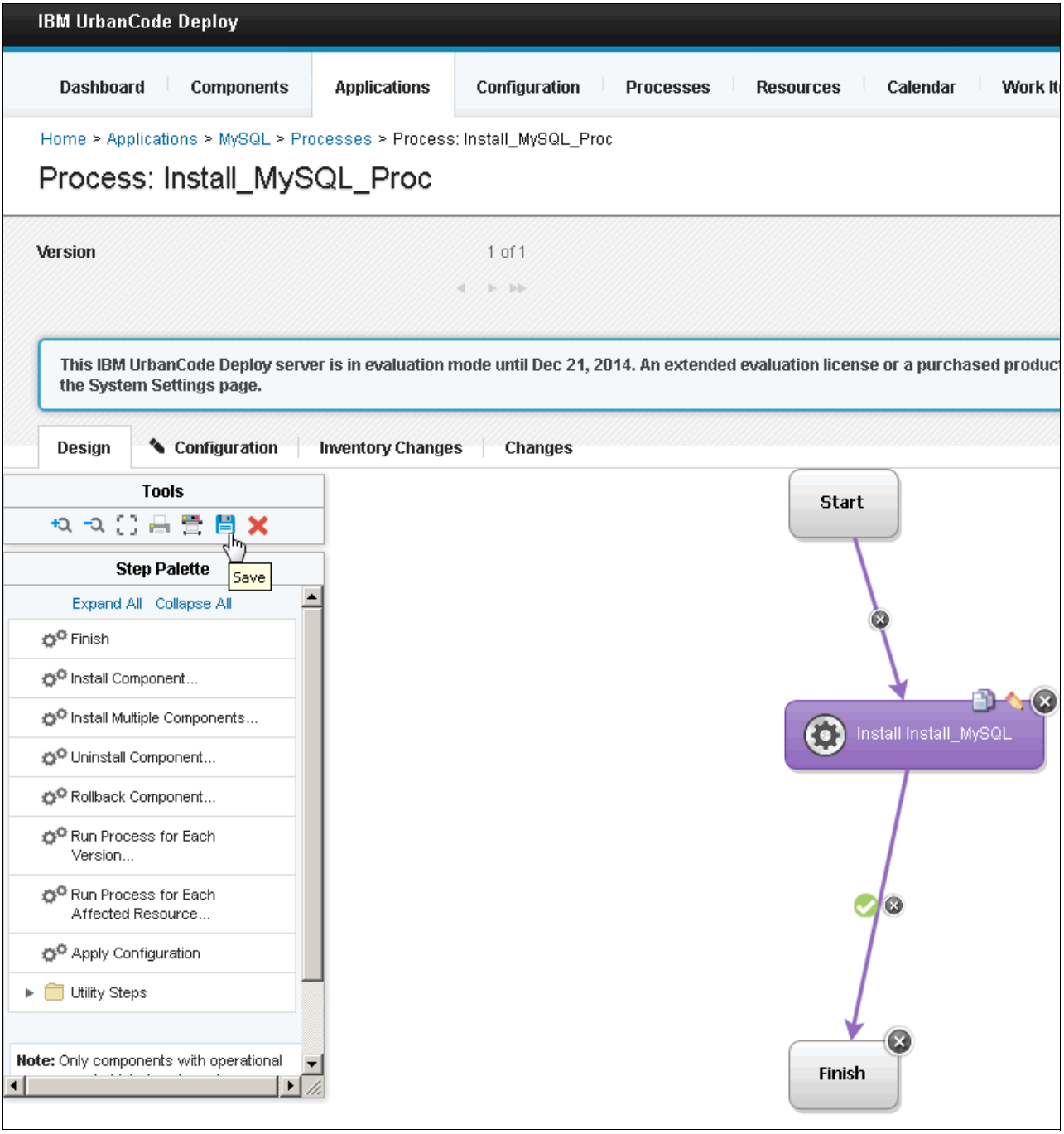


Figure 7-64 Linking the steps in the application process

After completing these steps, you should be able to use this application and component in the UrbanCode Deploy with Patterns, to deploy an instance (virtual machine) and automatically download and install the application using the component created in UrbanCode Deploy.

7.2.5 Exporting and importing components from UrbanCode Deploy

For portability purposes, the components and applications created in IBM UrbanCode Deploy can be exported to a JSON file. JSON files are JavaScript Object Notation files, an open standard format that uses text to transmit data objects in pairs of attribute and values. These exported JSON files can be imported on other UrbanCode Deploy servers, allowing you to migrate the application definition between environments.

This section explains how to export the application created in 7.2.4, “Creating components and applications in UrbanCode Deploy” on page 609, to a JSON file and then shows how you can import this application to UrbanCode Deploy, allowing it to be used by UrbanCode Deploy with Patterns, as described in 7.2.6, “Deploying environments in UrbanCode Deploy with Patterns using components created in UrbanCode Deploy” on page 635:

1. Log in to the UrbanCode Deploy server as the `admin` user.
2. Click **Applications**, select your application and click **Actions > Export**, as shown in Figure 7-65.

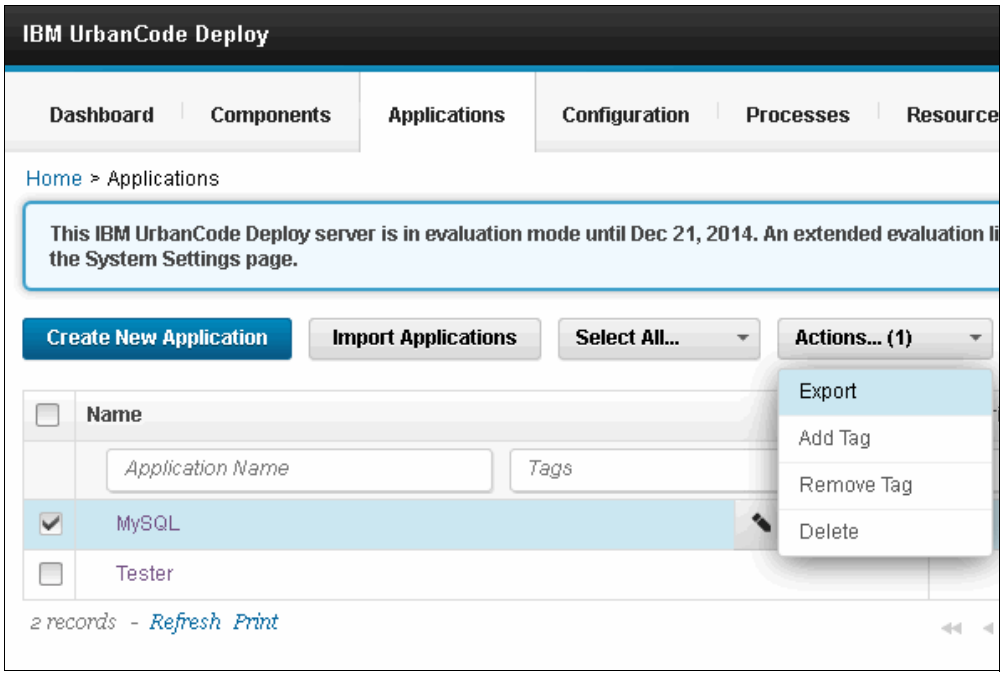


Figure 7-65 Exporting the application from IBM UrbanCode Deploy

3. You will be prompted to save the file with the `.json` extension. Save this file on an appropriate directory. This file contains the complete instructions for IBM UrbanCode to re-create this application, as shown in Example 7-33.

Example 7-33 Application exported from UrbanCode Deploy to a JSON file

```
[{
    "name": "MySQL",
    "description": "",
    "enforceCompleteSnapshots": false,
    "propSheet": {"properties": []},
    "components": [{
        "name": "Install_MySQL",
        "description": "",
```

```
        "created": 1413986093289,
        "componentType": "STANDARD",
        "importAutomatically": false,
        "useVfs": true,
        "active": true,
        "deleted": false,
        "defaultVersionType": "FULL",
        "cleanupDaysToKeep": 0,
        "cleanupCountToKeep": 0,
        "sourceConfigPluginName": "File System (Versioned)",
        "envPropDefs": [],
        "versionPropDefs": [],
        "propSheetGroup": {"propSheets": [
            {
                "name": "FileSystemVersionedComponentProperties",
                "properties": [
                    {
                        "name": "basePath",
                        "value": "\/root\/mysql",
                        "description": "",
                        "secure": false
                    },
                    {
                        "name": "extensions",
                        "description": "",
                        "secure": false
                    },
                    {
                        "name": "saveFileExecuteBits",
                        "value": "false",
                        "description": "",
                        "secure": false
                    }
                ]
            },
            {
                "name": "template",
                "properties": []
            },
            {
                "name": "custom",
                "properties": []
            }
        ]},
        "processes": [{
            "name": "Install_MySQL_Proc",
            "description": "",
               "defaultWorkingDir":
"${p:resource\/work.dir}\/${p:component.name}",
               "takesVersion": true,
               "inventoryActionType": "ADD",
               "status": "Active",
               "configActionType": "ADD",
               "active": true,
```

```
            "path":
"components\/952617df-daa4-42d9-a965-be768277369a\/processes\/fbc29251-eb33-
4611-9900-ed7fa6bdf1b2",
            "propDefs": [],
            "rootActivity": {
                "edges": [
                    {
                        "to": "Create Directories",
                        "type": "ALWAYS",
                        "value": ""
                    },
                    {
                        "to": "MySQL Shell",
                        "from": "Download MySQL Artifacts",
                        "type": "SUCCESS",
                        "value": ""
                    },
                    {
                        "to": "eea80cac4b86cc6da49983cbc78223",
                        "from": "MySQL Shell",
                        "type": "SUCCESS",
                        "value": ""
                    },
                    {
                        "to": "Download MySQL Artifacts",
                        "from": "Create Directories",
                        "type": "SUCCESS",
                        "value": ""
                    }
                ],
                "offsets": [
                    {
                        "name": "Download MySQL Artifacts",
                        "x": 7,
                        "y": 251,
                        "h": 60,
                        "w": 220
                    },
                    {
                        "name": "Create Directories",
                        "x": -38,
                        "y": 145,
                        "h": 60,
                        "w": 210
                    },
                    {
                        "name": "eea80cac4b86cc6da49983cbc78223",
                        "x": 0,
                        "y": 420,
                        "h": 50,
                        "w": 80
                    },
                    {
                        "name": "MySQL Shell",
                        "x": 66,
```

```
                    "y": 340,
                    "h": 60,
                    "w": 150
                }
            ],
            "type": "graph",
            "name": "6000f345-7115-4595-9b45-135bcfbe133e",
            "children": [
                {
                    "type": "finish",
                    "name": "eea80cac4b86cc6da49983cbc78223",
                    "children": []
                },
                {
                    "allowFailure": false,
                    "useImpersonation": false,
                    "showHidden": false,
                    "impersonationUseSudo": false,
                    "workingDir": "\/root\/mysql-installer",
                    "commandName": "Create Directories",
                    "pluginName": "File Utils",
                    "pluginVersion": 31,
                    "properties": {"dir": "."},
                    "type": "plugin",
                    "name": "Create Directories",
                    "children": []
                },
                {
"allowFailure": false,
"useImpersonation": false,
"showHidden": false,
"impersonationUseSudo": false,
"workingDir": "\/root\/mysql-installer",
"commandName": "Download Artifacts",
"pluginName": "IBM UrbanCode Deploy Versioned File Storage",
                 "pluginVersion": 21,
                    "properties": {
                        "directoryOffset": ".",
                        "artifactSetBaseDir": "",
                        "fileIncludePatterns": "**\/*",
                        "fileExcludePatterns": "",
                        "syncMode": "true",
                        "fullVerification": "true",
                        "setFileExecuteBits": "false",
                        "verifyFileIntegrity": "false",
                        "versionId": "${p:version.id}",
                        "serverUrl": "${p:server.url}",
                        "compId": "${p:component.id}",
                        "resId": "${p:resource.id}",
                        "charset": ""
                    },
                    "type": "plugin",
                    "name": "Download MySQL Artifacts",
                    "children": []
                },
```

```
                    {
                        "allowFailure": false,
                        "useImpersonation": false,
                        "showHidden": false,
                        "impersonationUseSudo": false,
                        "commandName": "Shell",
                        "pluginName": "Shell",
                        "pluginVersion": 3,
                        "properties": {
                            "directoryOffset": ".",
                            "shellInterpreter": "",
                               "scriptBody": "#!\/bin\/sh\n\nrpm -ivh
\/root\/mysql-installer\/mysql-5.1.71-1.el6.x86_64.rpm",
                               "runAsDaemon": "false",
                               "outputFile": ""
                           },
                           "type": "plugin",
                           "name": "MySQL Shell",
                           "children": []
                       }
                   ]
               }
           }],
           "configTemplates": [],
           "genericProcesses": []
       }],
       "environments": [
              {
               "name": "simple_mysql_component_env",
               "description": "",
               "color": "#ffffff",
               "requireApprovals": false,
               "exemptProcesses": [],
               "cleanupDaysToKeep": 0,
               "cleanupCountToKeep": 0,
               "order": 0,
               "active": true,
               "propSheet": {"properties": []},
               "baseResources": [{
                   "path": "\/Base Resource for environment
simple_mysql_component_env",
                   "id": "618f8c39-bb49-432c-8113-7db4fd745a57"
               }],
               "componentPropSheets": [{
                   "properties": [],
                   "componentName": "Install_MySQL"
               }],
               "versionConditions": []
           },
           {
               "name": "mysql_blueprint_env",
               "description": "",
            "color": "#ffffff",
            "requireApprovals": false,
            "exemptProcesses": [],
```

```
        "cleanupDaysToKeep": 0,
        "cleanupCountToKeep": 0,
        "order": 1,
        "active": true,
        "propSheet": {"properties": []},
        "baseResources": [{
            "path": "\/Base Resource for environment mysql_blueprint_env",
            "id": "9e9412d5-c115-4732-b860-b0f562fdf820"
        }],
        "componentPropSheets": [{
            "properties": [],
            "componentName": "Install_MySQL"
        }],
     "versionConditions": []
 }
],
"processes": [{
    "name": "Install_MySQL_Proc",
    "description": "",
    "inventoryManagementType": "AUTOMATIC",
    "offlineAgentHandling": "PRE_EXECUTION_CHECK",
    "rootActivity": {
        "edges": [
            {
                "to": "945fcbfe6a10187710239096ce43d1",
                "from": "6cad735595c7495f7959f93fb83d57",
                "type": "SUCCESS",
                "value": ""
            },
            {
                "to": "6cad735595c7495f7959f93fb83d57",
                "type": "ALWAYS",
                "value": ""
            }
     ],
     "offsets": [
         {
             "name": "6cad735595c7495f7959f93fb83d57",
             "x": -3,
             "y": 170,
             "h": 50,
             "w": 190
         },
         {
             "name": "945fcbfe6a10187710239096ce43d1",
             "x": 0,
             "y": 420,
             "h": 50,
             "w": 80
         }
     ],
     "type": "graph",
     "name": "b788b9f4-3f03-4f2e-a076-8b7e13b552a2",
     "children": [
         {
```

```
                    "type": "finish",
                    "name": "945fcbfe6a10187710239096ce43d1",
                    "children": []
                },
                {
                    "componentName": "Install_MySQL",
                    "properties": [
                        {
                            "name": "failFast",
                            "value": "false"
                        },
                        {
                            "name": "runOnlyOnFirst",
                            "value": "false"
                        },
                        {
                            "name": "maxIteration",
                            "value": "-1"
                        }
                    ],
                    "type": "componentEnvironmentIterator",
                    "name": "6cad735595c7495f7959f93fb83d57",
                    "children": [{
                        "componentName": "Install_MySQL",
                        "status": "Active",
                        "type": "inventoryVersionDiff",
                        "name": "2c236e81-ca4f-4252-b05e-a5b2fab56c76",
                        "children": [{
                            "componentProcessName": "Install_MySQL_Proc",
                            "componentName": "Install_MySQL",
                            "allowFailure": false,
                            "properties": {},
                            "type": "componentProcess",
                            "name": "Install Install_MySQL",
                            "children": []
                           }]
                       }]
                   }
               ]
            },
            "propDefs": []
    }],
    "genericProcesses": []
}]
```


4. When you decide to import this application, log in to your UrbanCode Deploy server where the application should be imported, and click **Applications > Import Applications**, as shown in Figure 7-66 on page 635.

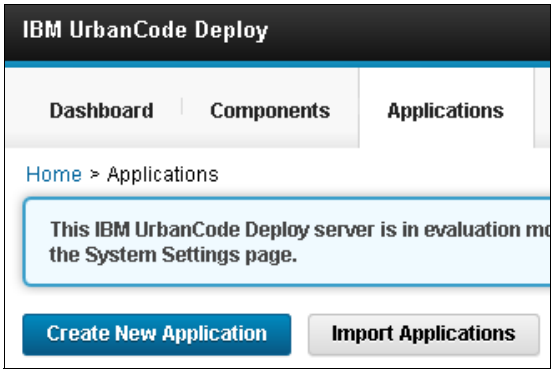


Figure 7-66 Importing applications to IBM UrbanCode Deploy

5. Click **Browse**, select the JSON file exported in step 3 on page 628, and then click **Submit**, as shown in Figure 7-67.

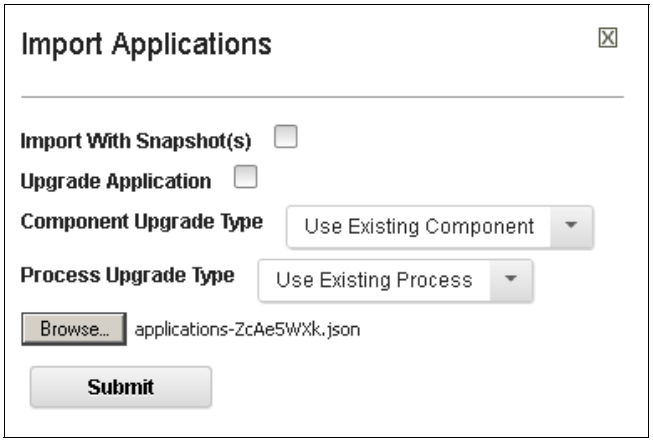


Figure 7-67 Submitting the import of the application in IBM UrbanCode Deploy

6. After completing this task, the application is imported and will be available to use by following the instructions described in 7.2.6, “Deploying environments in UrbanCode Deploy with Patterns using components created in UrbanCode Deploy” on page 635.

7.2.6 Deploying environments in UrbanCode Deploy with Patterns using components created in UrbanCode Deploy

After connecting UrbanCode Deploy with Patterns to your OpenStack cloud, and creating components in UrbanCode Deploy, you can use the UCDP Server to design and deploy environments in your cloud using the available images, volumes, networks, and SSH key pairs from IBM Cloud Manager and the application and components from UrbanCode Deploy.

When you design a blueprint and provision an environment by using this blueprint, UrbanCode Deploy with Patterns will prepare a HOT template according to the blueprint instructions, and use its heat engine to create a stack in IBM Cloud Manager.

This section describes how to create a blueprint with an instance connected to a network and to a volume, and also install an application in this instance by using a component from UrbanCode Deploy:

1. Log in to IBM UrbanCode Deploy with Patterns using a username that is connected to your OpenStack cloud, according to instructions from 7.2.2, “Connect IBM UrbanCode Deploy with Patterns to the OpenStack cloud” on page 591. In this example, we used the `admin` user.
2. Click **Blueprints**, as shown in Figure 7-68.

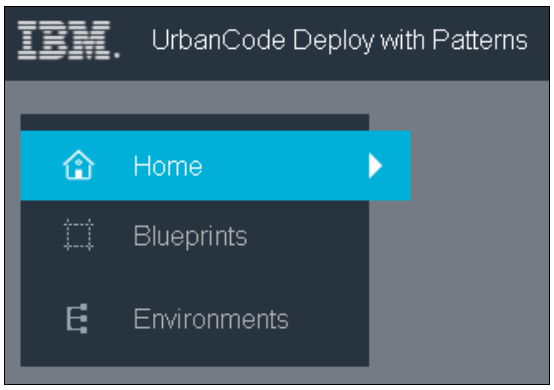


Figure 7-68 Click Blueprints

3. Click **New** to start a new blueprint. Provide a name for your blueprint and click **Save**, as shown in Figure 7-69.

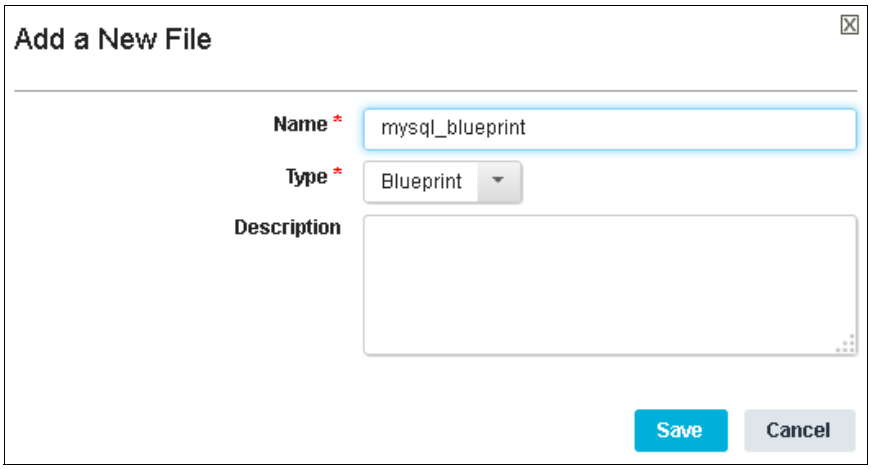


Figure 7-69 Creating a new blueprint

4. Drag your desired Glance image to the diagram to create an instance. Drag the desired network and move the blue arrow to connect the network with the instance, as shown in Figure 7-70 on page 637.

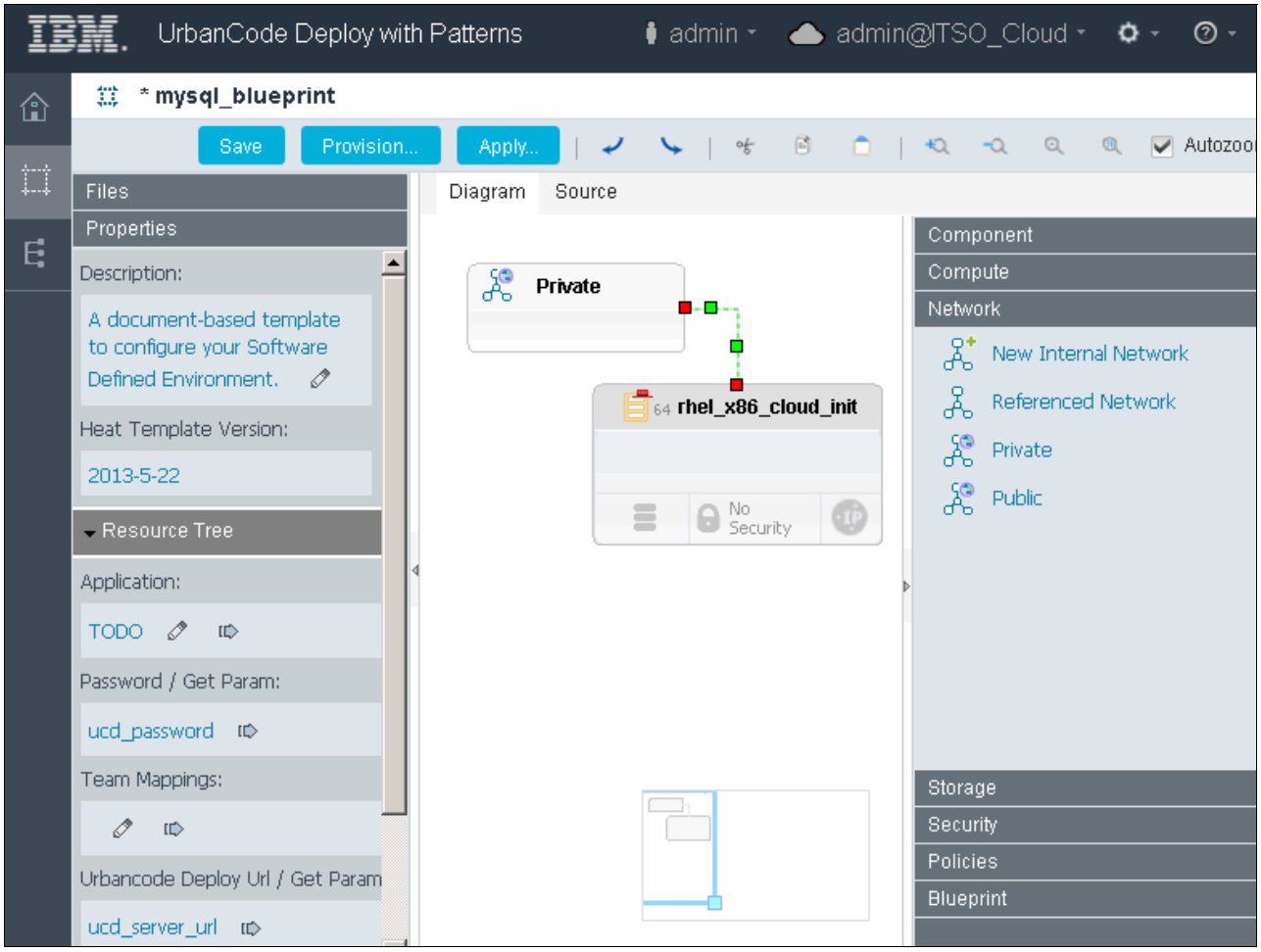


Figure 7-70 Creating the instance and network in the UCDP diagram

5. Click the **Component** section, and drag the component that you previously created in UrbanCode Deploy onto the instance, as shown in Figure 7-71 on page 638.

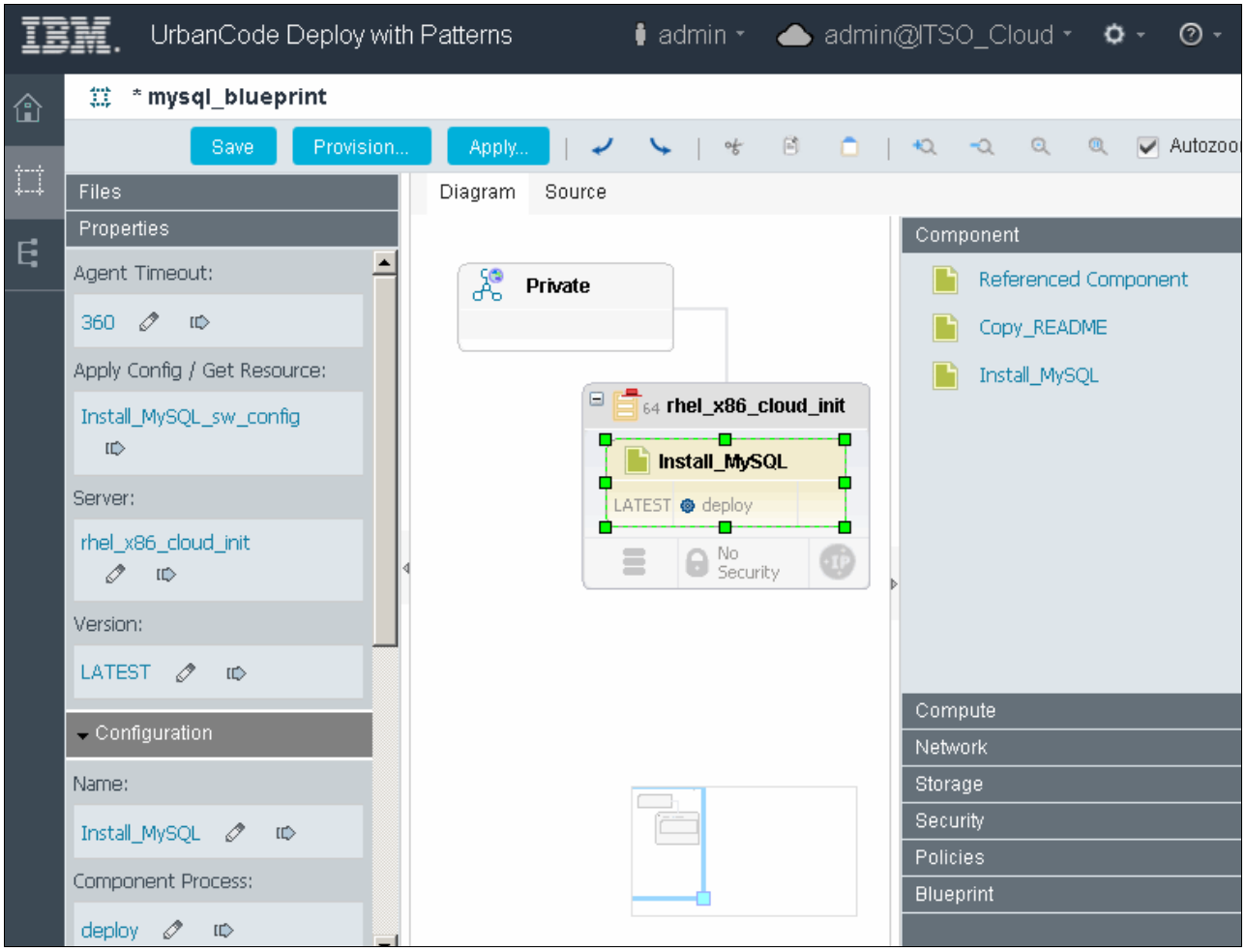


Figure 7-71 Adding a component to an instance

You can repeat this action for as many components you want to add to your instances. This way you can design and provision a complete environment in your OpenStack cloud, including enterprise applications, such as IBM DB2, IBM InfoSphere BigInsights, IBM Websphere Application Server, and so on.

6. In case you want to add a Cinder volume to your instance, click in **Storage** and drag the desired volume onto your instance, as shown in Figure 7-72.

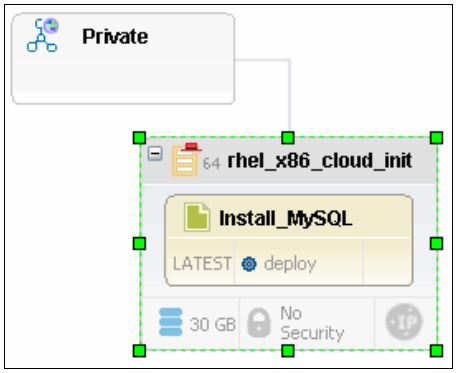


Figure 7-72 Instance with an UrbanCode Deploy component and a Cinder volume

Note: You can see on the lower left corner of the instance the 30 GB disk sign, which indicates that a volume has been attached to this instance.

7. Click **Source** and look for the UrbanCode `resource_tree`. Modify the `application` entry to provide the name of your application from UrbanCode Deploy, created in step 21 on page 621, as shown in Example 7-34.

Example 7-34 Providing the UrbanCode Deploy application name

```
resources:
## REFERENCE {"Private":{"type": "OS::Neutron::Net", "properties": {"name":
"Private", "network_id": "d828cdb1-e926-4f94-b276-29a41471e5b6"}}}

  # declaration of template resources
  resource_tree:
    type: IBM::UrbanCode::ResourceTree
    properties:
       urbancode_deploy_url: { get_param: ucd_server_url }
       username: { get_param: ucd_user }
       password: { get_param: ucd_password }
       team_mappings:
       application: MySQL # Your application name, e.g. MoneyThatMatters
```

Note: Note that this parameter was initially filled with the string `TODO`. As described in section 7.2.3, “Deploying environments in UrbanCode Deploy with Patterns” on page 598, all the `TODO` entries must be solved before provisioning an environment. Some of these entries have combo boxes for you to choose the available values when you click `Provision`.

8. You can also modify the HOT template to set a different hostname and tell the instance to use a Configuration Drive rather than the metadata service to obtain the information from the Heat Engine, as shown in Example 7-35.

Example 7-35 Modifying the HOT template

```
rhel_x86_cloud_init:
    type: OS::Nova::Server
    properties:
      networks:
          - port: { get_resource: rhel_x86_cloud_init_Private_port }
      flavor: { get_param: flavor }
      image: "f98aead8-ebbc-4cea-901c-a72b97366487"
      key_name: { get_param: key_name }
      name: "mysql_server"
      config_drive: "true"
```

9. Look for the Volume Attachment section, and check whether the mount point of the volume is correct. In this case, the Glance image has a single disk (/dev/vda) and the new Cinder volume should be attached as the second disk (/dev/vdb), as shown in Example 7-36.

Example 7-36 Providing mount point attachment information

```
rhel_x86_cloud_init_gpfs_vol1_attachment:
    type: OS::Cinder::VolumeAttachment
    properties:
      volume_id: 37b5bbce-aebd-4bf8-bdc5-e29c63ccdd2e
      instance_uuid: { get_resource: rhel_x86_cloud_init }
      mountpoint: /dev/vdb
```

10. Example 7-37 shows the complete HOT template prepared by UrbanCode Deploy with Patterns in this section.

Example 7-37 HOT template prepared by UrbanCode Deploy with Patterns

```
heat_template_version: 2013-05-23

description: >
  A document-based template to configure your Software Defined Environment.

parameters:
  server_url:
    type: string
    description: The base URL for the server which hosts files required by user data. Do *not*
add a trailing slash.
    default: http://172.16.21.151:8080/landscaper
  ucd_server_url:
    type: string
    description: The server URL for agent communication to UrbanCode Deploy. Do *not* add a
trailing slash.
    default: https://172.16.21.151:7443
  ucd_user:
    type: string
    description: The user credential for the UrbanCode Deploy server.
    default: admin
  ucd_password:
    type: string
    description: The user credential for the UrbanCode Deploy server.
    hidden: true
  flavor:
    type: string
    description: Flavor to be used for compute instance
    default: "m1.small"
  key_name:
    type: string
    description: Name of key-pair to be used for compute instance
    default: "TODO"
  network_id_Private:
    type: string
    description: Generated to reference 'Private' network
    default: "d828cdb1-e926-4f94-b276-29a41471e5b6"

resources:
```

```
## REFERENCE {"Private":{"type": "OS::Neutron::Net", "properties": {"name": "Private",
"network_id": "d828cdb1-e926-4f94-b276-29a41471e5b6"}}}

  # declaration of template resources
  resource_tree:
    type: IBM::UrbanCode::ResourceTree
    properties:
       urbancode_deploy_url: { get_param: ucd_server_url }
       username: { get_param: ucd_user }
       password: { get_param: ucd_password }
       team_mappings:
       application: MySQL # Your application name, e.g. MoneyThatMatters

  rhel_x86_cloud_init:
    type: OS::Nova::Server
    properties:
      networks:
          - port: { get_resource: rhel_x86_cloud_init_Private_port }
      flavor: { get_param: flavor }
      image: "f98aead8-ebbc-4cea-901c-a72b97366487"
      key_name: { get_param: key_name }
      name: "mysql_server"
      config_drive: "true"
      user_data:
        str_replace:
          template: |
              #!/usr/bin/python
              import subprocess, urllib, urllib2, sys, time
              from StringIO import StringIO

              def main():
                  query_string = urllib.urlencode({ "ucd_server_url" : "_ucd_server_url_",
"agent_name" : _agent_name_ })
                  url = ("%s/rest/user_data/profiles?%s") % ("_server_url_", query_string )
                  print "**** Attempting to curl given URL endpoint \"%s\"" % url
                  for x in range(0, 5):
                      try:
                          version = urllib2.urlopen("_server_url_/server/version",
timeout=5).read()
                      except urllib2.URLError, e:
                          print "***** Network access not yet available. Will try again ..."
                          time.sleep(5)
                  script = urllib2.urlopen(url, timeout=300).read()
                  print "****** SCRIPT RESULT: %s" % str(subprocess.call(script, shell=True))
              if __name__ == "__main__":
                  sys.exit(main())

          params:
            _agent_name_:
               str_replace:
                  template: |
                    "_application_name_._environment_name_._server_name_"
                  params:
                    _application_name_: { get_attr: [resource_tree, application_name ] }
                    _environment_name_: { get_param: "OS::stack_name" }
```

```
                    _server_name_: "rhel_x86_cloud_init"
              _ucd_server_url_: { get_param: ucd_server_url }
              _server_url_: { get_param: server_url }

  rhel_x86_cloud_init_gpfs_vol1_attachment:
    type: OS::Cinder::VolumeAttachment
    properties:
      volume_id: 37b5bbce-aebd-4bf8-bdc5-e29c63ccdd2e
      instance_uuid: { get_resource: rhel_x86_cloud_init }
      mountpoint: /dev/vdb

  Install_MySQL:
    type: IBM::UrbanCode::SoftwareDeploy::UCD
    properties:
      apply_config: { get_resource: Install_MySQL_sw_config }
      server: rhel_x86_cloud_init
      version: "LATEST"
      agent_timeout: "360"

  Install_MySQL_sw_config:
    type: IBM::UrbanCode::SoftwareConfig::UCD
    properties:
      name: Install_MySQL
      component_process: deploy
      component_process_timeout: "300"
      ucd_server_url: { get_param: ucd_server_url } # these should come from resource_tree
      ucd_username: { get_param: ucd_user }
      ucd_password: { get_param: ucd_password }
      application: { get_attr: [resource_tree, application_name] }
      environment_name: { get_attr: [resource_tree, environment_name] }
      inputs:

  rhel_x86_cloud_init_Private_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_param: network_id_Private }

outputs:
  blueprint_url:
    description: Blueprint Origin URL
    value:  http://172.16.21.151:8080/landscaper/view/projects?open=mysql_blueprint
```

11. When you complete modifying the HOT template according to the desired values, click **Save** and then click **Provision.** Provide an environment name, and then complete the relevant fields for your instance, including `flavor` (that your instance will use), `key_name` (SSH key pair to inject), and `network_id_Private` (network to connect with the instance). In the `ucd_password`, type the password of your UrbanCode Deploy user, which has access to the components you plan to use with this instance. In this case, the `admin` user is used in UrbanCode Deploy. When you complete providing the required information, click **Provision**, as shown in Figure 7-73 on page 643.

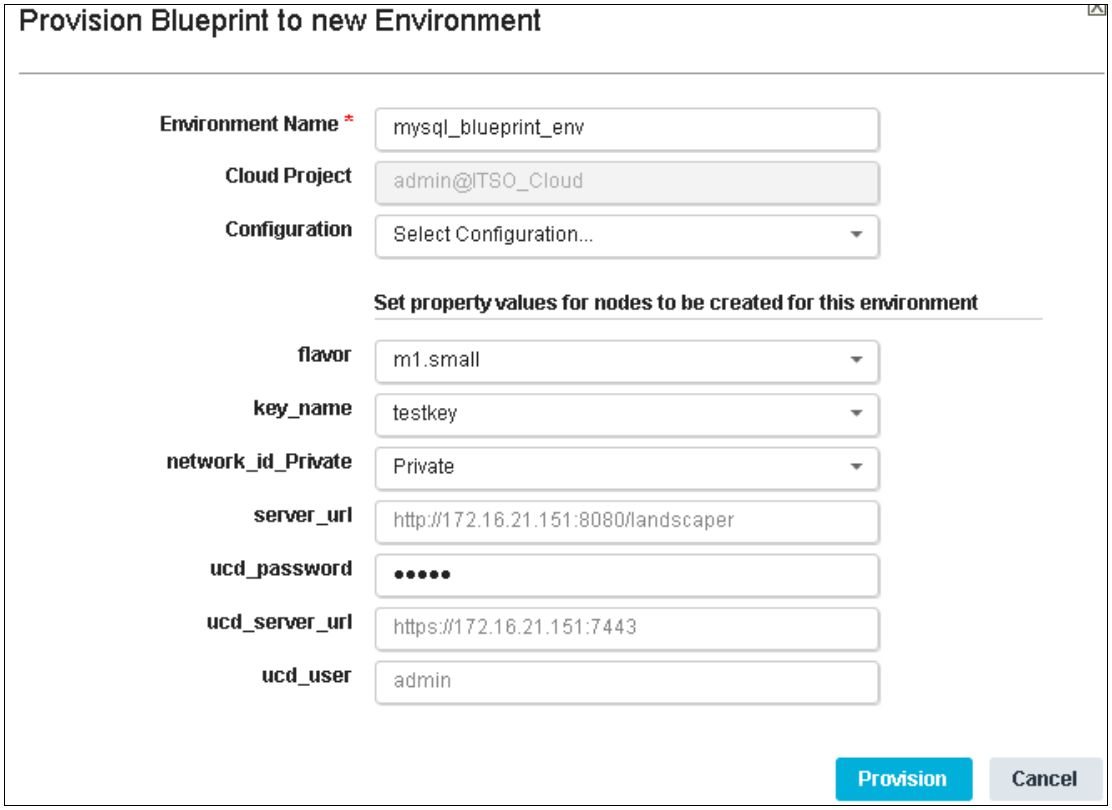


Figure 7-73 Provisioning an environment

12. Click **Environments**, and you should be able to see your environment being created, as shown in Figure 7-74 on page 644.

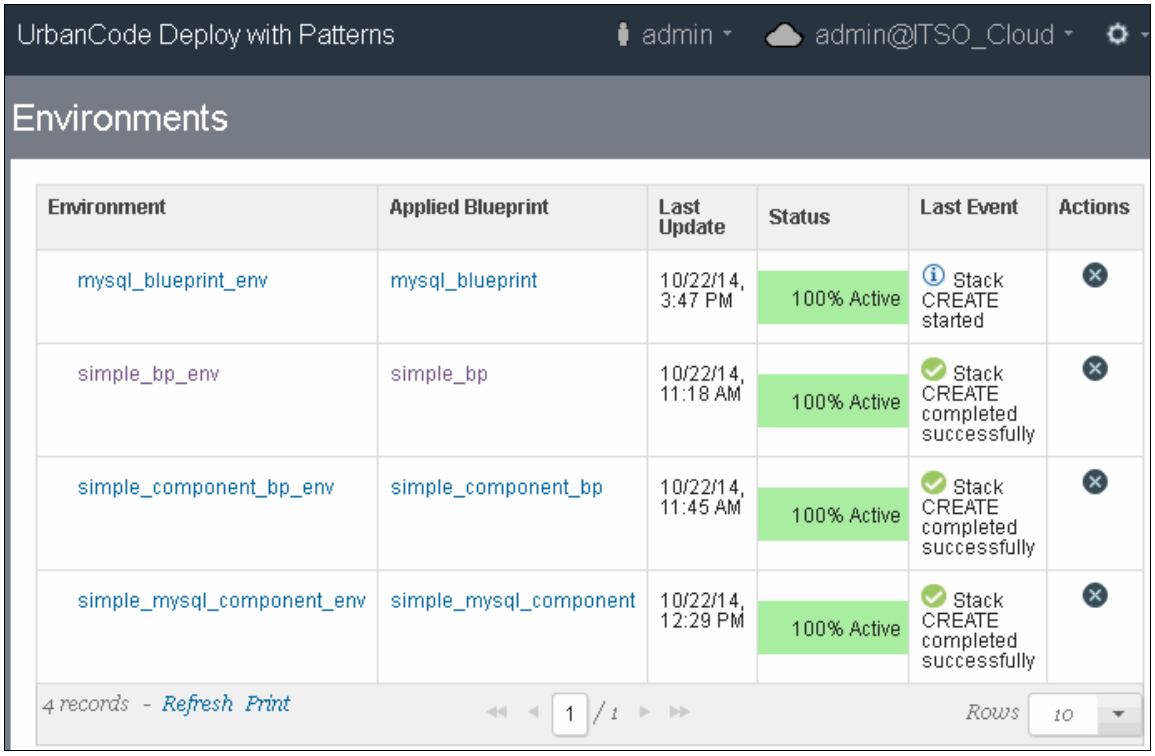


Figure 7-74 Environment using UrbanCode Deploy with Patterns with UrbanCode Deploy components

13. Click the environment name and you should be able to see all the components from this environment, including the instances that were deployed (and their IP addresses), the volumes that were attached to the instances, network connections, UrbanCode Deploy components, and so on, as shown in Figure 7-75.

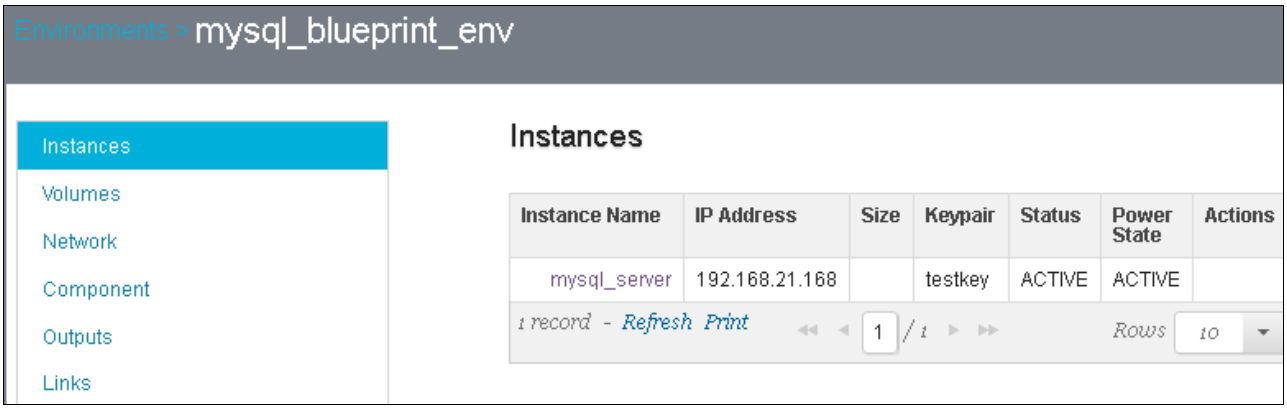


Figure 7-75 Details of the environment provisioned using UrbanCode Deploy with Patterns

14. If you go back to UrbanCode Deploy, click **Components**, and then click your component name, and you should be able to see a history of the instances and environments that were deployed and are using this component, as shown in Figure 7-76 on page 645.

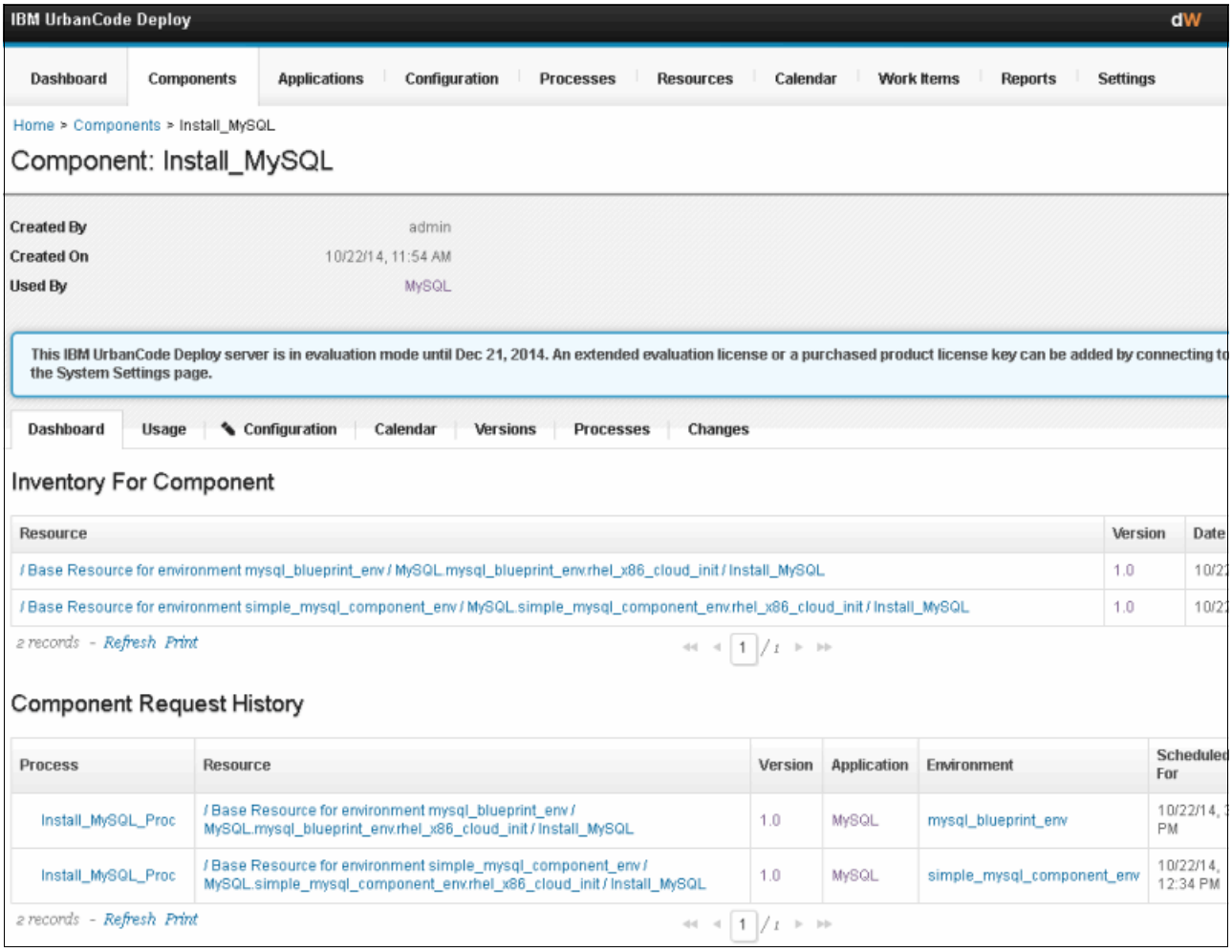


Figure 7-76 Environments provisioned using the UrbanCode Deploy component

15. In Component Request History, click **View Request** in the Actions column, so you can view details of the environment that is provisioned with this component. You should be able to see the details of each step executed by the component, including the start time and duration, as shown in Figure 7-77 on page 646.

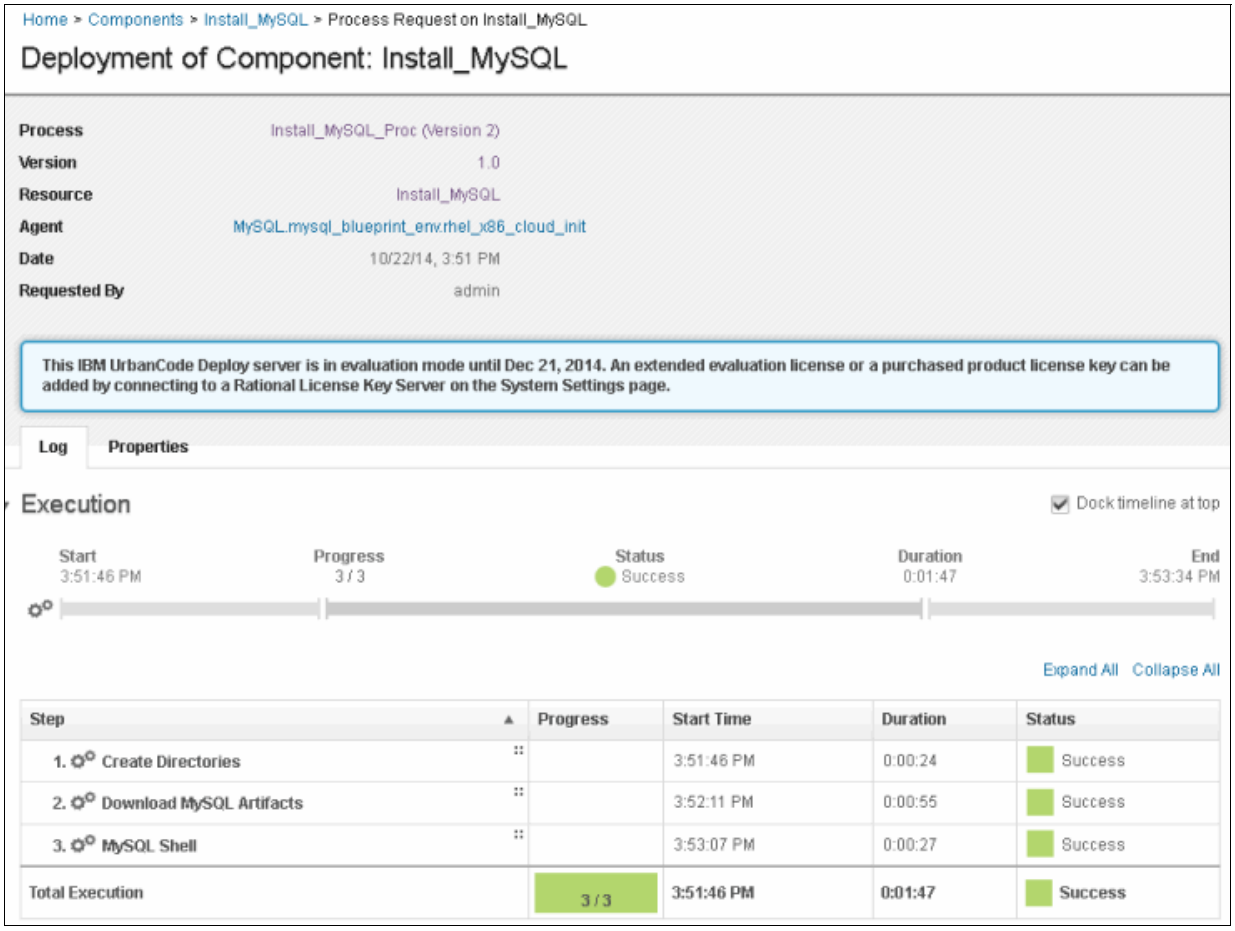


Figure 7-77 Details of the component execution in the instance

16. Move your cursor a specific Step, and you should be able to see the `Output Log` icon (similar to a `Command Prompt` icon). Click this Icon to see the output of the execution of this specific step in the instance. Figure 7-78 on page 647 shows the Output Log for the Shell step in the MySQL component, created in section 7.2.4, “Creating components and applications in UrbanCode Deploy” on page 609.

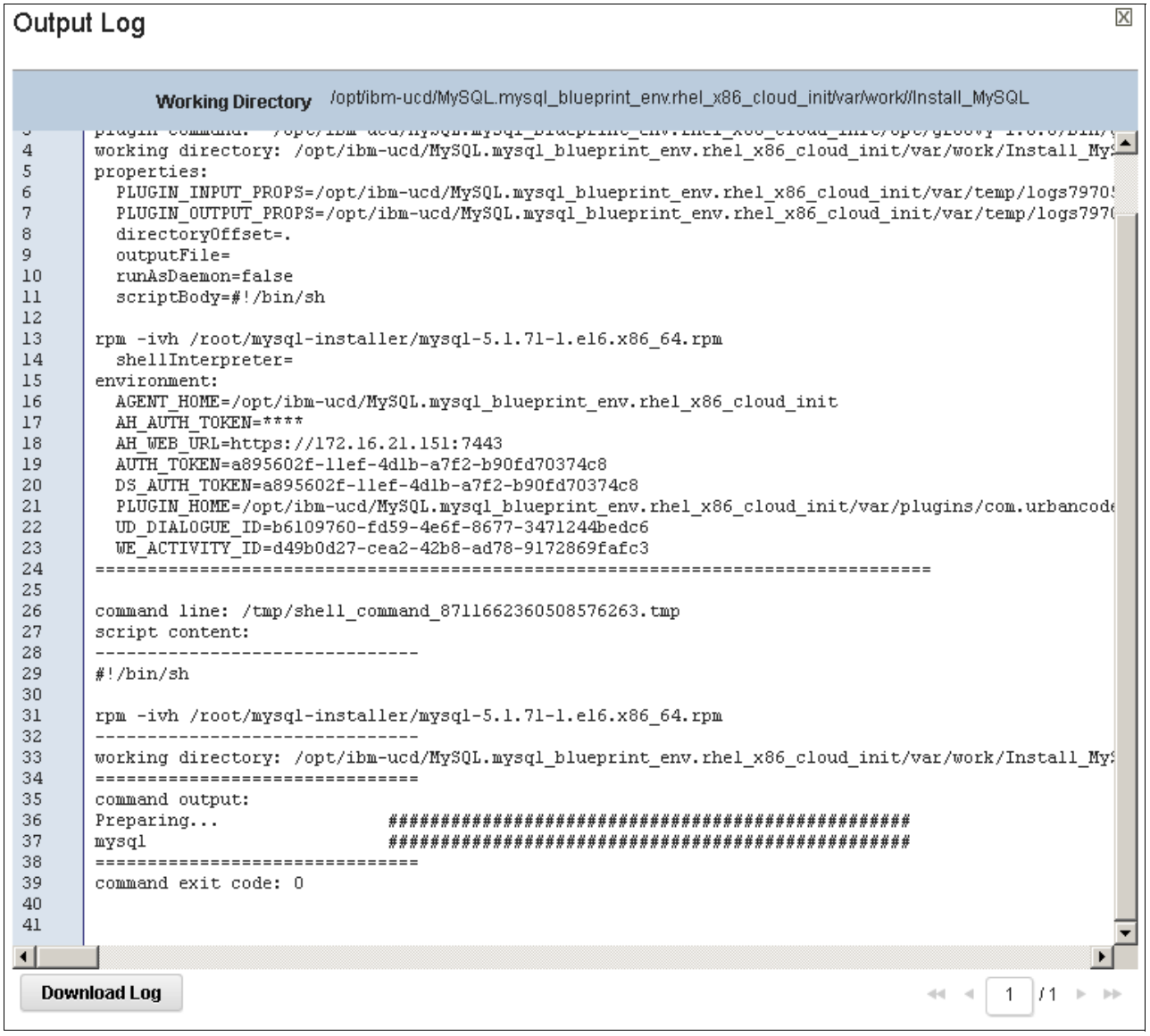


Figure 7-78 Output Log of the Shell Step in the MySQL component from UrbanCode Deploy

17. Log in to the instance provisioned by UrbanCode Deploy with Patterns. Use the IP obtained from step 13 on page 644. You can see in Example 7-38 that the hostname was set according to the HOT template, Cinder volume was attached, MySQL Server was installed (according to UrbanCode Deploy component) and SSH key pair was injected.

Example 7-38 Instance provisioned using UrbanCode Deploy with Patterns

```
[root@mysql-server ~]# hostname
mysql-server

[root@mysql-server ~]# fdisk -l /dev/vdb
Disk /dev/vdb: 32.2 GB, 32212254720 bytes
16 heads, 63 sectors/track, 62415 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

```
[root@mysql-server ~]# rpm -qa | grep mysql
mysql-libs-5.1.71-1.el6.x86_64
mysql-5.1.71-1.el6.x86_64

[root@mysql-server ~]# cat .ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA2FpRfVuEVDgEdAVuvw52PR13uwZVtezVXMRQ926ftpYOYx6k5lE
+bzLnMTiVB1+A5IDG49+5+IaskEvOF6bJLOfGO1+DREz9sGrfPu66pvIlLIIbF4A7rqsBETw/svNDAR
lOaU+xu7btbLocDKm/yPokPHB4coIAforR9u/KWDJbOkFAt/CwF6GddLnOtOCWA65uewjv28JRlx1nu
QEYFsjmkfunyY6WeB2GTpYj4rnaSB2B1OnptZhnLLEWgBkKyjBReDxe87/ksXBUws36UmAX+ArPalwX
rPmB4oO1sSbx7zWdVVgqkfrBZ6/jP81TGofFtu7nLpZq8R6vGsXkuQ==
root@cmocontroller.itso.ibm.com
```

Note: You can find details and logs of the UrbanCode Deploy agents in your instance. They will be available in the /opt/ibm-ucd directory with the name of the Agent. In the cloud setup for this publication, the agent log is:

```
/opt/ibm-ucd/MySQL.mysql_blueprint_env.rhel_x86_cloud_init/var/log/agent.
out
```

7.2.7 Importing blueprint definitions in IBM UrbanCode Deploy with Patterns

You can save the HOT template prepared by the IBM UrbanCode Deploy with Patterns (UCDP) to a text file and use this template to prepare a blueprint in the same or another UCDP Server, importing all its definitions and saving time of preparing the blueprint.

Note: If your blueprint uses components from the UrbanCode Deploy, the same components must be available at the UrbanCode Deploy with Patterns server where the HOT template will be used to build a blueprint. You can use the steps described in 7.2.5, “Exporting and importing components from UrbanCode Deploy” on page 628 to import such components.

The next steps explains how you can complete these tasks:

1. Create an application.
2. Delete the default HOT template from this application.
3. Copy a HOT template saved in a text file.
4. Paste the HOT template in the IBM UrbaCode Deploy with Patterns.
5. Launch the blueprint.

The next steps require a HOT template with the desired components for your blueprint, including Compute nodes, network configuration, Cinder volumes and applications. In case you have previously prepared a blueprint using the steps from section 7.2.6, “Deploying environments in UrbanCode Deploy with Patterns using components created in UrbanCode Deploy” on page 635, you can use its `Source` tab and save the HOT template from step 10 on page 640 to import it onto UrbanCode Deploy with Patterns:

1. Log in to IBM UrbanCode Deploy with Patterns using your OpenStack cloud user ID (which was previously connected to UrbanCode Deploy with Patterns using steps from section 7.2.2, “Connect IBM UrbanCode Deploy with Patterns to the OpenStack cloud” on page 591).
2. Click **Blueprints**, as shown in Figure 7-79 on page 649.

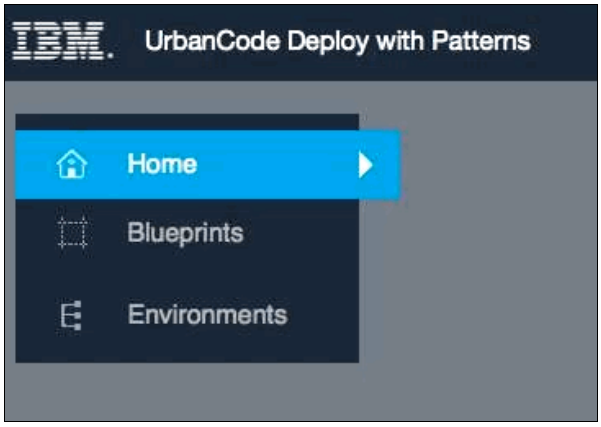


Figure 7-79 Creating a new blueprint

3. Click **New** and provide a name for your blueprint, as shown in Figure 7-80.

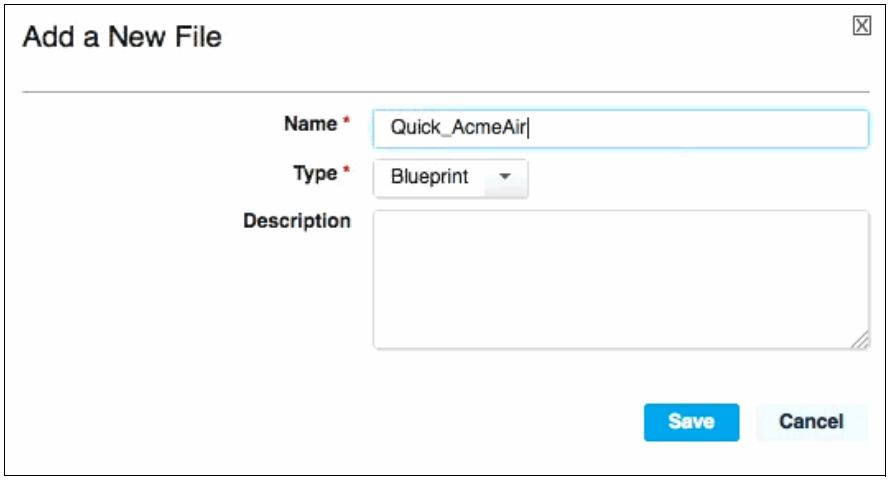


Figure 7-80 Providing a name for the blueprint

4. Click the **Source** tab, select all its content and delete all the lines. Do not leave any blank line (just the one where the cursor is inserted), as shown in Figure 7-81.

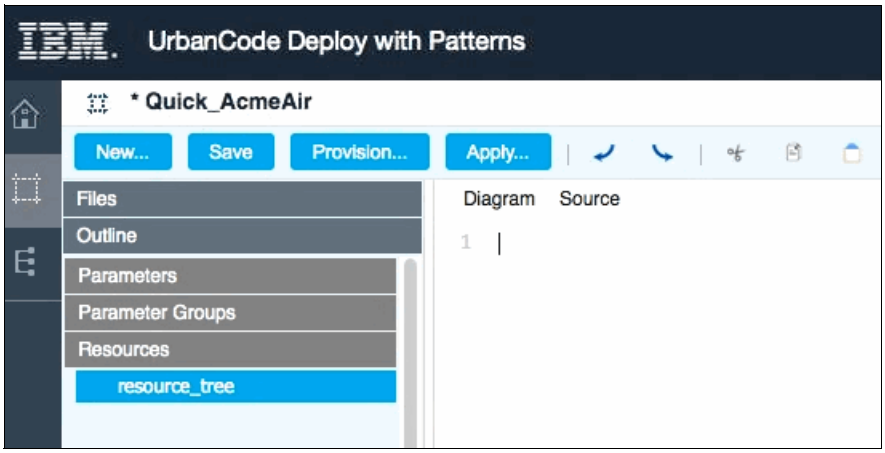


Figure 7-81 Clearing all blank lines from the Source tab

5. Copy your previously created HOT template to the Clipboard, as shown in Figure 7-82:

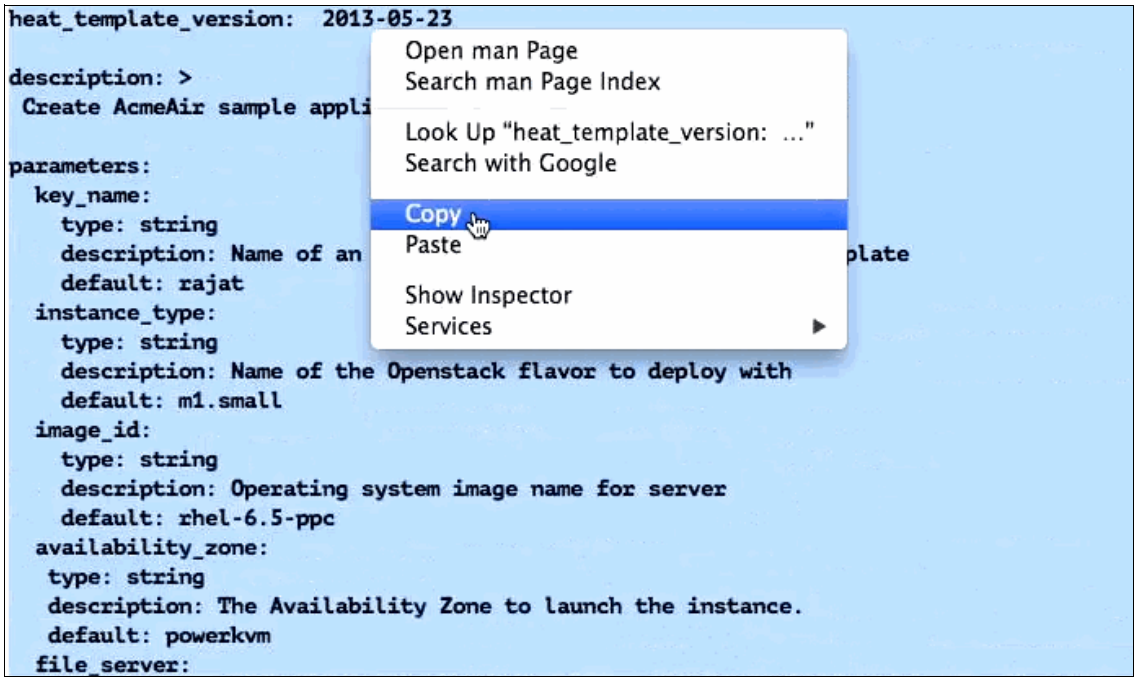


Figure 7-82 Copying the HOT template to the Clipboard

6. Go to the UrbanCode Deploy with Patterns, click the **Source** tab in your blueprint, paste it and click **Save**, as shown in Figure 7-83 on page 651.

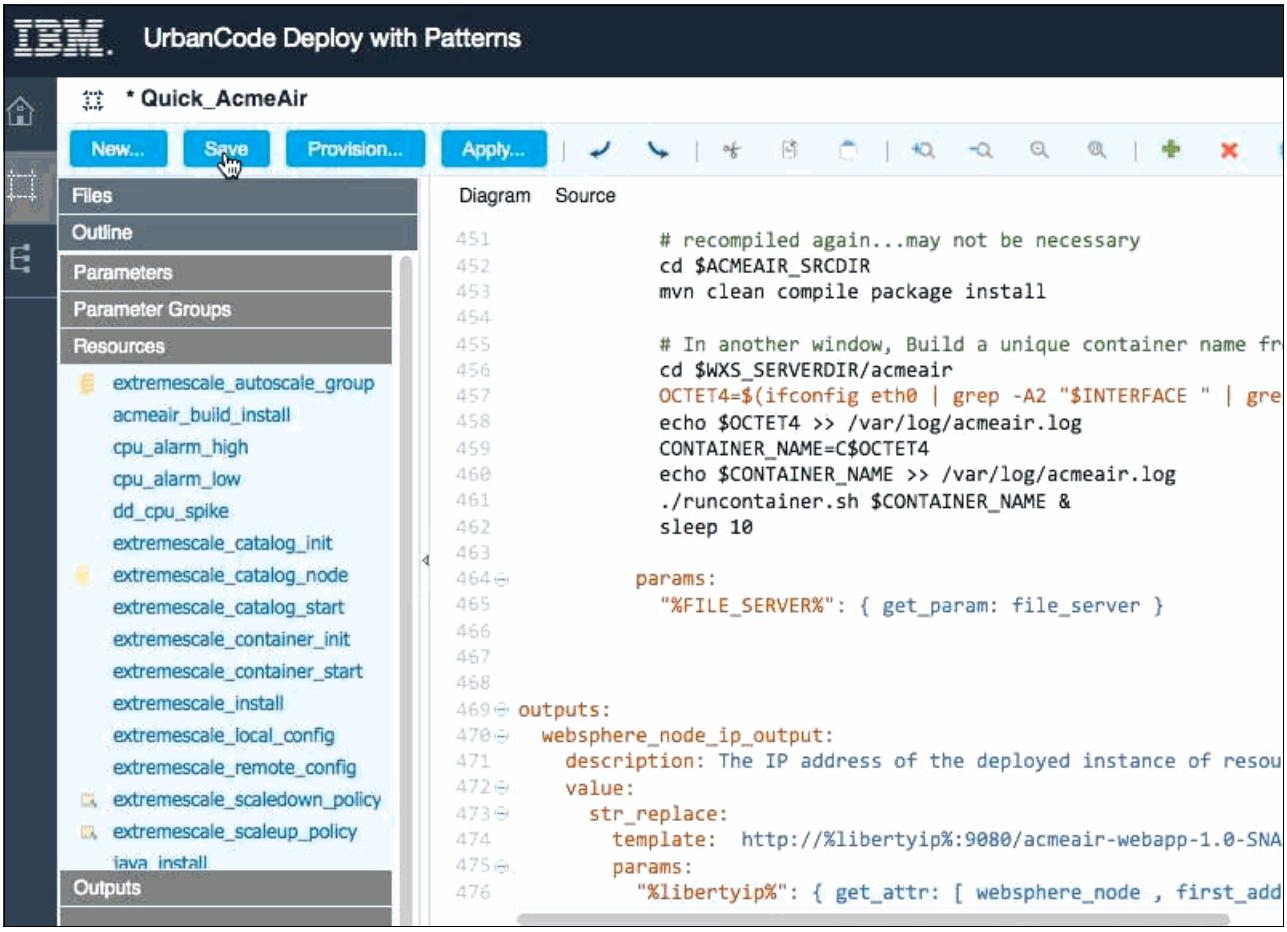


Figure 7-83 Pasting the HOT template and saving the blueprint

7. If you click the **Diagram** tab, you will be able to see the Diagram view for your HOT template, with the appropriate components, as shown in Figure 7-84.

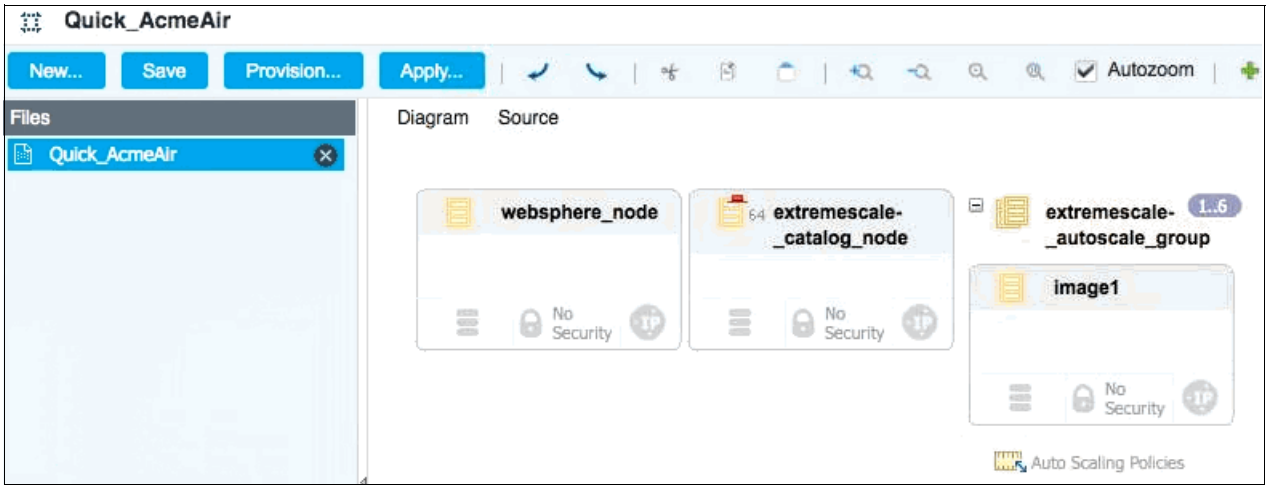


Figure 7-84 Diagram view of the HOT template

8. Click **Provision**. Select the property values for the nodes, including the `image_id` and `key_name` and click **Provision**, as shown in Figure 7-85 on page 652.

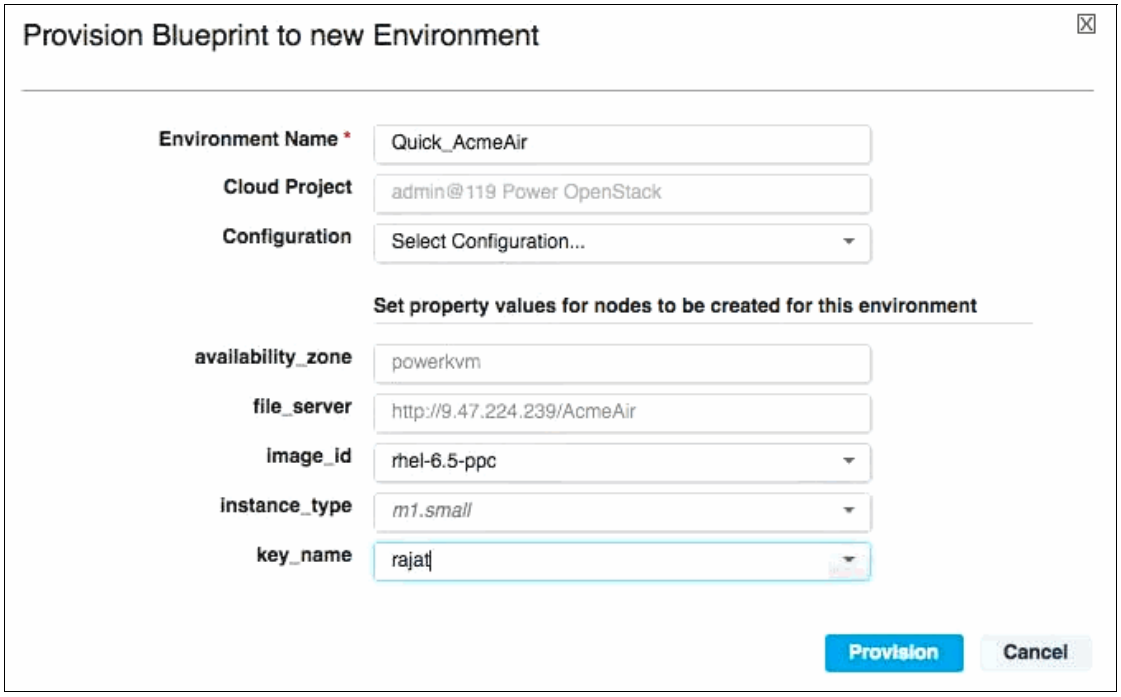


Figure 7-85 Selecting property values for the nodes and provisioning the blueprint

9. Click the **Environments** tab to see your blueprint being provisioned, as shown in Figure 7-86.

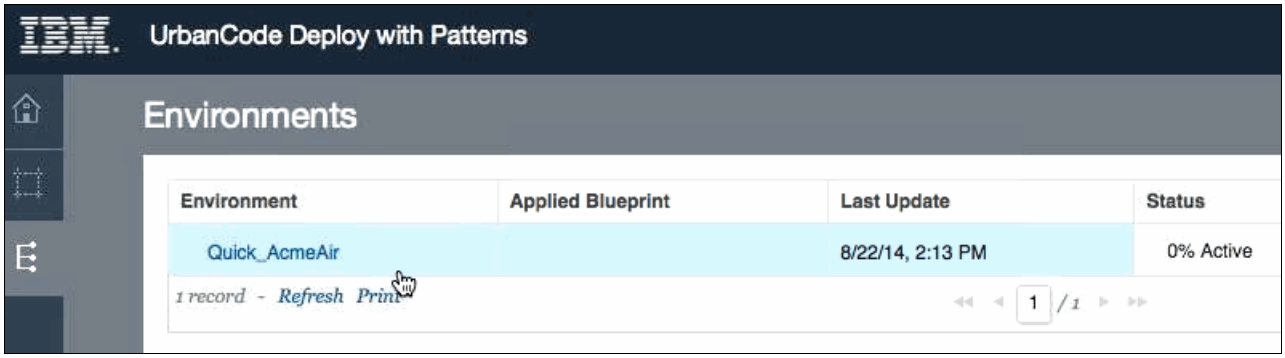


Figure 7-86 Tracking the provisioning progress in the Environments tab

10. You can click your environment to track the progress of each component, as shown in Figure 7-87 on page 653.

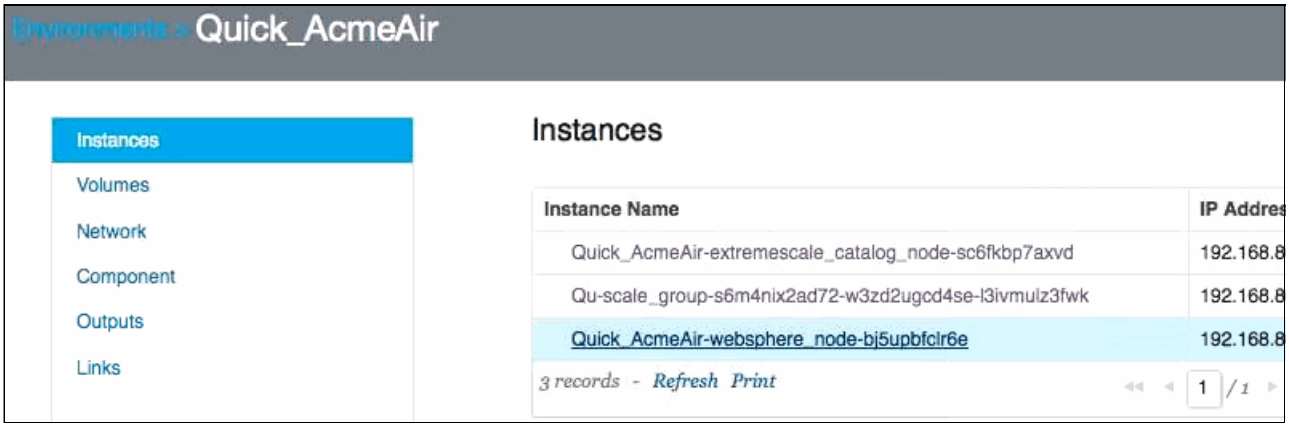


Figure 7-87 Tracking the progress of each component

After completing these steps, the components from your HOT template are deployed in your OpenStack cloud and your environment will be available for use.

7.2.8 Required corrections in the cloud setup for this publication

In the cloud setup for this publication, we used IBM UrbanCode Deploy Version 6.1.0.1.557329 and IBM UrbanCode Deploy with Patterns Version 6.1.0.1.557335. Provisioning environments was not possible after completing the UrbanCode Deploy with Patterns connection to the OpenStack cloud and creating components and applications in UrbanCode Deploy. This section describes the two error messages that were initially obtained and the steps taken to solve these problems.

Error due to personality property

When trying to provision a simple environment based on a blueprint with an instance connected to a network, the following error was seen in the UrbanCode Deploy with Patterns diagram (after hitting the `Provision` button) and also in the `/var/log/heat/engine.log` file:

```
2014-10-07 19:51:23.029 1463 ERROR heat.engine.parser
[req-97d9d907-d451-430e-ba7b-bbea4fa08d37 None] HT-EA96E26 HT-7CD4351 The
personality property may not contain greater than -1 entries.
```

This error is a known bug in the Heat engine, as described in the website:

`https://bugs.launchpad.net/heat/+bug/1333283`

At the time of writing this publication, the Heat engine available in UrbanCode Deploy with Patterns is exposed to this problem. To work around this error, follow these steps:

1. Create a backup of the following file:

   ```
   # cp -p /usr/lib/python2.6/site-packages/heat/engine/resources/server.py
   /usr/lib/python2.6/site-packages/heat/engine/resources/server.py.ORIG
   ```

2. Modify the file:

   ```
   # vi /usr/lib/python2.6/site-packages/heat/engine/resources/server.py
   ```

3. Look for the following line:

   ```
   if metadata is not None or personality is not None:
   ```

4. Modify it according to the following line:

   ```
   if metadata is not None or personality:
   ```

5. After completing this step, reboot the UrbanCode Deploy with Patterns server. This will prevent the error from happening.

Important: At the time of writing this publication, the IBM Cloud Manager version 4.1.0.1 also presents this error in its Heat engine. The same work-around applies, and the final correction is available in Fix Pack 3 (IBM Cloud Manager 4.1.0.3).

Error due to cipher misconfiguration between UCD and UCDP

After correcting the problem described in section “Error due to personality property” on page 653, the following error was seen in `/var/log/heat/engine.log` when trying to Provision an instance:

```
2014-09-29 11:13:28.104 5141 ERROR heat.engine.resource [-] CREATE :
UCDResourceTree "resource_tree" Stack "test_bp7"
[3b290929-25c8-4a32-868b-35c05602f75c]
.
.
.
2014-09-29 11:13:28.104 5141 TRACE heat.engine.resource SSLError: [Errno 1]
_ssl.c:492: error:14077438:SSL routines:SSL23_GET_SERVER_HELLO:tlsv1 alert
internal error
```

This error means that UrbanCode Deploy is rejecting the connection from UrbanCode Deploy with Patterns, which was attempted using TLSv1 security as a cipher. In this case, we have modified the ciphers used by UrbanCode Deploy with Patterns and UrbanCode Deploy, ensuring that both applications use compatible ciphers:

1. Back up the following files in the UrbanCode Deploy with Patterns server:

   ```
   # cp -p /opt/ibm-ucd/server/opt/tomcat/conf/server.xml
   /opt/ibm-ucd/server/opt/tomcat/conf/server.xml.ORIG
   # cp -p /etc/tomcat6/server.xml /etc/tomcat6/server.xml.ORIG
   ```

2. Modify the `/opt/ibm-ucd/server/opt/tomcat/conf/server.xml` file. Look for the `ciphers` entry and modify it, specifying the desired `cipher`, as shown in Example 7-39.

   Example 7-39 Modifying ciphers from UrbanCode Deploy

   ```
   <Connector port="${install.server.web.https.port}"
                address="${install.server.web.ip}"
                server="SERVER"
                maxThreads="150"
                minSpareThreads="25"
                maxSpareThreads="75"
                enableLookups="false"
                acceptCount="100"
                debug="0"
                connectionTimeout="20000"
                disableUploadTimeout="true"
                compression="1024"
                noCompressionUserAgents="gozilla, traviata"

   compressableMimeType="text/html,text/xml,text/javascript,text/css,text/plain,ap
   plication/json"
                algorithm="SunX509"
                SSLEnabled="true"
                scheme="https"
   ```

```
              secure="true"
              clientAuth="false"
              URIEncoding="UTF-8"
               ciphers="SSL_RSA_WITH_RC4_128_SHA"
              sslProtocol="TLSv1.2"
              keystoreFile="conf/tomcat.keystore"
              keystorePass="changeit" />
```

3. Modify the `/etc/tomcat6/server.xml` file, and look for the two existing `ciphers` entries and change them, specifying the desired `cipher`, as shown in Example 7-40.

Example 7-40 Modifying ciphers from UrbanCode Deploy with Patterns

```
<Connector port="8080" protocol="HTTP/1.1"
              connectionTimeout="20000"
              URIEncoding="UTF-8"
              proxyPort=""
              redirectPort="8443" />
              ciphers="SSL_RSA_WITH_RC4_128_SHA"
   <!-- A "Connector" using the shared thread pool-->
   <!--
   <Connector executor="tomcatThreadPool"
              port="8080" protocol="HTTP/1.1"
              connectionTimeout="20000"
              redirectPort="8443" />
   -->
   <!-- Define a SSL HTTP/1.1 Connector on port 8443
        This connector uses the JSSE configuration, when using APR, the
        connector should be using the OpenSSL style configuration
        described in the APR documentation -->
   <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
              proxyPort=""
              keystoreFile="/etc/tomcat6/keystore.jks"
              keystorePass="changeit"
              keystoreType="jks"
              truststorePass="changeit"
              maxThreads="150" scheme="https" secure="true"
              clientAuth="false" sslProtocol="TLS" />
              ciphers="SSL_RSA_WITH_RC4_128_SHA"
```

4. After completing this step, reboot the UrbanCode Deploy with Patterns server. This will ensure the ciphers compatibility between UrbanCode Deploy and UrbanCode Deploy with Patterns and prevent the mentioned error from happening.

In the next section, we describe a BigInsights user scenario.

7.3 IBM BigInsights big data analytics software

In this section, we will describe how to deploy the BigInsights software in the SDE solution environment. This ties back to Chapter 1, Introduction to the IBM Software Defined Environment (SDE). See 1.12.1, “Enable line-of-business cloud users” on page 39.

We will describe how the HOT template is put together, and how the BigInsights software stack is deployed in a SDE environment.

Example 7-41 shows the `BigInsights_autoscale.yaml` HOT template, which is defined to set up the provisioning of the BigInsights software stack and set other parameters, such as autoscaling on demand for the software.

Example 7-41 BigInsights HOT template with autoscaling

```
heat_template_version: 2013-05-23
description:
parameters:
  key_name:
    type: string
    description: Name of an existing key pair to use for the instance
  flavor:
    type: string
    description: Instance type for the instance to be created
    default: m1.large
  image_id:
    type: string
    description: ID of the image to use for the instance
  software_repository:
    type: string
    description: IP address of the software repository where to download the
biginsights installation package
    default: 10.10.0.1
  data_node_min:
    type: string
    description: the minimum number of the hadoop data nodes
    default: 1
  data_node_max:
    type: string
    description: the maximum number of the hadoop data nodes
    default: 100
```

Table 7-1 provides an overview of the BigInsight HOT template with autoscaling, along with a description and objective or purpose for each of the input parameters.

Table 7-1 BigInsight HOT template overview

HOT template input parameter	Purpose or objective (description)	Parameters settings and values examples
key_name	Name of an existing key pair to use to assign to servers for the instance.	Set at server and instance creation. For example: ► testkey ► prodkey
flavor	Type of the instance to be created. For example, this is the relative size of the instance allocated. Also known as *glance* in OpenStack or *image*. See Figure 3-12 on page 82, SDE image (Glance) solution instance flavor listing.	For example: ► `m1.tiny` ► `m1.medium` ► `m1.large`
image_id	Name of the image to use for the instances and servers.	For example: ► `rhel_x86_cloud_init` ► `1f6596de-2708-42b2-B1fc-f324bffbb3cde6`
software_repository	IP address of the software repository where to download BigInsights installation package. Location of the BigInsights package.	For example: `192.168.21.176`
data_node_min	► The minimum number of Hadoop data nodes. ► The lower boundary parameter setting for scaling.	For example: 1
data_node_max	► The maximum number of Hadoop data nodes. ► The upper boundary parameter setting for scaling.	For example: 100

Now that the core HOT template is defined for the BigInsights software, as shown in the example above, the next step is to set the permissions to set up the directory structure and to enable passwordless root access as the BI Admin installer. This can be seen in the highlighted code from the BigInsights autoscale HOT template shown in Example 7-42.

Example 7-42 Set up and configure BI Admin User Group and Hadoop directory structure

```
heat_template_version: 2013-05-23
description:
parameters:
  key_name:
    type: string
    description: Name of an existing key pair to use for the instance
  flavor:
    type: string
    description: Instance type for the instance to be created
    default: m1.large
  image_id:
    type: string
    description: ID of the image to use for the instance
```

```
  software_repository:
    type: string
    description: IP address of the software repository where to download
biginsights installation package
    default: 10.10.0.1
  data_node_min:
    type: string
    description: the minimum number of the hadoop data nodes
    default: 1
  data_node_max:
    type: string
    description: the maximum number of the hadoop data nodes
    default: 100
resources:
  master_node:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name}
      image: { get_param: image_id }
      flavor: { get_param: flavor}
      user_data:
        str_replace:
          template: |
            #!/bin/bash -v
            LOGFILE=/var/tmp/user_data.log
            echo "Start.." > $LOGFILE
            master_node_ip=`ifconfig eth0 |grep "inet addr"| cut -f 2 -d ":"|cut
-f 1 -d " "`
            if [ "$master_node_ip" = "" ]
            then
              master_node_ip=`ifconfig eth0 |grep "inet " | sed 's/^[ \t]*//g' |
cut -d " " -f 2`
            fi
            echo "master_node_ip=$master_node_ip" &>>$LOGFILE

            echo "Enable passwordless access for root" >> $LOGFILE
            mkdir -p ~/.ssh &>/dev/null
            chmod 700 ~/.ssh
            cd ~/.ssh
            rm -rf id_rsa id_rsa.pub &>>$LOGFILE
            curl
http://$software_repository/repos/install_images/biginsights/id_rsa >
~/.ssh/id_rsa
            curl
http://$software_repository/repos/install_images/biginsights/id_rsa.pub
>~/.ssh/id_rsa.pub
            chmod 600 ~/.ssh/id_rsa ~/.ssh/id_rsa.pub
            cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
            chmod 600 ~/.ssh/authorized_keys

            #add biadmin group and user which are required by Biginsights
installer
            groupadd -g 168 biadmin
            useradd -g biadmin -u 168 biadmin
            echo "%biadmin ALL=(ALL) NOPASSWD: ALL" >>/etc/sudoers
```

```
            echo "Enable passwordless access for biadmin" >> $LOGFILE
            mkdir -p /home/biadmin/.ssh &>/dev/null
            chmod 700 /home/biadmin/.ssh
            cd /home/biadmin/.ssh
            rm -rf id_rsa id_rsa.pub &>>$LOGFILE
            curl
http://$software_repository/repos/install_images/biginsights/id_rsa >
/home/biadmin/.ssh/id_rsa
            curl
http://$software_repository/repos/install_images/biginsights/id_rsa.pub >
/home/biadmin/.ssh/id_rsa.pub
            chmod 600 /home/biadmin/.ssh/id_rsa /home/biadmin/.ssh/id_rsa.pub
            cat /home/biadmin/.ssh/id_rsa.pub >>
/home/biadmin/.ssh/authorized_keys
            chmod 600 /home/biadmin/.ssh/authorized_keys
            chown -R biadmin:biadmin /home/biadmin/.ssh

            #disable host key (listed in the /root/.ssh/known_hosts file) checking
            echo "StrictHostKeyChecking no" >>/etc/ssh/ssh_config

            echo "Create disk directories for Hadoop and HDFS" >> $LOGFILE
            for((i=1;i<=10;i++)) do
               mkdir -p /disk$i/hadoop/hdfs
               mkdir -p /disk$i/hadoop/mapred
               chown -R biadmin:biadmin /disk$i
            done

            hostname=`hostname`
            echo "$master_node_ip $hostname" >> /etc/hosts

            yum install -y expect
            yum install -y httpd

            curl
http://$software_repository/repos/install_images/biginsights/BIGINSIGHTS_EE_NP_V2.
1_MP_EIMG.tar.gz  > /var/tmp/BIGINSIGHTS_EE_NP_V2.1_MP_EIMG.tar.gz
            curl
http://$software_repository/repos/install_images/biginsights/fullinstall_single_no
de.xml >/var/tmp/fullinstall.xml

            rm -rf /var/tmp/biginsights-enterprise-linux64__b20130531_1113
&>>$LOGFILE

            tar -xvzf /var/tmp/BIGINSIGHTS_EE_NP_V2.1_MP_EIMG.tar.gz -C /var/tmp

            #Turn off problematic prereq check which doesn't work reliably in VMs

/var/tmp/biginsights-enterprise-linux64_b20130531_1113/installer/bin/enableOrDisab
lePrechecker.sh disable port.connectivity
            #update the configuration file for BigInsights silent installer
            cp /var/tmp/fullinstall.xml /var/tmp/fullinstall_updated.xml
            sed -i "s/controller_ip/$master_node_ip/g"
/var/tmp/fullinstall_updated.xml
            #sed -i "s/computenode_ip//g" /var/tmp/fullinstall_updated.xml
```

```
            #run BigInsights silent installer

/var/tmp/biginsights-enterprise-linux64_b20130531_1113/silent-install/silent-insta
ll.sh /var/tmp/fullinstall_updated.xml &>>$LOGFILE
            echo "Biginsights installed successfully!" >
/root/Biginsights_completed
          params:
            $software_repository: { get_param: software_repository}
  data_node_group:
    type: OS::Heat::AutoScalingGroup
    properties:
      min_size: { get_param: data_node_min }
      max_size: { get_param: data_node_max }
      resource:
        type: OS::Nova::Server
        properties:
          key_name: { get_param: key_name}
          image: { get_param: image_id }
          flavor: { get_param: flavor}
          metadata: {"metering.stack": {get_param: "OS::stack_id"}}
          user_data:
            str_replace:
              template: |
                #!/bin/bash -v
                LOGFILE=/var/tmp/user_data.log
                echo "Start.." > $LOGFILE
                echo "master_node_ip=$master_node_ip" &>>$LOGFILE

                echo "Enable passwordless access for root" >> $LOGFILE
                mkdir -p ~/.ssh &>/dev/null
                chmod 700 ~/.ssh
                cd ~/.ssh
                rm -rf id_rsa id_rsa.pub &>>$LOGFILE
                curl
http://$software_repository/repos/install_images/biginsights/id_rsa >
~/.ssh/id_rsa
                curl
http://$software_repository/repos/install_images/biginsights/id_rsa.pub
>~/.ssh/id_rsa.pub
                chmod 600 ~/.ssh/id_rsa ~/.ssh/id_rsa.pub
                cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
                chmod 600 ~/.ssh/authorized_keys

                #add biadmin group and user which are required by Biginsights
installer
                groupadd -g 168 biadmin
                useradd -g biadmin -u 168 biadmin
                echo "%biadmin ALL=(ALL) NOPASSWD: ALL" >>/etc/sudoers

                echo "Enable passwordless access for biadmin" >> $LOGFILE
                mkdir -p /home/biadmin/.ssh &>/dev/null
                chmod 700 /home/biadmin/.ssh
                cd /home/biadmin/.ssh
                rm -rf id_rsa id_rsa.pub &>>$LOGFILE
```

```
                curl
http://$software_repository/repos/install_images/biginsights/id_rsa >
/home/biadmin/.ssh/id_rsa
                curl
http://$software_repository/repos/install_images/biginsights/id_rsa.pub >
/home/biadmin/.ssh/id_rsa.pub
                chmod 600 /home/biadmin/.ssh/id_rsa /home/biadmin/.ssh/id_rsa.pub
                cat /home/biadmin/.ssh/id_rsa.pub >>
/home/biadmin/.ssh/authorized_keys
                chmod 600 /home/biadmin/.ssh/authorized_keys
                chown -R biadmin:biadmin /home/biadmin/.ssh

                #disable host key (listed in the /root/.ssh/known_hosts file)
checking
                echo "StrictHostKeyChecking no" >>/etc/ssh/ssh_config

                echo "Create disk directories for Hadoop and HDFS" >> $LOGFILE
                for((i=1;i<=10;i++)) do
                  mkdir -p /disk$i/hadoop/hdfs
                  mkdir -p /disk$i/hadoop/mapred
                  chown -R biadmin:biadmin /disk$i
                done

               ip=`ifconfig eth0 |grep "inet addr"| cut -f 2 -d ":"|cut -f 1 -d "
"`
                if [ "$ip" = "" ]
                then
                  ip=`ifconfig eth0 |grep "inet " | sed 's/^[ \t]*//g' | cut -d "
" -f 2`
                fi
                hostname=`hostname`
                echo "$ip $hostname" >> /etc/hosts
                echo "$hostname $ip" >> $LOGFILE
```

Next, the two default data nodes (Master and Slave nodes) will be set for Hadoop, as shown in Example 7-43.

Example 7-43 Configure Hadoop data nodes

```
heat_template_version: 2013-05-23
description:
parameters:
  key_name:
    type: string
    description: Name of keypair to assign to servers
    default: testkey
  flavor:
    type: string
    description: Instance type for the instance to be created
    default: m1.large
  image_id:
    type: string
    description: Name of image to use for servers
    default: rhel_x86_cloud_init
```

```
  software_repository:
    type: string
    description: IP address of the software repository where to download
biginsights installation package
    default: 192.168.21.176
  private_net_id:
    type: string
    default: d828cdb1-e926-4f94-b276-29a41471e5b6

resources:
  vm1:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name}
      image: { get_param: image_id }
      flavor: { get_param: flavor}
      networks:
        - network: { get_param: private_net_id }
      config_drive: "true"
      user_data:
        str_replace:
          template: |
            #!/bin/bash -v
            export LOGFILE=/var/tmp/user_data.log
            echo "Start..." >> $LOGFILE
            echo "Ajusting disk size and filesystem size" >> $LOGFILE
            (echo n; echo p; echo 3; echo 20807; echo; echo w) | fdisk /dev/vda >>
$LOGFILE 2>&1
            (echo t; echo 3; echo 8e; echo; echo w) | fdisk /dev/vda >> $LOGFILE
2>&1
            partprobe >> $LOGFILE 2>&1
            partx -a /dev/vda >> $LOGFILE 2>&1
            pvcreate /dev/vda3 >> $LOGFILE 2>&1
            vgextend vg_tempvm /dev/vda3 >> $LOGFILE 2>&1
            lvextend /dev/vg_tempvm/lv_root /dev/vda3 >> $LOGFILE 2>&1
            resize2fs /dev/vg_tempvm/lv_root >> $LOGFILE 2>&1

            echo "Fix SSH configuration for DNS" >> $LOGFILE 2>&1
            sed -i "/AcceptEnv XMODIFIERS/a UseDNS no" /etc/ssh/sshd_config >>
$LOGFILE 2>&1
            service sshd restart >> $LOGFILE 2>&1

            echo "Enable passwordless access for root" >> $LOGFILE
            mkdir -p ~/.ssh >> $LOGFILE 2>&1
            chmod 700 ~/.ssh >> $LOGFILE 2>&1
            cd ~/.ssh >> $LOGFILE 2>&1
            rm -rf id_rsa id_rsa.pub >> $LOGFILE 2>&1
            wget ftp://$software_repository/id_rsa -O ~/.ssh/id_rsa >> $LOGFILE
2>&1
            wget ftp://$software_repository/id_rsa.pub -O ~/.ssh/id_rsa.pub >>
$LOGFILE 2>&1
            chmod 600 ~/.ssh/id_rsa ~/.ssh/id_rsa.pub >> $LOGFILE 2>&1
            cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
            chmod 600 ~/.ssh/authorized_keys >> $LOGFILE 2>&1
```

```
            echo "Fix Yum repositories" >> $LOGFILE 2>&1
            rm -f /etc/yum.repos.d/epel.repo >> $LOGFILE 2>&1
            rm -f /etc/yum.repos.d/epel-testing.repo >> $LOGFILE 2>&1
            yum clean all >> $LOGFILE 2>&1

            echo "add biadmin group and user which are required by Biginsights
installer" >> $LOGFILE 2>&1
            #add biadmin group and user which are required by Biginsights
installer
            groupadd -g 168 biadmin >> $LOGFILE 2>&1
            useradd -g biadmin -u 168 biadmin >> $LOGFILE 2>&1
            echo "%biadmin ALL=(ALL) NOPASSWD: ALL" >>/etc/sudoers
            sed -i "s/Defaults    requiretty/#Defaults    requiretty/g"
/etc/sudoers >> $LOGFILE 2>&1

            echo "Enable passwordless access for biadmin" >> $LOGFILE
            mkdir -p /home/biadmin/.ssh >> $LOGFILE 2>&1
            chmod 700 /home/biadmin/.ssh >> $LOGFILE 2>&1
            cd /home/biadmin/.ssh >> $LOGFILE 2>&1
            rm -rf id_rsa id_rsa.pub >> $LOGFILE 2>&1
           wget ftp://$software_repository/id_rsa -O /home/biadmin/.ssh/id_rsa >>
$LOGFILE 2>&1
            wget ftp://$software_repository/id_rsa.pub -O
/home/biadmin/.ssh/id_rsa.pub >> $LOGFILE 2>&1
            chmod 600 /home/biadmin/.ssh/id_rsa /home/biadmin/.ssh/id_rsa.pub >>
$LOGFILE 2>&1
            cat /home/biadmin/.ssh/id_rsa.pub >>
/home/biadmin/.ssh/authorized_keys
            chmod 600 /home/biadmin/.ssh/authorized_keys >> $LOGFILE 2>&1
            chown -R biadmin:biadmin /home/biadmin/.ssh >> $LOGFILE 2>&1

            echo "Create disk direcotries for hadoop and HDFS" >> $LOGFILE
            for((i=1;i<=10;i++)) do
               mkdir -p /disk$i/hadoop/hdfs
               mkdir -p /disk$i/hadoop/mapred
               chown -R biadmin:biadmin /disk$i
            done

            ip=`ifconfig eth0 |grep "inet addr"| cut -f 2 -d ":"|cut -f 1 -d " "`
            if [ "$ip" = "" ]
            then
              ip=`ifconfig eth0 |grep "inet " | sed 's/^[ \t]*//g' | cut -d " " -f
2`
            fi
            hostname=`hostname`
            echo "$ip $hostname" >> /etc/hosts
            echo "$hostname $ip" >> $LOGFILE

          params:
            $software_repository: { get_param: software_repository}
  vm2:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name}
      image: { get_param: image_id }
```

```
      flavor: { get_param: flavor}
      networks:
        - network: { get_param: private_net_id }
      config_drive: "true"
      user_data:
        str_replace:
          template: |
            #!/bin/bash -v
            export LOGFILE=/var/tmp/user_data.log
            echo "Start.." > $LOGFILE
            echo "Ajusting disk size and filesystem size" >> $LOGFILE
            (echo n; echo p; echo 3; echo 20807; echo; echo w) | fdisk /dev/vda >>
$LOGFILE 2>&1
            (echo t; echo 3; echo 8e; echo; echo w) | fdisk /dev/vda >> $LOGFILE
2>&1
            partprobe >> $LOGFILE 2>&1
            partx -a /dev/vda >> $LOGFILE 2>&1
            pvcreate /dev/vda3 >> $LOGFILE 2>&1
            vgextend vg_tempvm /dev/vda3 >> $LOGFILE 2>&1
            lvextend /dev/vg_tempvm/lv_root /dev/vda3 >> $LOGFILE 2>&1
            resize2fs /dev/vg_tempvm/lv_root >> $LOGFILE 2>&1

            echo "Fix yum repositories" >> $LOGFILE 2>&1
            rm -f /etc/yum.repos.d/epel.repo >> $LOGFILE 2>&1
            rm -f /etc/yum.repos.d/epel-testing.repo >> $LOGFILE 2>&1
            yum clean all >> $LOGFILE 2>&1

            echo "Fix SSH configuration for DNS" >> $LOGFILE 2>&1
            sed -i "/AcceptEnv XMODIFIERS/a UseDNS no" /etc/ssh/sshd_config >>
$LOGFILE 2>&1
            service sshd restart >> $LOGFILE 2>&1

            master_node_ip=`ifconfig eth0 |grep "inet addr"| cut -f 2 -d ":"|cut
-f 1 -d " "`
            if [ "$master_node_ip" = "" ]
            then
              master_node_ip=`ifconfig eth0 |grep "inet " | sed 's/^[ \t]*//g' |
cut -d " " -f 2`
            fi
            echo "master_node_ip=$master_node_ip" &>>$LOGFILE
            echo "compute_node_ip=$slave_node_ip" &>>$LOGFILE

            echo "Enable passwordless access for root" >> $LOGFILE
            mkdir -p ~/.ssh >> $LOGFILE 2>&1
            chmod 700 ~/.ssh >> $LOGFILE 2>&1
            cd ~/.ssh >> $LOGFILE 2>&1
            rm -rf id_rsa id_rsa.pub >> $LOGFILE 2>&1
            wget ftp://$software_repository/id_rsa -O ~/.ssh/id_rsa >> $LOGFILE
2>&1
            wget ftp://$software_repository/id_rsa.pub -O ~/.ssh/id_rsa.pub >>
$LOGFILE 2>&1
            chmod 600 ~/.ssh/id_rsa ~/.ssh/id_rsa.pub >> $LOGFILE 2>&1
            cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
            chmod 600 ~/.ssh/authorized_keys >> $LOGFILE 2>&1
```

```
            echo "Adding biadmin user and group" >> $LOGFILE 2>&1
            #add biadmin group and user which are required by Biginsights
installer
            groupadd -g 168 biadmin >> $LOGFILE 2>&1
            useradd -g biadmin -u 168 biadmin >> $LOGFILE 2>&1
            echo "%biadmin ALL=(ALL) NOPASSWD: ALL" >>/etc/sudoers
            sed -i "s/Defaults    requiretty/#Defaults    requiretty/g"
/etc/sudoers >> $LOGFILE 2>&1

            echo "Enable passwordless access for biadmin" >> $LOGFILE
            mkdir -p /home/biadmin/.ssh >> $LOGFILE 2>&1
            chmod 700 /home/biadmin/.ssh >> $LOGFILE 2>&1
            cd /home/biadmin/.ssh >> $LOGFILE 2>&1
            rm -rf id_rsa id_rsa.pub >> $LOGFILE 2>&1
           wget ftp://$software_repository/id_rsa -O /home/biadmin/.ssh/id_rsa >>
$LOGFILE 2>&1
            wget ftp://$software_repository/id_rsa.pub -O
/home/biadmin/.ssh/id_rsa.pub >> $LOGFILE 2>&1
            chmod 600 /home/biadmin/.ssh/id_rsa /home/biadmin/.ssh/id_rsa.pub >>
$LOGFILE 2>&1
            cat /home/biadmin/.ssh/id_rsa.pub >>
/home/biadmin/.ssh/authorized_keys
            chmod 600 /home/biadmin/.ssh/authorized_keys >> $LOGFILE 2>&1
            chown -R biadmin:biadmin /home/biadmin/.ssh >> $LOGFILE 2>&1

            echo "Create disk directories for Hadoop and HDFS" >> $LOGFILE
            for((i=1;i<=10;i++)) do
               mkdir -p /disk$i/hadoop/hdfs
               mkdir -p /disk$i/hadoop/mapred
               chown -R biadmin:biadmin /disk$i
            done

            hostname=`hostname`
            echo "$master_node_ip $hostname" >> /etc/hosts
            echo "$master_node_ip $hostname" >> $LOGFILE 2>&1

            echo "Installing additional packages" >> $LOGFILE 2>&1
            yum install -y expect >> $LOGFILE 2>&1
            yum install -y httpd >> $LOGFILE 2>&1

            echo "Downloading software" >> $LOGFILE 2>&1
            wget ftp://$software_repository/BIGINSIGHTS_EE_NP_V2.1_MP_EIMG.tar.gz
-O /var/tmp/BIGINSIGHTS_EE_NP_V2.1_MP_EIMG.tar.gz  >> $LOGFILE 2>&1
            wget ftp://$software_repository/fullinstall.xml -O
/var/tmp/fullinstall.xml  >> $LOGFILE 2>&1

            rm -rf /var/tmp/biginsights-enterprise-linux64__b20130531_1113
&>>$LOGFILE

            tar -xvzf /var/tmp/BIGINSIGHTS_EE_NP_V2.1_MP_EIMG.tar.gz -C /var/tmp
>> $LOGFILE 2>&1

           echo "Turn off problematic prereq check which doesn't work reliably in
VMs" >> $LOGFILE 2>&1
            #Turn off problematic prereq check which doesn't work reliably in VMs
```

```
/var/tmp/biginsights-enterprise-linux64_b20130531_1113/installer/bin/enableOrDisab
lePrechecker.sh disable port.connectivity >> $LOGFILE 2>&1

            echo "update the configuration file for BigInsights silent installer"
>> $LOGFILE 2>&1
            #update the configuration file for BigInsights silent installer
            cp /var/tmp/fullinstall.xml /var/tmp/fullinstall_updated.xml >>
$LOGFILE 2>&1
            sed -i "s/controller_ip/$master_node_ip/g"
/var/tmp/fullinstall_updated.xml >> $LOGFILE 2>&1
            sed -i "s/computenode_ip/$slave_node_ip/g"
/var/tmp/fullinstall_updated.xml >> $LOGFILE 2>&1

            echo "run BigInsights silent installer" >> $LOGFILE 2>&1
            #run BigInsights silent installer

/var/tmp/biginsights-enterprise-linux64_b20130531_1113/silent-install/silent-insta
ll.sh /var/tmp/fullinstall_updated.xml &>>$LOGFILE
          params:
            $slave_node_ip: { get_attr: [vm1, networks, Private, 0] }
            $software_repository: { get_param: software_repository }
```

Now that BigInsights is installed successfully, based on the HOT template, an autoscaling group will be defined to provide policies for the nodes to allow them to scale dynamically on demand for additional resource use, driven by Hadoop (see Example 7-44). In this example, a ceilometer alarm is also configured so that if that if the average CPU increases > 50% for one minute, it will scale up for more capacity. If that the average CPU < 15% for one minute, scale-down will occur.

Example 7-44 Autoscaling group definition and dynamic nodes

```
data_node_group:
    type: OS::Heat::AutoScalingGroup
    properties:
      min_size: { get_param: data_node_min }
      max_size: { get_param: data_node_max }
      resource:
        type: OS::Nova::Server
        properties:
          key_name: { get_param: key_name}
          image: { get_param: image_id }
          flavor: { get_param: flavor}
          user_data:
            str_replace:
              template: |
                #!/bin/bash -v
                LOGFILE=/var/tmp/user_data.log
                echo "Start.." > $LOGFILE
                echo "master_node_ip=$master_node_ip" &>>$LOGFILE

                echo "Enable passwordless access for root" >> $LOGFILE
                mkdir -p ~/.ssh &>/dev/null
                chmod 700 ~/.ssh
                cd ~/.ssh
```

```
                rm -rf id_rsa id_rsa.pub &>>$LOGFILE
                curl
http://$software_repository/repos/install_images/biginsights/id_rsa >
~/.ssh/id_rsa
                curl
http://$software_repository/repos/install_images/biginsights/id_rsa.pub
>~/.ssh/id_rsa.pub
                chmod 600 ~/.ssh/id_rsa ~/.ssh/id_rsa.pub
                cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
                chmod 600 ~/.ssh/authorized_keys

                #add biadmin group and user which are required by Biginsights
installer
                groupadd -g 168 biadmin
                useradd -g biadmin -u 168 biadmin
                echo "%biadmin ALL=(ALL) NOPASSWD: ALL" >>/etc/sudoers

                echo "Enable passwordless access for biadmin" >> $LOGFILE
                mkdir -p /home/biadmin/.ssh &>/dev/null
                chmod 700 /home/biadmin/.ssh
                cd /home/biadmin/.ssh
                rm -rf id_rsa id_rsa.pub &>>$LOGFILE
                curl
http://$software_repository/repos/install_images/biginsights/id_rsa >
/home/biadmin/.ssh/id_rsa
                curl
http://$software_repository/repos/install_images/biginsights/id_rsa.pub >
/home/biadmin/.ssh/id_rsa.pub
                chmod 600 /home/biadmin/.ssh/id_rsa /home/biadmin/.ssh/id_rsa.pub
                cat /home/biadmin/.ssh/id_rsa.pub >>
/home/biadmin/.ssh/authorized_keys
                chmod 600 /home/biadmin/.ssh/authorized_keys
                chown -R biadmin:biadmin /home/biadmin/.ssh
datanode_scaleup_policy:
    type: OS::Heat::ScalingPolicy
    properties:
      adjustment_type: change_in_capacity
      auto_scaling_group_id: {get_resource: data_node_group}
      cooldown: 60
      scaling_adjustment: 1
  datanode_scaledown_policy:
    type: OS::Heat::ScalingPolicy
    properties:
      adjustment_type: change_in_capacity
      auto_scaling_group_id: {get_resource: data_node_group}
      cooldown: 60
      scaling_adjustment: -1
  datanode_cpu_alarm_high:
    type: OS::Ceilometer::Alarm
    properties:
      description: Scale-up if the average CPU > 50% for 1 minute
      meter_name: cpu_util
      statistic: avg
      period: 60
      evaluation_periods: 1
```

```
      threshold: 70
      alarm_actions:
        - {get_attr: [datanode_scaleup_policy, alarm_url]}
      matching_metadata: {'metadata.user_metadata.stack': {get_param:
"OS::stack_id"}}
      comparison_operator: gt
  datanode_cpu_alarm_low:
    type: OS::Ceilometer::Alarm
    properties:
      description: Scale-down if the average CPU < 15% for 1 minute
      meter_name: cpu_util
      statistic: avg
      period: 60
      evaluation_periods: 1
      threshold: 5
      alarm_actions:
        - {get_attr: [datanode_scaledown_policy, alarm_url]}
      matching_metadata: {'metadata.user_metadata.stack': {get_param:
"OS::stack_id"}}
      comparison_operator: lt
```

In the next section, we examine a cloud user scenario from various roles or personas and explain and illustrate autoscaling in action.

7.4 Acme Air scenario

In this section, we continue with the cloud user scenario for the airline company example, Acme Air. We described the representative SDE reference solution architecture for this scenario in Chapter 3, “SDE reference architecture” on page 57. Also see 1.12.1, “Enable line-of-business cloud users” on page 39.

The following key benefits of this solution are illustrated and demonstrated in this section:

- Simple and consistent application deployment using patterns-based tools
- Faster delivery of features resulting from continuous application development
- Flexible access through a mobile device to SOE and browser to SOR
- Access to same data and enterprise services through hybrid cloud delivery model
- Improved resiliency and response times due to infrastructure scale out and optimization
- Secure interconnect between off-premises and on-premises
- Allows the enterprise to maintain and use system of record applications and data, with full confidence in the security and compliance of their on-premises systems
- Access to a suite of tools and products to manage infrastructure and applications services

Figure 7-88 illustrates the four cloud users in the SDE environment:

- Michael, the infrastructure admin, needs to manage the off-premises and on-premises cloud resources.
- Maureen, the DevOps pro, needs to provision applications
- Joe and Sally are the typical cloud users who interact with applications in different ways, as seen between and with the on-premises systems of record versus the off-premises systems of engagement.

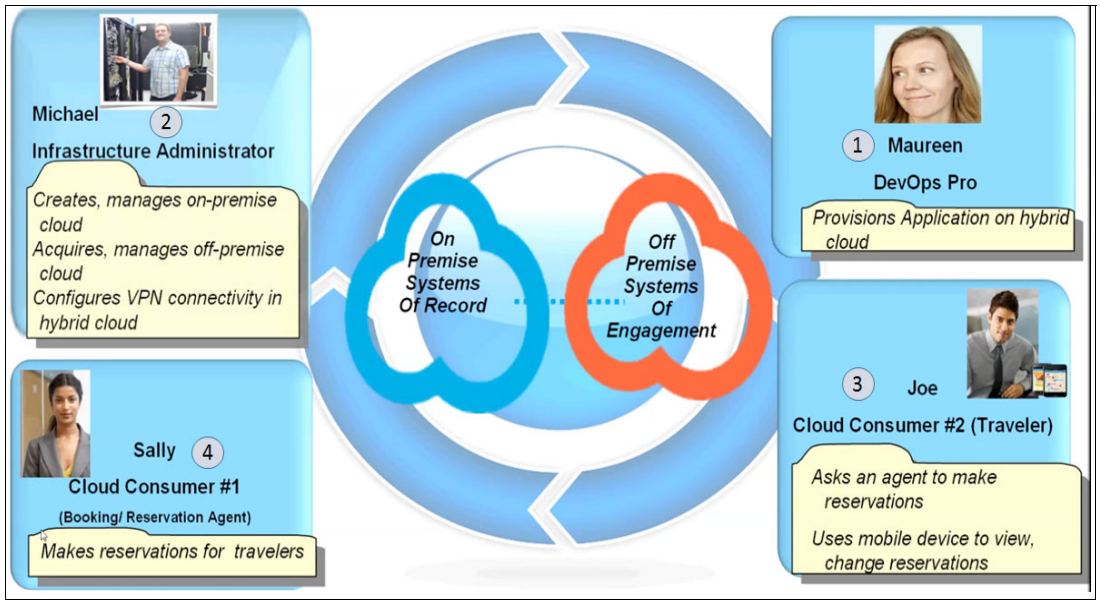


Figure 7-88 Hybrid cloud user, Acme Air Commerce scenario

Let's look at each of the roles and step through their actions in working with the SDE-based hybrid cloud environment.

7.4.1 DevOps Pro (Maureen) provisions an application to the hybrid cloud

We will now demonstrate how easy it is for a Maureen, the DevOps pro, to build application patterns and deploy them to a hybrid cloud by using IBM UrbanCode Deploy with Patterns. Maureen will also connect the application instances across the hybrid cloud using a REST adapter.

1. As seen in Figure 7-89 on page 670, Maureen logs in to the IBM UrbanCode Deploy with Patterns console and inspects the blueprint for her application. Remember that the blueprint template she is inspecting and using (`AcmeAir_Deploy_to_Power`) is engineered to fit her specific workload type and application in the IBM Power environment.

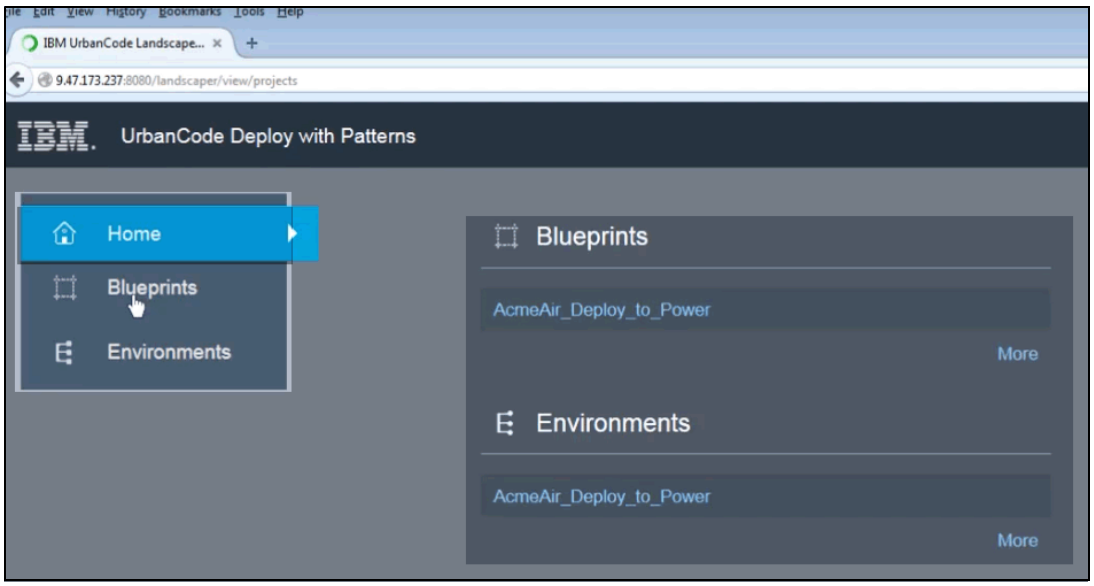


Figure 7-89 Inspect the Acme Air application blueprint for deployment

2. As seen in Figure 7-90 and Figure 7-91 on page 671, Maureen can inspect each software configuration element of the application and define further autoscaling policies for it in the blueprint.

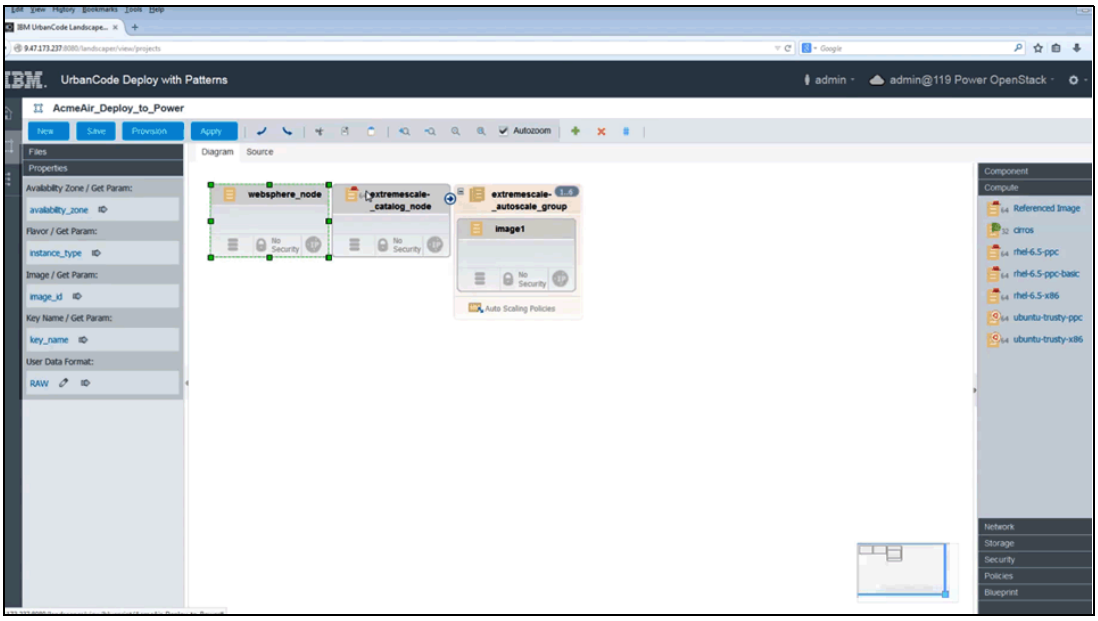


Figure 7-90 Inspect software configuration

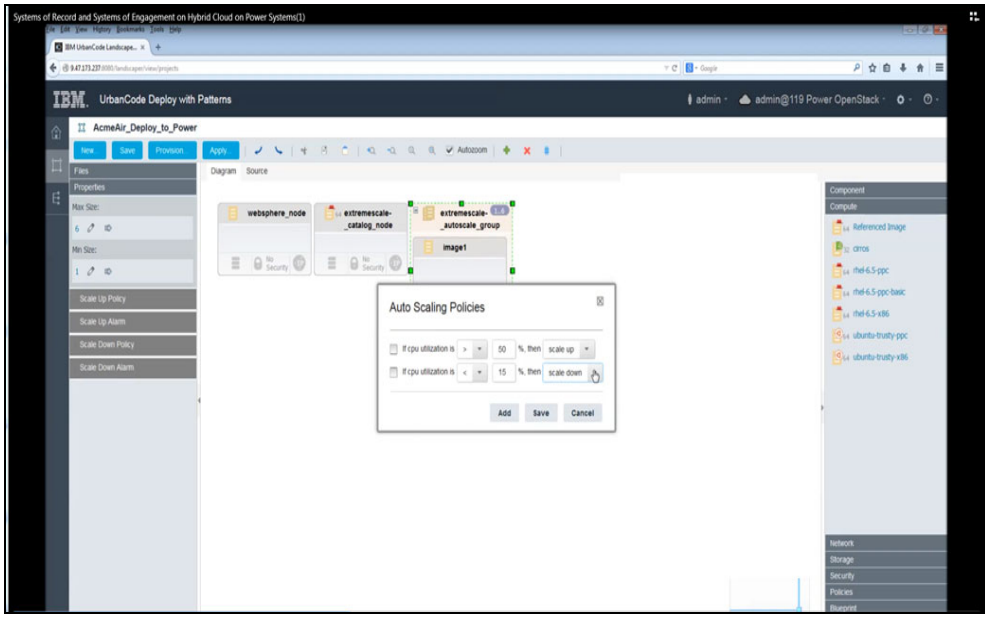


Figure 7-91 Set autoscaling policies

Note: In the previous section (7.3, “IBM BigInsights big data analytics software” on page 655) an autoscaling policy was defined in the HOT template. Compare this to setting the autoscaling policy here with the UCDP GUI. The code in the underlying template will be similar to the BigInsights HOT template, with the differences based on application specifics. A copy of the `AcmeAirDataScale-PowerKVM.yaml` file (HOT template) is provided at the end of this section for reference and use.

Maureen can also switch her view to a text-based definition of the application blueprint, which is based on the OpenStack Heat Orchestration template (HOT) format, as shown in Figure 7-92 on page 672.

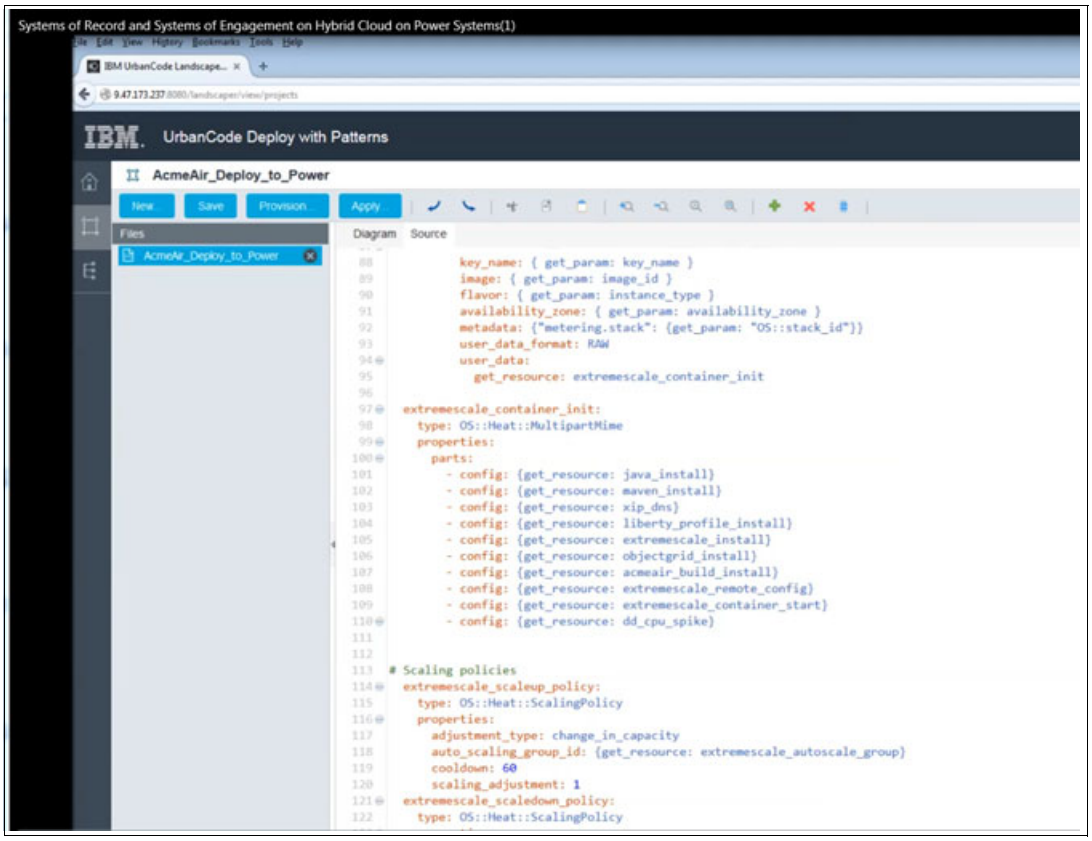


Figure 7-92 Inspect template via text option

3. After finalizing the application blueprint, Maureen is now ready to deploy it to the hybrid cloud, as shown in Figure 7-93.

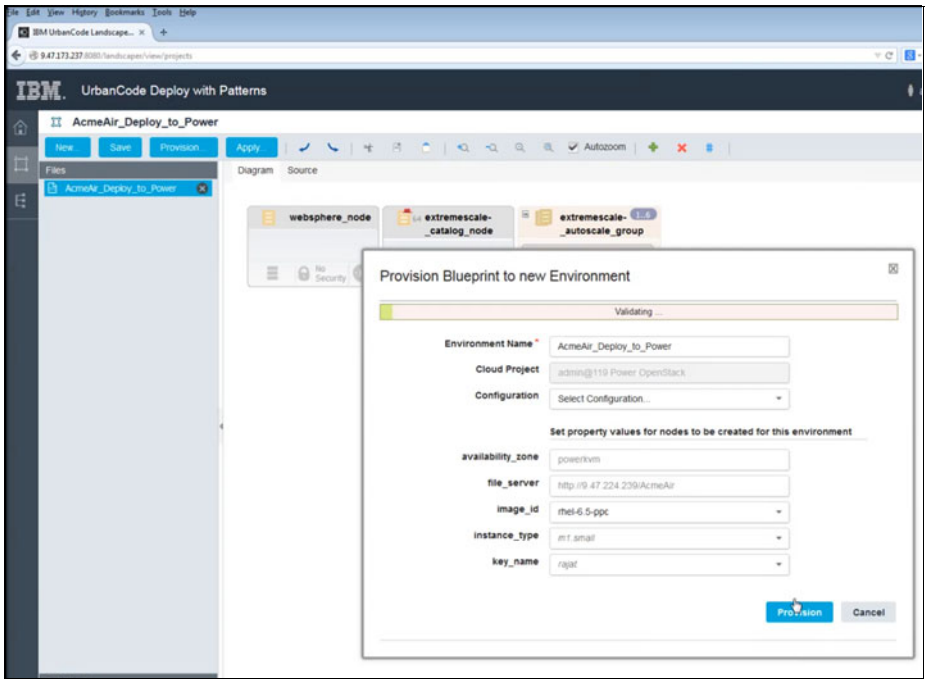


Figure 7-93 Deploy application to the hybrid cloud

4. After the deployment is complete, Maureen can check its status in the Environment section, as shown in Figure 7-94.

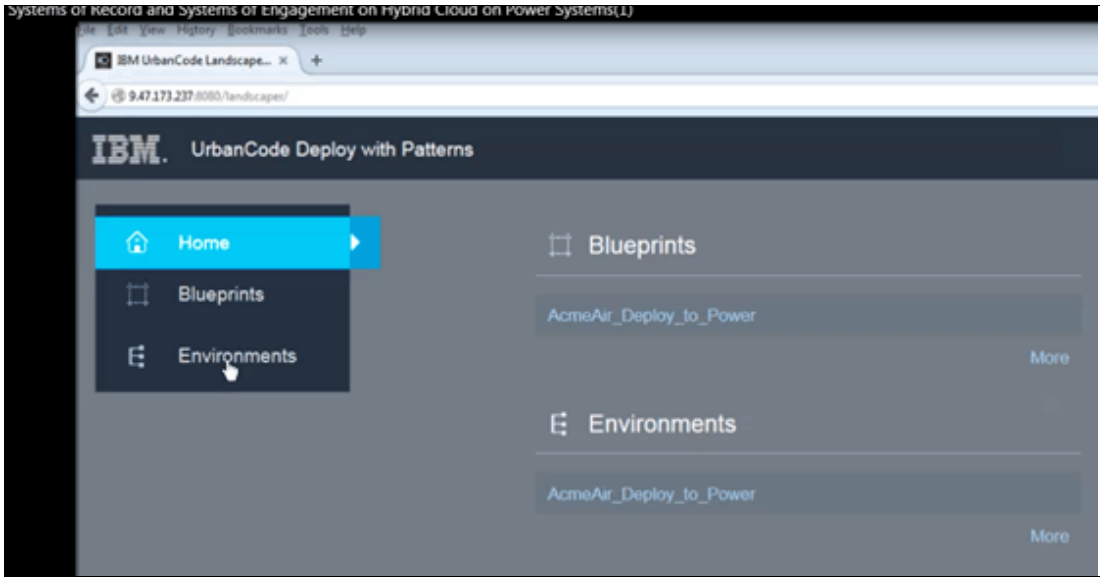


Figure 7-94 Check the environment after application deployment

Maureen can see that the required instances of each software configuration in her blueprint are present in the environment, as shown in Figure 7-95.

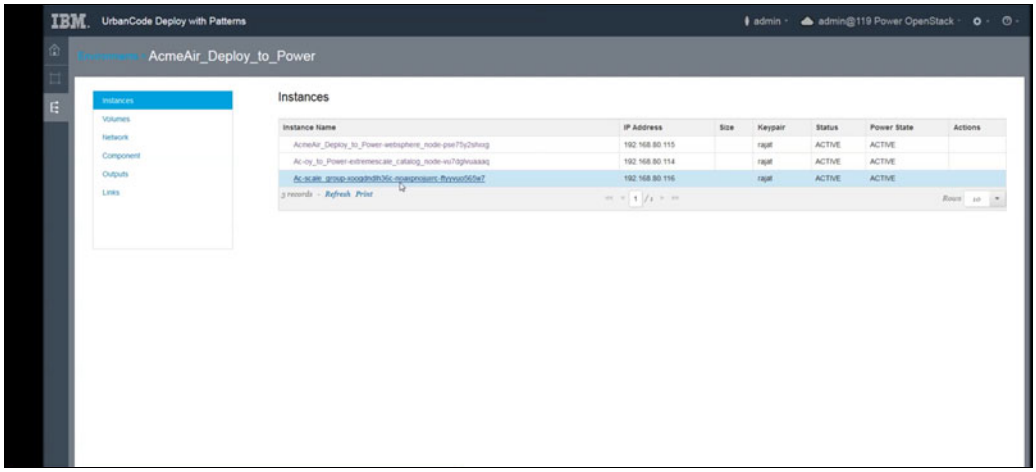


Figure 7-95 Validate instances for software configuration

5. Now, Maureen will connect the application instances in the hybrid cloud by using REST API adapters, as illustrated in Figure 7-96.

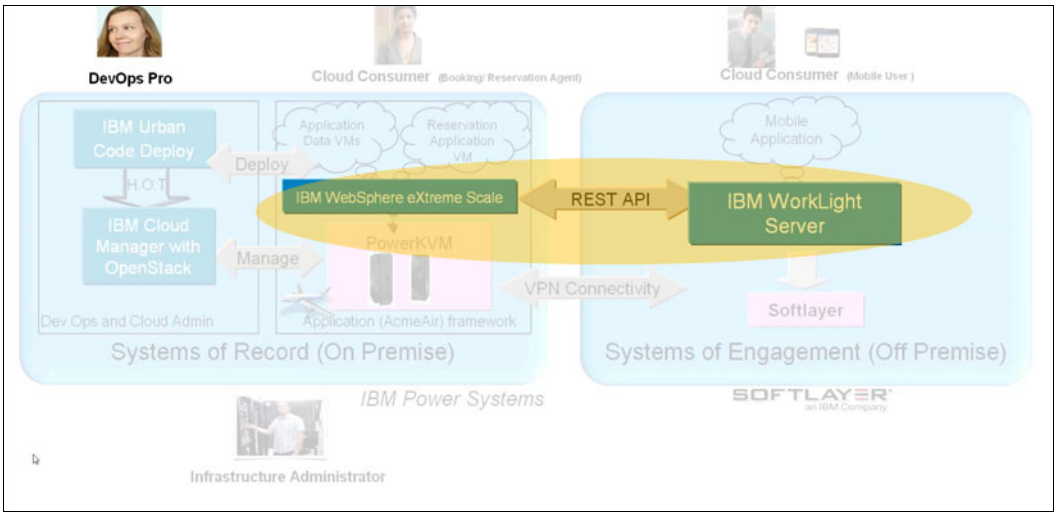


Figure 7-96 Connect application instances using REST API adapters

6. Next, Maureen edits the XML files to connect port 9080 on the on-premises application end with port 3080 on the off-premises application end, as shown in Figure 7-97.

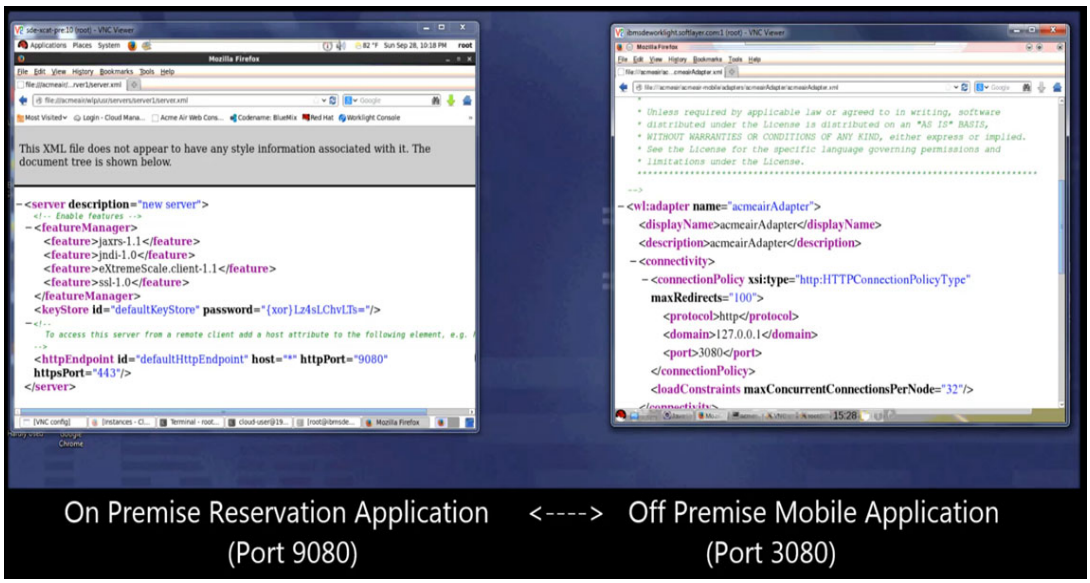


Figure 7-97 Connect application instances

7.4.2 Infrastructure Administrator (Michael) inspects and configures the hybrid cloud

As Figure 7-98 on page 675 shows, Michael, the Infrastructure Administrator, uses IBM Cloud Manager to monitor and configure the hybrid cloud.

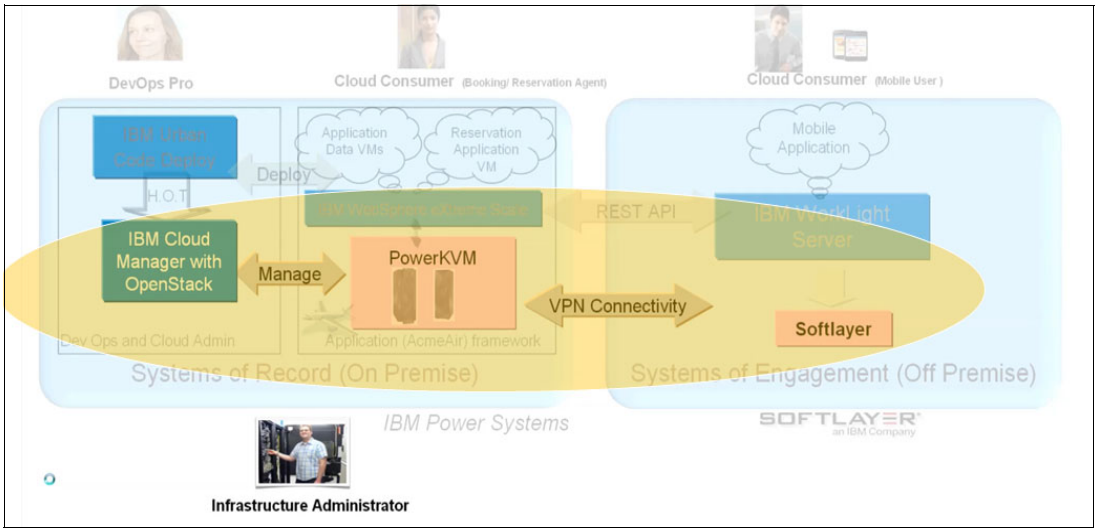


Figure 7-98 Michael monitors the hybrid cloud

1. Michael uses the console for IBM Cloud Manager with OpenStack, as shown in Figure 7-99.

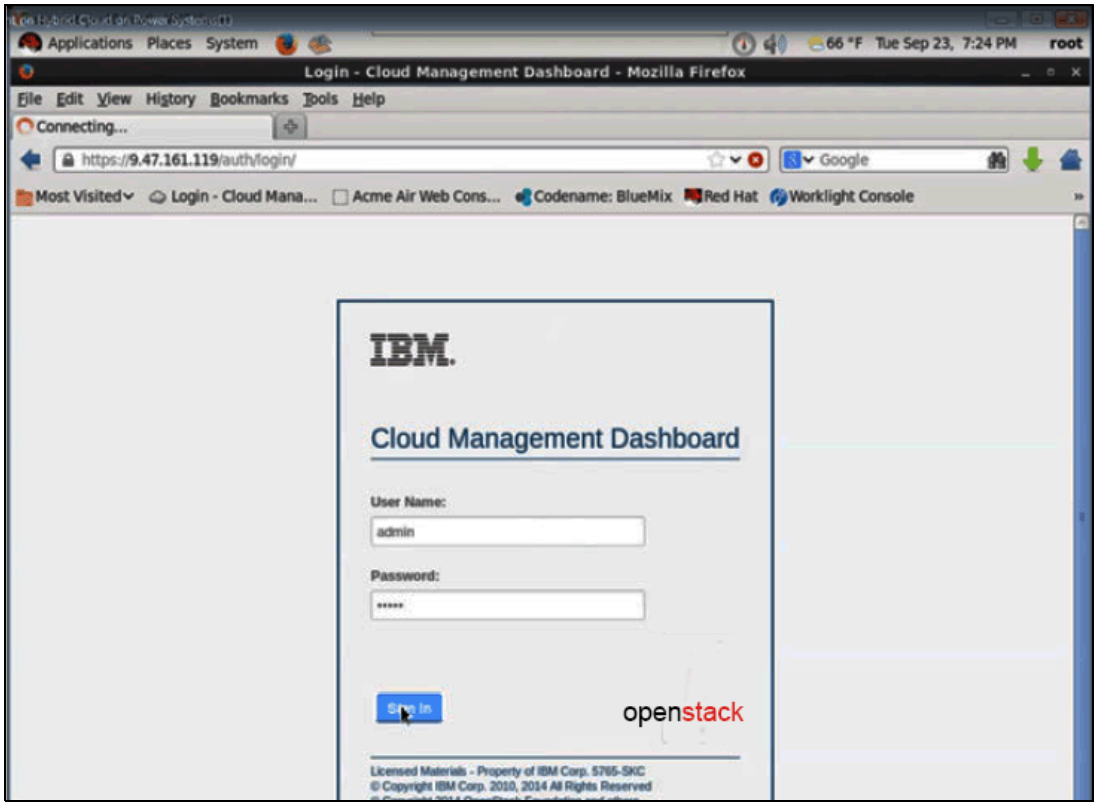


Figure 7-99 Cloud Management dashboard

2. Now Michael can view the Power nodes and also inspect the application stack, as shown in Figure 7-100.

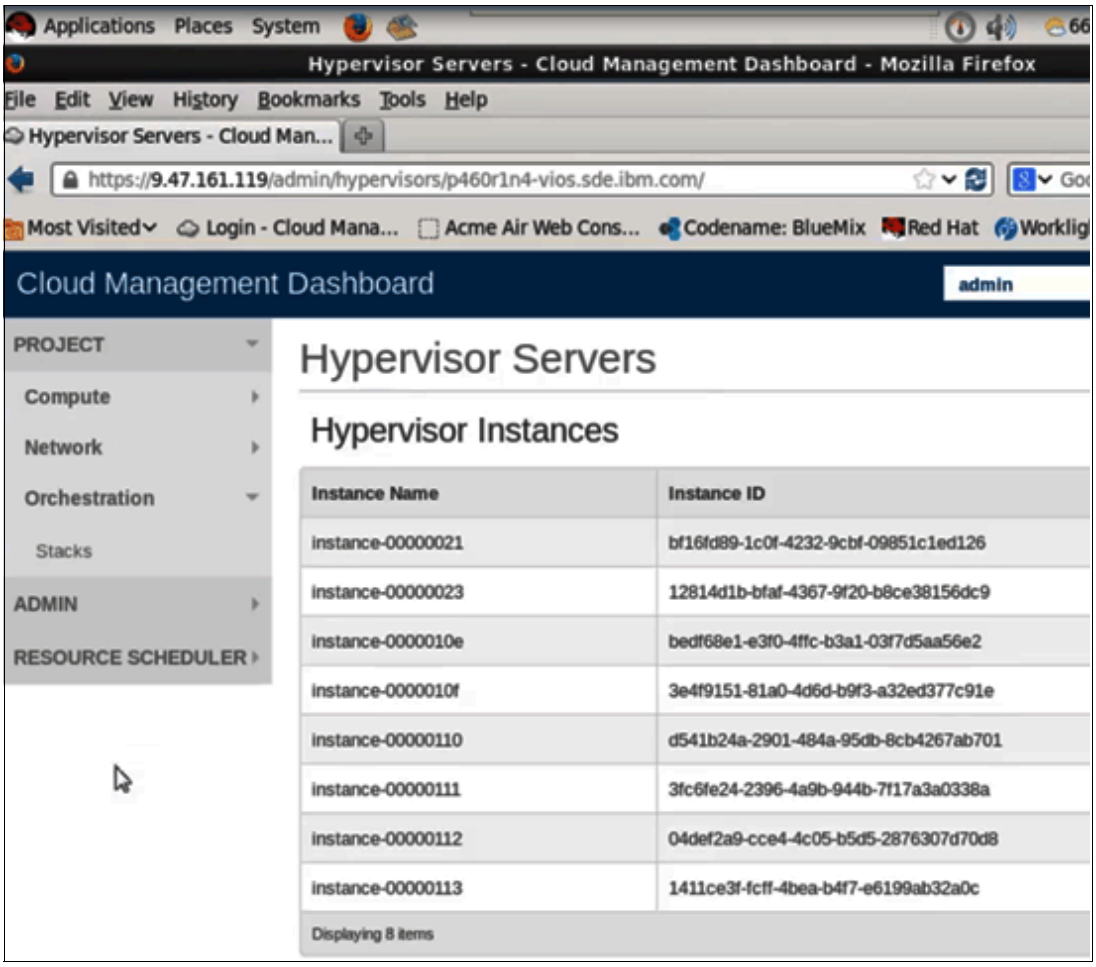


Figure 7-100 View Power nodes and inspect application stack

The topology, detailed resources, related events, and so on for the entire application stack can be inspected, as shown in Figure 7-101 on page 677 and Figure 7-102 on page 678.

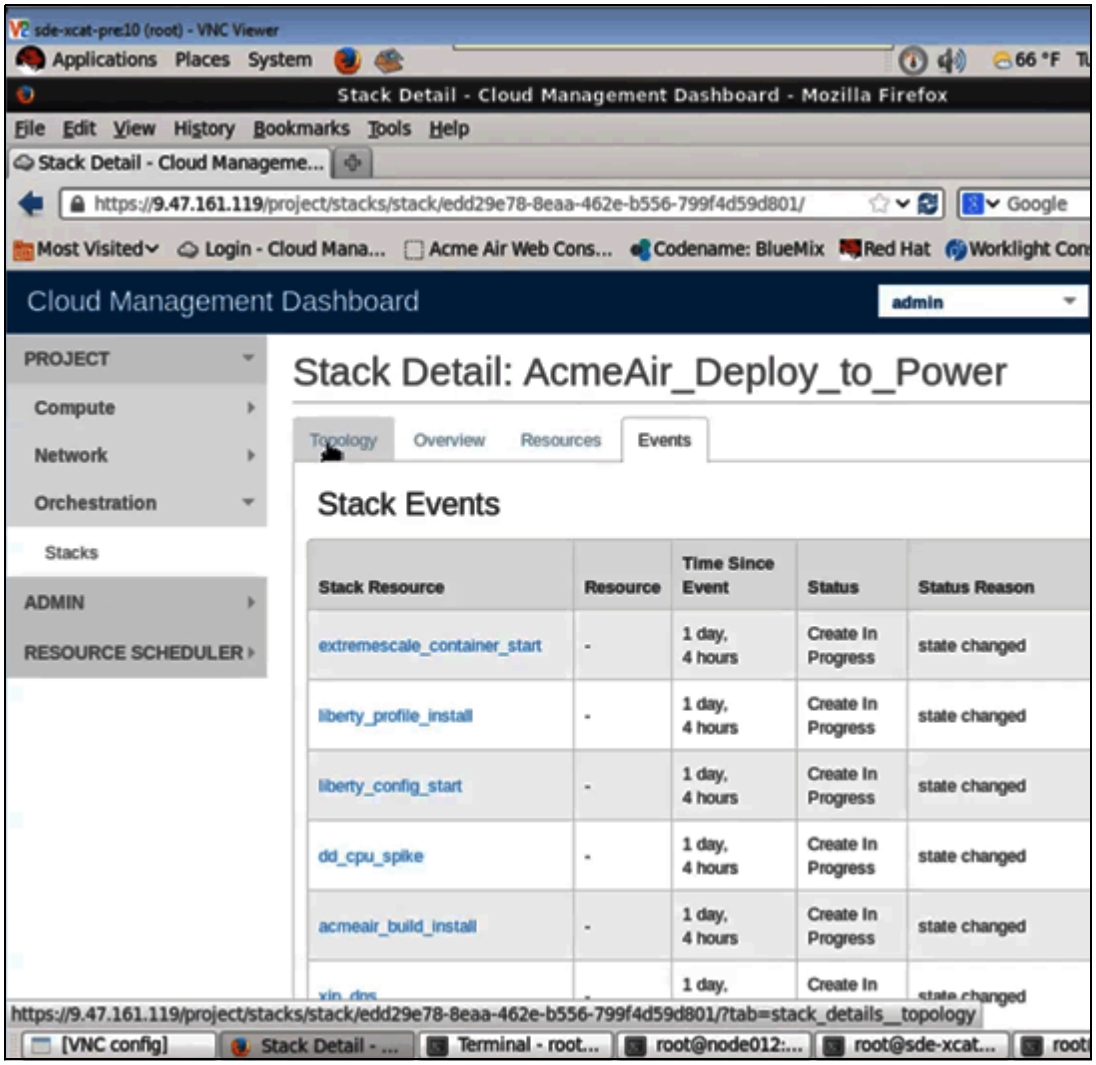


Figure 7-101 Inspect topology stack events

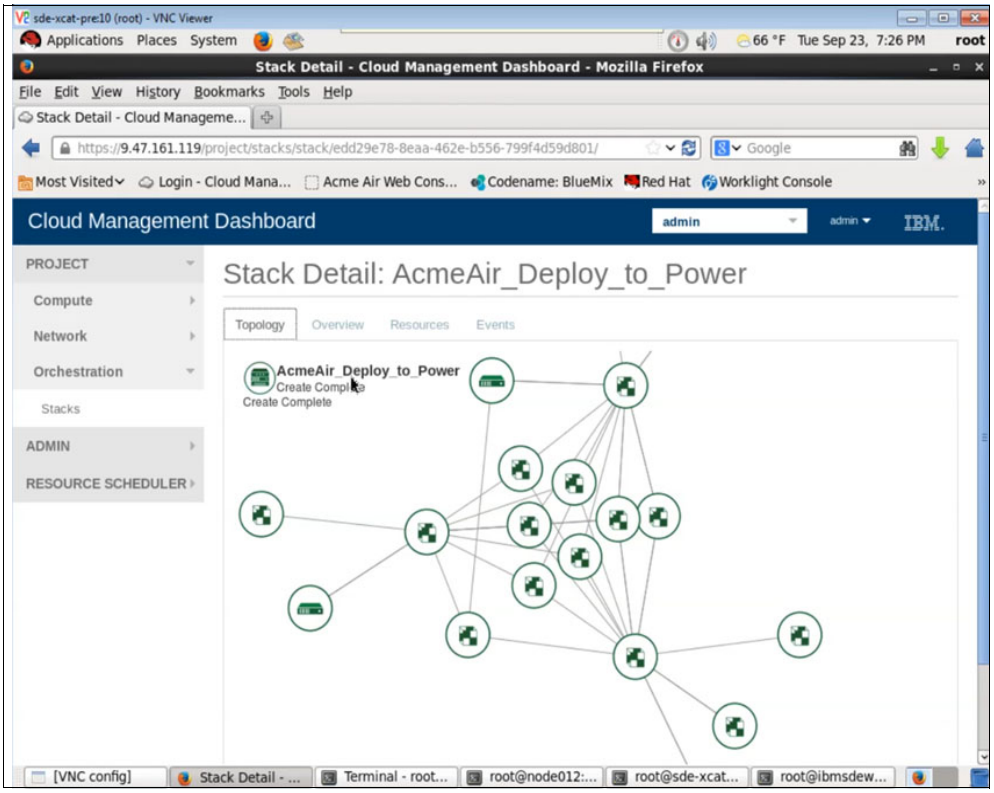


Figure 7-102 Inspect topology

3. The compute, storage, and network elements of the cloud can also be inspected and configured from the console, as shown in Figure 7-103.

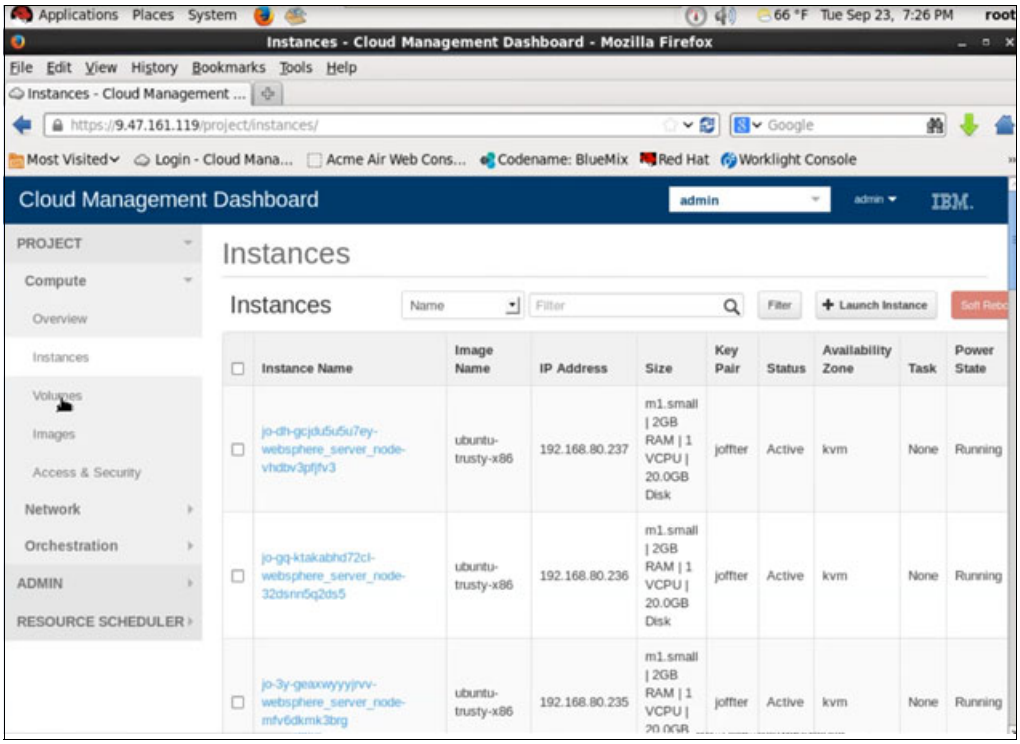


Figure 7-103 Inspect and configure resources

4. Now Michael provides secure network connectivity in the hybrid cloud, as shown in Figure 7-104.

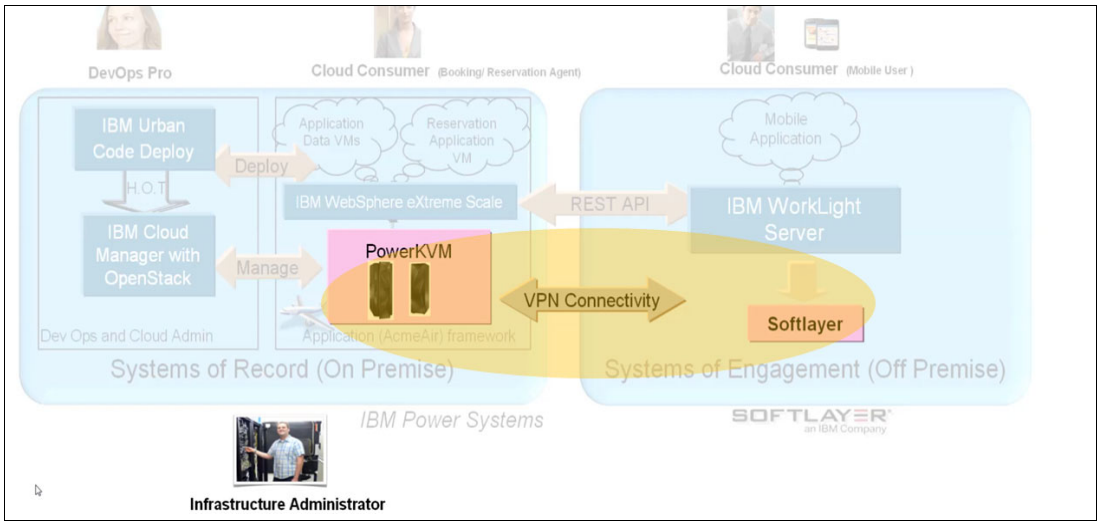


Figure 7-104 Provide secure network connectivity in the hybrid cloud

5. Using the command line, Michael creates an SSH tunnel between the on-premises (port 9080) and off-premises (port 3080), as shown in Figure 7-105.

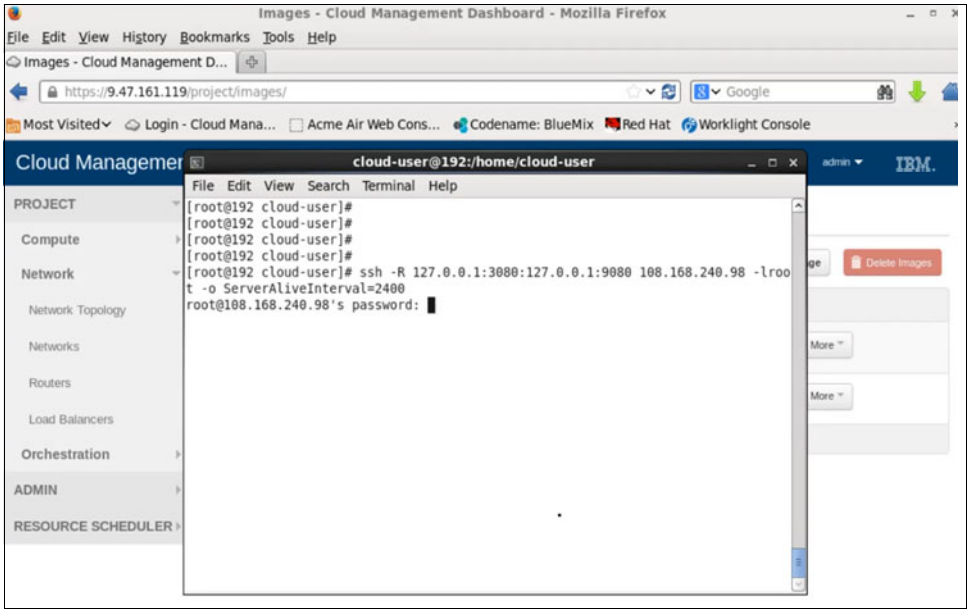


Figure 7-105 Create SSH tunnel for secure

Note: Software VPN is another alternative here.

7.4.3 Cloud users interact with a mobile application

Now we show how a typical cloud user such as Joe can interact with the systems of engagement application using a mobile device, as shown in Figure 7-106.

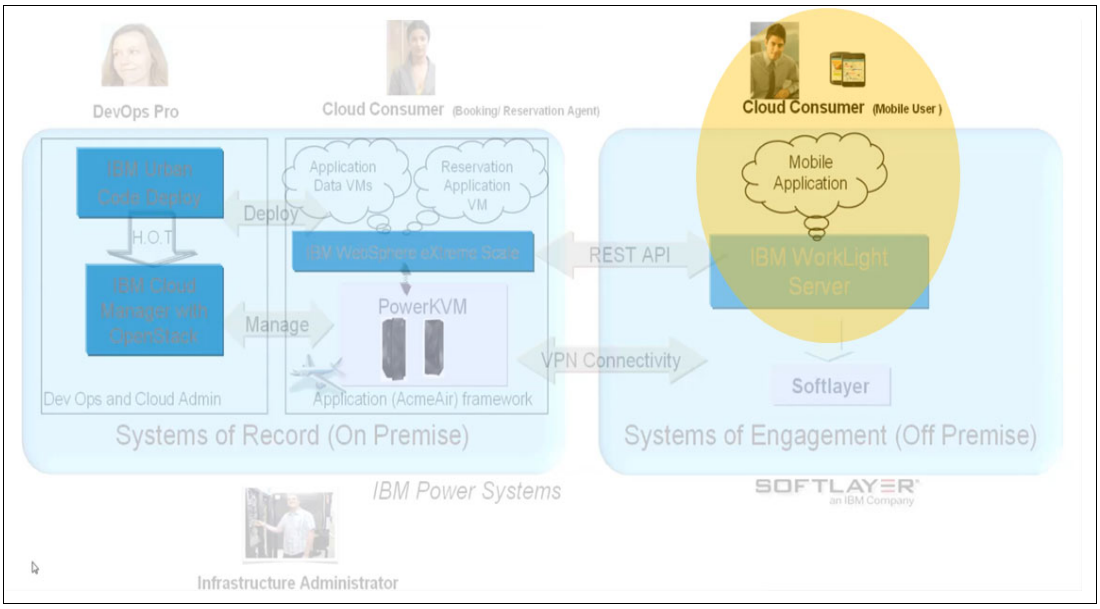


Figure 7-106 Interaction with mobile device

1. Joe launches the application from his mobile device and logs in to the app, as shown in Figure 7-107.



Figure 7-107 Launch mobile application

2. Joe modifies his flight information, which is quickly written back to the systems of record, as shown in Figure 7-108.



Figure 7-108 Modify flight information

7.4.4 Cloud user interactions with the reservations application

Now let's see how another cloud user, Sally, can interact with the systems of record application using the existing application front end, as shown in Figure 7-109.

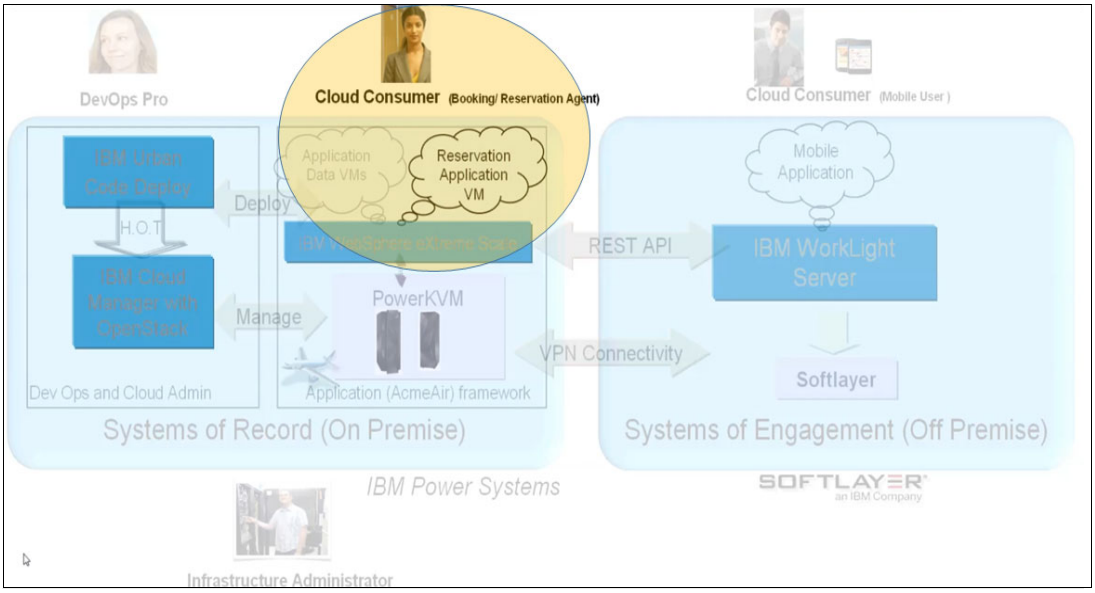


Figure 7-109 Cloud user interaction with reservation application

1. Sally logs in to the traditional browser-based system, as shown in Figure 7-110.

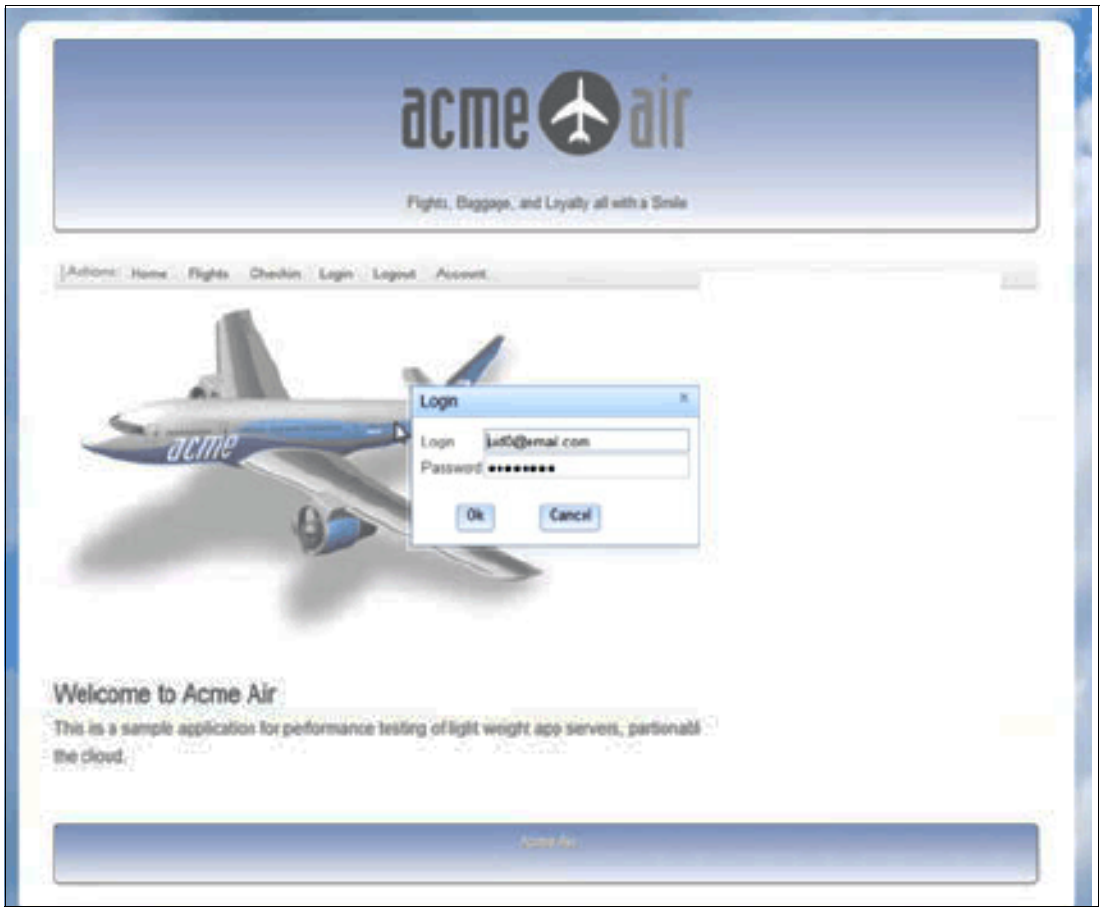


Figure 7-110 Log in to reservation system

2. Sally can search for flights, and then make and cancel reservations, as shown in Figure 7-111.



Figure 7-111 Reserve flights

The data is quickly written back to the same backend systems of record, as shown in Figure 7-112. The data is secure and reliable.

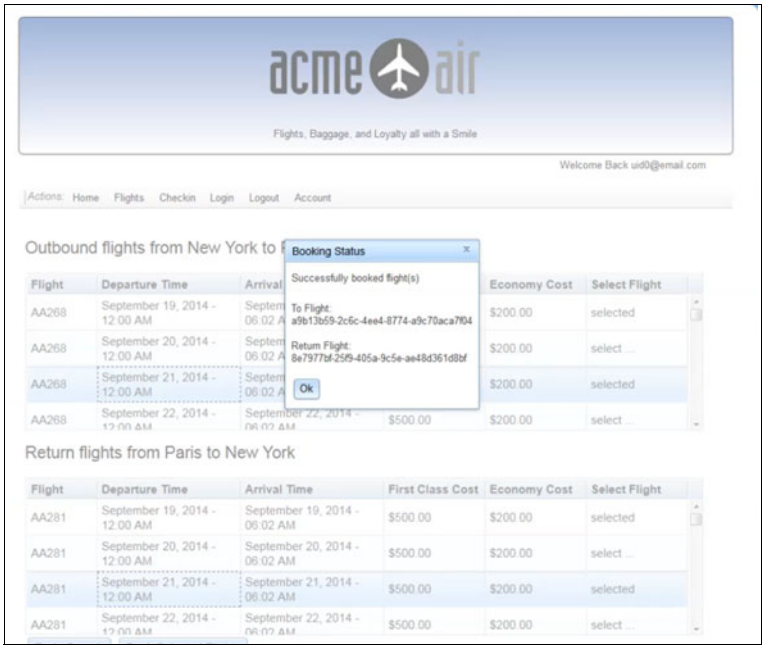


Figure 7-112 Flights booked

7.4.5 AcmeAirDataScale-PowerKVM.yaml file (HOT template)

Example 7-45 shows the `AcmeAirDataScale-PowerKVM.yaml` HOT template, which is defined to set up the provisioning of the Acme Air software stack on PowerKVM and to set other parameters, such as autoscaling, on demand for the software. Notice the bold code that represents the autoscaling policy setting and ceilometer alarm settings for peak resource use.

Example 7-45 AcmeAirDataScale-PowerKVM.yaml file (HOT template)

```
description: >
 Create AcmeAir sample application using simple autoscaling

parameters:
  key_name:
    type: string
    description: Name of an existing key pair to use for the template
    default: rajat
  instance_type:
    type: string
    description: Name of the OpenStack flavor to deploy with
    default: m1.small
  image_id:
    type: string
    description: Operating system image name for server
    default: rhel-6.5-ppc
  availability_zone:
   type: string
   description: The Availability Zone to launch the instance.
   default: powerkvm
  file_server:
    type: string
    description: File Server
    default: http://9.47.224.239/AcmeAir
resources:
  websphere_node:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      image: { get_param: image_id }
      flavor: { get_param: instance_type }
      availability_zone: { get_param: availability_zone }
      user_data_format: RAW
      user_data:
        get_resource: websphere_init

  websphere_init:
    type: OS::Heat::MultipartMime
    properties:
      parts:
        - config: {get_resource: java_install}
        - config: {get_resource: maven_install}
        - config: {get_resource: xip_dns}
        - config: {get_resource: liberty_profile_install}
        - config: {get_resource: extremescale_install}
        - config: {get_resource: objectgrid_install}
        - config: {get_resource: acmeair_build_install}
```

```
      - config: {get_resource: extremescale_remote_config}
      - config: {get_resource: liberty_config_start}

extremescale_catalog_node:
  type: OS::Nova::Server
  properties:
    key_name: { get_param: key_name }
    image: { get_param: image_id }
    flavor: { get_param: instance_type }
    availability_zone: { get_param: availability_zone }
    user_data_format: RAW
    user_data:
      get_resource: extremescale_catalog_init

extremescale_catalog_init:
  type: OS::Heat::MultipartMime
  properties:
    parts:
      - config: {get_resource: java_install}
      - config: {get_resource: maven_install}
      - config: {get_resource: xip_dns}
      - config: {get_resource: liberty_profile_install}
      - config: {get_resource: extremescale_install}
      - config: {get_resource: objectgrid_install}
      - config: {get_resource: acmeair_build_install}
      - config: {get_resource: extremescale_local_config}
      - config: {get_resource: extremescale_catalog_start}

extremescale_autoscale_group:
  type: OS::Heat::AutoScalingGroup
  properties:
    min_size: 1
    max_size: 6
    resource:
      type: OS::Nova::Server
      properties:
        key_name: { get_param: key_name }
        image: { get_param: image_id }
        flavor: { get_param: instance_type }
        availability_zone: { get_param: availability_zone }
        metadata: {"metering.stack": {get_param: "OS::stack_id"}}
        user_data_format: RAW
        user_data:
          get_resource: extremescale_container_init

extremescale_container_init:
  type: OS::Heat::MultipartMime
  properties:
    parts:
      - config: {get_resource: java_install}
      - config: {get_resource: maven_install}
      - config: {get_resource: xip_dns}
      - config: {get_resource: liberty_profile_install}
      - config: {get_resource: extremescale_install}
      - config: {get_resource: objectgrid_install}
```

```
        - config: {get_resource: acmeair_build_install}
        - config: {get_resource: extremescale_remote_config}
        - config: {get_resource: extremescale_container_start}
        - config: {get_resource: dd_cpu_spike}


# Scaling policies
  extremescale_scaleup_policy:
    type: OS::Heat::ScalingPolicy
    properties:
      adjustment_type: change_in_capacity
      auto_scaling_group_id: {get_resource: extremescale_autoscale_group}
      cooldown: 60
      scaling_adjustment: 1
  extremescale_scaledown_policy:
    type: OS::Heat::ScalingPolicy
    properties:
      adjustment_type: change_in_capacity
      auto_scaling_group_id: {get_resource: extremescale_autoscale_group}
      cooldown: 60
      scaling_adjustment: -1

# Ceilometer alarms for CPU usage
  cpu_alarm_high:
    type: OS::Ceilometer::Alarm
    properties:
      description: Scale-up if the average CPU > 50% for 1 minute
      meter_name: cpu_util
      statistic: max
      period: 60
      evaluation_periods: 1
      threshold: 50
      alarm_actions:
        - {get_attr: [extremescale_scaleup_policy, alarm_url]}
      matching_metadata: {'metadata.user_metadata.stack': {get_param:
"OS::stack_id"}}
      comparison_operator: gt
  cpu_alarm_low:
    type: OS::Ceilometer::Alarm
    properties:
      description: Scale-down if the average CPU < 15% for 1 minute
      meter_name: cpu_util
      statistic: max
      period: 60
      evaluation_periods: 1
      threshold: 15
      alarm_actions:
        - {get_attr: [extremescale_scaledown_policy, alarm_url]}
      matching_metadata: {'metadata.user_metadata.stack': {get_param:
"OS::stack_id"}}
      comparison_operator: lt

  dd_cpu_spike:
    type: OS::Heat::SoftwareConfig
    properties:
```

```
      config: |
        #!/bin/bash
        nohup dd if=/dev/zero of=/dev/null &

  java_install:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
            service iptables stop
            curl %FILE_SERVER%/ibm-java-ppc64-sdk-7.1-1.1.ppc64.rpm --insecure >
/tmp/ibm-java-ppc64-sdk-7.1-1.1.ppc64.rpm
            rpm -ivh /tmp/ibm-java-ppc64-sdk-7.1-1.1.ppc64.rpm
            echo export JAVA_HOME=/opt/ibm/java-ppc64-71 >> ~/.bashrc
            echo export PATH=/opt/ibm/java-ppc64-71/bin:$PATH >> ~/.bashrc
            source ~/.bashrc
            export
          params:
            "%FILE_SERVER%": { get_param: file_server }

  maven_install:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
            mkdir -p /acmeair
            curl %FILE_SERVER%/apache-maven-3.0.5-bin.tar.gz >
/acmeair/apache-maven-3.0.5-bin.tar.gz
            cd /acmeair
            tar -xvzf apache-maven-3.0.5-bin.tar.gz
            mv apache-maven-3.0.5 /usr/local
            ln -s /usr/local/apache-maven-3.0.5 /usr/local/maven
            echo export M2_HOME=/usr/local/maven >> ~/.bashrc
            echo export PATH=/usr/local/maven/bin:$PATH >> ~/.bashrc
            source ~/.bashrc
            export
            set
            mvn -version >> /var/log/acmeair.log
          params:
            "%FILE_SERVER%": { get_param: file_server }

  xip_dns:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
```

```
            # Set up system to use xip.io for forward and reverse dns lookups
            echo nameserver xip.io >> /etc/resolv.conf
            IP_ADDRESS=$(ifconfig eth0 | grep  'inet addr' | awk -F : {'print $2'}
| awk {'print $1'})
            echo "This node has an IP of "  $IP_ADDRESS  > /var/log/acmeair.log
            echo Setting hostname to $IP_ADDRESS.xip.io >> /var/log/acmeair.log
            hostname $IP_ADDRESS.xip.io
          params:
            "%FILE_SERVER%": { get_param: file_server }

  liberty_profile_install:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
            # Install  WebSphere Liberty Profile Developers Runtime to
/acmeair/wlp
            # Accept agreement and download from
https://developer.ibm.com/wasdev/downloads/liberty-profile-using-non-eclipse-envir
onments/wlp/8.5.5.2/wlp-developers-runtime-8.5.5.2.jar,
            # then host on any server of choice (10.10.0.7 in this case)
            cd /acmeair
            curl %FILE_SERVER%/wlp-developers-runtime-8.5.5.2.jar --insecure >
/acmeair/wlp-developers-runtime-8.5.5.2.jar
            java -jar wlp-developers-runtime-8.5.5.2.jar --acceptLicense /acmeair
          params:
            "%FILE_SERVER%": { get_param: file_server }

  extremescale_install:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
            # Install  WebSphere eXtreme Scale V8.6.0.5 For Developers - Liberty
Profile to /acmeair/wlp
            # Go to https://www.ibm.com/developerworks/downloads/ws/wsdg/ , and
select WebSphere eXtreme Scale for Developers - Liberty Profile for AIX, HP-UX,
Linux, Solaris, Windows, z/OS,
            # e.g. java -jar wxs-wlp_x.x.x.x.jar, then host on any server of
choice (10.10.0.7 in this case)
            cd /acmeair
            curl %FILE_SERVER%/wxs-wlp_8.6.0.4.jar --insecure >
/acmeair/wxs-wlp_8.6.0.4.jar
            java -jar wxs-wlp_8.6.0.4.jar --acceptLicense /acmeair
            echo export  WLP_SERVERDIR=/acmeair/wlp >> ~/.bashrc
            source ~/.bashrc
          params:
            "%FILE_SERVER%": { get_param: file_server }
```

```
  objectgrid_install:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
            # Install  WebSphere eXtreme Scale  /acmeair/ObjectGrid
            # Go to https://www.ibm.com/developerworks/downloads/ws/wsdg/ , and
select WebSphere eXtreme Scale trial,
            # e.g. extremescaletrialxxxx.zip , then host on any server of choice
(10.10.0.7 in this case)
            cd /acmeair
            curl %FILE_SERVER%/extremescaletrial860.tgz  --insecure >
/acmeair/extremescaletrial860.tgz
            tar -xvzf extremescaletrial860.tgz
            echo export  WXS_SERVERDIR=/acmeair/ObjectGrid >> ~/.bashrc
            source ~/.bashrc

            # Install the dependency that isn't available in Maven public
repositories
            cd $WXS_SERVERDIR/lib
            mvn install:install-file -Dfile=objectgrid.jar
-DgroupId=com.ibm.websphere.objectgrid -DartifactId=objectgrid -Dversion=8.6.0.2
-Dpackaging=jar
          params:
            "%FILE_SERVER%": { get_param: file_server }

  acmeair_build_install:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
            #yum install git -y
            # Get the acmeair code and build it
            mkdir -p  /acmeair/eclipse
            #git clone https://github.com/acmeair/acmeair.git
            curl %FILE_SERVER%/acmeair-src.tgz  --insecure > /tmp/acmeair-src.tgz
            cd /
            tar -xvzf /tmp/acmeair-src.tgz
            echo export ACMEAIR_SRCDIR=/acmeair/eclipse/acmeair >> ~/.bashrc
            source ~/.bashrc
            export >> /var/log/acmeair.log
            cd $ACMEAIR_SRCDIR
            mvn clean compile package install
            # The objectgrid.jar dependevy gets resolved when maven is run the
second time.
            cd $ACMEAIR_SRCDIR
            mvn clean compile package install
          params:
```

```
            "%FILE_SERVER%": { get_param: file_server }

  extremescale_remote_config:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
            # Create a WebSphere eXtreme Scale configuration..
            cd $WXS_SERVERDIR
            mkdir acmeair
            cp gettingstarted/* -R acmeair
            # Make a copy of the env.sh file and then edit it to coment out
current server classpath and customize it by adding a new line right after it with
the right values
            cp $WXS_SERVERDIR/acmeair/env.sh
$WXS_SERVERDIR/acmeair/env.sh.original
            sed -i 's/SAMPLE_SERVER_CLASSPATH=/#SAMPLE_SERVER_CLASSPATH=/g'
$WXS_SERVERDIR/acmeair/env.sh
            sed -i
'/#SAMPLE_SERVER_CLASSPATH/aSAMPLE_SERVER_CLASSPATH="${SAMPLE_COMMON_CLASSPATH}:${
SAMPLE_HOME}/server/bin:$ACMEAIR_SRCDIR/acmeair-common/target/classes:$ACMEAIR_SRC
DIR/acmeair-services-wxs/target/classes:/root/.m2/repository/commons-logging/commo
ns-logging/1.1.1/commons-logging-1.1.1.jar"' $WXS_SERVERDIR/acmeair/env.sh
            sed -i 's/localhost/%CATALOG_IP%.xip.io/g'
$WXS_SERVERDIR/acmeair/env.sh

            #For the distributed setup, make the webspehere server point to the
catalog server
            # Make a backup of the spring-config-acmeair-data-wxs-direct.xml and
spring-config-acmeair-data-wxs-direct-notx.xml files, and
            cp
/acmeair/eclipse/acmeair/acmeair-services-wxs/src/main/resources/spring-config-acm
eair-data-wxs-direct.xml
/acmeair/eclipse/acmeair/acmeair-services-wxs/src/main/resources/spring-config-acm
eair-data-wxs-direct.xml.original
            cp
/acmeair/eclipse/acmeair/acmeair-services-wxs/src/main/resources/spring-config-acm
eair-data-wxs-direct-notx.xml
/acmeair/eclipse/acmeair/acmeair-services-wxs/src/main/resources/spring-config-acm
eair-data-wxs-direct-notx.xml.original
            sed -i 's/127.0.0.1/%CATALOG_IP%.xip.io/g'
/acmeair/eclipse/acmeair/acmeair-services-wxs/src/main/resources/spring-config-acm
eair-data-wxs-direct.xml
            sed -i 's/127.0.0.1/%CATALOG_IP%.xip.io/g'
/acmeair/eclipse/acmeair/acmeair-services-wxs/src/main/resources/spring-config-acm
eair-data-wxs-direct-notx.xml

          params:
            "%FILE_SERVER%": { get_param: file_server }
           "%CATALOG_IP%": { get_attr: [extremescale_catalog_node, first_address]
}
```

```
  extremescale_local_config:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
            # Create a WebSphere eXtreme Scale configuration..
            cd $WXS_SERVERDIR
            mkdir acmeair
            cp gettingstarted/* -R acmeair
            # Make a copy of the env.sh file and then edit it to coment out
current server classpath and customize it by adding a new line right after it with
the right values
            cp $WXS_SERVERDIR/acmeair/env.sh
$WXS_SERVERDIR/acmeair/env.sh.original
            sed -i 's/SAMPLE_SERVER_CLASSPATH=/#SAMPLE_SERVER_CLASSPATH=/g'
$WXS_SERVERDIR/acmeair/env.sh
            sed -i
'/#SAMPLE_SERVER_CLASSPATH/aSAMPLE_SERVER_CLASSPATH="${SAMPLE_COMMON_CLASSPATH}:${
SAMPLE_HOME}/server/bin:$ACMEAIR_SRCDIR/acmeair-common/target/classes:$ACMEAIR_SRC
DIR/acmeair-services-wxs/target/classes:/root/.m2/repository/commons-logging/commo
ns-logging/1.1.1/commons-logging-1.1.1.jar"' $WXS_SERVERDIR/acmeair/env.sh

          params:
            "%FILE_SERVER%": { get_param: file_server }

  liberty_config_start:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            echo export WLP_SERVERDIR=/acmeair/wlp >> ~/.bashrc
            source ~/.bashrc
            # Create and start the WebSphere Liberty server and then deploy the
application
            cd $WLP_SERVERDIR
            bin/server create server1
            # Make a copy of the server.xml config file and add a line to allow
http calls from all hosts
            cp $WLP_SERVERDIR/usr/servers/server1/server.xml
$WLP_SERVERDIR/usr/servers/server1/server.xml.original
            sed -i '/id="defaultHttpEndpoint"/ahost="*"'
$WLP_SERVERDIR/usr/servers/server1/server.xml
            sed -i 's/<feature>jsp-2.2<\/feature>/<feature>jaxrs-1.1<\/feature>
<feature>jndi-1.0<\/feature> <feature>eXtremeScale.client-1.1<\/feature>/g'
$WLP_SERVERDIR/usr/servers/server1/server.xml

            # Recompile to refresh packages and stage war file
            cd $ACMEAIR_SRCDIR
            mvn clean compile package install
```

```
            cp
$ACMEAIR_SRCDIR/acmeair-webapp/target/acmeair-webapp-1.0-SNAPSHOT.war
$WLP_SERVERDIR/usr/servers/server1/dropins/

            # Start the WebSphere Liberty server
            cd $WLP_SERVERDIR
            bin/server start server1 &
            sleep 10
            #echo Finally, launch the URL
http://youripaddress:9080/acmeair-webapp-1.0-SNAPSHOT
            #echo Note, there are 200 customers with email from uid0@email.com to
uid199@email.com, and the passwords are all "password".
            # You can run the following command to test out how many containers
you have...
            # /acmeair/ObjectGrid/bin/xscmd.sh -c showMapSizes -g AcmeGrid -ms
mapSet -cep %CATALOG_IP%.xip.io:2809

          params:
            "%FILE_SERVER%": { get_param: file_server }

  extremescale_catalog_start:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
            # Next we copy the Acme Air specific eXtreme Scale configuration files
from our source directory
            mv /acmeair/ObjectGrid/acmeair/server/config/deployment.xml
/acmeair/ObjectGrid/acmeair/server/config/deployment.xml.original
            cp
/acmeair/eclipse/acmeair/acmeair-services-wxs/src/main/resources/deployment.xml
/acmeair/ObjectGrid/acmeair/server/config
            mv /acmeair/ObjectGrid/acmeair/server/config/objectgrid.xml
/acmeair/ObjectGrid/acmeair/server/config/objectgrid.xml.original
            cp
/acmeair/eclipse/acmeair/acmeair-services-wxs/src/main/resources/objectgrid.xml
/acmeair/ObjectGrid/acmeair/server/config

            # recompiled again...may not be necessary
            cd $ACMEAIR_SRCDIR
            mvn clean compile package install

            # Now start the Acme Air WebSphere eXtreme Scale configuration catalog
server
            # In one window start the catalog server
            cd $WXS_SERVERDIR/acmeair
            ./runcat.sh &
            sleep 30

            # In another window start a single container server
            cd $WXS_SERVERDIR/acmeair
            ./runcontainer.sh c0 &
```

```
            sleep 30

            # Now we will load sample data into eXtreme Scale
            # In another window, we do this by running a Acme Air loader program
            cd $ACMEAIR_SRCDIR/acmeair-loader
            mvn exec:java &
            sleep 10

          params:
            "%FILE_SERVER%": { get_param: file_server }

  extremescale_container_start:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            source ~/.bashrc
            # Next we copy the Acme Air specific eXtreme Scale configuration files
from our source directory
            mv /acmeair/ObjectGrid/acmeair/server/config/deployment.xml
/acmeair/ObjectGrid/acmeair/server/config/deployment.xml.original
            cp
/acmeair/eclipse/acmeair/acmeair-services-wxs/src/main/resources/deployment.xml
/acmeair/ObjectGrid/acmeair/server/config
            mv /acmeair/ObjectGrid/acmeair/server/config/objectgrid.xml
/acmeair/ObjectGrid/acmeair/server/config/objectgrid.xml.original
            cp
/acmeair/eclipse/acmeair/acmeair-services-wxs/src/main/resources/objectgrid.xml
/acmeair/ObjectGrid/acmeair/server/config

            # recompiled again...may not be necessary
            cd $ACMEAIR_SRCDIR
            mvn clean compile package install

            # In another window, Build a unique container name from the last octet
of the IPand  start a single container server
            cd $WXS_SERVERDIR/acmeair
            OCTET4=$(ifconfig eth0 | grep -A2 "$INTERFACE " | grep 'inet addr:' |
cut -d: -f2 | awk '{ print $1}' | cut -f4 -d.)
            echo $OCTET4 >> /var/log/acmeair.log
            CONTAINER_NAME=C$OCTET4
            echo $CONTAINER_NAME >> /var/log/acmeair.log
            ./runcontainer.sh $CONTAINER_NAME &
            sleep 10

          params:
            "%FILE_SERVER%": { get_param: file_server }

outputs:
  websphere_node_ip_output:
    description: The IP address of the deployed instance of resource
Websphere_node
    value:
```

```
str_replace:
  template:  http://%libertyip%:9080/acmeair-webapp-1.0-SNAPSHOT
  params:
    "%libertyip%": { get_attr: [ websphere_node , first_address ] }
```

7.4.6 Section summary

This section described and illustrated the following personas and how they interact with the different entities in a hybrid cloud:

- DevOps pro (Maureen) deploys application and pattern to the hybrid cloud.
- Infrastructure Administrator (Michael) monitors and configures the hybrid cloud.
- Cloud user (Joe) interacts with off-premises mobile application (system of engagement).
- Cloud user (Sally) interacts with on-premises reservations application (system of record).

By using the suite of tools and products that this demo shows, the user, the DevOps administrator, and enterprise operations gain the following benefits:

- Users
  - *Flexible access* through a mobile device and browser
  - *Access to the same data and enterprise services* through a hybrid cloud delivery model
- DevOps
  - *Simple and consistent* application deployment by using pattern-based tools
  - *Faster delivery* of features introduced by continuous application development
- Enterprise operations
  - *Improved resiliency and response times* due to infrastructure scale out and optimization
  - *Secure connection* between off-premises and on-premises locations
  - Maintainance and use of system of record application and data with full confidence in the *security and compliance* of their on-premises systems
  - Access to a *suite of tools and products* to manage infrastructure and application services

For more information, see the following:

- Acme Air application wiki and code on GitHub

  https://github.com/acmeair/acmeair/wiki
- IBM Systems for Cloud Computing

  http://www.ibm.com/systems/cloud
- Power Systems Cloud Solutions

  http://www.ibm.com/systems/power/solutions/cloud
- IBM Cloud Manager trial

  http://www.ibm.com/systems/power/solutions/cloud/onpower/starterkit.html
- IBM SoftLayer

  http://www.softlayer.com
- IBM MobileFirst Platform Foundation

  http://www.ibm.com/software/products/en/mobilefirstfoundation
- Software Defined Environments

http://www.ibm.com/developerworks/servicemanagement/sds/index.html

- IBM UrbanCode Deploy

  https://developer.ibm.com/urbancode/

In the next section, we briefly describe high availability (HA) capabilities for OpenStack.

7.5 High availability for applications

This section explains the HA options for the applications in the IBM Cloud Manager with OpenStack.

7.5.1 Introduction to OpenStack High Availability

High availability systems seek to minimize two primary things:

System downtime	Occurs when a user-facing service is unavailable beyond a specified maximum amount of time
Data loss	Accidental deletion or destruction of data

Most high-availability systems ensure protection against system downtime and data loss only if of a single failure. However, they are also expected to protect against cascading failures, where a single failure deteriorates into a series of consequential failures.

A crucial aspect of high availability is the elimination of *single points of failure* (SPOFs). A *SPOF* is an individual piece of equipment or software that will cause system downtime or data loss if it fails. To eliminate SPOFs, check that mechanisms exist for redundancy of these components and actions:

- Network components, such as switches and routers
- Applications and automatic service migration
- Storage components
- Facility services such as power, air conditioning, and fire protection

Most high-availability systems will fail if multiple independent (non-consequential) failures occur. In this case, most systems will protect data over maintaining availability.

High-availability systems typically achieve an uptime percentage of 99.99% or more, which roughly equates to less than an hour of cumulative downtime per year. To achieve this, high-availability systems need to keep recovery times after a failure to about one to two minutes, sometimes significantly less.

OpenStack currently meets such availability requirements[1] for its own infrastructure services, meaning that an uptime of 99.99% is feasible for the OpenStack infrastructure. However, OpenStack does not guarantee 99.99% availability for individual guest instances.

Preventing single points of failure can depend on whether or not a service is stateless.

[1] http://docs.openstack.org/high-availability-guide/content/ch-intro.html

7.5.2 Stateless versus stateful services

A *stateless* service is one that provides a response after your request, and then requires no further attention. To make a stateless service highly available, you need to provide redundant instances and balance their loads. OpenStack services that are stateless include nova-api, nova-conductor, glance-api, keystone-api, neutron-api, and nova-scheduler.

A *stateful* service is one where subsequent requests to the service depend on the results of the first request. Stateful services are more difficult to manage because a single action typically involves more than one request, so simply providing additional instances and load balancing will not solve the problem. For example, if the Cloud Manager user interface resets every time you go to a new page, it would not be useful. OpenStack services that are stateful include the OpenStack database and message queue.

Making stateful services highly available can depend on whether you choose an active/passive or active/active configuration.

7.5.3 Active/passive configuration

In an *active/passive* configuration, systems are set up to bring additional resources online to replace those that have failed. For example, OpenStack writes to the main database, although maintains a disaster recovery database that can be brought online if that the main database fails.

Typically, an active/passive installation for a stateless service maintains a redundant instance that can be brought online when required. Requests may be handled by using a virtual IP address to facilitate return to service with minimal reconfiguration required.

A typical active/passive installation for a stateful service maintains a replacement resource that can be brought online when required. A separate application (such as Pacemaker or Corosync) monitors these services and brings the backup online if necessary.

7.5.4 Active/active configuration

In an *active/active* configuration, systems also use a backup but manage both the main and redundant systems concurrently. This way, if there is a failure, the user is unlikely to notice. The backup system is already online and takes on the increased load while the main system is fixed and brought back online.

Typically, an active/active installation for a stateless service maintains a redundant instance, and requests are load balanced by using a virtual IP address and a load balancer, such as HAProxy, and a configurable Load Balancing as a Service (LBaaS), which has been available since the Havana release.

A typical active/active installation for a stateful service includes redundant services with all instances having an identical state. For example, updates to one instance of a database also update all other instances. This way, a request to one instance is the same as a request to any other. A load balancer manages the traffic to these systems to ensure that operational systems always handle the request.

These are some of the more common ways to implement these high-availability architectures, but they are not the only ways to do it. The important thing is to make sure that your services are redundant and available. How you achieve that is up to you.

Note: In this version of the IBM SDE architecture, we did not configure HA for applications in the IBM SDE lab environment. This is merely for information.

For more information about some of the more common options for highly available systems for OpenStack, see the OpenStack website:

http://docs.openstack.org/high-availability-guide/content/

For more information about high availability reference architecture designs and best practices for IBM high availability, see these sites and publications:

- IBM High Availability Center of Competency

  http://www.ibm.com/systems/services/labservices/solutions/hacoc.html
- *IBM SAN Volume Controller Stretched Cluster with PowerVM and PowerHA*, SG24-8142

  http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg248142.html
- Savage, Shirley S. “How to Make High Availability Service Integral to Your Business.” *IBM Systems Magazine*, January 2014.

  http://bit.ly/14wtfR7

Also see Appendix A, “IBM Software Defined Network for Virtual Environments” on page 711.

8

Recently released architecture and future direction

Software defined environment architecture and technology and the workloads that run on these are constantly evolving at high speed. In this section, we provide a preview of what is coming in terms of the Next Generation and future directions.

© Copyright IBM Corp. 2015. All rights reserved.

8.1 Software-defined infrastructure: Next generation

SDE with OpenStack is one element of the full spectrum software-defined landscape and Infrastructure that is being designed and built today, and in the future. Figure 2-84 provides and illustrates the architectural view of the *software-defined infrastructure* (SDI) based on IBM Platform Computing and open cloud architecture and technology.

Note: As discussed in Chapter 1, “Introduction to the IBM Software Defined Environment” on page 1, the primary scope and focus of this IBM Redbooks publication is to cover SDE with IBM Cloud Manager. This information in this section is to provide an introduction to the software-defined infrastructure offerings and platform, which will be discussed further in future IBM Redbooks publications.

A software-defined infrastructure is the enabler for today’s and tomorrow’s workloads.

Do you have a multitude and proliferation of scale-out applications in your data center, as Figure 8-1 shows? IBM Platform Computing can help with scheduling and acceleration of these multiple workload types, with diverse characteristics through infrastructure sharing in your existing environment.

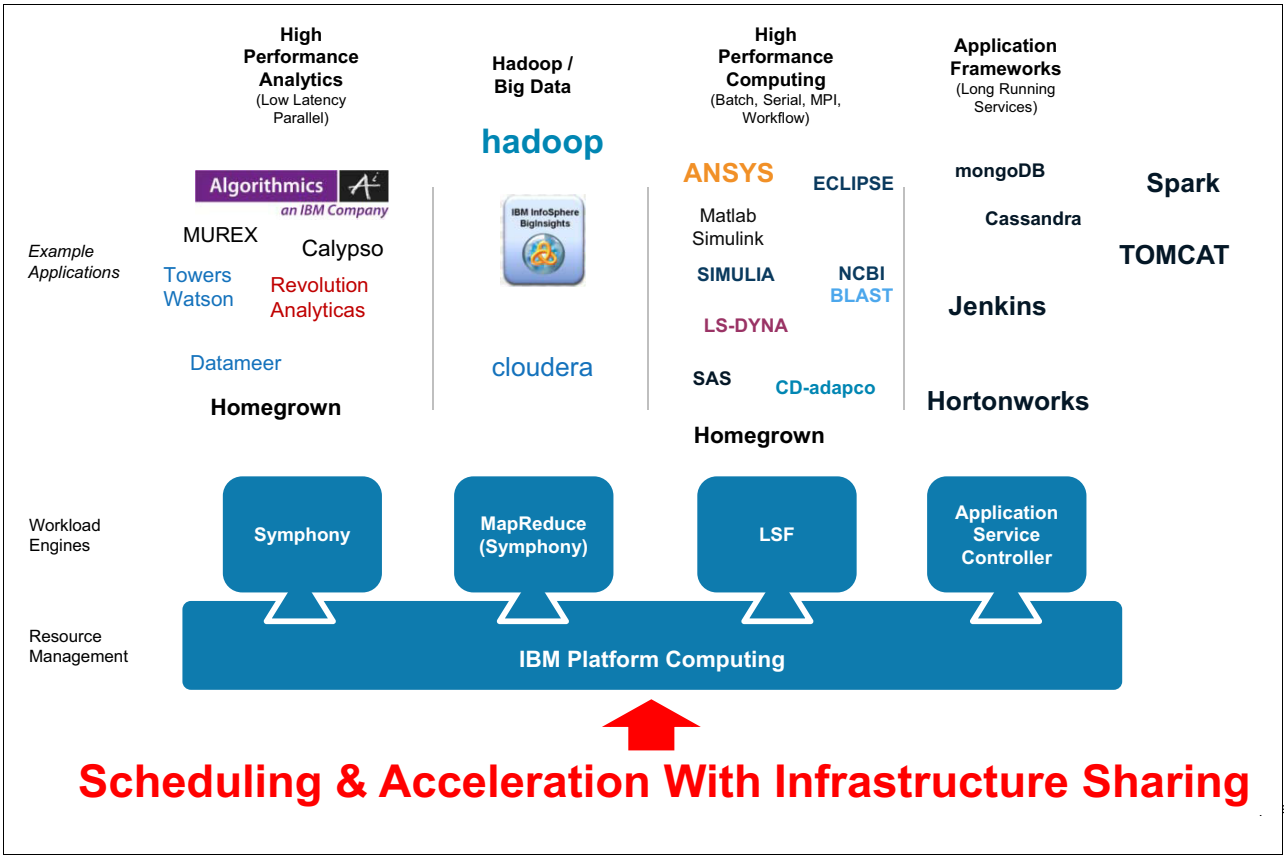


Figure 8-1 SDI - Platform Computing Scheduling and Acceleration with Infrastructure Sharing

As we know, the world does not contain all workloads that will run in virtualized environments managed by OpenStack and all applications are not one size fits all to run on the infrastructure. In addition as another example, some workloads are not all HPC or big data and analytics on bare metal either. This is the benefit of the SDE and SDI, further optimized by Platform Computing and Spectrum Scale, as a common data layer for unstructured file and object data.

Figure 8-2 illustrates the SDI Infrastructure with Platform Computing, and the capabilities and benefits of this layered reference architecture.

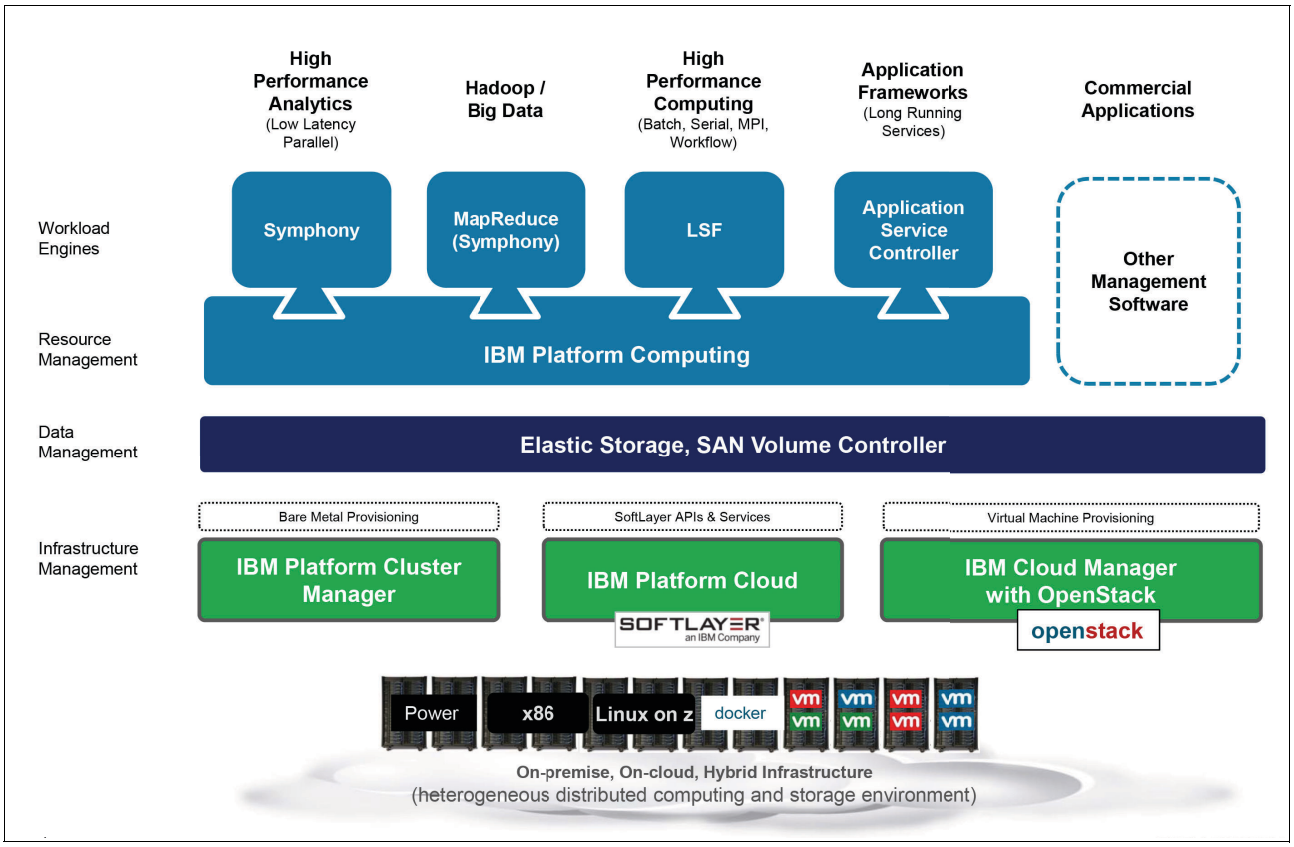


Figure 8-2 SDI with IBM Platform Computing reference architecture

The SDI with Platform Computing reference architecture is composed of the following layers and components.

8.1.1 Workload engines

The workload engines are optimized for specific application workload needs and characteristics, based on compute and data requirements, and integrated application support is provided via a rich API layer. Workload management, submission, and monitoring all occur through an user portal. Data movement and workflow automation are supported through policy and resource-aware job scheduling (business and technical parameter policies specific to the workloads: packing, striping, affinity, striping, topology, and so on). Unified and simplified administration is provided for cluster, grid, and cloud with real-time monitoring and analytics for capacity planning, health, and reporting.

Platform Symphony product family: High-performance analytics

Low-latency grid management platform for distributed computing and high performance analytics with sophisticated resource sharing. Deploy stand-alone or with IBM InfoSphere BigInsights. IBM Platform Symphony is the most powerful management software for running low-latency compute and data intensive applications on a scalable, multitenant, heterogeneous grid. It accelerates a wide variety of compute and data-intensive applications, quickly computing results while making optimal use of available infrastructure. Platform Symphony's efficient low-latency middleware and scheduling architecture is designed to provide the performance and agility required to predictably meet and exceed throughput goals for the most demanding analytic workloads infrastructure.

Key capabilities and features

- Leading experience due to 50%+ major investment banks as customers (translates to other industries)
- High scalability and better application performance due to fast, low latency processing (sub-millisecond)
- Proven business model with ROI for sharing grid infrastructure
- Accelerates service-oriented applications
- Extreme scalability and throughput with low latency
- Compute and data intensive applications on a single platform
- Sophisticated, hierarchical resource sharing
- Open and flexible: choice of OS, frameworks and languages
- Low latency, high throughput **(**<1ms response, 17,000 tasks per second)
- Large-scale 40 k cores per application, 128 k cores per grid
- Efficient shared services
- Heterogeneous and Open Linux, Windows, AIX, C/C++, C#, Java, Excel, Python, R, Hadoop
- Unique resource sharing model that enables lending and borrowing for maximum efficiency
- Optimized, low latency MapReduce implementation

Benefits

- Increase performance for more thorough analysis
- Reduces IT costs - increase use, simplify application onboarding, reduce administration costs

Platform Symphony product family: Hadoop and big data

Proven industrial-strength resource management and sharing solution addressing gaps in open source Hadoop to tap the hidden value of big data.

Key capabilities

- 100% Hadoop compatible; includes highly efficient MapReduce distribution
- Sophisticated resource management provides power and versatility far exceeding those of YARN
- Works either with HDFS or with IBM Spectrum Scale (powered by IBM GPFS technology) for fast, reliable, flexible, POSIX-compliant storage
- Interoperable with open source tools including Pig, Hive, and Apache HBase

Benefits

- Augmented Hadoop programming capability
- Fully compatible with existing Hadoop investments
- Resource scheduler allows multiple instances and versions of Hadoop to share resources

IBM Platform LSF

Powerful workload management and scheduling for demanding, distributed, and mission-critical high performance computing environments.

Key capabilities

- Complete:
  - Advanced feature-rich workload and resource management
  - Robust set of add-on features
  - Integrated application support
- Powerful
  - Policy, energy and resource-aware management
  - High-throughput scheduling for optimal performance
  - Advanced self-management
- Flexible
  - Heterogeneous platform support
  - Policy-driven automation
  - CLI, web services, APIs
  - Run Docker, Hadoop workloads
- Scalable
  - Thousands of concurrent users and jobs
  - Virtualized pool of shared resources
  - Flexible control, multiple policies

Benefits

- Optimal use: To reduce infrastructure cost
- Robust capabilities: improved administration, user productivity
- High throughput: faster time to results
- Save compute cycles: intelligent data management

Application Service Controller

Generalized resource management for a wide variety of distributed application frameworks, extending Platform Symphony's multitenant capabilities to include complex, long-running cloud-native services.

Key capabilities

- Configurable, optimized resource sharing among a broad set of distributed frameworks
- High-availability performance on native hardware with dynamic runtime elasticity
- Centralized interface to monitor and manage distributed system and application services

Benefits

- Increased application performance
- Optimized use of existing hardware resources
- Improved management efficiencies
- Greater resiliency and high availability

Platform HPC

IBM Platform HPC is easy-to-use, yet comprehensive technical computing management software. Its robust cluster and workload management capabilities are accessible using the latest design in web-based interfaces, which makes it powerful yet easy to use. Platform HPC simplifies the application integration process so that users can focus on their work rather than on managing a cluster. For applications requiring MPI, the robust commercial MPI library accelerates and scales HPC applications for shorter time to solution.

Other HPC cluster solutions just combine multiple tools and interfaces, which are not integrated, certified, or tested together. Platform HPC is a single product with a unified set of management capabilities that make it easy to harness the power and scalability of a technical computing cluster, resulting in shorter time to system readiness and user productivity as well as optimal throughput. Backed by the industry's best customer support, Platform HPC incorporates nearly two decades of product and technology leadership.

Key capabilities

- Highly configurable, easy-to-use cluster management
- Robust, optimized workload scheduling
- Web-based interface for access anywhere and simplified application integration
- Reporting and monitoring for workload, system
- NVIDIA GPU, Intel Xeon Phi co-processor scheduling and reporting
- Robust commercial MPI library, including IBM Platform MPI

Benefits

- Faster time to cluster and system readiness
- Improved user and administrator productivity
- Reduced user learning curve
- Shorter time to results
- Easy application integration
- Improved workload throughput
- Simplified administration and troubleshooting
- Reduced infrastructure and management costs
- Optimal resource use

8.1.2 Resource management

Prioritizes workload with business priorities to maximize performance according to SLAs (initial and ongoing). It enables consolidation of resource silos with diverse sharing frameworks (min/max, ratio, fair share, and so on) with varying levels of centralized control. It also maximizes throughput to ensure full use of resources. This provides production grade capabilities for the most stringent infrastructure requirements for scalability (scale up/down), failover, reliability, availability and security, for all application types across and throughout a heterogeneous infrastructure.

Data management

This is a combination of compute files, objects, and block.

Spectrum Scale (formerly GPFS)

Data management support for high-performance, scale-out file and object storage for multiple use cases (technical computing, big data, cloud). Handles up to yottabytes of data.

Key capabilities:

- Single global namespace with high performance access scales from departmental to global, multisite needs
- Automated tiering, data lifecycle management from Flash (6x acceleration) to Tape (10x savings)
- Enterprise ready - data security (encryption), availability, reliability, large scale proven
- Open, POSIX compliant, integrated with OpenStack and Hadoop (37% faster)

Benefits:

- Improves performance by removing data-related bottlenecks
- Improves collaboration across geographies
- Improves cost per performance by placing data on most applicable storage (flash to tape)
- Simplification by using a single file system for multiple applications

IBM Spectrum Scale Server

A fully-integrated data-centric storage system with building block scaling optimized for multi-workload access

Key capabilities:

- Integrated offering: IBM Power servers, storage enclosures and disks along as well as Spectrum Scale with native RAID
- Declustered RAID software with platter-to-client data protection
- Common repository, infrastructure, and namespace for workflows
- Automated, policy-driven management of storage
- Scalability from 40 TB to hundreds of PB and GBps
- 10 GigE, 40 GigE, FDR InfiniBand support

Benefits:

- Performance and scalability. Streaming data (up to 15GB/Sec) from a single system with near linear scalability as systems are added.
- Shared data platform. Simplify deployment and sharing
- Security and data protection. Multi-disk failure recovery in minutes.
- Ease of ownership. Comes as a complete solution with preloaded software, installation and three years of support.

Infrastructure management

Multiple infrastructure deployment models are supported through SDI and platform computing (bare metal hosting, private cloud, public cloud or dynamic hybrid)

IBM Platform Cluster Manager

Powerful lifecycle management for scale-out cluster environments

Key capabilities:

- Simplified management with cluster template designer
- Scales from single clusters to complex multi-team environments
- Robust, scalable alerting and reporting
- Automated infrastructure management with one-click cluster deployment
- Spectrum Scale cluster support

Benefits:

- Faster time to cluster readiness
- Unified interface for management and monitoring
- Increased administrator productivity
- Single infrastructure supporting multiple business needs

IBM Cloud Manager with OpenStack

An easy to deploy and use cloud management software offering based on OpenStack with IBM enhancements and support.

Key capabilities:

- Self-service provisioning of infrastructure services for rapid IT response
- Standard VM image capture and management with support for common business processes for administrators
- Usage metering with tracking and cost correlation to align departmental level use and services with the expense
- Production-grade cloud operations and interoperability at scale through an enhanced foundation and full OpenStack API compatibility
- Open computing cloud as an alternative to proprietary vendors, with world-class support from IBM
- Hybrid capability to IBM SoftLayer

Benefits:

- Accelerate time to market for new applications
- Optimized infrastructure use
- Reduced cost of cloud ownership
- Higher workload quality of service
- Integrated management: Approvals, metering, billing, users, and projects through a view
- Flexible, modular design: Based on OpenStack IaaS. Access to OpenStack APIs. Extensible via REST API, so it allows partners to easily customize the UI.

IBM Platform Computing Cloud Service

Enables deployment of hybrid cloud environments optimized for analytics and high performance applications.

Key capabilities:

- Complete, ready to run IBM Platform LSF® and Platform Symphony clusters in the SoftLayer cloud
- Option to include data management with elastic storage
- Data encryption, dedicated physical machines and network
- Run and fully supported by dedicated cloud ops team with extensive technical computing expertise
- Seamless job management across local and cloud based resources

Benefits:

- Add additional capacity in hours rather than months
- Security through physical isolation
- Reduced administrative burden, peace of mind
- Transparent user experience and easy application deployment

Heterogeneous infrastructure support

Across and through platforms (dynamic hybrid cloud, on-premises, off-premises, Power Systems (AIX, Linux), x86 (VMware, Linux), z Mainframe (Linux for System z), Docker containers.

8.1.3 OpenStack: Juno, latest release information

Enterprise readiness and user experience improvements are just a few of the highlights of the latest release of OpenStack, code-named Juno.[1] As the OpenStack ecosystem celebrates yet another successful milestone, I wanted to honor all of OpenStack's developers by highlighting some of the best new features that users can expect in this latest release, as well as celebrate some truly amazing contributions by team IBM.

Ecosystem growth

The OpenStack ecosystem continues to experience outstanding growth. In the Juno release, the number of contributors has reached the impressive milestone of more than 2,500 with an average of 400 contributing monthly in the last 12 months. There have been almost 130,000 commits with 61,000 in the last 12 months and an average of 4,000 monthly commits since the beginning of 2014, leaving no doubt that OpenStack is the place to be for IaaS open source development.

Like many of these loyal developers and sponsors, IBM remains committed to the success of OpenStack by increasing contributions focused on improving the OpenStack ecosystem for the benefit of the entire OpenStack community and accelerating its growth.

Increased enterprise security

IBM continues to lead several enterprise integration efforts in OpenStack's Identity Service Project, code-named Keystone. Key enhancements by the IBM team include federating multiple Keystones, enhanced auditing, and better support for multiple identity backends. Consistent with the Icehouse release, IBM contributors collaborated heavily with contributors from CERN and Rackspace on new enhancements that enable Keystones from multiple clouds to work together using industry standard federation protocols thus laying a solid foundation for enabling OpenStack based hybrid clouds.

[1] Topol, Brad. "A guide to the OpenStack Juno release." IBM Thoughts on Cloud blog, October 2014. http://thoughtsoncloud.com/2014/10/guide-openstack-juno-release/

For users, this means it will be simpler to combine their on-premises OpenStack cloud with the public cloud of their choice. IBM contributors also continued to expand critical CADF Standard-based auditing work to include audit support for users authenticating through a federated environment or when a role assignment is created or deleted.

Furthermore, full support for multiple identity backends was realized in the Juno release. This enables Keystone to support different corporate directories, plus it allows a deployer to specify using an SQL backend for local service accounts and an LDAP backend for all other users. This results in providing the users a much simpler approach for integrating with their read only LDAP backends.

Block storage improvements

For two consecutive release cycles, IBM has continued to lead in contributions to OpenStack's Block Storage project, code-named Cinder. This includes contributions such as the base enablement of volume replication, making it possible for Cinder to create volumes with a replica on the storage backend that can be quickly switched to in the case of a catastrophic storage system failure.

IBM also committed patches to enable globalization support in Cinder logs, making it possible for OpenStack users to view Cinder logs in languages other than English. Security was enhanced via patches to scrub passwords from logs and to enhance the existing SSH function to not ignore unexpected system keys.

Enhanced user experience

In an effort to improve the OpenStack user experience, IBM increased its contributions in the Juno cycle to the OpenStack dashboard project, code-named Horizon. IBMers contributed more than 1,500 code reviews and also were Horizon's third-highest committer. New key features were added such as custom tooltip support, filtering and pagination support, improved help text, and increased globalization support. They also continue to work on improving performance and responsiveness by moving Horizon to a client-side rendering model.

New API

For the OpenStack Compute project code-named Nova, IBM contributors led the development of the design of the new V2.1 API. This work supports the ability to make API changes in the future with much less overhead than the current process including backwards incompatible changes which were not previously possible. We also made numerous bug fixing contributions across Nova and were part of the team monitoring the health of the OpenStack community development system and applied fixes to a variety of bugs affecting OpenStack Continuous Integration as problems arose.

It's all about interoperability

IBM has also been a strong contributor to Refstack, which is the OpenStack community project focused on assessing OpenStack interoperability and provides an API test certification process for OpenStack service and product vendors. IBM was the top contributor to the Refstack-client project which is a test tool that runs compliance tests against a target OpenStack deployment and the second highest contributor to the Refstack project which is a community portal that stores test results, and performs analysis and reporting of compliance outcomes. Refstack ensures that OpenStack based offerings are interoperable which is the true value behind building on open technologies.

8.1.4 Next-generation architecture for information and insights

The next generation of the IBM big data and analytics reference architecture is shown in Figure 8-3 with the corresponding IBM products and solutions. This also represents an illustration of the next generation BigInsights and analytics reference architecture at the detailed capabilities level. SDE with OpenStack and also with Platform Computing are ideal for meeting the needs of this next generation architecture.

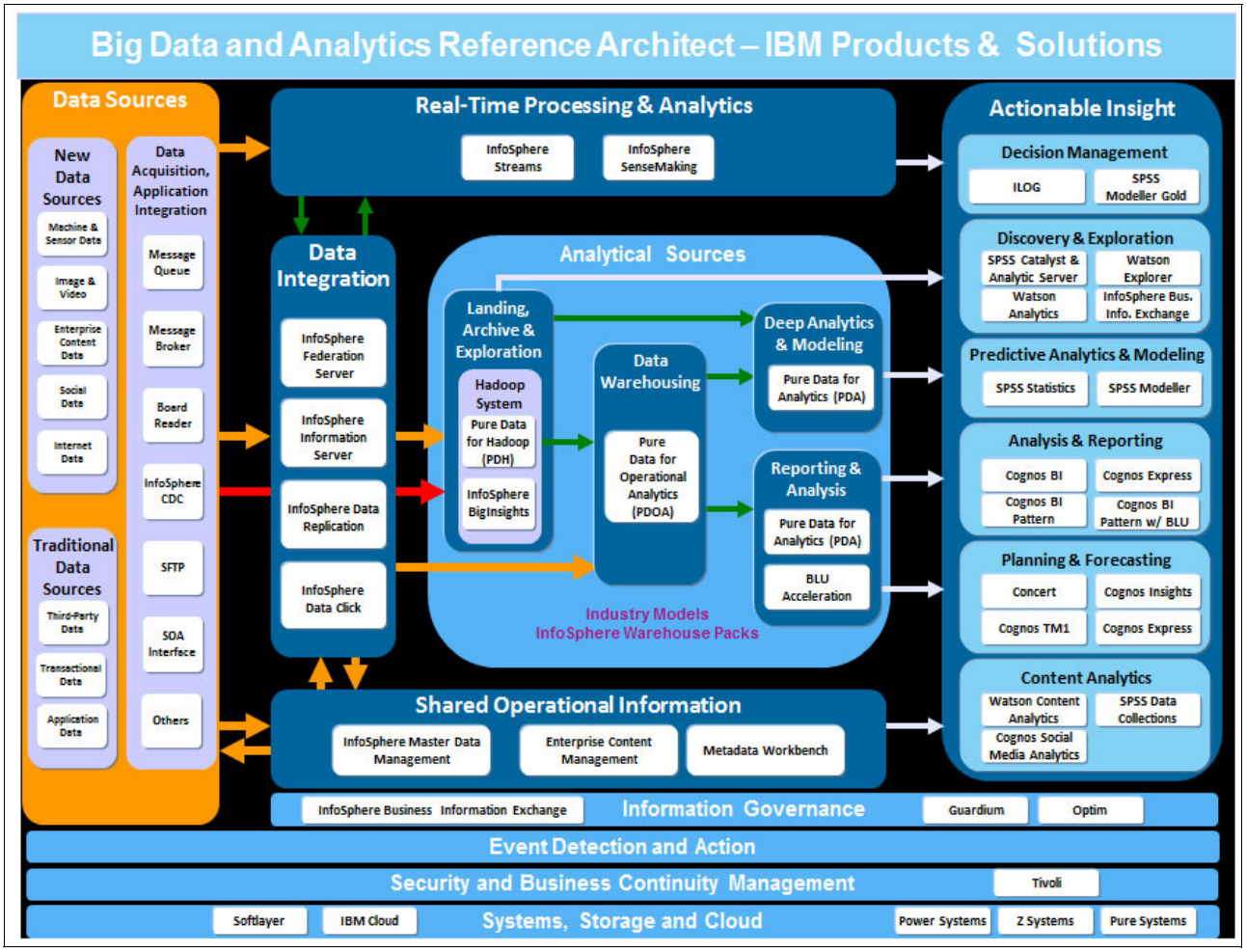


Figure 8-3 Next Generation - big data and analytics reference architecture - Products and Solutions

In Chapter 4, “Cloud provider installation guidelines and implementation scenarios” on page 195 in the next chapter, we cover the cloud provider installation guidelines and configuration.

A

IBM Software Defined Network for Virtual Environments

This appendix provides an overview of IBM Software Defined Network for Virtual Environments (SDN VE), how it plays a role in the holistic view the software-defined environment (SDE), and on a more finite basis, the software-defined IT infrastructure (SDI).

This appendix covers the following topics:

- "Software-defined networking overview and relationship to SDE" on page 712
- "SDN reference architecture and standards" on page 723
- "IBM Software Defined Network for Virtual Environments context" on page 730
- "IBM SDN scenarios and use cases" on page 738
- "SDN installation process of the SDN VE and DOVE components" on page 741

© Copyright IBM Corp. 2015. All rights reserved.

Software-defined networking overview and relationship to SDE

As explained in this IBM Redbooks publication, a Software Defined Environment (SDE) optimizes the entire computing infrastructure (compute, storage, and network resources) so that it can adapt to the type of work required. In today's environment, resources are assigned manually to workloads; this happens automatically in SDE. In an SDE, workloads are dynamically assigned to IT resources based on application characteristics, security and service level policies, and the best-available resources to deliver continuous, dynamic optimization and reconfiguration to address infrastructure issues in a rapid and responsive manner. Continuous optimization to instantly address infrastructure issues and improve response to business needs and proactive management of IT resources to improve efficiency and control costs of service delivery are other drivers that make this approach so powerful. Underlying all of this are policy-based compliance checks and updates, used by Open Standards-based application programming interfaces (APIs) and protocols in a centrally managed environment.

Remember that, in an SDE, workloads do not depend on specific systems. That is a major strength of the approach: Everything runs on the most appropriate combination of hardware assets, whatever that might be. Thanks to open standards, the enterprise is free to choose infrastructure as it sees fit. Investments are protected and there is no need to "lock in" to any one vendor's technology. In addition, responsiveness to new demands is greatly enhanced because assets can be readily repurposed. The infrastructure adapts to business need, rather than the other way around.

The networking industry is at a crossroad. Software-defined networking (SDN) and network functions virtualization (NFV) are presenting both significant challenges and unlimited opportunities for the future of transferring information and communicating across the globe. SDN enables users to program network layers, separating the data plane from the control plane while NFV allows for agile placement of networking services when and where they are needed. By enabling this level of programmability, SDN and NFV can enable users to optimize their network resources, increase network agility, service innovation, accelerate service time-to-market, extract business intelligence and ultimately enable dynamic, service-driven virtual networks.

Exploding levels of data, virtualized infrastructure and cloud computing leave users with tremendous demands on their networks and management resources. To date approaches to virtualize the network layer and add programmability have left the industry fragmented making adoption complex. Leading vendors realize this cannot continue. At the same time, enterprise users and service providers are seeking solutions to enable virtual and mobile networks to support virtual devices that are mobile and disconnected from a physical network design.

SDN is a new network paradigm that separates each network service from its point of attachment to the network, creating a far more dynamic, flexible, automated and manageable architecture. Administrators can easily move virtual resources throughout the network, create private virtual networks that meet specific performance and security needs, and a host of other high-value applications.

The key to SDN is an innovative approach to controlling how data flows through a network. In a traditional network, data flow is controlled by switches and routers. Each switch and router contains the following basic elements:

- *Data plane.* Physically carries data packets from one port to another by following rules that are programmed into the device hardware. The data forwarding plane operates at the speed of the network (wire speed).
- *Control plane.* Contains the logic that the device uses to program the data plane, so packets are forwarded correctly throughout the network.

- *Management plane.* Lets an administrator log in to the device and configure it for basic activities. Most devices can be configured locally or through a network management tool.

Vendors use control plane software to optimize data flow to achieve high performance and a competitive advantage. The switch-based control plane gives network administrators little opportunity to increase data flow efficiency across the network as a whole.

SDN abstracts flow control from individual devices to the network level. Similar to server virtualization where virtual machines are de-coupled from the physical server, network-wide virtualization gives administrators the power to define network flows that meet the connectivity requirements of end stations and address the specific needs of discrete user communities.

Figure A-1 shows, in the first diagram, a traditional network, and also illustrates the differences between a traditional network and SDN.

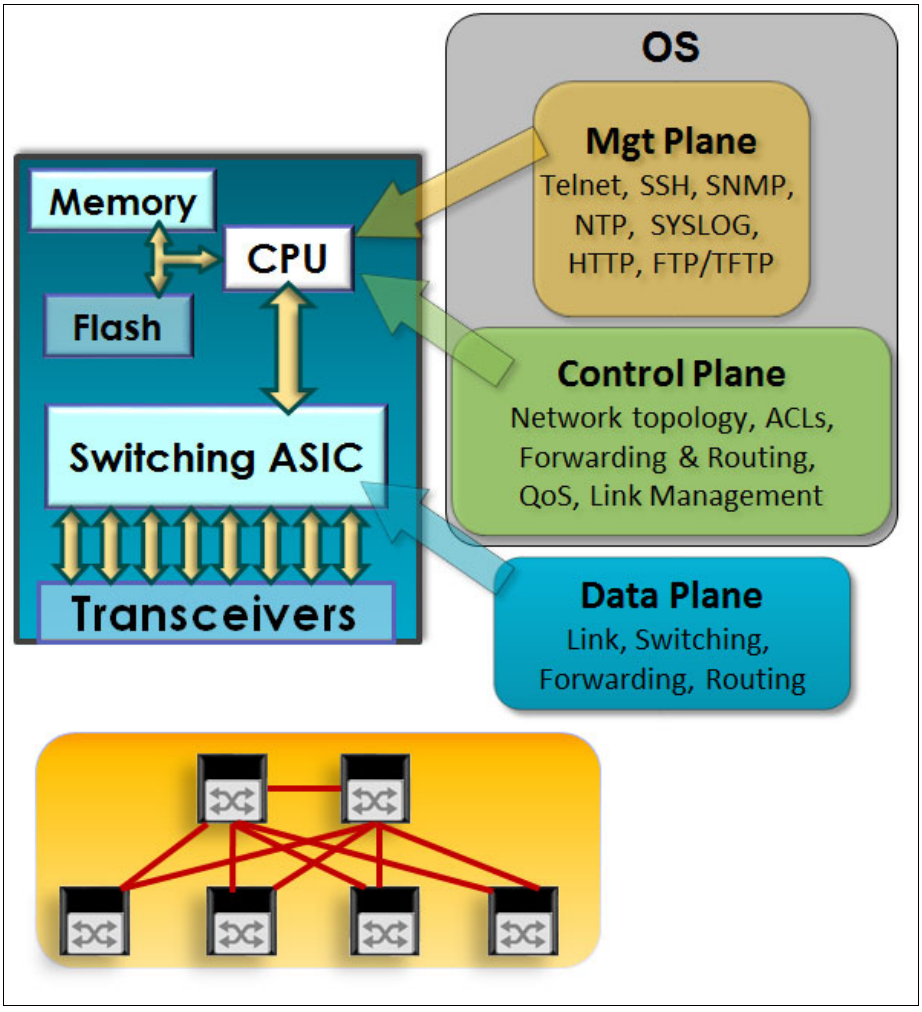


Figure A-1 Traditional network

You can see the three separated planes: Management plane, control plane, and data plane. In a traditional network, each network element has its own management and control plane, and there is distinct separation of each of the three planes.

Limitations of current networking technologies

Meeting current market requirements is virtually impossible with traditional network architectures.[1] Faced with flat or reduced budgets, enterprise IT departments are trying to squeeze the most from their networks using device-level management tools and manual processes. Carriers face similar challenges as demand for mobility and bandwidth explodes; profits are being eroded by escalating capital equipment costs and flat or declining revenue. Existing network architectures were not designed to meet the requirements of today's users, enterprises, and carriers. Rather, network designers are constrained by the following limitations of current networks:

- Complexity that leads to static

  Networking technology to date has consisted largely of discrete sets of protocols designed to connect hosts reliably over arbitrary distances, link speeds, and topologies. To meet business and technical needs over the last few decades, the industry has evolved networking protocols to deliver higher performance and reliability, broader connectivity, and more stringent security.

  Protocols tend to be defined in isolation, however, with each solving a specific problem and without the benefit of any fundamental abstractions. This has resulted in one of the primary limitations of today's networks: complexity. For example, to add or move any device, IT must touch multiple switches, routers, firewalls, Web authentication portals, and so on, and update ACLs, VLANs, quality of services (QoS), and other protocol-based mechanisms using device-level management tools. In addition, network topology, vendor switch model, and software version all must be taken into account. Due to this complexity, today's networks are relatively static as IT seeks to minimize the risk of service disruption.

  The static nature of networks is in stark contrast to the dynamic nature of today's server environment, where server virtualization has greatly increased the number of hosts requiring network connectivity and fundamentally altered assumptions about the physical location of hosts. Before virtualization, applications resided on a single server and primarily exchanged traffic with select clients. Today, applications are distributed across multiple virtual machines (VMs), which exchange traffic flows with each other. VMs migrate to optimize and rebalance server workloads, causing the physical end points of existing flows to change (sometimes rapidly) over time. VM migration challenges many aspects of traditional networking, from addressing schemes and namespaces to the basic concept of a segmented, routing-based design.

  In addition to adopting virtualization technologies, many enterprises today operate an IP converged network for voice, data, and video traffic. While existing networks can provide differentiated QoS levels for different applications, the provisioning of those resources is highly manual. IT must configure each vendor's equipment separately, and adjust parameters such as network bandwidth and QoS on a per-session, per-application basis. Because of its static nature, the network cannot dynamically adapt to changing traffic, application, and user demands.

- Inconsistent policies

  To implement a network-wide policy, IT might have to configure thousands of devices and mechanisms. For example, every time a new virtual machine is brought up, it can take hours, in some cases days, for IT to reconfigure ACLs across the entire network. The complexity of today's networks makes it difficult for IT to apply a consistent set of access, security, QoS, and other policies to increasingly mobile users, which leaves the enterprise vulnerable to security breaches, non-compliance with regulations, and other negative consequences.

[1] https://www.opennetworking.org/sdn-resources/sdn-library/whitepapers

- Inability to scale

  As demands on the data center rapidly grow, so too must the network grow. However, the network becomes vastly more complex with the addition of hundreds or thousands of network devices that must be configured and managed. IT has also relied on link oversubscription to scale the network, based on predictable traffic patterns. But in today's virtualized data centers, traffic patterns are dynamic and, therefore, unpredictable.

  Mega-operators, such as Google, Yahoo!, and Facebook, face even more daunting scalability challenges. These service providers employ large-scale parallel processing algorithms and associated data sets across their entire computing pool. As the scope of user applications increases (for example, crawling and indexing the entire World Wide Web to instantly return search results to users), the number of computing elements explodes and data set exchanges among Compute nodes can reach petabytes. These companies need so-called hyperscale networks that can provide high-performance, low-cost connectivity among hundreds of thousands—potentially millions—of physical servers. Such scaling cannot be done with manual configuration.

  To stay competitive, carriers must deliver ever-higher value, better-differentiated services to customers. Multi-tenancy further complicates their task, as the network must serve groups of users with different applications and different performance needs. Key operations that appear relatively straightforward, such as steering a customer's traffic flows to provide customized performance control or on-demand delivery, are complex to implement with existing networks, especially at carrier scale. They require specialized devices at the network edge, and that increases capital and operational expenses and time to market to introduce new services.

Vendor dependence

Carriers and enterprises seek to deploy new capabilities and services in rapid response to changing business needs or user demands. However, their ability to respond is hindered by vendors' equipment product cycles, which can range to three years or more. Lack of standard, open interfaces limits the ability of network operators to tailor the network to their individual environments.

This mismatch between market requirements and network capabilities has brought the industry to a tipping point. In response, the industry has created the SDN architecture and is developing associated standards.

Figure A-2 on page 716, illustrates the SDN-based network.

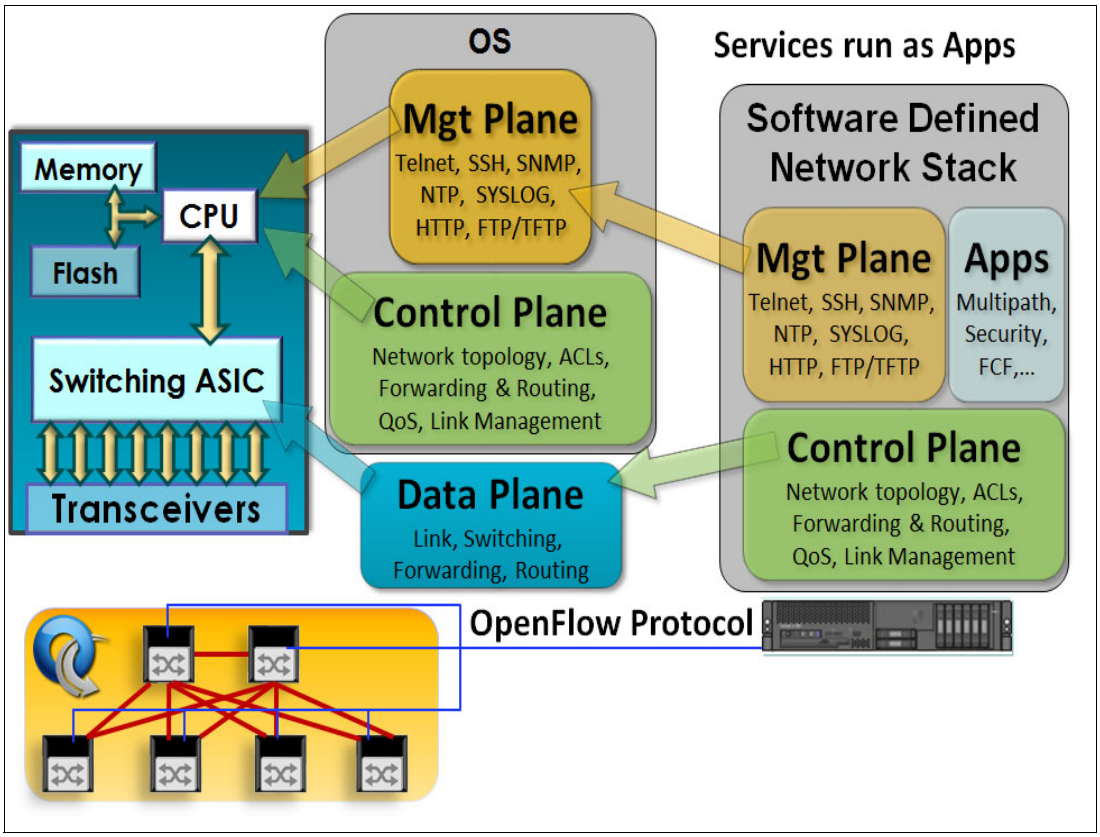


Figure A-2 Software-defined network

In the SDN, services run as applications, and the control plane is actually extracted from the network. The physical separation of the network control plane occurs from the forwarding plane, and where a control plane controls several devices.

The Ethernet and IP aspects of networking are not described in this book (aside from section 3.4, “Hypervisors xKVM, PowerKVM, PowerVM, z/VM, VMware” on page 102). The Open Systems Interconnection Reference Model, or OSI model that was developed by the International Organization for Standardization (ISO/OSI) is not described, either. This OSI approach has provided IT professionals with a structured, layered model that has been applied to serve the applications. Instead, we compare and contrast the monolithic approach of networking devices versus the principles of SDN. The networking devices approach is monolithic in the sense that there is currently no modularity, structure, or layered model in networking devices. The intelligence is kept inside each networking device (for example, routers and switches). Therefore, each device needs to “think” autonomously and apply networking features independently.

SDN pulls the intelligence away from the hardware while still implementing the rich feature set that the networking industry has had: Access Control List (ACL) and spanning tree are examples of a few areas that SDN can help rethink. Imagine an evolution of how the network itself can serve the applications, in further extending the solution for a truly application-aware environment that responds faster to the needs of the business. This evolution uses a modular approach that is structured and layered to provide the same functions as a traditional network device, yet in a centralized and highly available fashion. This controller provides a level of abstraction that lets networking embrace this technological shift to start enabling businesses at the same pace that Software Defined Computing and software-defined storage are already delivering in existing data centers.

Benefits of software-defined networking

So, why SDN rather than other options? The immediate advantage is simplicity. A good example that illustrates the benefits of SDN is provisioning and deprovisioning. We know that workflows can automate networking. Examples of working deployments are everywhere.

The SDN architecture has several advantages:

- *Directly programmable.* Network control is directly programmable because it is decoupled from forwarding functions.
- *Agile.* Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
- C*entrally managed.* Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
- *Programmatically configured.* SDN lets network managers configure, manage, secure, and optimize network resources quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
- *Open standards-based and vendor-neutral.* When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers rather than multiple, vendor-specific devices and protocols.

People and tools are required to automate the provisioning and de-provisioning of traditional networking resources. We need people: skilled personnel that know how to program “scripts” and manage tools. We need tools: tools that can run “scripts” that access each and every networking device part of the workflow and configure it using a set of commands that are vendor-dependent. If you consider an intrusion detection system (IDS) from vendor A, a firewall from vendor B, and a switch from vendor C, these devices combined can make the workflow quite complex to plan, implement, and manage. This is true even if we choose the same vendor.

SDN can simplify automation by using the abstraction of the structured layered model described above. We still need skilled personnel to know what result is needed. These skilled personnel do not need to know the complexity to obtain the result. The same is true for an SDN enabled network: the administrator can pass data to the controller, and the controller will handle the complexity.

Provisioning (or deprovisioning) is just an example of what SDN can do for the client. Think about the complexity of handling multitier systems that require complex interaction between server, storage, and network resources. When the client manages multitenant resources, the ability to bring new customers (or services) online is expected by CIOs that report back to the business. Imagine a simpler way to handle merges and acquisitions where you need to make overlapping IP subnets and VLAN rag IDs on the same physical infrastructure.

Software Defined Environments that are paired with Software Defined Networks can positively affect how we build data centers now. Table A-1 on page 718 provides a comparison between classical or traditional networks and SDNs.

Table A-1 Comparison between classical and traditional networks and SDN

Comparison	Classic or traditional network	Software Defined Networks (SDNs)
Network topology	► Network consists of many tiers, where each layer duplicates many of the IP/Ethernet packets, this adds cumulative end-to-end latency and requires significant amounts of processing and memory. ► Data traffic between racks of servers and storage needs to travel up and down a logical tree structure, which adds latency and potentially creates congestion on inter-switch links (ISLs). ► Network loops are prevented by using Spanning Tree Protocol (STP) which allows only one active path between any two switches. This means that ISL bandwidth is limited to a single logical connection, which may lead to ISL bottlenecks.	► Removing tiers from a traditional hierarchical data center network and collapses into a two tier network (access switches, also known as top of rack (TOR) switches, and core switches), connected devices can communicate with each other without using an intermediate router. ►Flatter networks also include elimination of STP. Replacing the STP protocol allows the network to support a fabric topology (tree, ring, mesh, or core/edge) while avoiding ISL bottlenecks.
Scaling up and down	► Do not scale in a cost effective or performance effective manner. Scaling requires adding more tiers to the network, more physical switches, and more physical service appliances.	► Fabrics use multiple least cost paths for high performance and reliability, and are more elastic (scaling up or down as required).
Capex and Opex	► Installation and maintenance of this physical compute model requires both high capital expense and high operating expense. The high capital expense is due to the large number of underused servers and multiple interconnect networks. ► High operational expense is driven by high maintenance and energy consumption of poorly used servers, high levels of manual network and systems administration.	► Flattening the network reduces capital expense through the elimination of dedicated storage, cluster and management adapters and their associated switches, and the elimination of traditional networking tiers. ► Operating expense is also reduced through management simplification by enabling a single console to manage the resulting converged fabric.
Network management	► Conventional data centers use several tools to manage their server, storage, network and hypervisor elements.	► Converging and flattening the network leads to simplified physical network management.

Comparison	Classic or traditional network	Software Defined Networks (SDNs)
Network subscription level	► Network can be over-provisioned most of the time. This approach provides an acceptable user experience, but it does not scale in a cost effective manner.	► To help subscription levels, Software Defined Networks provide a network with characteristics such as any to any connectivity, fairness, and non-blocking.
Virtualization environment	► Conventional data centers have consisted of lightly used servers running a bare metal operating system or a hypervisor with a small number of virtual machines (VMs).	► Highly virtualized, which leads to high availability and better performance.

In conclusion, the explosion of mobile devices and content, server virtualization, and advent of cloud services are among the trends driving the networking industry to reexamine traditional network architectures. Many conventional networks are hierarchical, built with tiers of Ethernet switches arranged in a tree structure. This design made sense when client/server computing was dominant, but such a static architecture is ill-suited to the dynamic computing and storage needs of today's enterprise data centers, campuses, and carrier environments.

Trends such as user mobility, server virtualization, IT-as-a-service, big data, and the need to rapidly respond to changing business conditions place significant demands on the network—demands that today's conventional network architectures cannot handle. Software-Defined Networking provides a new, dynamic network architecture that transforms traditional network backbones into rich service-delivery platforms.

By decoupling the network control and data planes, OpenFlow-based SDN architecture abstracts the underlying infrastructure from the applications that use it, allowing the network to become as programmable and manageable at scale as the computer infrastructure that it increasingly resembles. An SDN approach fosters network virtualization, enabling IT staff to manage their servers, applications, storage, and networks with a common approach and tool set. Whether in a carrier environment or enterprise data center and campus, SDN adoption can improve network manageability, scalability, and agility.

The relationship between SDE and SDN

Figure A-3 shows how SDN relates with SDE and with the physical infrastructure.

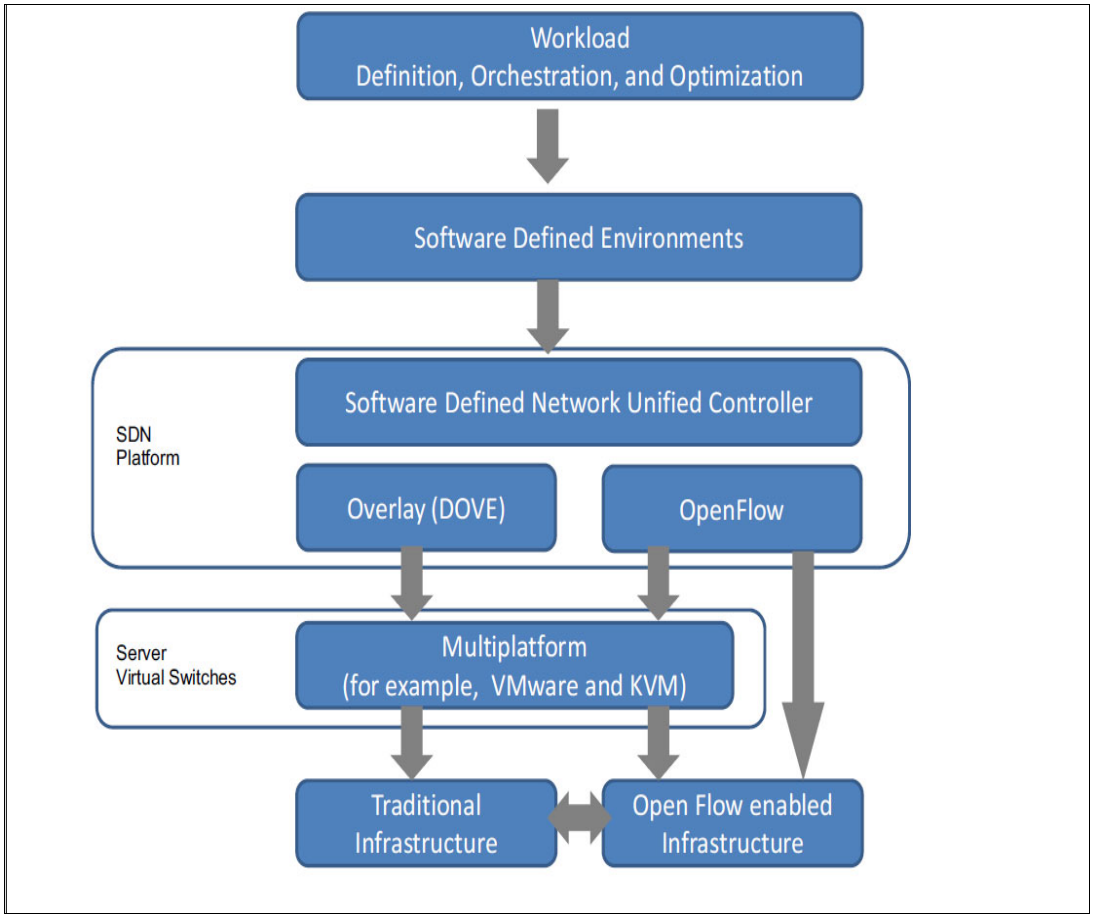


Figure A-3 SDN in the new software-defined IT infrastructure

The relationship between the business, workload, SDE, and SDN from a top-down perspective has already been described in this book. This section focuses on the layers below the SDN platform. With a single control point (SDN Unified Controller), we can abstract the resources and use them in two ways:

- Overlay (DOVE)
- OpenFlow

Next, we describe the high-level differences and interactions between the two methods.

Overlay (DOVE)

Overlay networks enable the design of services in which the communication section is decoupled from the underlaying physical networking infrastructure (for example, switches and routers). Basic network connectivity and (future) network services (security, compression, acceleration, and QoS) can benefit from this abstraction by simplifying how networking serves the IT infrastructure.

Overlay (DOVE) provides overlay technology that is used on a multiplatform environment that is contained in the edge of the network, which is inside the hypervisor virtual switches in our case. The overlay has special components at the edge that make it possible to interact with both traditional and OpenFlow enabled networks for data communications between

overlay-enabled endpoints and also to communicate with traditional physical appliances that are not overlay capable (for example, a physical firewall or an Internet service provider physical terminating device).

OpenFlow

OpenFlow is the first standard communications interface defined between the control and forwarding layers of an SDN architecture. OpenFlow allows direct access to and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual (hypervisor-based). It is the absence of an open interface to the forwarding plane that has led to the characterization of today's networking devices as monolithic, closed, and mainframe-like. No other standard protocol does what OpenFlow does, and a protocol such as OpenFlow is needed to move network control out of the networking switches to logically centralized control software.

OpenFlow can be compared to the instruction set of a CPU. As Figure A-4 shows, the protocol specifies basic primitives that can be used by an external software application to program the forwarding plane of network devices, just like the instruction set of a CPU would program a computer system.

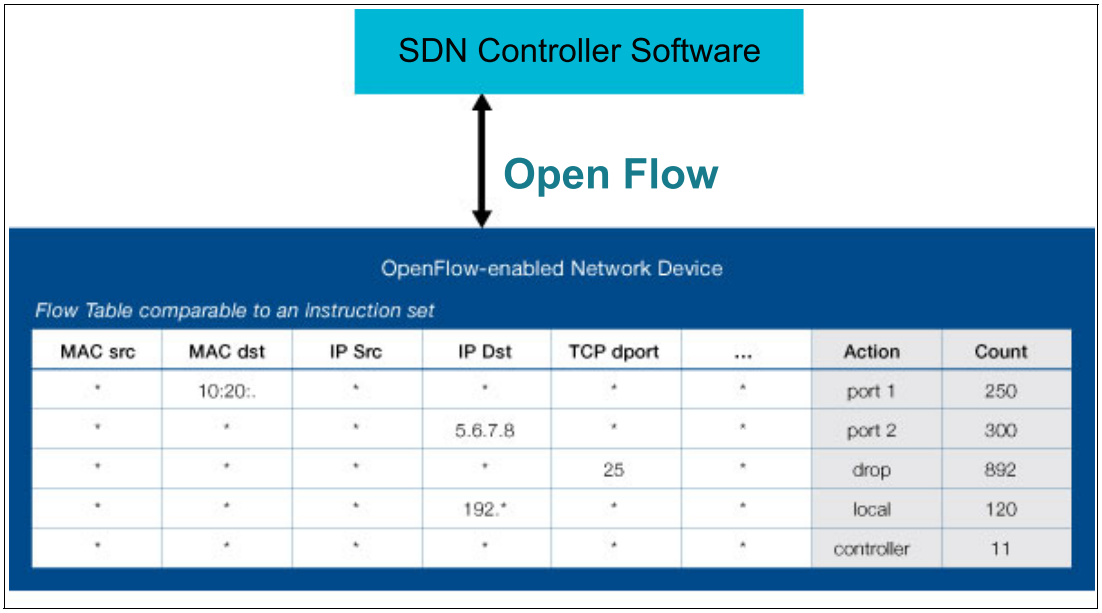


Figure A-4 Example of OpenFlow instruction set

The OpenFlow protocol is implemented on both sides of the interface between network infrastructure devices and the SDN control software. OpenFlow uses the concept of *flows* to identify network traffic based on predefined match rules that can be statically or dynamically programmed by the SDN control software. It also allows IT to define how traffic should flow through network devices based on parameters such as usage patterns, applications, and cloud resources. Because OpenFlow allows the network to be programmed on a per-flow basis, an OpenFlow-based SDN architecture provides extremely granular control, which enables the network to respond to real-time changes at the application, user, and session levels. Current IP-based routing does not provide this level of control, because all flows between two endpoints must follow the same path through the network regardless of their different requirements.

Software-defined networking with OpenFlow is a compelling way to build fast, agile, and intelligent networks. Data flow control is abstracted from static individual switches to dynamic programmable network-level control. Administrators can quickly create and control virtual networks for each application environment or network service. They can scale highly virtualized application infrastructures, multitenant networks on public or private clouds.

The Unified Controller provides an OpenFlow-based fabric with centralized control of network flows. The Unified Controller continuously manages and optimizes the OpenFlow network topology and makes it possible for traffic flows to use an OpenFlow-based data center network environment.

The OpenFlow protocol is a key enabler for software-defined networks and currently is the only standardized SDN protocol that allows direct manipulation of the forwarding plane of network devices. While initially applied to Ethernet-based networks, OpenFlow switching can extend to a much broader set of use cases. OpenFlow-based SDNs can be deployed on existing networks, both physical and virtual. Network devices can support OpenFlow-based forwarding as well as traditional forwarding, which makes it easy for enterprises and carriers to progressively introduce OpenFlow-based SDN technologies, even in multi-vendor network environments.

The Open Networking Foundation is chartered to standardize OpenFlow and does so through technical working groups responsible for the protocol, configuration, interoperability testing, and other activities, helping to ensure interoperability between network devices and control software from different vendors. OpenFlow is being widely adopted by infrastructure vendors, who typically have implemented it via a simple firmware or software upgrade. OpenFlow-based SDN architecture can integrate seamlessly with an enterprise or carrier's existing infrastructure and provide a simple migration path for those segments of the network that need SDN functions the most.

Benefits of OpenFlow-based Software Defined Networks

For enterprises and carriers alike, SDN makes it possible for the network to be a competitive differentiator, not just an unavoidable cost center. OpenFlow-based SDN technologies enable IT to address the high-bandwidth, dynamic nature of today’s applications, adapt the network to ever-changing business needs, and significantly reduce operations and management complexity.

These are among the benefits that enterprises and carriers can achieve through an OpenFlow-based SDN architecture:

- Centralized control of multi-vendor environments

  SDN control software can control any OpenFlow-enabled network device from any vendor, including switches, routers, and virtual switches. Rather than having to manage groups of devices from individual vendors, IT can use SDN-based orchestration and management tools to quickly deploy, configure, and update devices across the entire network.

- Reduced complexity through automation

  OpenFlow-based SDN offers a flexible network automation and management framework, which makes it possible to develop tools that automate many management tasks that are done manually today. These automation tools will reduce operational overhead, decrease network instability introduced by operator error, and support emerging IT-as-a-Service and self-service provisioning models. In addition, with SDN, cloud-based applications can be managed through intelligent orchestration and provisioning systems, further reducing operational overhead while increasing business agility.

- Higher rate of innovation

  SDN adoption accelerates business innovation by allowing IT network operators to literally program and reprogram the network in real time to meet specific business needs and user requirements as they arise. By virtualizing the network infrastructure and abstracting it from individual network services, for example, SDN and OpenFlow give IT—and potentially even users—the ability to tailor the behavior of the network and introduce new services and network capabilities in a matter of hours.

- Increased network reliability and security

  SDN makes it possible for IT to define high-level configuration and policy statements, which are then translated down to the infrastructure via OpenFlow. An OpenFlow-based SDN architecture eliminates the need to individually configure network devices each time an end point, service, or application is added or moved or a policy changess, which reduces the likelihood of network failures due to configuration or policy inconsistencies.

Because SDN controllers provide complete visibility and control over the network, they can ensure that access control, traffic engineering, quality of service, security, and other policies are enforced consistently across the wired and wireless network infrastructures, including branch offices, campuses, and data centers. Enterprises and carriers benefit from reduced operational expenses, more dynamic configuration capabilities, fewer errors, and consistent configuration and policy enforcement.

- More granular network control

  OpenFlow's flow-based control model allows IT to apply policies at a granular level, including the session, user, device, and application levels, in a highly abstracted, automated fashion. This control enables cloud operators to support multitenancy while maintaining traffic isolation, security, and elastic resource management when customers share the same infrastructure.

- Better user experience

  By centralizing network control and making state information available to higher-level applications, an SDN infrastructure can better adapt to dynamic user needs. For instance, a carrier could introduce a video service that offers premium subscribers the highest possible resolution in an automated and transparent manner. Today, users must explicitly select a resolution setting, which the network may or may not be able to support, resulting in delays and interruptions that degrade the user experience. With OpenFlow-based SDN, the video application would be able to detect the bandwidth available in the network in real time and automatically adjust the video resolution accordingly.

SDN reference architecture and standards

In this section, we describe the software-defined networking (SDN) reference architecture key foundation research projects and the foundations that defined SDN and Open Networking Standards.

SDN reference architecture

SDN separates the control plane from the data plane within the network, allowing the intelligence and state of the network to be managed centrally while abstracting the complexity of the underlying physical network. Great strides have been made within the industry toward this goal with standardized protocols such as OpenFlow.

However, greater collaboration using open source development best practices will significantly accelerate real, deployable solutions for the industry at large. Similarly, by evolving network services from an appliance model to one that uses virtual compute, storage, and networking, Network Functions Virtualization (NFV) promises to drastically improve both the agility of when and where to run network functions and the cost structure of doing so.

SDN and NFV are a new way of deploying network infrastructure. A software-defined network adapts to the requirements of applications deployed on the network. Current generation networks and architectures are statically configured and vertically integrated. New generation applications such as Hadoop, video delivery, and virtualized network functions require networks to be agile and to flexibly adapt to application requirements.

SDN is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services.

From a high-level view, software-defined networkingg is commonly described in layers. Figure A-5 depicts a logical view of the SDN architecture. Network intelligence is (logically) centralized in software-based SDN controllers, which maintain a global view of the network. As a result, the network appears to the applications and policy engines as a single, logical switch. With SDN, enterprises and carriers gain vendor-independent control over the entire network from a single logical point, which greatly simplifies the network design and operation. SDN also greatly simplifies the network devices themselves, because they no longer need to understand and process thousands of protocol standards but merely accept instructions from the SDN controllers.

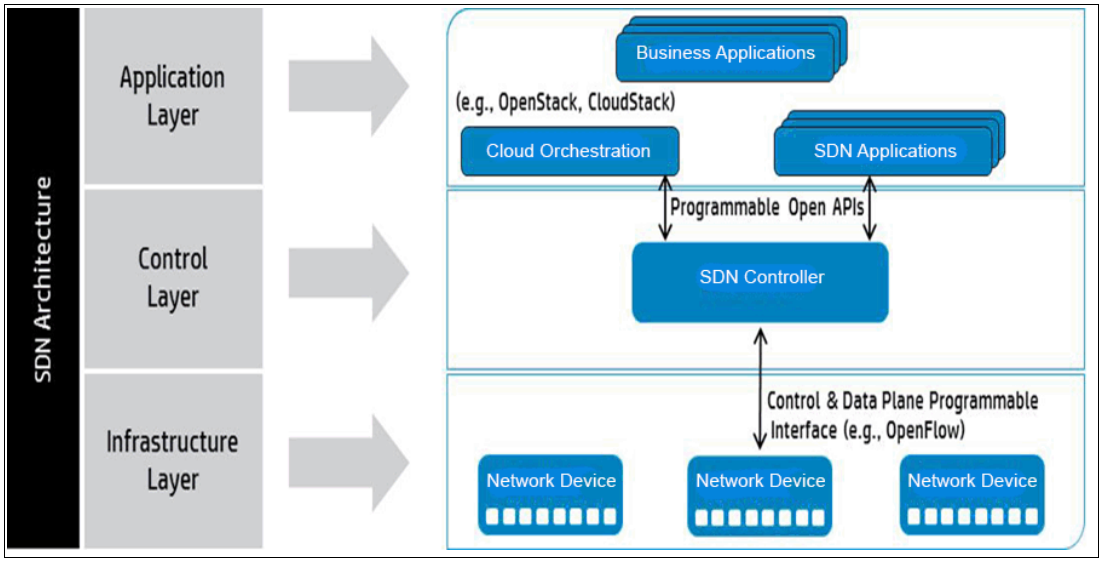


Figure A-5 SDN architecture

Perhaps most importantly, network operators and administrators can programmatically configure this simplified network abstraction rather than having to hand-code tens of thousands of lines of configuration scattered among thousands of devices. In addition, using the SDN controller's centralized intelligence, IT can alter network behavior in real-time and deploy new applications and network services in a matter of hours or days, rather than the weeks or months needed today. By centralizing network state in the control layer, SDN gives network managers the flexibility to configure, manage, secure, and optimize network resources via dynamic, automated SDN programs. Moreover, they can write these programs

themselves and not wait for features to be embedded in vendors' proprietary and closed software environments in the middle of the network.

In addition to abstracting the network, SDN architectures support a set of APIs that make it possible to implement common network services, including routing, multicast, security, access control, bandwidth management, traffic engineering, quality of service, processor and storage optimization, energy use, and all forms of policy management, custom tailored to meet business objectives. For example, an SDN architecture makes it easy to define and enforce consistent policies across both wired and wireless connections on a campus.

Likewise, SDN makes it possible to manage the entire network through intelligent orchestration and provisioning systems. The Open Networking Foundation is studying open APIs to promote multi-vendor management, which opens the door for on-demand resource allocation, self-service provisioning, truly virtualized networking, and secure cloud services.

Therefore, with open APIs between the SDN control and applications layers, business applications can operate on an abstraction of the network, using network services and capabilities without being tied to the details of their implementation. SDN makes the network not so much “application-aware” as “application-customized” and applications not so much “network-aware” as “network-capability-aware”. As a result, computing, storage, and network resources can be optimized.

The Open Networking Foundation

The Open Networking Foundation, established in 2011, is a nonprofit, mutually beneficial trade organization, founded by Deutsche Telekom, Facebook, Google, Microsoft, Verizon, and Yahoo! to improve networking through SDN and standardizing the OpenFlow protocol and related technologies. [2]The standards-setting and SDN-promotion group was formed out of recognition that cloud computing will blur the distinctions between computers and networks. The[3] initiative is meant to speed innovation through simple software changes in telecommunications networks, wireless networks, data centers and other networking areas.

[4]As of December 31, 2013, the organization had 123 member companies; a full list of members may be viewed on the organization's membership page. Member companies include many networking-equipment vendors, semiconductor companies, computer companies, software companies, telecom service providers, hyperscale data center operators, and enterprise users. Google’s system-wide adoption of ONF’s OpenFlow software in 2012 was discussed by Urs Hölzle at the Open Networking Summit in April, 2012.[5] Hölzle is Chairman and President of ONF's board of directors, serving on the Board along with representatives of the other five founding board members plus NTT Communications Corp and Goldman Sachs; Stanford professor Nick McKeown and U.C. Berkeley professor Scott Shenker also serve on the board as founding directors representing themselves.

By[6] June 2014 ONF had grown to more than 150 member companies, including 24 start-up companies in software-defined networking.[7]

For more information, see the Open Networking Foundation website:

---

[2] Erickson, D. Open Networking Foundation Formed to Speed Network Innovation. OpenFlow Blog, 21 March 2011 http://archive.openflow.org/wp/2011/03/open-networking-foundation-formed-to-speed-network-innovation/

[3] Markoff, John. Open Networking Foundation Pursues New Standards. New York Times, 22 March 2011. http://www.nytimes.com/2011/03/22/technology/internet/22internet.html?_r=0

[4] Noyes, Catherine. Google and other titans form Open Networking Foundation. *PC World*, 22 March 2011. http://www.pcworld.com/article/222856/google_and_other_titans_form_open_networking_foundation.html

[5] Levy, Steven. “Going with the Flow: Google’s Secret Switch to the Next Wave of Networking.” *Wired*, April 17, 2012. http://www.wired.com/2012/04/going-with-the-flow-google/

[6] Pitt, Dan. ONF overview, ONF website. https://www.opennetworking.org/about/onf-overview

[7] “SDN Consortium Adds 24 Startups to Its Membership List.” *eWeek,* 18 June 2014. http://bit.ly/1GqPiqQ

https://www.opennetworking.org/about/onf-overview

OpenDaylight project

The adoption of new technologies and pursuit of programmable networks has the potential to significantly improve levels of functionality, flexibility, and adaptability of mainstream data center architectures. To use this abstraction to its fullest requires the network to adapt and evolve to a Software-Defined architecture. One of the architectural elements required to achieve this goal is a SDN and NFV platform that enables network control and programmability.

Today, SDN and NFV are a hotbed of innovation with a broad spectrum of vendors bringing products and technologies to market. Ironically, the myriad options may prove counterproductive to SDN and NFV adoption. Many organizations, overwhelmed by choice, also must avoid the pitfalls of nascent technologies such as inconsistent interoperability, limited management tools or fragmentation.

At this early stage of SDN and NFV adoption, the industry acknowledges the benefits of establishing an open, reference framework for programmability and control through an open source SDN and NFV solution. Such a framework maintains the flexibility and choice to allow organizations to deploy SDN and NFV as they please, yet still mitigates many of the risks of adopting early stage technologies and integrating with existing infrastructure investments.

With OpenDaylight, a community has come together to fill this need through the combination of open community developers and open source code and project governance that provides an open, community decision-making process on business and technical issues. Establishing an open source project in this way is designed to help accelerate the development of technology available to users and enable widespread adoption of SDN and create a solid foundation for NFV.

OpenDaylight can be a core component within any SDN architecture. Building upon an open source SDN and NFV controller enables users to reduce operational complexity, extend the life of their existing infrastructure hardware and enable new services and capabilities only available with SDN. Whether your organization is an enterprise IT provider, a network service provider or a Cloud services provider, you can begin taking advantage of SDN and NFV using a community-driven, open source controller framework available today.

To facilitate an open ecosystem, OpenDaylight was structured with an open source license, open design, development and contribution model and finally, an, appropriate, open governance model. The project is a Linux Foundation Collaborative project and implements many open source best practices familiar to other leading projects. Further, any individual, company or organization can engage directly and immediately to begin shaping the future of the project.

OpenDaylight is an open platform for network programmability to enable SDN and NFV for networks at any size and scale. The community's second release "Helium" comes with a new user interface and a much simpler and customizable installation process thanks to the use of the Apache Karaf container. The second open source software release "Helium" is available for download. Testing and integration are still underway for Helium. Figure A-6 on page 727 illustrates the Helium release in context to the SDN Architecture.

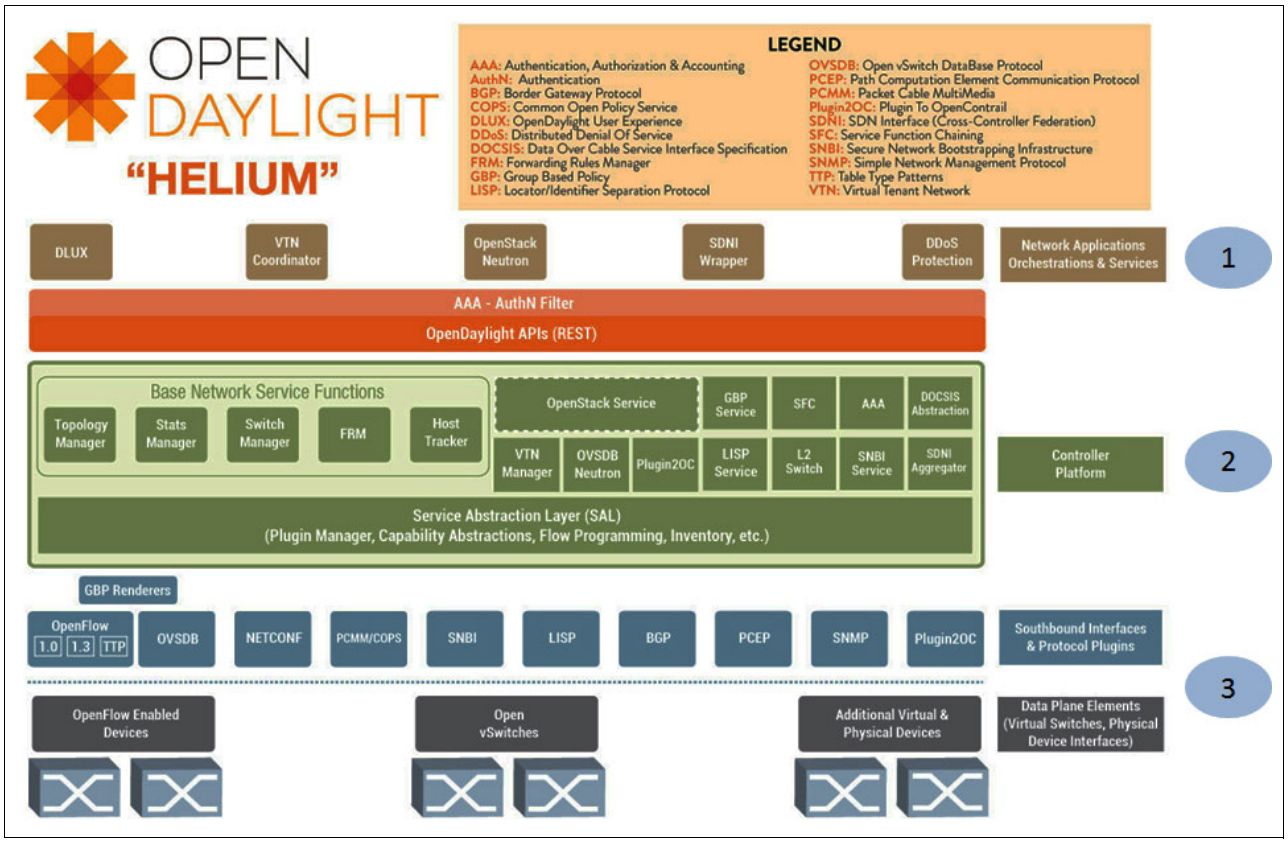


Figure A-6 Open Daylight reference architecture

The description shown in Figure A-6 are as follows:

1. Network Apps and Orchestration. The top layer consists of business and network logic applications that control and monitor network behavior. In addition, more complex solution orchestration applications needed for cloud and NFV thread services together and engineer network traffic in accordance with the needs of those environments.

2. Controller Platform. The middle layer is the framework in which the SDN abstractions can manifest, providing a set of common APIs to the application layer (commonly referred to as the northbound interface) while implementing one or more protocols for command and control of the physical hardware within the network (typically referred to as the southbound interface).

3. Physical and Virtual Network Devices. The bottom layer consists of the physical and virtual devices, switches, routers, and so on, that make up the connective fabric between all endpoints within the network.

OpenDaylight is an open source project with a modular, pluggable, and flexible controller platform at its core. This controller is implemented strictly in software and is contained within its own Java virtual machine (JVM). As such, it can be deployed on any hardware and operating system platform that supports Java.

The controller exposes open northbound APIs, which are used by applications. OpenDaylight supports the OSGi framework and bidirectional REST for the northbound API. The OSGi framework is used for applications that will run in the same address space as the controller while the REST (web-based) API is used for applications that do not run in the same address space (or even necessarily on the same machine) as the controller. The business logic and algorithms reside in the applications.

These applications use the controller to gather network intelligence, run algorithms to perform analytics, and then use the controller to orchestrate the new rules, if any, throughout the network.

The controller platform itself contains a collection of dynamically pluggable modules to perform needed network tasks. There are a series of base network services for such tasks as understanding what devices are contained within the network and the capabilities of each, statistics gathering, and so on. In addition, platform-oriented services and other extensions can also be inserted into the controller platform for enhanced SDN functions.

The southbound interface is capable of supporting multiple protocols (as separate plug-ins), for example, OpenFlow 1.0, OpenFlow 1.3, BGP-LS, and so on. These modules are dynamically linked into a Service Abstraction Layer (SAL). The SAL exposes device services to which the modules north of it are written. The SAL determines how to fulfill the requested service irrespective of the underlying protocol used between the controller and the network devices.

OpenDaylight is key to an SDE. As data centers struggle to meet the demands of today's business, there is an increasing need for an SDE that brings together the trends in computing, storage, and networking into a common set of infrastructure attributes. IBM intends to build on the OpenDaylight platform to offer a complete solution for SDN, which is a key building block of the IBM SDE strategy.

For a list of the members of the OpenDaylight project, visit the following link:

`http://www.opendaylight.org/project/members`

IBM co-developed the technical blueprint (Dixon-Erickson Proposal) for the SDN base controller project based on code from multiple contributors. This is now the plan of record for the controller architecture.

IBM has contributed the following to OpenDaylight:

- Open DOVE, which is an open source version of SDN VE network virtualization
- OpenFlow Load Balancing, integrated with the base controller
- Controller enhancements for performance, scalability, and improved routing and forwarding services

The OpenDaylight project is ever evolving based on the contributions of its developer community. For additional documentation on the different components that make up the OpenDaylight project, visit the Developer WIKI:

`https://wiki.opendaylight.org/view/Main_Page`

For more information about the OpenDaylight project:

`http://www.opendaylight.org/`

OpenDaylight and OpenStack Integration

OpenDaylight simply shows a single common northbound OpenStack service. The API exposed precisely matches the Neutron API. There are multiple implementations of the Neutron networks in the OpenDaylight project. The OpenStack plug-in is greatly simplified, and the complexity is pushed to OpenDaylight through a pass-through API.

Figure A-7 illustrates the high-level integration.

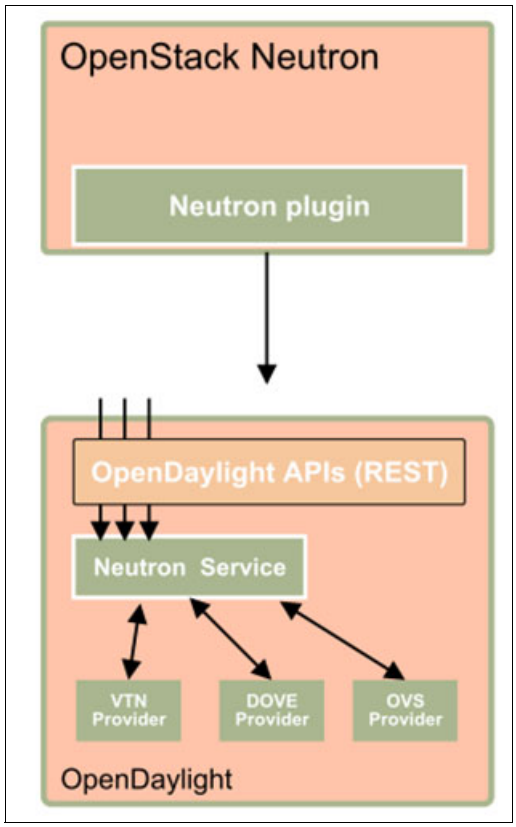


Figure A-7 OpenStack and OpenDaylight integration

Based on the layered reference architecture model for SDN, the applied integration between SDN and OpenStack is shown in Figure A-8.

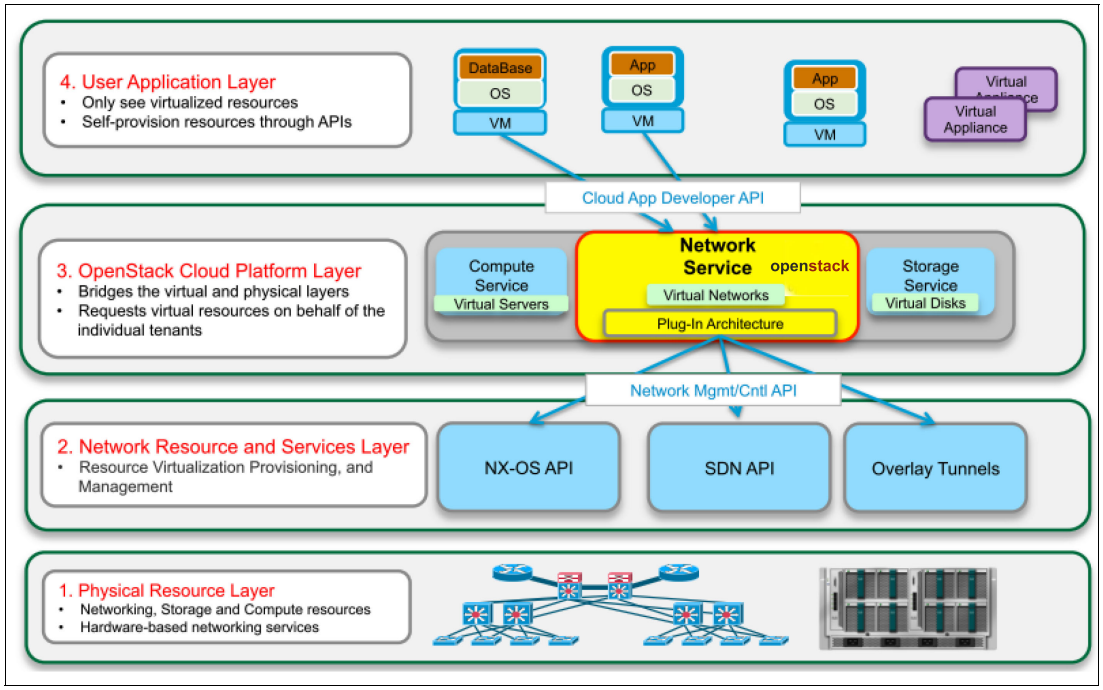


Figure A-8 SDN and OpenStack

Pulling this altogether in an integrated reference architectural view with SDI provides and extends the benefits of the SDE environment. The complete integrated reference architecture is shown in Figure A-9, which includes the systems of engagement, systems of record, SDI domain based on OpenStack, SDN with OpenDaylight and OpenFlow, DOVE Overlay server virtual switches, and the heterogeneous physical infrastructure.

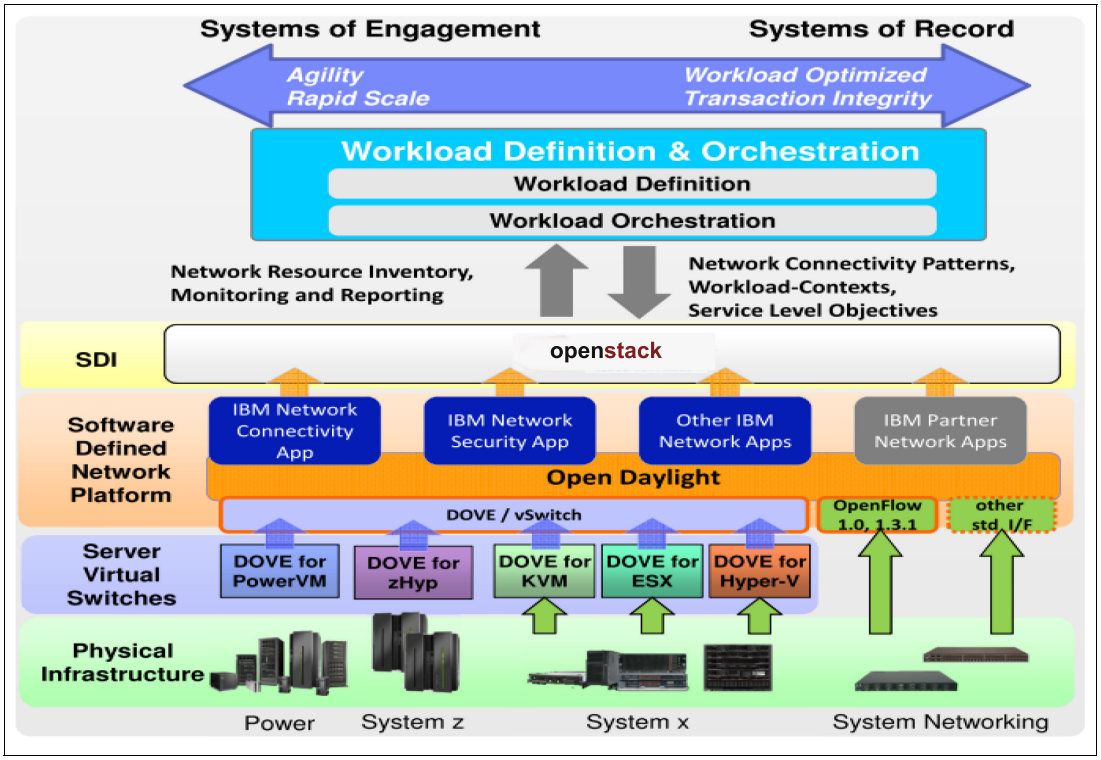


Figure A-9 Integrated SDN and SDI reference architecture

Other vendor solution products and implementation frameworks

For a complete view of the SDN and NFV Products and Services, see the following:

https://www.sdncentral.com/nfv-sdn-products-directory/

IBM Software Defined Network for Virtual Environments context

This section describes the IBM Software Defined Network for Virtual Environments (SDN VE) solution.

Introduction to IBM SDN VE

SDN VE supplies a complete implementation framework for network virtualization. IBM SDN VE is a distributed overlay virtual network software solution suite that provides a virtualized form of a physical network without requiring any changes to the real physical network itself. The IBM SDN for Virtual Environments architecture abstracts the underlying network for the virtual environment and presents it as a service or as an infrastructure in the form of an overlay network. SDN VE components provide network virtualization within the IBM SDN platform irrespective of the physical network.

SDN VE creates a more flexible network by creating a completely virtualized network for virtual machines, without requiring any type of multicast to be enabled on the physical network. This virtual network is de-coupled and isolated from the physical network much like a virtual machine is de-coupled and isolated from its host server hardware. SDN VE takes a host-based overlay approach, which achieves advanced network abstraction that enables application-level network services in large-scale multitenant environments. It provides a multi-hypervisor, server-centric solution comprising multiple components that overlay virtual networks onto any physical network that provides IP connectivity.

Figure A-10 shows a high-level overview of the various components in the SDN VE solution.

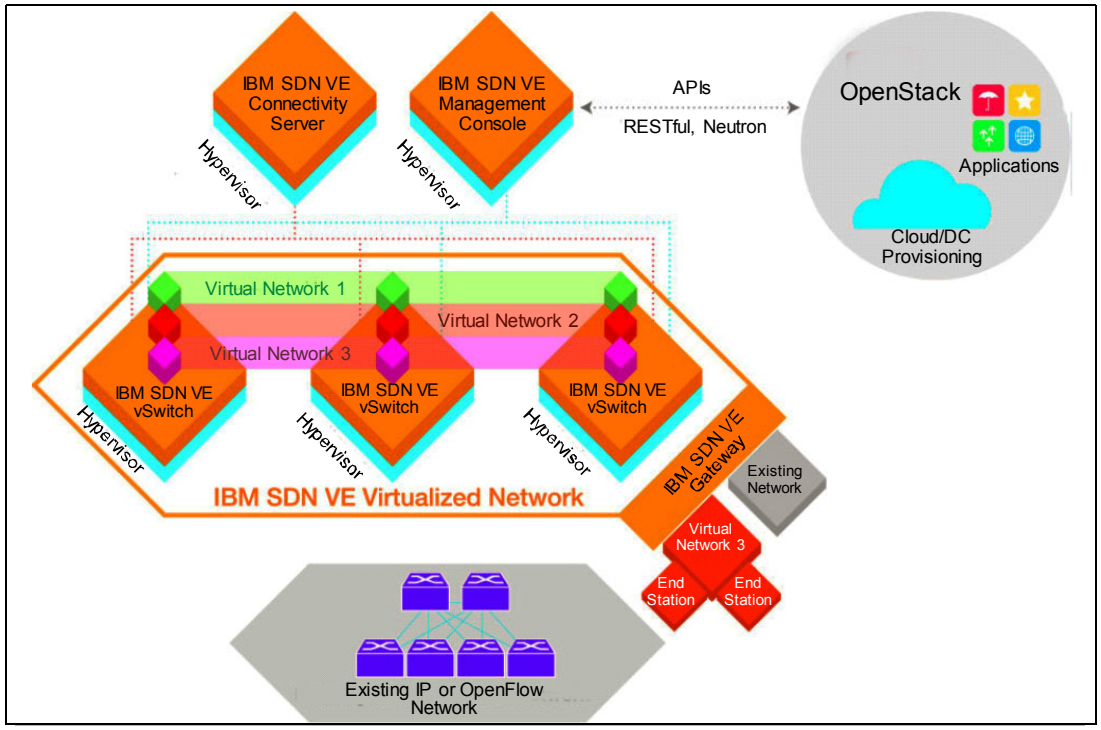


Figure A-10 IBM SDN VE overview

Product versions

The IBM SDN VE product suite includes the following versions:

- IBM Software Defined Network for Virtual Environments VMware Edition

  This package represents the first release of the SDN VE solution, announced on March 26, 2013. As the name implies, this version is specifically targeted at running in conjunction with the vSphere hypervisor that is published by VMware, Inc. VMware vSphere is a closed-source, commercial, highly proliferated x86 hypervisor platform that enjoys worldwide industry adaptation, and is currently the market leader in the x86 virtualization space at the time of this writing.

- IBM Software Defined Network for Virtual Environments KVM Edition

  This product version represents the adaptation of the IBM SDN VE VMware Edition components (minus the SDN VE 5000V, or the IBM SDN VE 5000V Distributed vSwitch for VMware vSphere, as abstracted in Figure A-2 on page 716) for the Kernel-based virtual machine hypervisor, or KVM.

KVM is an open source hypervisor that provides enterprise-class performance, scalability and security to run Windows and Linux workloads. KVM provides organizations a cost-effective alternative to other x86 hypervisors, and enables a lower cost, more scalable, and open cloud.

- IBM Software Defined Network for Virtual Environments OpenFlow Edition

  This product version is focused on the providing Unified Controller integration with support for version 1.0 of the OpenFlow communications protocol. OpenFlow is an emerging industry standard protocol that moves the network control plane into software running on an attached server. The flow of network traffic can then be controlled dynamically, without the need to rewire the data center network. OpenFlow also offers a global view of the network, including traffic statistics, and is fully compatible with existing Layer 2 and 3 protocols. In contrast to a traditional switch, which provides a separate management/control plane for each switch element in the network, OpenFlow extracts the control plane from the network. Some of the benefits of this approach include better scalability, larger layer 2 domains and virtual devices, faster convergence, and better scalability.

Elements of the IBM SDN VE solution

SDN VE implements the principles and vision of software-defined networking through several discrete entities that we refer to hereafter as the *components*. These components emulate the traditional control and data forwarding planes of standard physical switches for the virtual network overlay environment, and are critical pieces of the solution.

Figure A-11 on page 733, shows a logical representation of the SDN VE components and how they fit into the existing data center physical network, referred to as the *Underlay Network*. The *DOVE Overlay* network represented in the center of the picture is the cornerstone of the product, and is used throughout this book to showcase the capabilities and benefits of using SDN VE. This virtual network transparently permits VM to VM communication throughout the infrastructure, and requires no changes to the existing underlay network that it rides over.

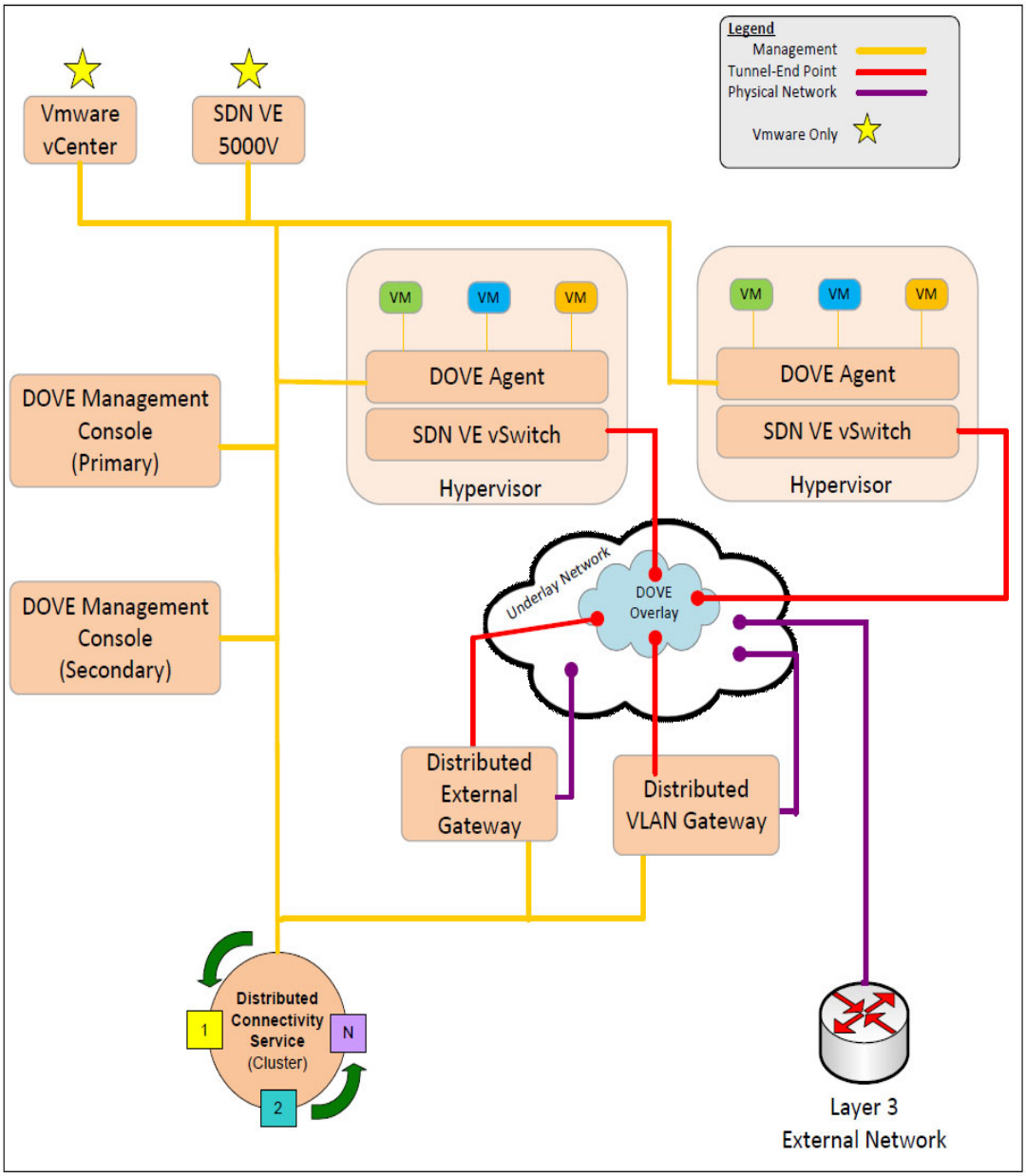


Figure A-11 IBM SDN VE components diagram

SDN VE uses the industry standard Virtual Extensible local area network (VXLAN) frame format. The control plane for a virtual network implementation is not yet standardized. Therefore, SDN VE has a proprietary control plane as the core of DOVE technology.

SDN VE components

Regardless of the hypervisor platform that is used by the customer, the SDN VE components represent one of the building blocks of the virtualized network environment. These components provide similar functions in both VMware and KVM, with a few exceptions where noted.

The SDN VE solution is made up of four software components that work in combination to provide effective host-based network virtualization.

Dove Management Console (DMC)

A management console is the centralized point of control for configuring SDN VE. It configures each virtual network, controls policies, and disseminates policies to the virtual switches. It also helps administrators manage individual virtual networks. The software resides on a server as a virtual appliance.

Note: The DMC is not the same as the Unified Controller. The Unified Controller is an management interface that also includes the DMC, among other things. The DMC itself is the controlling entity for the SDN VE deployment.

Distributed connectivity service (DCS)

A connectivity service disseminates VM addresses to the virtual switches participating in an SDN VE virtual network. The connectivity server configures the virtual network, controls policies, and disseminates policies to the virtual switches. The DCS software is deployed as a cluster of virtual appliances, and offers more scalability and reliability of the traditional network control plane.

DOVE gateways (DGW)

DOVE gateways are specialized appliances that connect the SDN VE overlay network environment with a non-SDN VE environment. This includes data connectivity between SDN VE virtual machines to the public network, and the interaction between the SDN VE environment and existing hardware and software entities. Examples of these resources include physical servers, management tools, and network appliances that are not able to be migrated into the SDN VE environment.

Two gateways are provided: Distributed VLAN gateways and distributed external gateways. Both are packaged as and also referred to as a *distributed services appliance* (DSA).

Distributed VLAN gateways:

- Enable VMs in an SDN VE domain to connect with networks and servers that are external to the overlay network from a layer 2 (VLAN) perspective.

Distributed external gateways:

- Enable VMs in an SDN VE domain to connect to non-SDN VE or DOVE external systems.
- Enable VMs in an SDN VE domain to connect to SDN VE VMs in another domain through policy allocations.
- Enable external systems to connect to VMs inside SDN VE or DOVE domain.

Hardware switch-based gateways are to be released in the future.

5000V Host Module

The 5000V Host Module is software that resides in the VMware hypervisor. It serves as the start and end point of each virtual network. The 5000V Host Module provides Layer 2 and Layer 3 network virtualization over a UDP VXLAN overlay, and implements the data path of the virtual network. The virtual switch also performs control plane functions to support virtual machine (VM) address auto discovery, VM migration, and network policy configuration. KVM does not use the 5000V Host Module, but rather a DOVE agent process that runs on each individual hypervisor and listens for direction from the Distributed Connectivity Service. The DOVE agents are also responsible for creating the individual bridged interfaces that ultimately show up in the virtual machine Manager or virt-manager.

SDN VE 5000V Distributed vSwitch for VMware vSphere (SDN VE 5000V)

For VMware, the SDN VE 5000V Distributed vSwitch for VMware vSphere (SDN VE 5000V) virtual appliance is used to aggregate the individual 5000V Host Modules and serves as the responsible entity for communication with the VMware vSphere vCenter server.

KVM does not use a distributed vSwitch Controller component as a part of its software suite, but instead uses the DOVE agents that are installed on the hypervisor to communicate with the connectivity services directly through its own vSwitch, named the SDN VE vSwitch.

SDN VE system and solution context

We now describe the system context related to the SDN VE solution. The purpose of the system context document is to clarify and confirm the environment in which the system has to operate to verify that the information flows between the solution to be installed and external entities are in agreement with any business process or context diagrams.

This section describes the possible interactions between SDN VE and the rest of the IT infrastructure. It tries to answer these two questions:

- How does an SDN VE infrastructure integrate into an existing IT environment?
- What external IT services will my solution interface with and which will it be dependent upon?

Figure A-12 contains a graphical representation of the system context in which an SDN VE infrastructure can be deployed.

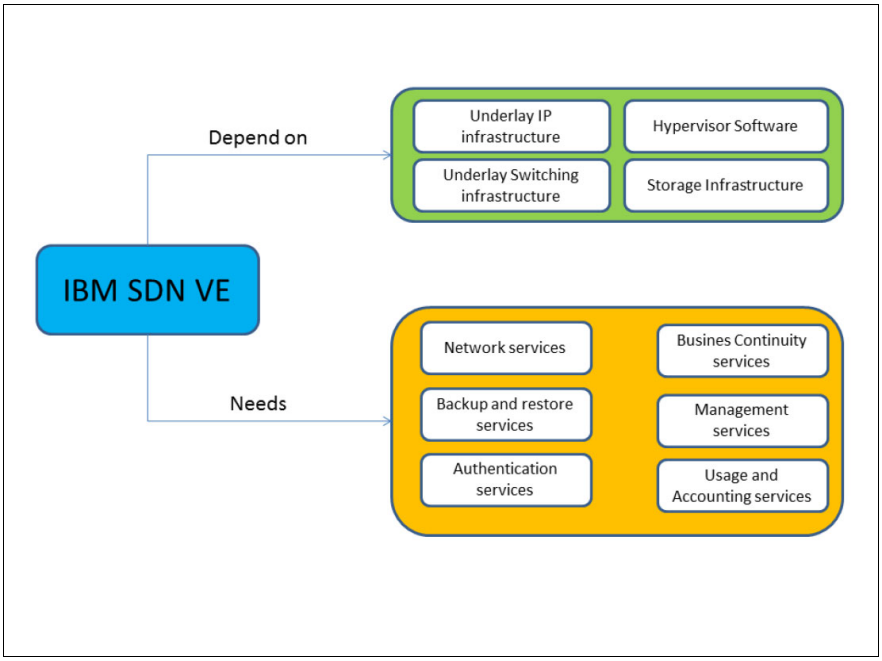


Figure A-12 SDN VE system context

The SDN VE solution depends on the following components for the SDN VE solution to function. These components are based on a combination of hardware devices and firmware or software components.

Underlay switching infrastructure

The switching infrastructure is made of cables and switches that depend on the environment (for example, core, distribution, and access approach). The switching infrastructure is used by

the SDN VE to access other systems on the local area network and to provide access to its guests to the enterprise network. Special configuration is required on physical network equipment for the SDN VE 5000V Host Module to work corectly. While typical physical servers are connected to the access or aggregate layer with ports in access mode, the 5000V Host Module can also be connected to 802.1Q trunk ports that carry multiple VLANs. Also the Distributed VLAN Gateway heavily depends on the switching infrastructure because it operates at layer 2 when interfacing the underlay. Additionally, IBM System Networking switches are capable of enhanced features such as 802.1Qbg. These types of standards require attention in the design and collaboration between the virtualization administrator and the network administrator.

Underlay IP infrastructure

The IP infrastructure is made of all the features, functions and devices that, again, depend on the environment (for example, routers and firewalls). The IP infrastructure is used by the SDN VE solution to communicate with both the control and management plane and to send production traffic both between 5000V Host Modules (for VM to VM flows) and, for example, between 5000V Host Modules and Distributed Gateways (for flows between VMs and the outside world). All the communication relies on IP (including, for example, VRRP on the DMC).

Storage infrastructure

This infrastructure is used by the SDN VE solution to store significant data (examples are installation files, guest virtual disks or the fault tolerance-related files for the high availability requirements of the SDN VE 5000V).

We now review the components where the SDN VE solution might need to provide essential services for overlay infrastructure to operate as expected. We want to highlight the concept that the SDN VE solution does not depend on these components and might function corectly even without them. We are referencing them to help you decide whether these components are needed in your environment and, if they are, to plan for this interaction. For instance, if you need to measure Operation Level Agreement (OLA), you might want to consider “management services” and how the SDN VE will interact with an existing environment.

Network services

There is no real network service that is absolutely mandatory for correct operation of the SDN VE infrastructure. Examples of services that the SDN VE solution can interact with are:

- Name resolution services such as DNS services
- Remote access to the virtual infrastructure (for example, remote desktop services to access the VMware infrastructure)
- DHCP servers used for optional service appliance IP address provisioning

Business continuity services

Some environments might require the SDN VE solution to be recovered at an alternate site. The business continuity server plan is usually based on a recovery time objective (RTO), a recovery point objective (RPO), or both.

Backup and restore services

A plan might be needed in case an SDN VE component becomes unavailable or needs to be recovered after a change management activity or else (for example, a software upgrade).

Authentication services

Local authentication is supported for the system administrators to manage the SDN VE components. External authentication, such as RADIUS, can be used on the SDN VE 5000V.

With external authentication, features such as password format or time-based password change can be enforced.

Management services

The management services can include several distinct domains. In general, Software Defined Environment solutions complement the following management services for the SDN VE infrastructure:

- Performance

  This might be a wanted external service. Software Defined Environments can provide this service to the SDN VE infrastructure. Store performance data and integration with the performance management reporting are only few aspects that you might want to consider. For example, while the DMC provides performance counters, it only provides raw data which must be analyzed and integrated in reports so that it can be used or presented.

- Capacity planning

  The virtual infrastructure relies on capacity planning services to determine if there are enough resources to meet the demand. Although the SDN VE solution has mechanisms in place to provide availability and performance to its managed workloads, the solution has no visibility on the future demands of the business. Again, Software Defined Environments can provide this service to the SDN VE infrastructure.

- Security compliance

  Security compliance depends on the security policies and guidelines that are in place in the environment. SDE can be used to continuously check the SDN VE infrastructure for compliance with technical security policies. Without it, the infrastructure might become exposed to known security flaws and susceptible to denial of service or other security threats.

  Security compliance services might also require additional services such as log collection services (for example, from the DMC) and user account management.

- Asset management

  Asset management can collect information from the SDN VE infrastructure to be put into the configuration management database or any other repository. The components can move, increase, or decrease (for example, Distributed Gateways) inside the dynamic virtualized infrastructure. It is important that asset information is collected periodically so that automated mechanisms can be used. A comprehensive SDE strategy can also support this service.

- Event management

  The event management services monitor the health of the SDN VE infrastructure, alert when components are not functioning corectly or when performance metrics have been breached, and sometimes even take automated action to try to remediate the condition without human intervention.

- Usage and accounting services

Usage and accounting services might be needed to generate reports for the SDN VE resource usage. They can also be used to collect and correlate information, for example, per tenant or per department. For SDN VE, these services can also be included in a broader Software Defined Environment solution.

For more information about the SDN VE solution, see the following IBM Redbooks:

Implementing IBM Software Defined Network for Virtual Environments, SG24-8203:

`http://www.redbooks.ibm.com/abstracts/sg248203.html`

In the next section, we will cover some of the most popular SDN (including SDN VE) use cases.

IBM SDN scenarios and use cases

The Open Networking Foundation is guided by prominent enterprises and service providers, systems and applications developers, software and computer companies, and semiconductor and networking vendors. This diverse cross-section of the communications and computing industries is helping to ensure that SDN and associated standards effectively address the needs of network operators in each segment of the marketplace, including[8] these examples:

The enterprise

Campus SDN's centralized, automated control and provisioning model supports the convergence of data, voice, and video as well as anytime, anywhere access by enabling IT to enforce policies consistently across both the wired and wireless infrastructures. Likewise, SDN supports automated provisioning and management of network resources, determined by individual user profiles and application requirements, to ensure an optimal user experience within the enterprise's constraints.

Data center The SDN architectures facilitates network virtualization, which enables hyper-scalability in the data center, automated VM migration, tighter integration with storage, better server use, lower energy use, and bandwidth optimization.

Cloud Whether used to support a private or hybrid cloud environment, SDN allows network resources to be allocated in a highly elastic way, enabling rapid provisioning of cloud services and more flexible hand-off to the external cloud provider. With tools to safely manage their virtual networks, enterprises and business units will trust cloud services more and more.

Carriers and service providers

SDN offers carriers, public cloud operators, and other service providers the scalability and automation necessary to implement a utility computing model for IT-as-a-Service, by simplifying the roll-out of custom and on-demand services, along with migration to a self-service paradigm. SDN's centralized, automated control and provisioning model makes it much easier to support multitenancy; to ensure network resources are optimally deployed; to reduce both CapEx and OpEx; and to increase service velocity and value.

[8] https://www.opennetworking.org/sdn-resources/sdn-library/whitepapers

Table A-2 is a detailed table that provides more information about the most popular SDN scenarios and use cases.

Table A-2 Often implemented SDN scenarios and use cases

Use case	Location	Why SDN?	Benefits achieved
Network Virtualization - Multi-Tenant Networks	Data center	To dynamically create segregated topologically-equivalent networks across a data center, scaling beyond typical limits of VLANs today at 4K.	Better use of data center resources, claimed 20-30% better use of resources. Faster turnaround times in creating segregated network, from weeks to minutes via automation APIs.
Network Virtualization - Stretched Networks	Data center	To create location-agnostic networks, across racks or across data centers, with VM mobility and dynamic reallocation of resources.	Simplified applications that can be made more resilient without complicated coding, better use of resources as VMs are transparently moved to consolidate workloads. Improved recovery times in disasters.
Service Insertion (or Service Chaining)	Data center, service provider DMZ or WAN	To create dynamic chains of L4-7 services on a per tenant basis to accommodate self-service L4-7 service selection or policy-based L4-7 (for example, turning on DDoS protection in response to attacks, self-service firewall, intrusion prevention services in hosting environments, deep packet inspection in mobile WAN environments).	Provisioning times reduced from weeks to minutes, improved agility and self-service allows for new revenue and service opportunities with substantially lower costs to service.

Use case	Location	Why SDN?	Benefits achieved
Tap Aggregation	Data center or campus access networks	Provide visibility and troubleshooting capabilities on any port in a multi-switch deployment without use of numerous expensive network packet brokers (NPB).	Dramatic savings and cost reduction, savings of $50-100K per 24 to 48 switches in the infrastructure. Less overhead in initial deployment, reducing need to run extra cables from NPBs to every switch.
Dynamic WAN reroute –move large amounts of trusted data bypassing expensive inspection devices	Service provider or enterprise edge	Provide dynamic yet authenticated programmable access to flow-level bypass using APIs to network switches and routers.	Savings of hundreds of thousands of dollars of unnecessary investment in 10Gbps or 100Gbps L4-7 firewalls, load-balancers, IPS/IDS that process unnecessary traffic.
Dynamic WAN interconnects	Service provider	To create dynamic interconnects at Internet interchanges between enterprise links or between service providers using cost-effective high-performanc e switches.	Ability to instantly connect Reduces the operational expense in creating cross-organization interconnects, providing ability to enable self-service.
Bandwidth on Demand	Service provider	Enable programmatic controls on carrier links to request extra bandwidth when needed (for example, DR, backups).	Reduced operational expense allowing self-service by customers and increased agility saving weeks of manual provisioning.

Use case	Location	Why SDN?	Benefits achieved
Virtual Edge - Residential and Business	Service Provider Access Networks	In combination with NFV initiatives, replace existing Customer Premises Equipment (CPE) at residences and businesses with lightweight versions, moving common functions and complex traffic handling into POP (points-of-presence) or SP data center.	Increased usable lifespan of on-premises equipment, improved troubleshooting, fewer truck rolls, flexibility to sell new services to business and residential customers.

For more information about the following use cases and scenarios:

- VM mobility and relocation
- Data center V2V
- Firewall and DNS in multiple domains
- Multitenancy

Please see the IBM Redbooks *Implementing IBM Software Defined Network for Virtual Environments*, SG24-8203:

http://www.redbooks.ibm.com/abstracts/sg248203.html

In the next section, we describe the SDN installation process of the SDN VE and DOVE components.

SDN installation process of the SDN VE and DOVE components

This section describes the SDN installation process of the SDN VE and DOVE components.[9]

Introduction

This section is intended to provide a step-by-step installation process of the SDN VE and DOVE components required to setup and SDN environment. It describes the steps to install and configure the SDN VE environment and also configure VLAN overlays using this technology.

As mentioned earlier, there are four major management components that make up SDN VE:

- Dove Management Console (DMC)

  The DMC is the centralized point of control for configuring the SDN VE.

- Distributed Connectivity Service (DCS)

  The DCS provides VM addresses to the virtual switches in the SDN VE environment.

[9] Special thanks to John Simpson and Jamie Kuchler - IBM who provided the content for this section

- Distributed External Gateway (DGW)

  The DGW connect the SDN VE overlay network with a non-SDN VE environment.

- Dove Agent

  The DOVE agent is a process that communicates with the DCS directly through its own SDN VE vSwitch.

In the following sections, we document in detail how to install and configure each of these components to implement SDN VE.

Install and configure the Dove Management Console

The basis for the SDN VE GUI interface is the DMC. SDN- VE requires that you install two DMCs for the HA capabilities.

Software levels

This section shows the software levels required, and the installation process:

1. Copy the Unified Controller kvm file to the hypervisor where you will install the DMC.
2. Extract the Unified controller file.
3. Copy the qcow2 files over to the /var/lib/libvirt/images directory as shown in Figure A-13.

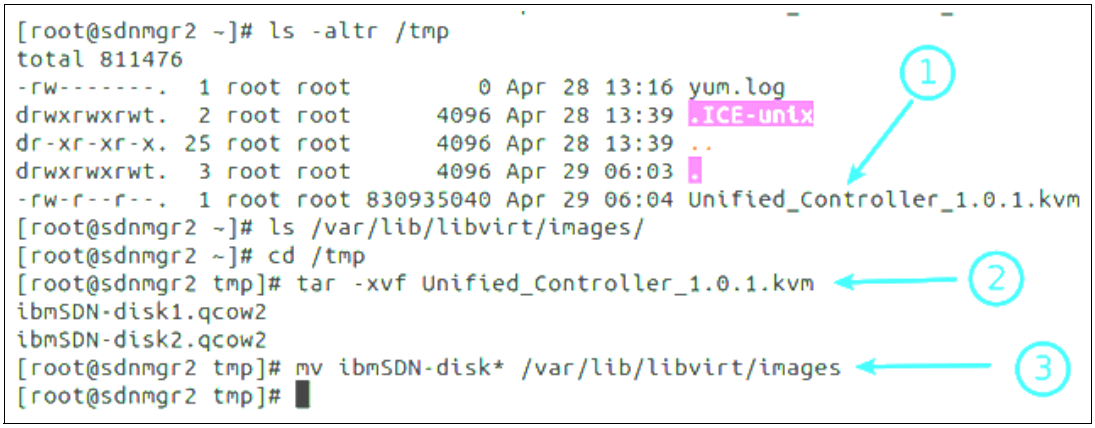


Figure A-13 Install and configure DMC

4. Create a new virtual machine for this hypervisor as shown in Figure A-14 on page 743. Provide the name and select **Import existing disk image** and then **Forward**.

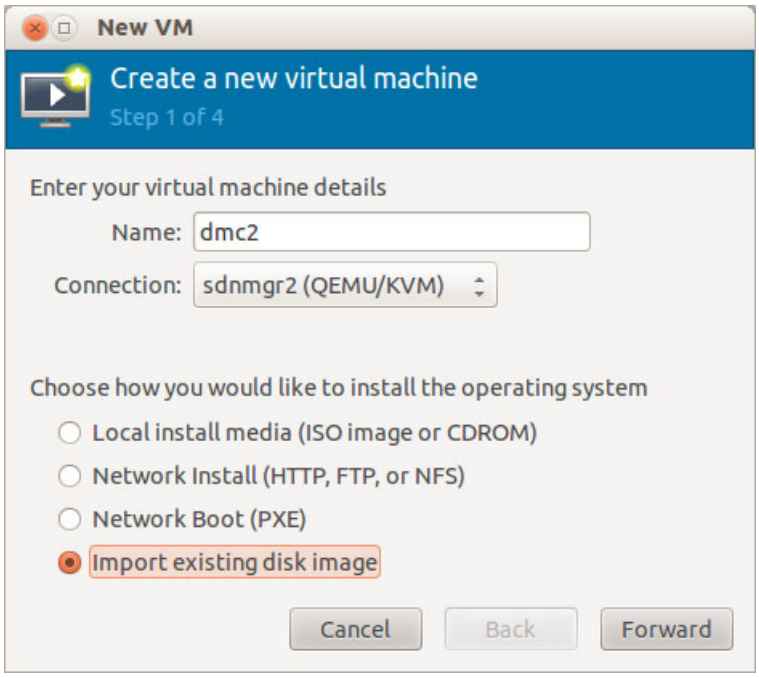


Figure A-14 Create a new virtual machine

5. Browse to select the file and highlight the ibmSDN-disk1.qcow2 file. Choose the operating system type of Linux. The version shown is Red Hat and click Forward as shown in Figure A-15.

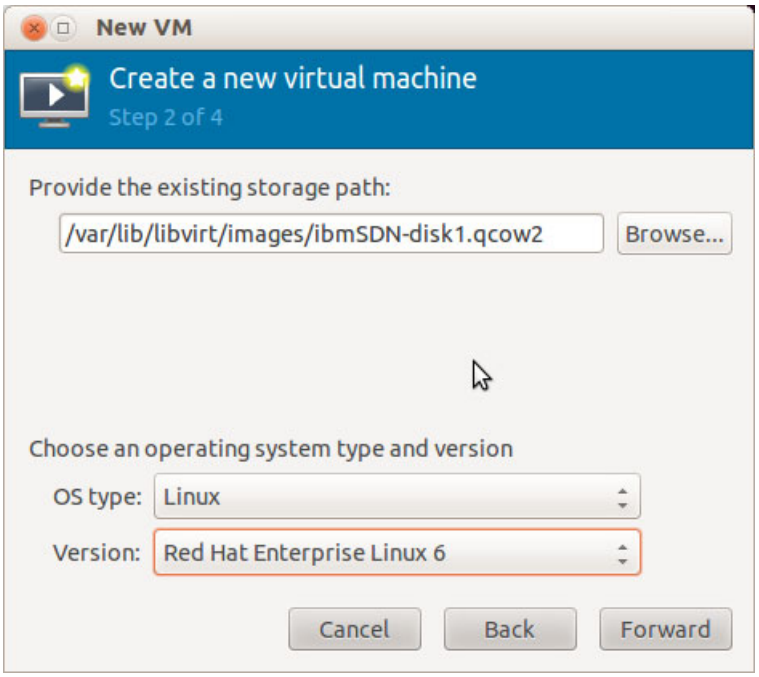


Figure A-15 Create new VM and select the OS

6. Change the memory to 8192 and select Forward as shown in Figure A-16.

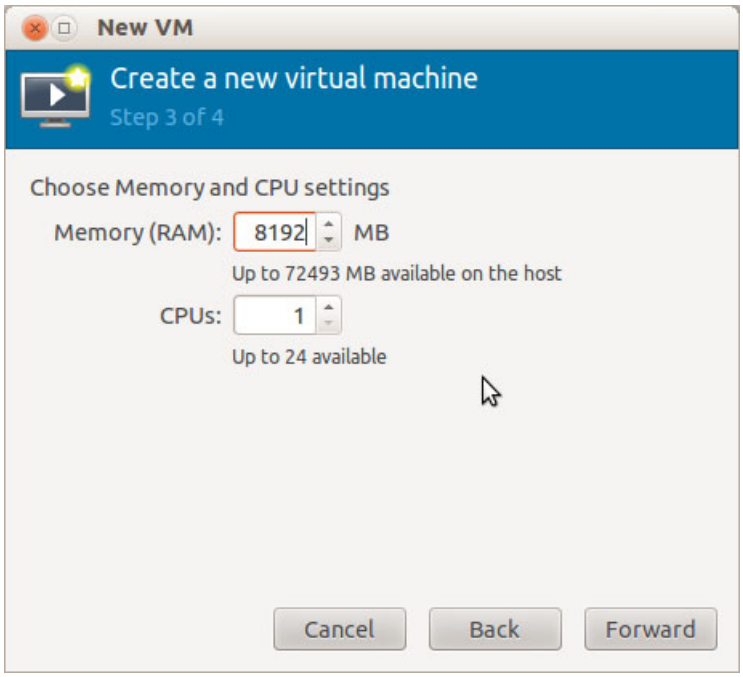


Figure A-16 Changing the memory

7. Select the *Customize configuration before installation box* and click Finish as shown in Figure A-17.

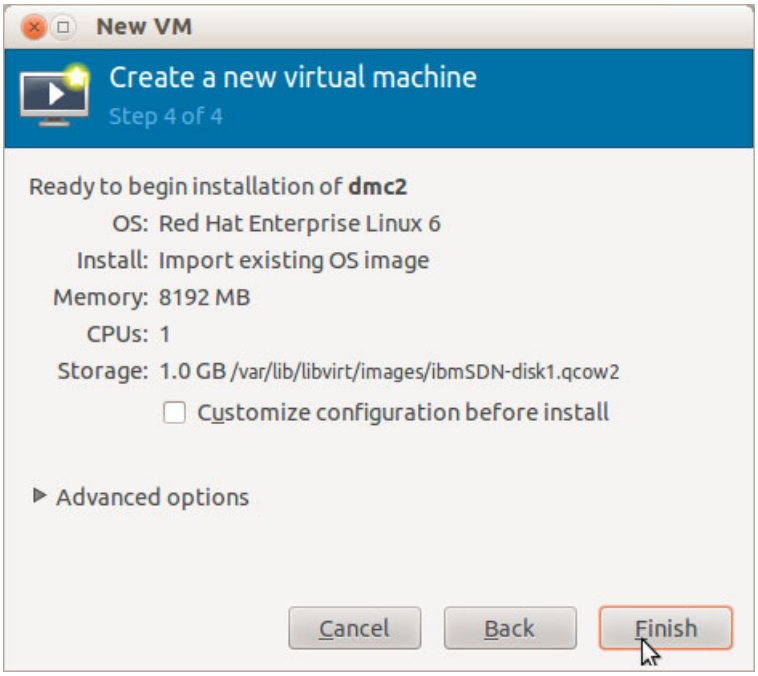


Figure A-17 Customize configuration before installation

8. Select **Add hardware > Select managed or other storage** to add the second `qcow2` file. Use the browse button to select the second `qcow2` file, and then change Device type to **Virtio disk** and storage format to **qcow2**, and click **Finish** as shown in Figure A-18.

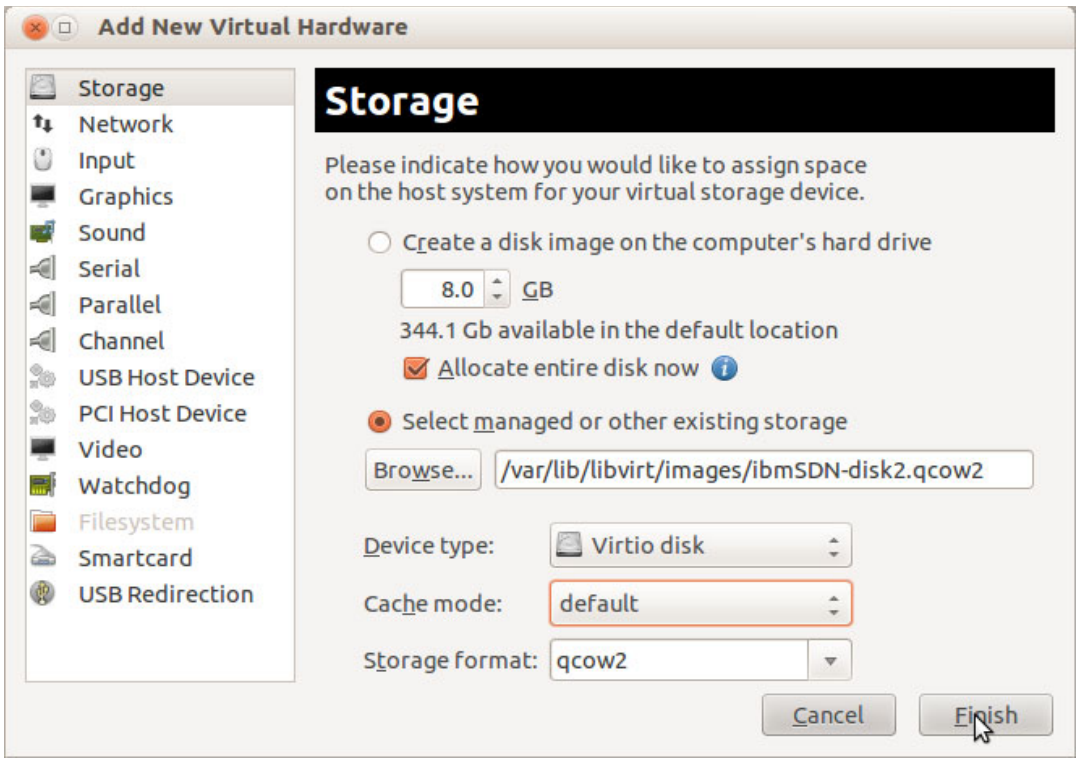


Figure A-18 Add storage

9. Select Begin installation as shown in Figure A-19.

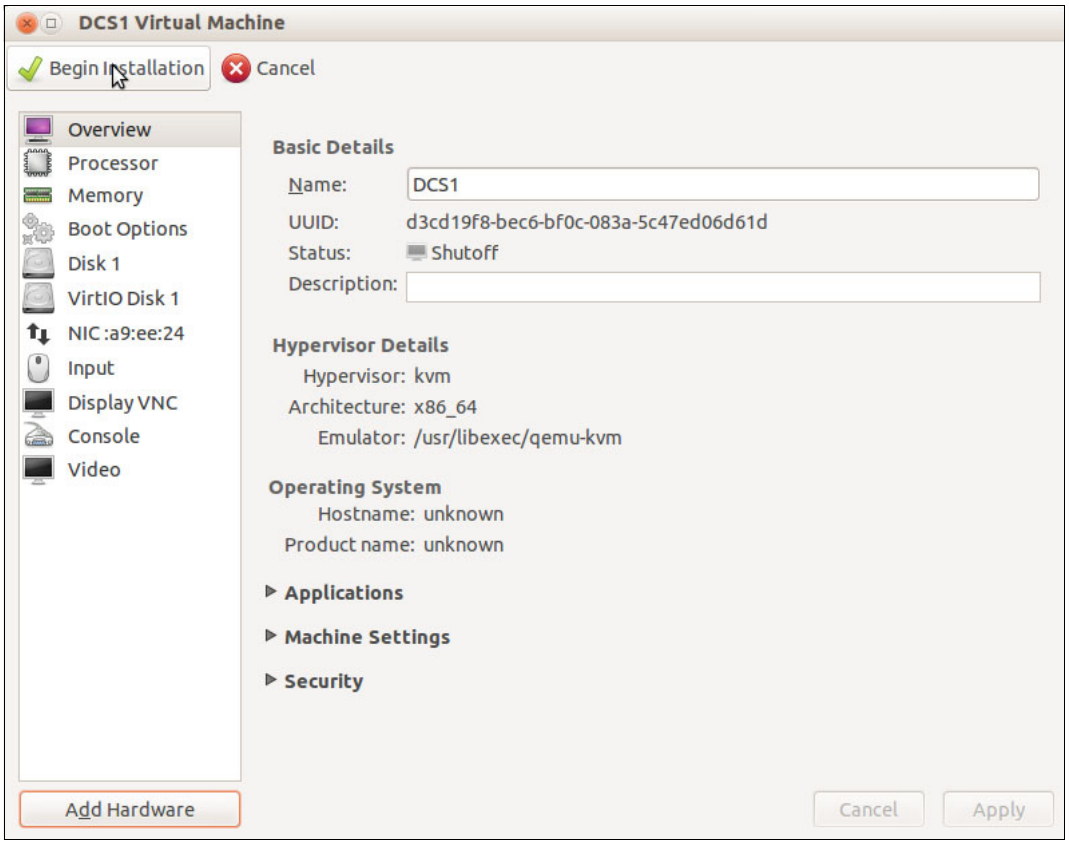


Figure A-19 Begin installation

Complete the DMC configuration using the command-line interface

Now that the Unified Controller is installed, we can log in and configure the DMC. In this section, you will configure the IP address and gateway (nexthop) for the DMC, set the language, name the DMC cluster, and assign this DMC as the primary.

Note: These steps are done the same on both Dipmgmt MCs for HA. They are both defined locally as a primary but later when configuring HA from the SDN VE GUI, you will pick an HA primary. Replace the IP address, netmask, and router with the information for your DMC.

Open the console and enter the following information:

1. User ID: `admin`
2. Password: `admin`
3. `enable`
4. `configure terminal`
5. IP address for the first DMC and its netmask:

   `ipmgmt set ip addr 129.40.122.137 mask 255.255.255.0,`
6. `nexthop set ip addr 129.40.122.254 mask 255.255.255.0`

7. `language`
8. `en_US`
9. `cluster name dmccluster primary-list ip 129.40.122.137`

SDN VE controller

The SDN VE controller is the graphical user interface to SDN. This section provides the initial licenses and configure the DMCs for HA.

Adding licenses

In this section you start the graphical user interface and add the DMC licenses.

Open your web browser to `<DMC-IP>:8080`, where DMC-IP is the IP address of your DMC. Note that the SDN VE controller will redirect port 8080 to port 8443 as shown in Figure A-20.

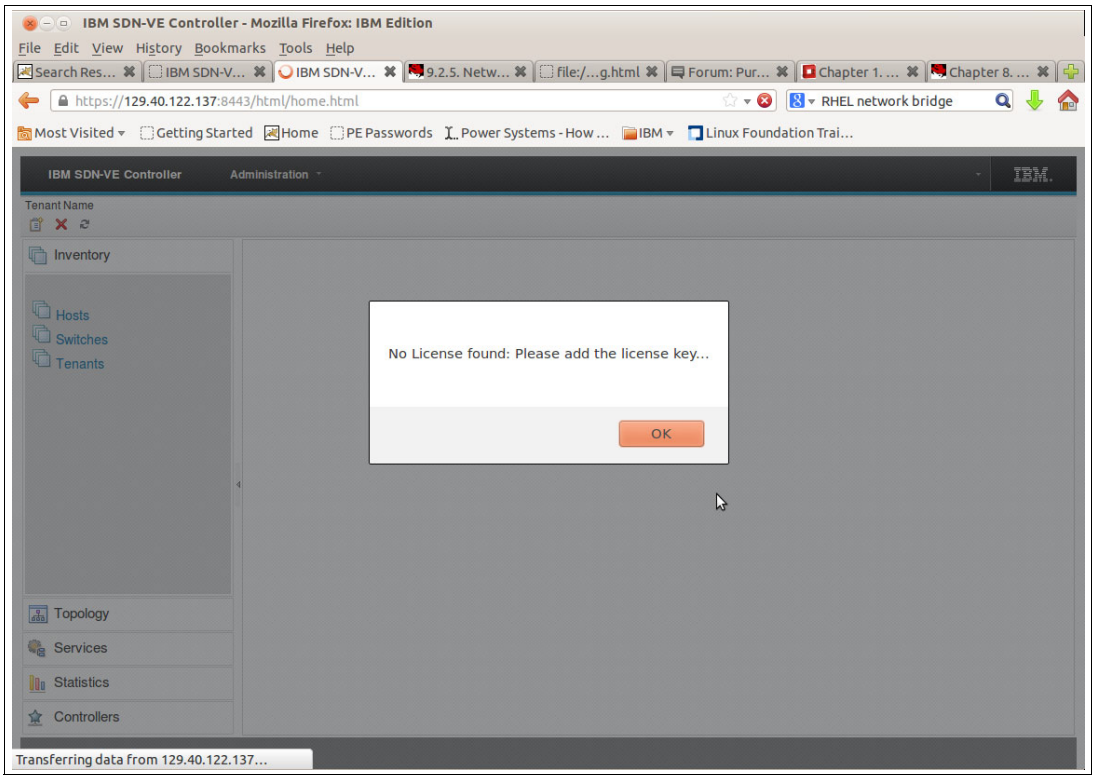


Figure A-20 Add licenses for DMC

Add the license for this DMC and Submit as shown in Figure A-21.

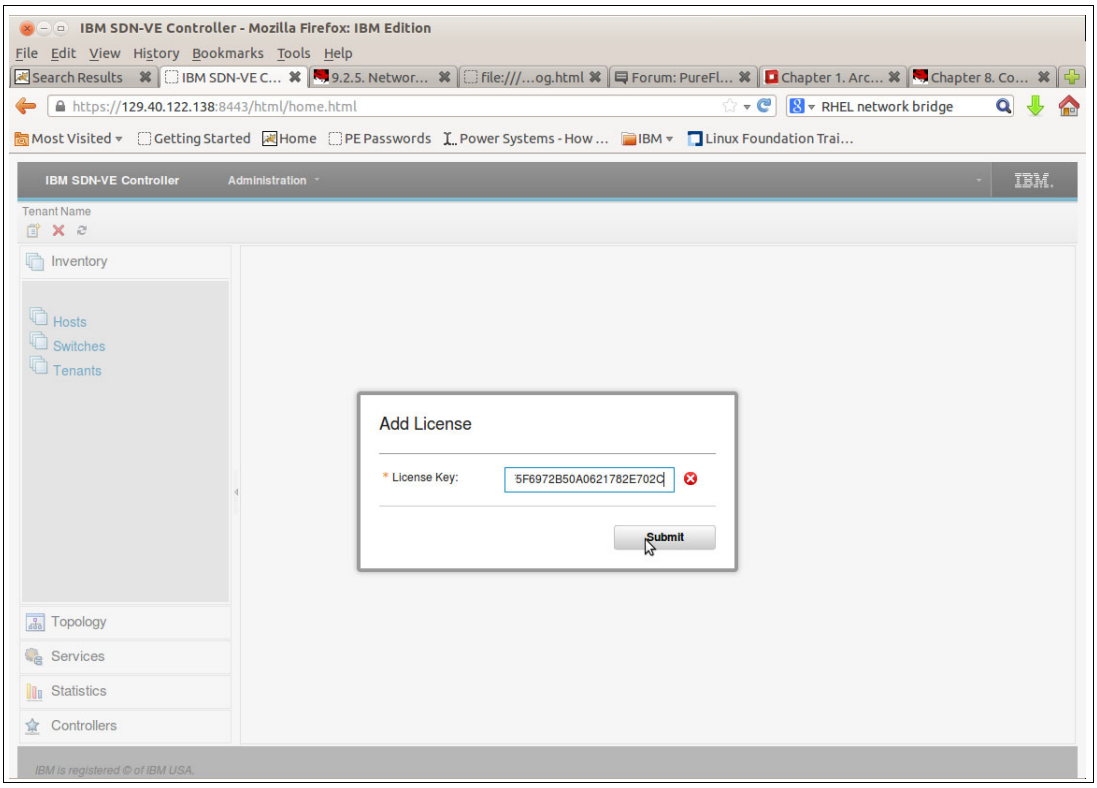


Figure A-21 Add license and submit

After adding the license, verify it by selecting **Administration > Licenses** as shown in Figure A-22.

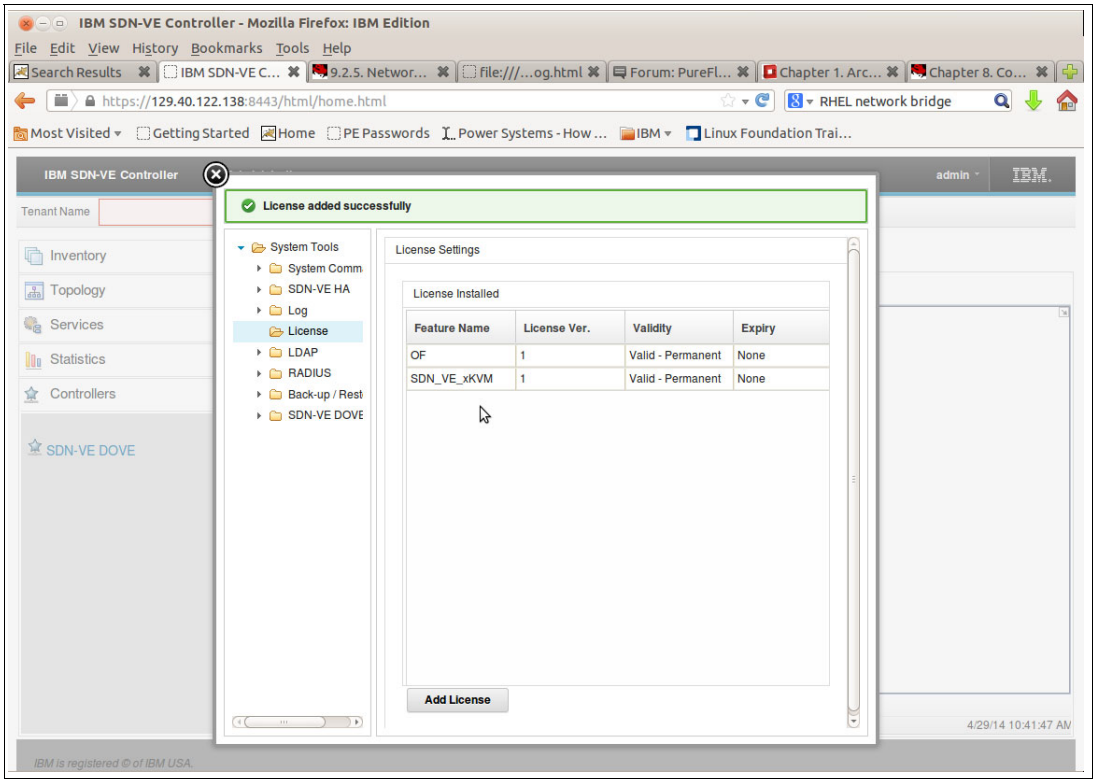


Figure A-22 Validate licenses

This completes the installation and configuration of the first DMC. This process of installing the DMC KVM image and configuring it needs to be repeated for the second DMC for HA. In our environment, we installed each DMC in its own external System x server KVM.

Using the SDN VE controller to configure the DMCs for HA

At this point two DMCs have been installed and configured. The next step is to configure these DMCs in an HA environment. To do this you need to provide an external DMC IP address. This external IP address will be used to access the SDN VE GUI rather than using the previous IP addresses which were set on each DMC. Think of this IP address as the HA IP address for the SDN VE. There are several steps to this process which include setting the Configuring the External DMC IP, Setting the peer, starting HA, and syncing the environment.

1. Select **Administration > SDN VE DOVE** and click **Configure External** at the bottom. Enter the IP address and netmask for this external DMC IP address and click **OK** to save as shown in Figure A-23.

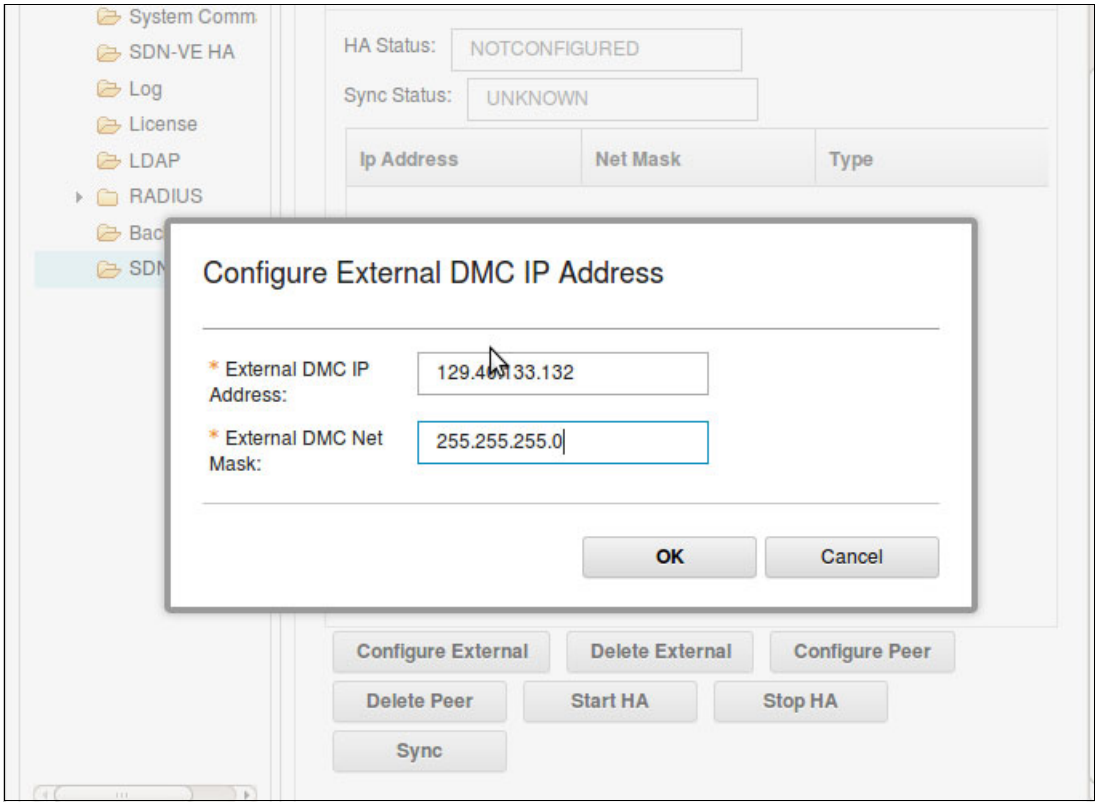


Figure A-23 Configure external DMC IP address

2. Set the Peer information by selecting **Configure Peer** and completing the Primary IP, Primarily Mask, Secondary IP, and Secondary mask, and then click **OK**, as shown in Figure A-24 on page 751. These are the IP addresses and netmasks for the two DMCs that you have previously configured.

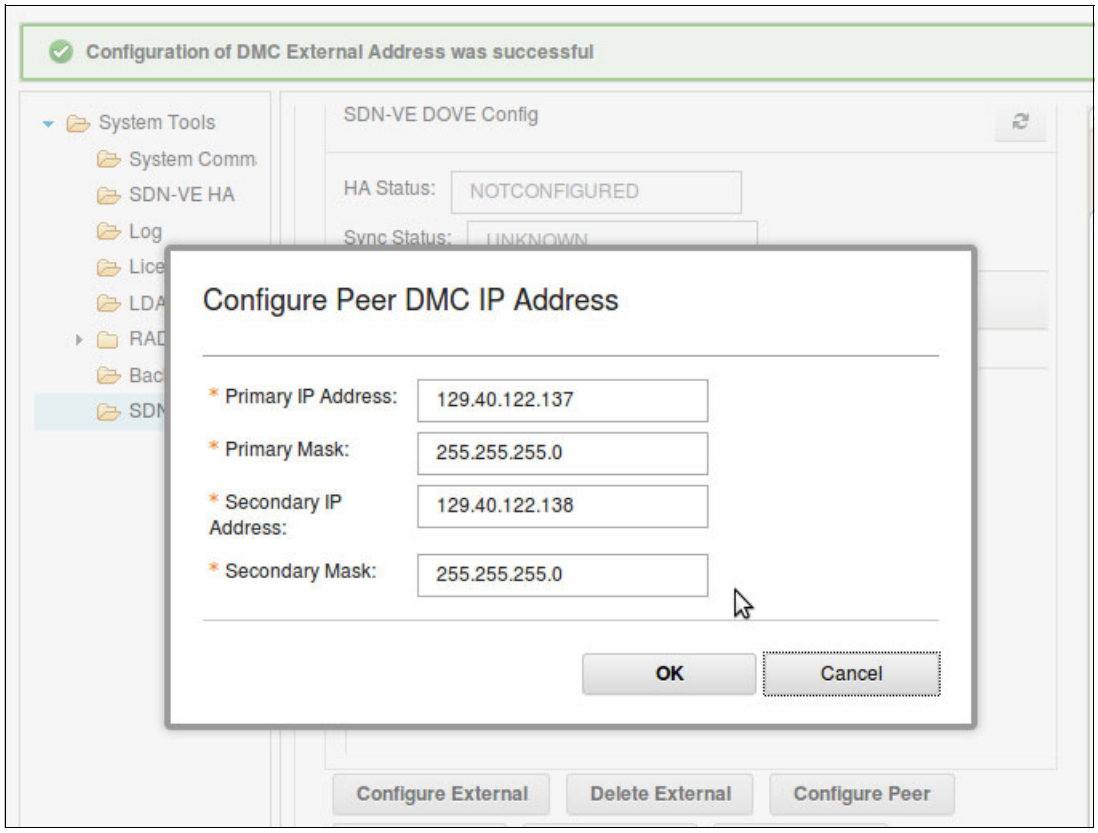


Figure A-24 Configure peer DMC IP address

You will see the IP addresses for the external, primary, and secondary displayed as shown in Figure A-25.

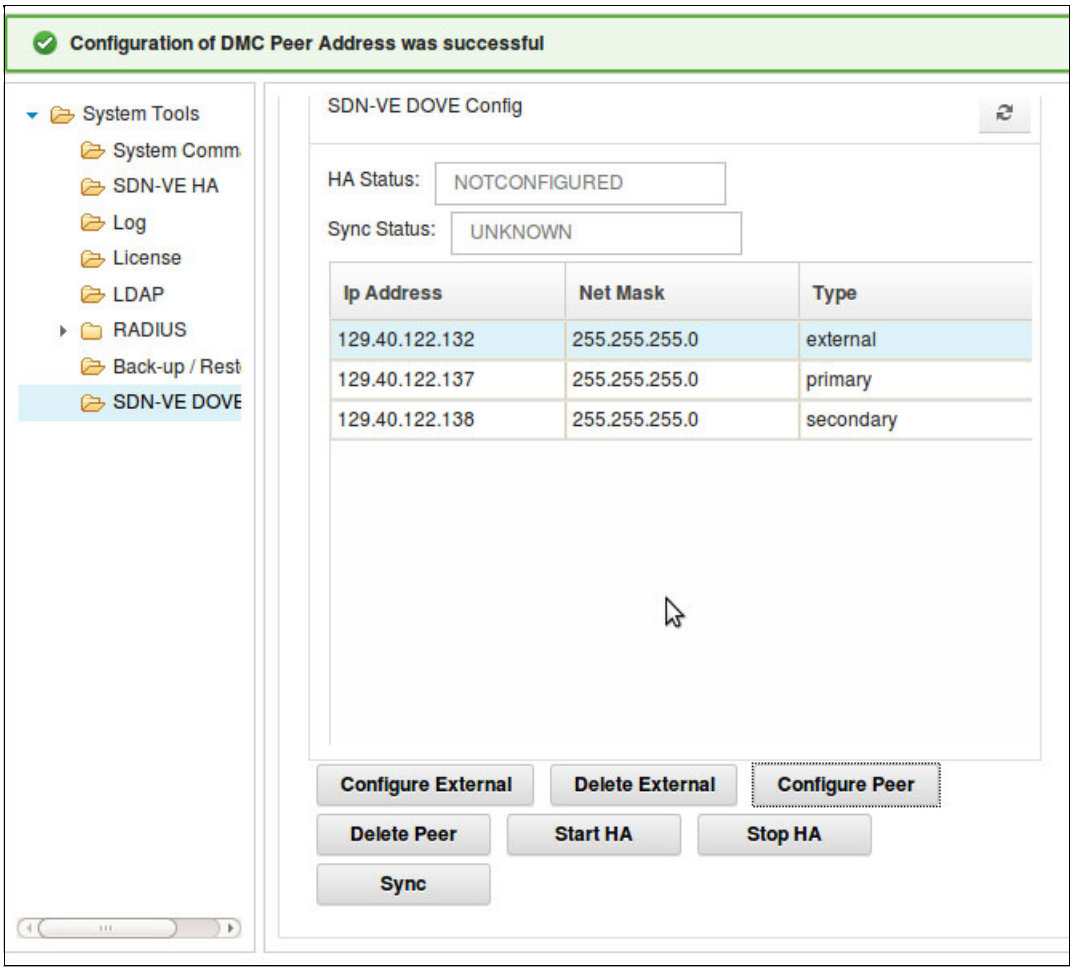


Figure A-25 Validate config of DMC peer address

3. Now, select **Start HA** and then, after starting HA, select **Sync**. When HA has started and the sync has finished, “HA status” and the “Sync Status” will be updated as shown in Figure A-26.

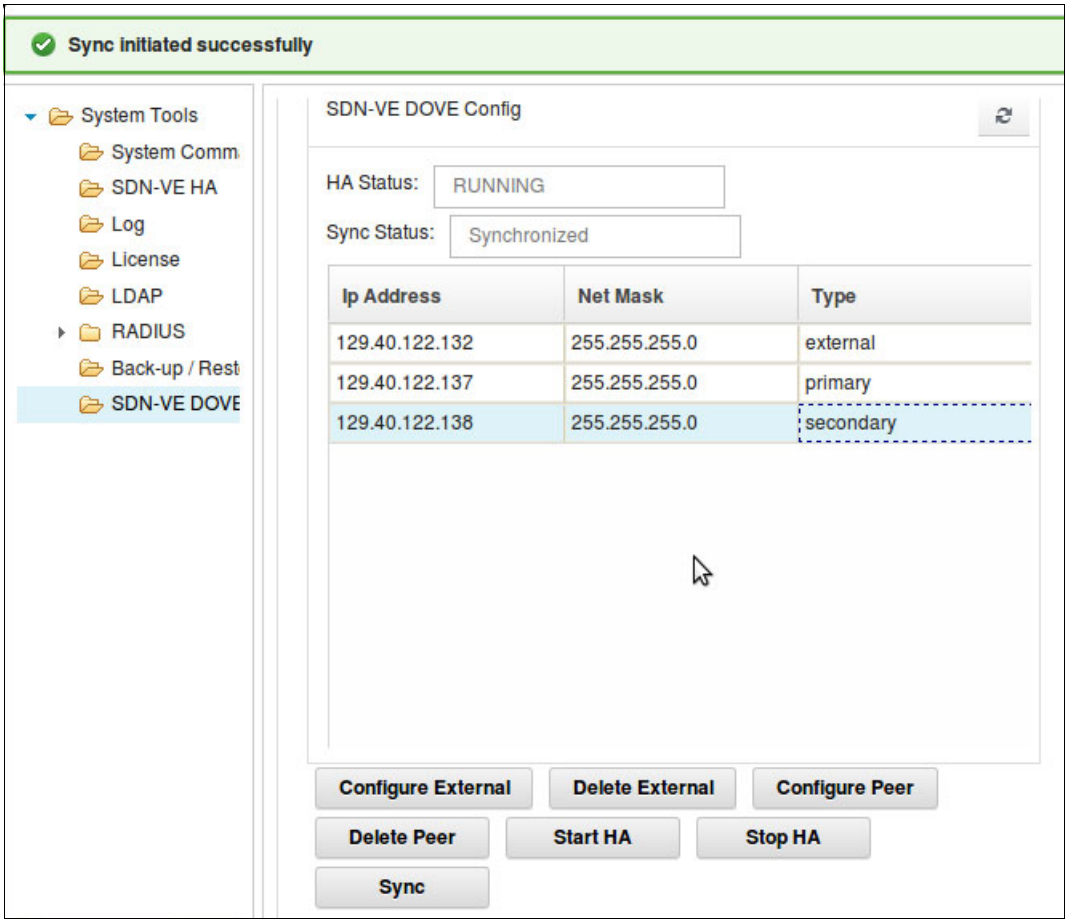


Figure A-26 HA Sync

This completes the HA setup for the DMCs.

Install and configure the Distributed Connectivity Service (DCS)

This installation process is similar to the DMC installation. You will install two DCS services which contain the database and be used by other SDN services to define and query the connectivity.

This process will include copying and setting up the DMC KVM files for installation, installing and configuring the DCS KVM, and adding it to the SDN VE controller.

Installing the DCS KVM

This section shows how to install the DCS KVM.

1. First, extract the `SDN_VE_Service_Appliance_DCS_DGW_1.1.0.kvm` file to extract the two `qcow2` files, and copy them to `/var/lib/libvirt/images`, as shown in Figure A-27.

```
[root@sdnmgr tmp]# tar -xvf SDN_VE_Service_Appliance_DCS_DGW_1.1.0.kvm
ibmDOVEsvc-disk1.qcow2
ibmDOVEsvc-disk2.qcow2
[root@sdnmgr tmp]# mv ibmDOVEsvc-disk* /var/lib/libvirt/images/
[root@sdnmgr tmp]# cd /var/lib/libvirt/images/
[root@sdnmgr images]# ls
ibmDOVEsvc-disk1.qcow2  ibmDOVEsvc-disk2.qcow2  ibmSDN-disk1.qcow2  ibmSDN-disk2.qcow2
[root@sdnmgr images]#
```

Figure A-27 Extract SDN_VE_Service_Appliance_DCS_DGW_1.1.0.kvm file

2. Start the virtual machine Manager, connect to the KVM hypervisor where you want to install the DCS, and a select **NEW** to create the new DCS KVM.
3. On the first panel provide, the DCS name and select **Import existing disk image**.
4. Then, select **Forward**, as shown in Figure A-28.

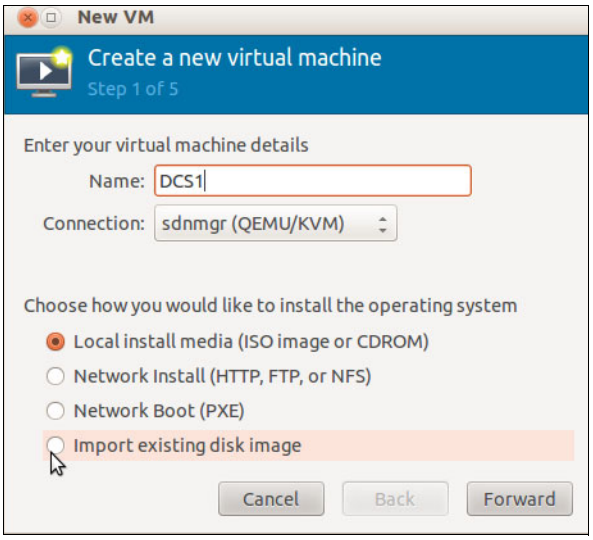


Figure A-28 Create the new VM - DCS1

5. Select the **ibmDovesvc-disk1.qcow2** file and **Choose Volume** to continue, as shown in Figure A-29.

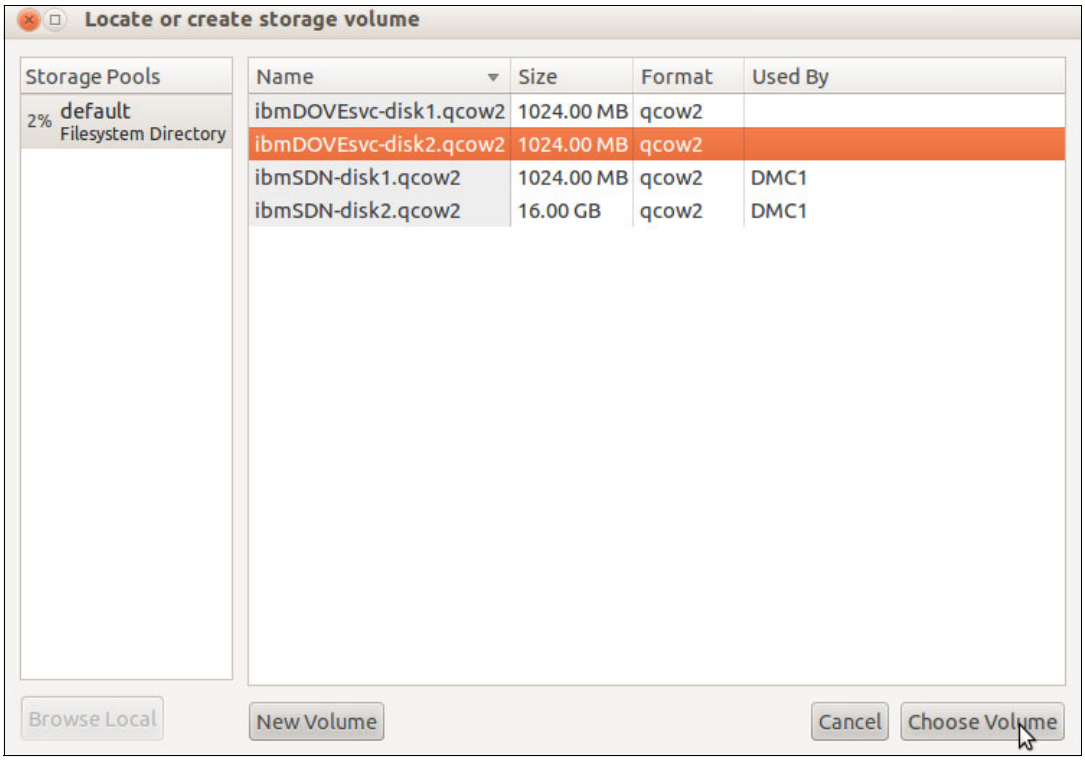


Figure A-29 Choose volume

6. Select **Linux** as the operating system type and **Red Hat Enterprise Linux 6** as version, and then select **Forward**, as shown in Figure A-30.

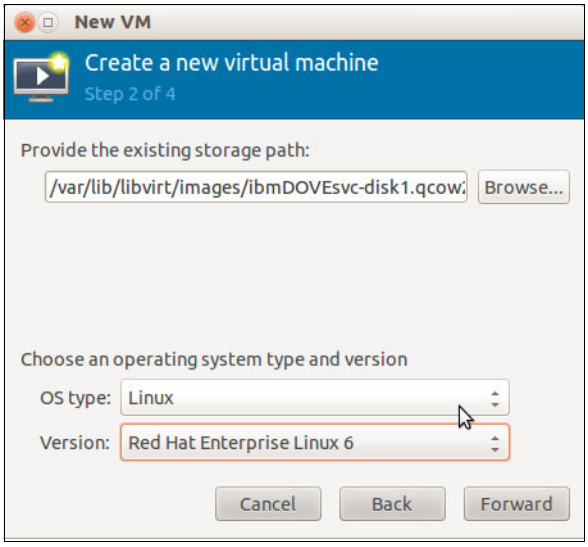


Figure A-30 Selecting the operating system

7. Select **8192** for memory and leave CPU as 1, as shown in Figure A-31.

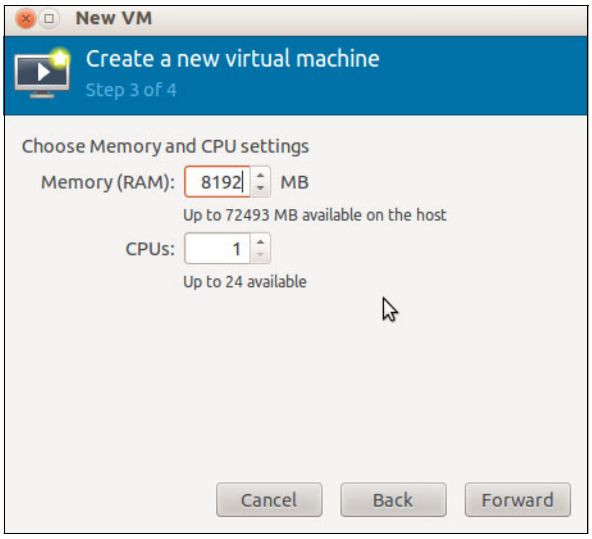


Figure A-31 Selecting the memory and CPU

8. Click **Customize configuration before install** and click **Finish**, as shown in Figure A-32.

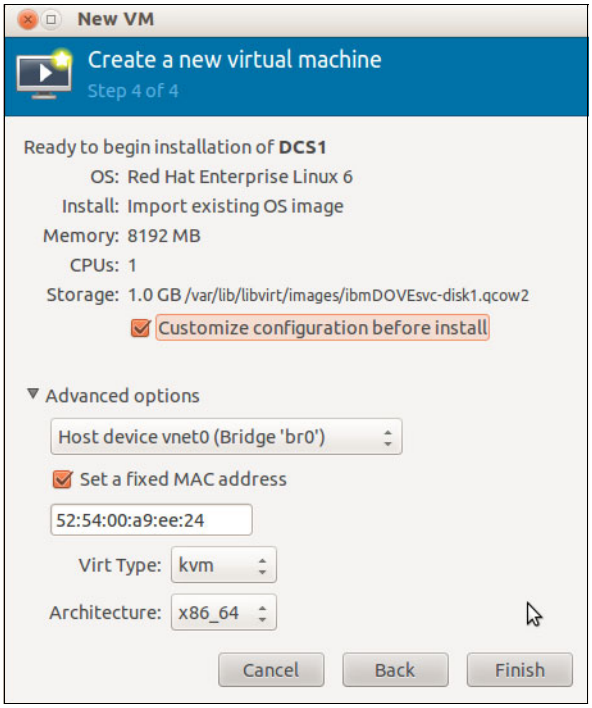


Figure A-32 Customizing the configuration before the install

9. Select the **ibmDOVEsvc-disk2.qcow2** file and **Choose Volume** as shown in Figure A-33.

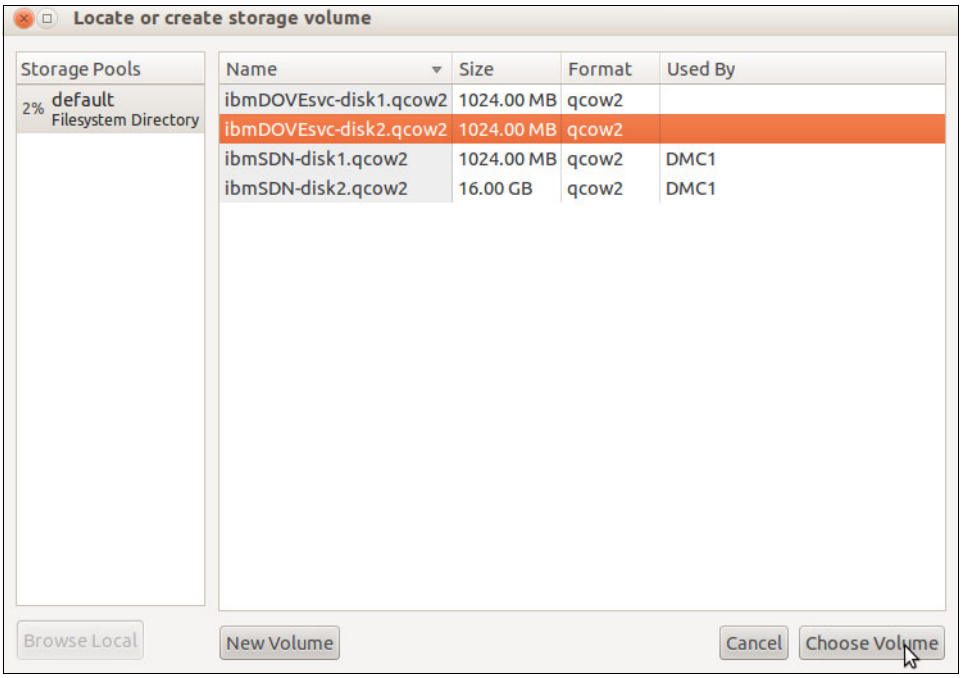


Figure A-33 Choose the volume

10. Click **Select managed or other existing storage**.

11. Choose the device type of **Virtio disk** and storage format of **qcow2**, and then click **Finish**, as shown in Figure A-34.

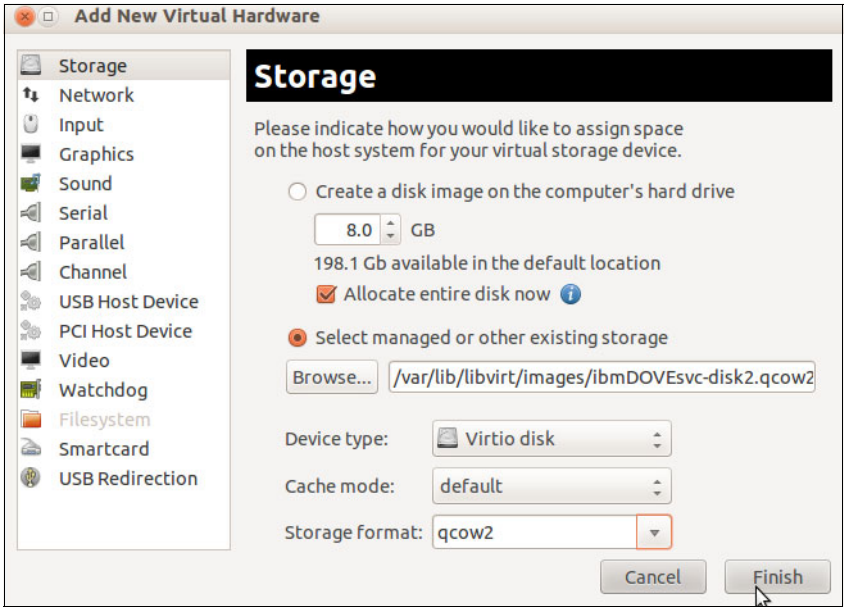


Figure A-34 Selecting the storage

12. Select **Begin Installation** to install the DCS server, as shown in Figure A-35.

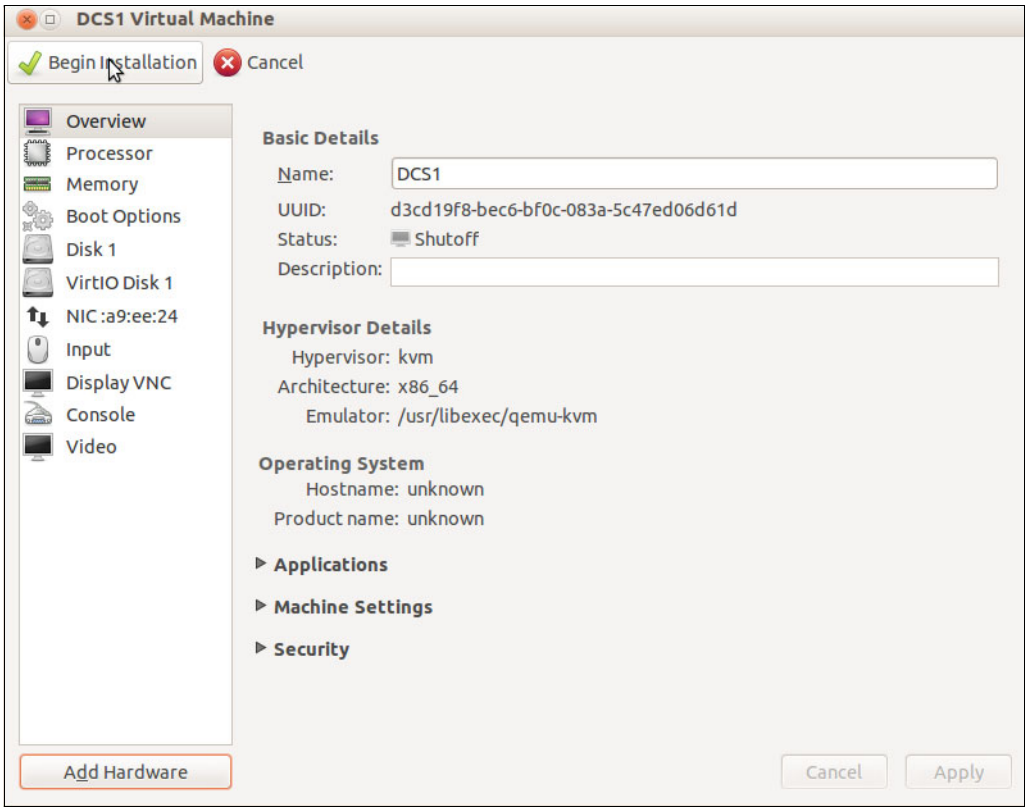


Figure A-35 Installing the DCS server

Completing the DCS configuration

Now that the DOVE Controller service is installed, we can log in and configure the DCS using the command-line interface. In this section, you will configure the IP address and gateway (nexthop) for the DCS, set the language, name the DCS cluster, and assign this DCS to its DMC.

For scaling, define more than one DCS. In our configuration we set up four DCSs: One DCS VM on each the two external servers and one in each of the two Underlay networks.

Note: These steps are the same for each DCS. Replace the IP address, netmask, and router with the information for your DCS.

1. Open the console and enter the following information:
   - User ID is admin
   - Password is admin
   - enable
   - configure terminal
   - ipmgmt set ip addr 129.40.122.128 mask 255.255.255.0 -- the IP address for the first DMC and its netmask
   - ipmgmt set nexthop ip 129.40.122.254
   - dmc set ip 129.40.122.132

2. Log in to the web interface for the DMC, select **Application**s, and then change the new DCS, which should have been discovered, from N to **Y** to define it as a DCS (as shown in Figure A-36). This needs to be done because the same KVM image is used for both DCS and DGW (gateway), so the administrator needs to specify if this service is a DCS or a DGW.

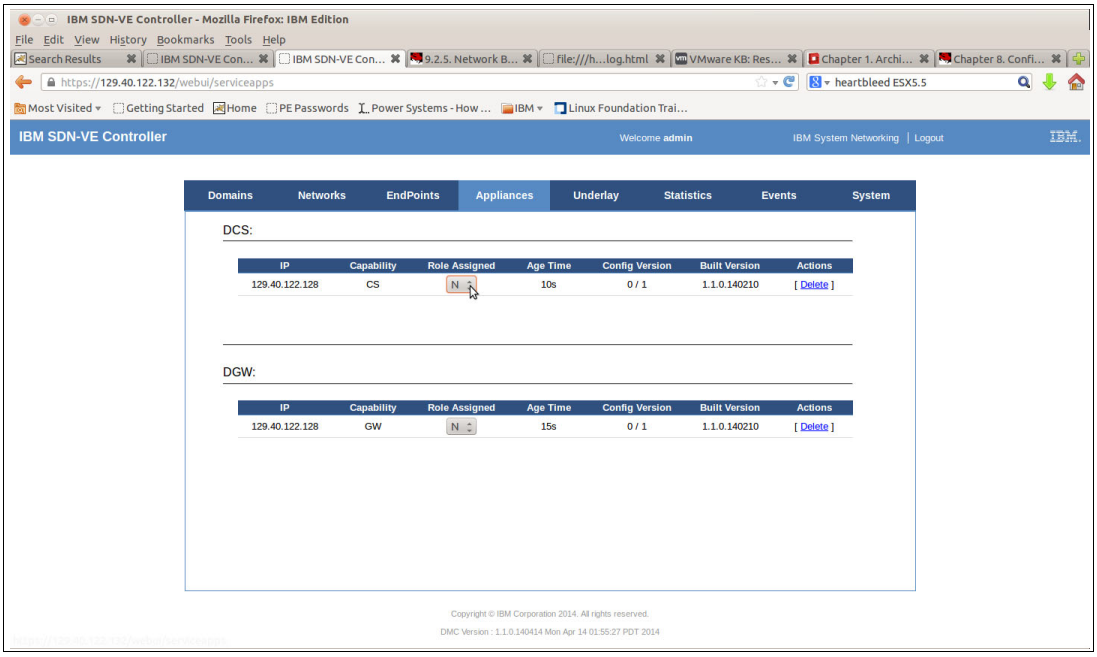


Figure A-36 Log in to the DMC

The steps to install and configure the DCS need to be repeated for each DCS that you want in your configuration. We had four DCSs configured. Figure A-37 shows the final DCS configuration.

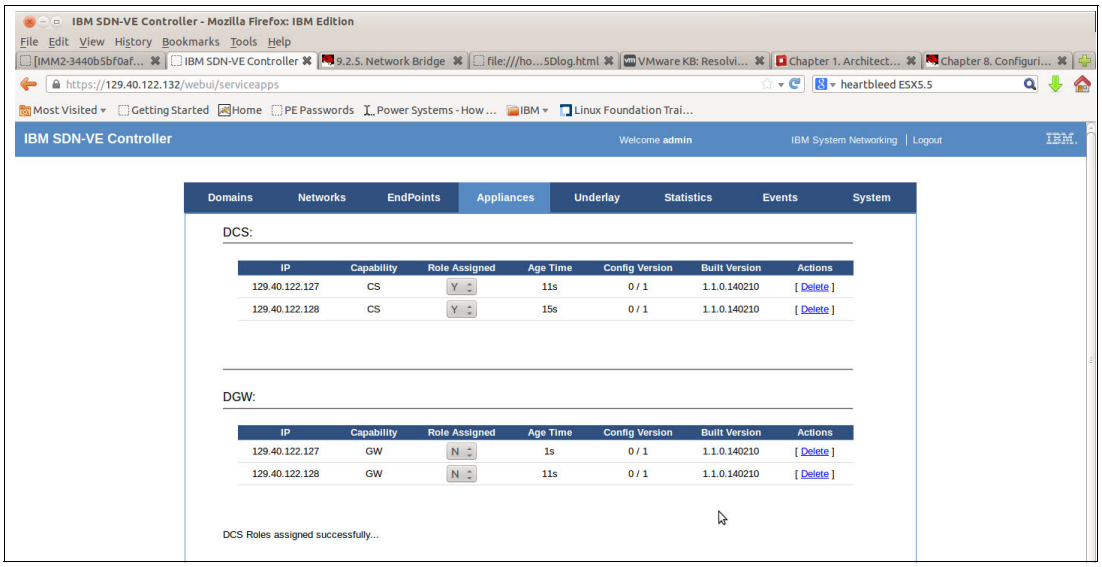


Figure A-37 Final DCS configuration

Install and configure the Distributed Gateway

At this point, the DMCs are installed and setup as HA and the DCSs have been installed and configured. In this section, you will be installing and configuring the Distributed Gateway (DGW). These steps involve installing deploying the DGW VM with KVM and then configuring the DGW.

1. Untar the compressed file to save the `qcow2` files for deployment and copy them to `/var/lib/libvirt/images` directory in preparation for DGW KVM VM installation, as shown in Figure A-38.

```
[root@f3av04 tmp]# ls
SDN_VE_Service_Appliance_DCS_DGW_1.1.0.kvm  yum.log
[root@f3av04 tmp]# tar -xvf SDN_VE_Service_Appliance_DCS_DGW_1.1.0.kvm
ibmDOVEsvc-disk1.qcow2
ibmDOVEsvc-disk2.qcow2
[root@f3av04 tmp]# mv ibmDOVEsvc-disk* /var/lib/libvirt/images/
[root@f3av04 tmp]# virsh pool-refresh default
Pool default refreshed

[root@f3av04 tmp]#
```

Figure A-38 Extract qcow2 files for DGW

2. Connect to the server where the DGW will be installed and select New VM. Complete the name, select **Import existing disk image**, and click **Forward**, as shown in Figure A-39.

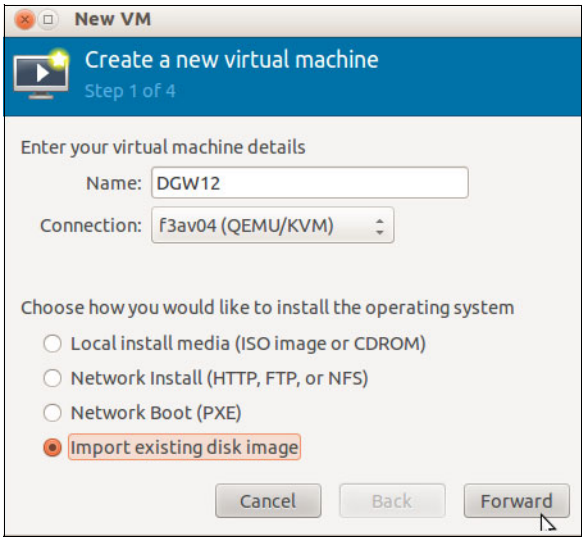


Figure A-39 Create new VM for DGW

3. Select **Browse** and choose the **ibmDOCEsvc-disk1.qcow2** file from `/var/lib/libvirt/images`, select **Linux** and **Red Hat** and click **Forward** as shown in Figure A-40.

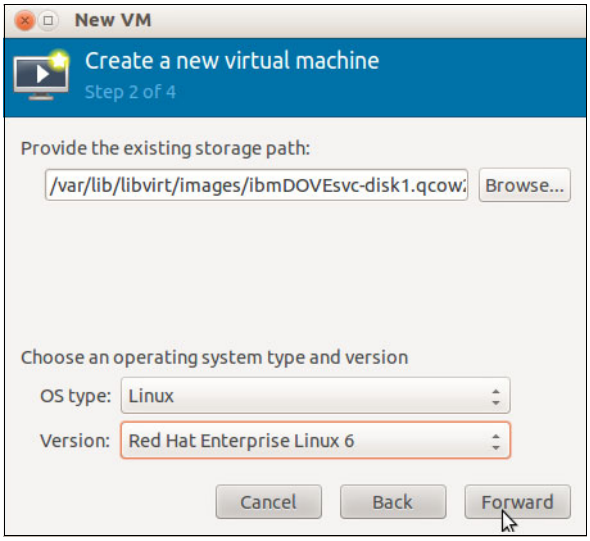


Figure A-40 Select the OS for DGW

4. Increase the memory to `8192` and then click **Forward** as shown in Figure A-41.

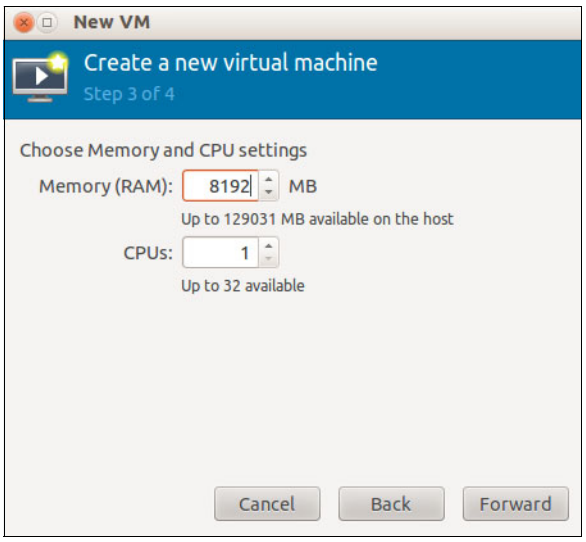


Figure A-41 Increase the memory for DGW

5. Select **Customize configuration before install** and click **Finish** as shown in Figure A-42.

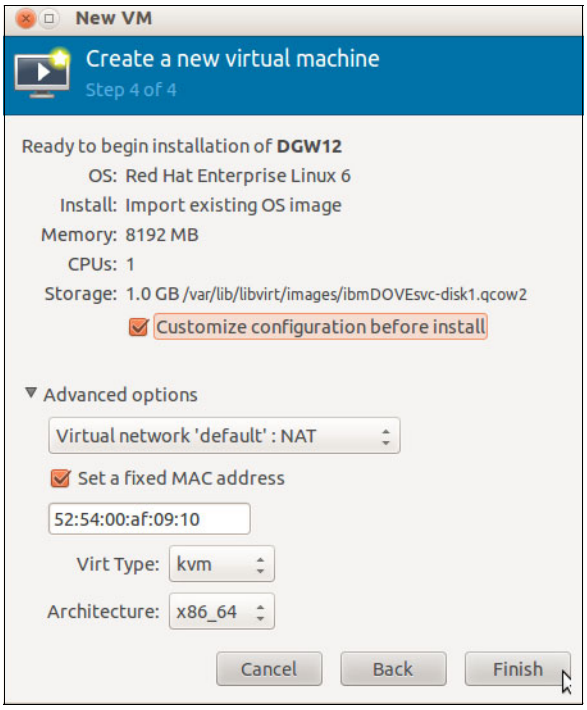


Figure A-42 Customize configuration before install - DGW

6. Select **Add Hardware** as shown in Figure A-43.

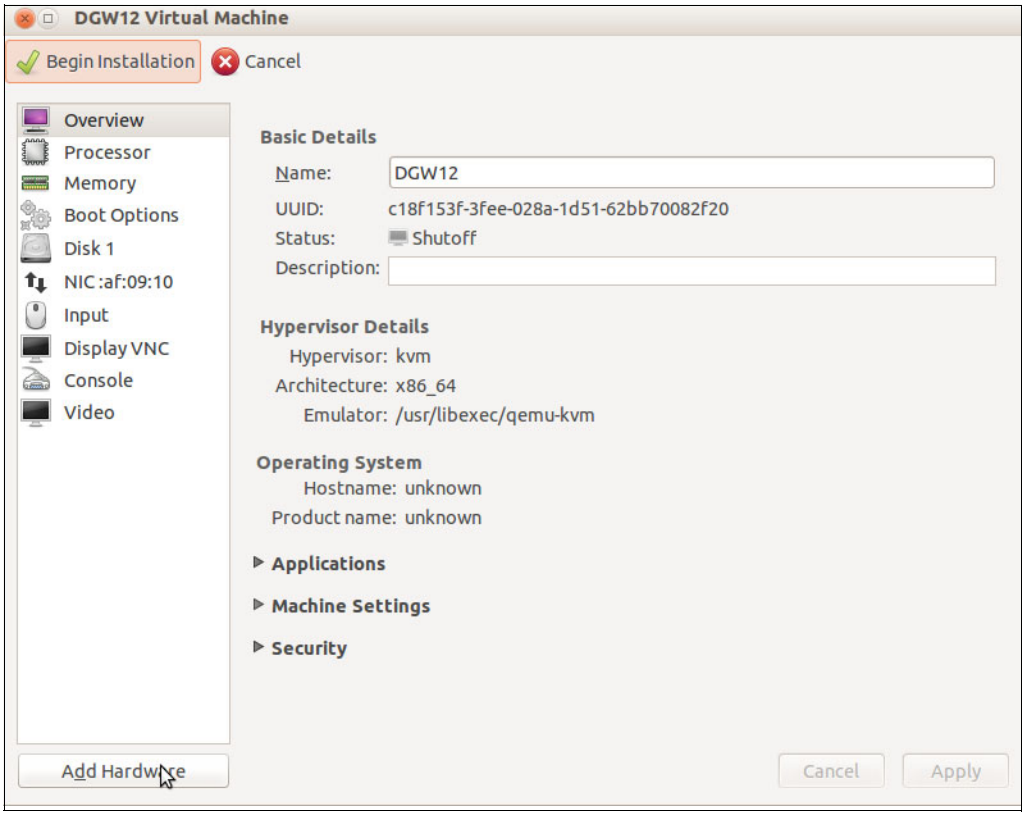


Figure A-43 Add hardware for DGW

7. Select **Storage** and then **Select managed or other existing storage**.
8. Click **Browse** and select the second the **ibmDOVEsvc-disk2.qcow2** file, select device type of **Virtio disk** and storage format of **qcow2** and click **Finish** as shown in Figure A-44.

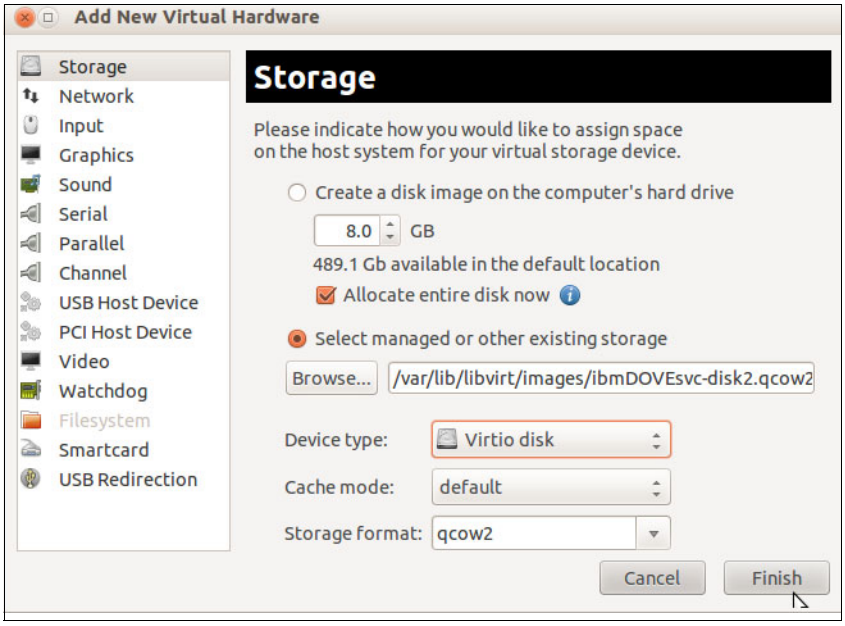


Figure A-44 Add storage for DGW

9. Click **Begin Installation**, as shown in Figure A-45.

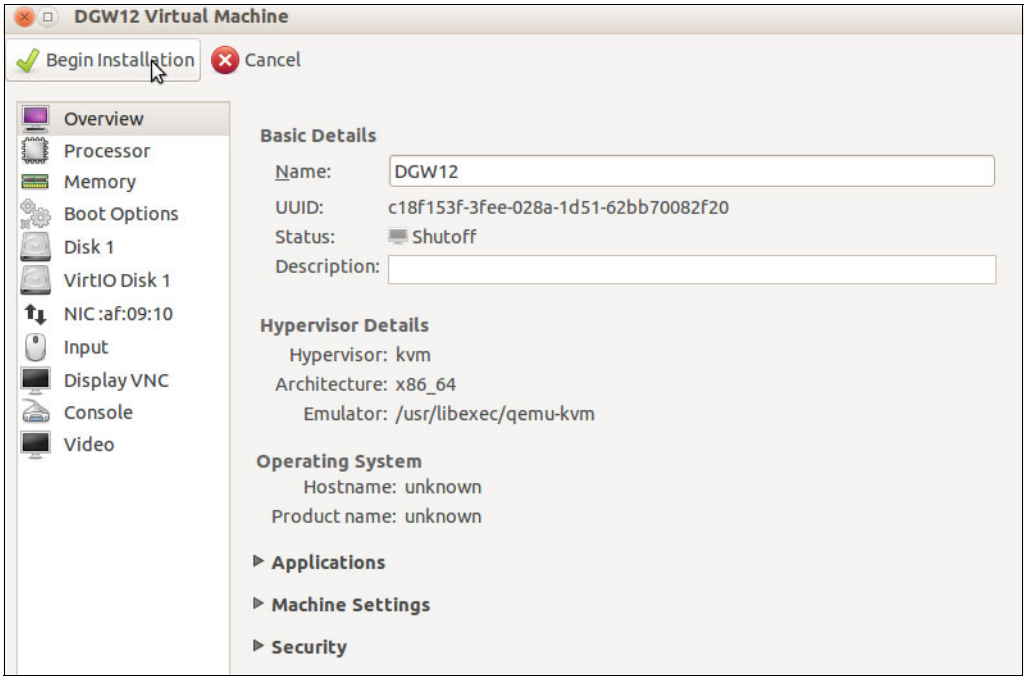


Figure A-45 Begin installation for DGW

10. Open a console to the new VM and configure the IP address using the following commands:

```
enable
configure terminal
ipmgmt set ip add <ip> mask <netmask>
```

Complete the DOVE gateway configuration

In this section, we complete the DGW configuration.

Define a domain

1. Start the SDN VE GUI by putting the IP address of the external DMC in the web browser. Log in to the GUI and select **Domain**, and then **Add New Domain**.
2. Enter the new domain name, and then click **Submit** as shown in Figure A-46.

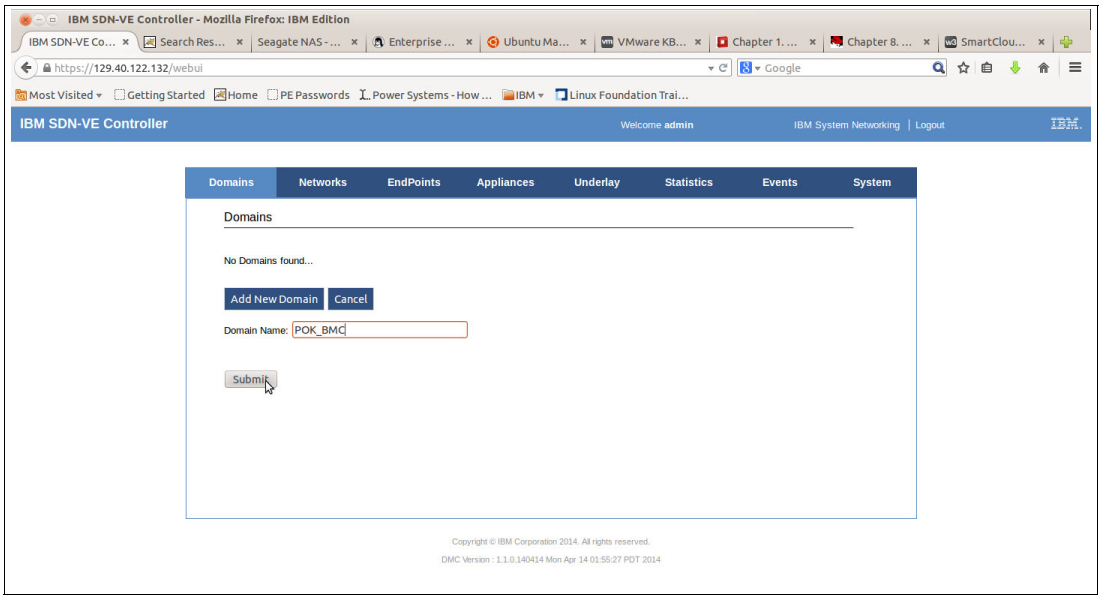


Figure A-46 Adding a new domain

The new domain is now shown in Figure A-47.

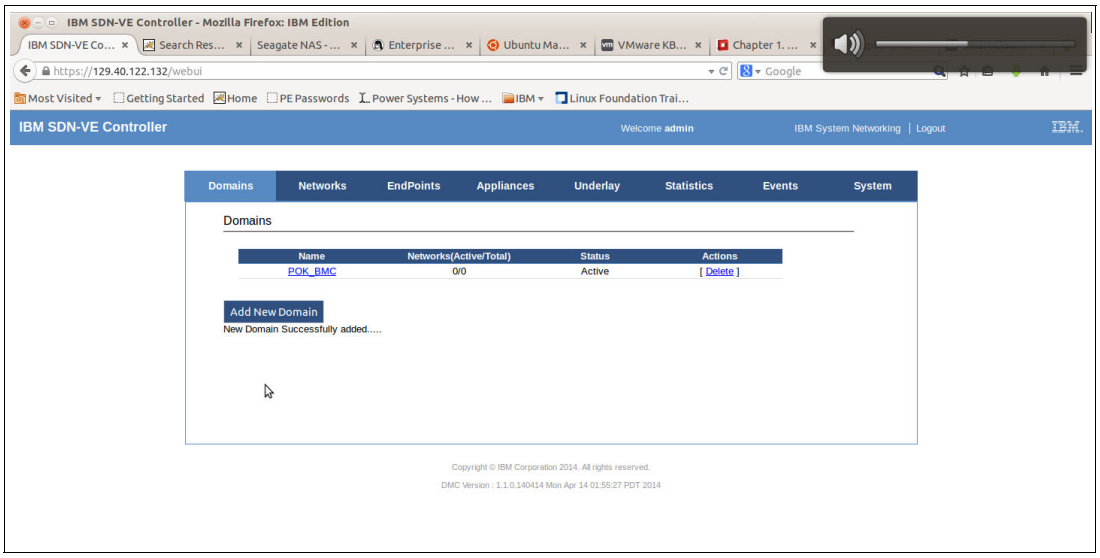


Figure A-47 New domain

Define networks

This section describes the SDN VE interface for adding networks and for configuring the Underlay and Overlay networking.

Add a new network definition

Click the new domain to be able to create new network definitions. Then, click **Add Network**, provide a number for the Network ID and the Network Name, and click **Submit** as shown in Figure A-48.

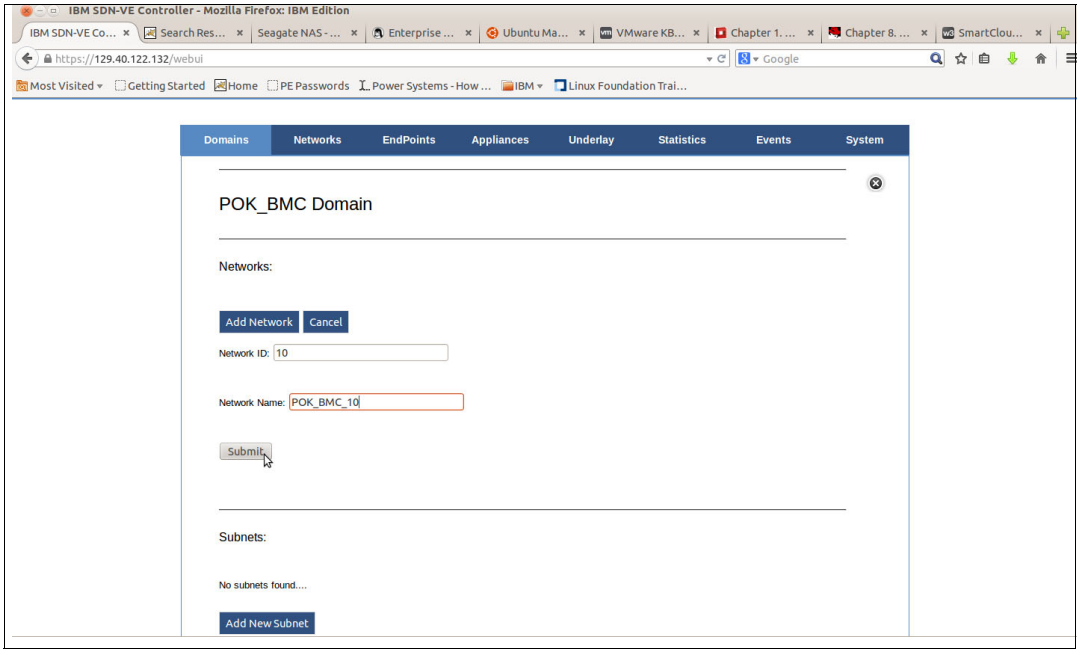


Figure A-48 New network

The resulting Network definition is displayed as shown in Figure A-49.

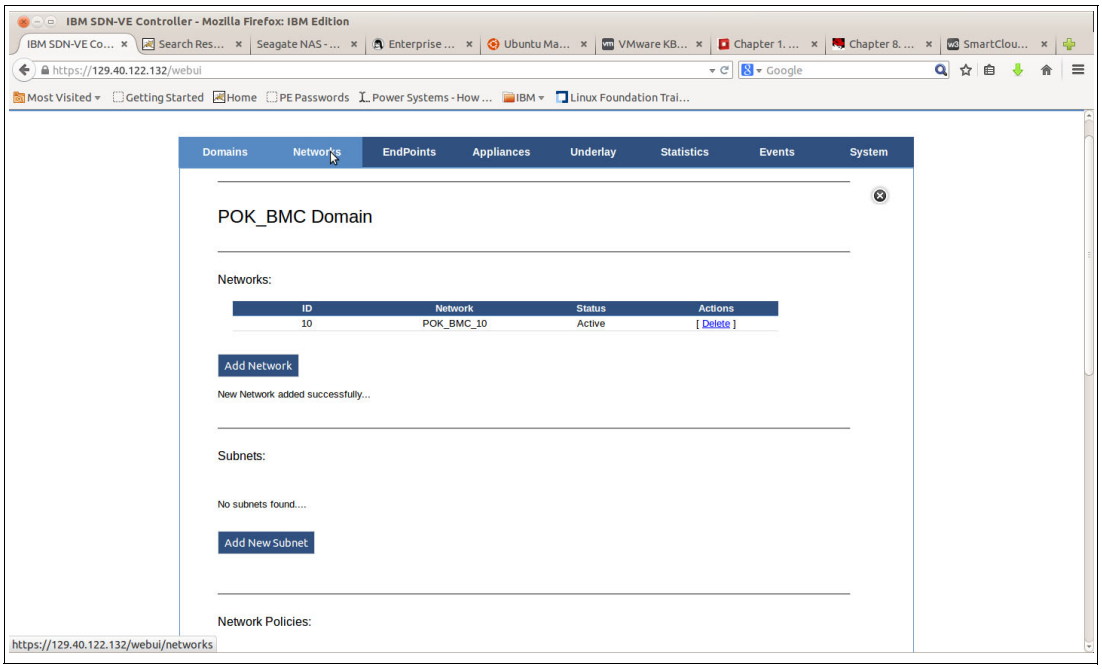


Figure A-49 New network displayed

Create a network subnet

1. Create the network subnet definition by selecting **Add NewSubnet** and completing the NetMask and Nexthop entries.
2. Then, click **Submit**, as shown in Figure A-50.

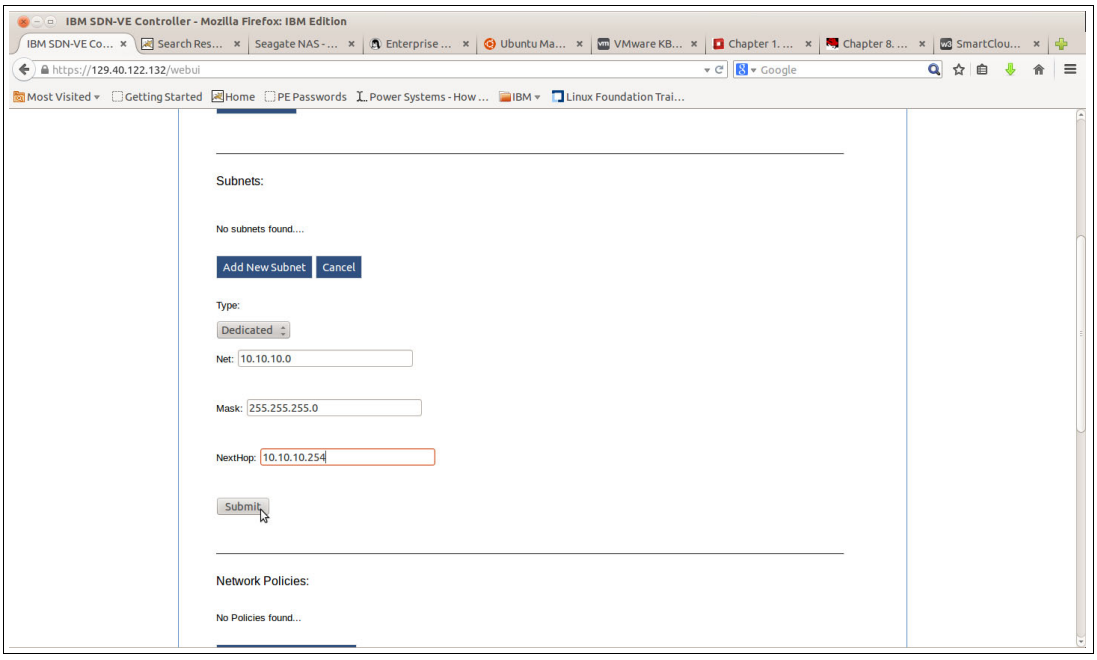


Figure A-50 Create network subnet

The resulting subnet definition is then displayed, as shown in Figure A-51.

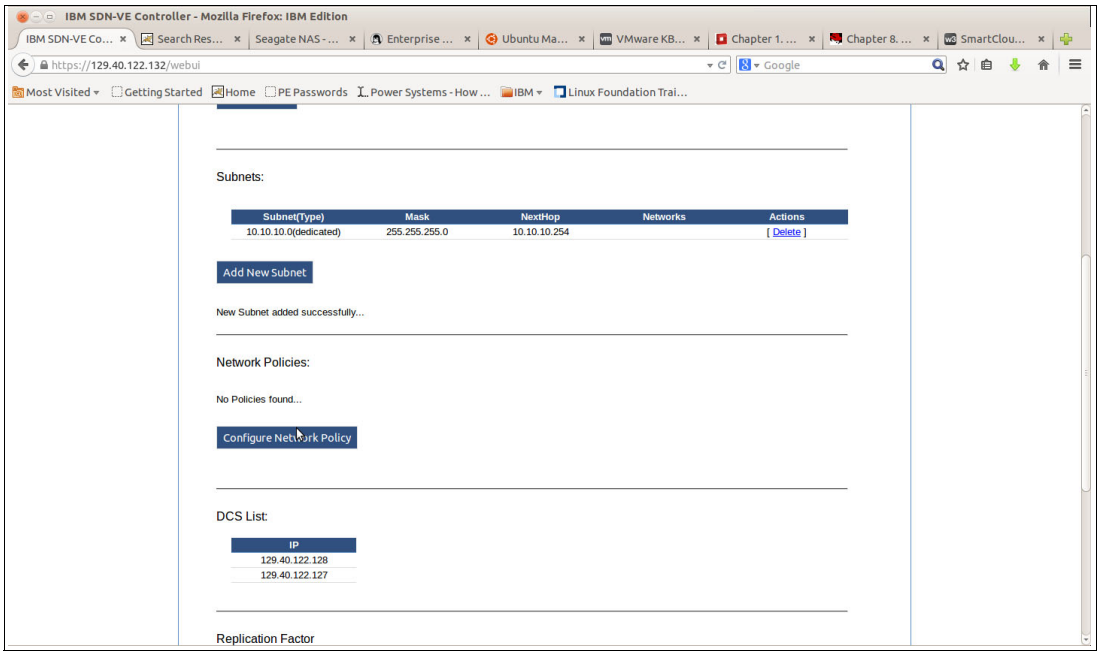


Figure A-51 Network subnet displayed

Assign the subnet to the network

Now that you have defined a network and a subnet, you can associate the subnet with the network by adding the subnet to the network.

1. Click the **Networks** tab, and then click the network highlighted in blue, where you will add a subnet, as shown in Figure A-52.

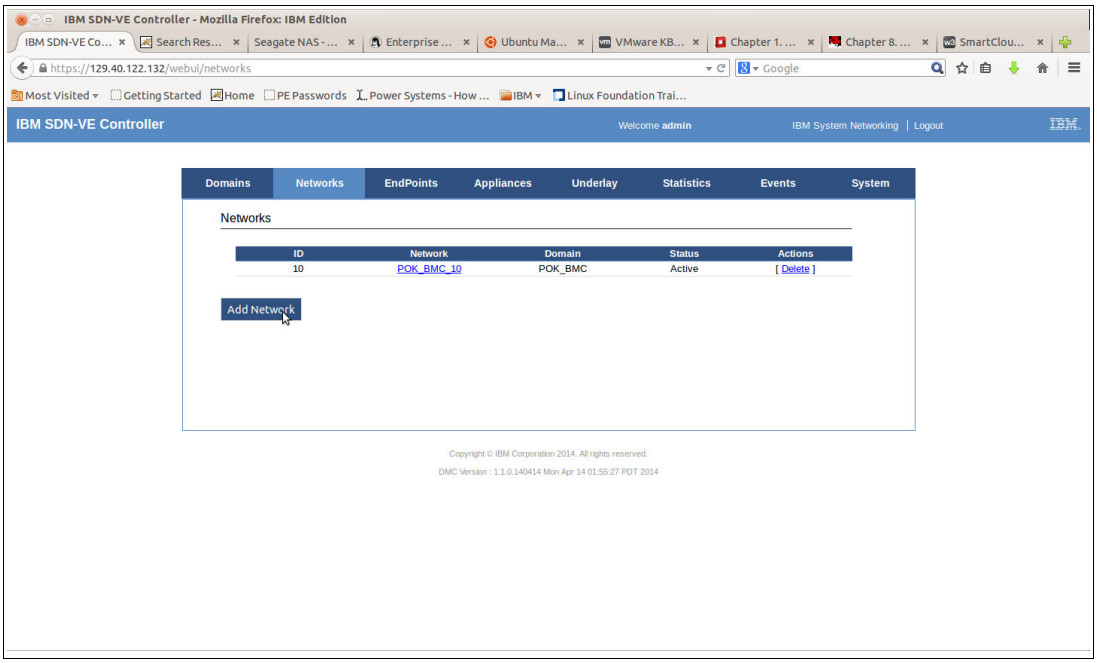


Figure A-52 Assign subnet to network

2. Click **Assign Subnet**, as shown in Figure A-53.

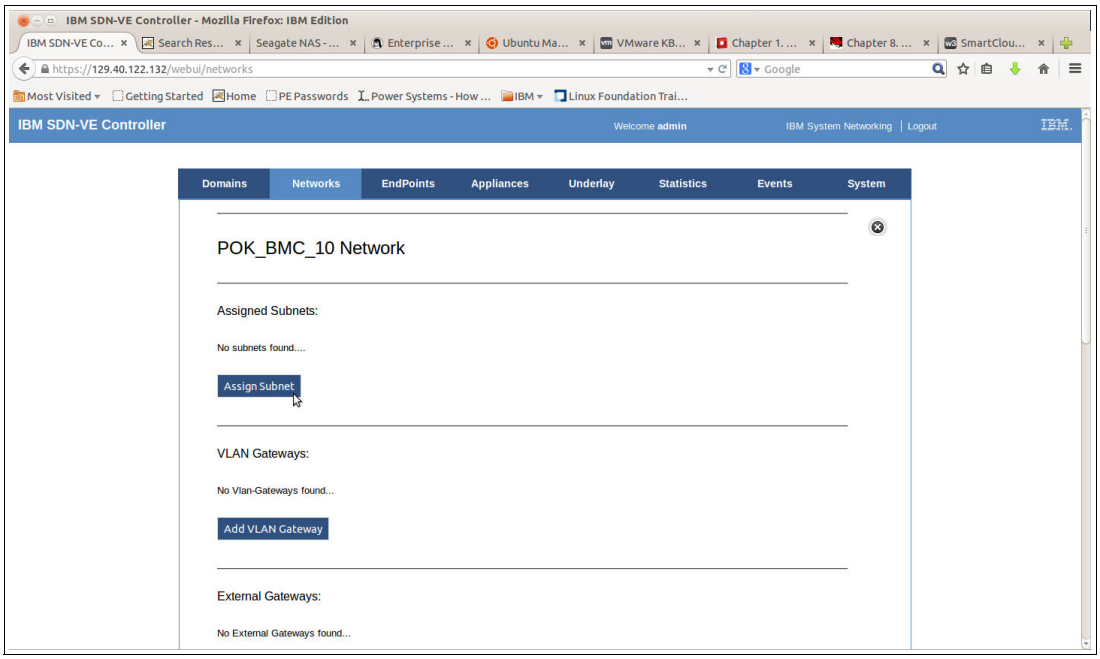


Figure A-53 Assign subnet

3. From the pull-down menu, select the subnet to add, and click **Submit** (see Figure A-54).

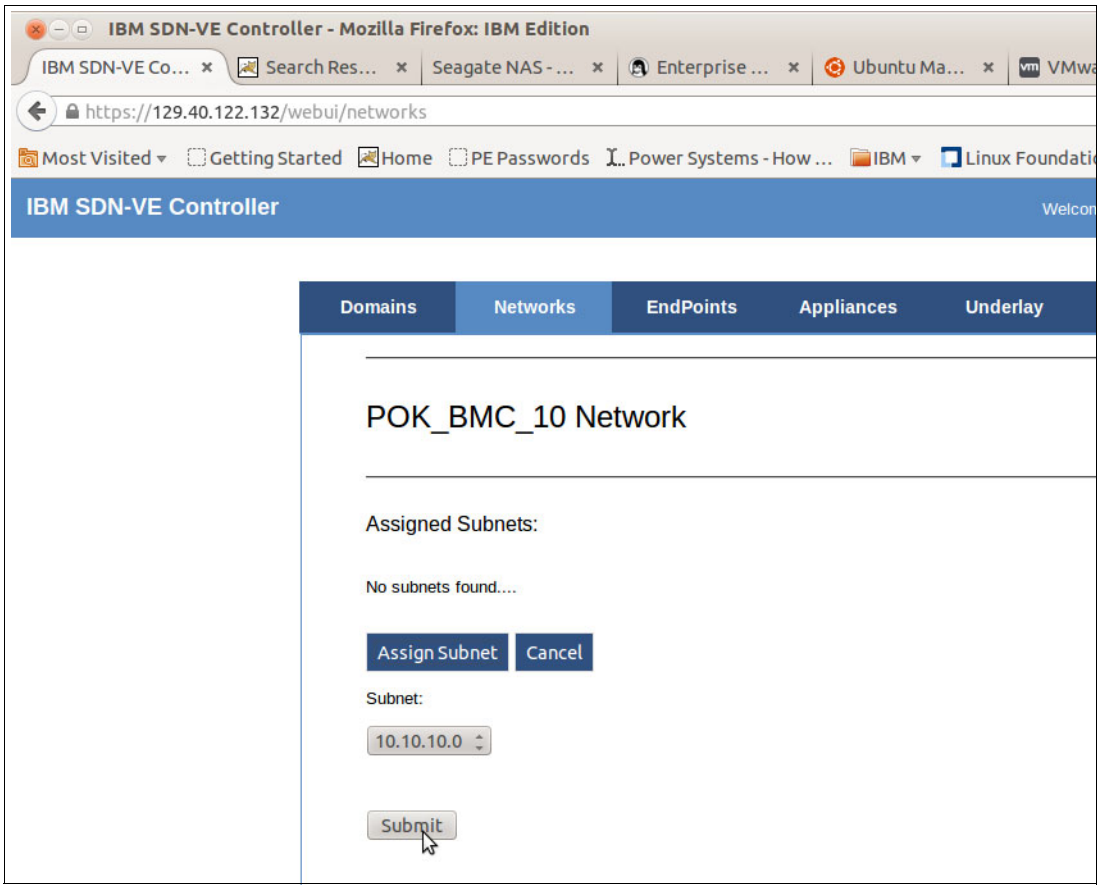


Figure A-54 Select subnet

Configuring gateway definitions

This section shows how to set the IPv4 IP addresses to be used for the DOVE Tunnel and external gateway settings for a gateway.

Adding the IPv4 IP addresses for the DOVE tunnel

1. Click the **Appliances** tab to show the DCS and DGW definitions.
2. Then, select the DGW for which you would like to configure the IPv4 addresses and select **Add IPv4 Interface** as shown in Figure A-55.

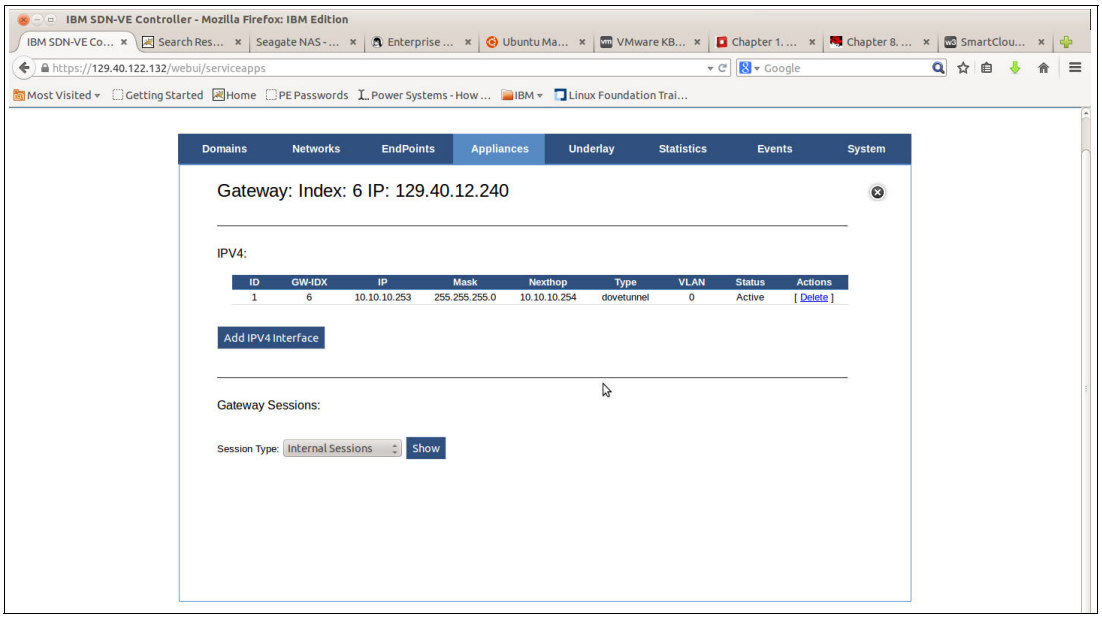


Figure A-55 Adding the IPv4 IP addresses for the DOVE tunnel

3. Complete the IP (overlay gateway IP), Mask (netmask), and Nexthop (gateway) information.

For this IPv4 definition, select **Dove Tunnel** as the type, and then click **Submit** as shown in Figure A-56.

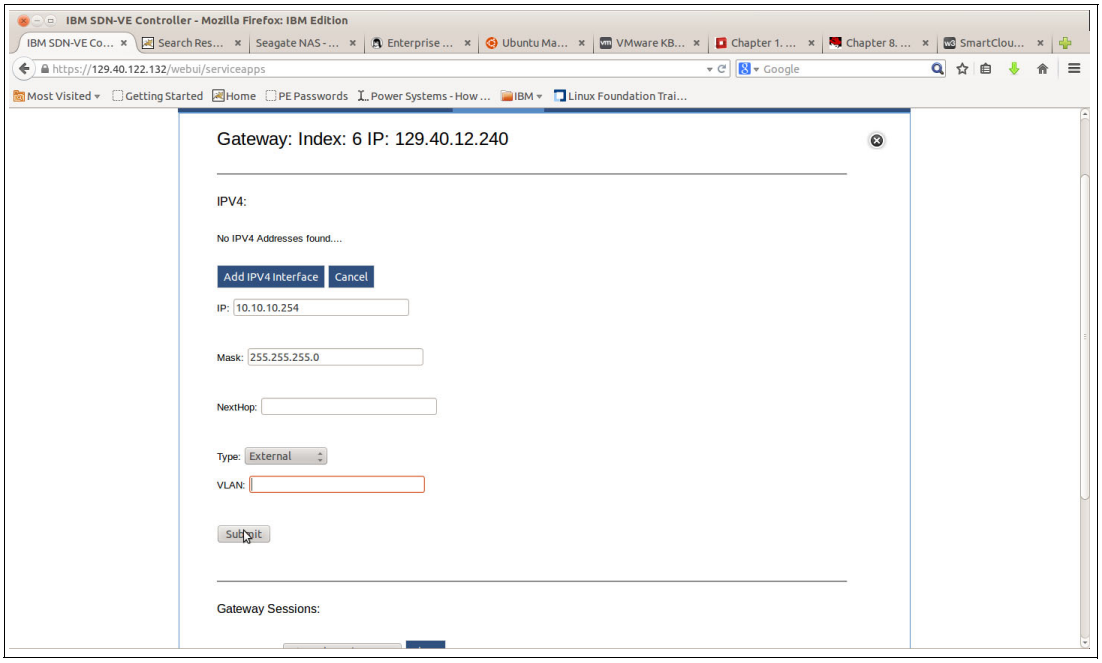


Figure A-56 Complete gateway information

The new IPv4 definition is then displayed as shown in Figure A-57.

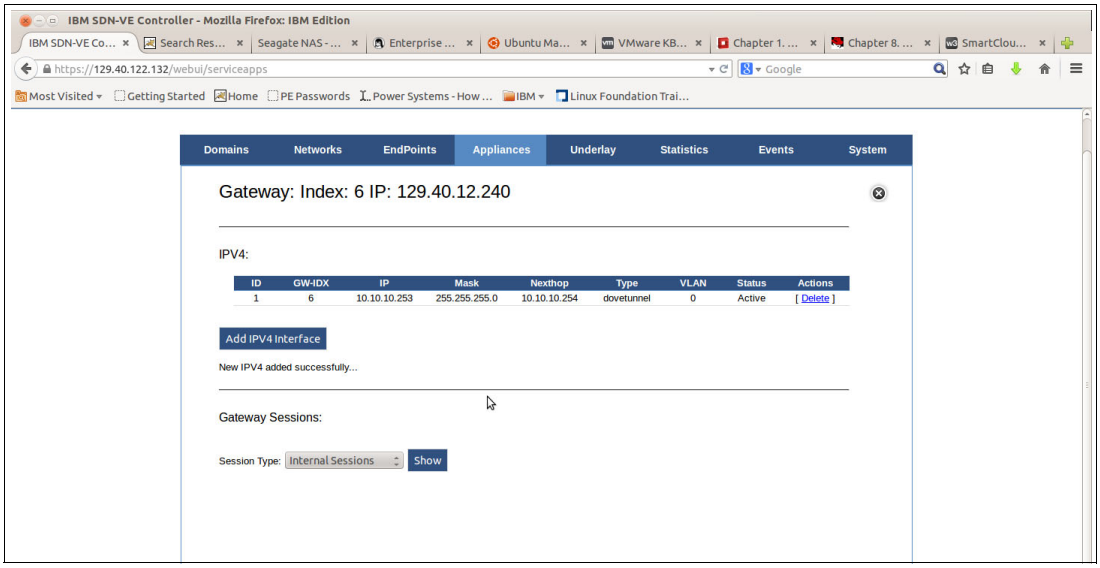


Figure A-57 New IPv4 definition

Adding the IPv4 IP addresses for the external gateway

Click **Add IPv4 Interface** to add the external gateway IP information. This section requires a network range for the NAT IP addresses as well as a port range and the gateway IP address as shown in Figure A-58.

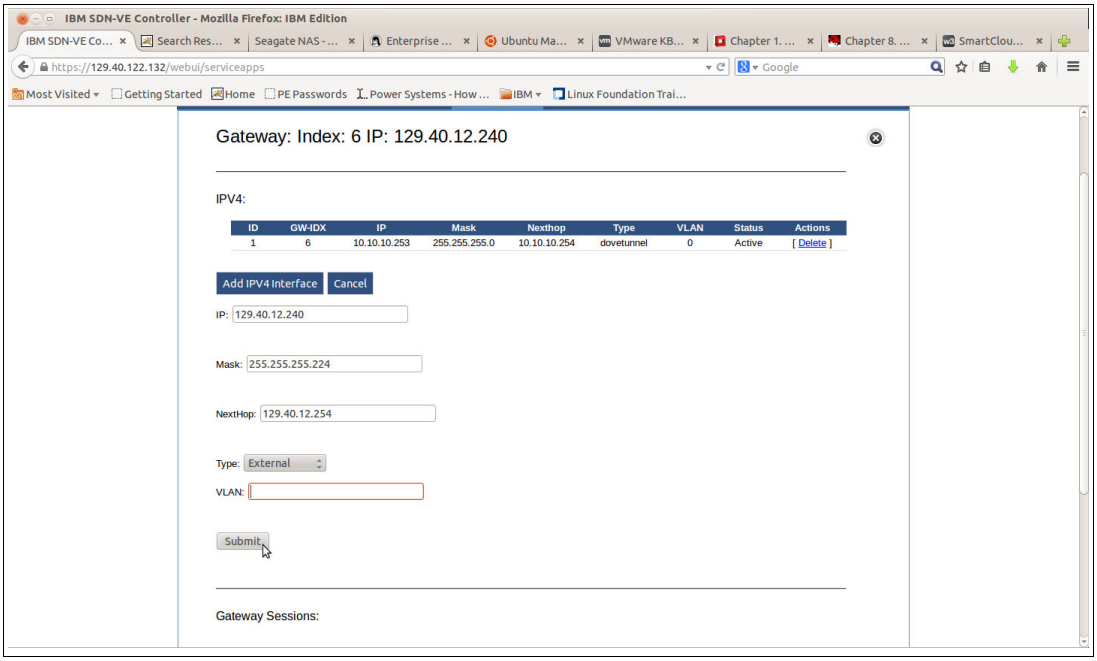


Figure A-58 Adding IPv4 interface

The External Gateway is now displayed as shown in Figure A-59.

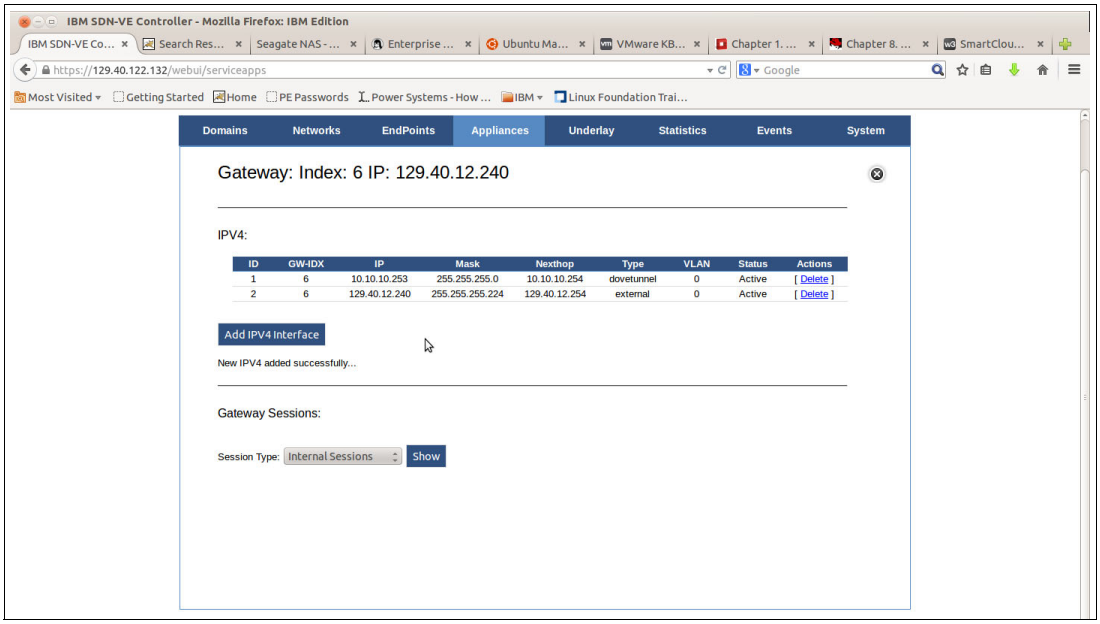


Figure A-59 External gateway

Defining the External Gateway Fwd-Rules

In this section you will set up the port forwarding rules. Click the Domain tab and select the Domain that you would like to add External Gateway FWD-Rules.

Click **Add External/Gateway Port/Forwarding Rule** and enter the Min IP, Max IP, Min Port, Max Port, and the Gateway, and then click **Submit** as shown in Figure A-60.

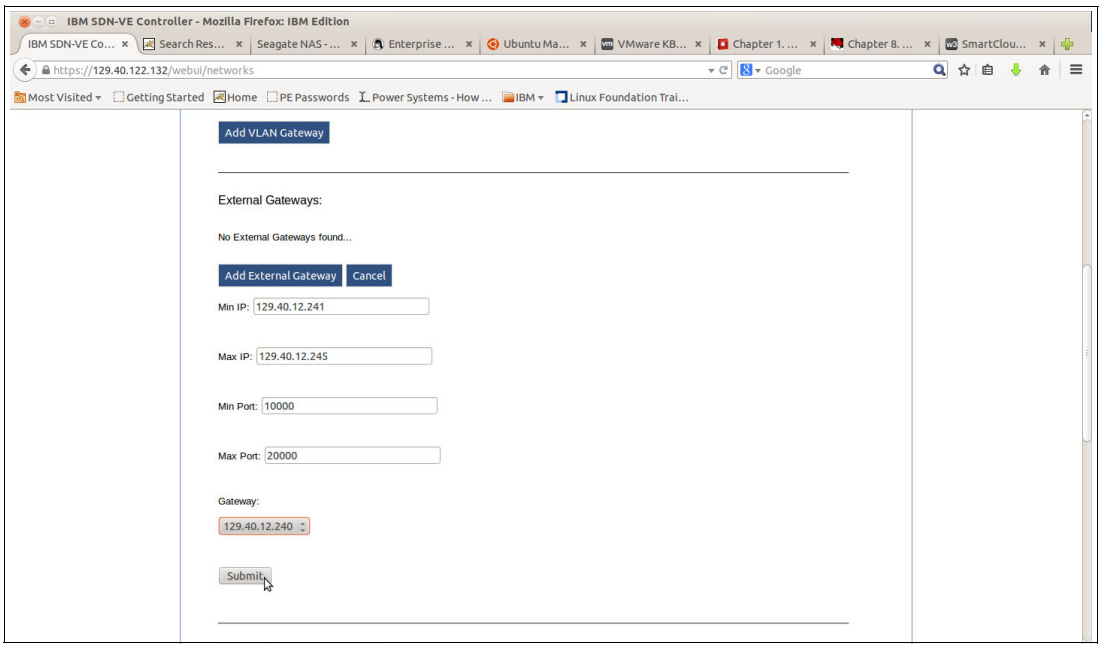


Figure A-60 Add external forwarding rule

Validate the Gateway IPs

1. Open a session for the DGW VM and log in.
2. Then, enter the commands to list the gateway IP addresses, as shown in Figure A-61.

```
[root@f3av03 ~]# ssh admin@129.40.12.240
admin@129.40.12.240's password:
SDN-VE-DSA>enab
enable <cr>
SDN-VE-DSA>enable
SDN-VE-DSA#show ip
ipmgmt                          Show ipmgmt
ipv4-interfaces                 Show IPV4-Interfaces
SDN-VE-DSA#show ipv
ipv4-interfaces                 Show IPV4-Interfaces
SDN-VE-DSA#show ipv4-interfaces
              0: 127.0.0.1
              1: 129.40.12.240
              2: 10.10.10.253
              3: 129.40.12.241
              4: 129.40.12.242
              5: 129.40.12.243
              6: 129.40.12.244
              7: 129.40.12.245
SDN-VE-DSA#
```

Figure A-61 Validate gateway IPs

3. For the sections that follow, see the IBM Redbooks publication titled *Implementing IBM Software Defined Network for Virtual Environments, SG24-8203*:

   http://www.redbooks.ibm.com/abstracts/sg248203.html

   – Defining networks
   – Adding a network
   – Configuring underlay
   – Configuring overlay networks
   – Installing VMs and assigning to the overlay network

B

Cloud computing review

This chapter reviews types of clouds and cloud services and steps for creating them.

This chapter includes the following sections:

- "Types of clouds" on page 776
- "Types of cloud services" on page 781
- "How technical computing fits into cloud deployment models" on page 783
- "Six steps toward cloud computing" on page 788
- "The result: Dynamic cloud through Software Defined Environments" on page 791

© Copyright IBM Corp. 2015. All rights reserved.

Types of clouds

In this section, we compare a hybrid on-premises cloud to public cloud reference architecture examples, using the SDE with OpenStack solution. First, let's make sure that we have a common understanding of what hybrid cloud architecture or environment is.

The National Institute of Standards and Technology (NIST) defines a "hybrid cloud" as:

> "The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (for example, cloud bursting for load balancing between clouds)."[1]

Although this definition sounds like a good starting point, it does not cover all aspects and characteristics of hybrid clouds, especially those of an elastic, dynamic, Software Defined Environment. Therefore, we'll review the possible cloud deployment models and definitions first. Figure B-1 shows the three common cloud deployment models as IBM defines them.

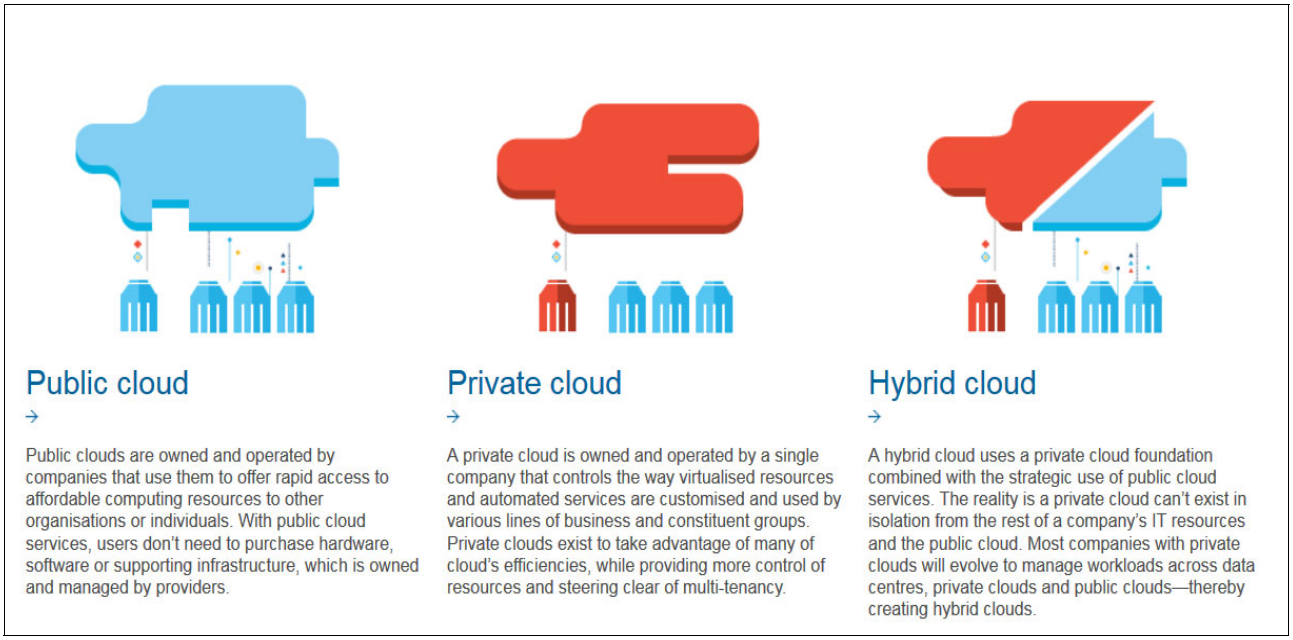


Figure B-1 Cloud deployment models

Public cloud

A public cloud provides standardized services for public use over the Internet. IT and information resources are provided as a service over the Internet, which allows access to technology-enabled services without knowledge of, expertise with, or control over the technology infrastructure that supports them. Usually, it is built on standard and open technologies, providing web page, API, or SDK for those who use the services, which include standardization, capital preservation, flexibility, and time to deploy.

[1] The NIST Definition of Cloud Computing, http://www.nist.gov/itl/cloud/

Many businesses are using software-as-a-service (SaaS) delivered from the public cloud for applications ranging from customer resource management (CRM), such as Salesforce.com, to transaction management and data analytics. Beyond SaaS applications, companies are using other public cloud services, including infrastructure as a service (IaaS) to add more storage or compute services on a moment's notice, and platform as a service (PaaS) for cloud-based application development and deployment environments.

Financial services have a rapidly growing appetite for computing resources that can more flexibly and efficiently deliver services such as conferencing, VoIP, desktop service/help desk WAN capacity and storage. Using a public cloud is the ideal solution for such requirements, where the emphasis is on achieving high use (and therefore low costs).

Private cloud

A private cloud is an architecture where the client encapsulates its IT capacities "as a service" over an intranet for their exclusive use. The cloud is owned by the client, and managed and hosted by the client or a third party. The client defines the ways to access the cloud. The advantage is that the client controls the cloud so that security and privacy can be ensured. The client can customize the cloud infrastructure based on its business needs. A private cloud can be cost effective for a company that owns many computing resources. Key characteristics of private clouds include:

- A self-service interface that controls common services, enabling IT staff to quickly provision, allocate and deliver on-demand IT resources
- Highly automated management of resource pools for everything from compute capability to storage, analytics, and middleware
- Sophisticated security and governance designed for a company's specific requirements to protect critical data and assets

Private clouds are built by an organization for its own users, and everything is delivered within the organization's firewall. The private cloud owner does not share resources with other entities, so sharing the same physical hardware (known as multitenancy) is not a concern. Private clouds are better suited to analytical and transactional workloads such as data warehouses, data mining, business continuity/data recovery, and industry-specific applications, where the emphasis is on performance and where data security is critical.

Hybrid cloud

Customers may integrate a private cloud and a public cloud to deliver computing services. This architecture is called hybrid cloud computing. Evolving to a hybrid cloud strategy will enable companies to keep critical line-of-business applications and sensitive data in a traditional data center environment or private cloud, while also taking advantage of public cloud resources such as SaaS for the latest applications and IaaS for elastic, economical virtual resources to scale.

In a hybrid cloud, the external and internal service delivery methods are integrated. Rules and policies are established by the organization based on factors such as security needs, criticality and underlying architecture, so that activities and tasks are allocated to external or internal resources as appropriate. The use of hybrid clouds enables the balance between cost, security and performance to be optimized.

The ability to efficiently and securely manage the combination of public and private cloud services as a single unified computing environment is the key capability to fully taking advantage of the cloud, and this is where IBM cloud solutions based on SDE and OpenStack can help.

As cloud adoption grows, companies have more and more resources—applications, data and services—residing on different platforms. These days, it is not public versus private. It is both. It is cloud and conventional IT. Hybrid is the new normal. According to IBM Center for Applied Insights™ research, 61% of surveyed enterprises will have a hybrid environment by the end of this year.

Scaling workloads takes planning. “Going viral” just happens. A celebrity mentions your website, or one of the popular news aggregators features you on the front page. It is a need you cannot plan for. Most workloads today cannot and do not scale this easily; they require real people doing real work in advance[2].

Moving steady-state workloads off-premises rarely results in cost savings. The incremental cost of a workload on existing infrastructure is near zero. The data center is already there, power and cooling are present, and often worst case is a one-time capital expense to drop in new hardware. And, in both on-premises and hosted cases, you still need administrators to manage the workloads[3].

Hybrid clouds can help in several ways:[4]

- *Capacity expansion.* When the incremental cost of adding capacity on-premises is high (think upgrading power and cooling in a data center to accommodate additional racks or building a whole new data center), a hybrid approach is a viable alternative. But going hybrid doesn’t have to affect existing operations. Selecting a hosted cloud that supports bare metal and extending the on-premises network could allow existing technologies, tools and techniques to be reused.
- *Development and testing.* Dev and test workloads are highly elastic. They are regularly stood up and torn down, and the number of instances at any one time varies widely based on the development phase. Placing these workloads on the hosted cloud enables you to scale capacity to match demand and pay only for what is used. Another benefit is the ability to tailor the infrastructure to the workload. The next dev/test workload to come along could have radically different hardware requirements.
- *Planned temporary need.* In contrast to bursting, most needs are known in advance: new product launches, holidays, peak season and so forth. And, when given time to plan and execute, most applications can be scaled. Much like dev/test, scaling the workload to the hosted cloud enables tailoring the hardware to the problem being solved and paying for what you need when you use it, turning a large capital expense into a smaller operational expense.
- *Network optimization.* Hosted cloud provides the opportunity shift the heavy lifting of the network off-premises and, in the process, improve the availability, scalability and reliability of the connection by using the provider’s network investment. Connecting branch offices back to the hosted provider rather than a central data center can improve latency by reducing network hops and simplify bandwidth scaling. Pushing the public Internet presence out to the hosted cloud and using their firewalls, intrusion detection systems (IDSes), load balancers, application delivery controllers, VPN terminators and so on can improve security and optimize the bandwidth into the data center.

[2] Michael J. Fork, IBM Thoughts on Cloud: Four Benefits of Hybrid Cloud Computing
http://thoughtsoncloud.com/2014/01/four-benefits-of-hybrid-cloud-computing/
[3] Michael J. Fork, IBM Thoughts on Cloud - Four Benefits of Hybrid Cloud Computing, previously cited
[4] Michael J. Fork, IBM Thoughts on Cloud: Four Benefits of Hybrid Cloud Computing, previously cited

Not all providers are created equal for hybrid, so it is imperative to match the hosted provider to use cases. For capacity expansion, a hosted provider that only supports virtual machines may require significant operational changes, but that same provider could be an excellent choice for dev/test workloads. Network optimization is highly dependent on the hosted provider's quality of service and latency and the breadth of network offerings.[5]

Any hybrid cloud setup has a few challenges that need to be considered during the planning and design phase[6]:

- The most obvious challenge is network connectivity, especially if remote cloud services such as a public cloud or a hosted private cloud are involved. Bandwidth, latency, reliability, and associated cost considerations must be taken into account, and the logical network topology must be carefully designed also (networks, routing, firewalls).
- Another huge challenge is the manageability of different cloud services. When different cloud services are used, every service provider will have its own management and provisioning environment. Those environments can be considered completely independent from each other. By having instances in different cloud services, there is no complete picture available showing the number of totally deployed instances and their statuses.

  An orchestration layer can be a possible solution for this problem. This layer provides a single interface for all cloud-related tasks. The orchestration layer itself communicates with the different cloud services through application programming interfaces (APIs). The big advantage of an orchestration layer with SDE is the ability to track and control activities on a central point to maintain the big picture. Today, plenty of cloud service providers maintain their own proprietary set of APIs. This makes the use of orchestration very complex as the orchestrator requires some kind of a driver component for each proprietary API set. However, the trend of standardized APIs is clearly seen in the industry. OpenStack seems to be the future cloud industry standard.

Hybrid clouds mainly work on an infrastructure and application level. On the infrastructure layer, a hybrid cloud means the combination of virtual machines from different cloud services. On the application or software as a service (SaaS) layer, a hybrid cloud describes an application setup with components in different SaaS offerings or existing applications within the data center of an enterprise. The challenge on a SaaS-based hybrid cloud is mainly the exchange of data between the different services and applications. As orchestration works on the infrastructure level, data integrators work on the application layer.

Dynamic clouds and how they differ from hybrid clouds

What if you approached the hybrid cloud differently? Not as a challenge, but as a potential source of competitive advantage—a way to use the best resources for the task at hand, a way to connect data and applications to create new value, and a way to respond faster to market changes.

We call this the *dynamic cloud*. It's having the strategic agility to dynamically use and deliver cloud-based capabilities in all forms to adapt to rapidly changing business conditions. For IT managers, it's a new way of designing and orchestrating IT resources. For business leaders, it opens a new—and growing—set of options for innovation. The SDE with OpenStack solution aligns and fits in with dynamic cloud.

[5] Michael J. Fork. IBM Thoughts on Cloud - Four Benefits of Hybrid Cloud Computing http://thoughtsoncloud.com/2014/01/four-benefits-of-hybrid-cloud-computing/

[6] Markus Eber. IBM Thoughts on Cloud- What is hybrid cloud?

A dynamic cloud provides choice and flexibility. Businesses can pick the best application for the job, regardless of delivery platform. Data can be located wherever regulatory or security requirements dictate, yet remain accessible. Computing workloads are placed where they fit best based on resource availability, operational costs, data location and a host of other factors. And they move when situations change. Following and keeping a broad view on cloud-related reference architectures is important because it helps evolve the solution architectures and parallel and because it helps enterprises upgrade the policies around service level agreements (SLAs), security, interaction contracts with partners, and many other areas.

A hybrid environment that's dynamic offers greater capacity and efficiency. At peak use times, companies can mix dedicated resources with public clouds to provide extra capacity – they do not have to buy more on one platform when there is some already paid for on another. For example, businesses may want to quickly tap temporary capacity for big data analytics processing or other high-performance computing needs. Users aren't aware where the computing happens; they just know their results are coming back faster. Organizations have the same level of visibility, control and automation whether that computing happens on- or off-premises.

Dynamic cloud also brings a new level of integration and composability. Data can be shared seamlessly across platforms. Enterprises can more easily integrate next-generation application platforms—for mobile phones and all sorts of intelligent devices—with their existing systems. By assembling ready-to-use services, on various cloud platforms, enterprises will be able to innovate more rapidly. Launch new digital products and services. Build new applications to engage customers. Capitalize on big data. Even implement new business models. In effect, dynamic cloud powers a composable business—one characterized by restless reinvention, analytics-driven decisions, and speed, and the capability to use an API economy and ecosystem fostered by an open cloud architecture which extends the business services to third-party innovators.

As illustrated in Figure B-2 on page 781, the future will be a dynamic hybrid, because that combines the best of both worlds and leads to better outcomes. For companies with large IT investments, the hybrid approach enables you to maximize its return on existing IT investments by transforming IT into self-service private clouds. The hybrid approach also enables enterprises to keep some workloads on a best fit infrastructure and move other workloads to public cloud for better economics. With the hybrid approach, you can keep critical, sensitive data in-house while innovating in the public domain so that you have the correct balance of risk to speed. For example, you can connect ERP to public CRM to enhance client experience. The hybrid approach also enables enterprises to extend current IT with private clouds with public capabilities to meet seasonal capacity requirements (for example, holiday shopping), without the need for additional capital expenditure. Also, the hybrid approach enables enterprises to set up development environments quickly for innovation, to add new capabilities, such as analytics or mobile SaaS, on the public domain quickly, and to compose more competitive business services and processes.

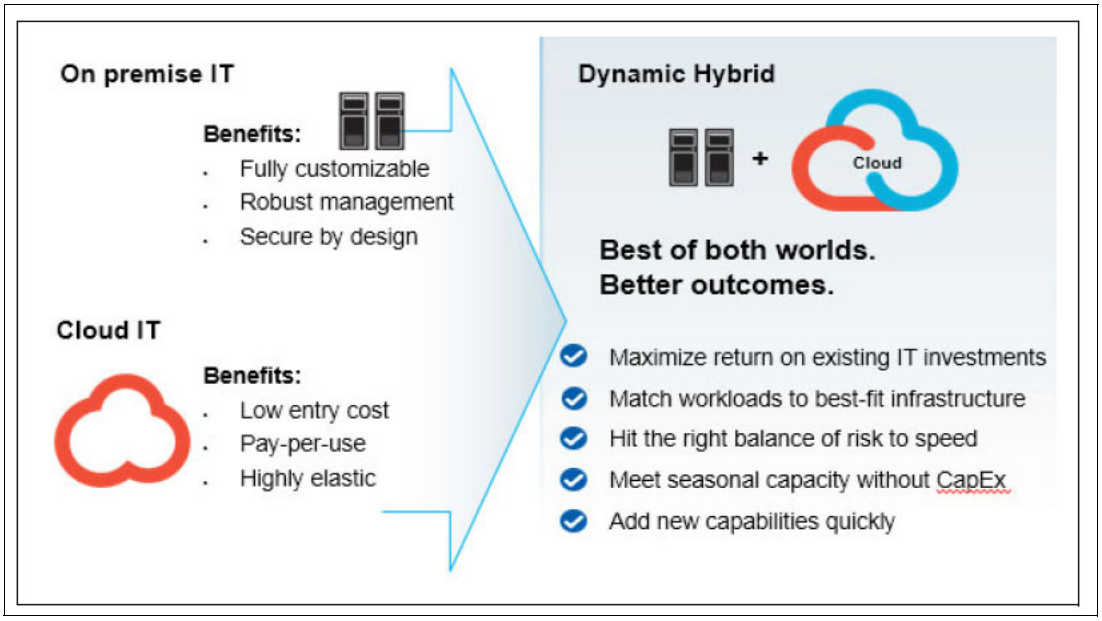


Figure B-2 Dynamic hybrid topologies

Types of cloud services

OK, but now (as a refresher to some), how do we know what cloud computing typically offers as services that you should take advantage of? This is an important question, but the answer depends on the business and IT strategy. To begin, the thinking ties back to the beginning of this chapter, with the 2.2, “The Enterprise Technology Framework” on page 49 as a starting point and an example of an approach. Then, the approach to determine which services might be cloud-based starting as aligned with the following cloud computing service choices, as shown in Figure B-3.

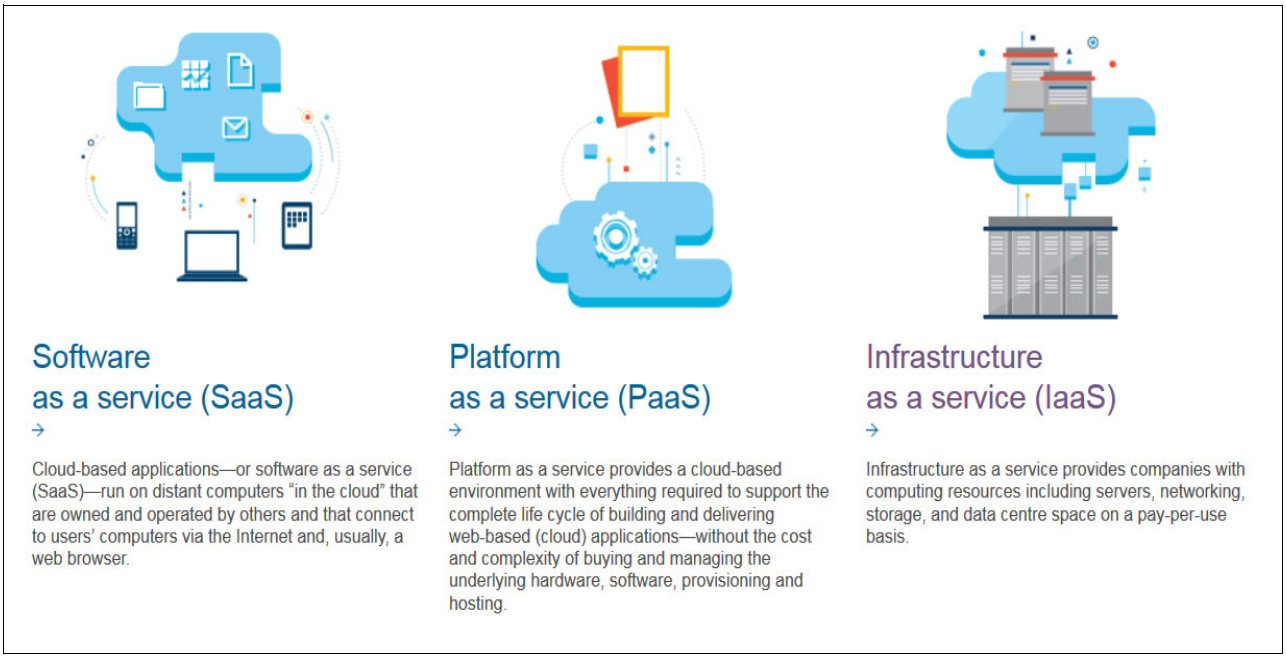


Figure B-3 Cloud computing services

Software as a service (SaaS)

There are several advantages of SaaS cloud applications. First, there's no software to purchase, install, update or maintain, which is handled by the service providers. You also get these benefits:

- You can sign up and rapidly start using cloud applications.
- Applications and data are accessible from any connected computer.
- No data is lost if your computer breaks, because data is in the cloud.
- The service can dynamically scale to the use needs of your organization.

The IBM software-as-a-service portfolio for enterprises includes offerings for social business (US), business process management (BPM), business analytics (US), web analytics (US), and more. SaaS is typically aligned with public off-premises clouds.

The decision for which computing services to host in a SaaS model are varied and wide, depending on both functional and non-functional qualities and requirements such as security, data integrity and privacy, performance and scalability, disaster recovery, service resilience and high availability, and management/manageability (including testing and monitoring). Integration to SaaS should be considered between off-premises public clouds and systems of engagement to on-premises private clouds and systems of record.

Platform as a Service (SaaS)

You can use this one for these purposes:

- Develop applications and get to market faster
- Deploy new web applications to the cloud in minutes
- Reduce complexity with middleware as a service

The platform-as-a-service offering, IBM SmartCloud Application Services, enables users to deploy and migrate applications to both public and private clouds. IBM Bluemix is another option:

`http://www.ibm.com/software/bluemix/`

The key benefit here is that PaaS is being used to integrate the off-premises (often public) systems of engagement to the on-premises systems of record often through very easy to develop, use and use APIs.

Infrastructure as a service (IaaS)

The definition of *infrastructure as a service (IaaS)* is pretty simple: You rent cloud infrastructure—servers, storage and networking—on demand, in a pay-as-you-go model. OpenStack is essentially IaaS, and the SDE solution joins and integrates SaaS, and PaaS to IaaS for heterogeneous infrastructure that can be rented, or hosted in a private cloud or as a managed service, for example.

The IT Industry has tried to maintain a balance between more demands from business to deliver services against cost considerations of hardware and software assets. Business growth depends on information technology (IT) being able to provide accurate, timely, and reliable services, but there is a cost that is associated with running IT services. The main challenge of IT is to provide timely, reliable, and accurate services for business needs, which have led to the growth and development of Software Defined Environments, software-defined infrastructure, and high performance computing (HPC). HPC traditionally has been the domain of powerful computers (called "supercomputers") that are owned by governments and

large multinationals. Existing hardware was used to process data and provide meaningful information by single systems working with multiple parallel processing units. Limitations were based on hardware and software processing capabilities. Because of the cost that was associated with such intensive hardware, the use was limited to a few nations and corporations.

The advent of workflow-based processing model and virtualization and the high availability concepts of clustering and parallel processing have enabled existing hardware to provide the performance of the traditional supercomputers. New technologies, such as graphics processing units (GPUs), has pushed existing hardware to perform more complicated functions faster than previously possible on the same hardware, while virtualization and clustering have made it possible to provide greater level of complexity and availability of IT services.

Sharing resources to reduce cost is also possible because virtualization moved from a traditionally static IT model that is based on maximum load sizing to a leaner IT model that is based on workflow-based resource allocation through smart clusters. With the introduction of cloud technology, the resource requirement is becoming more on demand as compared to the traditional forecasted demand, thus optimizing cost considerations.

These technological innovations have made it possible to push the performance limits of existing IT resources to provide high performance output. The technical power to achieve computing results can be achieved with much cheaper hardware by using smart clusters and grids of shared hardware. The workflow-based resource allocation has made it possible to achieve high performance from a set of relatively inexpensive hardware working as a cluster. Performance can be enhanced by breaking across silos of IT resources, lying dormant to provide on-demand computing power wherever required. Data-intensive industries, such as engineering and life sciences, can now use the computing power on demand that is provided by the workflow-based technology. Using parallel processing by heterogeneous resources working as one unit under smart clusters, complex unstructured data can be processed to feed usable information into the system.

Technical computing has seen demand from sectors such as aerodynamics, automobile design, engineering, financial services, and oil and gas Industries. Improvement of cooling technology and power management of these superfast computing grids have allowed more efficiency and performance from existing hardware. Increased complexity of applications and demand for faster analysis of data has led technical computing to become mainstream or widely available. Therefore, IBM Technical Computing is focused on helping clients transform their IT infrastructures to accelerate results. The goal of having technical computing in mainstream industries is to meet the challenges of applications that require high performance computing, faster access to data, and intelligent workload management, and make them easy to use and access from different applications with high performance outputs according to business demands.

How technical computing fits into cloud deployment models

A cluster is typically an application or set of applications whose primary aim is to provide improved performance and availability at a lower cost compared to a single computing system with similar capabilities. A grid is typically a distributed system of homogeneous or heterogeneous computer resources for general parallel processing of related workflow, which is usually scheduled by using advanced management policies. A computing cloud is a system (private or public) that allows on-demand self-service, such as resource creation on demand, dynamic sharing of resources, and elasticity of resource sizing that is based on advanced workflow models.

IBM Platform Computing solutions have evolved from cluster to grid to cloud because of their abilities to manage heterogeneous complexities of distributed computing resources. IBM Platform Computing provides solutions for mission-critical applications that require complex workload management across heterogeneous environment for diverse industries, from life sciences to engineering to financial sectors involving complex risk analysis. IBM Platform Computing has a 20-year history of working on highly complex solutions for some of the largest multinational companies, where there are proven examples of robust management of highly complex workflow across large distributed environments to deliver results.

Business value: Technical computing and HPC clouds

IBM HPC clouds can help enable the transformation of both the client's IT infrastructure and its business. Based on an HPC cloud's potential impact, clients are actively changing their infrastructure toward private clouds, and beginning to consider public and hybrid clouds. Clients are transforming their existing infrastructure to HPC clouds to enhance the responsiveness, flexibility, and cost effectiveness of their environment. This transformation helps a client enable an integrated approach to improve computing resource capacity and to preserve capital.

However, In a public cloud environment, HPC still must overcome several significant challenges, as shown in Table B-1.

Table B-1 Public cloud challenges

Public cloud challenges	Details
Security	► Cloud providers do not provide guarantees for data protection. ► IP is in-flight outside the firewall and on storage devices.
Application licenses	► Legal agreements (LTUs) can limit licenses to geographies or corporate sites. ► Unlimited licenses can be more expensive.
Business advantage	► Cloud resources are expensive compared to local resources if they are used incorrectly. ► Building and automating a business policy for using cloud can be difficult.
Performance	► If applications run poorly in a private cloud, the applications do not improve in public clouds.
Data movement	► Data must be replicated in the cloud before jobs can run. ► Providers charge for data I/O and storage.

If you use private clouds, HPC might not suffer from the challenges of public clouds, but it can have a common set of existing issues, as shown in Table B-2.

Table B-2 Private cloud challenges

Private cloud challenges	Details
Inefficiency	► Less than fully used hardware. ► High labor cost to install, monitor, and manage HPC environments. ► Constrained space, power, and cooling.
Lack of flexibility	► Resource silos tied to a specific project, department, or location. ► Dependency on specific individuals to run technical tasks.
Delayed time to value	► Long provisioning times. ► Limited ability to fulfill peak demand. ► Constrained access to special purposes devices (for example, GPUs).

Therefore, many IBM clients might integrate a private cloud and a public cloud to deliver optimal computing services. This architecture is called *hybrid cloud computing*, and Figure B-4 highlights the differences and relationships of these three types of clouds from a business model perspective.

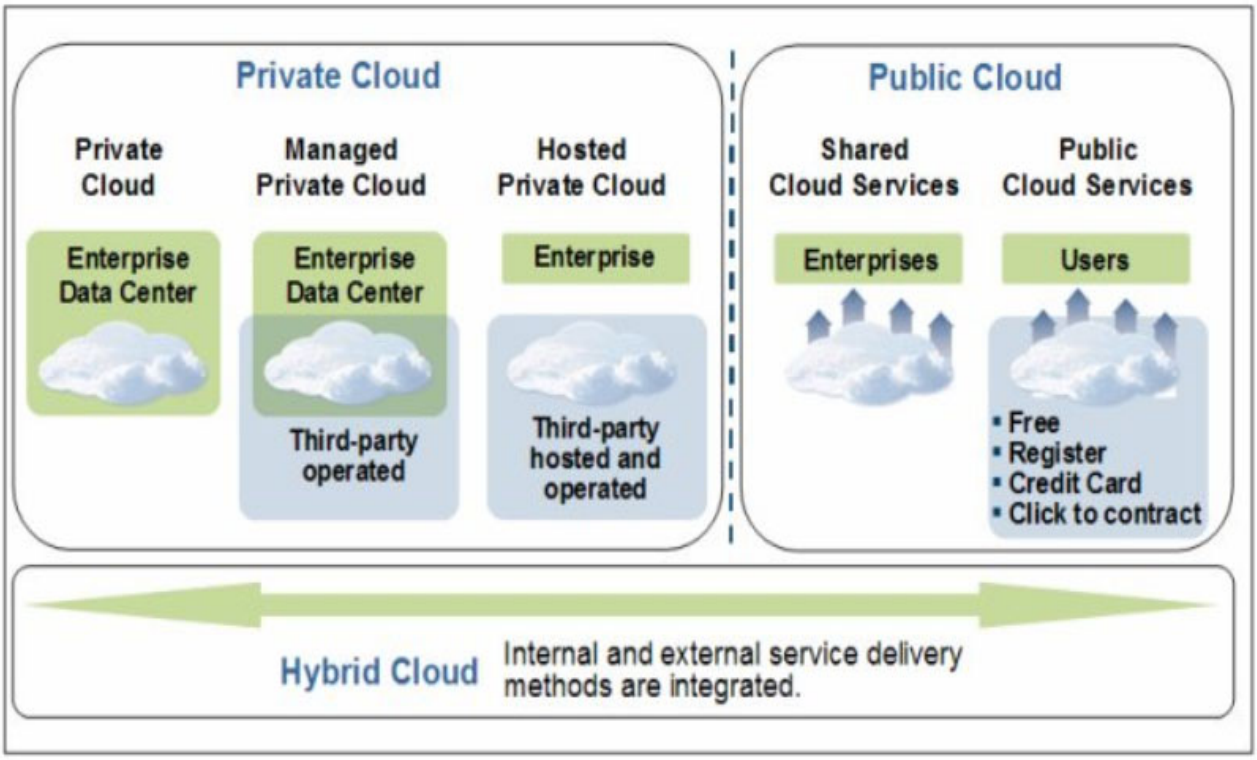


Figure B-4 Business model - Spectrum of deployment options for cloud

Figure B-5 shows the HPC reference cloud model.

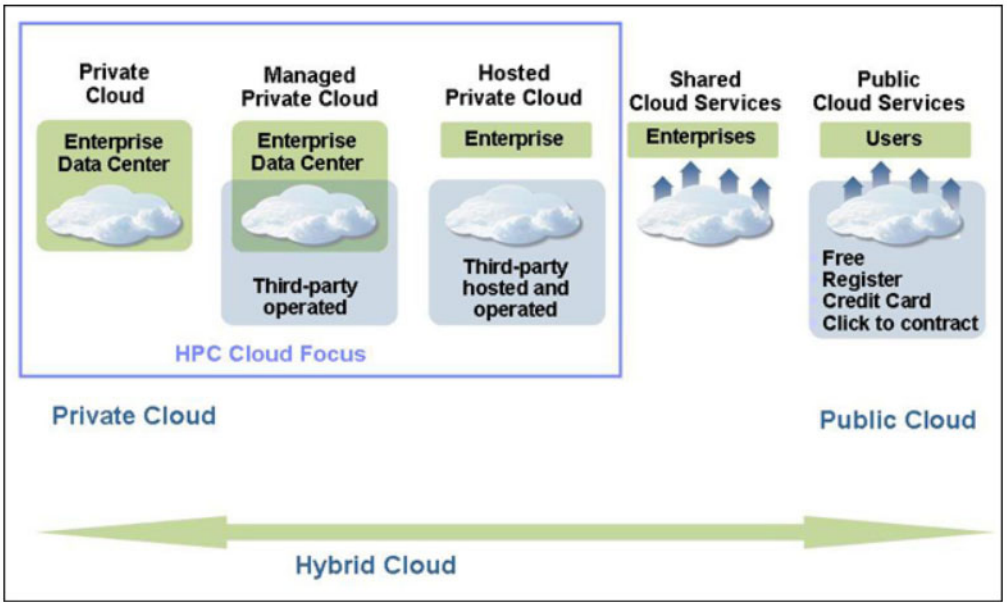


Figure B-5 HPC reference cloud model

The HPC private cloud has three hosting models: private cloud, managed private could, and hosted private cloud. Table B-3 provides the characteristics of each of these models.

Table B-3 HPC private cloud model

HPC private cloud model	Characteristics
Private cloud	Client self-hosted and managed
Managed private cloud	Client self-hosted, but third-party managed
Delayed time to value	Hosted and managed by a third party

For more information about IBM Technical Computing, Platform Computing and HPC clouds, see the following web pages and publications:

- *Introduction to IBM Technical Computing Clouds*, TIPS1023

  http://www.redbooks.ibm.com/abstracts/tips1023.html
- IBM Platform Computing

  http://www.ibm.com/systems/platformcomputing/
- *IBM Platform Computing Solutions Reference Architectures and Best Practices*, SG24-8169

  http://www.redbooks.ibm.com/abstracts/sg248169.html

IBM enables scalable, hybrid integrated cloud through its open cloud architecture. This means that our cloud architecture is based solidly on a composition of open source components and where possible, implements open standard formats, APIs, and protocols that deliver both interoperability and value to the customer. This open architecture has a number of benefits, both to IBM, as well as to our customers. For IBM, we benefit from the unparalleled innovation we find in open source communities such as OpenStack. Additionally, we benefit from the vibrant ecosystem of ISV partners that grow from these open source and open standards initiatives. Our customers benefit from the ability to use the open source and open standards APIs and to extend them—either through third-party capability (using the vibrant open source ecosystems we have chosen), or through innovation and extension of their own.

By using the same open cloud architecture and technologies across our cloud portfolio, addressing private, public, and hybrid cloud solutions, we also benefit from interoperability and portability of workloads and of capability between and across these various offerings, which benefits our clients.

Figure B-6 illustrates the IBM open cloud architecture by the three “layers” of computing services (SaaS, PaaS, IaaS) and some of the various open cloud technologies that IBM uses and supports in the architecture.

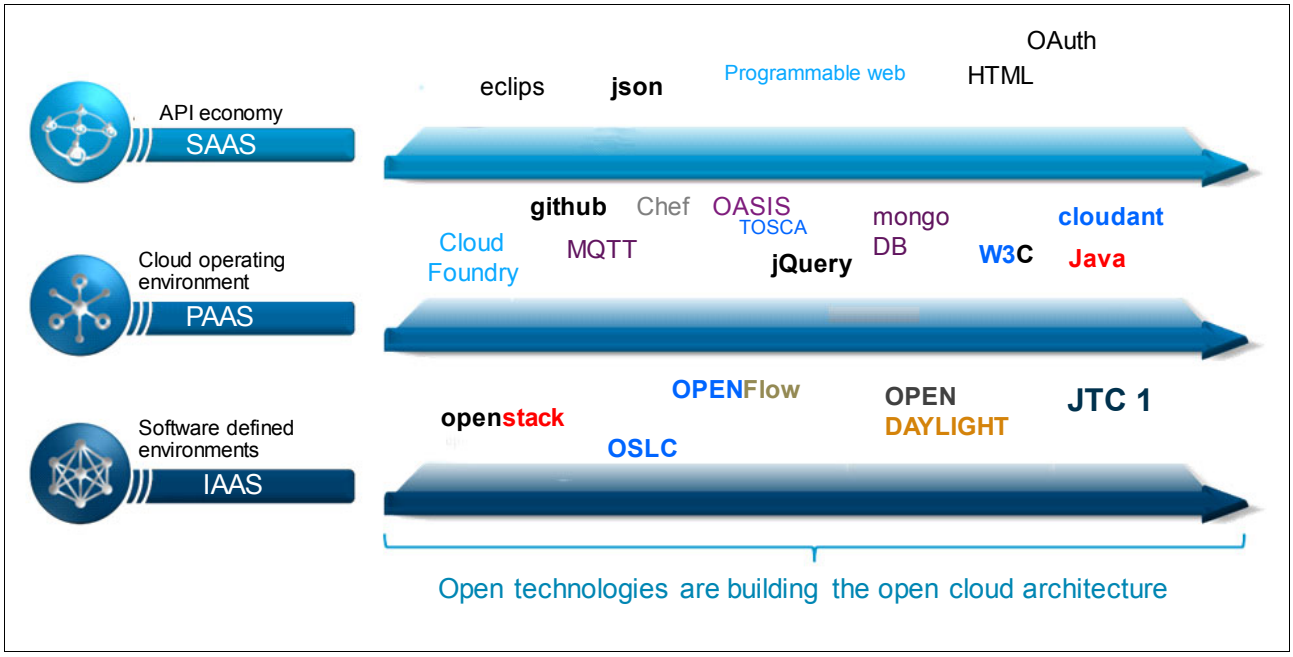


Figure B-6 Open cloud architecture

In fact, customers who choose open standards-based cloud computing are on the right course to take advantage of new opportunities. Specifically, they will be able to harness their interconnected data with high-value business analytics across traditional and mobile devices.

Important: For more information about the IBM open cloud architecture and vision, see the introduction in the IBM developerWorks technical library:

`http://ibm.co/1DChRlB`

Cloud workload adoption and deployment model choice

So, now that we have explained the various cloud computing services and deployment model options, how do you decide on what workloads to move to the cloud and where? Remember that there is not a hard a fast rule for only selecting one type of computing service, and only one type of deployment model. Based on specific workload demands and requirements, and as supported by SDE and OpenStack, you can have a hybrid cloud that is dynamic to respond to the unique needs of your business.

Figure B-7 illustrates the enterprise design considerations in terms of cloud workload adoption, by workload type.

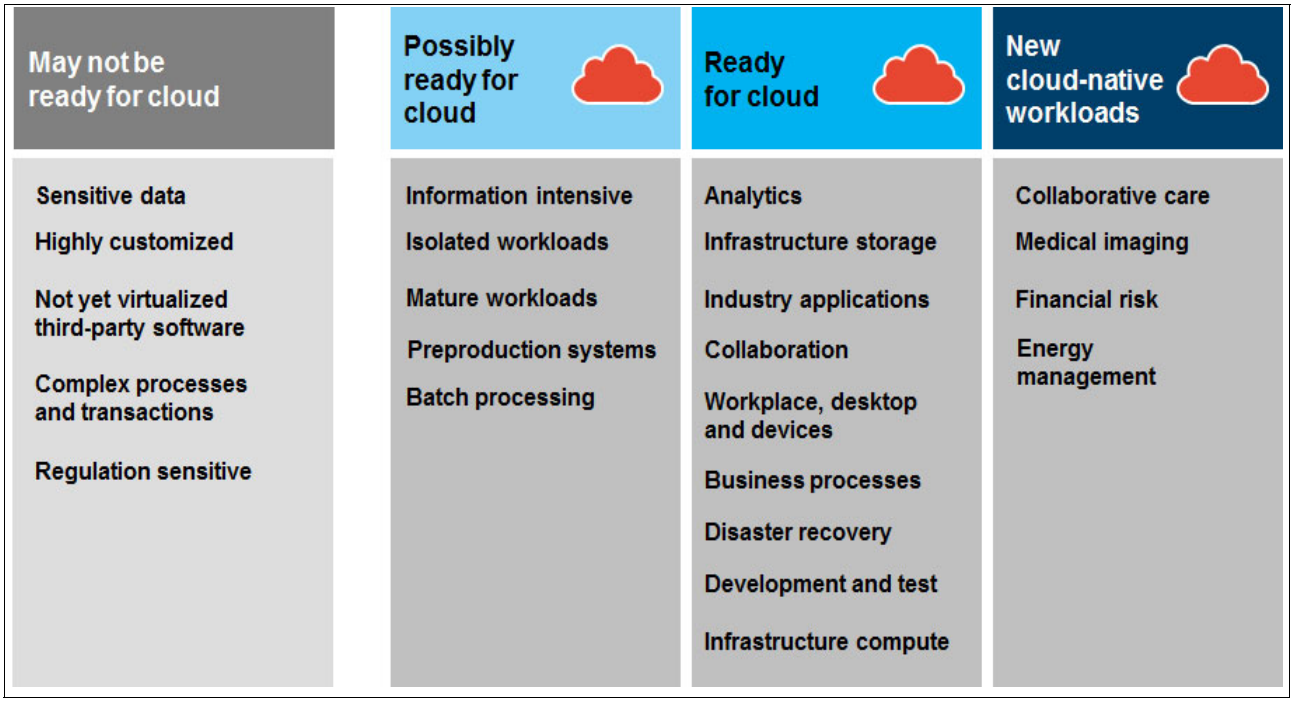


Figure B-7 Enterprise cloud design considerations - Workload type

Six steps toward cloud computing

Defining a cloud adoption strategy and following the correct pathway to adoption are essential to success. This can be the most challenging phase of the adoption process, and it is the stage at which most survey respondents said they would be likely to request outside assistance. Beyond the strategic vision, successful deployment of cloud technology will require a pragmatic, evolutionary approach—one that IBM has developed through its many cloud engagements with customers around the world—as well as the many cloud projects within IBM. Today, IBM is helping organizations get started on the journey toward cloud computing by following six strategic steps to cloud adoption.

Create the IT strategy and roadmap

Begin by determining the goals for the organization regarding cloud service delivery, and the platform requirements and complexity associated with each. The 2.2, “The Enterprise Technology Framework” on page 49 is an excellent starting point as an example approach to identify the enterprise services. The approach to determine which services may be cloud-based is described in 2.3, “Approaches and methods” on page 52.

Goals might include the following, among others:

- Centralization to reduce costs and number of data centers
- Consolidation of servers and storage to reduce capital costs, operational costs, software license fees, and the carbon footprint
- Virtualization to increase hardware use and standardize virtual images
- Identification and rationalization of services
- Automation of service management processes and application migration
- Optimization of resources and dynamic provisioning of services

Assess and select workloads for the cloud

Identifying and prioritizing workloads that are candidates for cloud delivery is the next step. Workloads that are amenable to standardization, that are self-contained applications, or that have a service-oriented architecture will potentially be easier to move into a cloud environment. Conversely, workloads that will be more difficult, risky and costly to migrate to a cloud environment are typically highly complex, require high amounts of data transfer, involve a high degree of customization or are existing systems that would require significant redesign to conform to cloud architecture.

Figure B-8, provides an overview of workloads in terms of standardization and cloud workload adoption, from the perspective of systems of record versus systems of engagement, customization versus standardization, relative costs, and time to value for ROI.

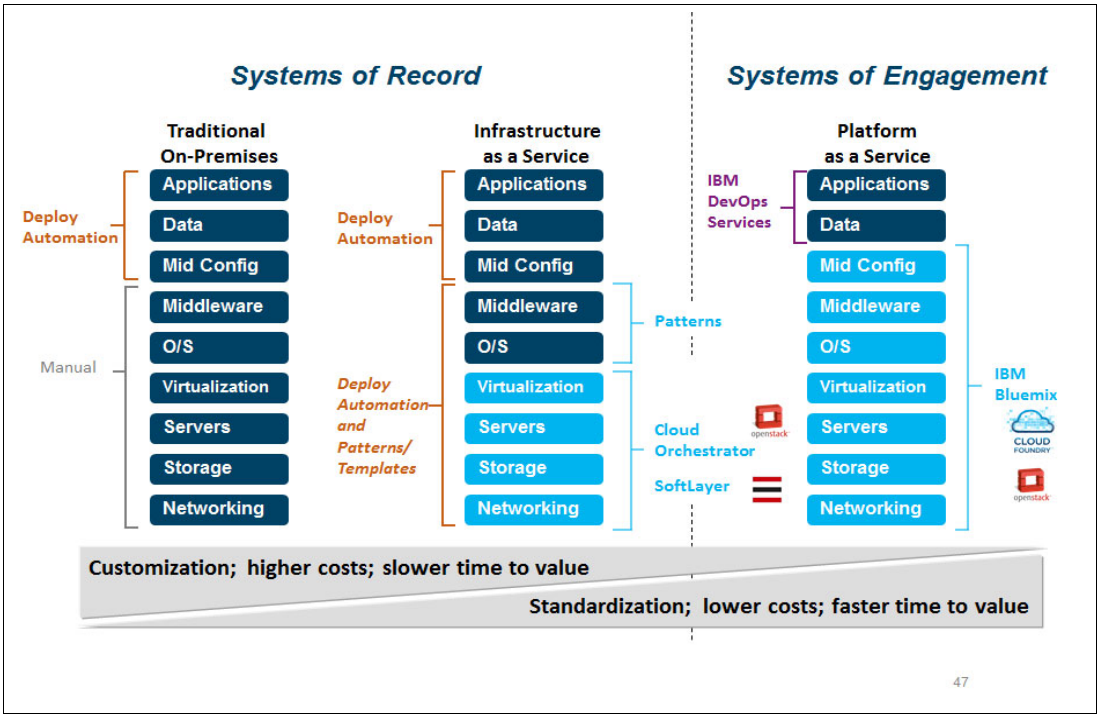


Figure B-8 Workloads and standards through cloud adoption

Determine the cloud delivery model

After you have identified candidate workloads, you can then establish which are best suited for delivery via a public cloud, a private cloud, or a hybrid of both. As an example, for financial services firms suggests that data mining, security and transactional databases are the workloads with the highest push factors and lowest barriers for delivery in a private cloud. As an example, for public cloud delivery, the IBM suggests that conferencing, voice-over- IP infrastructure and WAN capacity have the highest push factors and lowest barriers for financial services firms.

Determine the value

Examine the ROI that cloud computing can bring, including time required for initial payback as well as projected ROI. In determining ROI, start by calculating the absolute savings that will be realized from all elements of IT operations related to the workload in question—including, for example, hardware costs, software licenses and upgrades, system administration, system support, user support and provisioning. ROI calculations can also include business-related measures, such as increases in user productivity and resource use; avoidance of capital expense; and reduction of risk due to higher availability.

In addition to looking at first-year savings, IBM recommends that clients project operating costs of both the existing IT environment and the cloud environment over three years to calculate ROI over a longer term. Our research has shown that, although all IT costs may increase over time, typically cloud computing costs will increase at a lower rate—resulting in increasing value.

Establish the architecture

The final step before implementation is to establish the architecture that will support your cloud initiatives, public, private or dynamic hybrid. A cloud architecture should address three "domains:"

- What are the services you will deliver or acquire? These include infrastructure as a service ("raw" computing capacity), platform as a service (computing capacity plus network services and data management services), or software as a service (computing capacity plus network/data management services plus applications).
- How will you create and deliver the services? This includes the tools, procedures and governance required to plan, define, catalog, configure, deliver, monitor, measure, bill and report on cloud-delivered services.
- How will users access the services? A service catalog that enables users to select, order and configure cloud-delivered services is an essential component of a cloud architecture. In addition, for private clouds, the architecture should provide an operational console for service delivery and operational managers.

Underlying all three domains is the cloud platform: The architecture layer that defines the SDE dynamic infrastructure for cloud delivery, as well as the service management disciplines that enable service delivery and user access.

Implement the IT strategy, roadmap, and cloud services

A key step in preparing for cloud is conditioning your infrastructure for cloud delivery. This may include virtualizing and automating existing systems, adding the service management capabilities that are required for cloud computing.

The roadmap for cloud implementation defines a pilot project or projects that will enable the IT organization to gain experience with cloud computing delivery while users gain experience of working with standardized applications and services delivered over the network. This pilot could be delivered from a private cloud platform, or it could be a pilot using trusted public cloud services. Either way, the pilot should deliver an isolated, low-risk workload.

Building on lessons learned, the next steps in the roadmap will be to enable additional workloads and extend cloud services to new users. A well-designed cloud environment will typically aim to meet user needs through a single portal, using a catalog of services that can be automatically provisioned in many cases.

The result: Dynamic cloud through Software Defined Environments

The IBM open architecture and technology enables innovation to provide the best "fit for purpose," based on industry-standard interoperability, for IBM clients who are implementing cloud computing. The approach outlined enables using existing business and IT initiatives and existing investments from consolidation and virtualization (as in many cases, already funded) starting points and building blocks through to intelligent, SDE-based dynamic hybrid cloud services, as shown in Figure B-9 on page 792.

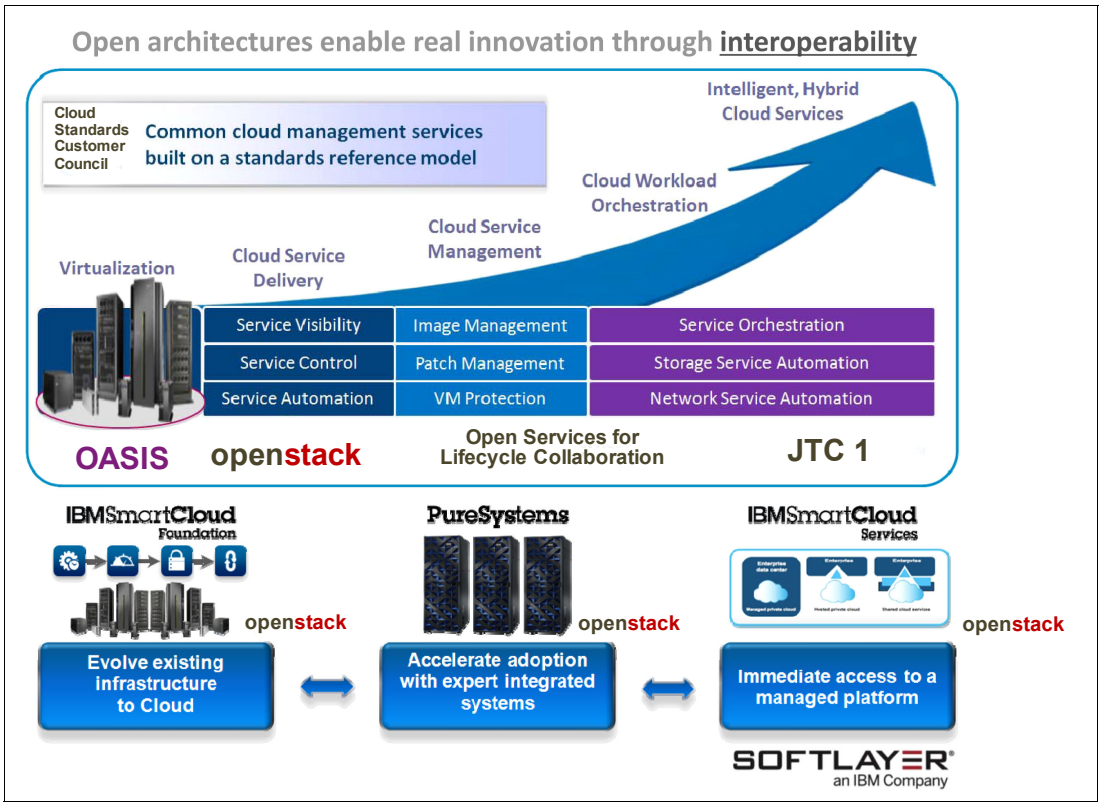


Figure B-9 Dynamic cloud through Software Defined Environments

Dynamic hybrid example SDE with OpenStack reference architecture

Figure B-10 illustrates a dual SDE on-premises and off-premises dynamic hybrid architecture with IBM SoftLayer.

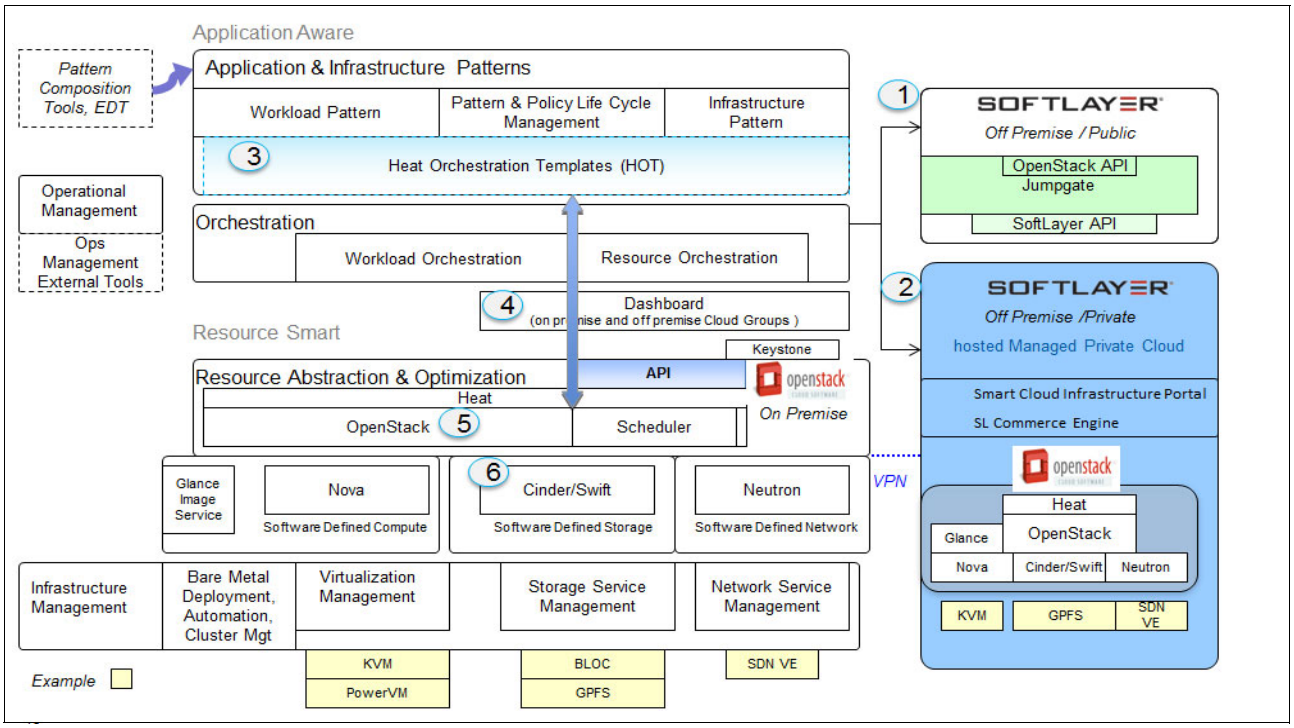


Figure B-10 SDE on-premises and off-premises dynamic hybrid cloud architecture with SoftLayer

Key to Figure B-10:

1. On-premises and off-premises OpenStack environments are deployed and configured to support a hybrid cloud based on client requirements.

2. The off-premises Hosted Managed private cloud with SoftLayer and OpenStack in this case can be for specific private customer workloads (for example, Dev/Test, HPC, and so on)

3. HOT templates define workload and infrastructure patterns aligned with location requirements.

4. Deployment occurs via OpenStack API or Jumpgate, the dashboard, or Cloud Manager. There is also a VPN network connection between the on-premises private cloud and off-premises managed private cloud.

5. HEAT policy engine introspects HOT template, and allows policy parameters to be passed.

6. Resource services are based on policies (in this example, storage type).

Related publications

The publications listed in this section are particularly suitable for more detailed information about the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information. Some publications referenced in this list might be available in softcopy only.

- *IBM PowerKVM Configuration and Use*, SG24-8231

  http://www.redbooks.ibm.com/abstracts/sg248231.html
- *IBM PowerVC Version 1.2.1 Introduction and Configuration, SG24-8199*

  http://www.redbooks.ibm.com/abstracts/sg248199.html
- *IIBM SAN Volume Controller 2145-DH8 Introduction and Implementation, SG24-8229*

  http://www.redbooks.ibm.com/abstracts/sg248229.html

You can search for, read, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

For further information, see “Installing Platform Resource Scheduler” in IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SS8MU9_2.2.0/prs_kc/prs_kc_installing.dita

Online resources

These websites are also helpful:

- IBM Cloud Manager with OpenStack User Guide

  http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html
- IBM Cloud Manager with OpenStack Administrator Guide

  http://www.ibm.com/support/knowledgecenter/SST55W_4.1.0/liaca/liaca_pdfs.html
- IBM Cloud Manager with OpenStack 4.1 Download Trial Quick Start Guide

  http://ibm.co/1bCYa2c
- Deploying a big data solution using IBM GPFS-FPO

  http://public.dhe.ibm.com/common/ssi/ecm/en/dcw03051usen/DCW03051USEN.PDF
- IBM Installation Toolkit for PowerLinux User's Guide for Version 5.6

  http://ibm.co/1Nk28KL

© Copyright IBM Corp. 2015. All rights reserved.

- IBM Blueprints Quick Start Guide for installing and running KVM

  http://ibm.co/1ChwVTv
- Configuring IBM PowerKVM on Power Systems

  http://www.ibm.com/support/knowledgecenter/linuxonibm/liabp/liabp_pdf.pdf
- Administering Platform Resource Scheduler

  http://publibfp.dhe.ibm.com/epubs/pdf/c2761170.pdf
- IBM Platform Resource Scheduler (PRS)
  - http://www.ibm.com/developerworks/servicemanagement/sds/prs/index.html
  - http://ibm.co/1D7JAMr
- IBM Software Defined Environments Technical Solution Brief

  http://ibm.co/1xkWOmj
- Software Defined Environments Solution Brief

  http://ibm.co/1F24w53

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services





Redbooks

IBM Software Defined Environment

(1.5” spine)
1.5”<-> 1.998”
789 <->1051 pages



SG24-8238-00

ISBN 0738440442

Printed in U.S.A.

Get connected



