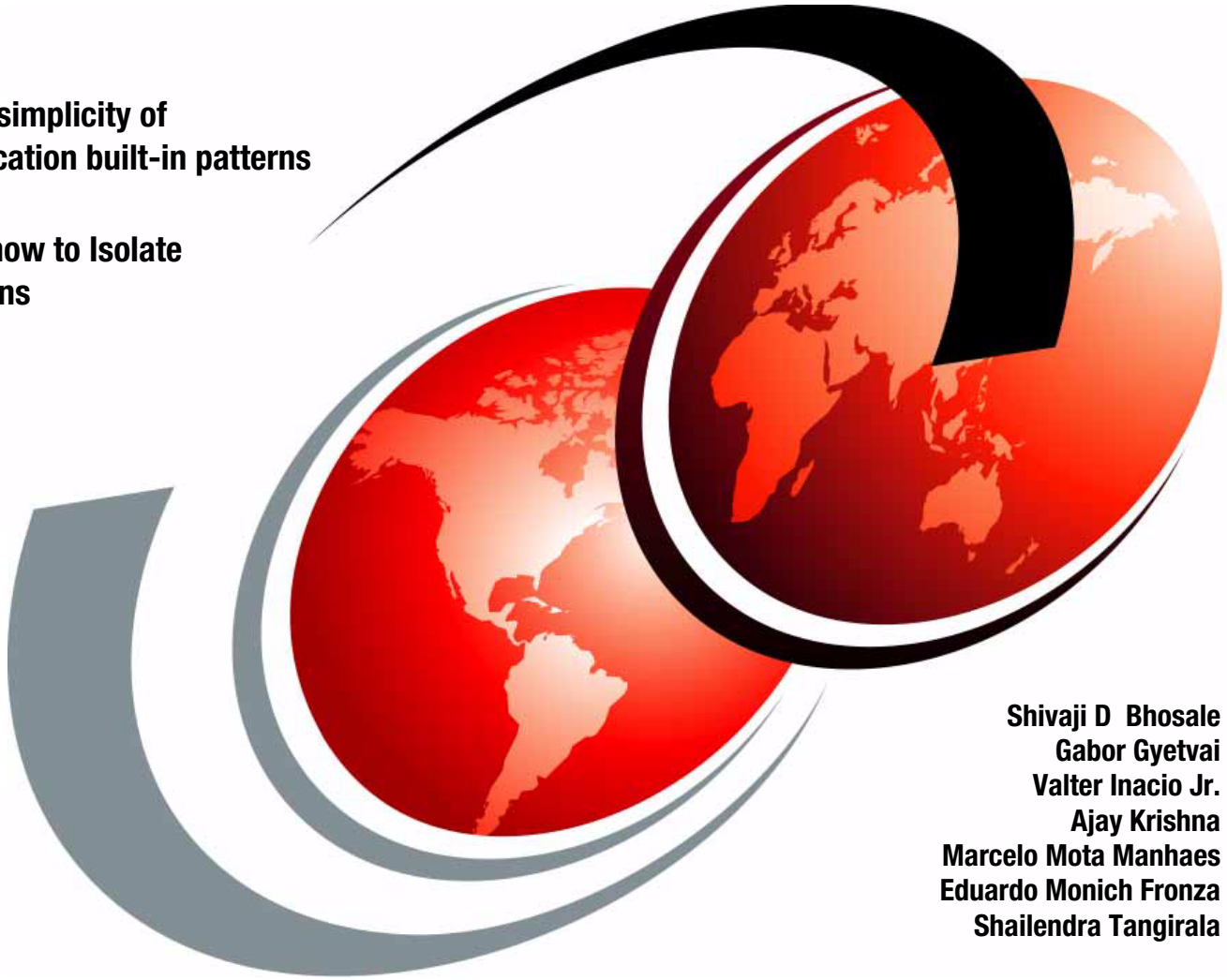
IBM

Adopting IBM PureApplication System V1.0

- Discover IBM PureApplication System agility
- Learn the simplicity of PureApplication built-in patterns
- Discover how to Isolate Applications



Shivaji D Bhosale
Gabor Gyetvai
Valter Inacio Jr.
Ajay Krishna
Marcelo Mota Manhaes
Eduardo Monich Fronza
Shailendra Tangirala

Redbooks

ibm.com/redbooks

International Technical Support Organization

Adopting IBM PureApplication System V1.0

May 2013

SG24-8113-00

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

First Edition (May 2013)

This edition applies to IBM PureApplication System V1.0.

© Copyright International Business Machines Corporation 2013. 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

Preface xi
The team who wrote this book xi
Now you can become a published author, too! xiv
Comments welcome. xiv
Stay connected to IBM Redbooks xiv

Chapter 1. Introduction. 1
1.1 The cloud environment 2
1.1.1 Characteristics 2
1.1.2 Cloud deployment models 3
1.1.3 Cloud service layers 4
1.1.4 Benefits of the cloud 6
1.1.5 Challenges and requirements 7
1.2 IBM PureSystems solutions 9
1.2.1 IBM PureFlex System 9
1.2.2 IBM PureApplication System. 10
1.2.3 IBM PureData System 11
1.2.4 Patterns of expertise 12
1.3 IBM PureApplication System overview 14
1.3.1 Configurations. 15
1.3.2 Virtual patterns 15
1.3.3 Management. 16
1.3.4 Problem determination 17
1.4 Scope of this book. 19
1.4.1 Intended audience 19
1.4.2 Assumptions 20

Chapter 2. Integrating IBM PureApplication System into an existing data center . . . 21
2.1 Prerequisites in the data center 22
2.1.1 Physical requirements. 22
2.1.2 Software infrastructure requirements 22
2.2 The Genesis process 22
2.3 Interfaces for using and administering IBM PureApplication System 23
2.3.1 Web GUI 23
2.3.2 Command-line interface 26
2.3.3 REST API 28
2.4 Security 29
2.4.1 User roles 29
2.4.2 Group level permissions 30
2.4.3 Customer engineer access 31
2.4.4 LDAP integration. 31
2.5 License management 32
2.5.1 Bundled software licenses 32
2.5.2 IBM License Metric Tool 32
2.5.3 License information. 33
2.6 Networking 34

© Copyright IBM Corp. 2013. All rights reserved.

2.6.1 Networking hardware in IBM PureApplication System 34
2.6.2 Main networks 36
2.6.3 Port setup 36
2.6.4 Link aggregation 39
2.6.5 VLANS 39
2.7 Auditing 40
2.7.1 Auditor roles 40
2.7.2 External storage server for audit logs 41
2.7.3 Handling audit data 42
2.8 Monitoring 44
2.8.1 Integrated solutions 44
2.8.2 PureApplication System Monitoring Portal 47
2.8.3 Monitoring layers 49
2.8.4 External monitoring 50
2.9 Integration with external backup systems 51
2.9.1 Tivoli Storage Manager System plug-in configuration 51
2.9.2 Configuring a scheduled Tivoli Storage Manager database backup 52
2.9.3 Creating a manual backup 54
2.10 Export and import features 56
2.11 External database integration 57
2.12 Integrating shared network storage to virtual machines 59
2.12.1 Network Access Server integration on virtual system level 59
2.12.2 NAS integration on virtual application level 60
2.12.3 iSCSI integration 60

Chapter 3. Usage patterns for isolating applications 61
3.1 Cloud computing overview 62
3.1.1 Cloud computing concepts 62
3.1.2 IBM PureApplication System features for cloud computing 63
3.2 Managing compute nodes 65
3.2.1 Starting compute nodes 66
3.2.2 Suspending compute nodes 66
3.2.3 Putting compute nodes in maintenance mode 66
3.2.4 Viewing compute node reports 66
3.3 Managing IP groups 68
3.3.1 Adding IP groups 69
3.3.2 Administering IP groups 71
3.3.3 Viewing IP group reports 72
3.3.4 Deleting IP groups and addresses 73
3.4 Managing cloud groups 75
3.4.1 Adding cloud groups 75
3.4.2 Administering cloud groups 77
3.4.3 Deleting cloud groups 78
3.4.4 Viewing cloud group reports 79
3.4.5 Managing storage volumes in cloud groups 81
3.4.6 Managing virtual machines in cloud groups 84
3.5 Managing shared services 87
3.5.1 Existing shared services 88
3.5.2 Deploying the System Monitoring service 89
3.5.3 Using an external caching service 90
3.5.4 Deploying an ELB proxy shared service 91
3.5.5 Deploying the Database Performance Monitor service 93
3.6 Managing environment profiles 94

3.6.1 Adding environment profiles 95
3.6.2 Editing an environment profile 96
3.6.3 Cloning an environment profile 98
3.7 IBM PureApplication System isolation strategies 99
3.7.1 Resource isolation via networking. 101
3.7.2 Resource isolation via access control. 101
3.7.3 Application environments 103
3.7.4 Multiple lines of businesses 104
3.7.5 Resource isolation by using a trust domain 105
3.7.6 Isolation for shared resources. 106

Chapter 4. PureApplication Deployment Models 107
4.1 Introduction 108
4.1.1 Virtual systems 111
4.1.2 Virtual applications 112
4.2 Trade-offs between control against total cost of ownership and total time to value . . 113
4.3 Virtual Appliances 114
4.4 Virtual Images 116
4.4.1 Hypervisor edition images. 116
4.4.2 Parts of hypervisor edition image 119
4.5 Virtual Systems 121
4.5.1 Virtual System Patterns 124
4.5.2 Planning and designing your virtual system pattern 129
4.5.3 Deploying virtual patterns 132
4.5.4 Customizing images and patterns. 137
4.5.5 Image Construction and Composition Tool and AMC tools usage 138
4.5.6 Logical to physical mapping for virtual systems 139
4.5.7 Benefits and trade-offs 140
4.6 Virtual applications 141
4.6.1 Concepts. 141
4.6.2 Virtual application patterns 143
4.6.3 Simple topology instantiation 170
4.6.4 Topology with scaling policy instantiation 171
4.6.5 Shared services 172
4.6.6 Benefits and trade-offs 183
4.7 Implementation strategy 184
4.7.1 Questions for consideration in application migration or creation. 185
4.7.2 Database best practices to deploy 188
4.8 PureSystems Centre 190

Chapter 5. Customizing Virtual System Patterns 199
5.1 Resources for customization 200
5.1.1 Customization targets 200
5.1.2 Tools for customization 201
5.2 Managing patterns by using Virtual System Pattern Editor 202
5.2.1 Editing virtual system patterns 202
5.2.2 Cloning virtual system patterns 209
5.2.3 Creating virtual system patterns 211
5.3 Build virtual images by using the IBM Image Construction and Composition Tool . . . 213
5.3.1 Installing Image Construction and Composition Tool 214
5.3.2 Enabling Image Construction and Composition Tool as a virtual application . . . 219
5.3.3 Working with Image Construction and Composition Tool 221
5.3.4 Creating software bundles with Image Construction and Composition Tool. . . . 238

5.3.5 Using command-line interface for Image Construction and Composition Tool . . 249
5.3.6 Advanced configuration with Image Construction and Composition Tool 249
5.4 Administering PureApplication System by using CLI 250
5.5 Working with script packages 255
5.6 Using Advanced Middleware Configuration 257
5.6.1 Overview 257
5.6.2 AMC User interfaces for PureApplications System 258
5.6.3 On-boarding applications with AMC 259
5.7 Add-ons and advanced options 264
5.7.1 Catalog add-ons to virtual system patterns 264
5.7.2 Advanced options 265

Chapter 6. Customizing Virtual Application Patterns 267
6.1 Prerequisites 268
6.1.1 Hardware requirements 268
6.1.2 Software requirements 268
6.2 Virtual Application Pattern elements 269
6.2.1 Components 269
6.2.2 Policy types customization 270
6.3 Creating virtual application patterns with the virtual application builder interface 277
6.4 Lifecycle management of virtual applications 279
6.4.1 Virtual pattern development kit 279
6.4.2 Plug-in overview 279
6.4.3 Virtual application lifecycle 280
6.5 Plug-in environment setup and creation of custom patterns 286
6.5.1 Troubleshooting and monitoring services 304
6.5.2 Deploying and managing virtual applications 306

Chapter 7. Integrating PureData for Transaction 309
7.1 Advantages of integrating PureData for Transaction with PureApplication System . . 310
7.2 Prerequisites for integrating the PureData for Transaction with PureApplication 311
7.3 Registration of PureData for Transaction with PureApplication System 311
7.4 Deploying pureScale Database from PureData for Transaction 313
7.5 Deploying PureApplication System middleware 318
7.5.1 Creation of a virtual application pattern 318
7.5.2 Deploying the virtual application 323

Chapter 8. Troubleshooting PureApplication System Environment 329
8.1 Troubleshooting overview 330
8.2 Consoles available for PureApplication System troubleshooting 332
8.2.1 System Console 332
8.2.2 Workload Console 341
8.3 Command-line interface 348
8.4 Product logs and tracing 351
8.4.1 Enable product tracing 351
8.4.2 Product logs 352
8.5 Image Construction and Composition Tool 359
8.5.1 Troubleshooting the Image Construction and Composition Tool 359
8.5.2 Image Construction and Composition Tool log files 365
8.6 Virtual image synchronization problems 371
8.6.1 ESX VMWare Server and Microsoft Windows related troubleshooting 371
8.6.2 Common problem troubleshooting 372

Chapter 9. High Availability and Disaster Recovery 377

9.1 High Availability and Disaster Recovery overview 378
9.1.1 Failover strategies 379
9.1.2 Software clustering modes 379
9.1.3 HADR solution components 380
9.1.4 Disaster recovery models 382
9.2 Workload failover and recovery scenarios 383
9.2.1 Virtual applications: WebSphere Application Server 383
9.2.2 Virtual systems: WebSphere Application Server without IMP 383
9.2.3 Virtual systems: WebSphere Application Server with IMP 384
9.2.4 Virtual applications: DBaaS 384
9.2.5 Virtual systems: DB2 385
9.2.6 Caching shared service 385
9.2.7 Proxy shared service 385
9.2.8 Custom image 386
9.3 Hardware high availability 386
9.4 High availability scenarios 387
9.4.1 Intra-rack high availability 387
9.4.2 Inter-rack high availability 388
9.4.3 Inter-rack high availability within the same data center 390
9.4.4 Inter-rack high availability across distributed data centers 392
9.5 Disaster recovery approaches 394
9.5.1 Virtual application pattern backup and restore 394
9.5.2 Virtual system pattern back up and restore 398
9.5.3 Virtual appliance pattern backup and restore 400
9.5.4 Configurations backup 402
9.6 Current limitations 402

Related publications 403
Online resources 403
Help from IBM 405

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. 2013. 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:

AIX®
CICS®
CloudBurst®
DataPower®
DB2®
Domino®
Global Technology Services®
IBM PureData™
IBM SmartCloud®
IBM®
IMS™
Informix®
Lotus®
PowerVM®
pSeries®
PureApplication™
PureData™
PureFlex™
pureScale®
PureSystems™
Rational®
Redbooks®
Redbooks (logo) ®
Storwize®
System x®
Tivoli®
Velocity™
WebSphere®

The following terms are trademarks of other companies:

Netezza, and N logo are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

Intel, 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.

ITIL is a registered trademark, and a registered community trademark of The Minister for the Cabinet Office, and is registered in the U.S. Patent and Trademark Office.

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.

Preface

This IBM® Redbooks® publication introduces users to the concepts of the IBM PureApplication™ System V1.0. This book covers the most common problems, solutions, best practices, and use cases about adopting the IBM PureApplication System V1.0.

The target audience for this book is anyone from the IT industry who wants to acquire a better understanding of IBM PureApplication System, including technical consultants, business partners, and independent software vendors who are considering migrating to a cloud computing solution. This book also is applicable to system administrators, middleware specialists, and software engineers who need a more in-depth approach to PureApplication System features and capabilities.

The team who wrote this book

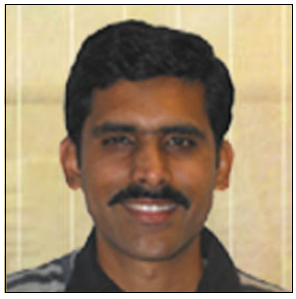
This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Shivaji D Bhosale is a Team Lead at Application and Infrastructure systems management (ISM), Systems and Technology group, Pune, India. He has 16 years of experience in the IT sector. He has a master's degree in Computer Management from Shivaji University Kolhapur. His areas of expertise include IBM Cloud implementation on VMware and PowerVM®, UNIX, pSeries®, middleware, and related technologies. He is an IBM Certified Specialist in Cloud Computing Infrastructure V1 Architect, pSeries AIX® System Administration, Tivoli® Monitoring, and EXIN ITIL. Shivaji is a co-author of IBM Redbooks publications *WebSphere® Cloudburst Appliance and PowerVM* , SG247806. His areas of interest include cloud technology, pSeries virtualization, middleware, and systems management automation.

Gabor Gyetvai is a Technical consultant in Hungary. He has 15 years of experience in different IT areas. He worked on the PureSystem enablements project since the announcement of PureSystem. He holds a degree in IT Technical Engineering from Hungary. His areas of expertise include Lotus® Domino® development, storage, System x, and Mobile application platform.

© Copyright IBM Corp. 2013. All rights reserved.

Valter Inacio Jr. is an IT specialist from Brazil with 12 years of experience in software development, infrastructure support, and systems administration. He works at CPqD as an architect for fraud prevention solutions that are designed for the financial market, which involve Business Rules and Analytics, Business Intelligence, Java Enterprise Edition, and service-oriented architectures (SOAs). CPqD is an independent institution that is focused on innovation in Information and Communication Technology (ICT). Valter holds a Bachelor's degree in Informatics and a Master's in Computing Science from University of São Paulo. He also is IBM Certified for several technologies, including WebSphere Application Server, Rational® Application Developer, and Object Oriented Analysis and Design.



Ajay Krishna is a Senior Staff Software Engineer at the IBM Software Group in India. He has nine years of experience at IBM working with the middleware platform. He worked on WebSphere Application Server as a testing, level3 support, and development engineer. His primary areas of expertise include Java programming, J2EE, and related technologies. He holds a Bachelor's degree in Engineering from the Visweswaraiah Technological University in Karnataka, India. Ajay also is an IBM Certified System Administrator for WebSphere Application Server and SOA solutions. He also holds professional certifications on Java technology and DB2®.

Marcelo Mota Manhaes is a Level 2 Certified IT Specialist working for IBM Global Technology Services® in Brazil. He has eight years of expertise in supporting IBM WebSphere Application Server and IBM WebSphere Portal Server. His main areas of expertise are automation for WebSphere administration, infrastructure design, implementation, maintenance, and problem determination of the WebSphere environment. Eduardo also is an IBM Certified System Administrator for WebSphere Application Server, WebSphere Portal Server, and SOA Solutions. He holds a Bachelor's degree in Computer Science from Universidade do Estado de Santa Catarina.

Eduardo Monich Fronza is a Level 2 Certified IT Specialist working for IBM Global Technology Services® in Brazil. He has eight years of expertise in supporting IBM WebSphere Application Server and IBM WebSphere Portal Server. His main areas of expertise are automation for WebSphere administration, infrastructure design, implementation, maintenance, and problem determination of the WebSphere environment. Eduardo also is an IBM Certified System Administrator for WebSphere Application Server, WebSphere Portal Server, and SOA Solutions. He holds a Bachelor's degree in Computer Science from Universidade do Estado de Santa Catarina.

Shailendra Tangirala is an IBM Certified Senior Architect, focused on the Oil and Gas industry in Houston, Texas. He has over 15 years of experience developing Enterprise IT solutions for various industries. Previously, Shailendra worked in the IBM WebSphere Application Server Professional services group where he leveraged WebSphere Portal Server and IBM Information Management products for creating and implementing Enterprise SOA solutions for clients. His interests include Business Analytics and Cloud solutions that are based on IBM portfolio of products, such as PureSystems™, Big Insights, and IBM SmartCloud®. Shailendra has a Master of Science degree in Computer Engineering.

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

- Margaret Ticknor
- Deana Coble
- Shari Deaina
- Tamikia Lee
- Linda Robinson

International Technical Support Organization, Raleigh Center

- Guiseppe Accardo
- Srihari Angaluri
- Chris A Bailey
- Joe Bohn
- Ching-Yun Chao
- Jarek Miszczyk
- Steve Tremper
- Bobby Woolf

IBM US

- Jeremy Brumer
- Eric Koeck
- Lewis Lo
- Kimberly Mungal
- Brian Olynyk

IBM Canada

- Deepak Khandelwal

IBM India

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, while 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 at this website:

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 found at this website:

  http://www.ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  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

This chapter gives an overview of the cloud computing concepts, explains the differences between IBM PureSystems offerings, and describes the fundamentals of IBM PureApplication System.

The following topics are covered in this chapter:

- The cloud environment
- IBM PureSystems solutions
- IBM PureApplication System overview
- Scope of this book

© Copyright IBM Corp. 2013. All rights reserved.

1.1 The cloud environment

As technology evolves, new concepts and paradigms emerge from the industry and achieve such an important position and public acceptance that they quickly become a de facto standard. With cloud computing, this fact is no different. Cloud computing is revolutionizing the way organizations provide, manage, and use IT services, and in a manner that is affecting all the connections in the information chain, from the infrastructure providers and independent software vendors (ISVs) to the users.

The term *cloud computing* has the following definitions, which can vary depending on the perspective where it is described:

- From the infrastructure provider perspective, cloud computing is a flexible way to deliver high-end computing resources that can be shared by multiple software vendors and scaled on demand as the workload grows.
- From the independent software vendor view, the cloud is a strategic platform that allows cost-effective business operation and rapid deployment of services, which are achieved through the optimized usage of IT resources and the immediate availability of the infrastructure.
- From the users position, the cloud is a ubiquitous environment that provides pay-per-use applications over the network that can be seamlessly integrated with multiple devices, regardless of their location.

In the next sections, we explore the characteristics, benefits, and challenges of this new cloud computing environment.

1.1.1 Characteristics

A cloud computing environment generally exhibits the following characteristics:

- Elasticity
- Resiliency
- Multitenancy
- Virtualization
- Workload migration

However, the following characteristics are essential for a solution to be referred to as a cloud:

- Resource provisioning

  A real cloud environment must have an abstraction layer between the physical hardware (for example, processing, memory, storage, and network) and the computing resources that are provisioned. The virtualization infrastructure must dynamically manage and allocate these resources and support changes in the hardware capacity without compromising other resources.

- Scalable environment

  Cloud environments are supposed to be elastic; that is, increase or decrease the computing capacity rapidly in response to the workload. The quality of service (QoS) must remain the same under any conditions. Computing capacity changes should be performed in a manual or automated fashion, without any disruption in the service.

- On-demand management

  In a cloud environment, the customer manages the computing resources and the software capabilities on-demand, without any human intervention from the service provider. It is expected that the customers can have a single, fully integrated interface where they can manage most of the aspects of the service, but with a higher level of abstraction (for example, response time, bandwidth, and CPU usage).

- Service measurement

  It is necessary for the cloud environment to provide different types of metering so that the resources consumption can be monitored, controlled, and charged back to the customer based on the service usage. This paradigm is important when there is a service level agreement (SLA) with which the customer can evaluate if the QoS is being achieved as per the contract.

- Pervasive access

  Applications and services that are made available in the cloud must be pervasive; users should be able to access them from any place over the network and through standard protocols that are supported by different devices (for example, desktops, tablets, and smartphones).

1.1.2 Cloud deployment models

The cloud computing environments usually are classified according to the levels of access, control over the infrastructure, and sharing of the data center by the customers of the service (businesses or individuals). Also known as *deployment models*, this classification is divided into the following types:

- Public cloud

  In a public cloud, the infrastructure is made available for the general public and is offered by organizations in a free or pay-per-use model. The computing resources are hosted by the service provider and shared between different customers (multi-tenancy), which have access to the cloud environment through an ordinary Internet connection.

- Private cloud

  In a private cloud, the infrastructure is provisioned for an exclusive organization or department, which introduces higher levels of security and privacy into the cloud environment. The computing resources can be hosted internally or externally and managed by the organization or a third party. Usually, the access to a private cloud is restricted, with servers that are standing behind multiple firewalls and reachable only through a virtual private network (VPN).

- Community cloud

  In a community cloud, the infrastructure is shared by a specific group of organizations with a common concern (for example, policy, security requirements, and compliance regulations). The computing resources can be owned, managed, and operated by one or more organizations or by a third party and the costs are spread among the members of the community.

- Hybrid cloud

  A hybrid cloud is a combination of two or more cloud infrastructures (public, private or community), which offers the benefits of multiple deployment models. The computing resources are bound together by proprietary or standardized technologies, which improves the cloud fault-tolerance capabilities and enables a more flexible environment by supporting different levels of requirements.

1.1.3 Cloud service layers

Cloud computing providers can be classified according to the service layers or delivery models that the resources are offered, from the basic infrastructure services up through the platform and application services. Although there are variations on the definitions and ranges of those layers, the IT community agreed on three service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

Infrastructure as a Service

The IaaS model takes place when a cloud service provider delivers infrastructure assets, such as servers, processing, storage, network, and other fundamental physical resources to the customer. Usually, those resources are exposed through a virtualization layer, which consists of the *hypervisor* and its associated *virtual machines* (VMs), as shown in Figure 1-1.

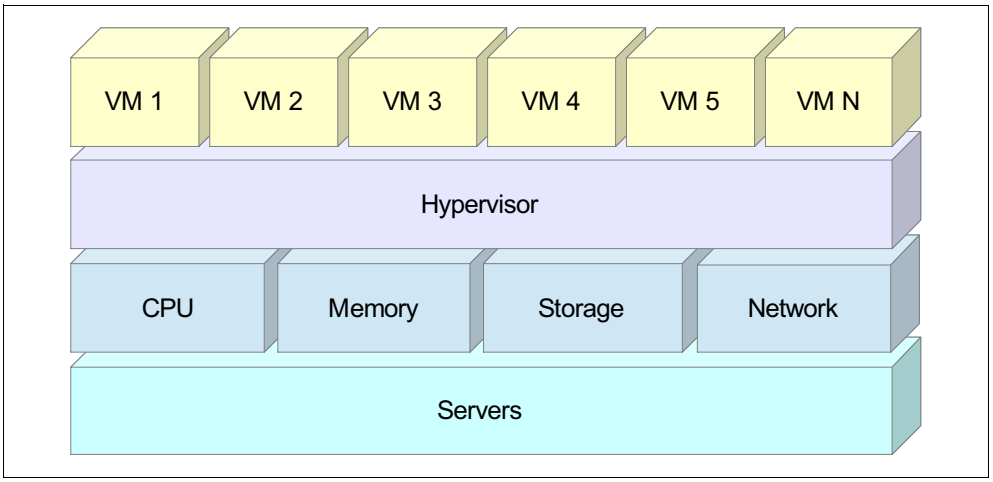


Figure 1-1 The IaaS model and its components

The customer is responsible for installing and managing the software, which might include operating systems, middlewares, and applications. Although this approach provides the organizations a higher degree of freedom and control over the environment, the service provider is not required to offer any cloud-specific feature, such as scaling capabilities and on-demand management.

Platform as a Service

In the Platform as a Service (PaaS) model the provider delivers a complete development platform, including operating systems, runtimes, and middlewares. The objective is to supply the necessary components for rapidly building and deploying applications in the cloud, as shown in Figure 1-2.

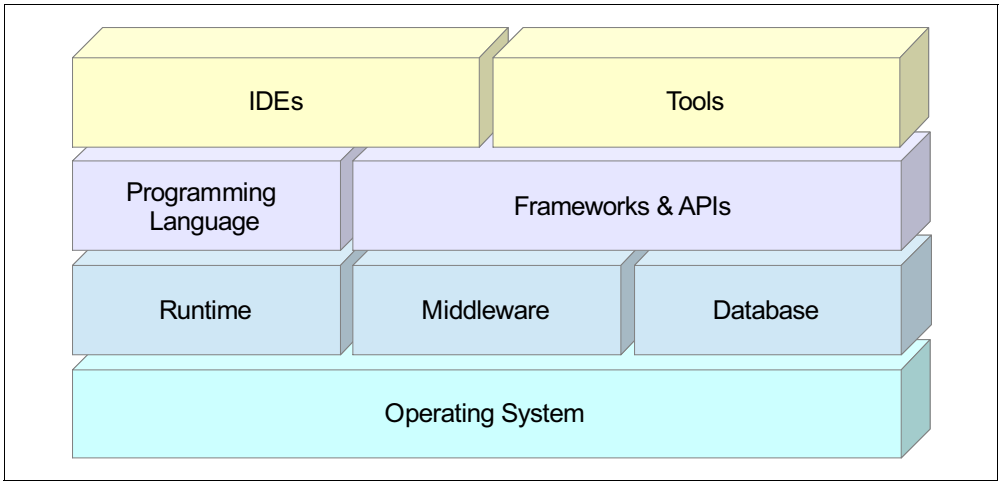


Figure 1-2 The PaaS model and its components

The PaaS model relieves the organization of the complexities of managing individual hardware and software elements, which can be time-consuming and require highly skilled professionals to support its operation. Some providers also require that applications be written by using specific programming languages, frameworks, and application programming interfaces (APIs) that are supported by their environment. In the PaaS model, the customer does not manage the underlying infrastructure directly, but has some control over it through the operating system or some administration console that is provided with the platform.

Software as a Service

In the Software as a Service (SaaS) model, the provider delivers a fully functional application, service, or business process that runs on a cloud environment. The customer can adopt the software with little or no effort, depending on the degree of integration and adaptation that is necessary. Also, customers can collaborate with each other and share and create custom applications by composing multiple services. The basic structure of a SaaS model is shown in Figure 1-3 on page 6.

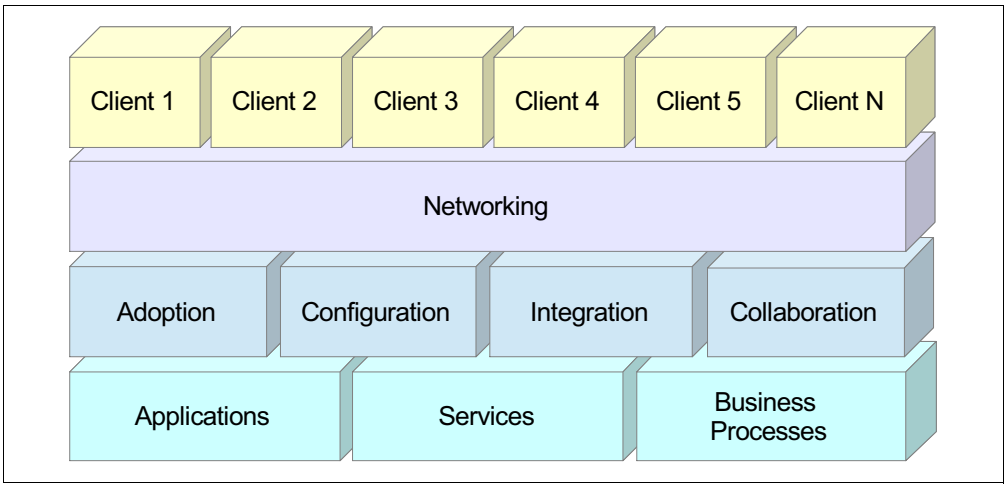


Figure 1-3 The SaaS model and its components

In a SaaS model, the software is managed in a centralized manner by the provider, which is responsible for fixing, updating, and monitoring the components of the system. The clients can configure some basic capabilities of the service, but have no control of the installed software or the underlying cloud infrastructure. The applications are accessible through a network connection from different clients, which might vary from a user's web browser that accesses an email application to an Enterprise Service Bus (ESB) that calls a web service that is provided by a business partner.

1.1.4 Benefits of the cloud

The benefits of cloud computing are spread over all service layers. Some of these benefits are described in the next sections.

Efficient usage of the infrastructure

The cloud computing model promotes efficient usage of the infrastructure by using virtualization capabilities and multi-tenancy features. It allows multiple operating systems, middlewares, and applications to run in a single physical server. This configuration assures that the computing resources are used in an optimized and effective manner. This model also permits each tenant to run in an isolated environment, with no effect on the resources that are allocated to other tenants, and by supporting a reasonable level of security.

Dynamic and scheduled provisioning of resources

The abstraction layer that is created by virtualization allows computing resources to be dynamically allocated to the tenants. Companies can request resources manually or by establishing rules that are based on the workload so that the resources are available where and when necessary. Typically, the companies develop a workload management process that keeps priorities in check. They can also create scheduling policies; for example, to accommodate peak loads that are expected for a determined period. This ability ensures that the pool of resources is managed in the most effective manner, without waste or lack of resources.

Resiliency and workload migration

The ability to completely isolate the failure of a computing resource without affecting users is mostly related to the capability of the cloud to transparently migrate its workloads to another physical resource and continue their processing. If a compute node or storage resource fails, the cloud resiliency feature enables it to move the workload without human intervention or disruption in the service. Migration also can be started based on certain policies, such as compliance regulations (sensitive data that must be processed in a specific geographic region or data center) and performance considerations, such as user data must be processed near the user location for faster access.

Optimization of operational and capital expenses

There are several financial benefits concerning the adoption of a cloud computing solution. These benefits generally can be placed into the following categories:

- Optimization of operational expenses (OpEx) often is associated with the maintenance and operation of the IT infrastructure, including physical assets and human resources. Cloud environments enable businesses to use these assets more efficiently and at a rate that is proportional to the value that is generated from the supported application workloads. Also, the automated and dynamic nature of the cloud allows the allocation of human resources only when necessary, which leaves the employees to focus on more productive activities.
- Optimization of capital expenses (CapEx) is related to the investments and costs that are incurred for buying the IT infrastructure, which includes hardware and software assets. As the infrastructure resources can be obtained according to their needs, organizations can better manage their budget by diluting the costs over time instead of spending the money all at once. More cost savings can be derived from a higher usage of the existing hardware assets without the need for complex forecasting models, which can be inaccurate. Software licensing also can be optimized because the number of required licenses per-installation can be reduced by vertically scaling the infrastructure.

1.1.5 Challenges and requirements

While the cloud environment provides benefits, some challenges can still exist that require solutions from the cloud service providers, as described in the next sections.

Automation

In a PaaS cloud environment, it is important not only to improve the process of developing and delivering an application, but the lifecycle of that application. Some common tasks, such as resource provisioning, middleware configuration, and application deployment must be fully automated to gain efficiency, reduce the number of errors, and ensure consistency in the process. This automation makes the dynamic characteristics of a cloud possible and simplifies the application lifecycle management.

Standardization

A common concern that is shared by any new client that is adopting a cloud computing model is the lack of standards. Because cloud computing is in its infancy and most of the standardization attempts are a work in progress, it is difficult for the clients to define what solution best fits their mid to long-term needs. Data conversion and portability, application interoperability, and integration are important aspects of the technology that allow clients to deal with the heterogeneity of the cloud environments. In addition, standardization enables flexibility and vendor-neutral applications, which gives the clients freedom of choice to move from one service provider to another. Service providers must work together and collaborate with standards organizations to make sure that the cloud technology is based on open standards and delivers the necessary agility that the IT industry demands.

Security

Besides the usual challenges of developing secure IT systems, cloud computing presents another level of risk because essential services usually are outsourced to a third party and shared by multiple tenants. The external aspect of outsourcing makes it harder to maintain data integrity and privacy, support IT governance, and ensure compliance. Even basic tasks, such as applying patches and configuring firewalls, can become the responsibility of the cloud service provider. As a result, clients must establish trust relationships with their providers and define strict SLAs in terms of how these providers implement, deploy, and manage security on their behalf and what compliance levels are required.

Transaction processing

Transaction processing traditionally is one of the areas with greater demand for high-performance systems. Normally restricted to financial institutions that run on dedicated mainframe platforms, these applications are CPU and storage intensive and they are required to process thousands of transactions per second. With the increase of B2B and B2C systems, those workloads must be managed across multiple computing environments and business processes, but with the same required response-times.

Big data analytics

Managing big data sets and making them available for deep analysis are complex computing and data management challenges. Most of the existing data warehouse systems are built on old database technologies, which were not designed to scale and perform consistently for huge data volumes. To run analytic processing at top speed, general-purpose database management systems must be tuned and configured, and the queries that are manually optimized by database administrators. It is necessary that new analytic infrastructures provide an abstraction layer that hides the increasing complexity of data processing from users, which enables greater efficiency while simplicity is maintained.

1.2 IBM PureSystems solutions

To meet current IT and business demands, IBM is introducing a new category of systems. These systems combine the flexibility of general purpose systems, the elasticity of a cloud computing environment, and the simplicity of an appliance that is tuned to the workload. IBM PureSystems solutions are the building blocks of those capabilities. This new category of systems represents the collective knowledge of thousands of deployments, established guidelines, best practices, innovative thinking, and distilled expertise.

IBM PureSystems includes integration across compute, storage, networking, and platform middleware for physical and virtual resources. It also includes built-in expertise from IBM factory pre-configuration, integration, and testing. This simplified experience enables clients to deliver new cloud services and applications faster and with improved economics over traditional IT.

IBM PureSystems offerings are designed to deliver value in the following ways:

- Built-in expertise: Systems must capture and automate best practices and expertise, which reduces manual steps that affect a project's time to value with an open architecture and allows participating solution providers to optimize their applications workloads.
- Integration by design: All hardware and software components must be integrated by design, tuned in the lab, and pre-packaged in the factory into a single, ready-to-go system that is optimized for the business task.
- Simplified experience: IT staff and the lines of business that use IT realize a simplified systems lifecycle. Collections of hardware, middleware, and application components no longer need to be separately procured, configured, tuned, and managed. IBM PureSystems are ordered, unpacked, plugged in, and managed as a single system from a centralized interface.

The IBM PureSystems offerings are available in the following types:

- IBM PureFlex™ System combines servers, storage, networking, and virtualization capabilities into a single, unified, and upgradeable infrastructure system.
- IBM PureApplication System provides a flexible platform that is designed to ease the process of deploying, customizing, and managing middlewares and applications.
- IBM PureData™ System integrates specialized hardware and software components for transactional, operational, and deep analytical workloads.

1.2.1 IBM PureFlex System

The IBM PureFlex System is an infrastructure system that provides a fully integrated computing platform. It combines servers, storage, networking, virtualization, and management into a single structure. Its built-in expertise enables organizations to manage and deploy integrated patterns of virtual and hardware resources through a unified management console. PureFlex System is ideally suited for customers who want a system that delivers the simplicity of an integrated solution while still able to tune middleware and the runtime environment.

IBM PureFlex System implements the IaaS layer and can provide higher levels of control, efficiency, and operating agility. By using built-in virtualization and workload placement that is based on resource availability, the infrastructure system enables automated scaling of resources and true workload mobility. This combination enables businesses to rapidly deploy IT services at a reduced cost. Moreover, it enables deep integration and central management through a comprehensive, open-choice infrastructure system. It dramatically reduces the skills and training that are required for managing and deploying the system.

The IBM PureFlex System is available in the following configurations:

- IBM PureFlex System Express is designed for small and medium-sized businesses and is the most affordable entry point for PureFlex System.
- IBM PureFlex System Standard is built for application servers with supporting storage and networking, and is designed to support key ISV solutions.
- IBM PureFlex System Enterprise is optimized for transactional and database systems. It has built-in redundancy for highly reliable and resilient operation to support the most critical workloads.

The PureFlex System is built from no-compromise building blocks that are based on IBM technologies that support open standards and offer roadmaps for integration with multiple systems. It is designed to give control and choice without increasing complexity. The customers are not limited to a single architecture or vendor's middleware; they can choose from a broad range of servers, operating systems and hypervisors, and design systems that are truly optimized for their needs. Customers also can consolidate different workloads from many resource-consuming servers onto a single PureFlex System and control it all from a unified interface.

1.2.2 IBM PureApplication System

The IBM PureApplication System is a platform that is designed and tuned specifically for transactional web and enterprise applications. It provides a full stack of software, including operating systems, middlewares, databases, and patterns of expertise that enable customers to achieve greater agility and to deliver new business capabilities. IBM PureApplication System uses IBM PureFlex System infrastructure as a foundation to support high workloads, simplify infrastructure management, and improve the application's lifecycle. This workload-aware, flexible platform is easy to deploy, customize, manage, and operate in a traditional or private cloud environment.

IBM PureApplication System implements the PaaS layer to provide superior IT economics. With the PureApplication System, customers can create their own patterns of software, middleware, and virtual resources. Customers can provision and share these patterns within a unique framework that is shaped by IT guidelines, best practices, and industry standards. These standards were gathered from many years of IBM experience with thousands of clients around the world and are infused throughout the system.

IBM PureApplication System provides the following advantages:

- Agility: By automating key processes such as operating system installation, middleware configuration and application deployment, PureApplication System can reduce the cost and time that is required to manage the platform.
- Efficiency: With PureApplication built-in expertise, customers can optimize critical business processes and conserve valuable resources, getting the most from the systems in terms of energy efficiency, maintenance, and fast response to problems.

- Simplicity: PureApplication built-in patterns of expertise can help customers to easily consolidate different servers, storage, and applications into an easy-to-manage, integrated system that can be controlled from a single management console.
- Scalability: By defining a high-level set of parameters and policies, administrators can take advantage of PureApplication System features to scale the application up and down automatically according to the workload.
- Reliability: Built-in application expertise reduces the number of unplanned outages through guidelines and the automation of manual processes that are identified as causes of outages. PureApplication System also can use data replication to increase high availability.

1.2.3 IBM PureData System

The IBM PureData System is a platform that is optimized exclusively for delivering data services to today's demanding applications. It provides data management expertise for each type of workload, such as automated pattern-based deployment and management of highly reliable and scalable transactional database services. It contains hardware and software capabilities that are designed and optimized for specific high-performance data workloads, such as data filtering by using programmable hardware (FPGAs) for ultrafast execution of analytic queries. The IBM PureData System provides a fully integrated management console for the entire system, with integrated system upgrades and maintenance.

Different applications rely on different types of data processing, as shown in the following examples:

- E-commerce: Relies on scalable relational databases for transaction processing.
- Customer analysis: Requires analytics data warehouses for reporting and analytics.
- Fraud detection: Needs operational data warehouses for real-time decision making.

Traditionally, IT staff are responsible for configuring and tuning general purpose systems according to each of those needs. This general purpose approach presents the following disadvantages:

- Configurations can be complex and rely on specialized skills, which may not be available.
- Configuration and tuning is error prone because it is a manual process.
- It is a time-consuming task that can affect the application time-to-market.

In response to these disadvantages, IBM adopted a smarter approach by designing, optimizing, and tuning IBM PureData System for each type of workload to deliver specialized data services for applications with simplicity, speed, and lower cost. As such, the following advantages are realized:

- Reduced complexity: Each system is pre-tuned for the different needs of different analytic and transaction processing.
- Accelerated time-to-value: Data management expertise is built in each system and ready for immediate use.
- Improved IT economics: The entire system lifecycle is simplified, from acquisition to maintenance and upgrade.

The IBM PureData System is available in the following configurations:

- IBM PureData System for Transactions: Contains factory-integrated and optimized server, storage, network, and software resources that are selected specifically for online transactional processing (OLTP) workloads. They are designed, integrated, and tuned to support transactional processing applications that require high performance, high scalability, and high throughput with fast response time.

- IBM PureData System for Analytics: Designed specifically for running complex analytics on large data volumes. This system, which is powered by Netezza® technology, is a high-performance, hardware-accelerated, massively parallel system that enables clients to perform analytics on big data. It provides an easy-to-use data warehouse appliance that runs business intelligence (BI) algorithms extremely fast and requires minimal administration or tuning.
- IBM PureData System for Operational Analytics: Optimized for the right balance of analytics performance and operational query throughput, this PureData System delivers rapid insight into high volumes of fast-moving data. It was designed for applications that must analyze multiple business variables to perform real-time decision making. By using multidimensional cubing services over a relational data warehouse schema, it optimizes performance for online analytical processing (OLAP) queries, providing more power for users to analyze data.

1.2.4 Patterns of expertise

IBM PureSystems are built to include *patterns of expertise*. Patterns of expertise accelerate the time-to-value of applications, services, and business processes by encapsulating guidelines and best practices into a repeatable and deployable form. They can automatically configure, manage, and optimize the elements of a solution, from the infrastructure resources up through the middleware and applications.

Patterns of expertise represent the knowledge and experience that is gained over decades by IBM while optimizing the deployment and management of thousands of data centers, software infrastructures, and applications around the world. They are categorized in three types, one for each service layer: infrastructure patterns, platform patterns, and application patterns. Figure 1-4 shows the relationship between the patterns and IBM PureSystems products.

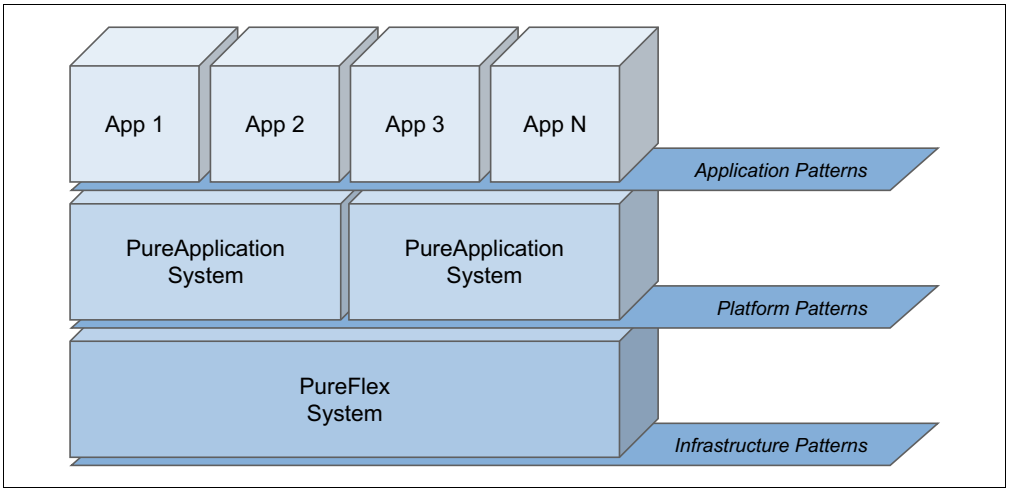


Figure 1-4 Patterns of expertise and IBM PureSystems products

Infrastructure patterns

Infrastructure patterns provide an integrated, automated, policy-driven infrastructure management across processing, storage, and networking resources. These patterns enable a faster hardware configuration and simplified management of low-level resources, which in turn reduces operational expenses and increases performance by using optimal settings.

Platform patterns

Platform patterns encapsulate pre-configured, policy managed platform services, such as caching elasticity, failover, load balancing, security, database, and middleware. These patterns enable faster application development via the automation of common platform management tasks, such as environment setup, middleware configuration, and application deployment.

Application patterns

Application patterns comprise predefined application architectures and required platform services that are deployed and managed by the system according to a set of policies. Application patterns encapsulate proven, well-designed solutions and best practices for recurring problems in enterprise architectures. They provide key software engineering solutions for building applications that are robust, scalable, and easy to maintain.

1.3 IBM PureApplication System overview

The IBM PureApplication System is a workload optimized and integrated hardware and software solution that is designed to simplify the development, provisioning, and management of applications in a private cloud environment. It features integrated management capabilities, which allows self-service provisioning of elastic applications, databases, and middlewares. In PureApplication System, the hardware and software are deeply integrated, which provides a high degree of automation, performance, and simplicity to the data centers.

PureApplication System is integrated by design. IBM took the collective knowledge of decades of experience in integrating and tuning servers, storage, networking, virtualization, and management and used it to design a new system. With PureApplication System, the middleware, development, and deployment expertise are integrated and optimized from the factory, as shown in Figure 1-5. PureApplication System inherits the infrastructure capabilities of PureFlex System and builds on them a complete platform, which is designed for enterprise applications that require high performance, scalability, and optimal usage of resources.

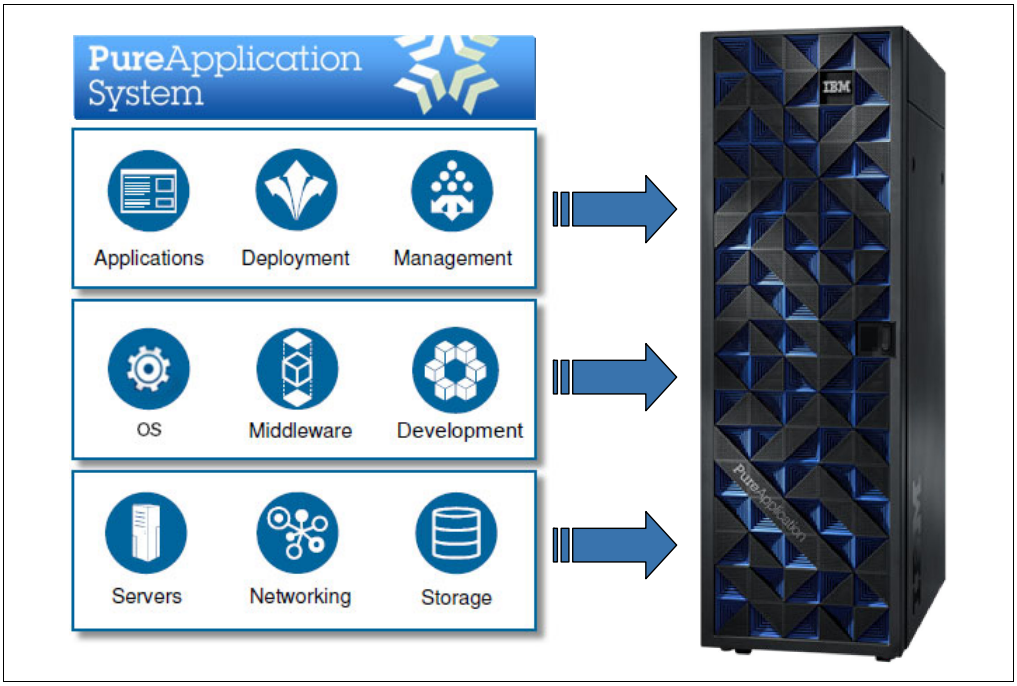


Figure 1-5 PureApplication System features are integrated by design

1.3.1 Configurations

There are four available configurations of PureApplication System, as shown in Table 1-1. All four configurations include the same storage and networking components and same middleware patterns and management capabilities. Each configuration includes 48 terabytes of hard disk storage and 6.4 terabytes of solid-state storage. The configurations vary by the number of included compute nodes. Each compute node consists of 16 physical cores and 256 gigabytes of memory. The smallest configuration includes 6 compute nodes and the largest includes 38. Upgrades to a larger system can be ordered and installed without any downtime for existing workloads.

Table 1-1 IBM PureApplication System configurations

	IBM PureApplication System W1500-96	**IBM PureApplication System W1500-192**	**IBM PureApplication System W1500-384**	**IBM PureApplication System W1500-608**
Compute nodes	6	12	24	38
CPU cores	96	192	384	608
RAM memory	1.5 TB	3.1 TB	6.1 TB	9.7 TB
SSD storage	6.4 TB			
HDD storage	48 TB			

The IBM PureApplication System comes with a set of preinstalled software that can be used in the system, including the operating system (Red Hat Enterprise Linux), middleware (WebSphere Application Server), and database (DB2 Enterprise). Other software products also are built and optimized for use in PureApplication System but are not included with the system. These products can be purchased and loaded onto PureApplication System through the PureSystems Centre, a repository of products that are built for the PureSystems family. PureApplication System includes license management capabilities so that the customers can track license usage and availability for non-entitled software products.

The system also includes built-in expertise that is provided through virtual system patterns and virtual application patterns, which enable applications to easily integrate and optimize the use of the underlying platform. It also comes with the Advanced Middleware Configuration tool for application on-boarding and the Image Construction and Composition Tool for building custom virtual images for use with PureApplication System.

1.3.2 Virtual patterns

IBM uniquely builds expertise into PureApplication System with patterns of expertise. Patterns of expertise are proven best practices and expertise for complex tasks that are learned from decades of client and partner engagements that are captured, tested, optimized, and then built into the system. They are not just a blueprint or set of instructions; the expertise is built in and inherently usable.

Patterns deliver faster time-to-value by removing manual steps and automating delivery. They improve efficiency and simplicity by reducing costs, resources, and the amount of in-house expertise that is required for deploying solutions. Patterns also increase control by allowing repeatable, automated, and optimized deployments, which reduce the risk of human error.

PureApplication System contains three types of patterns: virtual system patterns, virtual application patterns, and virtual appliances. These patterns are described in the following sections. For more information, see Chapter 4, “PureApplication Deployment Models” on page 107.

Virtual System Patterns

Virtual system patterns provide an automated model for deploying middleware topology patterns. They allow customers to quickly deploy traditional workloads in a virtualized environment in a repeatable fashion. Products that are deployed by using the virtual system patterns are managed by using the existing management tools that are provided by those products.

Virtual system patterns contain a collection of middleware parts that can be connected to build a topology for a particular type of deployment. The middleware parts that are used to build these patterns are known as *virtual images*. IBM provides many virtual images that contain IBM middleware products that are designed to run in virtual machine environments. It also is possible to include custom virtual images.

Virtual Applications Patterns

Virtual application patterns provide a highly automated, policy-based deployment model in which the customer defines application components and policies that specify the needs of the application. The virtual application pattern is application-centric, whereas the virtual system pattern is middleware topology-centric. It has a highly simplified administrative model, which exposes fewer administrative functions than the virtual system pattern.

IBM PureApplication System support the most common transactional web application types and includes several preinstalled virtual application patterns. After the application requirements are defined, such as what services are used and the QoS that should be applied, PureApplication System automatically deploys and configures the appropriate middleware components to run the application with optimal performance. All virtual application patterns support integrated monitoring, scaling, and failure recovery capabilities.

Virtual Appliances

PureApplication System also supports the virtual appliance deployment model, which allows the system to run custom software images that were previously modified. However, virtual appliances do not feature the robust management and monitoring features that are available for virtual systems and virtual applications. These features are useful when the pattern requires a software product from another vendor, or corporate standards require that all deployed instances contain a particular operating system component.

The following methods are supported to create custom images:

- The extend and capture function allows customizing a single image and then saving into the catalog as a new image.
- The Image Construction and Composition Tool allows the customer to build custom images for the operating system and middlewares.

1.3.3 Management

The IBM PureApplication System provides a single unified interface for managing the entire system. The integrated console provides management and monitoring interfaces for system hardware, virtualized storage and networking, license usage monitoring, user auditing, and security configuration. All of these system management functions are provided through the same interface that is used for monitoring workloads.

Monitoring

PureApplication System includes pre-configured capabilities for monitoring all the hardware and software components that are provided with the system. Workload monitoring capabilities include hypervisors, operating systems, and entitled middleware and database products. It also supports more workload types for non-included products through expanded monitoring capabilities.

Maintenance

PureApplication System maintenance is divided into the following categories:

- System maintenance: Includes updates for components that are part of the system, such as hardware and firmware components, compute nodes, switches, storage, hypervisors, and management software.
- Workload maintenance: Include updates to everything that is contained within the virtual machines that run workloads on the system, such as operating systems, middlewares, databases, and updates to virtual system and virtual application patterns.

These maintenance types can be applied independently because they are likely to be applied by different teams and on different schedules.

Licensing

PureApplication System also includes tools for managing software license usage within the system. License usage can be monitored on a per-server basis or by using the processor value unit (PVU) model, depending on which type of software license was purchased. PureApplication System tracks the licenses that are used for that product in real time as virtual machines are created and destroyed so that individual users do not have to know how many licenses are in use by other users. In addition to monitoring and optionally enforcing license usage limits, PureApplication System supports exporting historical license usage information for use in spreadsheets and other tools for analyzing license usage over time.

1.3.4 Problem determination

IBM PureApplication System includes several capabilities to assist with troubleshooting. The system logs track actions that are performed by the components and enables the administrators access to detailed operational information that is contained within the kernel, error, and trace files. It also provides information regarding the addition, modification, and deletion of auditable objects, such as users, virtual systems, patterns, and other items.

The user also has access to all of the logs that are created by the workloads, such as WebSphere Application Server, IBM HTTP Server, and DB2 logs. In addition to the log data, the system collects and notifies the user of events that are generated by the system hardware and software. They are displayed in the event notification panel in the system console, with which you can view events by severity, type, status, and time, among other criteria.

The integrated console also provides a tool that is called infrastructure map, which provides a graphical view of the system hardware that makes it easy to identify possible issues and drill down to gather more information. It displays data for all of the hardware components in the system, including overall status, graphical representations of hardware LEDs, usage, temperature, and performance data, as shown in Figure 1-6.

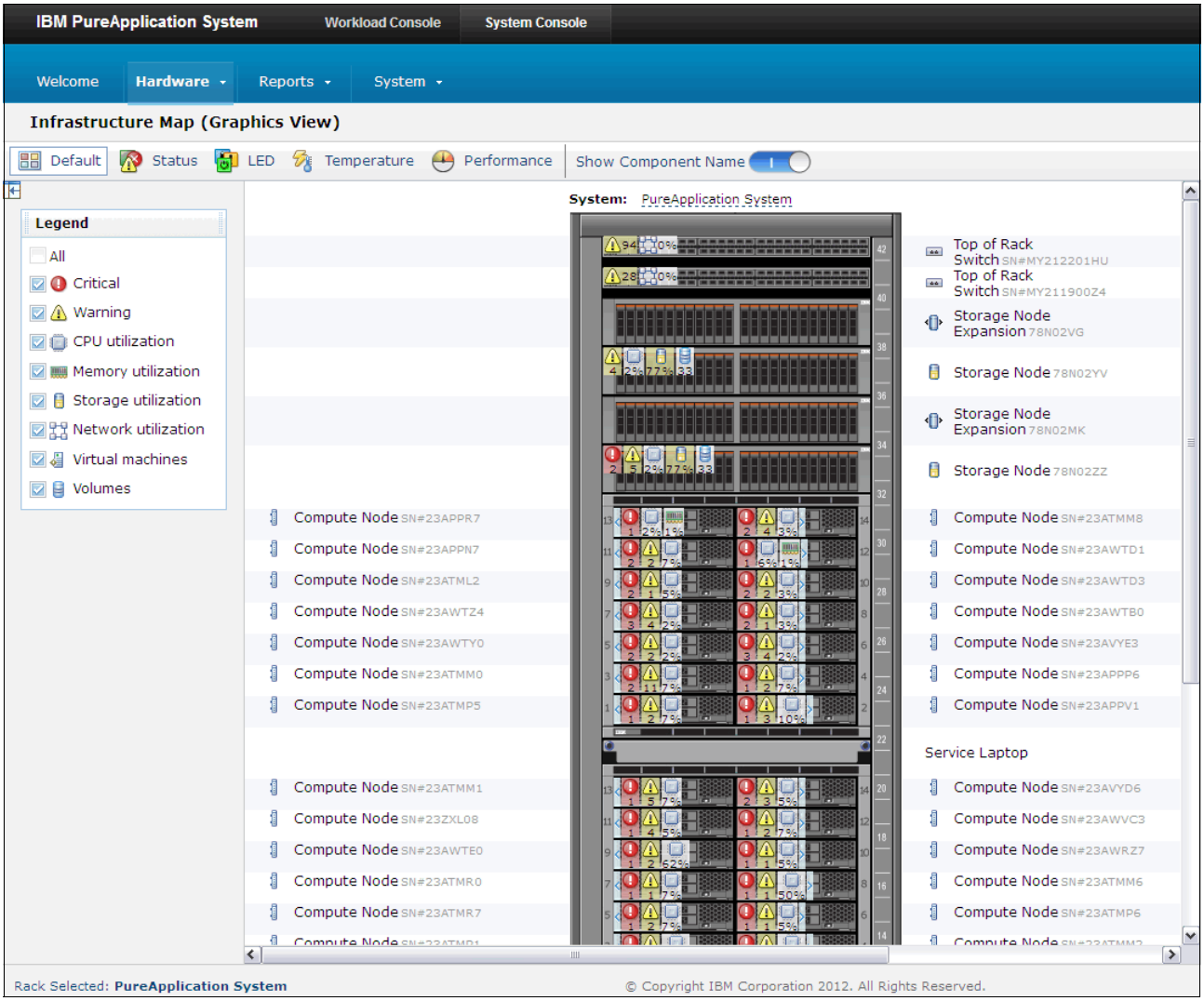


Figure 1-6 The hardware infrastructure map in PureApplication System console

The entire system and basic status are displayed by default in the infrastructure map. If there are errors or warnings for any of the hardware components, they are shown directly in this view. Clicking any component in the map displays more detailed information about the expansion unit as a whole, such as errors, warnings, and usage statistics.

For more information about problem determination and troubleshooting, see Chapter 8, “Troubleshooting PureApplication System Environment” on page 329.

1.4 Scope of this book

The goal of this book is to introduce new users to the concepts of IBM PureApplication System and describe the most common problems, solutions, best practices, and advanced use cases around adopting IBM PureApplication System V1.0.

The following topics are addressed in this book:

- Chapter 1, “Introduction” on page 1 provides a brief overview of the cloud computing concepts, explains the differences between IBM PureSystems offerings, and describes the fundamentals of IBM PureApplication System.
- Chapter 2, “Integrating IBM PureApplication System into an existing data center” on page 21 describes the necessary steps to integrate an existing environment with PureApplication System. It shows the prerequisites, main interfaces, and connection options.
- Chapter 3, “Usage patterns for isolating applications” on page 61 describes how applications that are deployed into a cloud environment can run independently from each other. It also explains how isolation separates application traffic through the network and prevents cross application impact that is caused by resource contention.
- Chapter 4, “PureApplication Deployment Models” on page 107 provides a review of the deployment models that are available in PureApplication System, explains their main features, implementation strategies, topologies, and tools for each model.
- Chapter 5, “Customizing Virtual System Patterns” on page 199 describes the various resources and tools that are available for customizing virtual system patterns, managing virtual images, script packages, add-ons, and configuring advanced options in PureApplication System.
- Chapter 6, “Customizing Virtual Application Patterns” on page 267 describes the various interfaces available for customizing virtual application patterns, such as extending existing *out-of-box* patterns or developing plug-ins.
- Chapter 7, “Integrating PureData for Transaction” on page 309 describes the advantages, prerequisites, and sample scenarios about how to use PureData System with PureApplication System.
- Chapter 8, “Troubleshooting PureApplication System Environment” on page 329 provides helpful information about the tooling, logging, and debugging features that are available in IBM PureApplication System for problem determination and troubleshooting.
- Chapter 9, “High Availability and Disaster Recovery” on page 377 covers the main failure scenarios for the different deployment models in PureApplication System to achieve high availability and perform disaster recovery.

1.4.1 Intended audience

The target audience for this book is anyone from the IT industry, including technical consultants, business partners and independent software vendors, who are considering migrating to a cloud computing solution or want to acquire a better understanding of IBM PureApplication System.

In addition, the audience includes system administrators, middleware specialists, and software engineers who seek a more in-depth exposure to PureApplication System features and capabilities.

1.4.2 Assumptions

This book makes the following assumptions to simplify the information and make it useful to as many customers as possible:

- The user has access to a deployed environment of IBM PureApplication System v1.0.
- The user is entitled to access the system and is assigned the proper roles and privileges to complete the required tasks.
- The user is familiar with the following information:
  - Hardware concepts: Computing, storage, memory, networking
  - Virtualization terminology: VMware ESX, hypervisor, virtual machine, image
  - Networking concepts: TCP/IP, DNS, NTP, LAN, VLAN, VPN, proxy, gateway
  - Security concepts: SSL, SSH, LDAP, user, group, role
  - Operating systems: UNIX, Red Hat Enterprise Linux (RHEL)
  - Middlewares: WebSphere Application Server, wsadmin
  - Web servers: IBM HTTP Server (IHS)
  - Enterprise applications: Java Platform, Enterprise Edition; JVM; EJB; EAR; WAR
  - Databases: DB2 Enterprise, SQL, DDL
  - Scripting languages and notations: Shell script, Python, Jython, JSON
- The user has access to the following software and tools that are required for completing the tasks that are described in this book:
  - Virtual Pattern Development Kit (VPDK)
  - Plug-in Development Kit (PDK)
  - Command-line interface (CLI)
  - Image Construction and Composition Tool
  - Advanced Middleware Configuration (AMC)

2

Integrating IBM PureApplication System into an existing data center

This chapter describes how to integrate the IBM PureApplication System into an existing environment. Sections include planning information and show the main connection options for the system. This chapter highlights the system-provided connection options and shows some best practices for consideration when you plan for these elements.

The following topics are covered in this chapter:

- Prerequisites in the data center
- The Genesis process
- Interfaces for using and administering IBM PureApplication System
- Security
- License management
- Networking
- Auditing
- Monitoring
- Integration with external backup systems
- Export and import features
- External database integration
- Integrating shared network storage to virtual machines

© Copyright IBM Corp. 2013. All rights reserved.

2.1 Prerequisites in the data center

PureApplication System is an integrated solution. Physically, it looks like another rack in the server room. This section describes the needs and preparation for the equipment in the new environment.

2.1.1 Physical requirements

Every component of the PureApplication System is in a standard 42-unit height enterprise rack. The necessary space for this equipment is the first physical requirement.

Electrical power consumption of the whole system is between 14.173 kVA and 32.770 kVA, depending of the configuration. Consider calculating and building for power consumption at maximum because this preparation supports future growth and development.

For more information about planning, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/gst_planning.html

2.1.2 Software infrastructure requirements

A central time server is necessary for workloads to function seamlessly. Hypervisors and the VMs synchronize their time to this server. The basic way to synchronize the time between the servers is by using a network time protocol (NTP). This protocol provides a common interface to keep synchronized systems in the network. If the NTP server is inaccessible, all hypervisors use their own hardware clock and sometimes can get out of sync. Being out of sync leaves the system vulnerable and can create many problems during deployments.

Domain Name System (DNS) is a hierarchical distributed naming system for all resources that are connected to the network (intranet and internet). This domain name server also is an important part of the whole infrastructure. IBM PureApplication System uses DNS lookup services for communication. A DNS server is one of the required parameters for system initialization.

2.2 The Genesis process

To ensure the integrity of the IBM PureApplication System, IBM has a standardized process that runs when a customer orders a new system.

That process is the Genesis process. As a part of this process, some steps are run at the IBM location. The final step is run at client location with the audience of the customer who operates the system. The following steps are completed at the IBM location:

- Collect all the necessary information about each part.
- Check the assembly of the parts.
- Check the integrity of cabling.
- Run standardized tests on the new system.
- Update firmware and other software components.
- Verify, check, and complete the initial setup of the IBM PureApplication System for the client environment by using the worksheet that is completed by the customer.

The product is then shipped to the customer and at the client location. With the supervision of the client, an IBM Customer Engineer makes the final system configuration. Finalizing the new system settings includes steps such as setting up the management VLANs, switch port setup, and management of IP addresses. After this process is finished, a full reinitialization is done to change some system parameters.

During the initialization process, you must specify the following parameters:

- VLAN ranges for what you want to use: This configuration can be modified later from the management interface. For quicker integration and the first deployment test, initial parameters are required.
- Internal VLANs: These VLANs are reserved for internal communication and cannot conflict with any other VLANs. These values cannot be changed later.
- Virtual Link Aggregation Group Tier ID (VLAG Tier ID): This unique ID is used to connect other VLAG switches.
- Port setup: Configure rack switches.
- Link aggregations and configurations: Every switch port must be aggregated for best performance and stability. Configuration parameters help build these links.
- Management IP address and host name: This IP address or host name can be used for system management.
- DNS server address.
- NTP server address.
- IP group for the first test deployment.
- Administrative user and password.

2.3 Interfaces for using and administering IBM PureApplication System

A suitable interface must be chosen to best use the system. Usually, the best interface is a web-based GUI. However, when many similar commands are processed, Command-line interface (CLI) can be the best. IBM PureApplication System provides several interface options for system management.

2.3.1 Web GUI

Web GUI interfaces are the most mainstream way to administer the whole system. These interfaces do not need any extra installation on the client machine; only a web browser is required. Based on configured permissions, the interface shows all available options to use the system.

GUI structure

At the top of the initial IBM PureApplication system web page, you find the common options such as user name, help, logout, and the console selector links. There are two consoles available, the workload console and the system console. Both of these consoles include a welcome page that shows the main console with specific tasks.

Workload console

The workload console provides system management and administration for the cloud environment resources. Figure 2-1 shows the Welcome page of workload console.

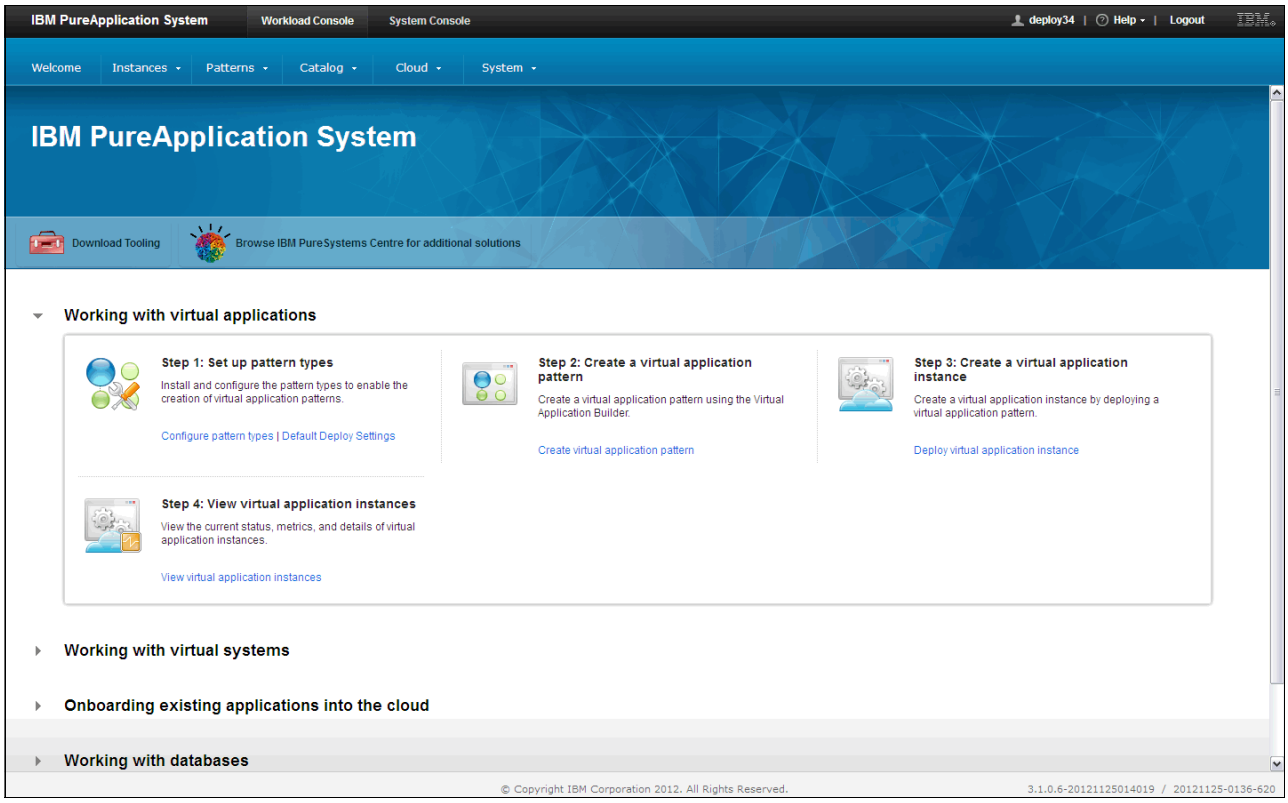


Figure 2-1 IBM PureApplication System: Workload console welcome page

System console

The system console covers system management and administration tasks for the hardware and software that are integrated in the system. Figure 2-2 shows the Welcome page of system console.

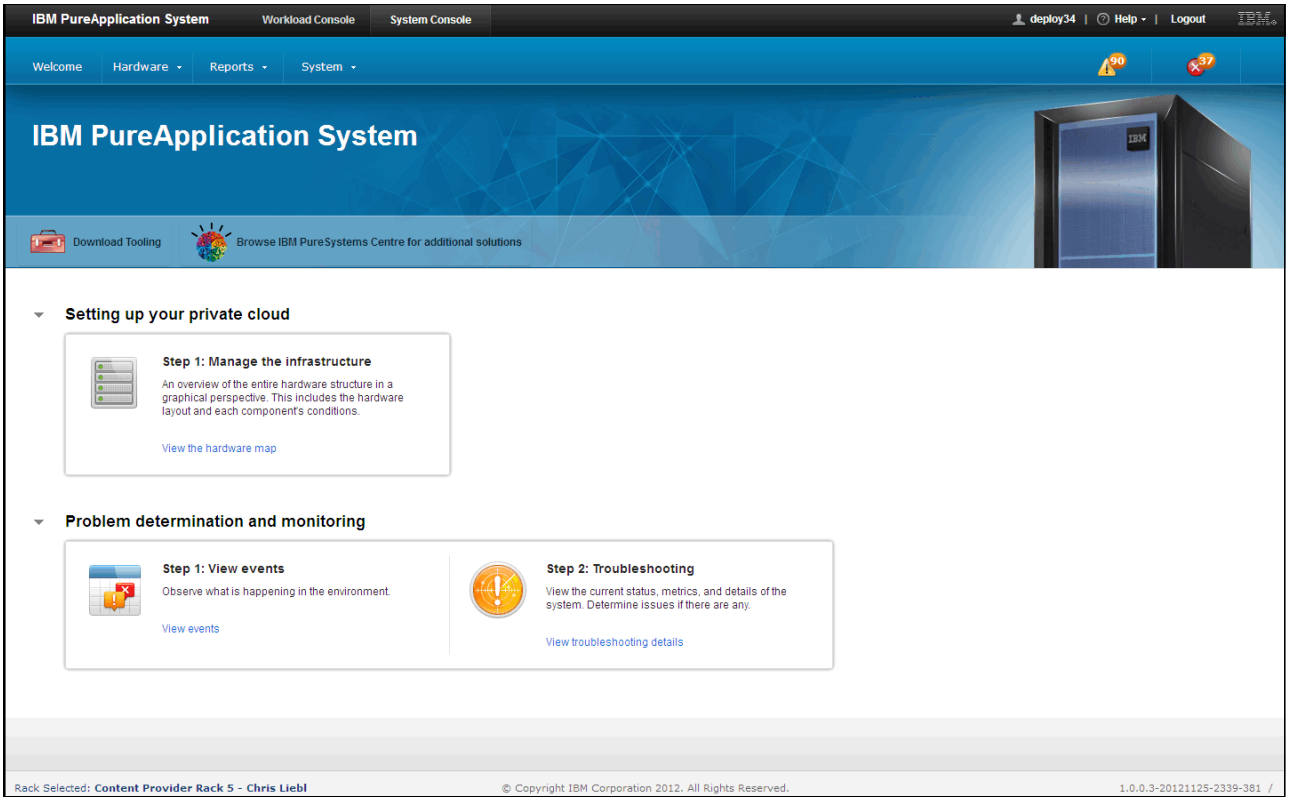


Figure 2-2 IBM PureApplication System and system console

2.3.2 Command-line interface

The IBM PureApplication System CLI provides a Jython-based scripting environment. The CLI is available for download from the GUI Welcome page. Figure 2-3 shows the download link on the welcome page.

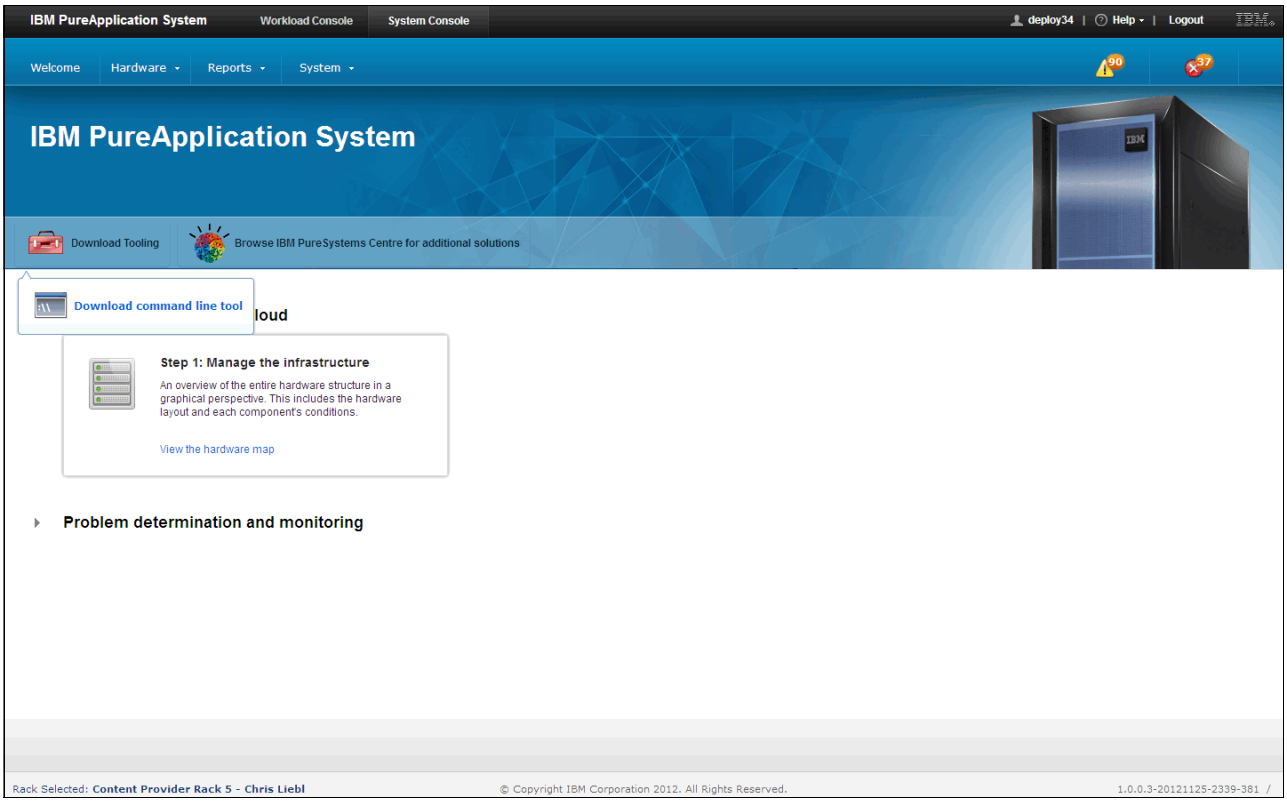


Figure 2-3 CLI download link on the welcome page

The CLI requires an installed Java V6+ runtime environment. Currently, this interface is supported on Linux and Windows OS.

The CLI contains all the same options that are available in the GUI and provides an interface to run complex automation scripts. The CLI package contains some other scripts for easy maintenance of the system.

For more information about CLI, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/cct_usingcli.html

The following options and examples show how to connect to the system by using the CLI:

- Connect to PureApplication System interactive shell:

  `pure -h` *`hostname`* `-u` *`username`* `- p` *`password`*

  Tip: Specify only the host name or IP address; never use the full URL.

- Run a single command on system:

  `pure -h` *`-hostname`* `-u` *`username`* `-p` *`password`* `-c` *`command`*

- Run a script file on system:

  `pure -h hostname -u username -p password -f file_name [args]`

- CLI commands:
  - Access the built-in content help:

    ```
    help(..)
    help(deployer)
    help(admin)
    ```

  - Query virtualsystems:

    ```
    deployer.virtualsystems
    ```

Example 2-1, Example 2-2 on page 28, and Example 2-3 on page 28 give more specific information about the use of CLI commands.

Example 2-1 Example session to query virtual system from a command line

```
D:\Work\PureSystem\pure.cli\bin>pure -h 172.18.64.32 -u demouser -p
 demopassword -c deployer.virtualsystems
[
  {
    "acl": (nested object),
    "created": 2012.11.29. 19:20:46,
    "currentmessage": "RMO7045",
    "currentmessage_text": "The virtual system has been deployed",
    "currentstatus": "RMO1OO6",
    "currentstatus_text": "Started",
    "desiredstatus": "",
    "desiredstatus_text": "",
    "environmentprofile": (nested object),
    "id": 24O,
    "maintenances": (nested object),
    "name": "RHEL-GYG",
    "owner": (nested object),
    "pattern": (nested object),
    "priority": 16,
    "snapshots": (nested object),
    "updated": 2012.11.29. 19:28:26,
    "virtualmachines": (nested object)
  },
  {
    "acl": (nested object),
    "created": 2012.11.3O. 14:25:15,
    "currentmessage": "RMO7045",
    "currentmessage_text": "The virtual system has been deployed",
    "currentstatus": "RMO1OO6",
    "currentstatus_text": "Started",
    "desiredstatus": "",
    "desiredstatus_text": "",
    "environmentprofile": (nested object),
    "id": 248,
    "maintenances": (nested object),
    "name": "VS RB-SA-W214-RO1 - Test 1",
    "owner": (nested object),
    "pattern": (nested object),
    "priority": 4,
```

```
    "snapshots": (nested object),
    "updated": 2012.11.30. 14:54:29,
    "virtualmachines": (nested object)
  }
]

D:\Work\PureSystem\pure.cli\bin>
```

Example 2-2 Deploy virtual system from command line

```
dep=deployer.applications.get("a-90d408a5-735a-429d-b974-711abd99c276").deploy("CL
I_Redbooks Test",
{"environment_profile":deployer.environmentprofiles['envProfile34'],
"cloud_group":deployer.environmentprofiles['envProfile34'][0].clouds.keys()[0],
"ip_group":deployer.ipgroups['ipgroup34'][0], "ip_version":'IPv4'})

print dep.id, dep.status
```

Example 2-3 Deploy three virtual system from same script

```
vms=[]
for vmname in ["CLI1","CLI2","CLI3"] :
vms.append(deployer.applications.get("a-90d408a5-735a-429d-b974-711abd99c276").dep
loy(vmname,{"environment_profile":deployer.environmentprofiles['envProfile34'],
"cloud_group":deployer.environmentprofiles['envProfile34'][0].clouds.keys()[0],"ip
_group":deployer.ipgroups['ipgroup34'][0] ,"ip_version":'IPv4'}))

for vm1 in vms: print vm1.id, vm1.status
```

2.3.3 REST API

Functional subsets are implemented in the representational state transfer (REST) application programming interface (API). The REST API is available by using the IP address or host name of the system. For more information about REST API configuration requirements and the full REST API reference, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/rat_overview.html

Example 2-4 gives a detailed display of the use a REST session.

Example 2-4 Sample REST session to query virtual systems

```
GET:
https://172.18.64.32/resources/virtualSystems
RESPONSE:
[
   {
      "desiredstatus_text": null,
      "currentstatus_text": "Started",
      "name": "RHEL-GYG",
      "desiredstatus": "",
```

```
      "pattern": "/resources/patterns/2",
      "id": 240,
      "currentmessage_text": "The virtual system has been deployed",
      "created": 1354213246476,
      "currentstatus": "RM01006",
      "creator": "deploy34",
      "currentmessage": "RM07045",
      "owner": "/resources/users/96",
      "updated": 1354213706895
   },
   {
      "desiredstatus_text": null,
      "currentstatus_text": "Started",
      "name": "VS RB-SA-W214-R01 - Test 1",
      "desiredstatus": "",
      "pattern": "/resources/patterns/208",
      "id": 248,
      "currentmessage_text": "The virtual system has been deployed",
      "created": 1354281915208,
      "currentstatus": "RM01006",
      "creator": "deploy34",
      "currentmessage": "RM07045",
      "owner": "/resources/users/96",
      "updated": 1354283669473
   }
]
```


2.4 Security

An internal security system controls all access to PureApplication System resources that is based on user and group permissions. The IBM PureApplication System has an internal database to store user information and privileges, but it can use external LDAP server data for user authentication. The use of an LDAP server can help the data center administrator to manage the users lifecycle from the same system. The use of internal permission control is required because the LDAP server cannot serve permission information for the PureApplication System. The system also provides audit capabilities to monitor the system changes.

For security considerations, it is appropriate to separate the duties within the system. One of the most important directives is to isolate the assignments of auditing permissions to users who do not have other administrative capabilities.

2.4.1 User roles

The PureApplication System provides comprehensive options to adjust privileges. Management privileges are divided into five major areas. Each administrative role has a read-only or a full access option. The read-only option allows viewing of current settings only. The full access option has privileges to modify the settings. Another important option is the delegation role. With the delegation role, the user can delegate their own administrative access to another user.

For control of the administrative privileges, review the following available options:

- Workload resource administration read-only permission: Users with this role can view all workload-related configuration and status.
- Workload resource administrator full permission: Users with this role can manage all workload-related resources. This role also contains all read-only functions.
- Cloud group administration read-only permission: Users with this role can view all cloud resource configurations.
- Cloud group administration full permission: This permission also contains all cloud group read-only administrator rights and the management permission for these resources.
- Hardware administration read-only permission: Users with this permission can view all hardware-related resources.
- Hardware administration full permission: Users with this permission can manage the system hardware resources.
- Auditing read-only permission: Users with this permission can view auditing settings and audit date.
- Auditing full permission: Users with this permission can change auditing settings; for example, setting the external storage server connection data to automatically archive the audit log.
- Security administration view user/groups permission: Users with this permission have read-only access to view users and groups.
- Security administrator view all security resources permission: Users with this permission have read-only access to view all security resources.
- Security administrator manages security permission: Users with this permission can manage all security resources and can grant and revoke access rights.
- Allow delegation when full permission is selected: By using this option, the owner of the administration privilege can delegate their permission to another user.

The other important part of the permission controls is the workload management roles. The system provides the following options to control the access of resources:

- Deploy patterns in cloud: Every user has this permission and can deploy existing patterns to the system.
- Create new pattern: By using this permission, the user can create virtual systems and virtual application patterns.
- Create new environment profiles: By using this permission, users create environment profiles to group related cloud topology settings for easy deployment of virtual system patterns. The environment profile creator can edit and delete their own profiles.
- Create new catalog contents: Users can add objects to the PureApplication System catalog with this permission.
- IBM License Metric Tool: Users with this permission can start tool-related REST API calls to manage product licensing.

2.4.2 Group level permissions

The security system also provides group-level permission control options. By using group-level control, you can create groups of users with the same level of access to resources. With groups, the administration of roles can be more transparent.

2.4.3 Customer engineer access

Sometimes assistance is needed from IBM after a client moved through their troubleshooting procedures but did not resolve the failure. The system provides a special access flag that enables an IBM CE access to the system for 36 hours.

This option is available by browsing to **System Console** → **System** → **Troubleshooting**.

2.4.4 LDAP integration

Most large data centers have their own directory system, which helps centralized users and group management. PureApplication System also provides options to integrate with an existing LDAP directory. The LDAP authentication settings are available by clicking **System Console** → **System** → **Security**.

Figure 2-4 shows an example for LDAP settings. The LDAP parameters depend on the directory server. This sample is connected to an OpenLDAP server. After the LDAP setup, the system uses the chose server for LDAP user authentication. The content of the LDAP directory does not synchronize automatically with the system user or group directory.

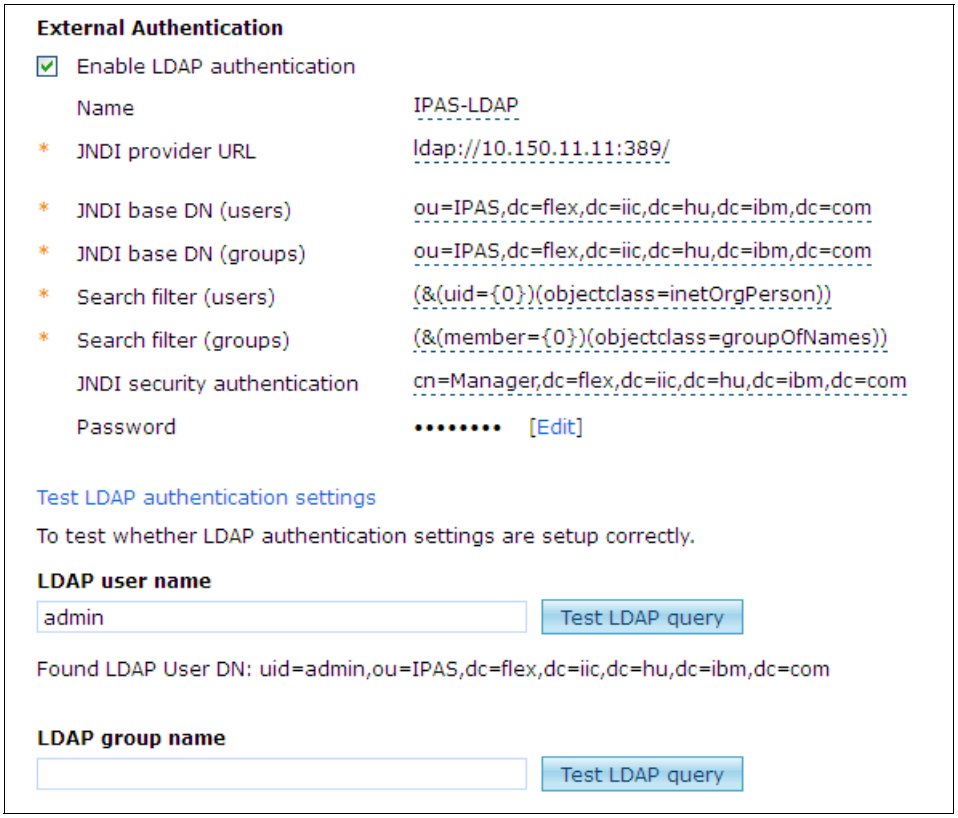


Figure 2-4 Sample external authentication settings

Automatic PureApplication System user creation that is based on LDAP

After LDAP authentication is enabled, the system can use LDAP passwords to authenticate the users. Before the user who was declared in LDAP can access the system resources, the security administrator must add the user to the PureApplication System user registry. This procedure can be a long process if many users need access to the system. In version 1.0 of PureApplication System, the GUI environment cannot automate the user creation process. If this ability is necessary, you must write a CLI script. Jython language provides elements to query LDAP user names and functions to create a user. In this case, the user creation method parameters must contain same user name as the LDAP user name.

2.5 License management

IBM PureApplication System contains some bundled software licenses and it provides some services to help the customer control the system's license usage. One of the usable license management tools is the IBM License Metric Tool. This component is available (at no cost) to monitor the IBM software license usage. Also, it can handle the virtualization environment special requirements.

2.5.1 Bundled software licenses

The following products are delivered with the system as integrated patterns. PureApplication System includes entitlement to run the following products up to the total capacity of that configuration:

- IBM WebSphere Application Server Hypervisor Edition, V7.0 and V8.0
- IBM DB2 Enterprise Server Edition, V9.7 FP5 and V10.1
- IBM Web Application Pattern, V1.0 and V2.0
- IBM Transactional Database Pattern V1.1
- IBM Data Mart Pattern V1.1
- IBM Application Pattern for Java V1.0
- IBM OS image for Red Hat Linux Systems V2.0

2.5.2 IBM License Metric Tool

The IBM License Metric Tool is used for license tracking purposes across a range of IBM software products. The IBM License Metric Tool is packaged independently and is available as a software download from IBM that is separate from the PureApplication System. The PureApplication System supports communication with the IBM License Metric Tool to produce enterprise-wide license usage reports. To use this feature, you must point to an IBM LMT server in the PureApplication System configuration. The virtual images that the PureApplication System deploys include the IBM License Metric Tool agent. This agent runs on the virtual machines (VMs) that are deployed in the cloud and collects data about the software that is running there and the licenses that are used. The agent then sends this information to the IBM License Metric Tool server.

2.5.3 License information

PureApplication System provides information about the used licenses and features capabilities to configure the license usage limits. The system also provides the option to download filtered license statistic information. The license information about the system is available by browsing to **System console** → **System** → **Product licenses**. Figure 2-5 shows an example license statistic.

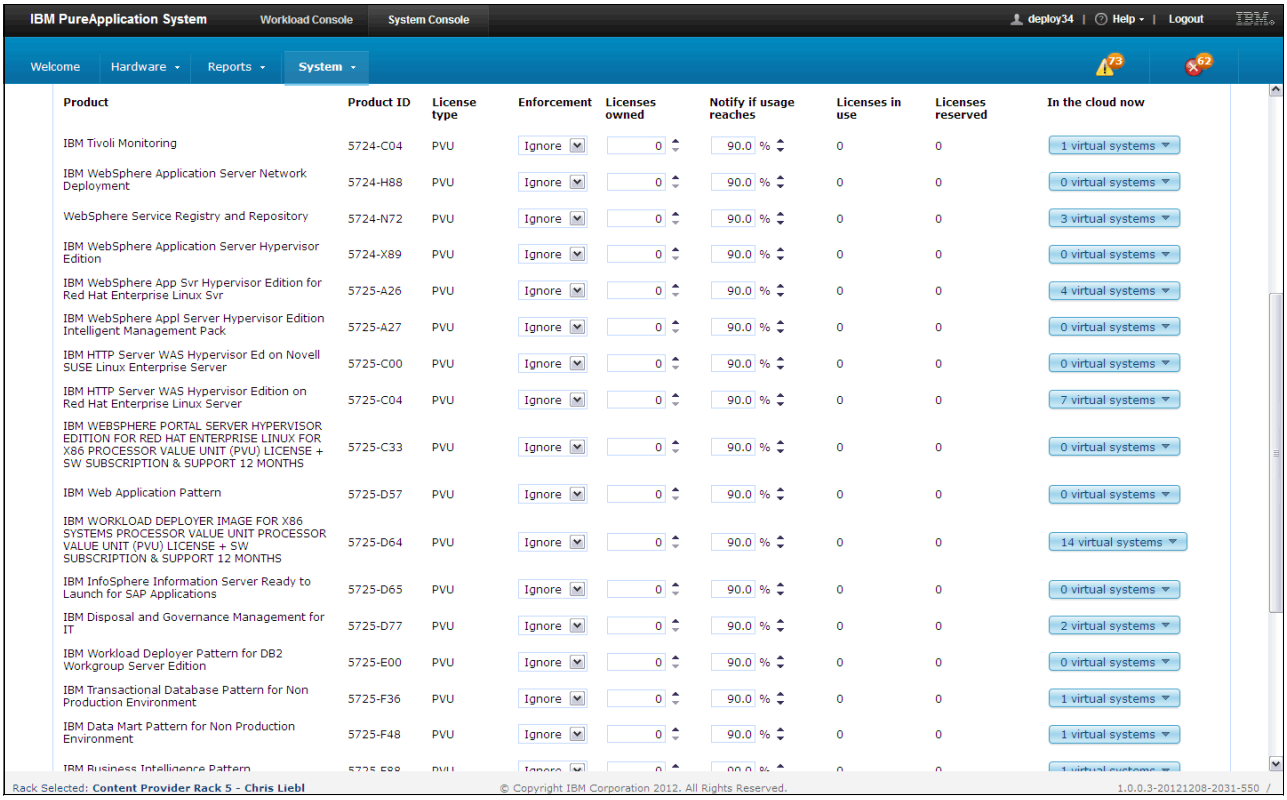


Figure 2-5 Sample license usage statistic and settings

2.6 Networking

The network is the only connection between the system and the outside world. If something is wrong in the network configuration, the whole system might be inaccessible. Therefore, care must be taken when the network environment is planned.

2.6.1 Networking hardware in IBM PureApplication System

From the hardware side, every PureApplication System contains three Flex System Enterprise chassis and two top-of-rack switches with 64x10 Gb ports. Every chassis contains two10 Gb Ethernet switches. The top-of-rack switches provide the connection to the outside network. Each compute node has two Ethernet controllers, each with 4x10 Gb Ethernet ports.

SAN network: The system also contains a storage area network (SAN), which is used internally and does not connect to any external SANs. In the current version, the SAN cannot be extended.

Details of all hardware components, including the network devices that are within the system, are available by browsing to **System console** → **Hardware** → **Infrastructure map**. The hardware menu also contains more options to filter the devices according to function. The details of the network part of the system are available by browsing to **System console** → **Hardware** → **Network devices**.

Figure 2-6 shows an example list from the network devices.

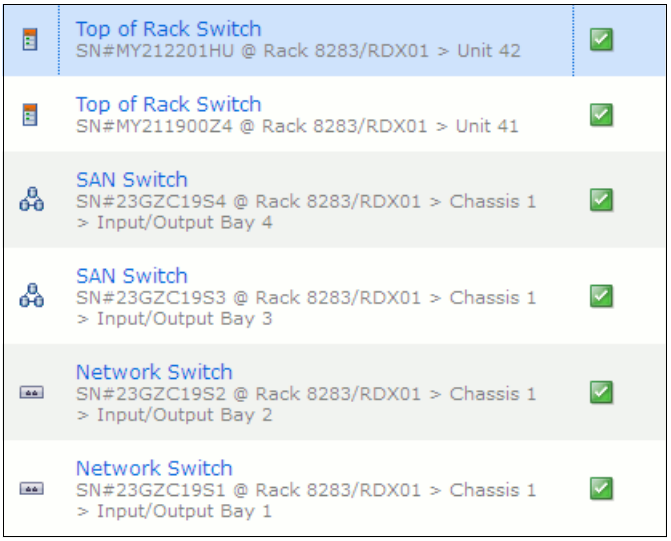


Figure 2-6 Example list from network devices

Figure 2-7 shows an example of top-of-rack switch details.

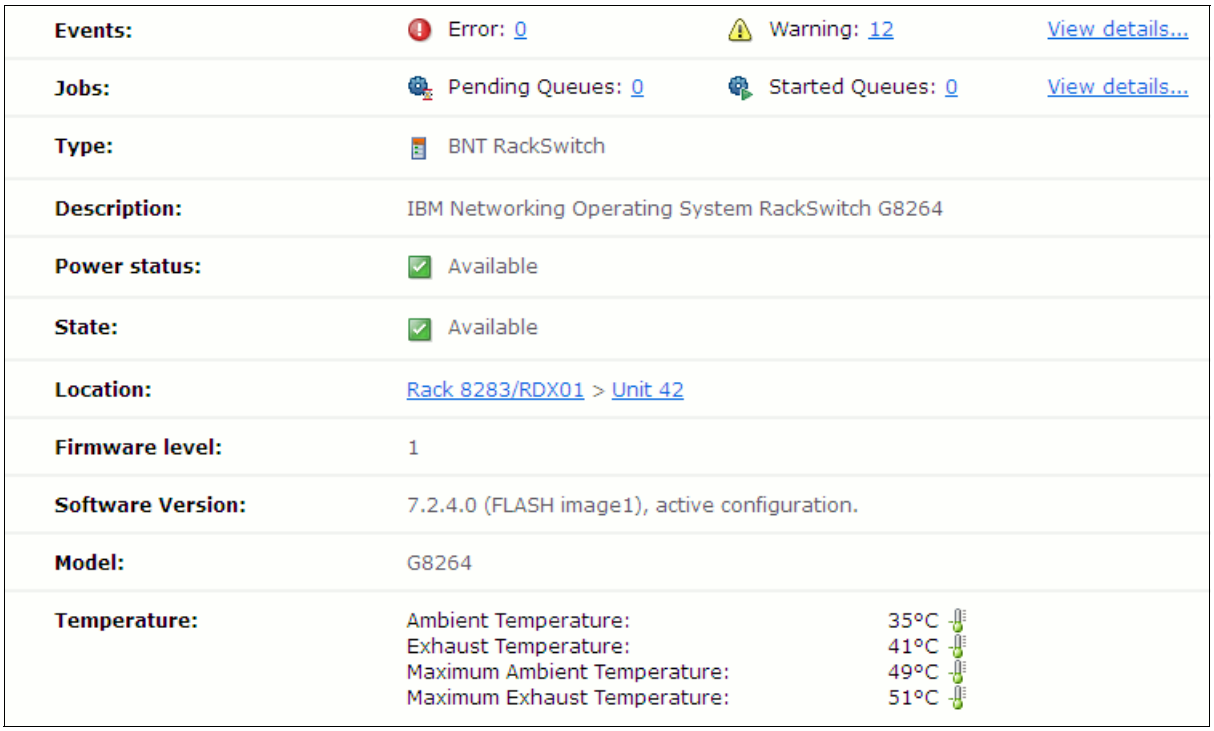


Figure 2-7 Example of top of rack switch details

Figure 2-8 shows top-of-rack for the customer port details.

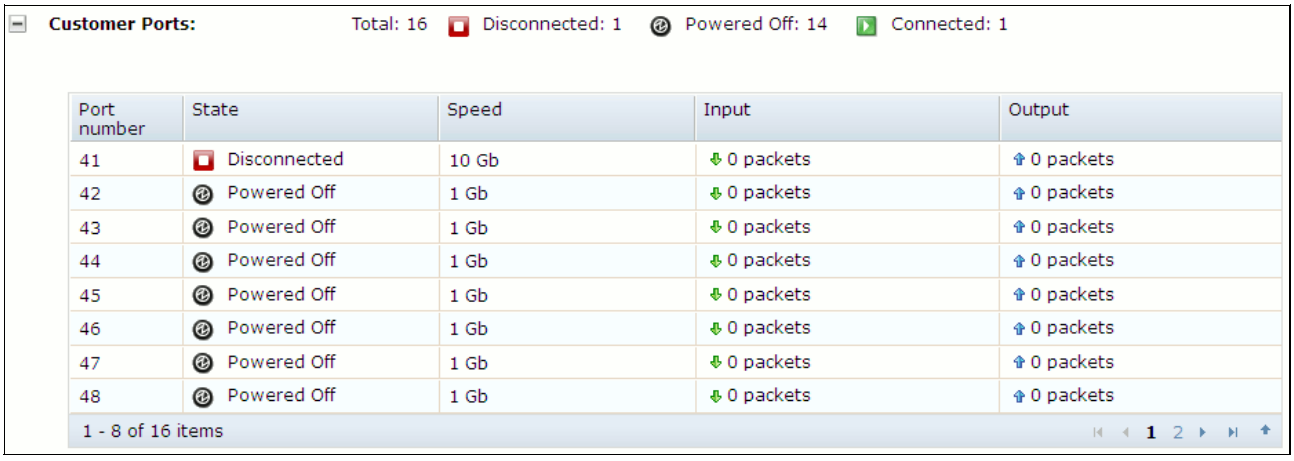


Figure 2-8 Example of top of rack switch customer ports details

2.6.2 Main networks

For secure and reliable operation, it is necessary to separate the network traffic that is based on functionality. IBM PureApplication System uses four main network segments. Each segment has its own functionality. The system uses IPV6 for internal communication and IPV4 or IPV6 for the external network.

The following list describes the four main network segments:

- External access network

  The external network connects to the customer network. It can be segmented into multiple VLAN ranges. It is configurable to 1/10 Gb, depending on the customer network needs. Workloads use the external network for all in and out traffic.

- Console network

  The system uses this internal network to communicate with management tools, such as the system console.

- Mobility network

  When workload relocation is required, the system uses this internal network to move the virtual machines between the nodes.

- Management network

  The management network is an internal network that is used for communication between the management and compute nodes.

2.6.3 Port setup

Top-of-rack switches provide 2x16 ports to connect the system to the external network. Ports 41-56 in each switch are available for connectivity to external networks. At least one cable is required in each switch for proper operation. Total bandwidth between the PureApplication System and the external network depends on the used ports and the port configuration. The minimum bandwidth, in and out from the rack, is 2 Gbps and the maximum is 320 Gbps. The configuration of each port can be changed individually. A best practice is to configure the two switches identically. This configuration ensures optimum traffic flow, easier debugging, and a resilient network configuration.

Figure 2-9 shows an example port setup. The customer network configuration is available by browsing to **System Console** → **System** → **Customer Network Configuration**.

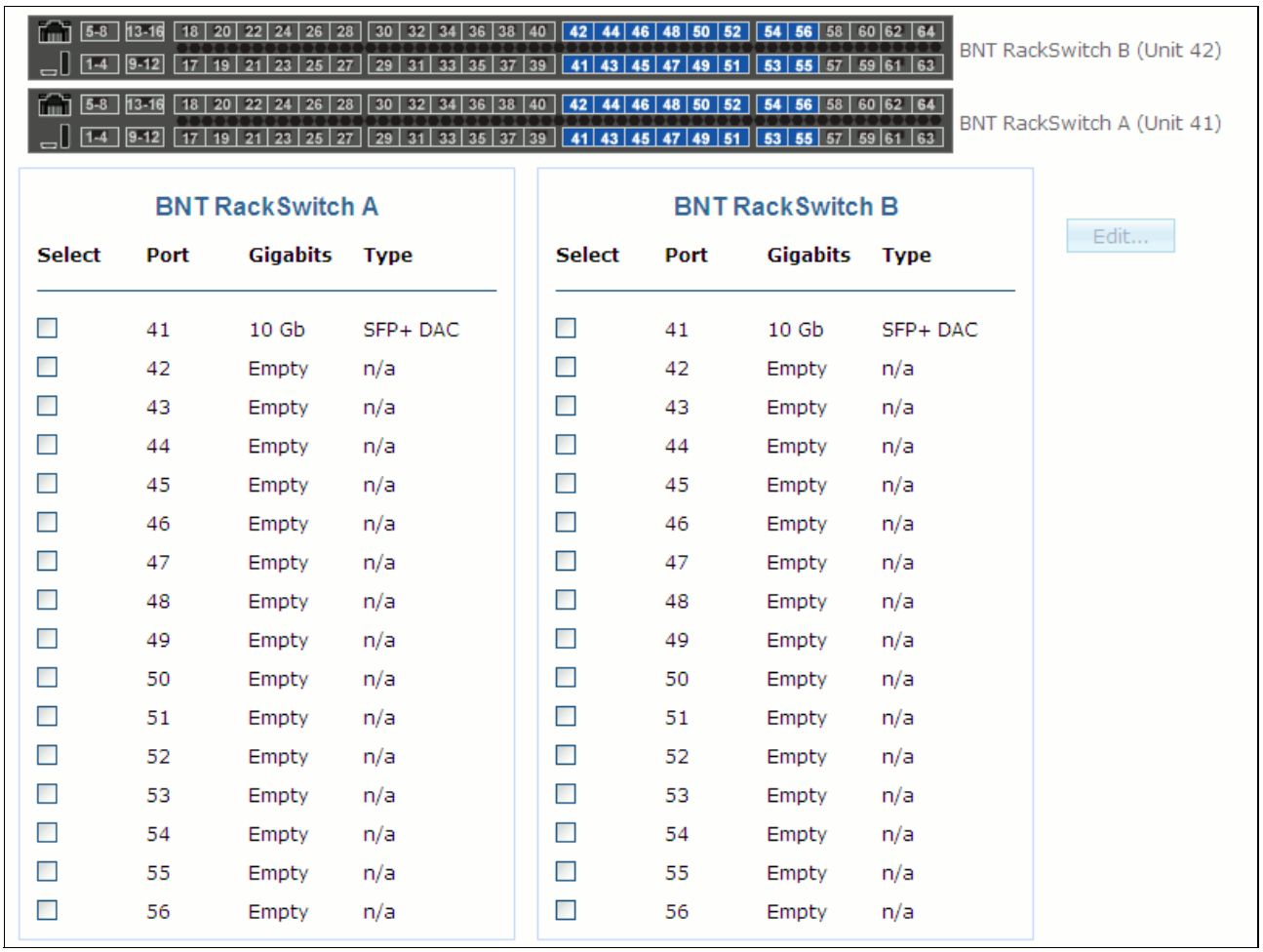


Figure 2-9 Example port configuration

Figure 2-10 shows ports that are selected at the same time to change both of their settings. Multiple selections are more efficient for port configuration. By default, when one port is selected on switch A, the same pair on switch B is selected. Ports can be cleared manually.

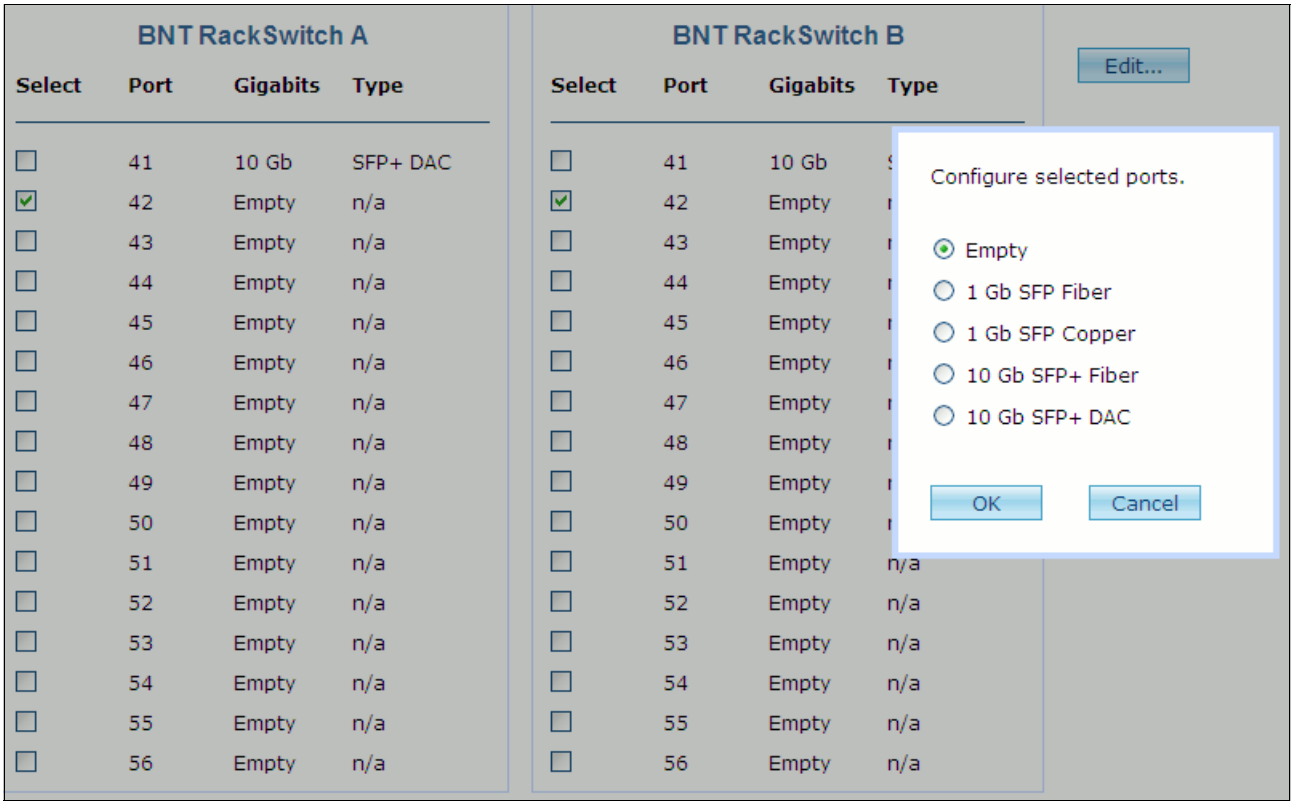


Figure 2-10 Port configuration options

An important port setting is the management port setting, as shown in Figure 2-11. The last port in every top-of-rack switch is reserved for external management access. This port provides access to the management interface of the PureApplication System.

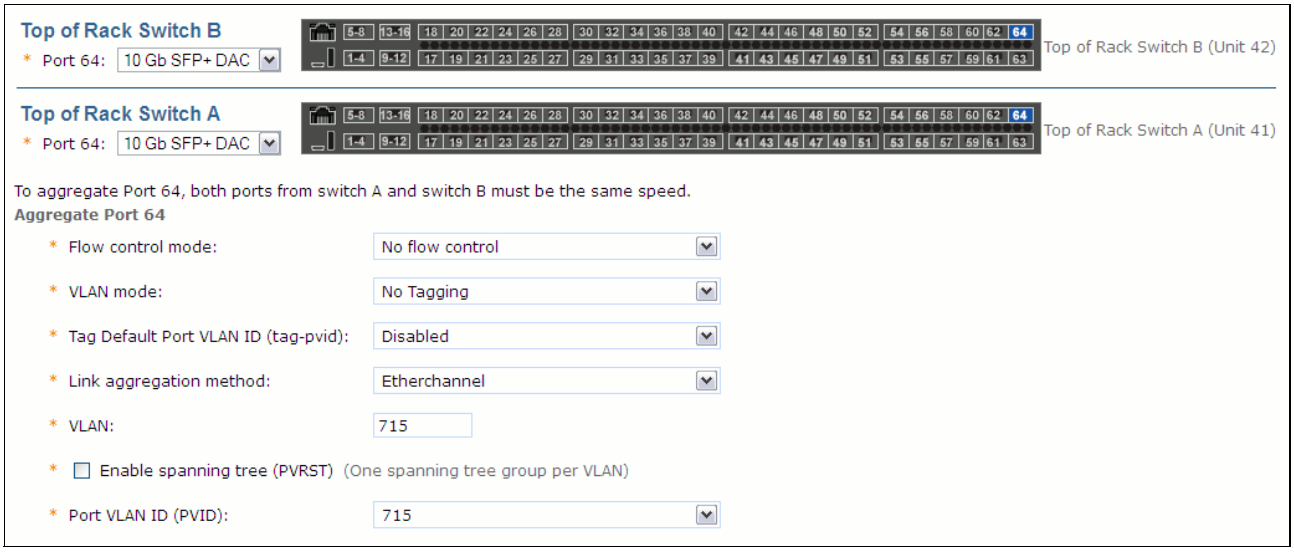


Figure 2-11 Management port settings

2.6.4 Link aggregation

After all usable external switch ports are configured, the next task is to combine these ports with an aggregated link. Link aggregation is a switch feature that enables the combination of multiple physical ports into a logical link. This configuration provides higher bandwidth and more resilient redundancy. All aggregated links must contain at least one port from each switch.

Important: Aggregated ports must be of the same speed.

2.6.5 VLANS

The basic way to sort the network traffic between the separated virtual applications is by using the virtual local area network (VLAN). VLANs can provide near physical levels of separation and protect the traffic of each virtual application. The entire system features three dedicated VLANs, which must be reserved and cannot be mixed with other VLANs.

Important: Internal network VLAN IDs, which are configured during the genesis process, cannot be changed from the administrative console.

The system provides a management interface to configure the VLAN ranges. Figure 2-12 shows the VLAN configuration part of the customer network configuration form.

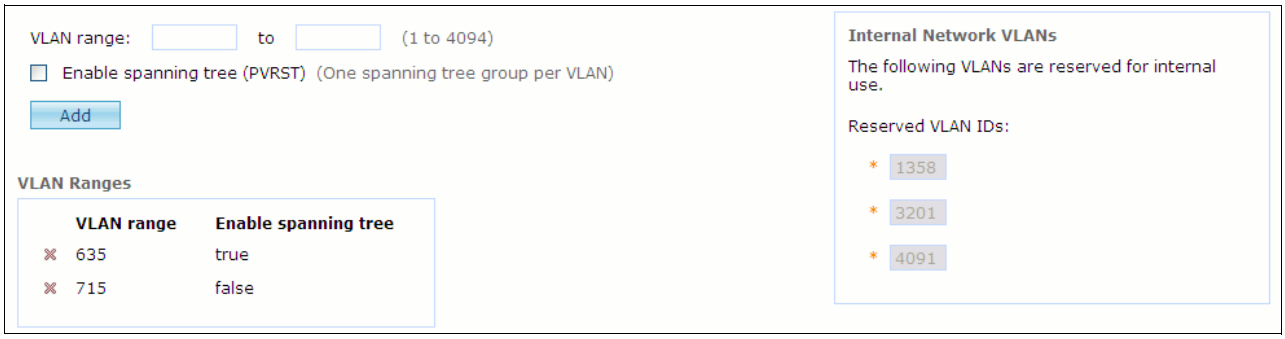


Figure 2-12 VLAN configuration form

When you consider the relation between ports, aggregations, VLANs, and IP groups, remember the following points:

- Every aggregation contains at least two physical ports.
- Every VLAN is mapped to an aggregation.
- Every IP group is mapped to a VLAN.

2.7 Auditing

PureApplication System provides a continuous logging function that records all security and administrative events. This log data can be the source of the audit log. A user who has auditing permission can read this information from the system. Audit records cannot be deleted by a user. This record stays in the system's internal database until storage has more space. If the storage full, the system automatically deletes the oldest records. To prevent the loss of audit log records, the system is configurable to back up the audit log records to an external storage. In this case, the old records are stored on the external storage before the system deletes them from the internal database.

2.7.1 Auditor roles

The system provides the following permission options to grant audit rights to a user:

- Read only permission: With read only permission, the user can view all audit data from system.
- Full access permission: With full access permission, the user can change the audit settings to use external storage for audit data archiving.

Important: Audit data modification or deletion is not possible in the system. The system deletes the oldest audit data when the audit storage is full. If external storage is configured, this data is archived into external storage.

2.7.2 External storage server for audit logs

The internal storage for the audit records is limited. Depending on the number of the events, it can be full. In this case, the system protects integrity by deleting the oldest record to make space for the new records. Based on the number of records and storage size, it might not be necessary to back up the records because deletion might not occur for some time. If the company decides it wants a longer time period to store and analyze these records, the records must be saved on external storage. The system provides options for this requirement. Only users with full auditing permissions can access this configuration option. Figure 2-13 shows the external storage options for audit logs.

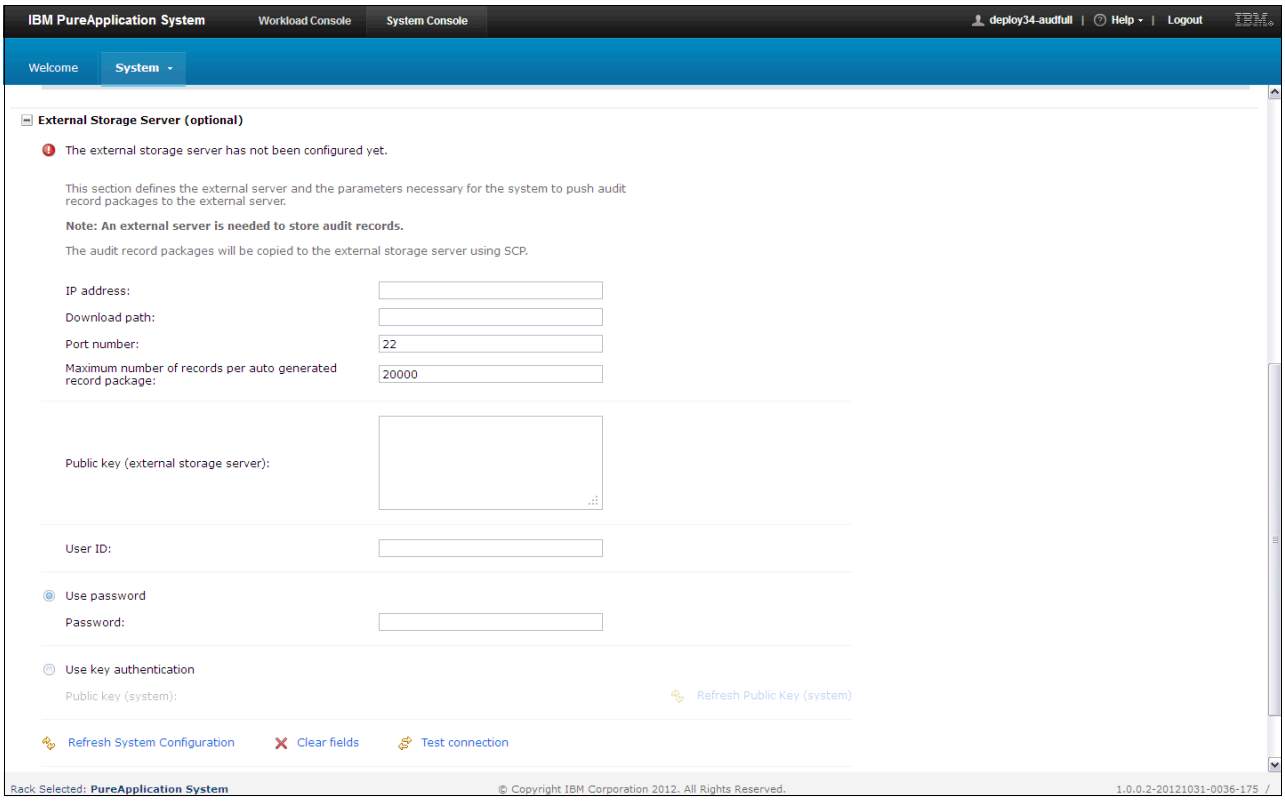


Figure 2-13 External storage server options for audit logs

2.7.3 Handling audit data

Users with at least audit reader permission can view all audit data on the system. This function is available by browsing to **System Console** → **System** → **Auditing**. Figure 2-14 shows an example of the audit data.

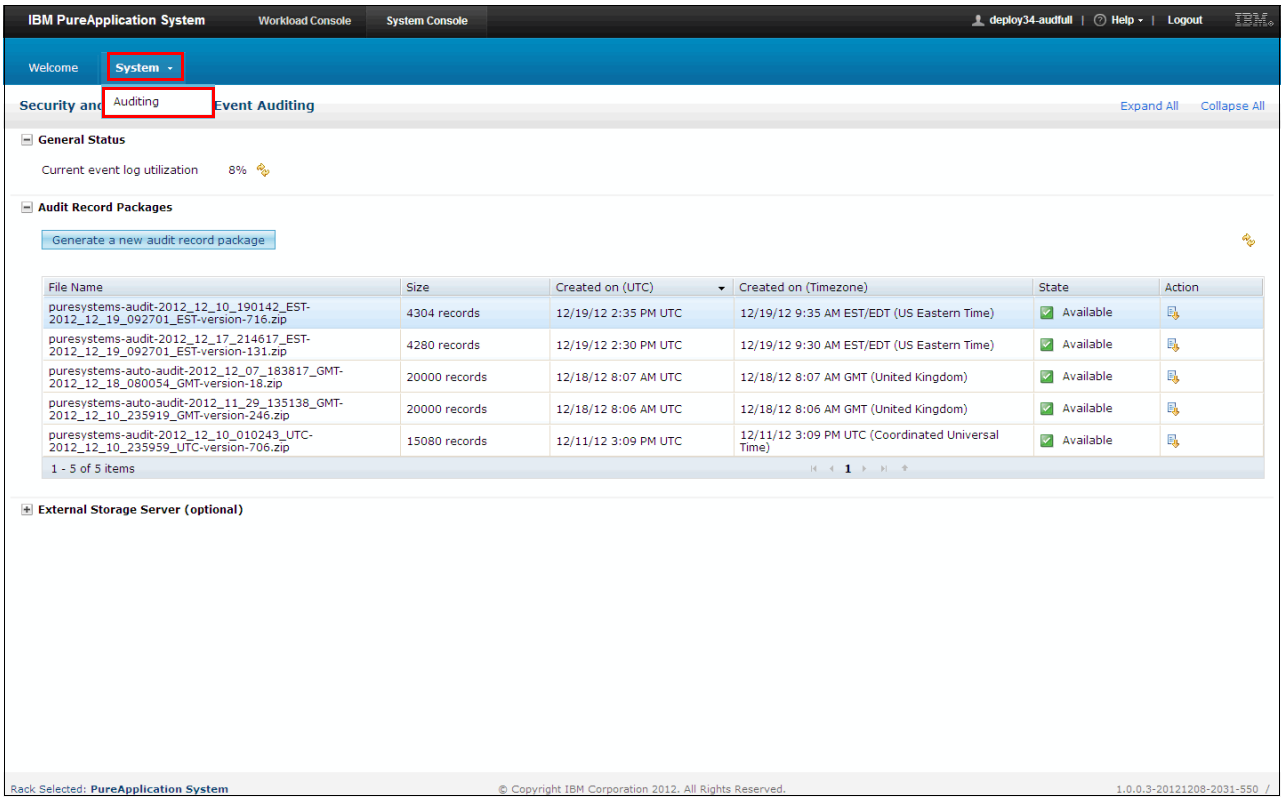


Figure 2-14 Security and Administrative Event auditing

The system provides a function to generate a new audit record package. This function enables filtering of the requested data between start and end time. Figure 2-15 shows the audit data generation panel.

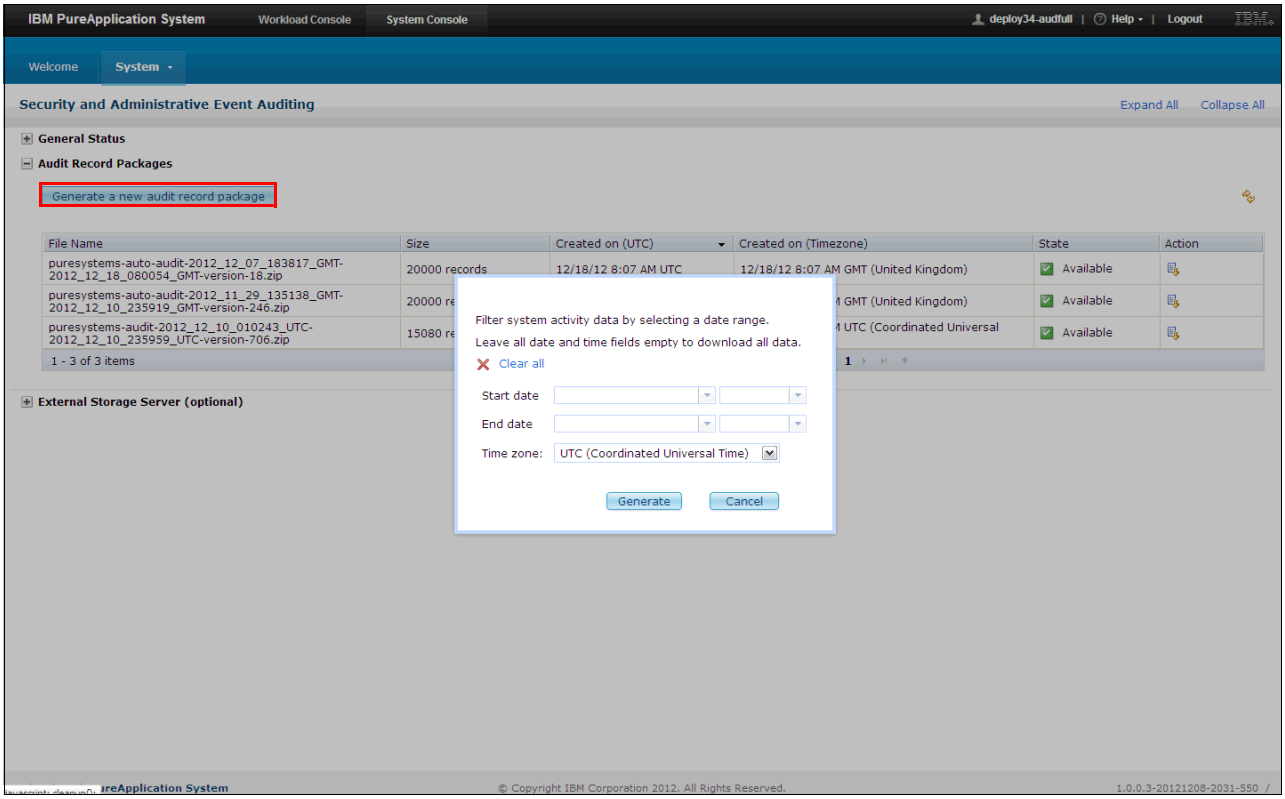


Figure 2-15 Generating an audit data package

The generated data can be download from the Audit record packages section. Figure 2-16 shows example content of the generated file.

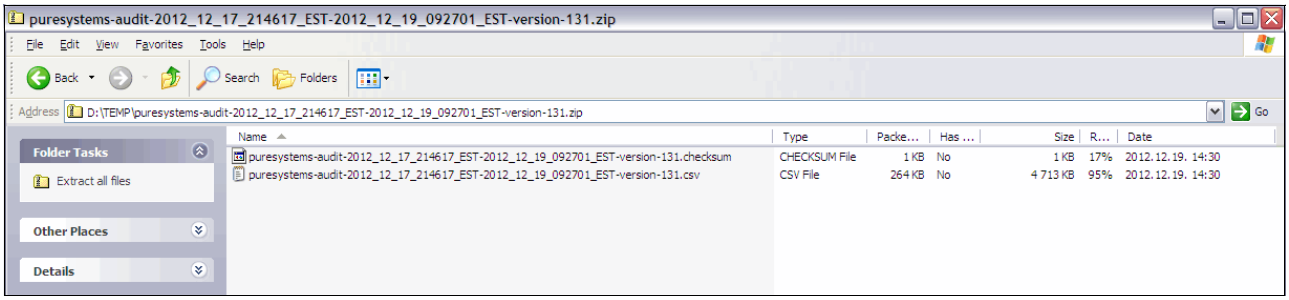


Figure 2-16 Generate audit log file content

The downloaded audit package is a `.zip` file that contains two files. The checksum file contains information to check the audit file integrity. The audit records are in a text file with a `.csv` extension. This file contains a header row line and the audit records. Every line contains comma-separated values. All audit records contain the following columns:

- Timestamp
- ComponentType
- ActionId
- ResourceID
- UserID
- IP
- AdditionalData

2.8 Monitoring

PureApplication System provides comprehensive features to monitor system activities. Integrated components provide a single place to track all events and check the status of the system. The system provides some standard interfaces that help to integrate your system into an existing monitoring infrastructure.

2.8.1 Integrated solutions

The monitoring infrastructure of the PureApplication System is designed to facilitate locating the necessary information for users and operators. From the security side, every user has their role in the system. Deployers can see only their middleware or database metric. Monitor operators can access all user deployments and the monitor administrator can access hardware metrics.

The dashboard provides drill-down navigation on the elements of workloads. This ability helps to focus attention on required elements. If more details are required, the user can dig deeper. This interface also provides a log viewer feature to help experts solve workload issues.

Figure 2-17, Figure 2-18 on page 46, and Figure 2-19 on page 47 show aspects of drill-down navigation between the components of a virtual application.

Figure 2-17 shows basic monitoring information. To open the details about this page, click the first VM.

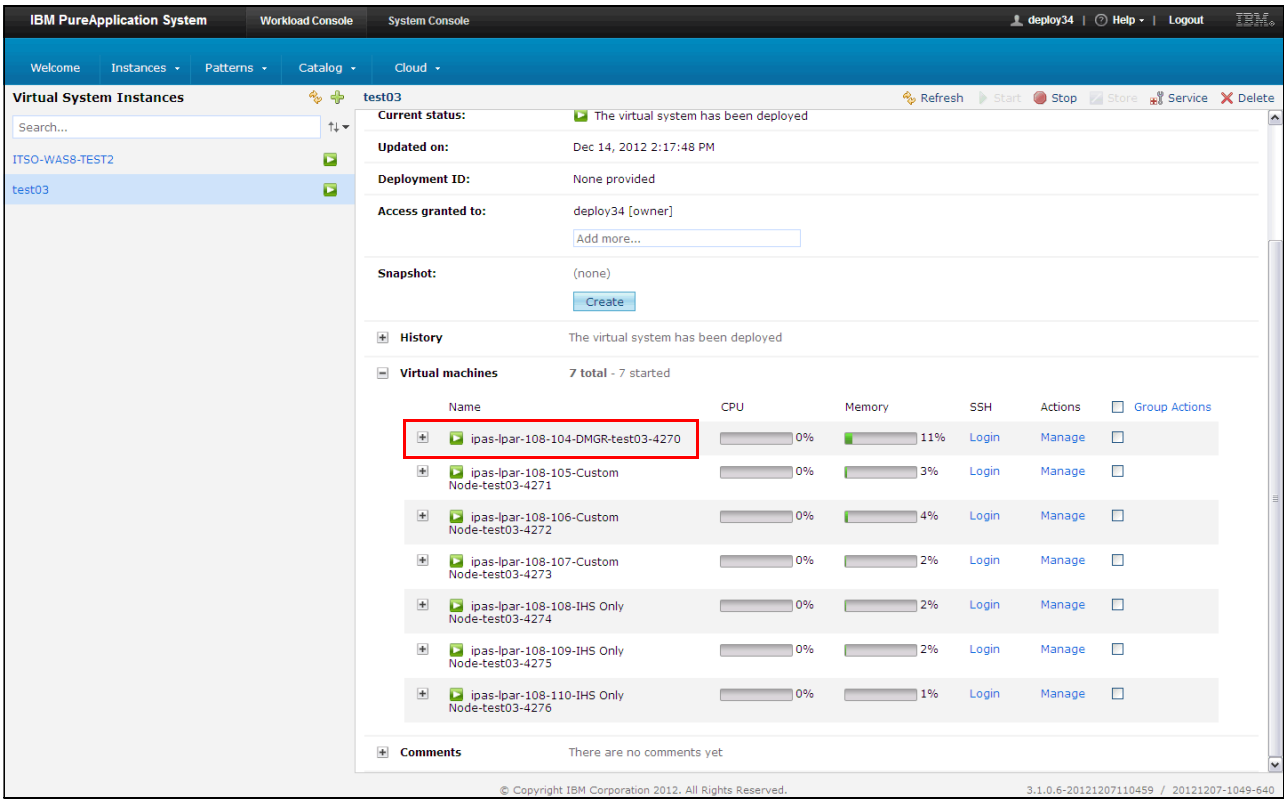


Figure 2-17 Deployed virtual system basic monitoring information

Figure 2-18 shows more information and management options. Click **Manage action**, then click **More** to drill down for more information.

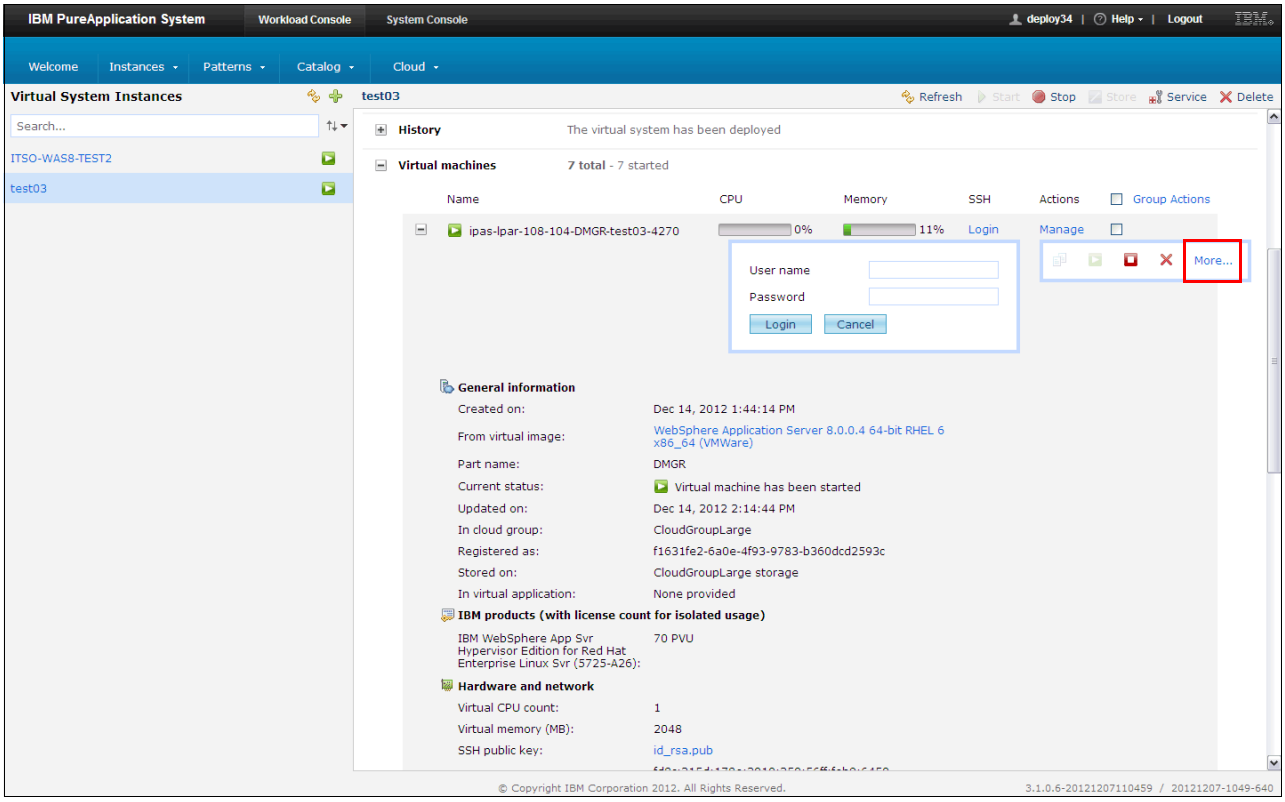


Figure 2-18 More information and management options

Figure 2-19 shows the available virtual machine monitoring.

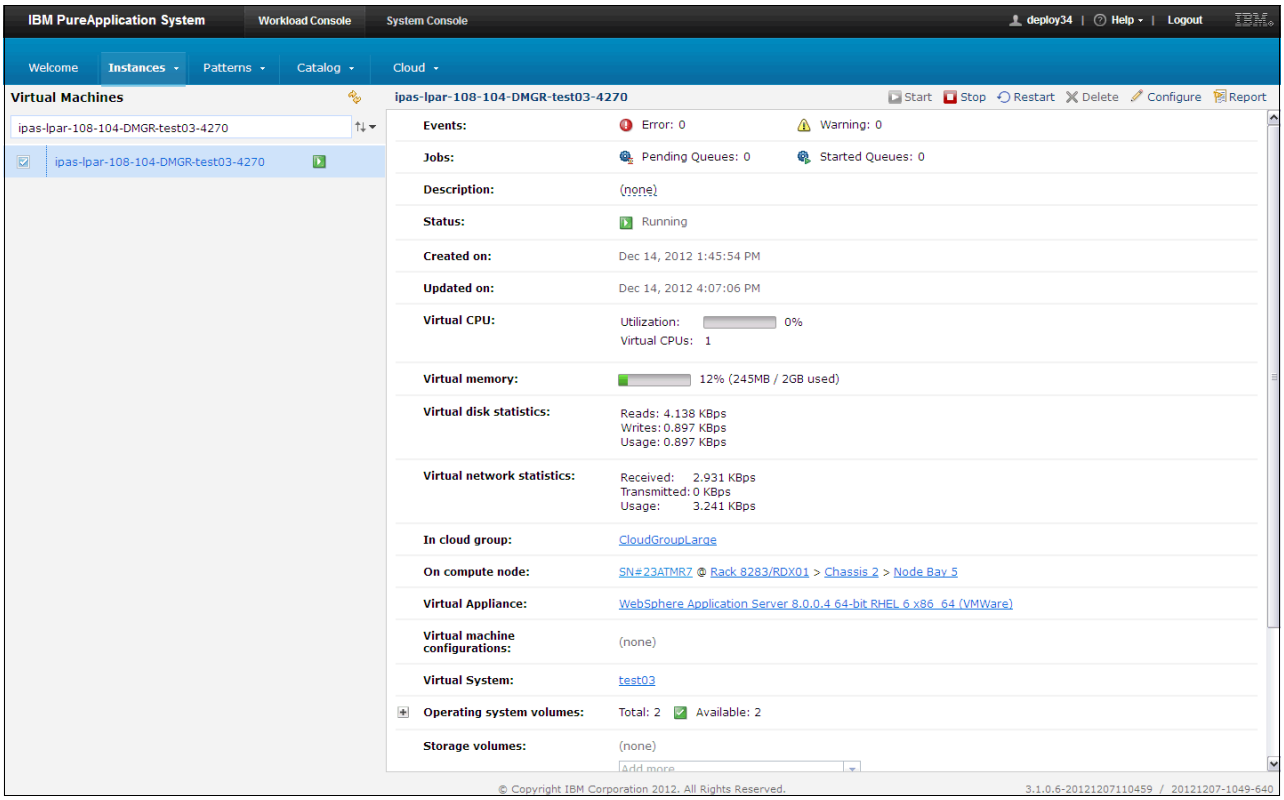


Figure 2-19 Next level and virtual machine monitoring information

The system provides more detailed monitoring information with the PureApplication System Monitoring portal. To access the portal client, complete the following steps:

1. Choose a virtual application instance.
2. Click **Manage**.
3. On the Monitoring tab, click **Advanced monitoring**, which starts the PureApplication System Monitoring Portal Client.

2.8.2 PureApplication System Monitoring Portal

The PureApplication System Monitoring Portal offers a dashboard view of your critical enterprise IT resources.

Requirements

The portal user interface is a Java-based GUI, which requires IBM Java 5 or later to be installed.

Overview

The following list describes the important elements for the monitoring portal:

- PureApplication System Monitoring Server uses installed agents to collect monitoring data from the monitored systems.
- PureApplication System Monitoring Portal Server connects to the PureApplication System Monitoring Server to retrieve collected events and performance data. The server also provides a collection of software services for the PureApplication System Monitoring Portal client.
- The PureApplication System Monitoring Portal client provides a Java based dashboard for analysis of the monitoring data.

Figure 2-20 shows sample content from the PureApplication System Monitoring Portal client. The navigator view provides a hierarchical view of your enterprise and helps to browse specific resources.

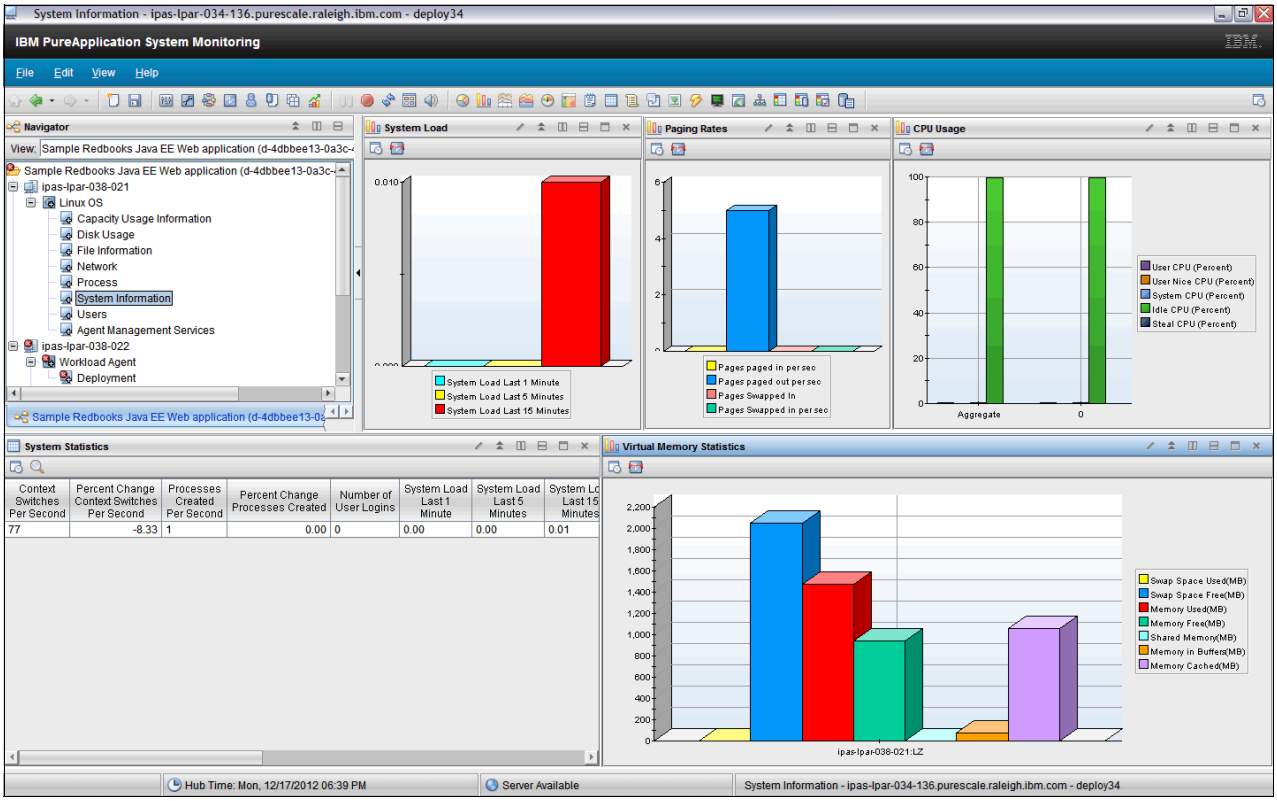


Figure 2-20 PureApplication System Monitoring Portal client

2.8.3 Monitoring layers

The entire monitoring infrastructure in PureApplication System can be divided into layers from the hardware level to the system level through the middleware. The following list describes the layers in greater detail:

- Hardware monitoring

  At the hardware level, the system provides an infrastructure map that contains all of the important monitoring information in one place. It shows an overview of the compute nodes, storage system, network switch, and LED status.

- Middleware monitoring

  You can monitor the workloads and instances with PureApplication System. For the monitoring facilities to be displayed in the workload console, you must deploy monitoring shared services. A subset of middleware monitoring functions is required to use the installed agents. Information for monitoring also is provided by the integrated parts of the system.

- System monitoring

  The system monitoring service provides the monitoring infrastructure that collects performance and availability information. Use the system monitoring service to troubleshoot your system and make business decisions about your hardware and service usage.

Layered monitoring is critical to a healthy system and allows IT teams to be proactive in preventive maintenance. Figure 2-21 shows the graphical infrastructure map of the rack that is used for monitoring.

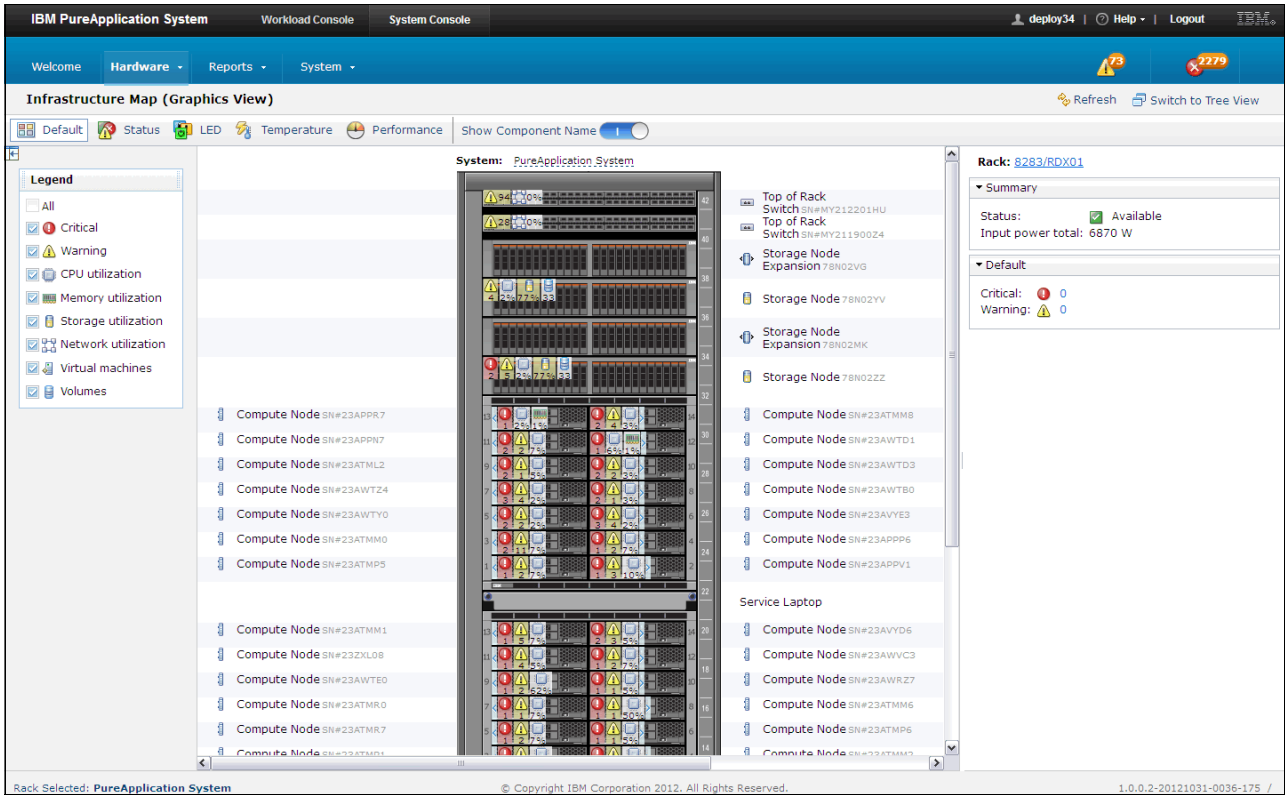


Figure 2-21 Hardware infrastructure map

2.8.4 External monitoring

Many external enterprise monitoring solutions can be used with PureApplication Server. For example, an IBM Tivoli/Omnibus external server can be configured as a trap destination for SNMP events from PureApplication System. These events include selected hardware and software events. A single PureApplication Server MIB is provided for external enterprise monitoring solutions and a set of OMNIbus rules are provided for IBM Tivoli Netcool/OMNIbus. The system provides a download link for MIB and OMNIbus rules from the GUI. Files are available by browsing to **System Console** → **System** → **Event forwarding**. Figure 2-22 shows the download links for the necessary rules.

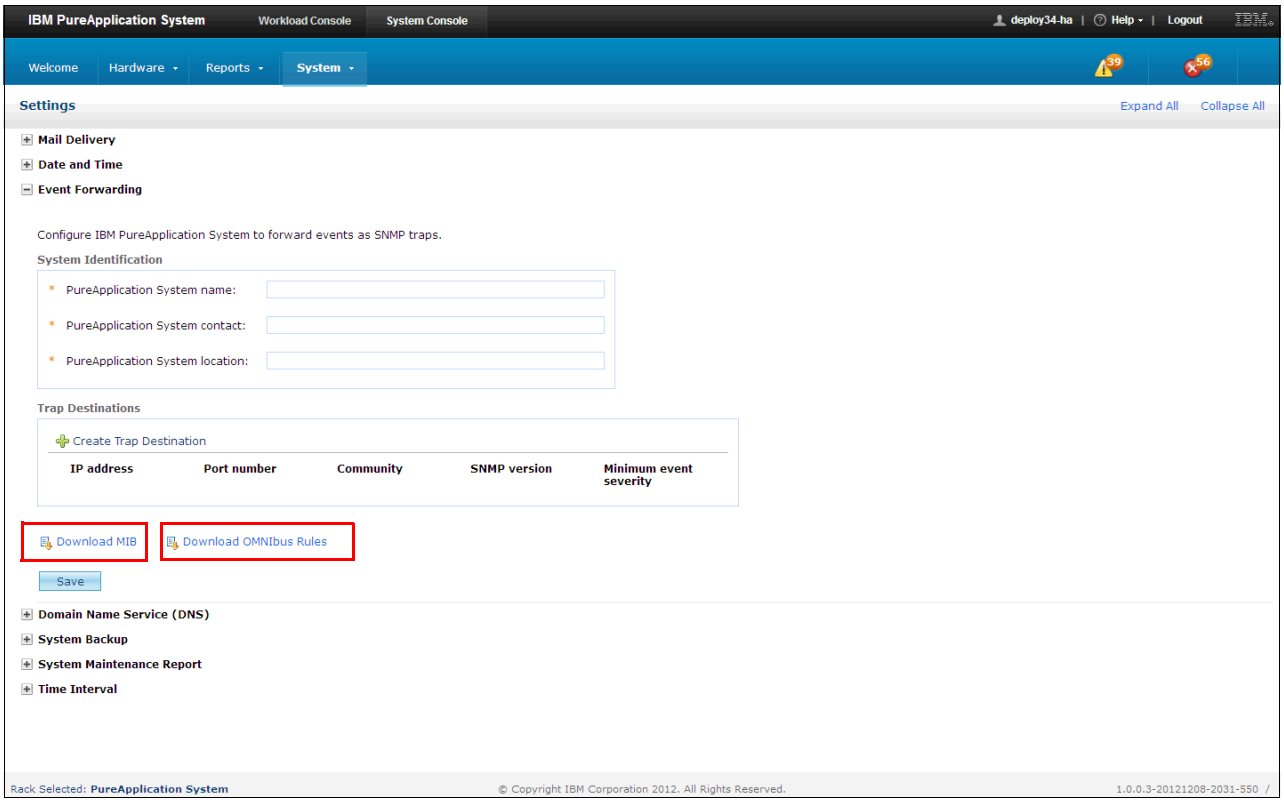


Figure 2-22 Download links for MIB and OMNIbus rules from the GUI

2.9 Integration with external backup systems

Tivoli Storage Manager backup capabilities are integrated into the PureApplication System user interface. You can create an online backup of the database image, schedule an automatic backup, and list the existing database image backups.

This functionality is not limited to databases that are defined in a database pattern. It also applies to databases defined in virtual applications. Backup functionality requires that the Tivoli Storage Manager system plug-in is configured. Tivoli Storage Manager is not included in PureApplication System.

If Tivoli Storage Manager is configured, the default schedule for backups is daily. If Tivoli Storage Manager is not configured, scheduled backups are set to OFF until you select a frequency. After the backups are scheduled, they run automatically at 23:00.

The restore of any of these backups must be done by using Tivoli Storage Manager. You also can create a manual backup. Manual backups are recommended if you plan to use the backup as a clone in a new database instance.

2.9.1 Tivoli Storage Manager System plug-in configuration

The Tivoli Storage Manager plug-in is available by clicking **Workload Console** → **Cloud** → **System plug-ins**, then selecting the plug-in from the IBM Database Pattern category. Figure 2-23 shows the configuration options for Tivoli Storage Manager integration. This form contains the connection details to access a Tivoli Storage Manager server.

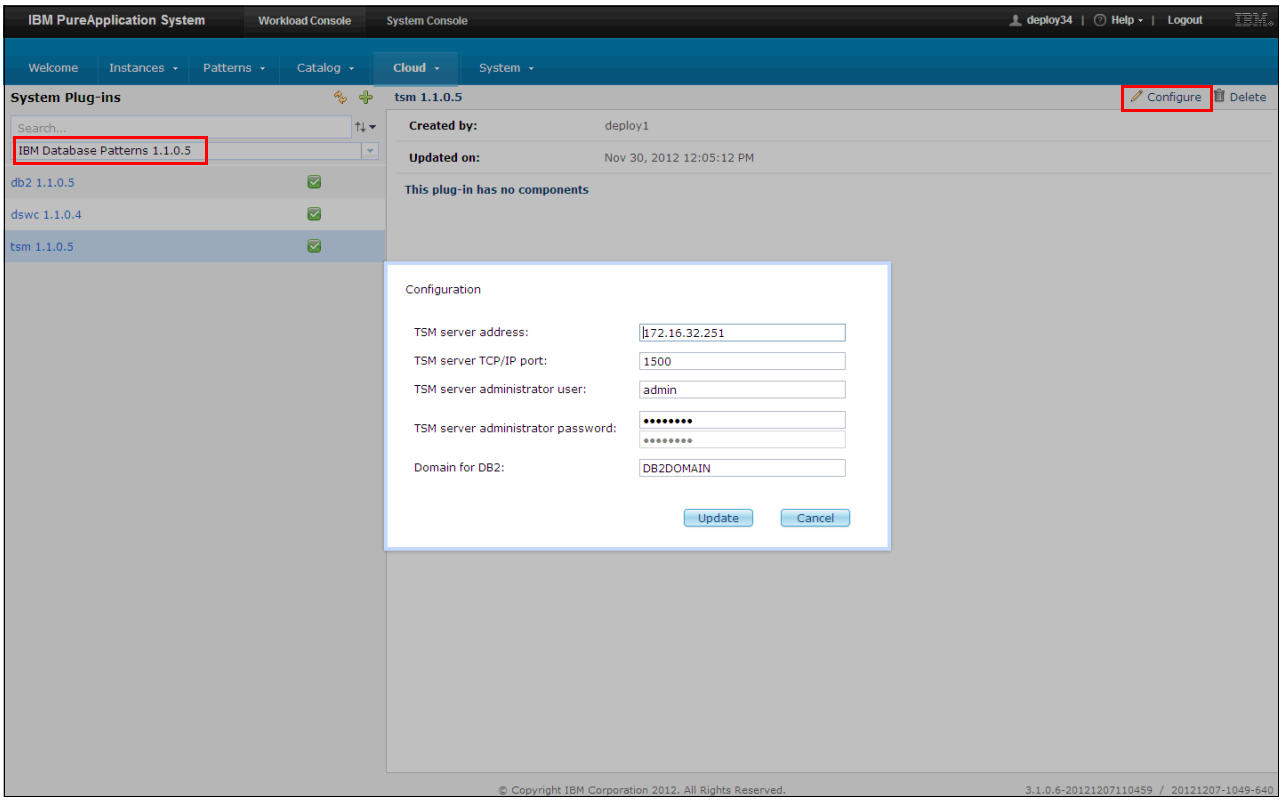


Figure 2-23 Tivoli Storage Manager plug-in configuration

2.9.2 Configuring a scheduled Tivoli Storage Manager database backup

Every instance of a database is configurable to use Tivoli Storage Manager for a scheduled backup. By default, the database instances are scheduled for a daily backup. Figure 2-24 shows the settings section and the options for automatic backup.

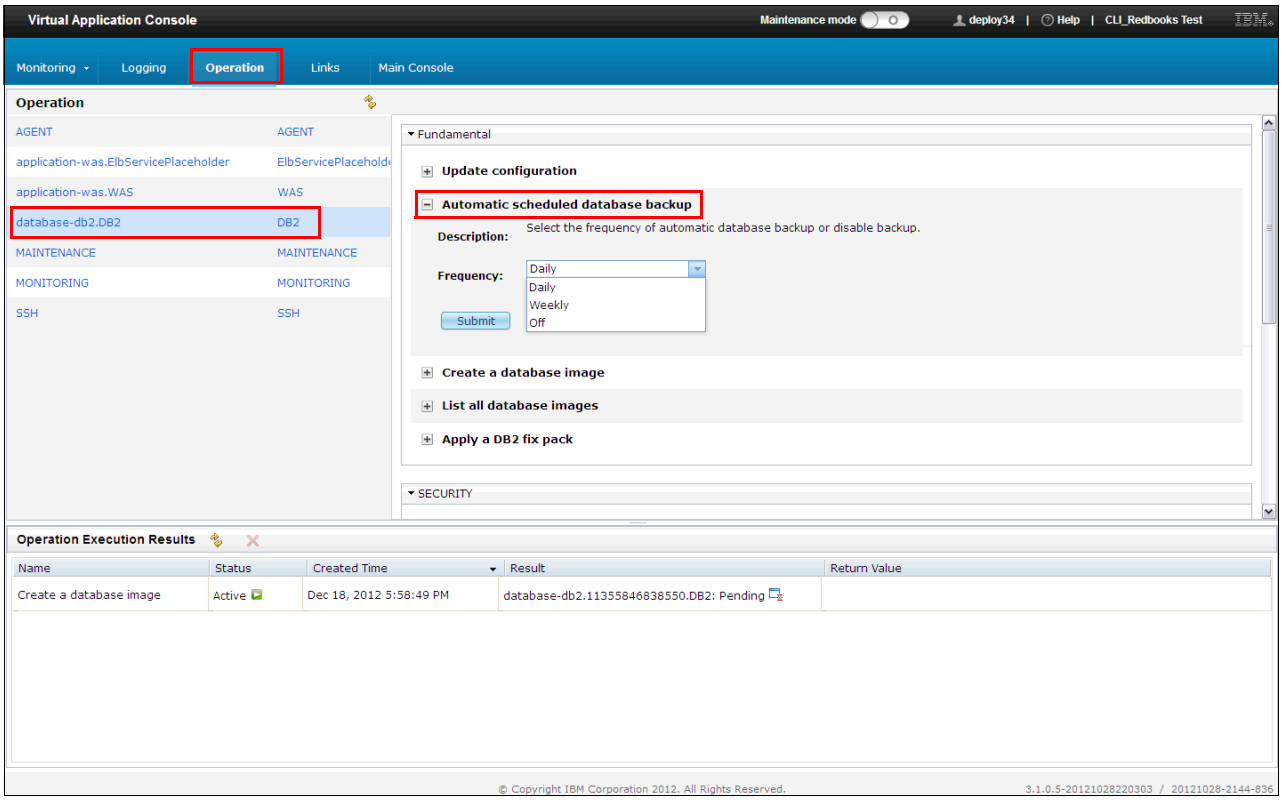


Figure 2-24 Virtual application database backup settings

This option also is available for stand-alone database instances. Figure 2-25 shows the configuration section on database console.

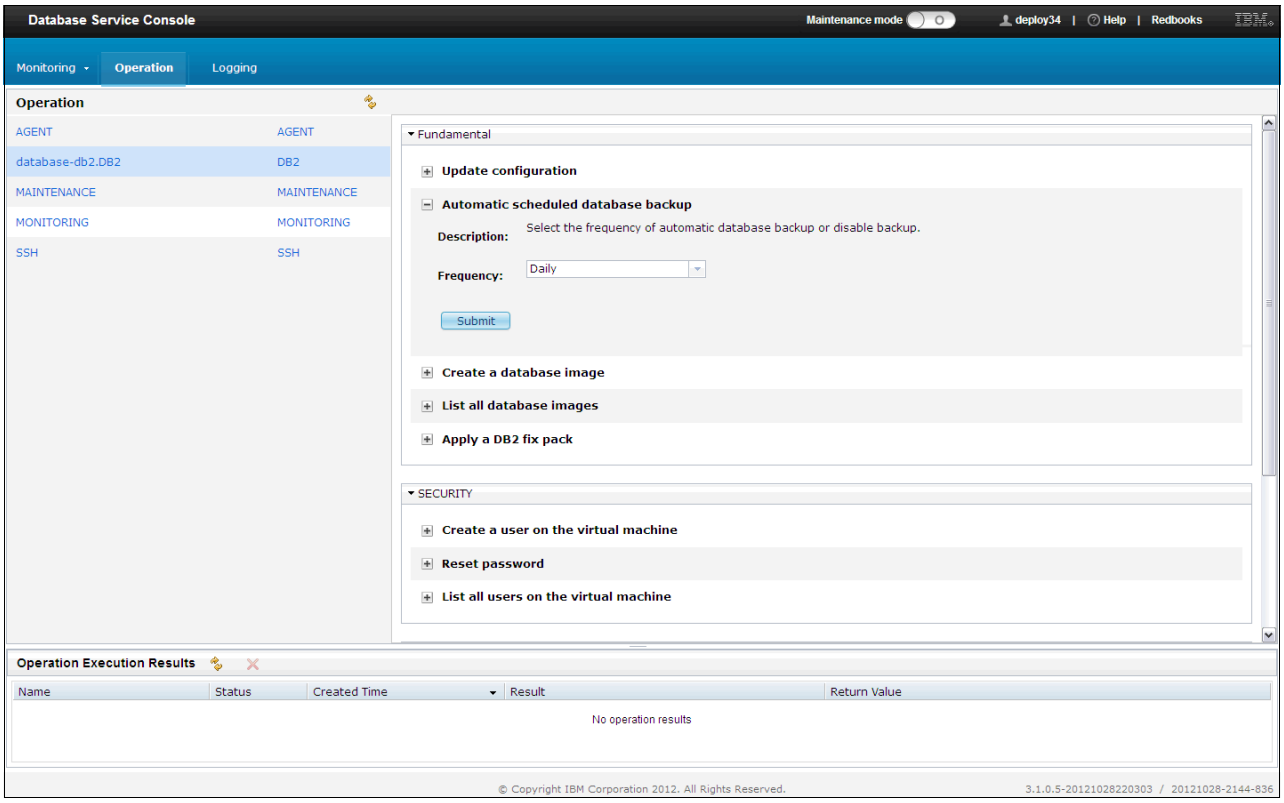


Figure 2-25 Database instance automatic backup setting

2.9.3 Creating a manual backup

The PureApplication System also provides a manual backup feature. This option requires configuring the Tivoli Storage Manager plug-in. This function is available by browsing to **Virtual Application Console** → **Operation**, then selecting the operation that you want to work with and clicking **Create database images**. Figure 2-26 shows from where the manual backup can be started. The interface for the stand-alone database instances is the same as the virtual application's database.

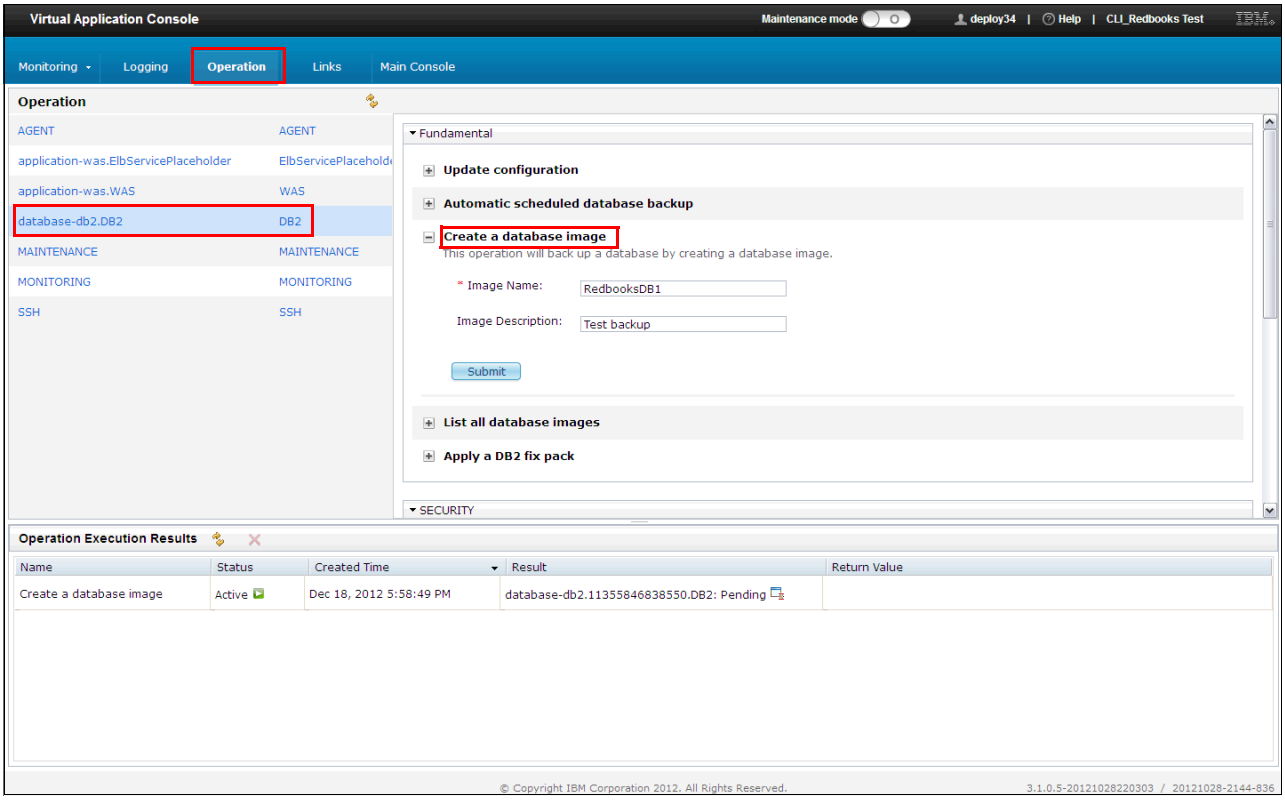


Figure 2-26 Manual database backup configuration

The system also provides a list of the existing database backup images. Figure 2-27 shows an example of this list. This list contains all database backup instances.

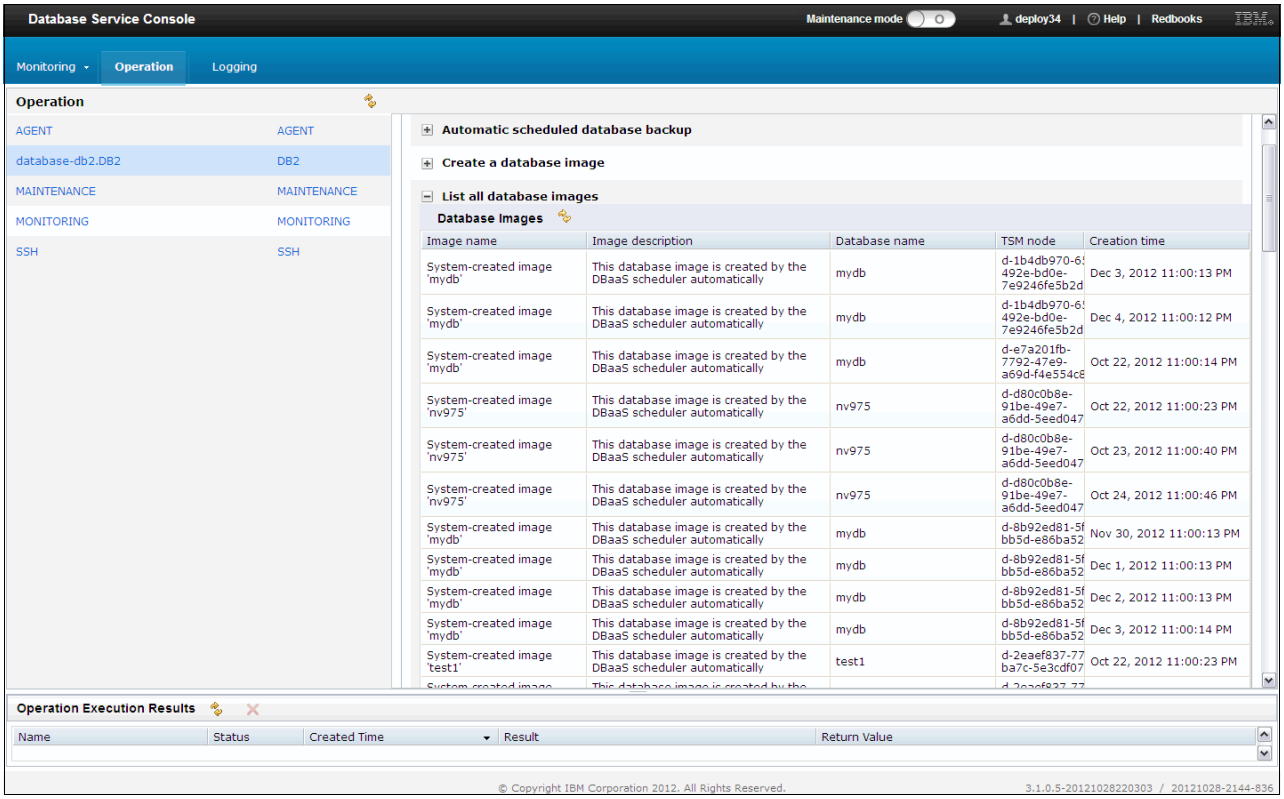


Figure 2-27 Existing database image backups

2.10 Export and import features

Although the development of the patterns is not complicated, the rebuilding of a well-functional system in a different environment can be a long process. Fortunately, PureApplication System also provides export and import functions.

The export and import features make it easier to perform the following tasks:

- Build a tested pattern in another system.
- Ensure matching of systems. In some cases, the exact pattern must be used in two systems and must be exported from one system and imported into another.

The GUI provides import and export options. However, another option is the use of the CLI, which can be more efficient in some cases.

Figure 2-28 shows the import and export options on the GUI.

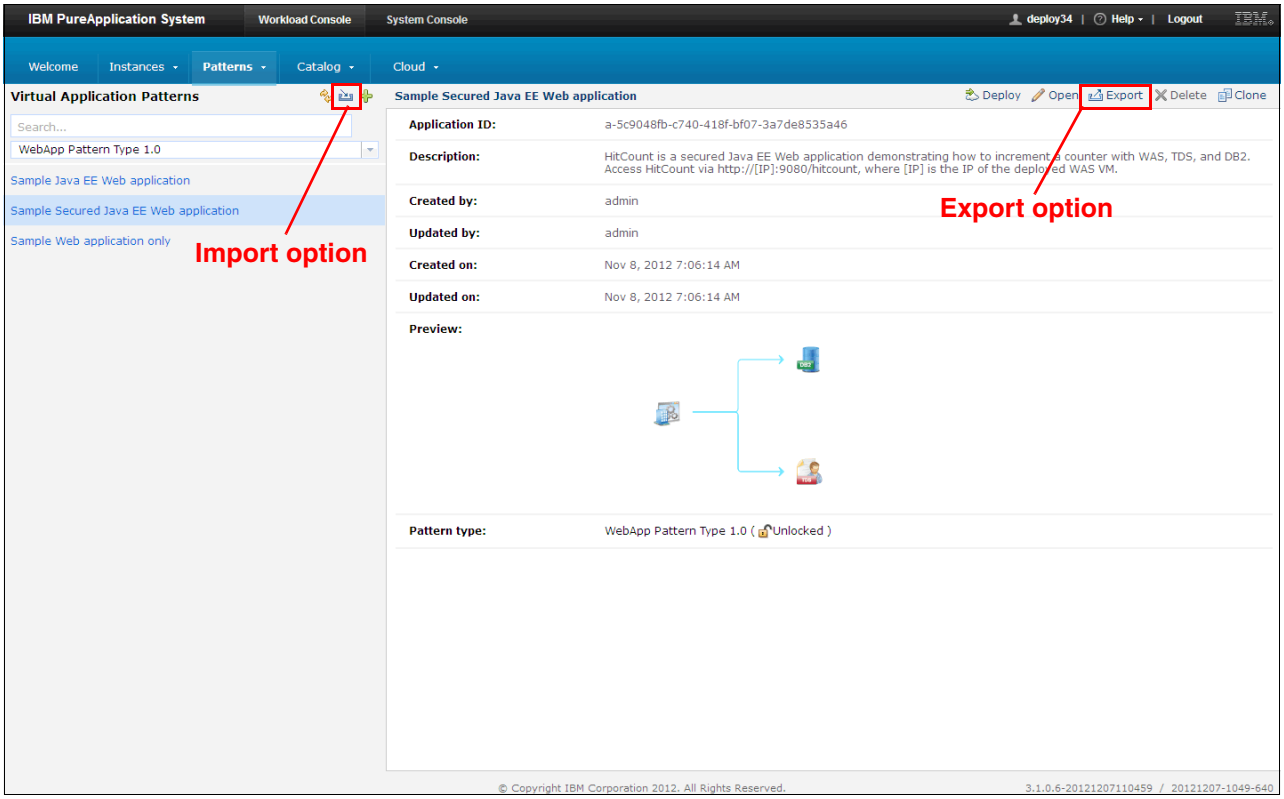


Figure 2-28 Sample virtual application pattern import and export feature

The CLI contains some sample scripts for pattern import and export features. These scripts are available in the `pure.cli/samples` directory in the `pure.cli.zip` file.

When more patterns or data migration is required, a best practice is to develop a CLI script to manage the process.

2.11 External database integration

PureApplication System provides predefined components for building virtual application patterns when an external DB2 database is used. This connection requires a working network link between the virtual application and the external database server. Every configuration option is available by using these predefined components.

Figure 2-29 shows a sample external database connection pattern.

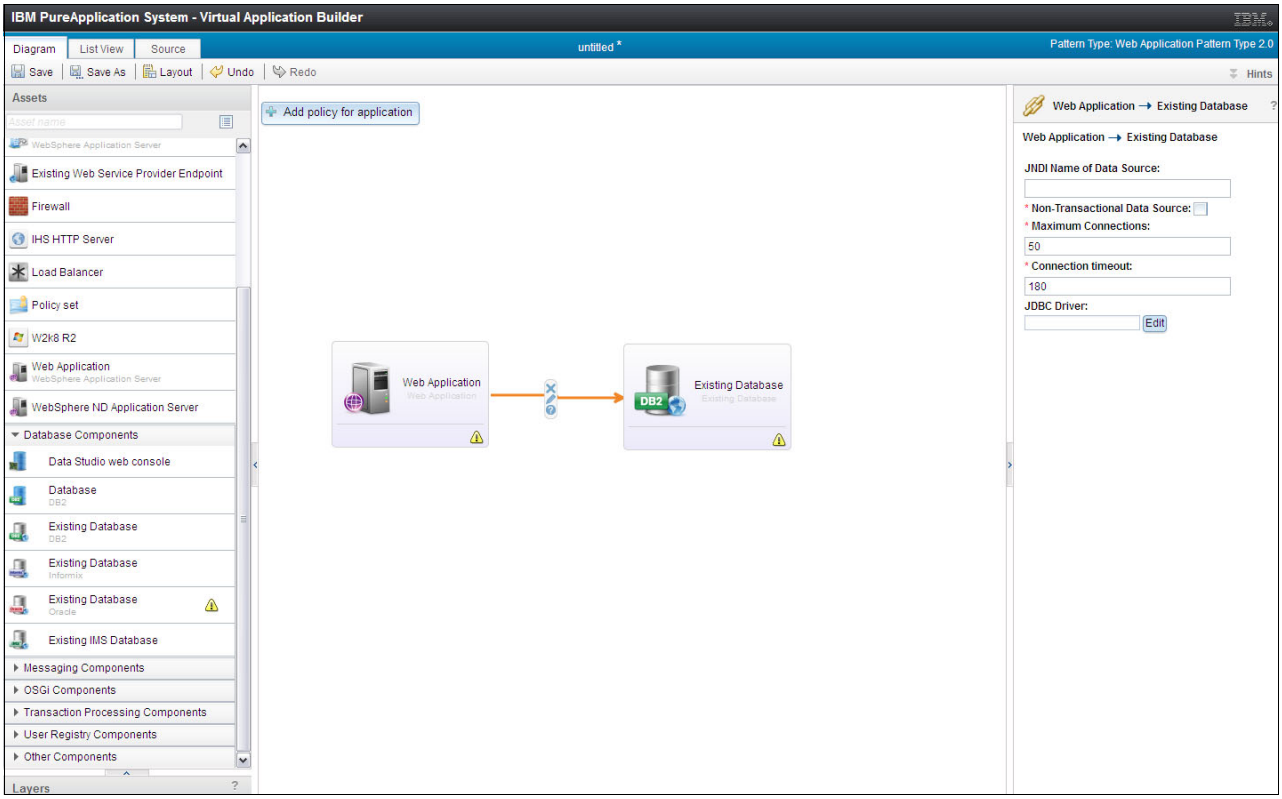


Figure 2-29 Diagram view of external database connection

Figure 2-30 shows the available configuration parameters for an external DB2 connection.

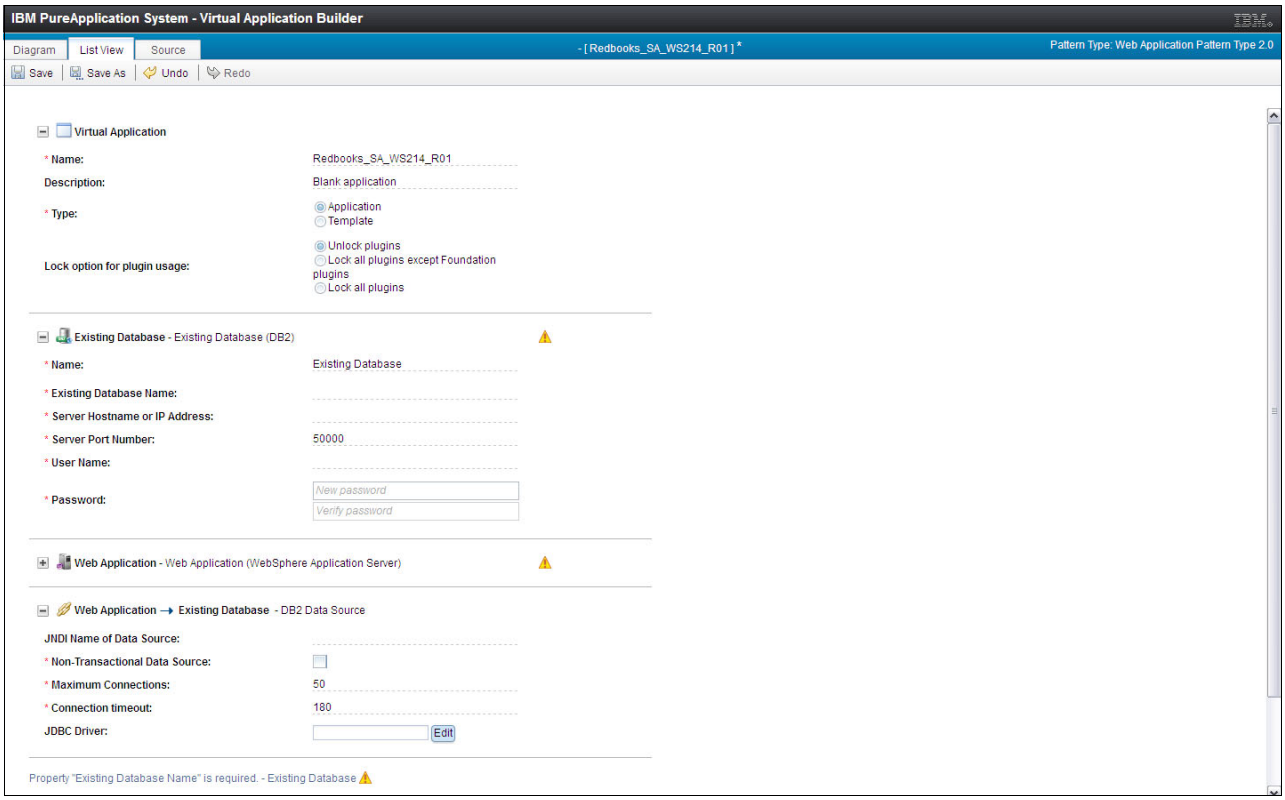


Figure 2-30 List view of external database connection parameters

2.12 Integrating shared network storage to virtual machines

The integration of external storage systems to the PureApplication System is not supported at the hypervisor level. The basic problem with this integration is system integrity. If we use any external device, the system cannot guarantee the integrity and performance of the infrastructure.

Physically, the system connects to the external network with an Ethernet interface. Although this connection excludes the SAN connection option, the system can communicate with other devices by using TCP/IP protocol. This configuration allows the connection to any external network file system. Although there are no options to integrate network resources with the hypervisor infrastructure, the deployed operation systems can connect and use those resources.

The implementation of this connection might be different at the virtual system and the virtual application level.

2.12.1 Network Access Server integration on virtual system level

The easiest way to make a reusable solution is the use of a simple script package that runs at server creation. This script can be added to every virtual server that must access an external file share.

Important: When you are developing scripts, the virtual machines can be restarted. Therefore, the script also needs to modify the operation system `fstab` file and mount the file system.

Figure 2-31 shows a sample script package implementation of a Network Access Server (NAS) connection.

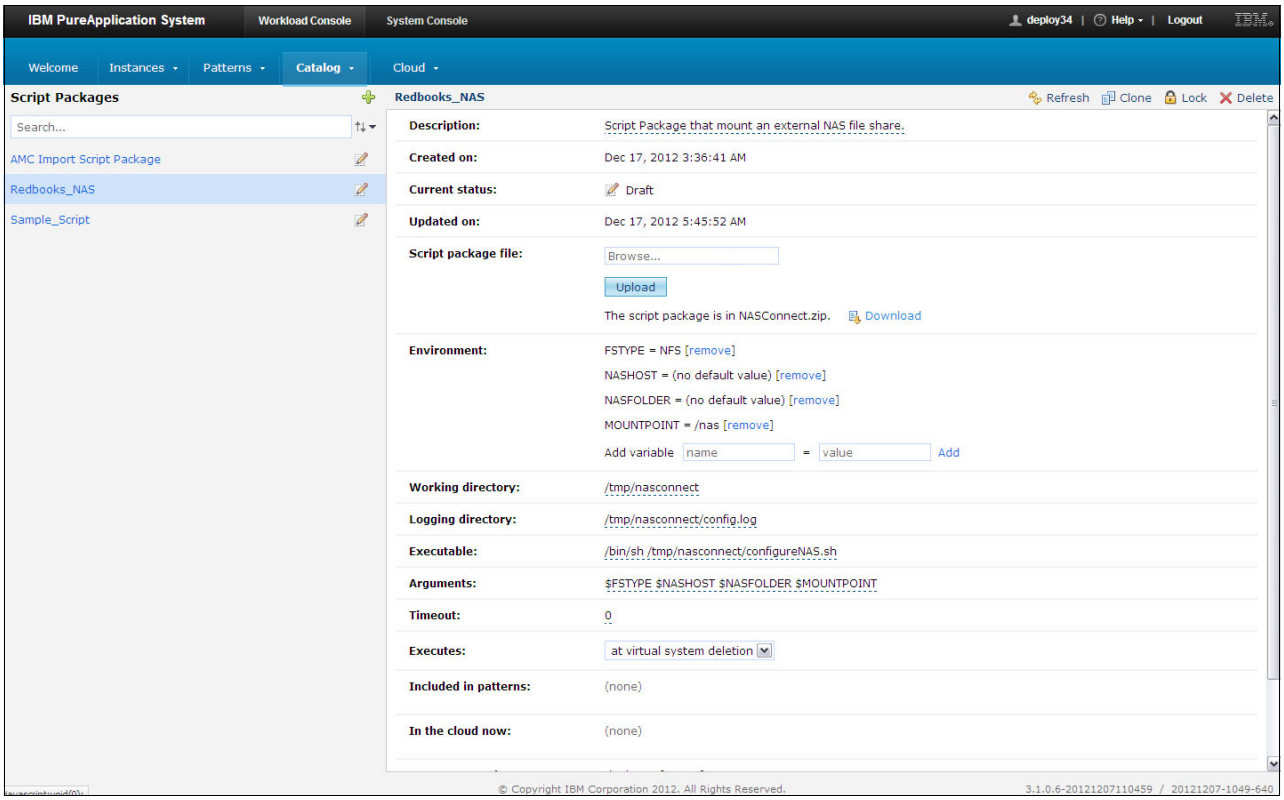


Figure 2-31 Sample NAS connection script package

2.12.2 NAS integration on virtual application level

The basic idea of file share integration in virtual applications is the same as with the virtual server-based solution. The main difference is the tool that is used for development. The most integrated solution for virtual applications is the Plug-in Development Kit (PDK). PDK is designed to help you build your own plug-ins.

2.12.3 iSCSI integration

The new operation system provides capabilities to use iSCSI volumes. Because iSCSI volumes use the TCP/IP protocol, they can be connected to virtual machines. The most important difference between NFS and iSCSI is the type of access. Every iSCSI volume has block level access and requires more caution in a cloud environment. The basic problem is the shared access. Most file systems cannot support the shared access. Considering this constraint, the implementation of an external iSCSI connection might be similar to the external NAS integration.

3

Usage patterns for isolating applications

Isolation is a key feature of cloud computing. Through it, applications that are deployed into the cloud environment can run independently from each other. This configuration separates application traffic through the network and prevents the cross application impact that is caused by resource contention.

IBM PureApplication Systems provide several functions to implement application and virtual systems isolation.

The following topics are covered in this chapter:

- Cloud computing overview
- Managing compute nodes
- Managing IP groups
- Managing cloud groups
- Managing shared services
- Managing environment profiles
- IBM PureApplication System isolation strategies

© Copyright IBM Corp. 2013. All rights reserved.

3.1 Cloud computing overview

An application development lifecycle normally requires separate runtime environments for development, testing, and production. These environments often are isolated from one another so that activities in each environment do not interfere with each other. There can be further requirements within companies to isolate critical applications or applications from different departments and lines of businesses.

IBM PureApplication System supports the deployment of applications (known as pattern instances) into runtime environments. Such runtime environments consist of cloud groups and environment profiles. These cloud groups and environment profiles give IBM PureApplication System the means to create separations or the necessary levels of isolation that are required for applications in different environments or lines of businesses.

To determine the best ways to setup cloud groups and environment profiles, you must first understand their concepts in cloud computing. It also requires an understanding of how they play a key role in hardware resources virtualization and how these features are used in IBM PureApplication System. Finally, you must also understand the best strategies for making the best use of these features.

3.1.1 Cloud computing concepts

Cloud computing provides shared computing resources to application workloads in a controlled way, thus allowing these workloads to successfully run in parallel. There are three types of computing resources: computational (CPU and memory), storage, and networking. Cloud computing enables workloads to share all three.

The following main definitions in cloud computing handle workload and provide resources to applications:

- Resource isolation

  Prevents problems within applications that are deployed into the cloud environment from impacting other applications in the cloud with issues of resource contention. This prevention is done by creating virtual barriers between cloud resources, which allows them to operate independently. Therefore, problems in one isolated application that is deployed into the cloud does not affect other isolated systems.

  The following main isolation aspects are available:

  - Computational isolation

    Groups of CPU and memory capacity are separated from each other. This type of isolation can be physical or virtual and include the following aspects:

    - Physical computational isolation provides dedicated hardware resources.
    - Virtual computational isolation, or virtualized resources, creates groups of separate resources that might actually share hardware.

  - Network isolation

    Communication flows between computational resources by separate connections. The isolation can be physical or logical and include the following aspects:

    - Physical network isolation provides the resources to allow data to travel in parallel sets of network equipment, such as switches.
    - Logical network isolation shares network equipment and bandwidth, but the data is routed separately by a different virtual local area network (VLAN).

- Resource sharing

  Common pools of resources, such as IP addresses, CPU, and memory are made available to systems that are deployed in the cloud. Each workload can acquire a different amount of resources, depending on its requirements. Resource sharing is dynamic; workloads acquire shared resources dynamically. However, it requires care and attention because sharing resources can trigger workloads to use excessive resources from any pool, which starves the other workloads in the cloud environment.

- Resource allocation

  Resource allocation, or logical isolation, enables resource sharing, but requires setting boundaries on a workload by putting lower and upper limits on resource sharing. Allocation balances isolation and sharing, which ensures that a workload gets at least the minimum resources it must run properly. Moreover, it prohibits workloads from using excessive amounts of a pooled resource. You can set allocation limits on any shared resource, such as CPU, memory, storage, bandwidth, and even software licenses.

3.1.2 IBM PureApplication System features for cloud computing

IBM PureApplication System provides the following features that allow the product capacity to provide resource sharing and isolation of workloads:

- Compute nodes

  Compute nodes contain components such as microprocessors, memory, network adapters, and storage adapters.

- IP groups

  IP Groups are logical groupings of one or more IP addresses and networking information, such as DNS, subnet, VLAN.

  IP Groups, and by association its IP addresses, are assigned to cloud groups. Cloud groups can be contained by more than one IP group.

  A virtual machine (VM) that is deployed into the IBM PureApplication System infrastructure is assigned its IP address. The addresses are assigned from the available pool of addresses that are defined in the IP group that is associated with the cloud group that is used by the VM.

- Cloud groups

  A cloud group is a logical grouping of computing resources (compute nodes) to target your deployments in IBM PureApplication System. It requires one or more compute nodes and one or more IP groups. A pattern is deployed to a cloud group (by using an environment profile). The pattern instance is composed of VMs.

  Important: Compute nodes cannot be shared across multiple cloud groups.

  To deploy virtual machines into the IBM PureApplication System environment, you must associate these VMs to cloud groups.

  Virtual machines are deployed on the compute nodes that are part of the chosen cloud group.

- Environment profile

  Environment profiles are policies for deploying patterns into cloud groups. They group related deployment configurations, such as virtual machine names, IP address assignment, and cloud groups. Environment profiles also associate users and user groups with cloud groups. These profiles specify the cloud groups to which each user or user groups can deploy patterns. Finally, Environment profiles create the logical isolation of resources by allocating these resources.

- User group

  A list of users in the same role. By using environment profiles, IBM PureApplication System associates one or more user groups with cloud groups, which give permission to users in the chosen user groups to deploy patterns into specific cloud groups.

Finally, you also can create and attach more storage volumes to deployed VMs.

Figure 3-1 shows the relationship between the multiple cloud resources within IBM PureApplication System.

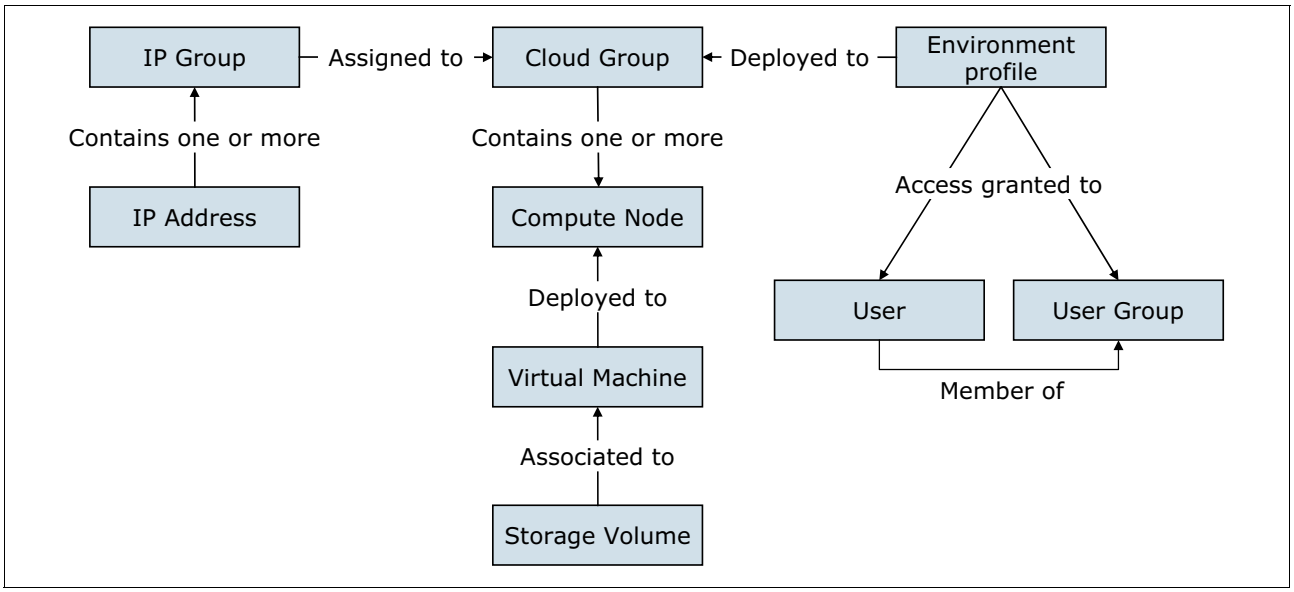


Figure 3-1 Relationship of PureApplication System resources

The focus of the next topics in this chapter is on how to manage and administer the cloud functions that are provided by IBM PureApplication System. This chapter also describes how to use the cloud management functions, and create isolation between environments, lines of businesses, and applications.

3.2 Managing compute nodes

A compute node is a computer that contains microprocessors, memory, Ethernet controllers, and hard disk drives (HDD). As such, each compute node runs exactly one hypervisor.

Each IBM PureApplication System model W1500 compute node consists of the following components:

- Intel CPUs that contain 16 physical (32 logical) cores
- 256 GB of RAM memory

Additionally, the compute nodes have access (as SAN) to two IBM Storwize® V7000 storage units with 6.4-TB solid-state drive (SSD) and 48-TB HDD that comes in the PureApplication System rack. The rack also includes two BLADE Network Technologies 64-port Ethernet switches that are used for internal networking and connectivity to the enterprise's network.

For more information, see 1.3, “IBM PureApplication System overview” on page 14.

When a pattern is deployed into a cloud group, IBM PureApplication System ensures that the required amount of RAM memory and CPU resources are available for each virtual machine in that pattern.

Finally, the following states and actions can be taken against a compute node in IBM PureApplication System:

- Power on

  Power on the compute node.

- Power off

  Power off the compute node.

- Start

  Start the compute node to accept workloads.

- Quiesce

  No new load is given to this compute node.

- Existing Node continues to run

  Locks the workload that is running on this node.

- Maintain:
  - Readies compute node for maintenance.
  - Moves existing VMs to other compute nodes (in the same cloud group) based on priorities. Also warns users if there is not enough capacity on other nodes to move the VM.

- Report

  View the compute node usage report.

In the next sections of this book, we describe how to run some of the possible actions on the compute node.

3.2.1 Starting compute nodes

To start a compute node, you must be assigned the hardware administration role with permission to manage hardware resources (full permission). Complete the following steps to use the system console pane to start a compute node:

1. Access the console pane.
2. Click **System Console** → **Hardware** → **Compute Nodes**.
3. Select the compute node that you want to start.
4. Click the **Start** icon on the toolbar.

3.2.2 Suspending compute nodes

To suspend a compute node, you must be assigned the hardware administration role with permission to manage hardware resources (full permission). Use the system console pane to put a compute node into quiesce state. When set to this state, the compute node is suspended and cannot accept system deployments.

Complete the following steps to put a compute node into quiesce state:

1. Access the console pane.
2. Click **System Console** → **Hardware** → **Compute Nodes**.
3. Select the compute node that you want to suspend.
4. Click the **Quiesce** icon on the toolbar.

3.2.3 Putting compute nodes in maintenance mode

To place a compute node in maintenance mode, you must be assigned the hardware administration role with permission to manage hardware resources (full permission). Use the system console pane to put a compute node into maintenance mode. However, before a compute node is put into maintenance mode, you must first suspend the compute node by putting it into a quiesce state.

Complete the following steps to put a compute node into maintenance mode:

1. Access the console pane.
2. Click **System Console** → **Hardware** → **Compute Nodes**.
3. Select the compute node and click the **Maintain** icon on the toolbar.

3.2.4 Viewing compute node reports

To view compute node reports, you must be assigned the hardware administration role with permission to view all hardware resources (read-only). Through the IBM PureApplication System console, you can access, view, and print reports for each compute node.

Complete the following steps to view compute node reports:

1. Access the console pane.
2. Click **System Console** → **Hardware** → **Compute Nodes**.
3. Select the compute node for which you want to run a report.
4. Click the **Report** icon on the toolbar.

Figure 3-2 shows how to access a compute node report.

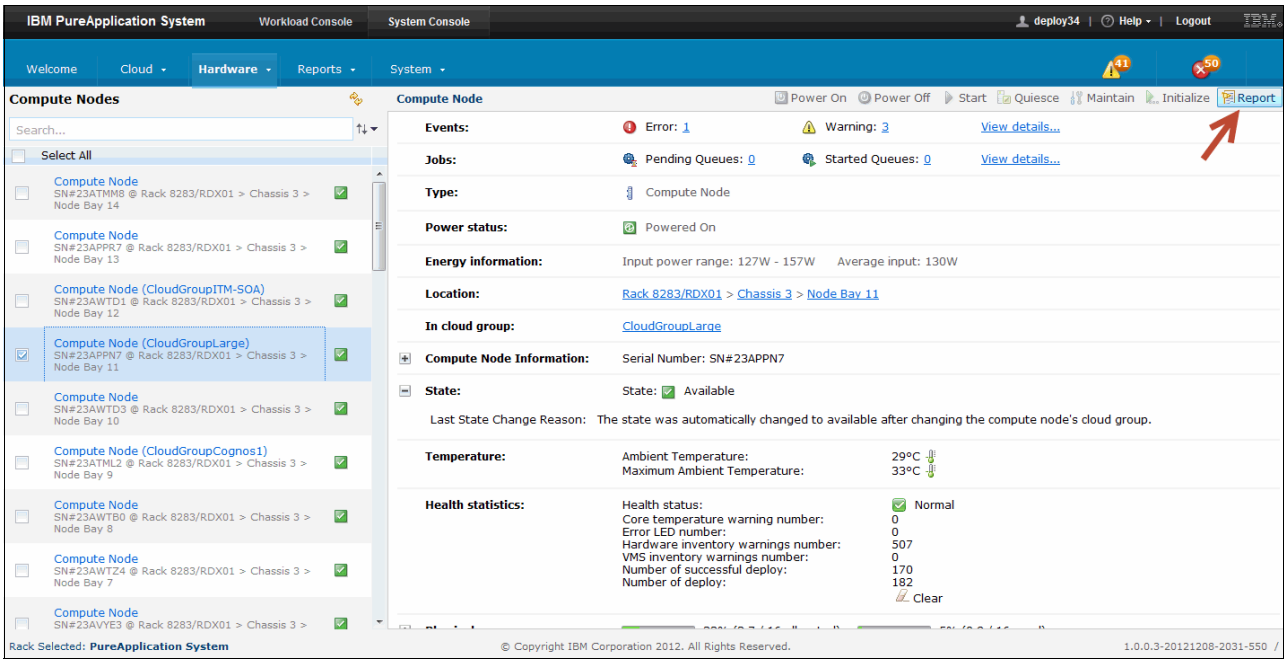


Figure 3-2 Accessing a compute node report

You also can access this information by browsing to the reports. Click **Reports** → **Machine Activity** and select **Allocation by Compute Node**.

Figure 3-3 shows an example of a compute node report.

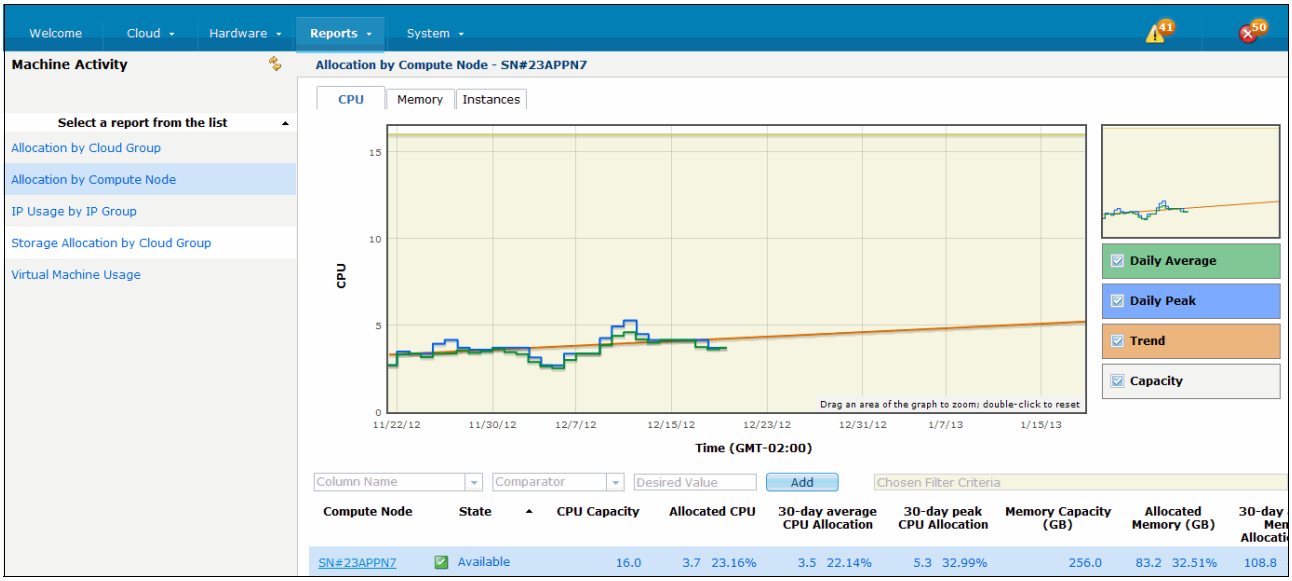


Figure 3-3 Compute node report view

For more information about compute node report options, see the IBM PureApplication Systems Information Center at this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/systemconsole/t_compnodereports.html

3.3 Managing IP groups

When IBM PureApplication System needs an IP address, it obtains the address from an IP group. An IP group is a collection of IP addresses that it assigns to virtual machines as it deploys them. An IP address can belong to only one group. It also has settings for the ID of the VLAN they belong to, and settings for how to connect to the external network the VLAN is part of.

IP groups perform the following functions:

- Dynamic IP addresses sharing

  An IP group forms a shared pool of IP addresses that are used when pattern instances are deployed.

- IP address pool isolation

  IP groups enable system administrators to partition subnets. Therefore, blocks of IP addresses from a subnet can be assigned to a hypervisor, department, or other entity. This configuration helps ensure that each team, environment, or line of business has independent IP pools; therefore, they do not affect other areas.

- Logical network isolation

  The VLAN ID that is specified for each IP group enables the group's pattern instances to communicate on isolated logical networks. This ability is helpful to isolate the network traffic of unrelated applications, such as development and production.

To create an IP group, you must provide the following information:

- A network address, a netmask, gateway, and DNS information.
- A pool of IP addresses within the subnet that are available to the PureApplication System.
- The ID of the VLAN that is used to connect to the external network.

By default, PureApplication System uses all available IP addresses from the IP group. When a pattern instance is deployed, a VM is deployed for each part and each VM needs a unique address. When the instance is deleted, so is each of its VMs, and each address is added back to the pool. So, to determine how many IP addresses you need, you must consider how many parts there are in your patterns and how many instances you want to deploy.

Important: If you do not enter any IP addresses in an IP group, you cannot deploy patterns to hypervisors by using that IP group. Messages alert you to possible errors.

3.3.1 Adding IP groups

IP groups cannot be randomly created. To create an IP group and apply valid network configurations in it, you must obtain these network settings from a network administrator. The administrator must make sure that the settings are valid for the current network configuration.

To create IP groups in IBM PureApplication System, the cloud group administration role is required. Access the administration console and complete the following steps:

1. Access the console pane. Click **System Console** → **Cloud** → **IP groups**, as shown in Figure 3-4.

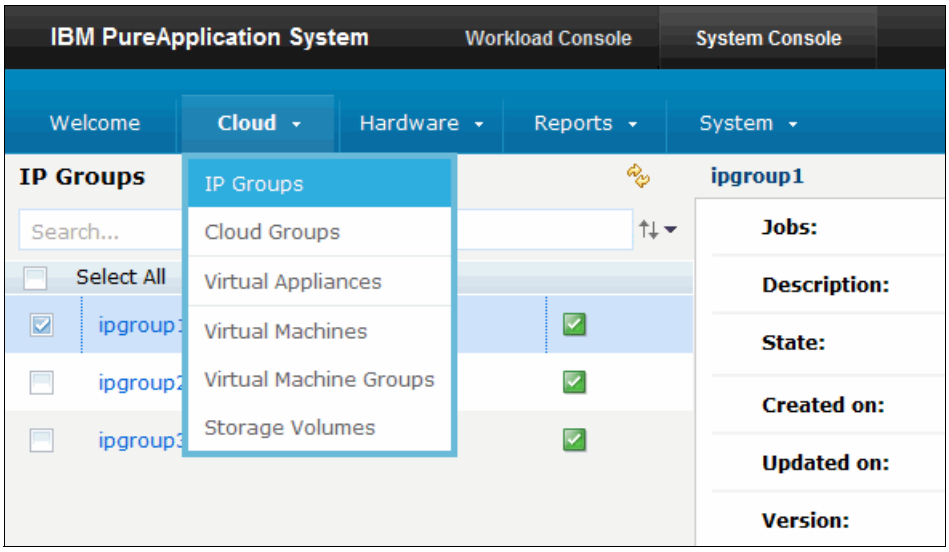


Figure 3-4 Accessing IP Groups configuration

2. Click the **New** icon in the toolbar.
3. Complete the following fields in the window to describe the IP group that you want to add:
   - Name

     Enter a unique IP group name to represent and identify the IP group.
   - Version

     Select `IPv4` or `IPv6` from the list to specify the version.
   - Network address

     Enter a valid network address. This address is associated with the IP group.

     Important: Workloads that require IP caching must be deployed to cloud groups with IPv4 IP groups only.

   - Netmask

     Enter a value for the netmask. This network mask is associated with the network address of the IP group.

     Important: This field applies only to IPv4.

– Gateway

  Enter a gateway name. This default gateway is associated with the IP group.

– Primary DNS

  Provide the primary Domain Name System (DNS) value for the IP group. This DNS server is used for the IP group.

– Secondary DNS

  You can add an optional secondary DNS value for the IP group.

– VLAN

  Specifies the virtual local area network.

– In cloud group

  Specifies the cloud group for which you want to add this IP group.

  The system includes three default cloud groups. You can select one of these cloud groups, or you can create your own. For more information about creating a cloud group, see 3.4.1, “Adding cloud groups” on page 75.

– Click **OK**.

Figure 3-5 shows the initial panel for creating an IP group.

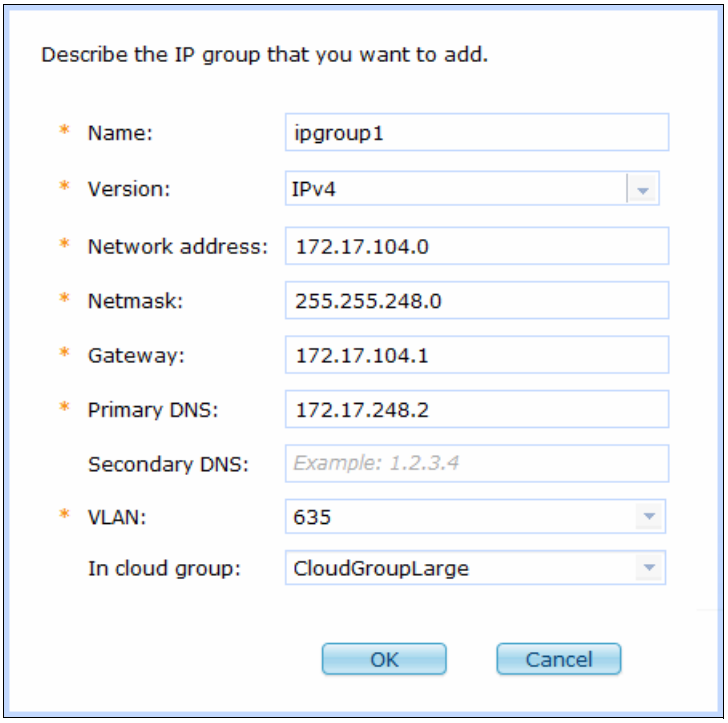


Figure 3-5 Creating an IP group

After the IP group is created, complete the following configurations:

1. In the IP addresses section, complete the following steps to add the range of IP addresses:
   a. Select **IP Range** in the Add by menu.

      This selection determines how your IP addresses are listed when they are added.

   b. Enter the starting IP address in the start ip field.
   c. Enter the ending IP address in the end ip field.
   d. Use the two entry fields to specify the first and last IP addresses in the range of IP addresses to include in the IP group.
   e. Click **Add**.

   Figure 3-6 shows the configuration of a created IP group.

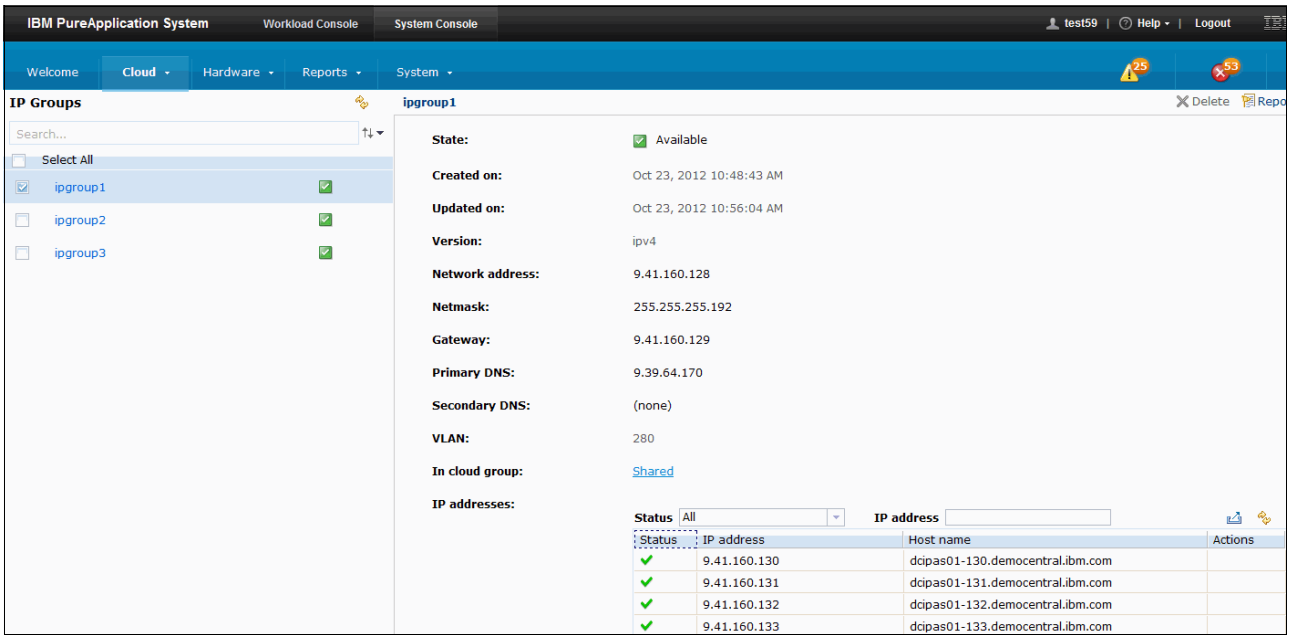


Figure 3-6 IP group configuration

3.3.2 Administering IP groups

IP addresses are only accessible to the system when they are included in IP groups. Edit an IP group to add the pools of IP addresses within the IP group that are available to the deployment process. These addresses are assigned to virtual machines as they are deployed.

To administer IP groups and IP addresses, the cloud group administration role is required. Use the IBM PureApplication System console to complete the following steps:

1. Access the console pane.
2. Click **System Console** → **Cloud** → **IP Groups**.
3. Select the IP group that you want to edit.
4. Click the **value that is adjacent** to each entry.

   Settings that can be changed display either edit or remove links, or are underlined with a dashed line.

3.3.3 Viewing IP group reports

To view IP Group reports, you must be assigned the cloud group administration role with permission to View all cloud groups (read-only). Complete the following steps to use the console to view and print reports for IP groups:

1. Access the console pane and click **System Console** → **Reports** → **Machine Activity**.
2. Select the IP group for which you want to view reports.
3. Click the **Report** icon on the toolbar.

   You also can access this view from the system console by selecting **Reports** → **Machine Activity**, choosing the **IP Usage by IP Group** report.

Figure 3-7 shows an example of an IP Group report.

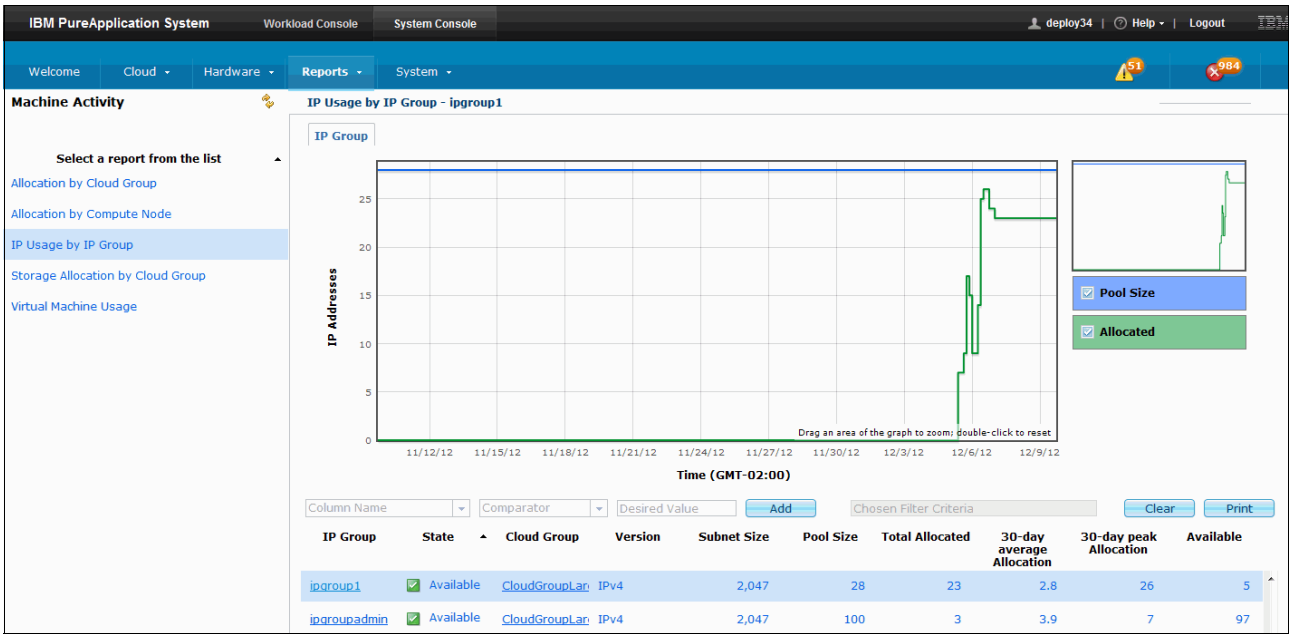


Figure 3-7 Sample of an IP group report

The report provides the following graphics:

- Pool Size
- Allocated

Optionally, you can customize these reports by choosing the individual data sets that can be displayed on each graphic:

- Daily average
- Daily peak
- Trend
- Capacity

A data table also is shown in the report. By clicking the column headers, you can sort the data in the chosen column. From the table, by right-clicking any of the table column headings you can add or remove data from the table. The following column headers are available:

- IP Group
- State
- Cloud Group
- Version
- Subnet Size
- Pool Size
- Total Allocated
- 30-day average Allocation
- 30-day peak Allocation
- Available

For more information about IP group monitoring, see the IBM PureApplication System Information Center website at this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/systemconsole/t_ipgroupreports.html

3.3.4 Deleting IP groups and addresses

To delete IP addresses from IP groups or delete the entire IP group, the cloud group administration role with permission to manage all cloud groups (full permission) is required. When an IP group is deleted, the pool of IP addresses within the IP group also is removed from the system.

Therefore, you should proceed with caution when you consider removing an IP group because it should not be deleted while it is assigned to any virtual machines. Rather, delete only when the IP groups addresses are all unassigned.

When an IP group is chosen for removal, you must also delete the pattern instance whose virtual machines are using those IP addresses. Also, make sure that you delete the wanted IP group before any of those addresses are assigned to another virtual machine.

Complete the following steps to use the console to delete IP addresses from an IP group or delete the IP group:

1. Access the console pane.
2. Click **System Console** → **Cloud** → **IP Groups**.
3. Complete the following steps to delete an IP address:
   a. Select the IP group that you want to edit.
   b. Locate the IP address that you want to delete from the IP group.
   c. Click the **[remove]** link that is next to the IP address.
4. Complete the following steps to delete an IP group:
   a. Select the IP group that you want to delete.
   b. Click the **Delete** icon in the row.

Figure 3-8 shows the process of deleting an IP group.

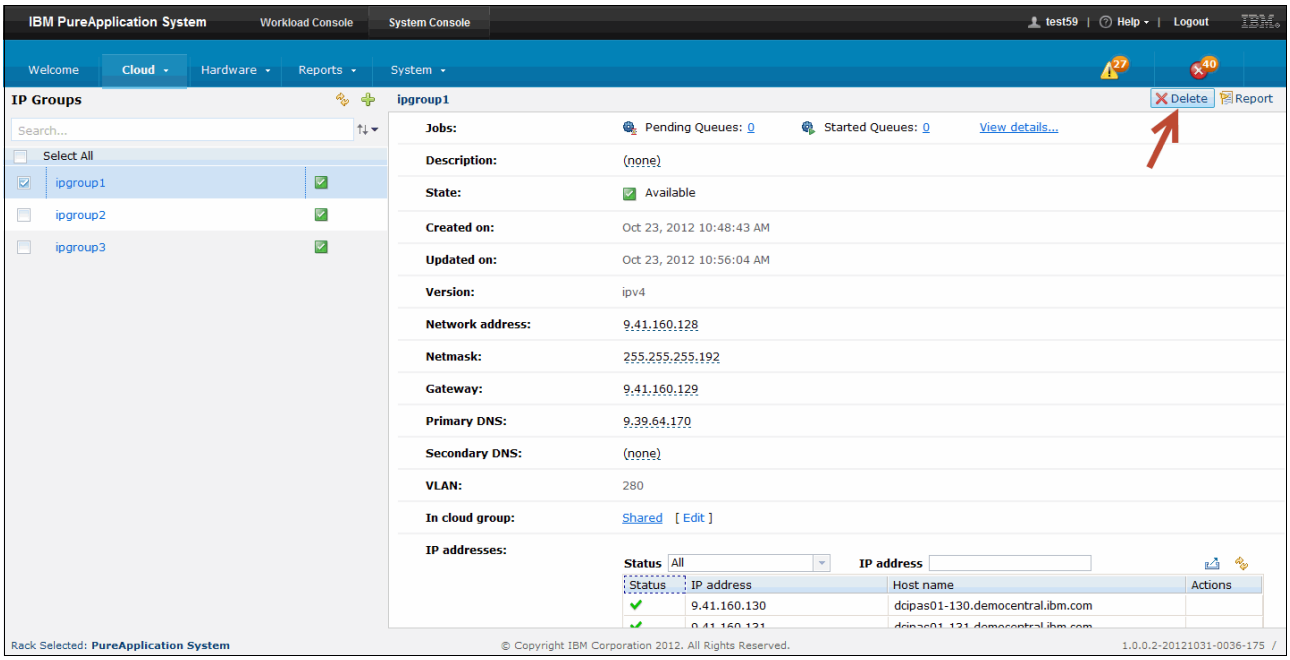


Figure 3-8 Deleting an IP Group

3.4 Managing cloud groups

A cloud group aggregates compute nodes with at least one IP group into a logical computer.

Each cloud group has their own computer equipment (compute nodes), which isolate runtime environments. This configuration ensures that processes that are running in one group are not affected by processes that are running in another group.

Each running virtual machine is run in a compute node that is a member of a cloud group. Each VM uses an IP address from the IP groups that are assigned to the cloud group.

Ultimately, a cloud group is used as the deployment target when patterns are deployed to create workloads, such as virtual systems or virtual application patterns.

To manage cloud groups in IBM PureApplication System, you must be assigned the Cloud group administration role with permission to manage all cloud groups (full permission).

3.4.1 Adding cloud groups

To create a cloud group, you need at least one assigned compute node (one that is not assigned to any pre-existing cloud group). In the IBM PureApplication System administration console pane, complete the following steps to create a cloud group:

1. Click **System Console** → **Cloud** → **Cloud Groups**, as shown in Figure 3-9.

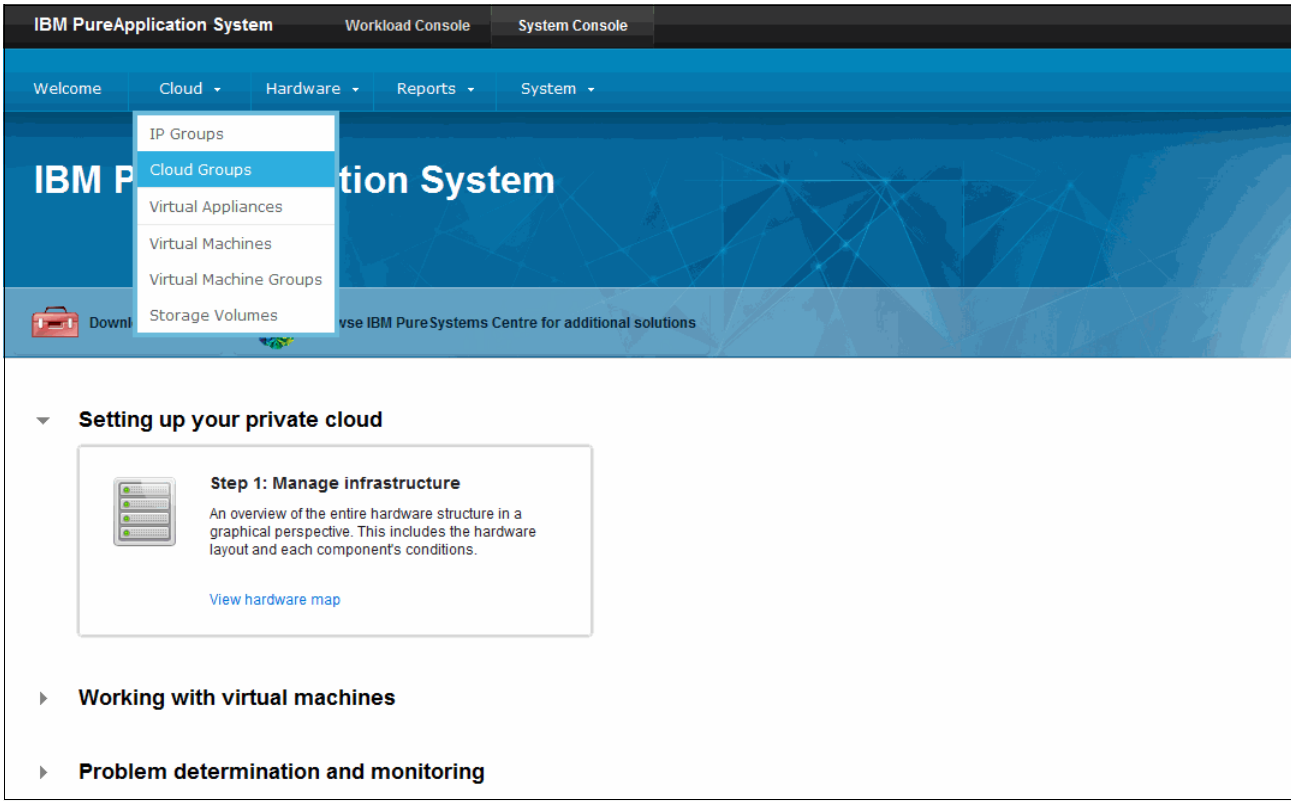


Figure 3-9 Accessing the cloud groups configuration from the system console

2. Click the **New** icon on the toolbar.

3. Define the cloud group by providing the following information:
   - Name

     The cloud group name. Each cloud group must have a unique name.

     Example: DEV, TEST, or PROD.
   - Description

     Detailed description of the cloud group.
   - Type

     Defines how resources, specifically CPUs, are allocated to virtual machines (VMs) during pattern deployment. Select one of the following types:

     - Dedicated

       Best suited for applications with high loads. Allocates the 16 cores on an Intel compute node as 16 CPUs.
     - Average

       Best suited for applications with low loads. Each of the 16 cores on an Intel compute node are allocated as 4 CPUs, which results in 64 logical CPUs.
   - Management VLAN ID

     An integer value 1 - 4094. The specified integer value cannot be in use in your data center. The VLANs that already are in use are listed in this field.
4. Click **OK**.

   Figure 3-10 shows an example of creating a cloud group.

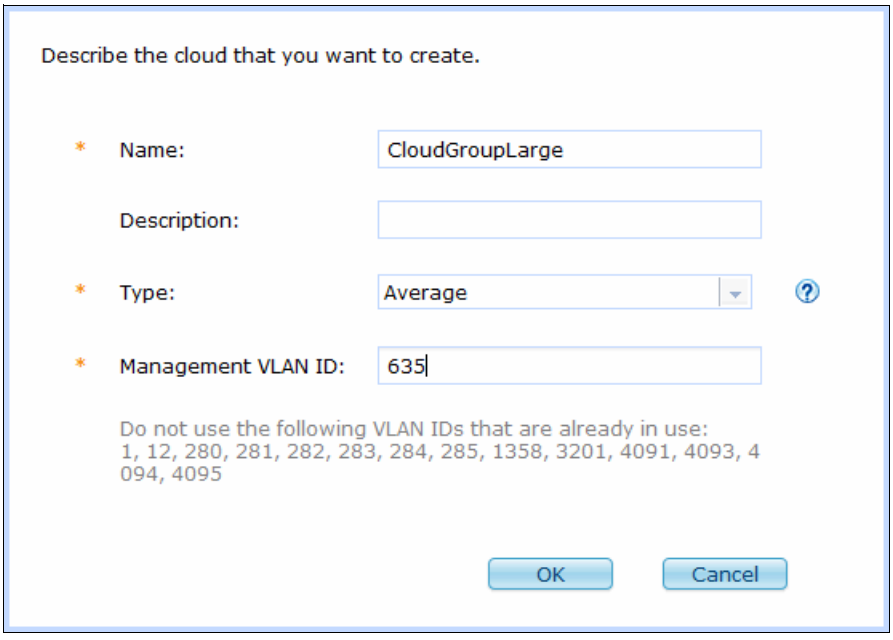


Figure 3-10 Creating a cloud group

The cloud group that is created cannot be started yet. To make it usable, you must add at least one compute node and one IP group. Then, it can be started and patterns can be deployed to it.

3.4.2 Administering cloud groups

To edit an existing cloud group in the IBM PureApplication System, you must be assigned the cloud group administration role with permission to manage all cloud groups (full permission). Complete the following steps to administer a cloud group:

1. Access the console pane.
2. Click **System Console** → **Cloud** → **Cloud Groups**
3. Select a cloud group to edit, as seen in Figure 3-11.

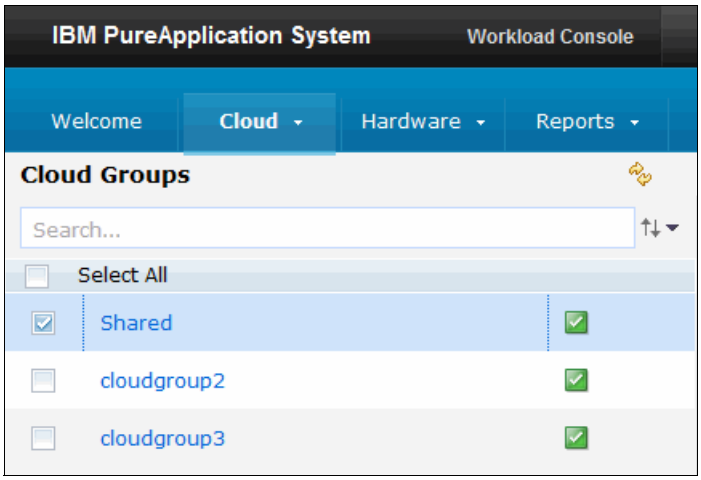


Figure 3-11 Select the Cloud Group to be edited

4. You can update the following cloud group settings:
   - Description
   - Type
   - Cloud group policy

     Specifies that the availability policy is enabled.
   - High Availability

     Specifies whether high availability is active.
   - Management VLAN ID
   - Access granted to:
     - Select **user IDs** from the Add more menu.
     - To remove a user, click **remove**.
   - IP groups:
     - To add an IP group to the cloud group, click **Add more** in the menu and select an IP group to add.
     - To remove an IP group, click **remove** in the Action column.
     - Click the IP group to review the IP group details in the IP Groups pane.
   - Compute nodes:
     - Click **Add more** to select a compute node to add.
     - To delete a compute node, click the **Delete** icon in the Action column.

– Volume configurations:
  - To create a volume for the chosen cloud group, click **Create a new configuration**.

    Enter the new volume configuration, defining the name, description, and size in megabytes.
  - Click **OK**.
– Hardware information

  Specifies the usage and allocation of CPU resources and physical memory.
– Hosts

  Specifies the memory and CPU usage by host name.

3.4.3 Deleting cloud groups

To remove a cloud group, no virtual machines can be a part of it. If there are virtual machines in the cloud, you cannot delete the cloud group (even if the VMs are not running).

To remove a Cloud group, use the IBM PureApplication System console. You must have the cloud group administration role with permission to manage all cloud groups (full permission). To remove a Clould group, complete the following steps:

1. Click **System Console** → **Cloud** → **Cloud Groups**.
2. Select the cloud group that you want to delete.

   Ensure that there are no virtual machines in the cloud. Otherwise, you must first stop the VMs and remove them before you can delete the cloud group.

   To see whether there are VMs in this cloud group, expand the virtualization statistics in the cloud group details.
3. Click the **Delete** icon on the toolbar.
4. Click **OK** to confirm that you want to delete the cloud group.

Figure 3-12 shows an example of how to delete a cloud group.

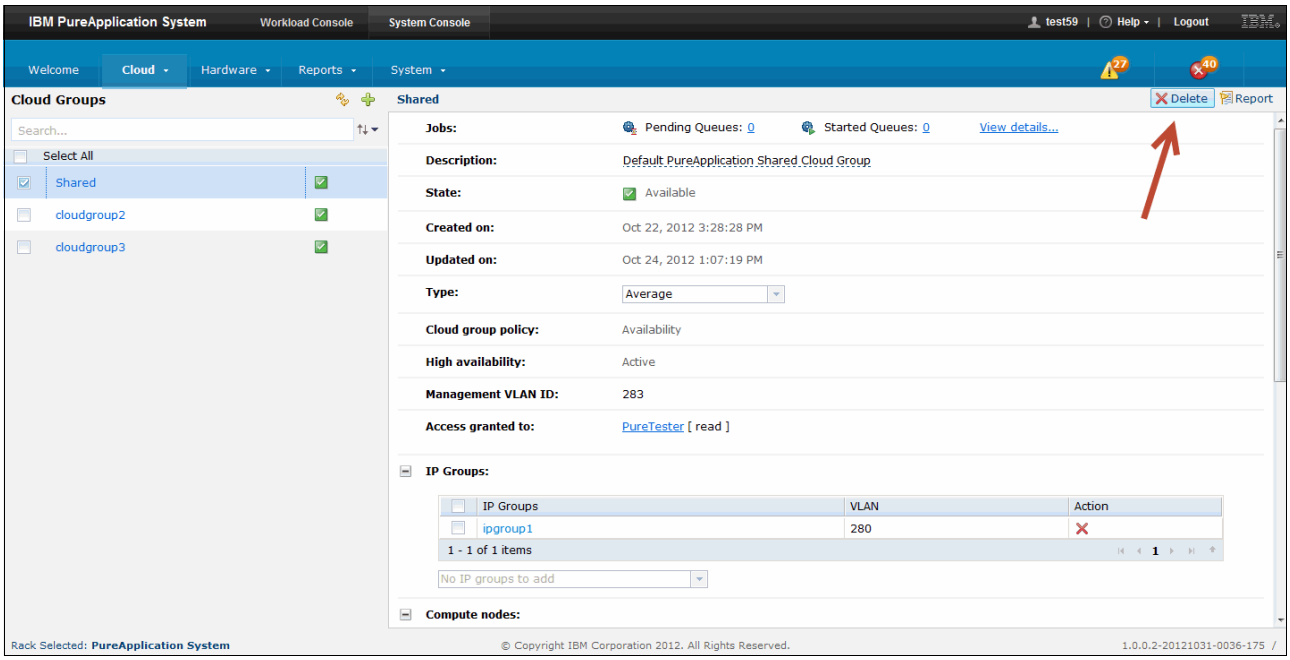


Figure 3-12 Deleting a cloud group

3.4.4 Viewing cloud group reports

The following procedures can be used to generate and view cloud group reports:

- Procedure 1:
  a. Click **System Console** → **Reports** → **Machine Activity**.
  b. Click **Allocation by Cloud Group** and the reports are shown in the right pane.
- Procedure 2:
  a. Click S**ystem Console** → **Cloud** → **Cloud Groups**.
  b. Select the target cloud group from the list of available cloud groups.
  c. Click the **Report** icon in the upper right pane to generate the Allocation by Cloud Group report.

Figure 3-13 shows an example of a cloud group report.

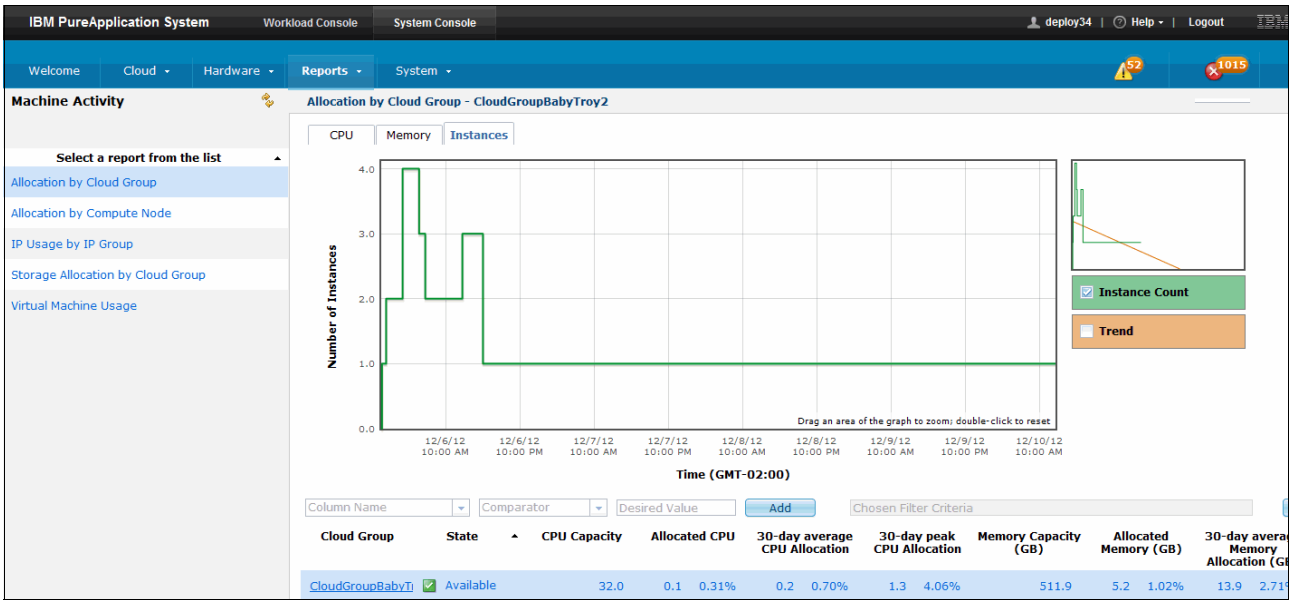


Figure 3-13 Sample of an Allocation by Cloud Group report

Customizing the report

The report provides three different graphics, which can be changed by clicking one of the following tabs that is provided in the console:

- CPU
- Memory
- Instances

Each graphic provides the following data sets:

- Daily average
- Daily peak
- Trend
- Capacity
- Instance count (Instances graph only)

The graphics are accompanied by a table with reference values. By clicking the column headers in the table, you sort the data that is shown in the graphic by the chosen column. You also can right-click any of the table column headings to add or remove a column heading from the table.

The following columns can be added or removed from the Column Name menu:

- Cloud Group
- State
- CPU Capacity
- Allocated CPU
- 30-day average CPU Allocation
- 30-day peak CPU Allocation
- Memory Capacity (GB)
- Allocated Memory (GB)
- 30-day average Memory Allocation (GB)
- 30-day peak Memory Allocation (GB)
- Instances

For more information about cloud group monitoring, see the IBM PureApplication System Information Center at this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/systemconsole/t_viewcloudgroups.html

3.4.5 Managing storage volumes in cloud groups

A storage volume is created through the IBM PureApplication System console to provide more storage for a virtual machine. Storage volumes are objects that exist outside of the virtual machine, though they are attached to the virtual machines. Therefore, its management is dissociated from the virtual machine administration.

Deleting a virtual machine does not automatically delete the storage volumes that are attached to that machine. The management, creation, and removal of storage volumes are self-contained administrative tasks that must be explicitly completed to achieve the result.

Adding a storage volume

A storage volume is created through the console and attached to a virtual machine for more storage. This process might be required when enterprise applications that are running on IBM WebSphere Application Server or IBM DB2 require more storage. It also might be required when you want to install packages and the operating system does not have enough space.

You must have the cloud group administration role with permission to manage all cloud groups (full permission) to create storage volumes.

Complete the following steps to add a storage volume:

1. Access the console pane and click **System Console** → **Cloud** → **Storage Volumes**.
2. Click the **New** icon on the toolbar.
3. Enter a name in the Name field.
4. Select a cloud group in the In cloud group field.
5. In the Volume configuration field, select one of the following options:
   a. **Existing settings** to use the existing size settings.
   b. **Custom settings** to create your own size settings for the storage volume.

   Important: The largest volume allocation that you can set for a virtual machine is a 1.8-TB raw disk.

6. Click **OK**.

Figure 3-14 shows the process of creating a storage volume.

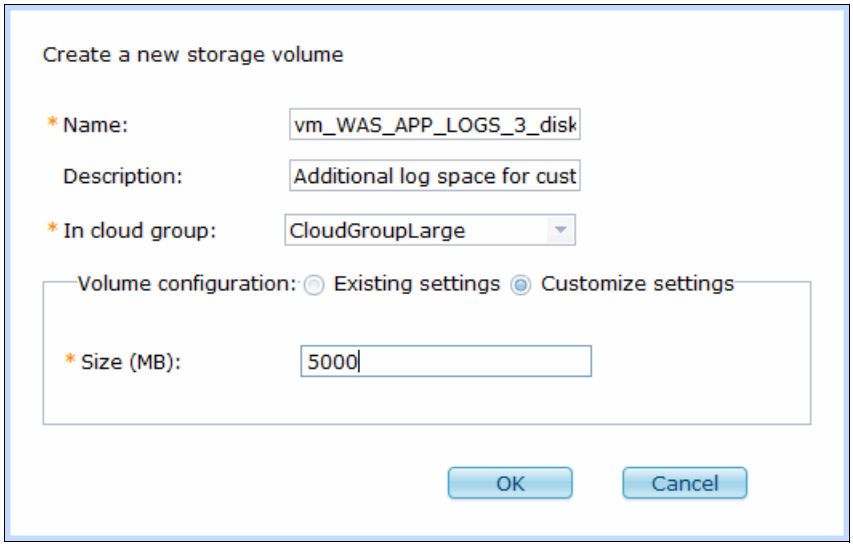


Figure 3-14 Creating a storage volume

A storage volume is now added to the cloud group.

Adding a storage volume to a virtual machine

After you add a storage volume to a cloud group, you can then add this volume to virtual machines.

To add this volume to virtual machines, complete the following steps from the System console:

1. Access the console pane and click **System Console** → **Cloud** → **Virtual Machines**.
2. Select a virtual machine from the list of available virtual machines.
3. Under the Storage Volume section in the Virtual Machine configuration panel, click the drop-down menu to select a volume, as shown in Figure 3-15.

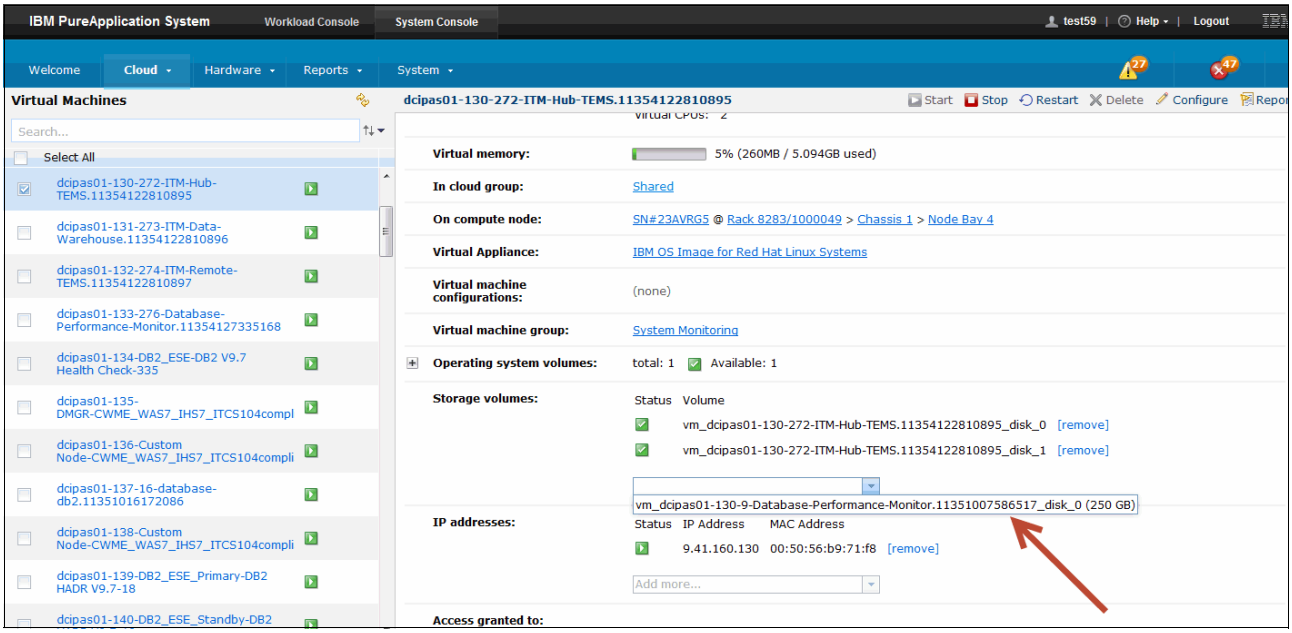


Figure 3-15 Adding more storage volumes to virtual machines

Administering a storage volume

By using the system console, you can manage existing storage volumes in the system. You must have the cloud group administration role with permission to manage all cloud groups (full permission).

Complete the following steps to manage storage volumes:

1. From the console pane, click **System Console** → **Cloud** → **Storage Volumes**.
2. Modify the following configurations as necessary:
   - Description
   - Storage Controller

Deleting a storage volume

Although they are associated with virtual machines, storage volumes are first-level objects that exist outside of virtual machines. Therefore, they must be explicitly deleted because deleting a virtual machine does not automatically delete the storage volumes that are associated with the virtual machine.

A storage volume can be removed by using the system console by an administrator that is assigned the cloud group administration role with permission to manage all cloud groups (full permission).

Complete the following steps to remove a storage volume:

1. Click **System Console** → **Cloud** → **Storage Volumes**.
2. Select the storage volume that you want to delete.
3. Click the **Delete** icon on the toolbar.
4. Click **Delete** to confirm that you want to delete the storage volume.

The storage volume configuration panel provides the information about the virtual machine that is using the target storage that is being deleted, as shown in Figure 3-16.

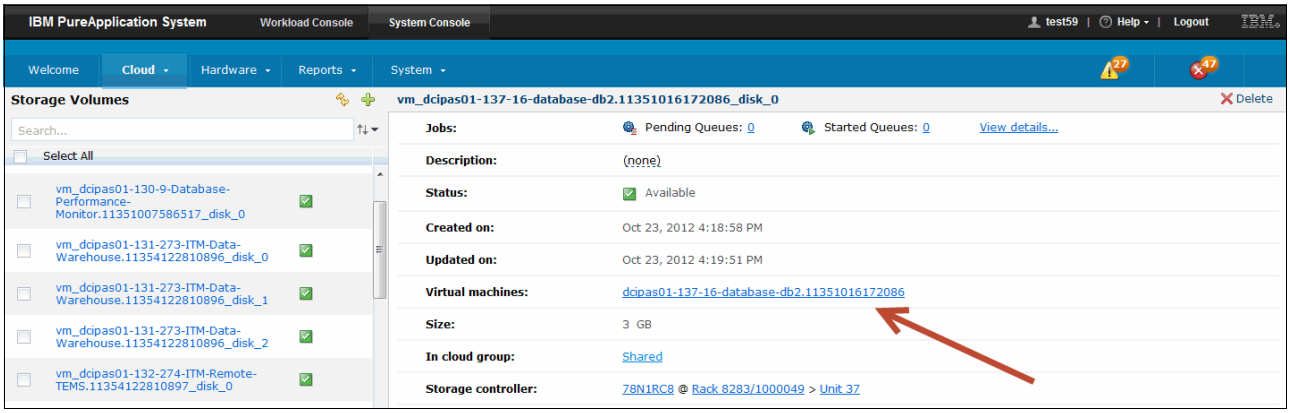


Figure 3-16 Virtual machine that is using the chosen storage volume

3.4.6 Managing virtual machines in cloud groups

You can manage virtual machines in a cloud group two ways: by having the cloud group administration role, with permission to manage all cloud groups (full permission) or by having permission for workloads that you deployed to a pattern instance. You can manage the virtual machines that are associated with the pattern instance because this permission is part of the access control list (ACL) for the pattern instance. The deployer is automatically added to the ACL.

From the virtual machines section in the IBM PureApplication System console, you can view the details of the virtual machines in the group. You can view information, including status, virtual memory usage, and virtual CPU usage.

This section describes the following processes that can be run against virtual machines in the system:

- Starting a virtual machine:
  a. From the console pane, click **System Console** → **Cloud** → **Virtual Machines**.
  b. Select the virtual machine that you want to start.
  c. Click the **Start** icon on the toolbar.
- Stopping a virtual machine:
  a. From the console pane, click **System Console** → **Cloud** → **Virtual Machines**.
  b. Select the virtual machine that you want to stop.
  c. Click the **Stop** icon on the toolbar.
  d. Click **OK** to confirm that you want to stop the virtual machine.
- Deleting a virtual machine:

To delete a virtual machine, the target virtual machine must be stopped already; otherwise, the delete icon is disabled.

a. From the console pane, click **System Console** → **Cloud** → **Virtual Machines**.
b. Select the virtual machine that you want to remove.
c. Click the **Delete** icon on the toolbar.
d. Click **OK** to confirm that you want to delete the virtual machine.

Figure 3-17 shows a sample of the management options for a virtual machine.

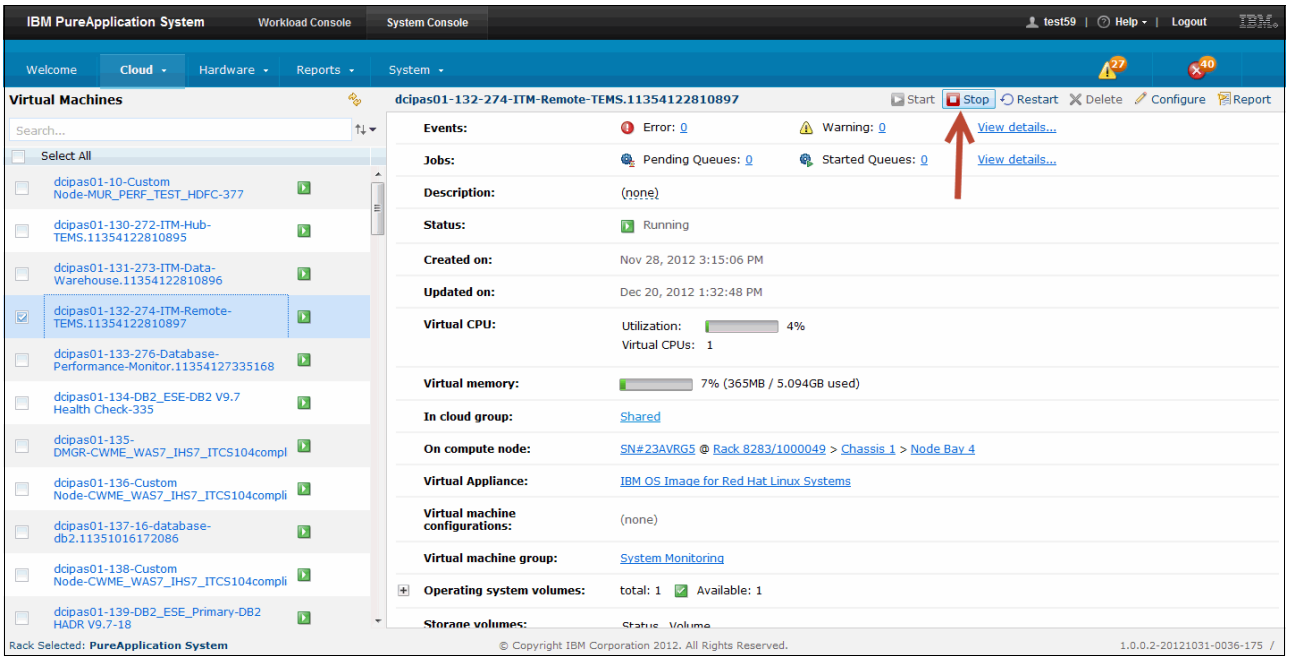


Figure 3-17 Managing virtual machines in a cloud group

- Viewing virtual machine reports:

  a. From the console pane, click **System Console** → **Cloud** → **Virtual Machines**.
  b. Select the virtual machine for which you want to review reports.
  c. Click the **Report** icon on the toolbar.
  d. From the Machine Activity pane, select the **Virtual Machine report**.
  e. Data in the monitoring graphics can be filtered by using the following options:
     - CPU usage, such as average CPU usage.
     - Memory usage, in megabytes (MB), including daily average memory allocation.
  f. Column headers in the monitoring reports provide information about the following specifics:
     - Virtual Machine
     - State
     - Application
     - Allocated CPU
     - CPU in use
     - 30-day average CPU Utilization
     - Allocated Memory (GB)
     - Allocated Disk (GB)
     - Memory in use (GB)
     - 30-day average Memory Utilization (GB)

- Allocated Disk

Figure 3-18 shows a sample report for a virtual machine.

Figure 3-18 Virtual machine report

- Configuring virtual machine memory and CPU

  Each virtual machine, in deployed patterns, include resource requirements settings. The following settings specify the resources that the VM requires to run properly:

  - CPU count: This count is the number of virtual CPUs assigned to this VM.
  - Virtual memory (MB): This value is the amount of virtual memory that is assigned to that VM.

  The accounting for the CPU count depends on the type setting of the cloud group in which the VM runs. For example, if the VM requires four virtual CPUs, the following actions occur:

  - A dedicated cloud group assigns the VM four physical CPUs.
  - An average cloud group assigns the VM one physical CPU.

  The numbers for these resource requirement settings should be higher for a VM that is expected to support a significant user load. If a VM's numbers are lower and it experiences high user load, this VM's users experience degraded performance (assuming the pattern instance does not load balance to other VMs). The degraded performance is related to not having enough resources to serve all of the requests with the customary response times.

Complete the following steps to configure CPU count and virtual memory settings:

1. From the console pane, click **System Console** → **Cloud** → **Virtual Machines**.
2. Select the virtual machine that you want to edit.
3. Click the **Configure** icon on the toolbar.
4. Complete the following fields:
   - CPU count: Number of virtual CPUs assigned to this virtual machine.

– Virtual memory (MB): Memory currently being used by the virtual machine.

5. Click **OK**.

Figure 3-19 shows the configuration pane for a virtual machine CPU and memory.

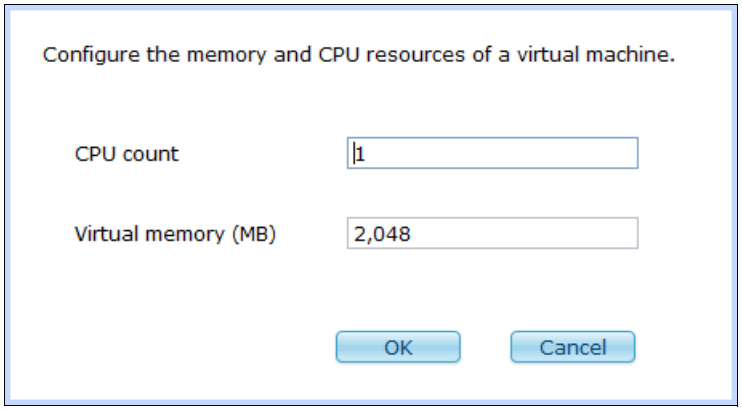


Figure 3-19 Configuring CPU and memory for a virtual machine

For more information about virtual machines management in IBM PureApplication System, see the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/systemconsole/t_mngvirtmachine.html

3.5 Managing shared services

Shared services provide virtual application patterns that can be deployed and shared by multiple virtual applications, virtual systems, and virtual appliances in the cloud.

Shared services provide the following features:

- A plug-in design service that provides full lifecycle management capabilities in the same way as a virtual application.
- Specific runtime services to multiple applications, or services to users on behalf of multiple applications.
- A simplified consumer (users or application deployments) and provider (implementation and Shared Service deployment) model.
- Viewed as a multi-tenant service.

Shared services are used in the following ways:

- Directly exposed by a resource in the Virtual Application Modeler (routing policy).
- Indirectly exposed though a setting that implies the usage of a shared service (HTTP session caching inside scaling policy).
- Injected though transformation process (injecting the logging service client).

Only one instance of a type of shared service can be deployed in a cloud group, which is a physical group of hardware that defines a cloud. This shared service can be used by all application deployments in the cloud group.

To view and manage shared services, you must be assigned the workload resources administration with full permissions or the cloud administrator role. With the correct access permissions, you can use the IBM PureApplication System administration console to manage shared services, such as caching, monitoring, and the elastic load balancing (ELB) proxy.

Section 4.6.5, “Shared services” on page 172 shows practical examples of shared services patterns deployment.

3.5.1 Existing shared services

IBM PureApplication System provides the following existing shared services that can be used with virtual application patterns in each cloud group:

- System Monitoring service

  The System Monitoring service provides the required resources (monitoring agents) to collect data that is related to performance and availability. The information that is provided by this feature is useful for system troubleshooting and making business decisions about your hardware and service usage.

  The System Monitoring service provides the following monitoring services:

  - Hub monitoring server
  - Remote monitoring server
  - IBM PureApplication System Monitoring Portal
  - Monitoring Agent for IBM PureApplication System
  - Automated load balancing to avoid overloading a remote monitoring server

- Caching service

  The caching service is a shared service that allows virtual deployments in the cloud to store, share, and access caching information among components. The caching service also provides auto-scale capabilities so that new component instances can be created or destroyed without retransmitting information.

  The caching service is based on WebSphere eXtreme Scale code and provides highly efficient caching. The caching service is self-managed and highly available, which provides simple and quick usage.

  When the caching service is managed, you can perform operations that are related to grid caching. Grid caching maintains data that can be accessed from multiple clients, which minimizes network latency and reduces bandwidth.

- Elastic load balancing (ELB) proxy service

  The elastic load balancer proxy service is a shared service that provides routing and load balancing to multiple deployed web applications. It provides a front end to virtual applications in the cloud by balancing the load across the instances of virtual applications. The service is shared by virtual applications that are deployed to the same cloud group.

  When the ELB service is deployed and running in a cloud group, all new deployments of virtual applications are automatically configured to use the ELB service.

- Database Performance Monitor service

  The database performance monitor service provides information about the health and availability of databases that are deployed in the PureApplication System. This information provides early warning of degraded database performance.

  The database performance monitor also provides drill-down capability to determine the root cause of performance problems.

3.5.2 Deploying the System Monitoring service

After the System Monitoring service is deployed, it uses three virtual machines that are created to host the monitoring infrastructure for IBM PureApplication System.

The following functions are deployed to facilitate the collection of performance and availability information by monitoring agents:

- Hub monitoring server
- Remote monitoring server
- PureApplication System Agent
- Tivoli Data Warehouse

To deploy the System Monitoring service in IBM PureApplication System, you must be assigned the following roles:

- Allow delegation when full permission is selected.
- Workload resources administration role with permission to manage workload resources (full permission).
- Cloud group administration role with permission to view all cloud groups (read-only).
- Hardware administration role with permission to view all hardware resources (read-only).
- Security administration role with permission to view users/groups (read-only).

Complete the following steps to deploy the System Monitoring service:

1. At the IBM PureApplication System main menu, click **Workload Console**.
2. Click **Cloud** → **Shared Services**.
3. Click **System Monitoring** under Monitoring Services.
4. Click **Deploy** (in the upper right of the pane).

   The Configure and deploy a shared service pane is displayed.
5. Complete the following fields:
   - Password for the user ID sysadmin: The initial user ID for logging on to IBM PureApplication System Monitoring Portal.
   - Password for the user ID itmuser: The itmuser ID is a user ID for collecting historical data.
   - Password for the user ID db2inst1: This user is a user ID for creating and maintaining the Tivoli Data Warehouse database.
   - Shared Service Sizing

     Set the expected size of the System Monitoring service, which is based on the number of monitoring agents that you plan to connect to the monitoring servers in the System Monitoring service. You have the following options:

     - Large (up to 5,000 agents)
     - Medium (up to 2,000 agents)
     - Small (up to 500 agents)
     - Tiny (up to 200 agents)
6. Click **OK**.

Figure 3-20 on page 90 shows the configuration form for the System Monitoring shared service.

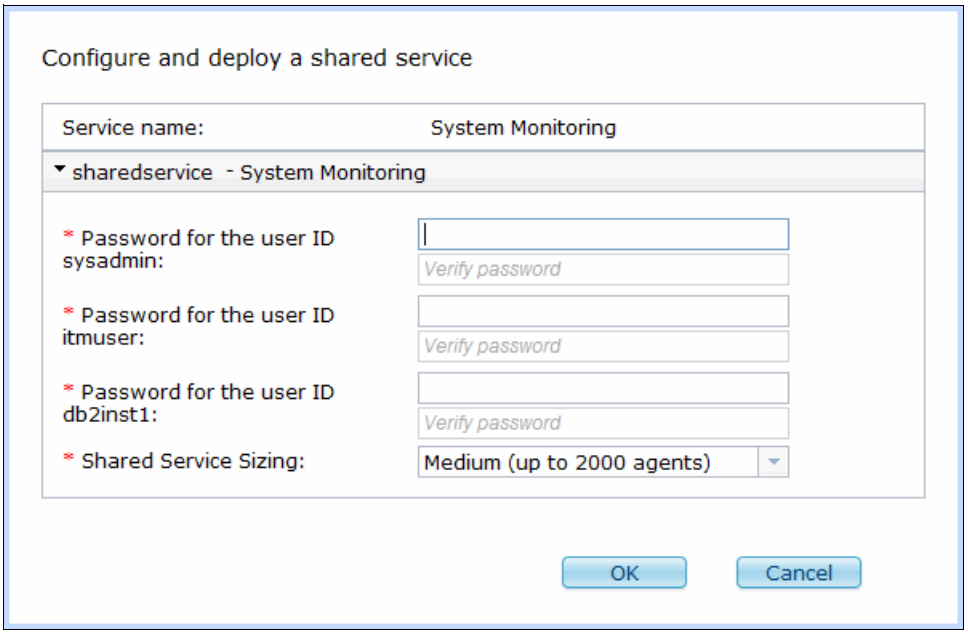


Figure 3-20 Configuring the System Monitoring shared service

7. In the Deploy Virtual Application pane, complete the following steps:
   a. Select the cloud group to which you deploy the virtual application from the Cloud group drop-down list.
   b. Select the profile that you want to use from the Profile drop-down list.
   c. Click **Generate** to generate an SSH Key.
   d. Click **Download** to save the key to your local system.
   e. Click **OK**.

3.5.3 Using an external caching service

A common caching service scenario is deployed virtual machines that become a caching service and are automatically used by the virtual applications. Another approach is to set up a caching service for the cloud environment that is external. A WebSphere DataPower® XC10 Appliance is set up outside the cloud environment and provides caching services. The virtual applications are connected to the external caching service. This approach saves cloud resources and the XC10 Appliance provides a larger space than the internal caching service.

Complete the following steps to set up an XC10 Appliance to cache your cloud environment:

1. Open the console.
2. In the Workload Console, click **Cloud** → **Shared Services**.
3. Select Caching Service (External).
4. Click **Deploy** in the upper right corner.
5. Complete the following fields:
   – External Caching Appliance Host Name

- External Caching Appliance Administrative User Name
- External Caching Appliance Administrative User Password
- External Caching Appliance Public Certificate

  To use the XC10 Appliance as the caching service provider, you must import its trust certificate from the XC10 truststore file.

6. Click **OK**.

For more information, see the IBM PureApplication System at this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/workloadconsole/t_addssinst.htm

3.5.4 Deploying an ELB proxy shared service

The steps to deploy the ELB proxy service that is shipped with the IBM PureApplication System product are described in the following process. This process includes adding the shared service and deploying the shared service into the cloud system:

1. Click the **Workload Console** tab at the IBM PureApplication System console.
2. Click **Cloud** → **Shared Services**.
3. Select **ELB Proxy Service** and click **Deploy** on the toolbar.
4. Complete the following information in the Configure and deploy a shared service pane:
   - ELB instances range

     Specify the minimum and maximum number of ELB instances.
   - Enable autowiring

     Select this option if you want all new virtual application pattern deployments in the cloud group to automatically use elastic load balancing.
   - Virtual Host

     Specify the default virtual host for virtual applications in the cloud group.
   - Scaling Properties

     Specify the following options for scaling ELB instances:
     - Select the **CPU based** option to enable or disable automatic scaling.
     - With automatic scaling enabled, specify a threshold range and minimum time to trigger automatic scaling.
5. Click **OK**.

Figure 3-21 shows the configuration of an ELB proxy shared service.

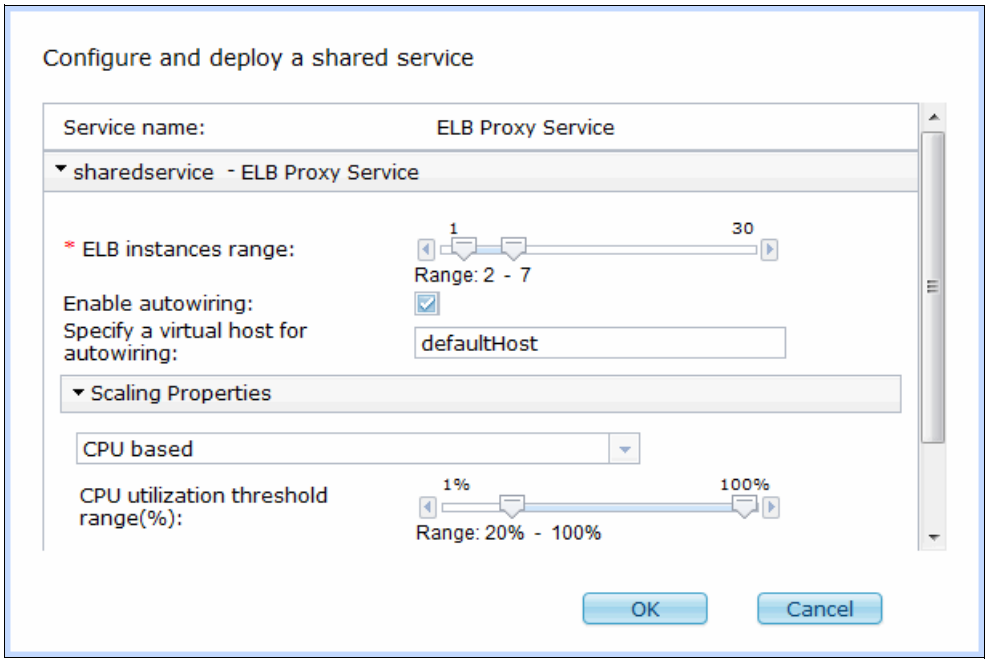


Figure 3-21 Deploying an ELB proxy shared service

6. Complete the following steps to deploy Virtual Application of the ELB proxy:
   a. Set the target cloud group.
   b. Set the IP version (IPv4 or IPv6).
   c. Select the cloud group.
7. In the Advanced section, you can set up SSH access to the virtual machines in the cloud group for troubleshooting and maintenance purposes.

   For more information about SSH settings, see this website:

   http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/apt_sshiwd.html
8. Click **OK**.

After these steps are completed, an ELB proxy shared service is deployed as an application. The instance is available in the list in the Shared Services Instances pane. The deployed service consists of one ELB Management virtual machine and a number of ELB Instance virtual machines as specified in the initial range setting.

3.5.5 Deploying the Database Performance Monitor service

Deploying the Database Performance Monitor service provides the monitoring infrastructure and information about the health and availability of DB2 databases that are deployed in the PureApplication System. This service detects signs of degraded database performance. Moreover, this monitoring service provides drill-down capability to determine the root cause of performance problems.

The database performance monitor service provides the following indicators and monitoring:

- A single health indicator for each database.
- Indicators to identify performance problems in the following areas:
  - I/O
  - Memory
  - Storage
  - Workload
  - Sorting
- Each database instance has a monitor that provides information about its health and performance, including the following areas:
  - SQL Statements
  - I/O
  - Memory
  - Connection dashboards

In addition to health and performance information, some dashboards have services to tune database performance.

The database performance monitor service must be running for the monitoring services to be available. To deploy the Database Performance Monitor service, you must have the cloud administrator role. Any administrative users have access to multi-database performance monitoring, which covers all databases instances that are deployed in a cloud group.

The deployment is performed only once for each cloud group. Where Database Performance Monitor is required, complete the following steps:

1. From the Workload Console tab, click **Cloud** → **Shared Services**.
2. From the list of shares services, select **Database Performance Monitoring**.
3. Click **Deploy**.
4. Select the following sizing that is most appropriate for your environment:
   - For eight or fewer databases with an aggregate database size of 400 GB or less, select **Small**.
   - For 20 or fewer databases with an aggregate database size of 1 TB, select **Medium**.
   - For anything larger, select **Large**.
5. Click **OK**.
6. In the Deploy Virtual Application pane, complete the following steps:
   a. Select the cloud group to which you want to deploy the virtual application.
   b. Select the **Advanced** option.
   c. Click **Generate** to generate an SSH Key.
7. Click **OK**.

Figure 3-22 shows the Database Performance Monitoring service.

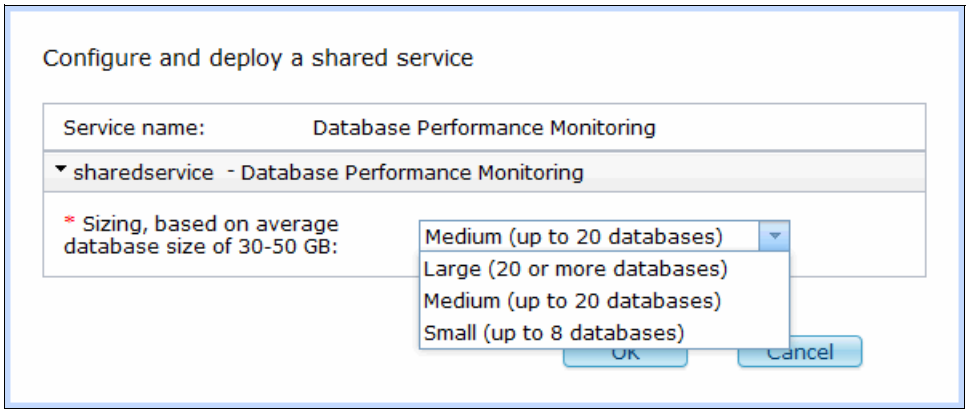


Figure 3-22 Deploying the Database Performance Monitor service

3.6 Managing environment profiles

Environment profiles are policies that group related deployment configuration, such as virtual machine names, IP address assignment, and cloud groups, which form logical deployment environments. Environment profiles also define how patterns are deployed to these resources and how these virtual pattern instances run in the cloud groups. This configuration provides IBM PureApplication System the capability to create logical isolation of resources through their allocation.

An environment profile can specify one or more cloud groups where it can be deployed. Also, a cloud group can be mapped by several environment profiles. Moreover, environment profiles grant access to user groups to specify who can use the profiles to deploy patterns. A profile can grant access to multiple user groups and a user group can be granted access to multiple profiles. Figure 3-23 shows how a few environment profiles might look and how they relate to other entities, such as cloud groups, user groups, and IP groups.

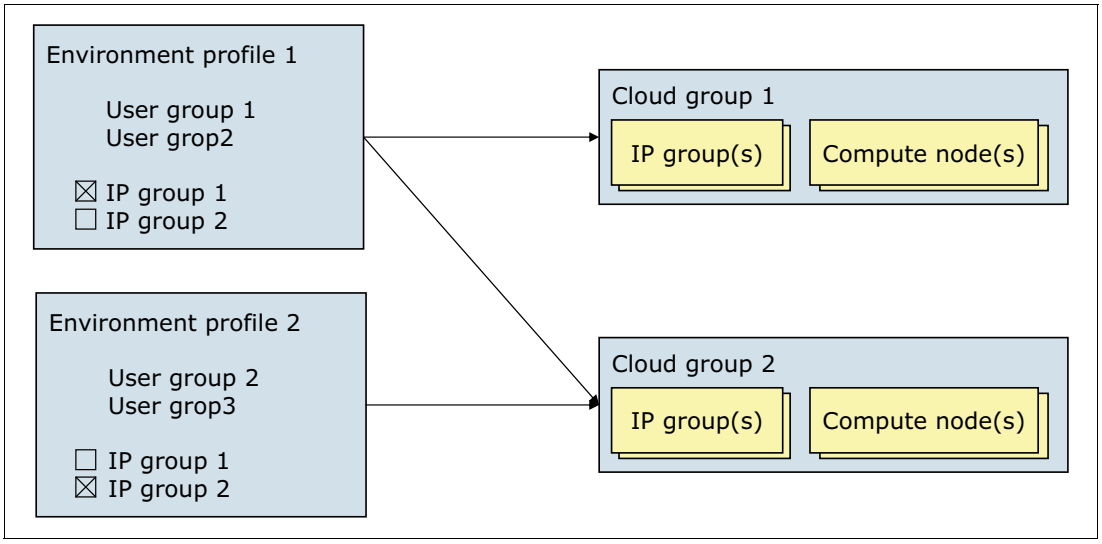


Figure 3-23 Typical PureApplication System resource instances

3.6.1 Adding environment profiles

To create an environment profile that is ready to be used for pattern deployment, you must have a cloud group that is configured and ready, with all hypervisors configured and available.

Complete the following steps to access the IBM PureApplication System administrative console to create an environment profile in the system:

1. Click the **Workload Console** tab at the top of the Welcome page.
2. Click **Cloud** → **Environment Profiles**.
3. Click the **New** icon (a plus sign image) on the toolbar.
4. Provide the following information:
   - Name

     Enter a unique name for the profile.
   - Description

     Enter a detailed description to identify the profile.
   - Hypervisor type

     Select the type of hypervisor in the cloud group or cloud groups that you are using. By default, it is `PureSystems_ESX` (VMware ESX).
   - Environment

     Select the environment in which this profile is to be created. The following options are available:

     - All
     - Development
     - Test
     - Quality Assurance
     - Performance
     - Research
     - Production
     - Pre-Production
5. Click **OK** to create the profile.

   Figure 3-24 shows the initial creation of an environment profile.

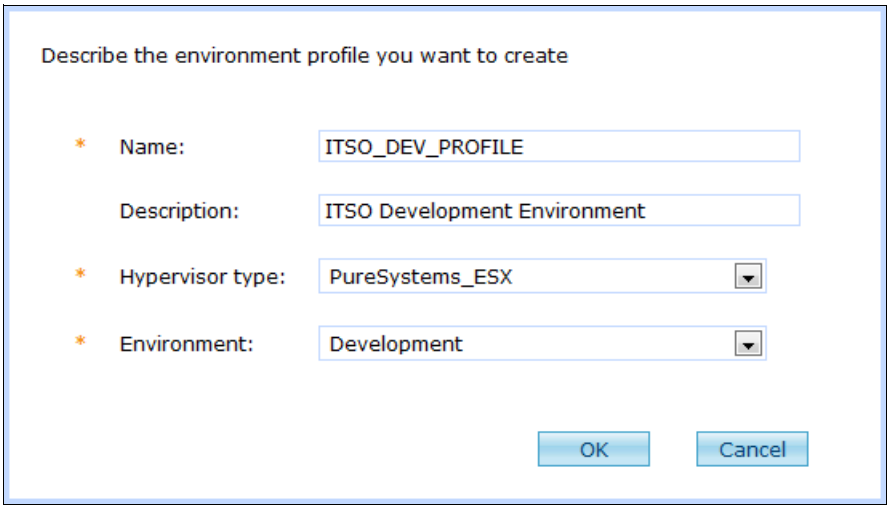


Figure 3-24 Creating an environment profile

6. When the information is processed, the new profile is added to the Environment profiles list on the left panel of the panel, as shown in Figure 3-25.

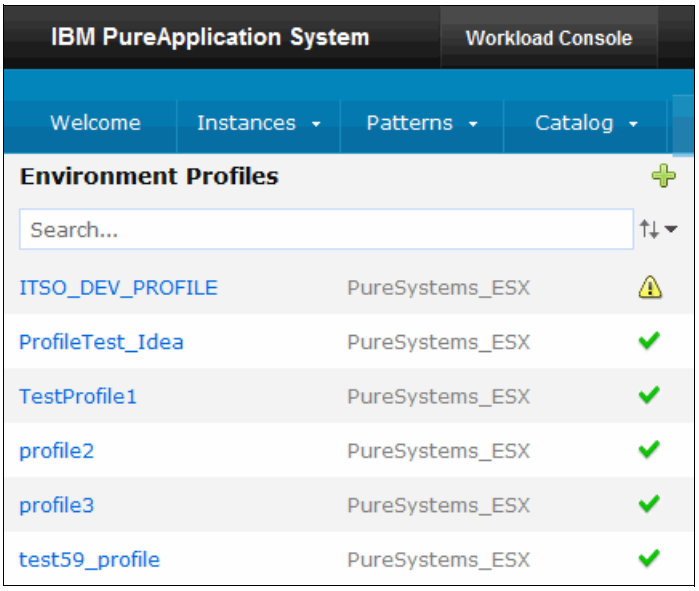


Figure 3-25 List of available Environment profiles

3.6.2 Editing an environment profile

After an environment profile is created, you can modify its settings to suit the changing needs of your environment.

Complete the following steps to use the console to edit environmental profile configurations that you can access:

1. Click the **Workload Console** tab at the top of the Welcome page.
2. Click **Cloud** → **Environment Profiles**.
3. Select the environment profile to edit.
4. Modify the following information as necessary:
   - Description of the environment profile.
   - The environment type in the Environment field.
   - Virtual machine name format (Optional)

     Use the following predefined variables:

     - ${*hostname*}

       Replaced with the host name of the virtual machine; for example, `My${hostname}VM`.

     - ${*vs-name*}

       Replaced with the name of the virtual system instance; for example, `My${vs-name}VM`. This variable must be used with one of the other formatting variables. Otherwise, all virtual machines have the same name and any deployment fails.

     - ${*x-counter*}

       Replaced with a counter of x digits; for example, `MyVM${3-counter}`.

– IP addresses provided by

  Specifies whether IBM PureApplication System or the pattern deployer provides the IP address on the deployment. If configured to IP Groups, the IP address is obtained automatically from the IP pool. Otherwise, when configured to Pattern Deployer, the IP address must be manually assigned by the user at pattern deployment time.

– Deployment priority

  This setting is used to prioritize pattern instances that deal with resource contention (deployment, runtime resources, and failover). The following priority levels are available:

  - Platinum: High(16) Medium(8) Low(4)
  - Golden: High(12) Medium(6) Low(3)
  - Silver: High(8) Medium(4) Low(2)
  - Bronze: High(4) Medium(2) Low(1)

– Deploy to cloud groups

  Select available cloud groups that are configured and ready for use. When you select a cloud group, the following information for the IP groups in that cloud group is provided:

  - In use

    Select this option to use the IP group in the environment profile.

  - Name

    Specifies the name of the IP group in the cloud that you selected.

  - Alias

    You can specify an alias name for the IP group for use in the environment profile.

  - Subnet address

    Specifies the subnet address of the IP group.

  - Gateway

    Specifies the gateway address of the IP group.

  - Netmask

    Specifies the netmask address of the IP group.

– Environment limits

  Provides information about the limits of the virtual CPU, virtual memory, storage, and product licenses. You can set the license limit for each product by clicking the up or down arrow in the license limit column.

– Access granted to

  Specify access to this environment profile for other users or groups of users. When a user or group is added, click the link beside the entry to toggle between the following access levels:

  - Read
  - Write
  - All

– Comments

  Add comment information that is relative to this environment profile.

Figure 3-26 shows the configuration of a new environment profile.

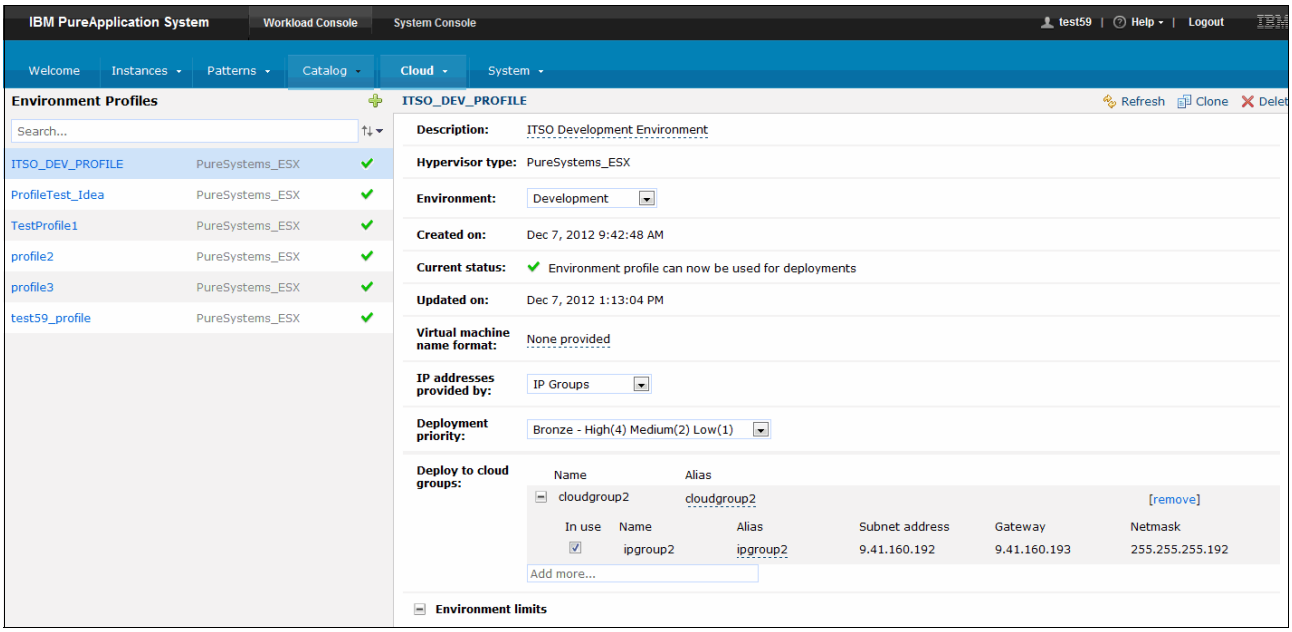


Figure 3-26 Environment profile configuration

3.6.3 Cloning an environment profile

Environment profiles that were created in IBM PureApplication System can be cloned and reused in some of the existing configurations. The cloning procedure is simple and requires only that you select an environment profile that most closely meets your needs. If the new profile is to deploy in a cloud other than the one that is specified in the original profile from which you are cloning, you must have a different cloud group that is configured and ready to be used.

All hypervisors must be configured and available in a cloud to create an environment profile that is ready to be deployed. Finally, when a profile is cloned, the hypervisor type cannot be changed.

From the IBM PureApplication System console, complete the following steps to clone environment profiles that are created in the system:

1. Click the **Workload Console** tab at the top of the Welcome page.
2. Click **Cloud** → **Environment Profiles**.
3. Click the wanted profile in the list of available profiles.
4. Click the **Clone** icon on the toolbar.
5. Provide the following information about the new environment profile:
   - Name

     Enter a new unique name for the environment profile.
   - Description

     Enter a description of the environment profile.
6. Click **OK**.
7. Enter the remaining profile configurations, following the same steps as described in 3.6.2, “Editing an environment profile”.

Figure 3-27 shows how to clone an environment profile.

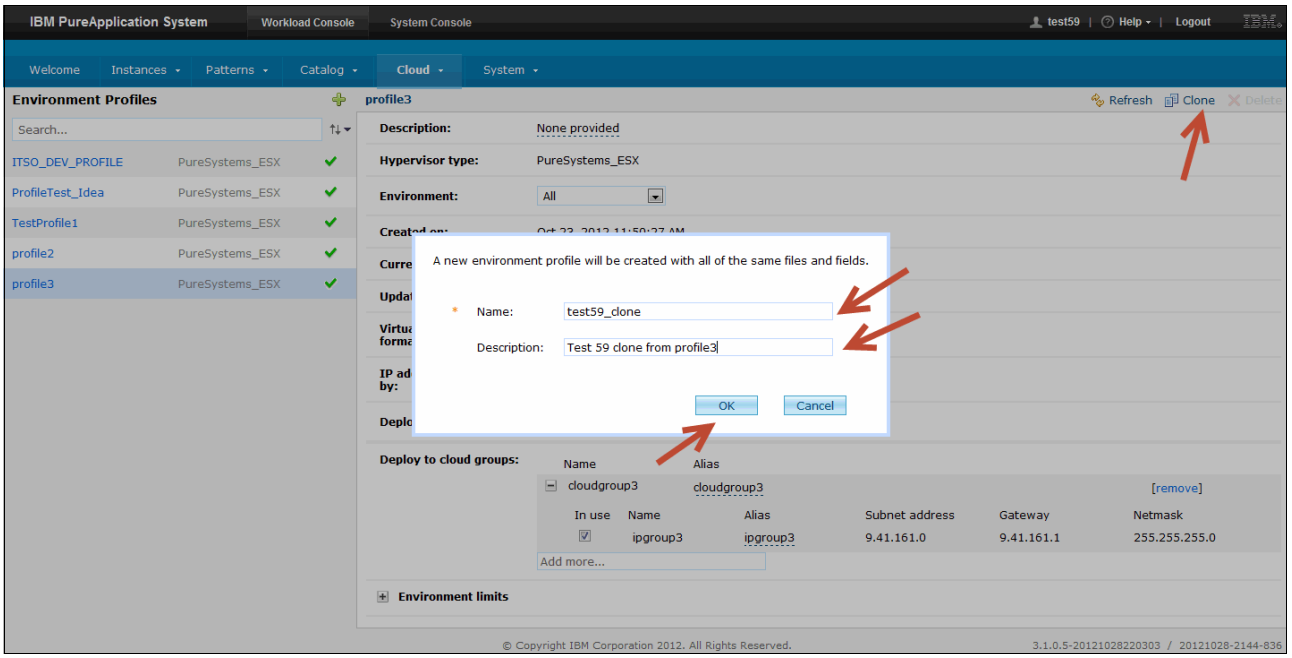


Figure 3-27 Cloning an environment profile

3.7 IBM PureApplication System isolation strategies

When an application is deployed to IBM PureApplication System, it is created as a pattern instance. Pattern instances contain the necessary virtual machines that run the middleware software that is needed to service the application requirements. The performance of these virtual machines that are created to run the pattern instances can be tuned through environment profiles and cloud group configurations. Those profiles and configurations influence the behavior of the virtual machines in the following respects:

- Prioritization

  Each virtual machine instance has prioritization settings. These settings define the importance of each virtual machine relative to all the other virtual machines in the same cloud group.

  These settings are significant during resource contention scenarios when a cloud group's virtual machines require more resources than the cloud group has available. In these situations, the system gives preference to the higher priority virtual machines.

  IBM PureApplication System prioritizes a VM based on the following settings:

  - Profile deployment priority

    The deployment priority, which is specified by the administrator, is in the environment profile that is used to deploy the pattern. The following priority values are available:

    - Platinum
    - Golden
    - Silver
    - Bronze

– Deployer deployment priority

  The priority that is used to deploy a pattern (set by the deployer) in the pattern deployment properties dialog. The following list notes the possible priority values:

  - High
  - Medium
  - Low

Important: All virtual machines that are in the same pattern instance have the same priority because they are deployed together.

Prioritization also is relevant during failover, as in the case of a compute node failure. In such a scenario, the system recovers virtual machines from a failed node by restarting them on other available compute nodes in the same cloud group. Virtual machines with higher priority are recovered before lower priority virtual machines.

If the target compute nodes in the cloud group do not have enough resources available to restore all the virtual machines from the failed compute node, the lower priority virtual machines are not restarted. This situation requires manual intervention from a system administrator.

► Resource requirement

Each virtual machine that is deployed in IBM PureApplication System requires resources to function properly. The amount of resources that are needed is defined by configuring what a virtual machine uses from the pool of resources. These resources are allocated by the environment profile and provided by the cloud group. The following resources are the most common:

– CPU count

  The number of virtual CPUs that are assigned to a virtual machine. This value depends on the following type setting of the cloud group that is associated with the virtual machine:

  - A dedicated cloud group uses one physical CPU as a virtual CPU.
  - An average cloud group creates four virtual CPUs from a physical CPU.

– Virtual memory (MB)

  The amount of virtual memory that is assigned to a virtual machine.

When the resource requirements for virtual machines in a pattern are set, you must find the correct balance between application performance and resource usage. You must ensure that a virtual machine has at its disposal all of the resources that are required to handle the expected application load (without performance degradation). But, at the same time, you also must ensure that only the amount of resources that are required to handle the loads is provided.

As resources are used, they have a direct effect on the environment limits that are set by the environment profile. When the limit for a profile reaches zero, you cannot use that profile to deploy any more patterns until some of its pattern instances are stored or deleted.

Likewise, patterns that are deployed by separate environment profiles can over-allocate the resources in a cloud group. VMs with overly generous resource settings increase this over allocation unnecessarily.

Achieving the correct combination of application performance and resource allocation is important. This balance helps maximize the number of pattern instances that can be deployed with an environment profile and that can run in a cloud group.

3.7.1 Resource isolation via networking

IBM PureApplication System isolates management traffic and application workload traffic by using separate external physical networks. Another level of resource isolation among deployments is supported by the construct of cloud groups, IP groups, and environment profiles.

Virtual network isolation can be configured on a management network. Each cloud group has a unique management VLAN, such that VMs in a cloud group cannot access VMs in a different cloud group through the management VLAN. Virtual network isolation can be configured on application network by using IP groups. Two IP groups are isolated from each other if they are on different VLANs.

Through a combination of cloud groups and IP groups, an environment profile can have virtual network isolation from other environment profiles. This isolation is for management networks and application networks.

Figure 3-28 shows an example of virtually isolated environment profiles.

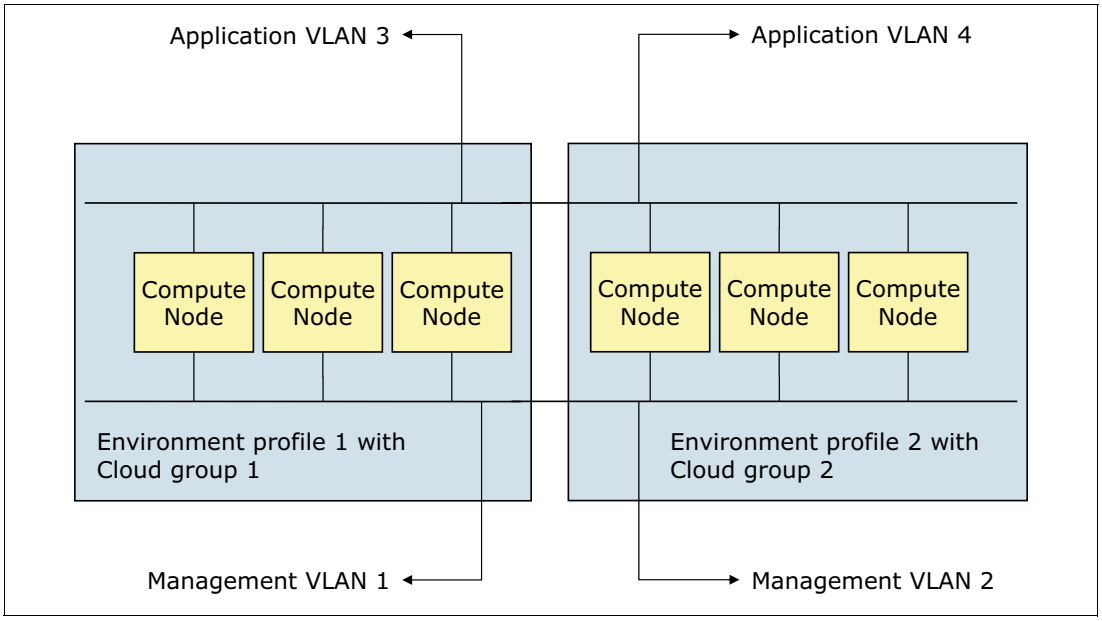


Figure 3-28 Environment profiles with physical compute node isolation and virtual network isolation

3.7.2 Resource isolation via access control

In addition to resource isolation by networking, virtual networking, and trust domains, PureApplication System supports resource isolation by using access control. This configuration provides workload resource and cloud resource isolation, such that users can deploy only workload patterns to cloud resources to which they are explicitly granted access rights. By using the fine-grained resource access control, you can set up environment profiles and isolate cloud resources. These controls are in place so that different groups of users are can use a different set of cloud resources.

Separate environment profiles can be configured to address different business needs for different users and user groups. Use an access control policy to maximize the environment profile access rights. User access to cloud resources is managed by using the environment profiles, which specify the set of cloud group resources, including IP groups, resource allocations, and license limitations that authorized users can use for deployment.

Users must be explicitly granted read access rights to an environment profile to deploy virtual application patterns and virtual system patterns to that environment profile. To deploy workload resources to an environment profile, a user must have the privileged Workload administration security role or read access rights to the environment profile.

Workload administrators with full permission roles must have read access rights to cloud resources to perform the following tasks:

- Create an environment by using those cloud resources.
- Grant other users access rights to the environment profile.
- Deploy shared services to the environment profile.

Users are automatically granted resource access rights to the virtual applications and virtual systems they deploy. However, users often are not granted access to system resources such as cloud groups, IP groups, compute nodes, and software licenses.

User management

IBM PureApplication System user management supports centralized user authentication by using an enterprise LDAP user directory server.

PureApplication System organizes security roles into the following areas of management responsibility:

- Workload administration

  Essentially managing and monitoring all management function on the Workload console. Users are automatically granted the following sub roles:

  - Create new patterns
  - Create new environment profiles
  - Create new catalog content
  - IBM License Metric Tool

- Cloud group administration

  Managing and monitoring cloud resources such as cloud groups, IP groups, virtual machines, and storage volumes.

- Hardware administration

  Managing and monitoring hardware resources, system configuration, events, job queues, and system activities.

- Security administration

  Managing security configuration, users, and user groups.

- Auditing administration

  Managing and monitoring auditing resources.

3.7.3 Application environments

Companies often use an approach to separate environments according to application development lifecycle. The following typical divisions are available:

- Development (DEV)

  An environment that is used for developing applications.

- Testing (TEST)

  Used for testing applications.

- Production (PROD)

  Used for running applications for use by business or users.

Each of these environments often are run on independent sets of hardware and network traffic to avoid cross environment issues.

A best practice in IBM PureApplication System is to create three cloud groups, one for each stage of the application development lifecycle, as shown in the following configuration examples:

- DEV cloud group, with one compute node, which gives developers 16 physical cores. Setting the cloud group type to `average` gives a 4-to-1 ratio of virtual CPUs, which gives developers 64 virtual CPUs.
- TEST cloud group with two compute nodes and cloud group type as `dedicated` to mimic production.
- A PROD cloud group, with three compute nodes and the cloud group type that is set to `dedicated`, since production applications are expected to be used heavily.

Each of these three cloud groups also need at least one IP group with an unused VLAN ID to keep their network traffic separated.

Figure 3-29 shows the isolation between the three application environments that are proposed in this scenario.

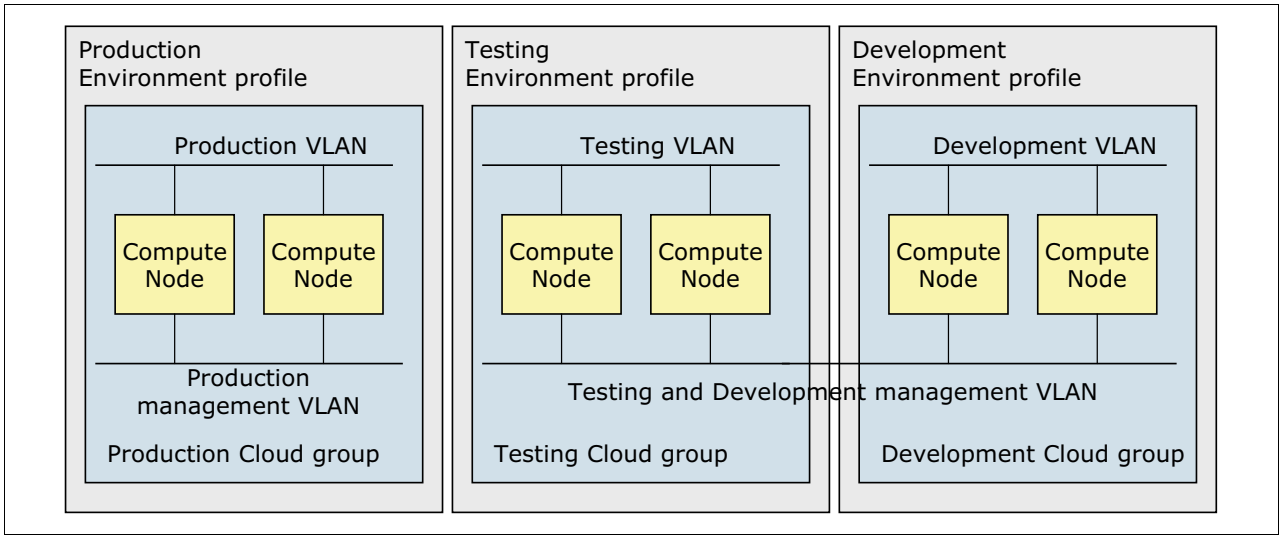


Figure 3-29 Resource isolation by using environment profiles

There are no hardware limitations for how many environment profiles can be created. However, there are some practical guidelines to indicate what is needed.

Typically, the set of users who can deploy patterns for a production environment is smaller than the set of deployers for a testing environment, which is smaller than the number for development. Likewise, the number of profiles that are needed for each runtime environment tends to decrease from development through production.

The following profiles examples each have a corresponding user group:

- Production application environment profiles

  Used to deploy applications into the production environment, with priority set to `golden` and environment that is set to `production`.

  Separated profiles for each production application, department, or line-of-business deploying production applications can be used. This configuration enables control over who can deploy patterns for each application and allocate resources differently for different applications or groups of applications.

- Test application environment profiles

  Used to deploy applications into the testing environment, with priority set to `silver` and environment that is set to `test`.

  Create one profile per team that is deploying one or more applications to be tested. The use of a separate profile for each team and assigning each profile separate resources, such as a separate IP group, helps keep that team's applications isolated within the cloud group. It also prevents one team from using up too much of the cloud group's resources so that not enough remains available for the other teams.

- Development application environment profiles

  Used to deploy applications into the development environment, with priority set to `bronze` and environment that is set to `development`.

  All developers can share one profile, but each must be trusted not to use too many resources. To enforce these limits, use a separate profile for each development team. A profile is useful only if it has settings that are different from other profiles, such as different settings for who can deploy patterns, what IP groups to use, or to enforce limits on resources such as CPU.

3.7.4 Multiple lines of businesses

As mentioned in 3.7.3, “Application environments” on page 103, another approach might be to isolate applications from different lines of business in a production environment. Because these applications are used for different business purposes, they should be isolated from each other.

Consider a use case in which an IBM PureApplication System in an enterprise is shared by the following departments:

- Marketing applications
- Human Resources applications
- Finance applications

The enterprise wants to divide the computing resources into a set for each of the following departments for resource isolation consideration:

- Marketing
- Human Resources
- Finance

To address the computing resource isolation and access control requirements, IBM PureApplication System provides the capabilities to create three cloud groups and environment profiles. Each department is granted access to its own separate environment profile. The three cloud groups provide physical isolation of the computing nodes among the departments. IP groups with distinct VLANs are used to provide virtual network isolation among the departments.

On a small configuration with six compute nodes, you can assign each cloud group two compute nodes with a different unused VLAN ID. You also can give each cloud group one or more IP groups.

The IP groups for a cloud group can have the same VLAN IDs, but the IP groups for different cloud groups need different VLAN IDs to isolate the cloud groups' network traffic from each other. Assuming that the applications are used heavily, set each cloud group's type to `dedicated`. Otherwise, if a cloud group has a surplus of applications that are used sparingly, set its type to `average`.

Create an environment profile for each production environment and assign it a user group whose users are responsible for deploying to that environment. The profiles deploy their patterns to the same cloud group, but they allocate resources with limits to help better isolate the teams' applications.

3.7.5 Resource isolation by using a trust domain

IBM PureApplication System uses management software within a management trust domain that contains a trust service to issue verifiable security tokens to virtual application patterns trust domains. A virtual application pattern deployment model organizes resources into trust domains to provide management and infrastructure isolation.

When deployed, each virtual application creates a separate deployment trust domain. These domains have a unique set of deployment security tokens and owner keys. Its resources can communicate only with other resources in the same trust domain. So, although several virtual application instances can be deployed in the same cloud group, with network visibility, they cannot communicate with one another. However, there is an exception; they can communicate if a trust relationship is explicitly established between the two trust domains.

Trust relationships can be established between deployment trust domains to allow communication across virtual application trust domains. This configuration is how shared services deployments establish trust relationships with other virtual application deployments. This process enables those applications to use the shared services.

Figure 3-30 shows an example of a cloud group that contains three virtual application deployments. Each deployment is in its own trust domain, which provides isolation from other deployments in the same cloud group.

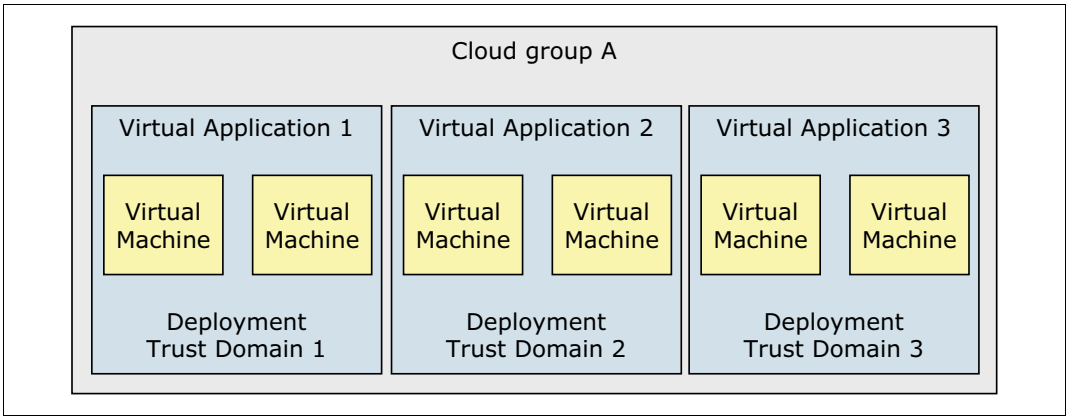


Figure 3-30 Deployments isolation by trust domains

Isolation through trust domain provides another level of resource isolation for virtual application pattern deployment. Isolation through trust domain extends network isolation. The level of isolation that is provided for security tokens and keys helps the system with security. It helps by preventing security attacks from spreading from one application deployment to other deployments, and to the system management trust domain in the event of a deployment compromise.

3.7.6 Isolation for shared resources

As noted in 3.5, “Managing shared services”, shared services can have only one instance of a type that is deployed in a cloud group.

Therefore, isolation of shared services is handled at the IP group level. This configuration means that although a specific instance of a deployed shared service is isolated between different IP groups, it is still shared between all patterns that are deployed into the same Cloud group as the shared service instance.

Different Environment profiles can use the same Cloud groups as deployment targets. This ability might indicate that patterns that are deployed by using different environment profiles would still share resources that are deployed to a common cloud group.

To create a deeper isolation, you can ensure that a specific cloud group is not shared among different environment profiles. In our previous example of isolation per lines of businesses, having a one-to-one relationship is beneficial. The one-to-one relationship was between the marketing cloud group and the marketing environment profile, which further helped isolate the shared resources between only marketing applications and patterns.

4

PureApplication Deployment Models

This chapter describes the deployment models that are available in IBM PureApplication System. It explains their main features, topologies, and the tools of each model. This chapter also describes implementation strategies so you can determine the best model for your application migration or create a new one. The main offerings of PureSystems Centre also are covered in the final section.

The following topics are covered in this chapter:

- Introduction
- Trade-offs between control against total cost of ownership and total time to value
- Virtual Appliances
- Virtual Images
- Virtual Systems
- Virtual applications
- Implementation strategy
- PureSystems Centre

© Copyright IBM Corp. 2013. All rights reserved.

4.1 Introduction

Before IBM PureApplication System deployment models are described, it is important to introduce some initial concepts that are behind the PureApplication System architecture principles, as shown in Figure 4-1.

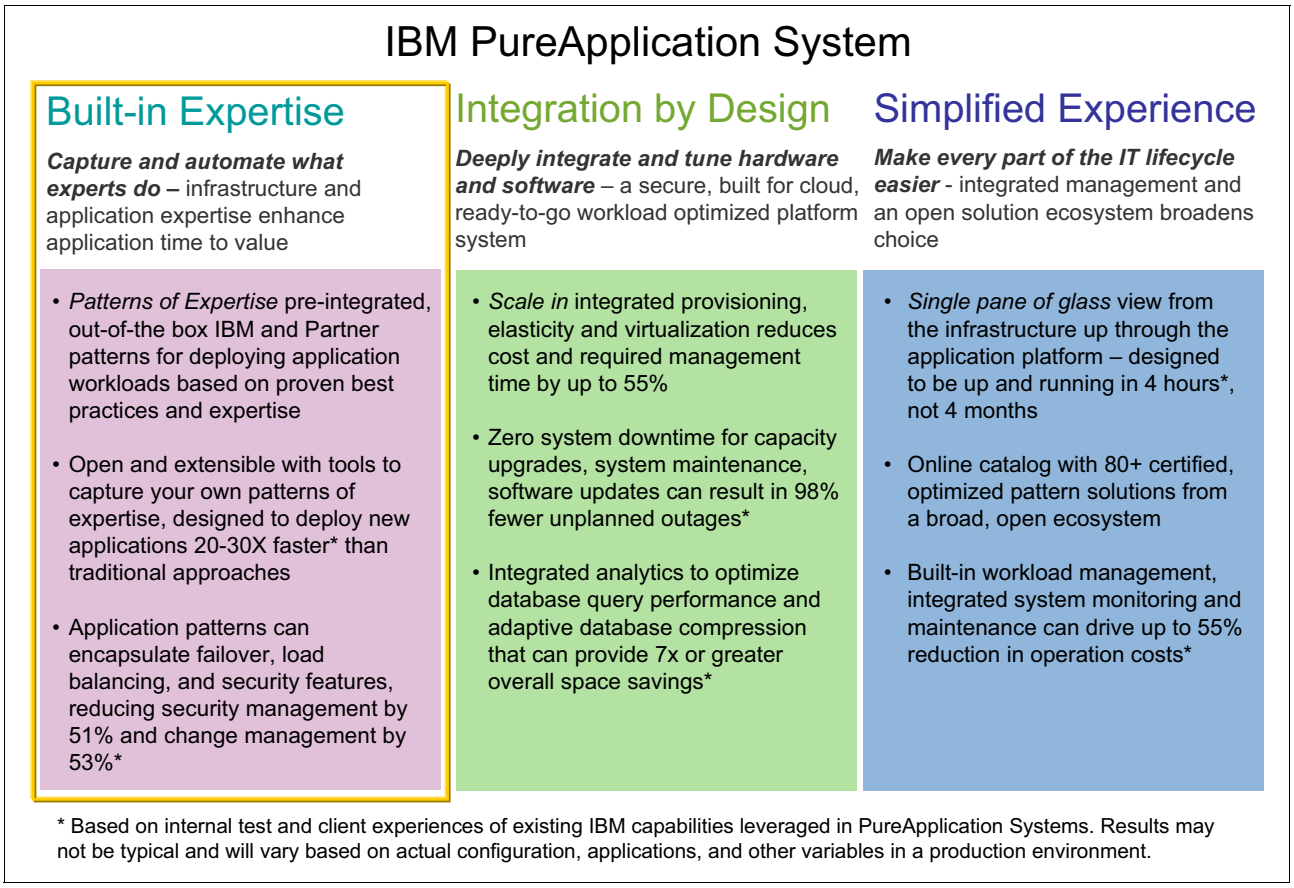


Figure 4-1 IBM PureApplication System architectural principles

IBM PureApplication System architecture is based on the following principles:

- Built-in expertise: Capture and automate what experts do when infrastructure and application expertise enhance application time to value.
- Integration by design: Deeply integrate and tune hardware and software.
- Simplified experience: Make every part of IT lifecycle easier by using an integrated management and an open solution ecosystem to broaden choices.

This section focuses on the Built-in expertise feature. One of the concepts within this feature is the *Patterns of expertise* that are proven best practices. These patterns are a collection of expertise that is gathered from solving complex tasks. This expertise was developed over decades of captured client and partner engagements, lab tested, and optimized into a deployable form.

A pattern is a model of deployment. IBM provides predefined deployment models with settings that are based on years of experience. These models include the following features:

- Pre-defined architecture of an application
- Each component of the application (such as the database or web server) has the following features:
  - Pre-installation of an operating system
  - Pre-installation across components
  - Pre-configured and tuned
  - Pre-configured monitoring
  - Pre-configured security
  - Lifecycle management

These features are packaged in a deployable form. This design results in a repeatable deployment with full lifecycle management and delivers the following results:

- Agility: Faster time to value
- Efficiency: Reduced costs and resources
- Simplicity: Simpler skills requirements
- Control: Lower risks and errors

There are three main pattern types with distinct IT domains, as shown in Figure 4-2.

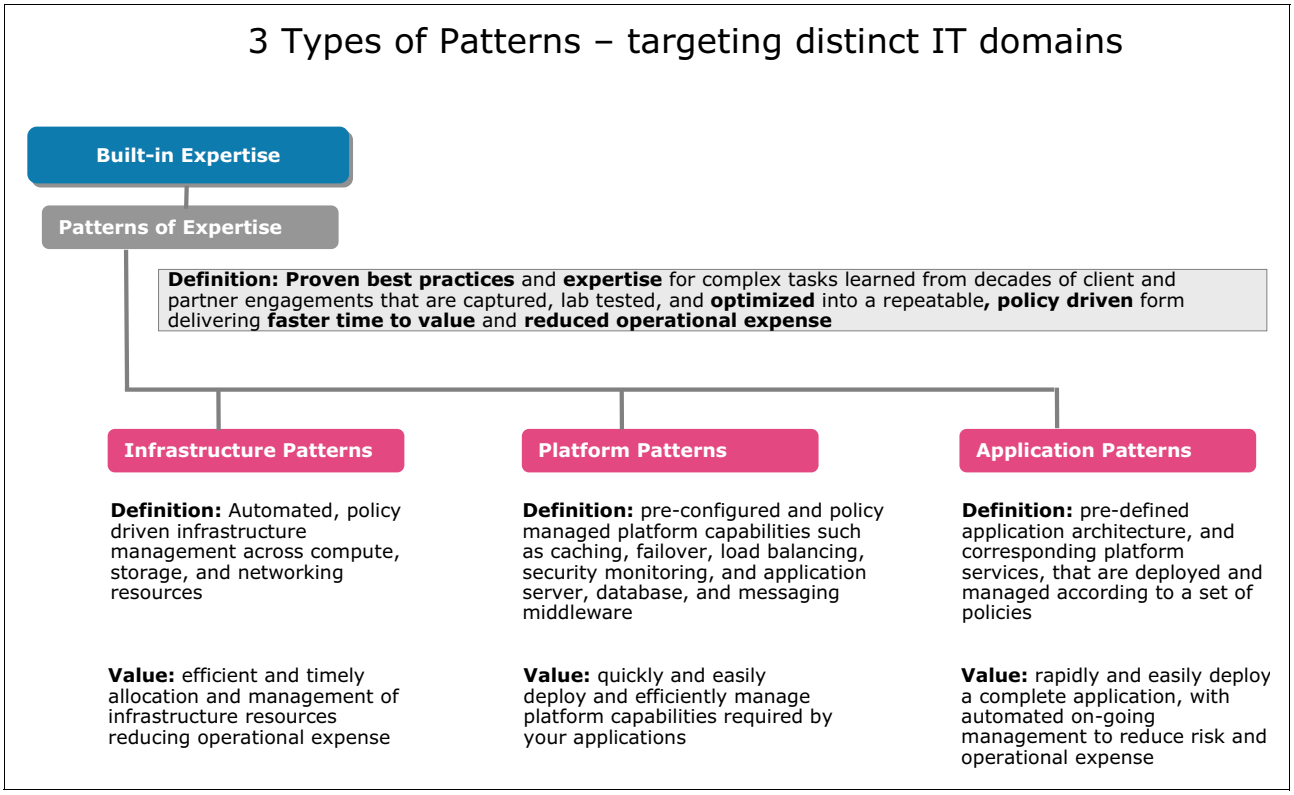


Figure 4-2 Three types of patterns

The following main pattern types are available:

- Infrastructure Patterns

  An automated, policy-driven infrastructure management approach across compute, storage, and networking resources. According to customer feedback, 60 - 70% of IT expenses are wasted in infrastructure management. Infrastructure patterns reduce operational expense by using an intelligent resource allocation and management.

- Platform Patterns

  A pre-configured and policy-managed platform. This platform's capabilities, such as caching, failover, load balancing, and security monitoring are combined with an application server, database, and messaging middleware. Applications require fast deploy and must efficiently manage platform capabilities to respond to business agility needs. Examples of this type of pattern are Web Application, DB2, and Business Process Manager (BPM).

- Application Patterns

  A predefined application architecture and corresponding platform services that are deployed and managed according to a set of policies. The value that is gained is the ability to rapidly and easily deploy a complete application, which reduces risks. One example of this pattern type is the SAP CRM pattern that provides specific set of policies that, when configured in advance, save time and costs and can be deployed into the cloud at any time.

IBM PureApplication System provides different patterns or deployment models according to business needs. As shown in Figure 4-3, there are three deployment model types.

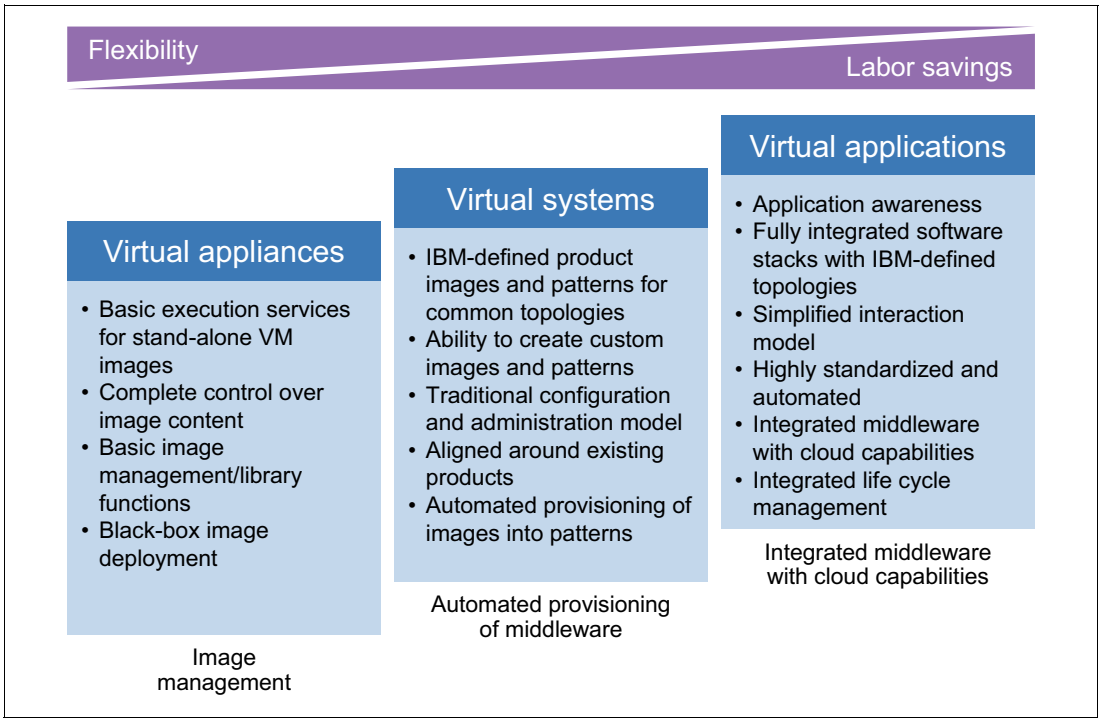


Figure 4-3 PureApplication System deployment models

Virtual appliances are the most generic type of deployment. They can be used to deploy any generic Open Virtualization Format (OVF) file to the IBM PureApplication System catalog and into a cloud. Though you have complete control over the image content, IBM PureApplication System is unaware of the internal features of this image. From the PureApplication perspective, the image is a black box.

The PureApplication System provides basic execution services such as stopping and starting the virtual machines where your image is deployed. It can be observed that you have more flexibility to build product images; however, it uses a few capabilities of IBM PureApplication System and it can be an intense labor activity that increases resource time, effort, and costs. An example of this approach is COBOL images that must be deployed by using this model. The PureApplication System also provides basic image management functions from the image catalog in the workload console for virtual appliances.

For business use, there are other styles of deployment models that are optimized for labor savings. These deployment models are virtual systems and virtual applications.

4.1.1 Virtual systems

The virtual systems deployment model uses IBM's hypervisor edition images. The hypervisor edition images are a set of virtual images that use VMware ESX hypervisor technologies with preinstalled middleware. By using this feature, you can define your topology as a pattern, customize it with script packages and other add-ons that represents your application customization, and deploy the designed pattern into the cloud. This model provides rich customization capabilities and allows fast, automated provisioning of IBM middleware that is based on the specific topology that you defined. You define the virtual machine images, the software components that are installed on them, the script packages that run to configure them, and any monitoring agents that you want them to include.

For example, you can define a virtual system pattern as a multi-node WebSphere Application Server topology that contains a deployment manager, one or more custom nodes, and an IBM HTTP Server. After you have that topology defined as a pattern, PureApplication System provisions that pattern for you with minimal effort on your part. That pattern can then be deployed multiple times with the same results each time as the deployment is fully automated.

IBM also provides a set of patterns that follow proven best practices that are based on years of experience that guarantees high reliability. You can use those patterns that are available from the product catalog or use them as a starting point for your own patterns, customizing them as needed. You also can create your own custom images to be deployed as virtual systems by using the IBM Image Construction and Composition Tool. This tool is useful for the cases where you need an image that contains middleware where IBM does not provide a hypervisor edition image and for vendor software you want to deploy.

The virtual systems deployment model is middleware-centric and does require you to configure the middleware. This model is designed to provide automated middleware provisioning. Virtual systems and virtual appliances have flexibility but differ in that virtual systems provide more labor savings.

4.1.2 Virtual applications

While virtual system patterns focus on the topology, virtual application patterns (as the name implies) take an application-centric approach. With virtual system patterns, you describe a middleware topology and IBM PureApplication System builds that topology in the cloud. With virtual application patterns, you describe an application and IBM PureApplication System builds the appropriate infrastructure and deploys the application to it. IBM PureApplication System then manages the lifecycle of the application, including growing and shrinking the resources that are needed to satisfy specified service levels. The virtual applications model is fully automated and is similar to the virtual systems deployment model but adds integrated lifecycle management. IBM PureApplication System includes a set of preinstalled web applications, database patterns, and Java patterns. You can also create your own patterns from your own design or by using a supplied pattern as a template. In system management scope, IBM PureApplication System manages the middleware that is needed to run those applications and determines the topology that is based on your artifacts and any policies you specify.

By using the virtual applications model, you have less flexibility than the use of virtual appliances or virtual systems. However, you can deploy your application quickly and easily. More information about these models and best practices is available in the rest of the chapter.

4.2 Trade-offs between control against total cost of ownership and total time to value

This section provides a comparison between deployment models from the business perspective in terms of total cost of ownership (TCO) and total time to value (TTV) business variables. Deployment models and more information about these variables are shown in Figure 4-4.

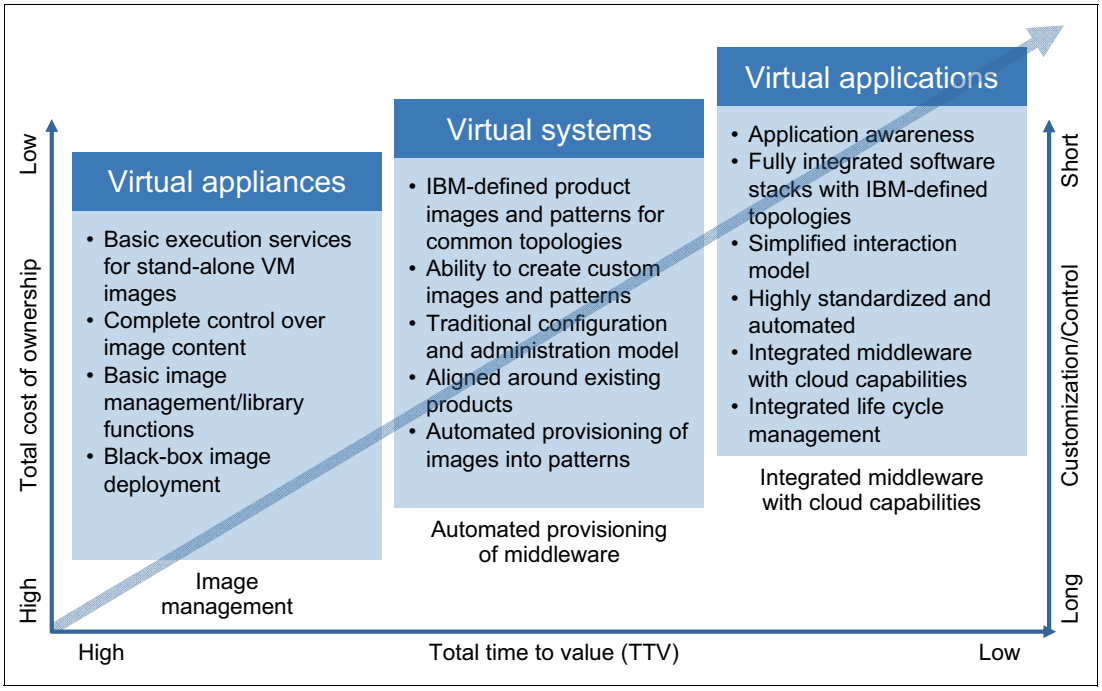


Figure 4-4 Deployment models comparison between TCO and TTV

Figure 4-4 shows that customization and control are high by using the virtual appliances model. However, there are significant increases in terms of TCO and TTV. The following factors contribute to the increases in resource costs when the virtual appliance model is used:

- Responsibility for standard software installation of middleware, applications, operational system level configuration, and image creation for deployment.
- Responsibility for all infrastructure updates to the image. IBM PureApplication System is unaware of what the image contains and runs only steady state activities, such as stop, start, and recycle. You have total control.
- Specialized team is needed to maintain the solution.

The virtual appliances model can be a useful solution, though the model does not use the potential of IBM PureApplication System and therefore reflects an increase in costs and time. When IBM PureApplication System does not provide the level of customization that is needed and you must move to a cloud environment as quickly as possible, the virtual appliances model is an option. As an example, with little effort you can package an existing, matured, market-tested COBOL application as a single-image virtual appliance, which immediately becomes deployable into the cloud.

Moving to virtual systems, you can save time and ownership cost by relying on IBM pre-built hypervisor edition images. These pre-built images support many product capabilities, such as WebSphere, IBM HTTP Server, and DB2. You still can customize your topology deployments through image extension, define the specific topology and middleware levels for your application, or use script packages to customize specific components as needed. Examples of these options include a WebSphere Application Server Cluster pattern that contains IBM Deployment Manager, one or more custom nodes, IBM HTTP Server, and configuration scripts for installing applications to the topology.

Administration scope is business as usual with the products you deploy. For example, with WebSphere Application Server, you can use the administration console or **`wsadmin`** command and the capabilities that are available to you in the PureApplication System administration console. This configuration allows for quick deployment of highly customized middleware, which reduces TTV. The cost of ownership also is reduced considerably when you rely on IBM to maintain the product images.

Looking at virtual applications, as shown in Figure 4-24 on page 142, you reduce the TCO and TTV even further because the solutions are pre-built and integrated for a specific use case. Core components of the pattern include web applications, databases, queues, connections to existing resources, business process models, batch jobs, and mediations. Core policies of the pattern include high availability, service level agreements (SLAs), security, multitenancy, and isolation.

Instead of defining topologies, you provide your application artifacts, and PureApplication System determines the appropriate underlying topology that is based on the SLA that you provide. While the use of the virtual application deployment model is the most cost-effective option, its high level of standardization and cross-product integration results in fewer product configuration options that are exposed for customization. For instance, with WebSphere Application Server deployments, you do not have access to the administrative console. Instead, you have a limited set of customizations that are available to you through the PureApplication System workload console.

4.3 Virtual Appliances

Virtual appliances are a key component of the cloud deployment model. A virtual appliance is a prepackaged software stack that combines the operating system, middleware, and applications in one package. Virtual appliances facilitate a quicker transition to cloud and require much less installation and configuration than traditional deployment methods. Virtual appliances address key issues to cloud computing, software licensing, and standardization, and it applies to traditional independent software vendors (ISVs) and software as a service (SaaS) providers.

Virtual Appliances include the following main features:

- You can create a virtualized environment for middleware that is not provided by IBM (for example, create a new virtual image for the Tomcat Server).

  You can easily host other software packages on the same shared cloud resources to which you deploy IBM middleware.
- You can extend an existing image that is provided by IBM to add software to the image.
- You can extend the use of the virtual appliance beyond what is provided in PureApplication System by adding a custom image to your virtual system pattern.

You often define templates and assemble parts to configure a system to deploy and then generate a file from that data. By using PureApplication System virtual appliances, you skip the creation steps and instead begin with a defined file. You can deploy that file repeatedly and create multiple virtual appliance instances from a single virtual appliance. You can supply some override values for properties in your deployments.

You also can add Virtual appliances to the catalog and deploy them directly into the VMware ESX cloud, as shown in the following examples:

- You can take an existing matured, market-tested COBOL application and with little effort, package it as a single-image virtual appliance, which immediately becomes deployable into the cloud.
- You can package a newly implemented, highly distributed, service-oriented application and integrate with the services that are provided by the cloud. This ability allows ISVs to respond more rapidly to their customer's changing business needs with flexibility and agility.

For more information about administering virtual appliances, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/systemconsole/t_mngvirtappliance.html

To create an image, you can use the Image Construction and Composition Tool. This tool is available for download from the PureApplication System on the Welcome page of the workload console. For more information, see Chapter 5.3, “Build virtual images by using the IBM Image Construction and Composition Tool” on page 213.

4.4 Virtual Images

Virtual images that are used by IBM PureApplication System are Open Virtualization Format (OVF) compliant images with special activation logic to help in deployment. There are a growing number of these virtual images for IBM Software products, which are named Hypervisor Editions.

The most fundamental of building blocks for virtual system patterns are parts that are delivered with hypervisor edition images. The following sections describe the content (hypervisor edition images with composing elements, and the parts that are used to deliver those elements and images). For more information about virtual system patterns, see 4.5, “Virtual Systems” on page 121.

4.4.1 Hypervisor edition images

A hypervisor edition image is the delivery of some middleware product that is packaged according to the OVF in an Open Virtualization Archive (OVA) file. These images are imported into a virtual image catalog within IBM PureApplication System.

A hypervisor edition image consists of some middleware product (such as WebSphere Application Server) that is preinstalled and pre-configured with an operating system (often Linux or AIX), and is designed for virtual environments. As an example, for the WebSphere Application Server, the following the virtual image includes the following features:

- An operating system
- WebSphere Application Server
- IBM HTTP Server binary files
- WebSphere Application Server profiles
- A combination of code and tuning that is built into the image to optimize the server for a virtual environment

To view virtual images on the IBM PureApplication System workload console, select **Catalog** → **Virtual Images,** as shown in Figure 4-5.

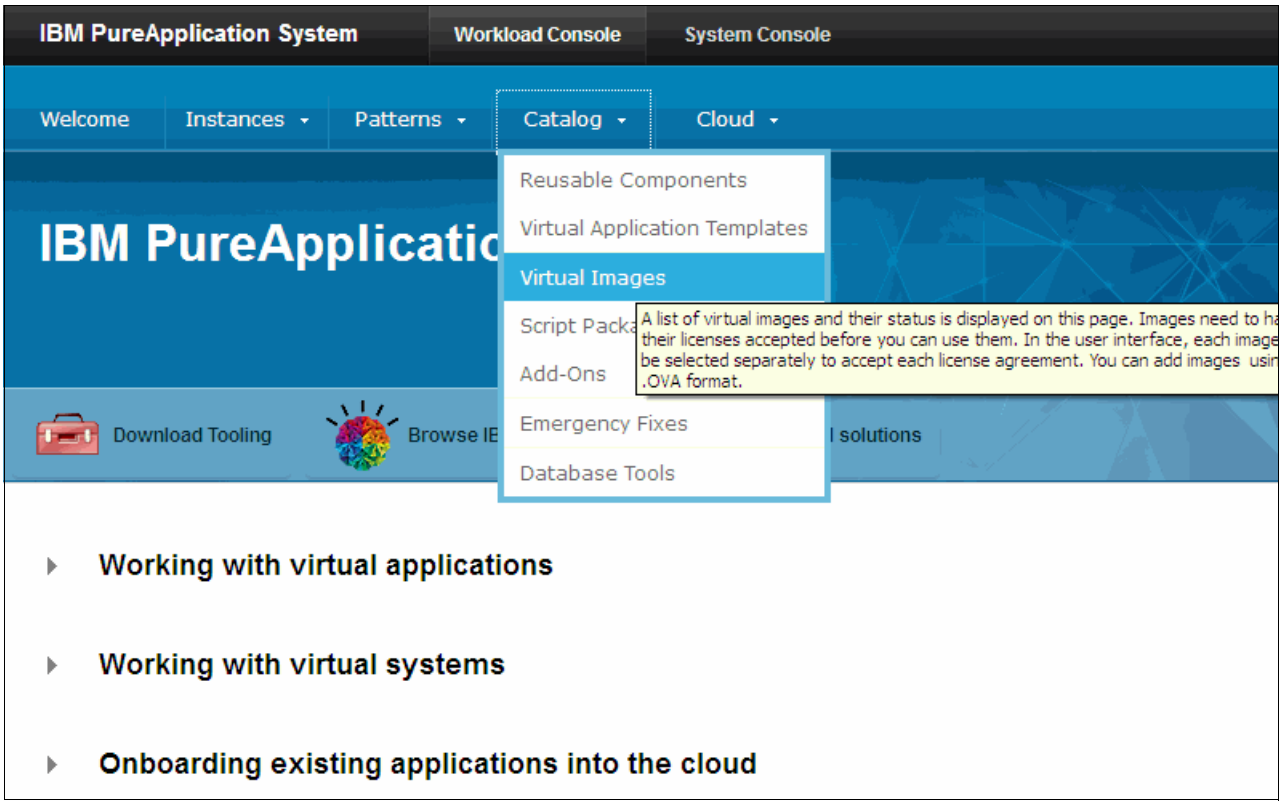


Figure 4-5 Browse to virtual images

Selecting this option displays the virtual images in the IBM PureApplication System catalog, as show in Figure 4-6.

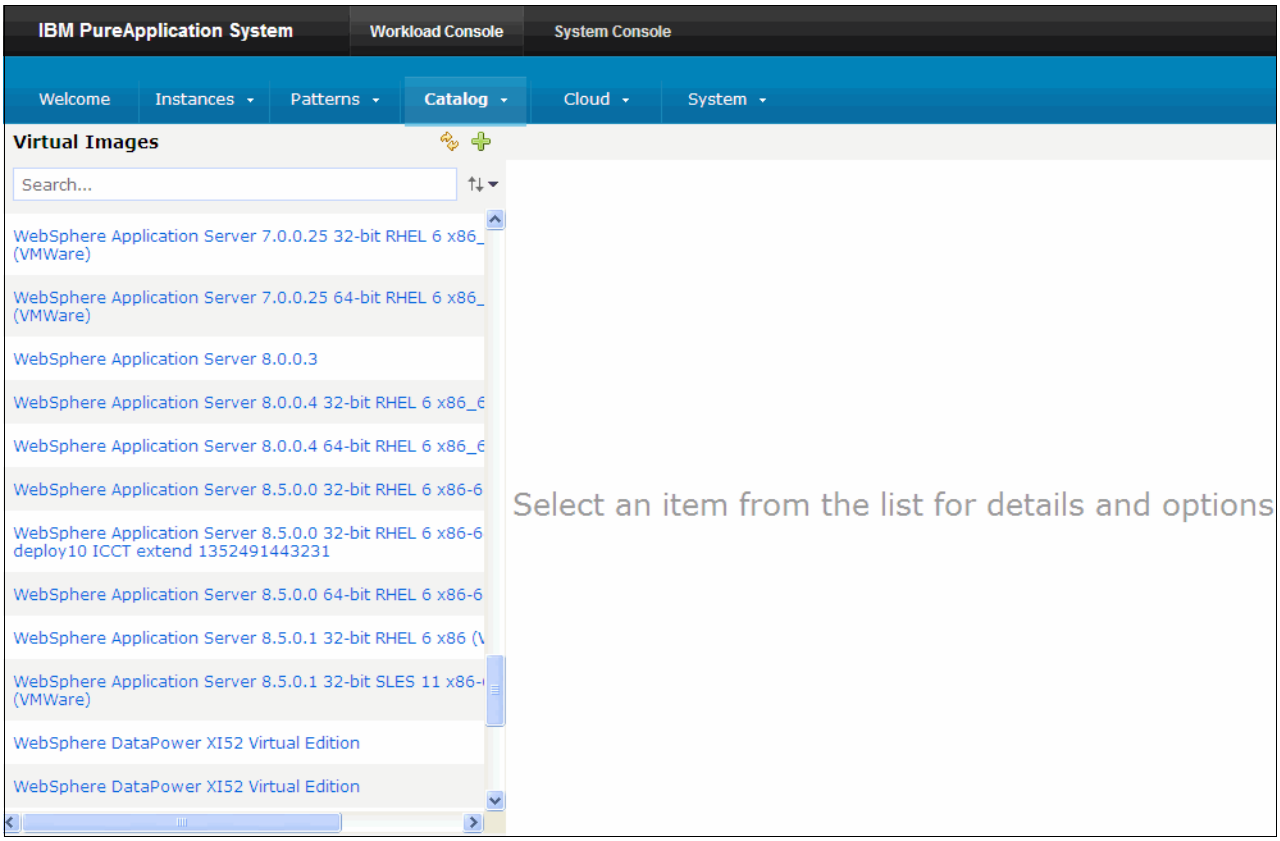


Figure 4-6 An example of loaded Virtual Images

The following preinstalled virtual images for the IBM PureApplication System offering are available:

- IBM WebSphere Application Server Hypervisor Edition virtual images

  A set of IBM WebSphere Application Server Hypervisor Edition virtual images for VMware ESX hypervisor technologies.

- IBM OS image for Red Hat Linux Systems

  IBM OS image for Red Hat Linux Systems in the system catalog. They provide the operating environment on which workload patterns run, including the operating system and infrastructure that is unique to the product.

- DB2 Enterprise virtual image

  IBM DB2 Enterprise 9.7 Fix Pack 6 and 10.1 in the system catalog.

Figure 4-7 shows the Hypervisor edition images.

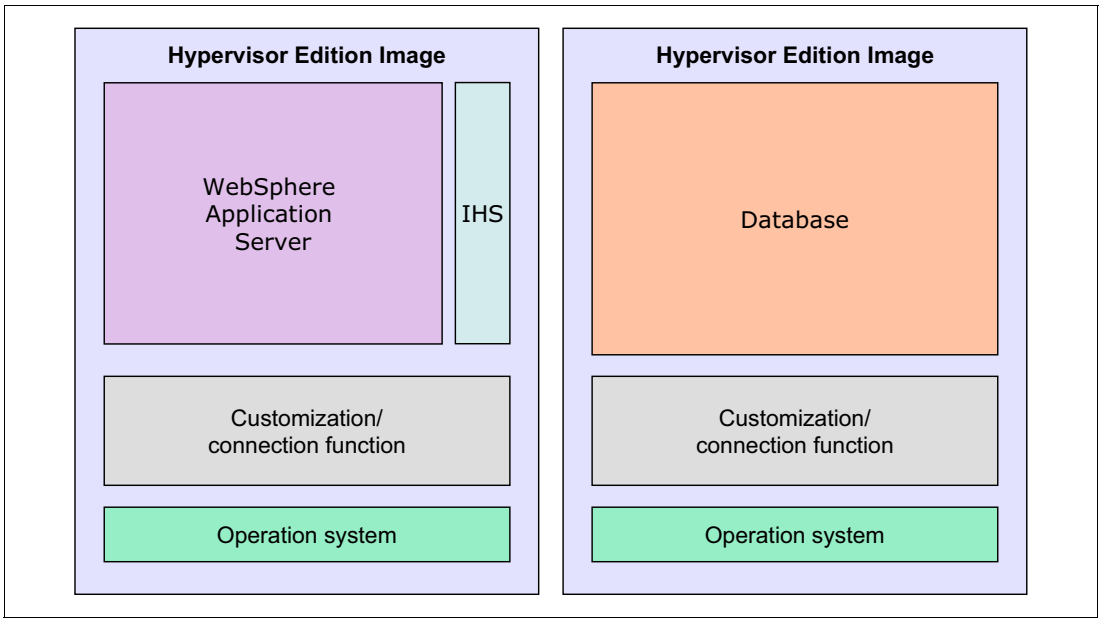


Figure 4-7 Overview of a Hypervisor Edition Image

For more information about virtual images, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/pct_manage_cat_vi.html

Hypervisor edition image elements

The hypervisor edition image features the following main elements:

- Preinstalled and pre-configured image
- Image-specific tuning
- Fast deploy-time activation capabilities

All of the elements, operating system, middleware, middleware dependencies and feature packs, and necessary maintenance for all elements are preinstalled into the image. You do not need to install the middleware, an operating system, or develop a script to perform any installation task. The process is handled automatically by using the hypervisor edition image.

Because IBM is preinstalling middleware and the underlying operating system, the image is tuned for best practices and optimal performance in a virtual environment. It is fast when deploying images because the installation and optimization is done already. All that is necessary to do during deployment is to refine the configuration and run some activation logic. Maintenance also is simplified because it is available as fully installed images for the complete solution.

4.4.2 Parts of hypervisor edition image

The elements of the middleware are delivered in the hypervisor image as parts. For example, the WebSphere Application Server Hypervisor Edition image includes parts for the deployment manager, custom node, stand-alone node, and job manager. Having these common profiles pre-configured in the image again saves significant deployment time when compared with traditional deployment processes where profile creation is done later by scripting.

Detailed middleware configuration and provisioning for specific purposes is handled by an activation agent. While the preinstallation, configuration, and tuning are strengths, you can consider activation the real power of the hypervisor edition image.

For WebSphere, the activation capabilities support having this one image transform into different WebSphere Application Server configurations when it is started. This capability enables one template image to be copied and quickly reconfigured for rapid provisioning of different WebSphere Application Server environments. This task is accomplished through an activation code that is included within the image that reads input parameters, maps these parameters to different pre-configured profiles, and performs reconfiguration tasks.

During activation, reconfiguration scripts inside the image complete the following tasks:

- Inject the new network settings for IP address, host name, passwords, and so on.
- Reconfigure WebSphere Application Server parameters for cell name, node name, and so on.
- Start the WebSphere Application Server profile corresponding to the server type.

Replacement or injection of the configuration metadata for the OS and WebSphere Application Server profiles provides a significant time savings. The activation enables an image to quickly assume and adjust for new network settings, passwords, and WebSphere Application Server personalities, from deployment managers to custom nodes and job managers.

Parts are the primary building block of any virtual system pattern. However, there are other fundamental pieces of a virtual system pattern that are necessary to support detailed customization; namely, script packages and add-ons.

Script packages

In IBM PureApplication System virtual system patterns, a script package is your vehicle to provide custom middleware configuration. This ability might mean installing applications, configuring application dependencies, or otherwise tuning the middleware layer.

Script packages are compressed files that include some executable files (shell script, wsadmin script, Java program, and so on) and optionally, artifacts that support the running of the script. There is not a singular, mandatory format for a script. You can reuse many of the same scripts that you were using in your traditional deployments.

As was the intention, you can achieve just about anything you want with a script package. By using a script package, you can be as flexible and creative as you must be. Scripts can be designed to accept input parameters at the time of deployment. This feature allows a common script to be applied for many purposes on many parts. Scripts are imported into the IBM PureApplication System script catalog and can then be associated with parts that are contained in virtual system patterns.

Add-ons

Add-ons are specialized scripts to customize the virtual machine configuration. By using add-ons, you can modify the virtual machine configuration during deployment without the need to modify and save a new image configuration. You can use add-ons to augment the hardware and OS configuration of a virtual machine.

Add-ons simplify the task of performing lower-level OS configuration changes. For example, with the `Add disk` add-on, you must drag only the add-on from the Pattern Editor palette to the appropriate part and then configure the parameters.

You use add-ons such as custom scripts. You create and clone them in the catalog as necessary and then drag them onto parts in the Virtual System Pattern Editor. The primary difference is that add-ons are run before any custom scripts and they target the virtual machine configuration.

However, while add-ons are like scripts, there are significant differences. First, add-ons are not listed with the custom scripts. They have their own category in the catalog. Add-ons are run at deployment time before any custom scripts that are associated with a part. Unlike custom scripts, you cannot specify the order of add-on runs on a part. Add-ons are run only during system creation; you cannot initiate them on demand. They use hypervisor-level APIs to configure new hardware in virtual machines during deployment.

4.5 Virtual Systems

Virtual systems that consist of one or more virtual images are a foundational deployment model of PureApplication System.

A virtual system is defined in PureApplication System through a *virtual system pattern*. A virtual system pattern is a provisionable unit of one or more virtual images to be installed, configured, and integrated together to implement a topology. Virtual system patterns can be as simple as a single server product instance or as complex as a multi-product, multi-node deployment. Several virtual system patterns, which are provided by IBM that uses best practice design experience, are preinstalled in the catalog. After a virtual system pattern is deployed, it is referred to as a *virtual system instance.*

Virtual system patterns can be customized or patterns can be created by using the PureApplication System workload console. Customization is achieved by using parts, script packages, and add-ons.

Virtual system pattern concepts are shown in Figure 4-8.

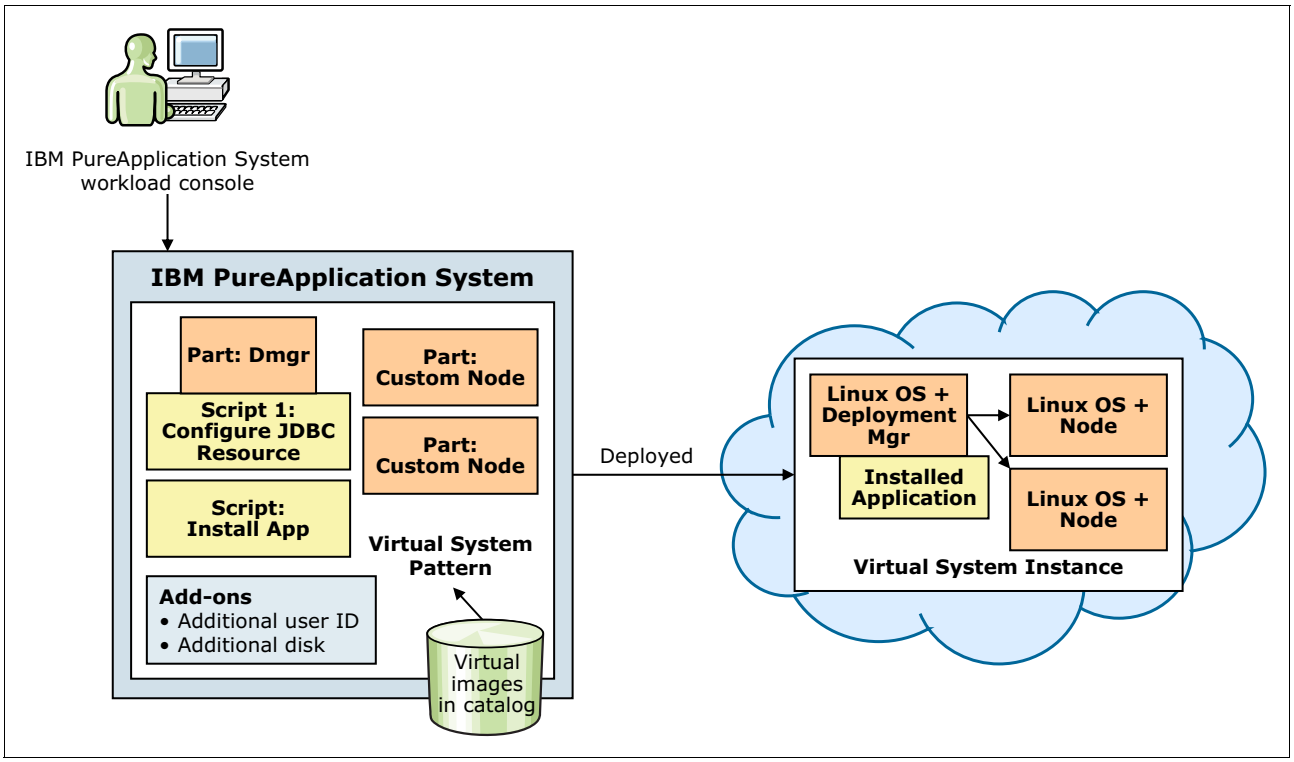


Figure 4-8 Virtual system pattern concepts

The IBM PureApplication System comes with a set of Hypervisor Edition virtual images in the virtual image catalog. These virtual images consist of parts that can be added to virtual system patterns. For example, the WebSphere Application Server virtual image consists of the following parts: administrative agents, custom nodes, deployment manager, IBM HTTP Server, job manager, stand-alone server, and on-demand routers. When you create a pattern, the parts of the virtual images in the catalog are available for you to add to the pattern.

Some patterns have advanced options, for example, a virtual system pattern that includes parts for WebSphere Application Server deployment manager and custom nodes provides advanced options. These options are used to define clusters, enable the default messaging provider, configure session persistence, and enable global security. By using patterns, you also can define the startup order for parts and script packages.

As described in “Script packages” on page 120, a script package is an archive (`.zip`) file that contains artifacts to be run and artifacts to be run upon. The code that is included in the script package can be as simple as a `.war` file or as complex as a complete product.

During deployment, script packages are transferred to the target virtual machines at a file location you specify in the configuration. After they transfer, they are extracted in that same location. When the virtual machines successfully start, script packages are then extracted and the scripts are run by using the supplied command line. The goal of the use of script packages is to customize your middleware environment beyond the customization provisions that are standard with IBM PureApplication System. A typical scenario might be to install a WebSphere Application Server application and configure the required JDBC resources into a server or cluster environment that is rendered by IBM PureApplication System. The product provides a catalog of script packages that perform customization tasks. You can clone and then tailor these packages for your use, or you can create script packages.

Add-ons, which are available for parts in the pattern, include the capability to add a new virtual disk to the virtual machine (formatted or unformatted). They also canbe used to add and configure a virtual network interface controller (NIC), and add another user ID to the virtual machine. As another and important feature, you also have access to the IBM Image Construction and Composition Tool to build customized virtual images. These images can then be deployed in a virtual system pattern, as shown in Figure 4-9.

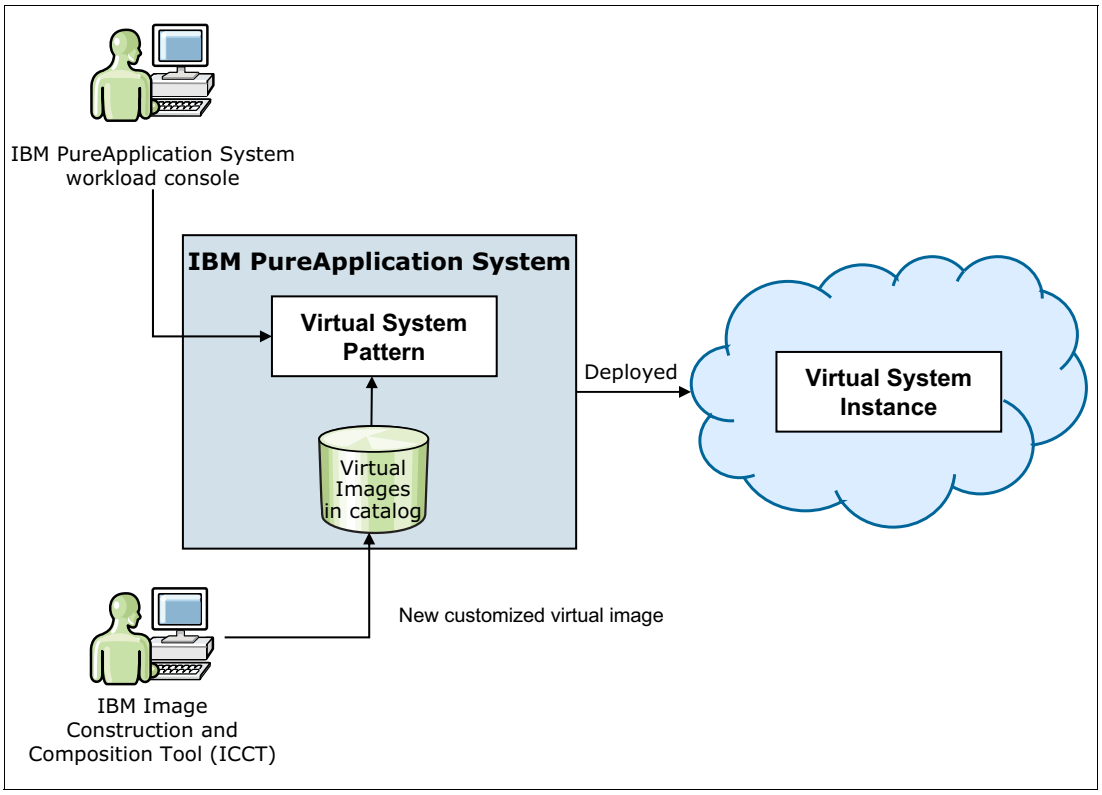


Figure 4-9 Image Construction and Composition Tool and PureApplication System integration

4.5.1 Virtual System Patterns

Virtual system patterns that are provided with the product from IBM represent hardened topologies of IBM middleware, which can be provisioned immediately.

To view virtual system patterns, from the PureApplication System workload console, select **Patterns** → **Virtual Systems**, as shown in Figure 4-10.

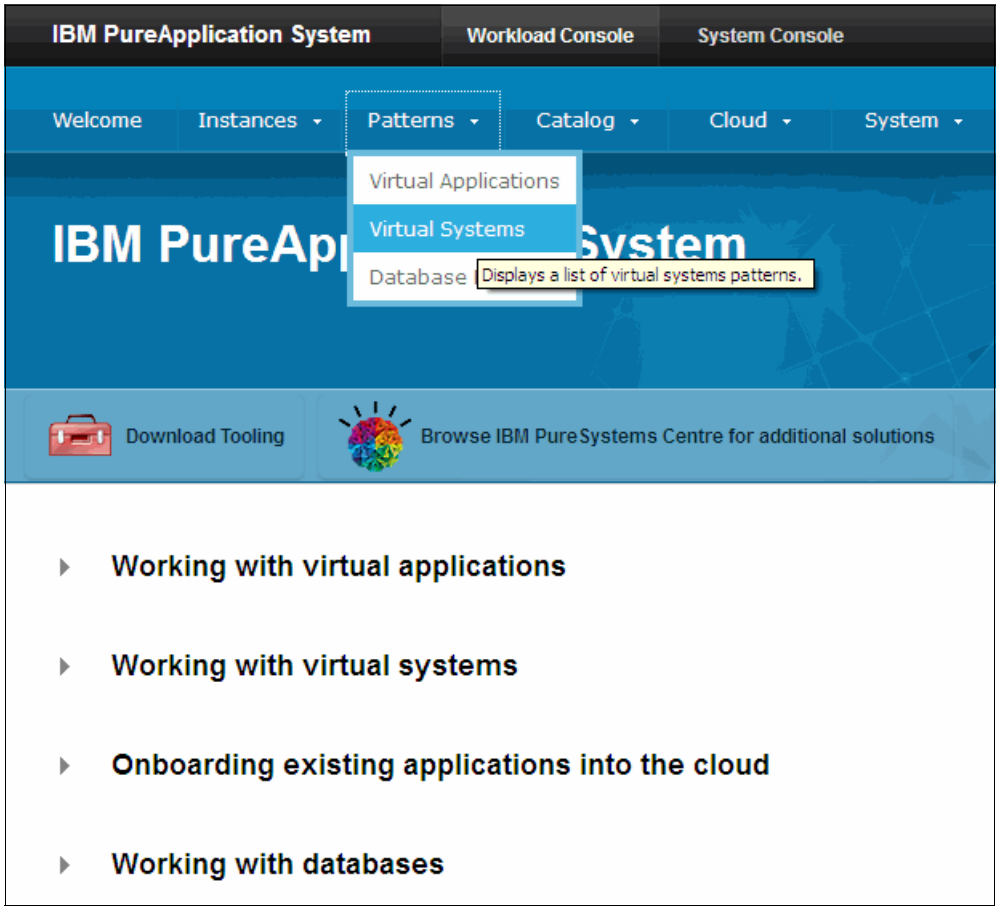


Figure 4-10 Browse to virtual systems

Creating the virtual system pattern

The virtual system pattern editor is an easy drag-and-drop interface that is used to create your virtual system topology. The initial pane is shown in Figure 4-11.

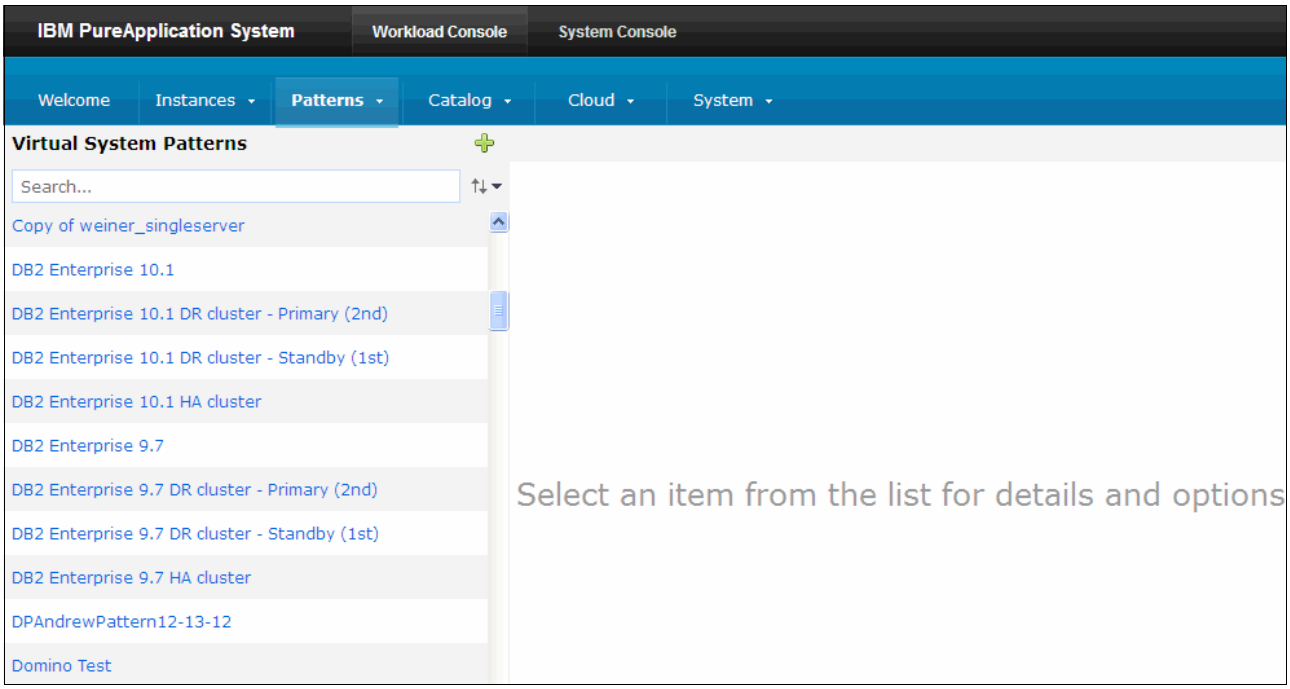


Figure 4-11 Virtual system patterns in the workload console

The Virtual Systems Patterns pane

When you select a virtual system pattern, the details about that pattern are shown in the workload console. A view of the topology for the pattern is displayed with the detailed information.

The topology for a virtual system pattern is described graphically for editing purposes. Virtual image parts, add-ons, and script packages can be dropped onto an editing canvas to create or change relationships between the parts that define the topology. All of these tasks are done in the Pattern Editor.

The Pattern Editor pane

Clicking the edit icon in the toolbar in the Virtual System Patterns pane opens the Pattern Editor for the selected virtual system pattern. The Virtual System Patterns pane provides lists to select virtual image parts, add-ons, and script packages. Figure 4-12 on page 126 shows the details of the selected virtual system pattern.

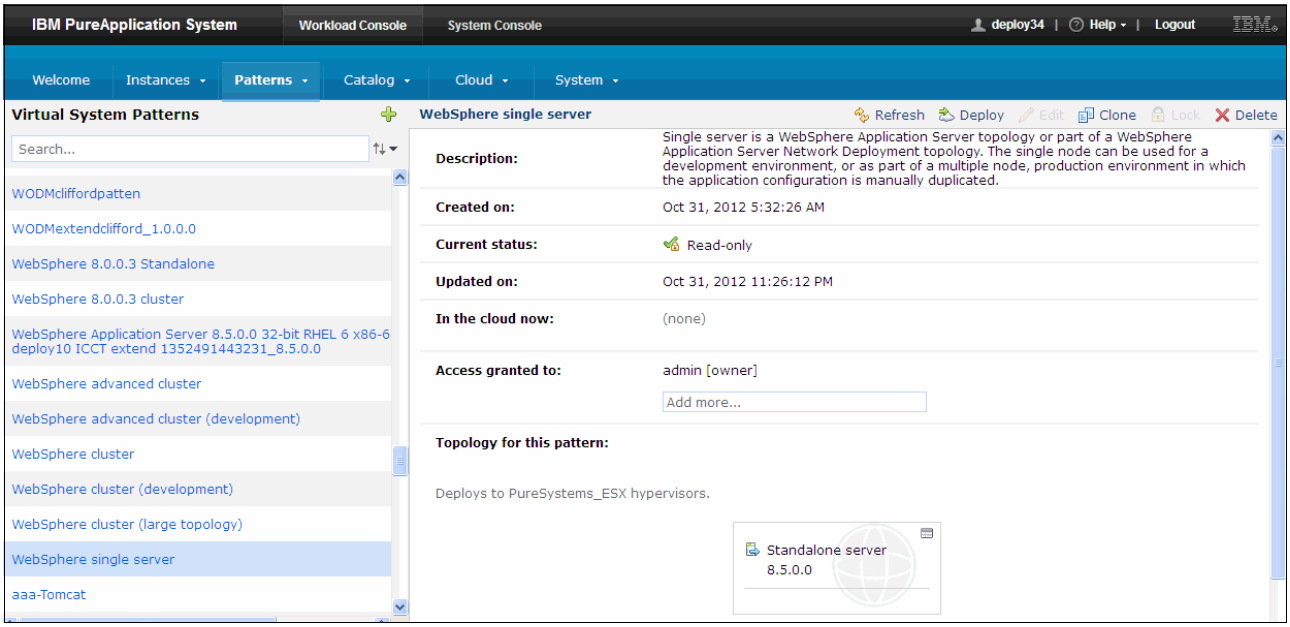


Figure 4-12 Virtual system pattern in Pattern Editor

Virtual image parts

Select **Parts list** in the Pattern Editor to see a listing of the parts that can be dropped onto the Virtual System pattern canvas. The Virtual System pattern canvas is on the right side panel of the Pattern pane. The following common virtual image parts are available:

- Administrative agents
- Custom nodes
- Deployment managers
- HTTP servers
- Job managers
- Stand-alone servers
- On-demand routers
- DB2 servers
- Others

The parts are determined by the virtual images that you are using. Some virtual image parts represent multiple nodes. There is an indicator on the part when you drop it onto the canvas that indicates the number of nodes of each part.

You can configure the properties of a selected part in the Pattern Editor or later when the pattern is deployed. To configure the part in the Pattern Editor, click the Properties icon in the part on the editing canvas. Selecting to lock a property prevents changes in that property during deployment.

Script packages

The Parts list on the Pattern Editor provides a listing of the script packages that can be dropped onto the virtual image parts. This list can contain script packages that are associated with the virtual image and any that you defined for use with IBM PureApplication System.

Add-ons

The following common add-ons can be added to parts on the editing canvas:

- Default add disk: Adds a virtual disk to the virtual machine and, optionally, formats and mounts the disk.
- Default add NIC: Adds a virtual NIC to the virtual machine, configures IP address information for the virtual NIC, and activates it.
- Default add user: Defines another user on the virtual machine.
- Default add raw disk: Adds a virtual disk to the virtual machine but does not format or mount the disk.

Customized versions of add-on types also can be created and made available to meet your particular needs and added to the catalog. An add-on can be created as a new add-on or cloned and modified from the default set.

Interaction between virtual image parts

Virtual image parts can be defined to interact with other virtual image parts. When the interacting virtual image parts are included in the same virtual system pattern, cross-configuration results. For example, when a custom node and a deployment manager are placed in the same virtual system pattern, they are automatically cross-configured. This configuration results in the custom node that is federated to the deployment manager. Similarly, administrative agents (or deployment managers) are registered with a job manager.

Virtual image parts can be cross-configured if the virtual system pattern editor can determine a unique relationship. If it is unable to do so, no cross-configuration occurs. For example, if a custom node is added to a virtual system pattern with two deployment managers, no federation takes place. However, if one of the deployment managers is later removed, cross-configuration occurs because a unique relationship now exists.

You can use the version indicator on the parts to ensure that they are referencing the same version of the virtual image in the catalog. If the version of a part is incorrect, you can change it when the part is on the Editing canvas. Hovering the cursor over the part name opens a window that has more information about the virtual image.

Preinstalled virtual system patterns

IBM PureApplication System ships with predefined virtual system patterns that represent best practices that are derived from years of experience in working with customers. These patterns represent common configurations from simple to advanced WebSphere environments and various DB2 configurations.

The predefined patterns might fit your needs exactly and you can deploy them without any changes. However, it is more likely that you want to clone and extend these patterns or create your own new custom patterns. For more information about how to create your own custom pattern, see Chapter 5, “Customizing Virtual System Patterns” on page 199.

An example of an application-ready topology that comes preinstalled on the IBM PureApplication System is the WebSphere advanced cluster virtual system pattern, as shown in Figure 4-13.

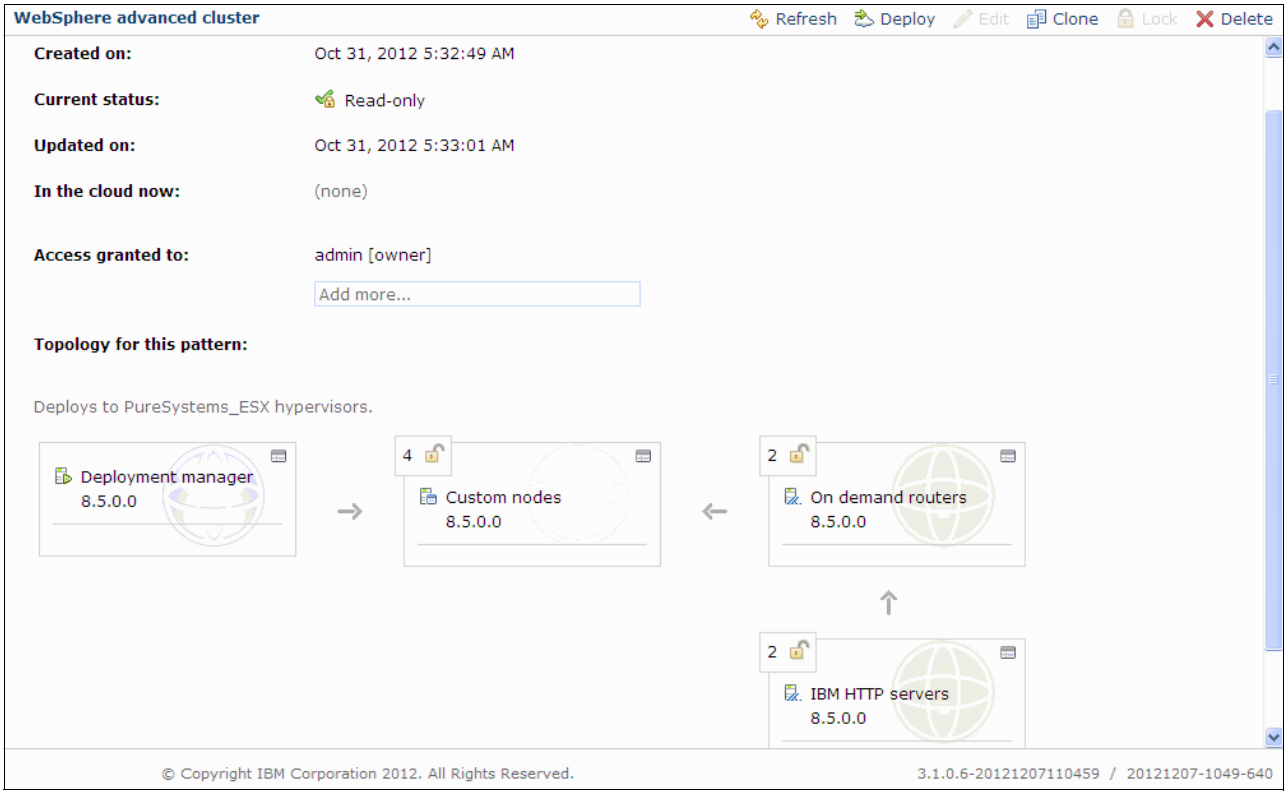


Figure 4-13 WebSphere advanced cluster

DB2 virtual system patterns

Like other applications that run on IBM PureApplication System, DB2 is available as a DB2 virtual system pattern or as part of a DB2 database workload pattern. The DB2 virtual system pattern allows for more flexibility in the control and configuration of the middleware environment. The following images are available that can be deployed as a DB2 virtual system pattern:

- DB2 Enterprise
- DB2 Express
- DB2 Enterprise (Primary Node for High Availability Disaster Recovery (HADR) feature)
- DB2 Enterprise (Secondary Node for HADR)
- DB2 Express (Primary Node for HADR)
- DB2 Express (Secondary Node for HADR)

4.5.2 Planning and designing your virtual system pattern

Virtual system patterns fully automate the deployment of complex applications and platforms while taking advantage of best practices. The most important technical role in virtual system pattern development is that of the application deployer.

The application deployer is the subject matter expert in the following areas:

- Identifying application prerequisites (hardware and software).
- Understanding the solution architecture from the perspective of high availability, scalability, failover, and fault tolerance.
- Applying best practices for application deployment and understanding the installation and configuration bottlenecks.
- Installing all components of the application.
- Scripting the installation of the application (by using shell, Jython, or DDL scripts).
- Administering prerequisite middleware and software products.
- Running basic functional tests on the application.

Ideally, the application deployer has enough installation, deployment, and configuration experience to identify the automation touch points of key manual tasks and to build into the pattern industry best practices. For example, if most customers or users run with a specific Java virtual machine (JVM) heap size in WebSphere, this setting should be built into the pattern.

Key pattern design concepts

You must consider the following concepts when a virtual system pattern is designed and developed:

- Elasticity
- Topology
- Orchestration
- Security

These concepts are described next.

Elasticity

Elasticity in a cloud environment involves automatic horizontal and vertical scaling of your application by using dynamic assignment of resources. In a virtual system pattern, WebSphere Application Server environments can be made elastic by using the Intelligent Management Pack (IMP) feature in IBM PureApplication System.

The IMP feature can grow or shrink a WebSphere Application Server cell in a virtual system pattern on demand. This elasticity is based on service level agreements or performance metrics that are described by policies. An example of how IMP achieves horizontal scaling is when it detects a workload spike in the WebSphere Application Server cell. The spike might exhaust all available current CPU capacity. To prevent this problem, the IMP feature automatically provisions a new WebSphere Application Server node to meet workload demand. Furthermore, IMP is flexible enough to implement vertical scaling when it is configured. To fulfill a response time SLA to prevent performance degradation, IMP can trigger the starting of new virtual machines in a WebSphere cluster.

If elasticity is a requirement for your application, consider the use of the IMP enhanced WebSphere Application Server environment on IBM PureApplication System.

Topology

If existing topological best practices were applied within your current environment, these best practices also are relevant to the virtual system pattern.

For example, if you use a clustered WebSphere Application Server setup with eight virtual machines and in-memory session replication as your best practice for production, the same configuration applies to the production trend of your virtual system pattern.

For a development or test environment of the virtual system pattern, you can choose a single server configuration and smaller heap sizes on JVMs.

As part of designing the virtual system pattern, it is helpful to create a diagram of the topology in which each product is listed (including the number of VMs per product) and the relationship between each VM is reflected. For example, if WebSphere Application Server must connect to a WebSphere MQ server, this communication should be reflected in the topology diagram, as shown in Figure 4-14.

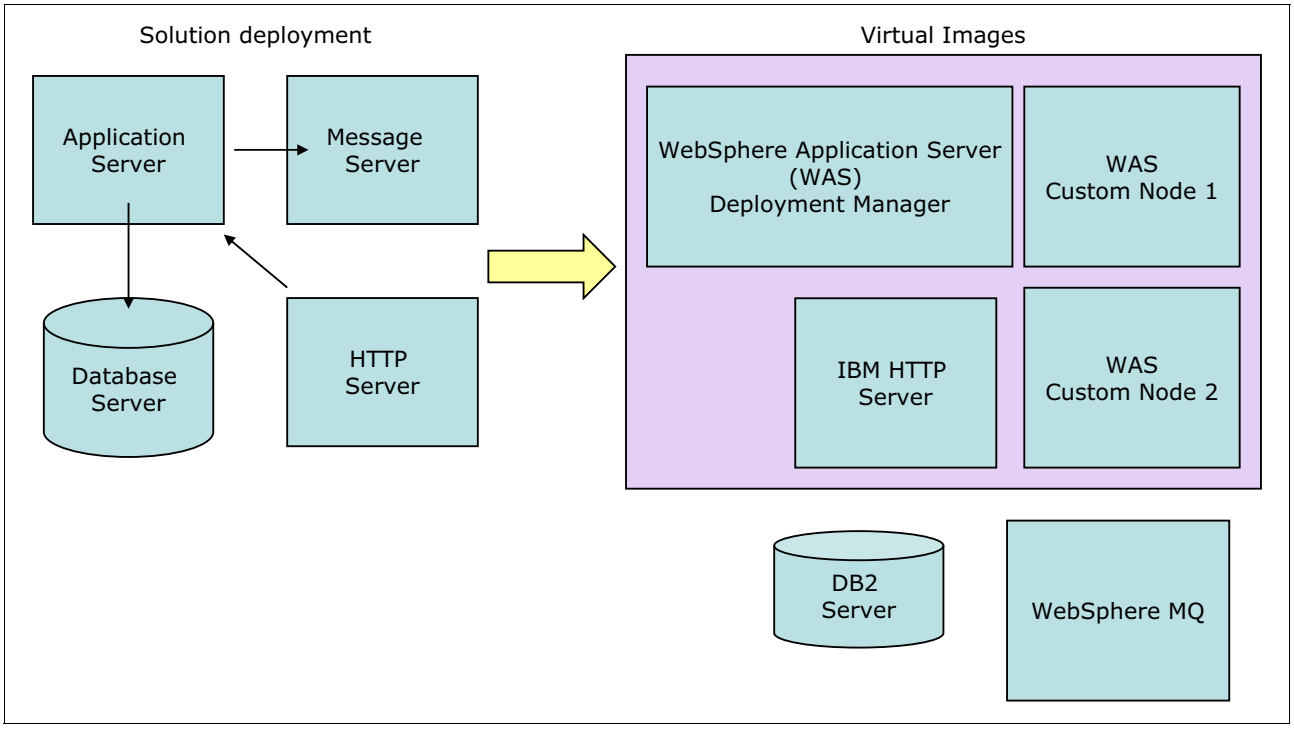


Figure 4-14 Topology best practice example

Orchestration

After a topology is identified for the virtual system pattern, the next logical step is to list the actions that are needed in each VM to orchestrate the startup of the system. The order of each action also should be determined.

For example, if your application's installation process requires that a database should be running with a schema in place, orchestrate the database setup before the application installation process is begun, as shown in Figure 4-15.

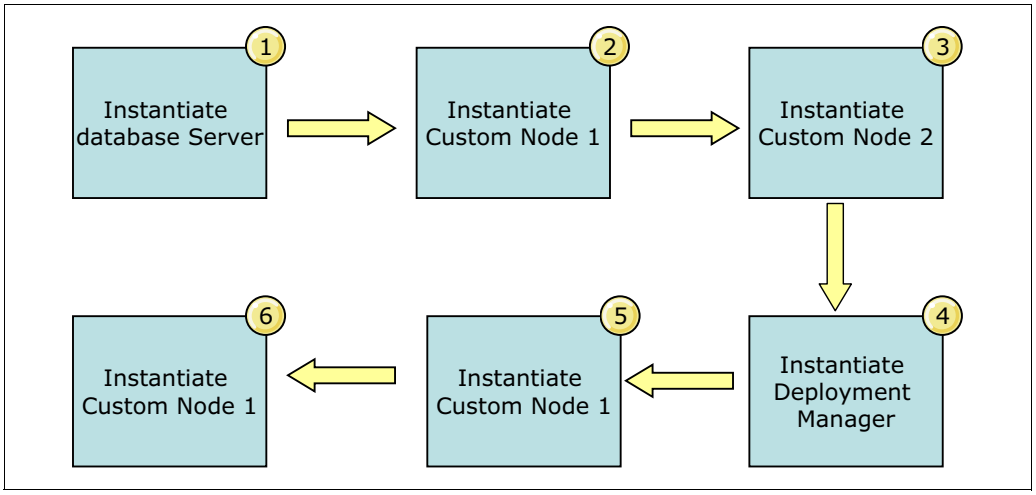


Figure 4-15 Orchestration approach example

To enable this kind of orchestration, by using virtual system patterns the designer can specify two orders: the order in which virtual machines are brought up, and the order in which automation scripts are run across the virtual machines.

Security

Lightweight Directory Access Protocol (LDAP) support is one of the security-related topics to consider when a virtual system pattern is designed. Applications often do not mandate dedicated LDAP servers. Most applications connect to an existing LDAP server (such as a corporate LDAP directory) for authorizing access to protected resources. In such a situation, an LDAP server component is not included in a virtual system pattern.

From a WebSphere Application Server perspective, connection to an existing LDAP server in a virtual system pattern can be captured by using a script package that takes LDAP server information (host, user, password, and so on) as input parameters. The script package automates the configuration of an LDAP connection in WebSphere Application Server by a Jython script. The script alleviates the need for you to perform this configuration manually.

If an application requires a dedicated LDAP server, a new Tivoli Directory Server instance can be started first by using the Web Application virtual application pattern. WebSphere instances in the virtual system pattern can then connect to the Tivoli Directory Server LDAP server. Script packages in the virtual system pattern can be used to configure WebSphere Application Server with the new Tivoli Directory Server.

4.5.3 Deploying virtual patterns

Virtual system patterns are deployed to the cloud to build complex, application-ready middleware topologies. Weeks of assembling hardware and software can be replaced by specifying a few parameters in the IBM PureApplication System virtual system pattern deployment wizard. The pattern example, which is shown previously in Figure 4-13 on page 128, can be deployed through an easy-to-follow wizard by completing the following steps:

1. Select **Patterns** → **Virtual Systems**.
2. Select **WebSphere Advanced cluster** in the list of patterns and click **Deploy** in the cloud icon, as shown in Figure 4-16.

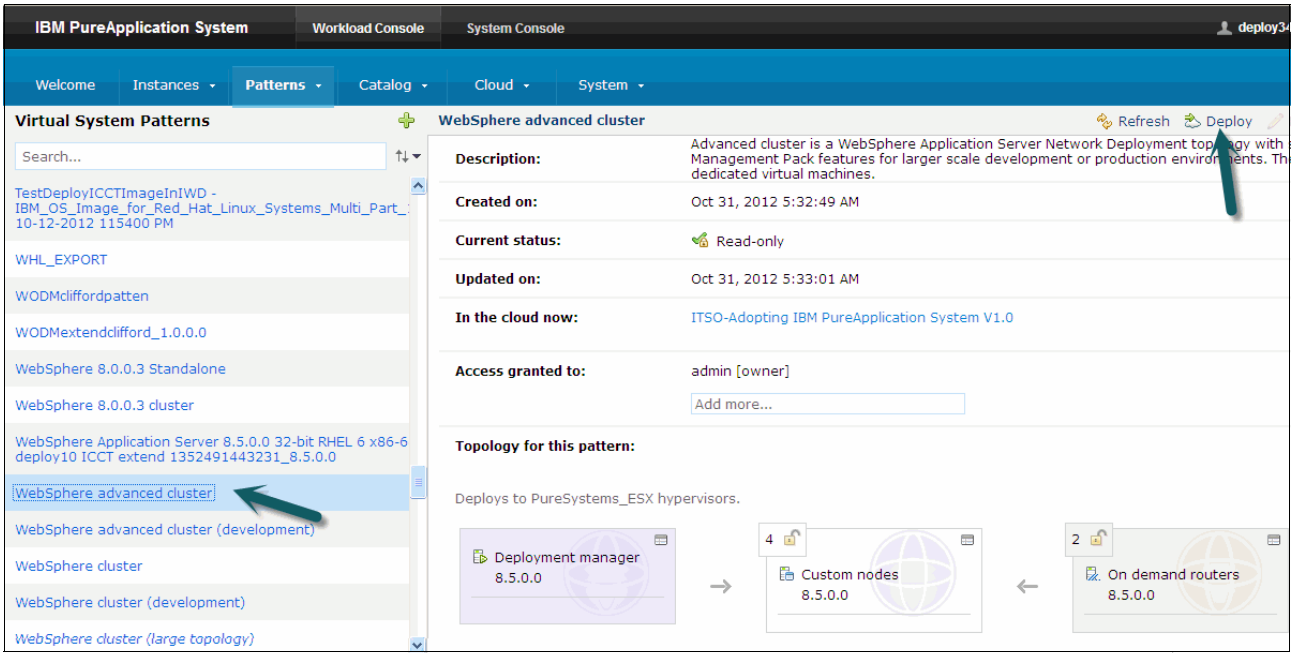


Figure 4-16 Deploy the virtual system pattern

A pop-up window opens with links to each configurable category. Each link can be selected to view or configure the options. The check mark to the left of the Choose Environment and Schedule deployment links indicate that they need no other configuration.

3. Enter a name for the virtual system. Figure 4-17 shows the name as ITSO-Adopting IBM PureApplication System. Click **Configure virtual parts** to expand that section.

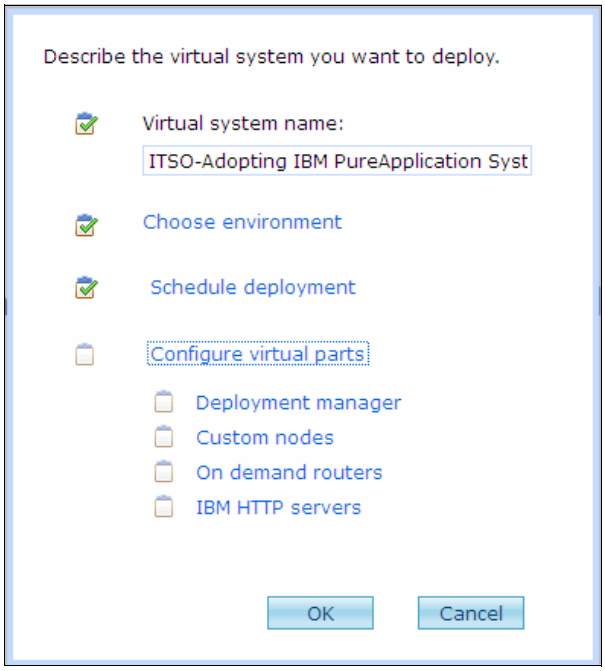


Figure 4-17 Deployment configuration

4. Click **Deployment manager**. You see several settings that you can modify to customize the deployment manager, as shown in Figure 4-18. At minimum, enter the passwords for the root user, the WebSphere administrator, and the database administrator.

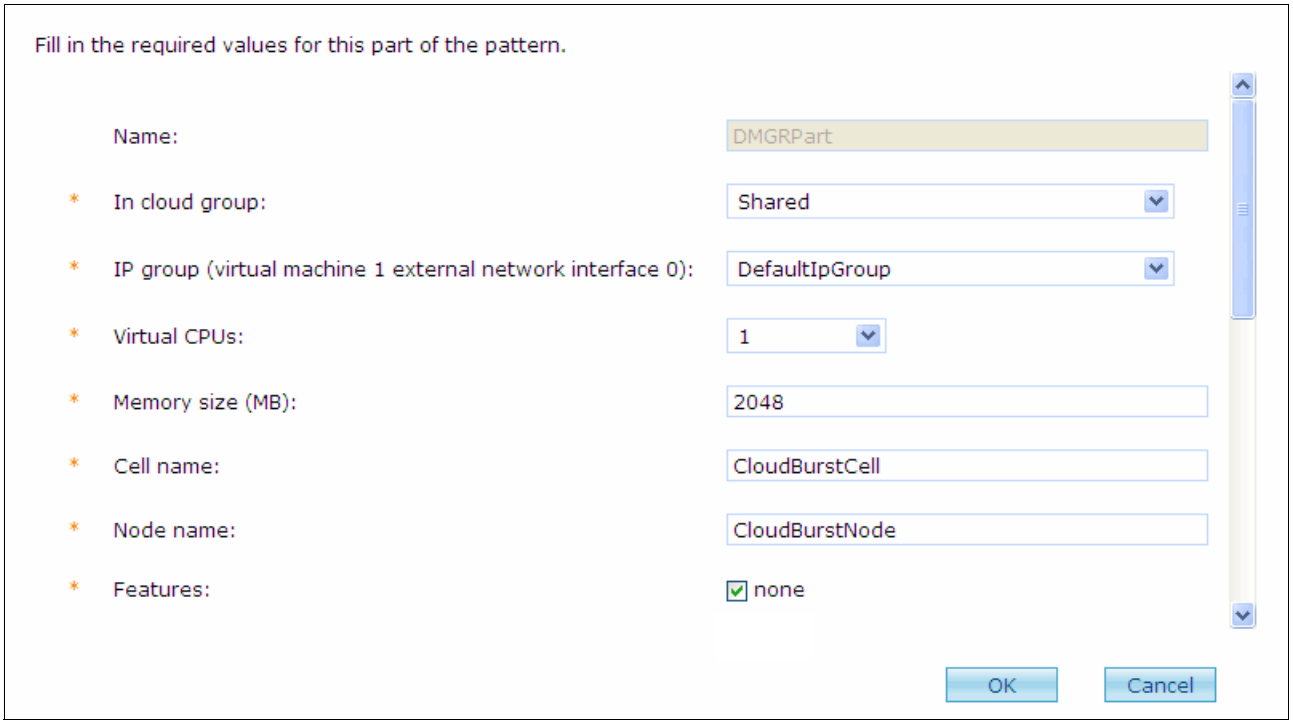


Figure 4-18 Settings for deployment manager

The following parameters are available to optionally configure, which provides optimal flexibility for customizing the environment:

- Virtual CPUs
- Memory size
- Reserve physical CPUs
- Reserve physical memory
- Cell and node names
- Feature packs to install
- Passwords for root
- User ID and password for the WebSphere administrator
- Data source name and JNDI name
- Database settings, including the database name, user ID and password, host, and port
- Web cluster prefix, number of clusters, and number of servers per node

5. Click **OK**.
6. Expand Custom nodes and enter the passwords for the root and WebSphere administrative users. Click **OK**.
7. Expand IBM HTTP servers and enter the passwords for the root and WebSphere administrative users. Click **OK.**
8. Click **OK** to deploy the virtual system, as shown in Figure 4-19.

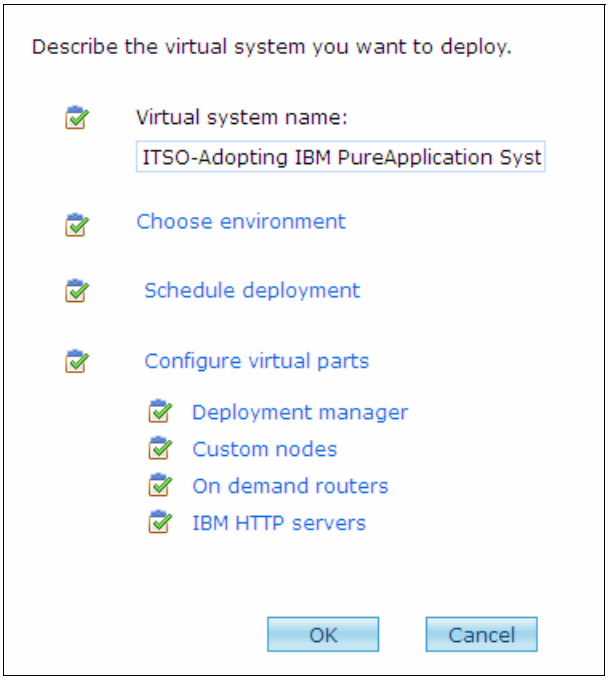


Figure 4-19 Deployment wizard that completed

9. The user interface opens the new virtual system instance for you to monitor, as shown in Figure 4-20. The following information is available:
   - The Current status section shows the current state of the virtual system instance.
   - The History section has a log with information about the transfer of the files to the system. Click **Refresh** occasionally to follow the progress of the deployment.

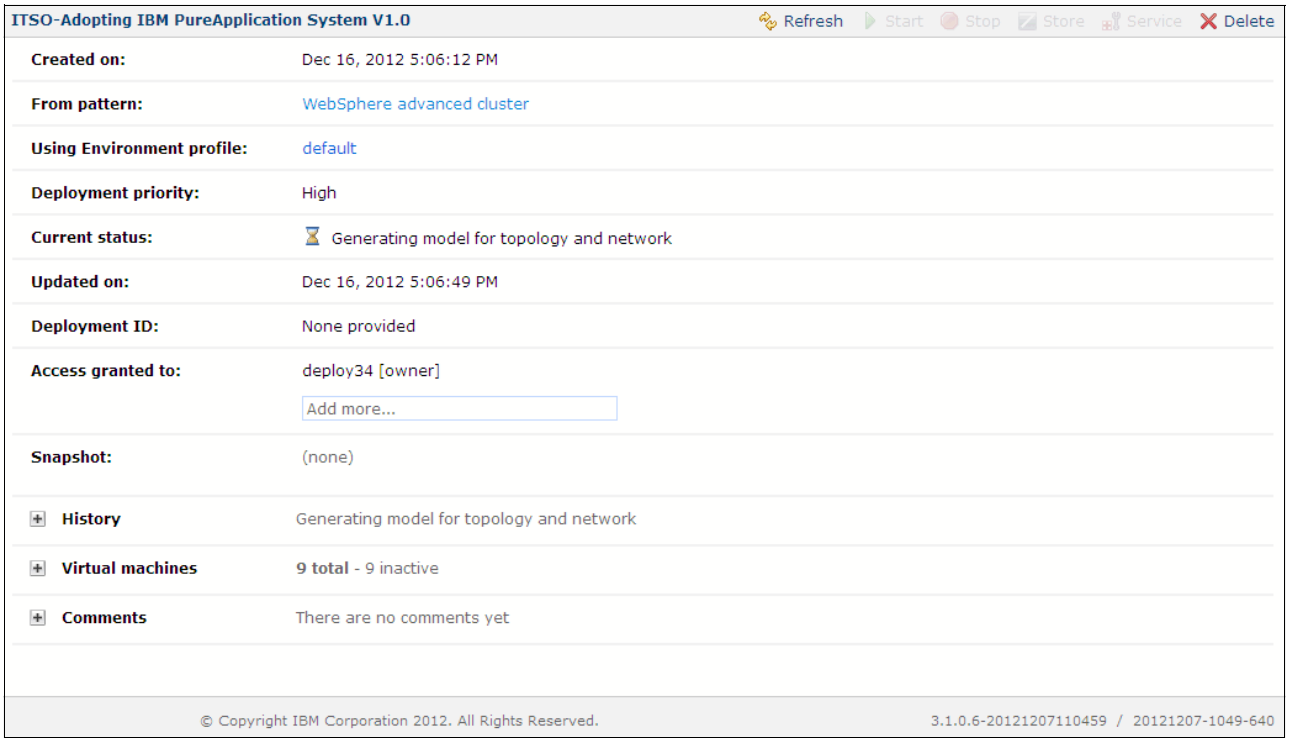


Figure 4-20 Pattern status

– The virtual machines that are created for the instance are listed in the Virtual machines section, as shown in Figure 4-21.

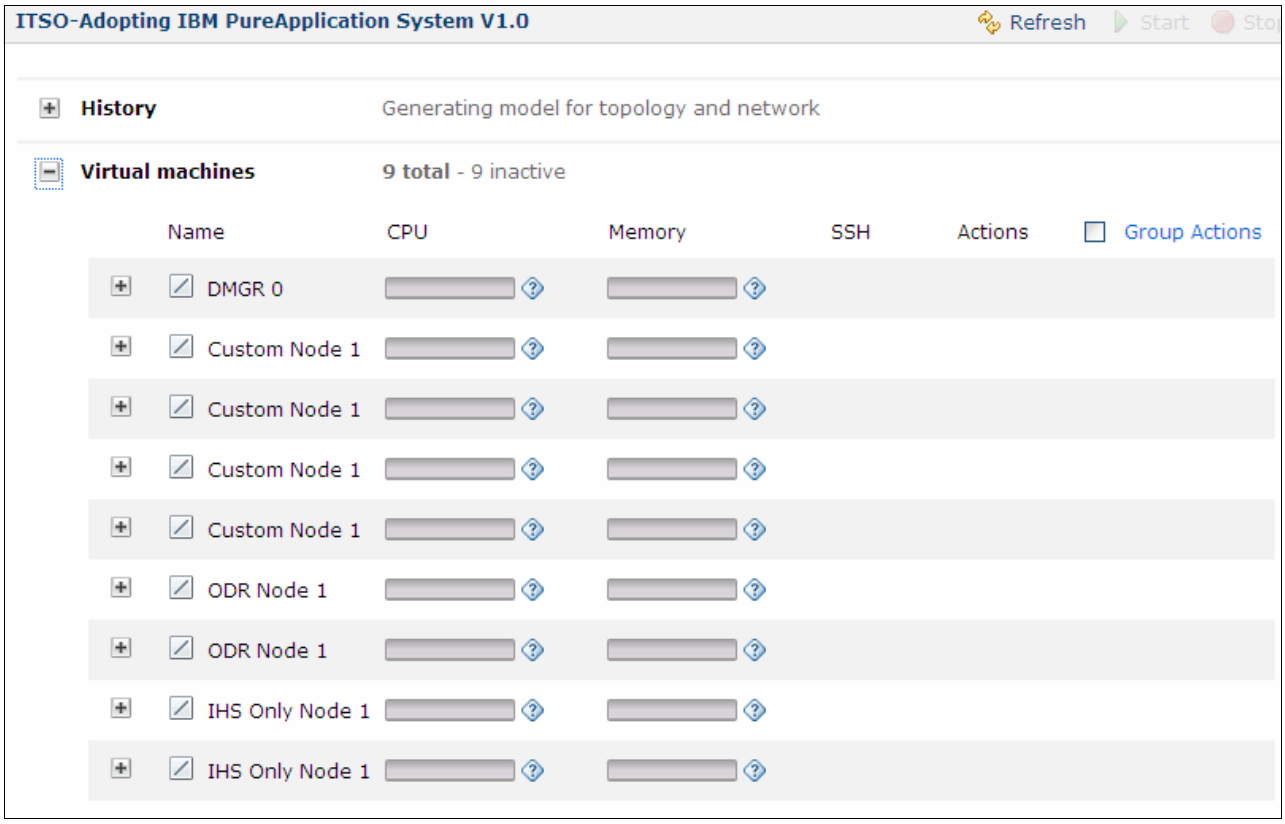


Figure 4-21 Virtual machines

– You can expand each virtual machine to find more information, including the status of the machine and the hypervisor where it is deployed. Figure 4-22 shows the virtual machine for the deployment manager. The systems are operational and the middleware is configured and started. This display shows information about the deployment manager configuration. It also provides links to log in to the console or to the system by using VMC. You can review the output of the script packages.

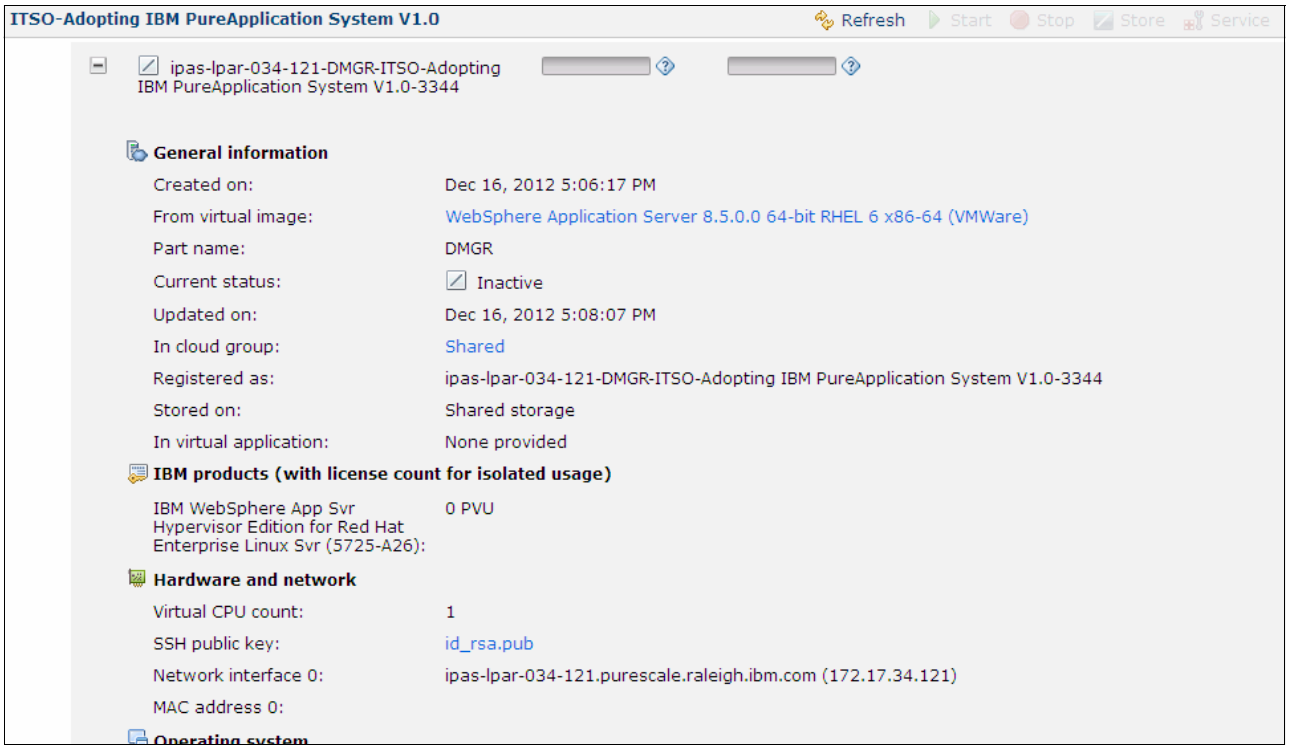


Figure 4-22 Virtual machine details

4.5.4 Customizing images and patterns

When the preinstalled content does not meet the needs of an enterprise, IBM PureApplication System provides many powerful options for customization. These customization options enable more flexibility to satisfy various requirements. Customization can occur in virtual system patterns and in virtual images. For more information about how to perform customization options in virtual system patterns, see Chapter 5, “Customizing Virtual System Patterns” on page 199.

4.5.5 Image Construction and Composition Tool and AMC tools usage

The IBM Image Construction and Composition Tool is a web-based application that simplifies the construction and creation of virtual images through several wizards. It also aids in the packaging of automation scripts that can be used to extend (customize) existing virtual images and to deploy software on these images.

The Image Construction and Composition Tool provides the capabilities to combine your own operating system definition with custom software bundles to compose virtual images that can be provisioned into the cloud. With Image Construction and Composition Tool, you can configure the PureApplication System as the cloud provider. By using this feature, you can import x86 or AIX images from the catalog, extend those images with software bundles, then capture the new image back into PureApplication System. You can then deploy the new image in PureApplication System.

For more information about how to use the IBM Image Construction and Composition Tool, see Chapter 5.3, “Build virtual images by using the IBM Image Construction and Composition Tool” on page 213.

PureApplication System also includes Advanced Middleware Configuration (AMC) as a workload. AMC stores configuration data on a framework server and uses the AMC Import Script Package to update that data. The AMC Integration Script Package incorporates that configuration data (including applications) in new virtual system patterns.

Use Advanced Middleware Configuration when one or more of the following conditions applies to you:

- You want to deploy applications as virtual system patterns.
- You do not have reliable end-to-end automation for the installation and configuration of applications.
- Your existing automation is specific to a single topology.
- You want to reduce your investment in low-level automation.
- You want to migrate WebSphere products into the cloud.

4.5.6 Logical to physical mapping for virtual systems

The virtual system focus becomes topology-centric when the user creates the topology pattern and deploys the solution. Application and configuration scripts that customize the environment according to a specific client environment are added to the virtual system pattern when needed. This environment is shown in Figure 4-23.

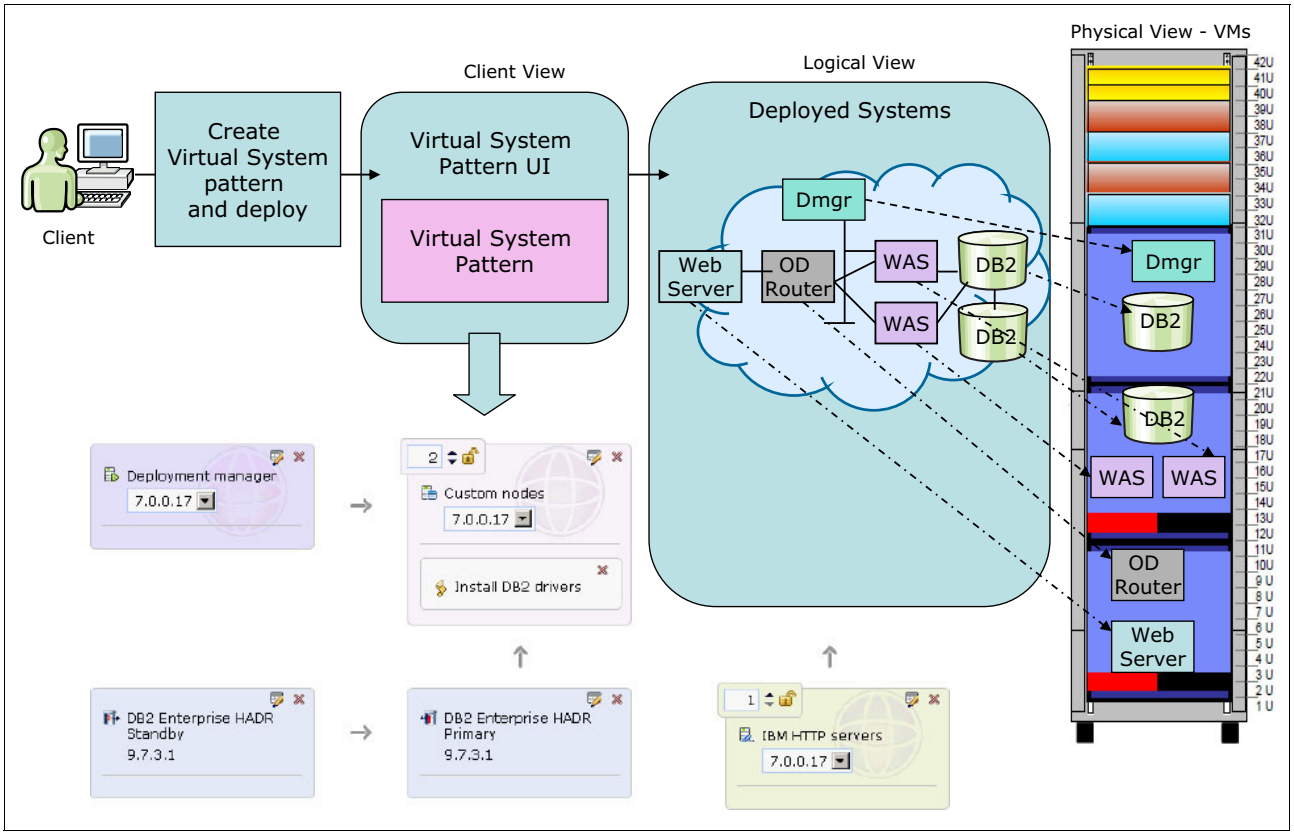


Figure 4-23 Logical to physical mapping for virtual systems

Showing the virtual system deployment model from the client view, you see that all that is needed is to create a virtual system pattern and deploy it. Rather than focusing on the application, you focus instead on the topology of the system to be deployed. To deploy an application in this model, you must provide application and configuration scripts rather than leaving PureApplication System to do it for you automatically. The virtual system pattern translates in the logical view to seven distinct instances. There is an instance that is created for each part that is contained in your virtual system pattern. This configuration includes a deployment manager, two custom nodes, a non-demand router, a DB2 standby instance, a DB2 primary instance, and an HTTP server. Finally, this configuration translates to seven distinct virtual machines that are created for you with the associated virtual image middleware in the PureApplication System rack. Unlike virtual application deployments, you determined this topology explicitly in your pattern definition.

4.5.7 Benefits and trade-offs

Determining what works best in your environment is a balance of benefits and trade-offs.

Virtual systems include the following benefits:

- Virtual system patterns provide repeatable, reproducible system deployments

  Virtual systems offer repeatability, consistency, reproducible system deployment, and rapid deployment times for simple and complex middleware configurations. Virtual systems also preserve the control and flexibility of traditional middleware environments. As you define and customize a topology, you can reproduce in any place without more effort.

- Virtual system patterns are simple to create and deploy

  The virtual system patterns are simple to create and deploy but it does take a little more work on your part than a virtual application deployment. To install your applications during the virtual system deployment, you must provide script packages that you created.

  After deployment, users can access the environment and middleware infrastructure as before. This ability means they could run administrative scripts, access the workload console that is provided by the deployed middleware software, and any other task they would normally perform.

- Takes existing middleware topology and provides instant migration

  With the virtual system deployments, you can take existing middleware topology along with any middleware configuration scripts and create a similar topology for the supported IBM middleware on PureApplication System. This ability is possible because the virtual system deployment model allows for far more customization than the virtual application deployment model. Because virtual system patterns use middleware topology, it provides an instant migration path from existing topology.

- Provides more control and administration

  You have full access to the administration model of the middleware components in the topology. This access provides more control and the ability for you to manage the middleware components in the topology.

The use of virtual systems includes the following trade-offs:

- Virtual system configuration scripts are required

  To provide this customization, all virtual system deployments require script packages to automate the configuration of your virtual system and make the deployment repeatable. This requirement might result in an investment of time and effort on your part (as opposed to money) to create these script packages.

  To simplify this process for WebSphere applications, PureApplication System includes the AMC tool. AMC makes it easier for you to create repeatable and deployable virtual system patterns. This AMC tool includes applications and configurations by inspecting an existing application cell, extracting all of the configuration details, and encapsulating them in a script package that re-creates that configuration when the pattern is deployed. In this context, an existing application cell refers to the WebSphere Application Server cell definition in which an application is deployed.

A deployed application is made up of the application binary (WAR, EAR, and so on), the server topology, the configuration of that topology to support the application, and external resources. The inspection of this application (from a WebSphere pattern perspective) includes attaching to and inspecting a WebSphere cell and identifying the wanted topology to support the application. The configuration settings that are contained with the WebSphere Cell definition and the application deployment options and artifacts also are inspected. It does not include the analysis of the application source. This configuration is useful for WebSphere Application Server applications that do not conform to the constraints of any PureApplication System virtual application patterns and do not have a complete, reusable, and reliable set of deployment and configuration scripts.

- Putting to much content in virtual machine images

  When virtual system patterns are created, it is helpful to think about how a pattern can support many applications, which requires taking a layered approach. If you put too much content in the virtual machine images, the patterns become difficult to reuse. It is common to include the operating system and middleware in the images and then use the script packages to lay down the application and configure the middleware. This configuration affords greater reuse.

4.6 Virtual applications

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. Each virtual application pattern is associated with a pattern type, which is a collection of plug-ins that provide these resources and services for a particular business purpose in the form of components, links, and policies. The pattern types, product extensions of the cloud system, and the types of virtual application that you build depend on the pattern types that you enabled. In the next sections, the concepts and several topics of virtual applications are described.

4.6.1 Concepts

Virtual application patterns are application-centric in their design. They provide a mechanism to represent middleware applications in a simplified model that abstracts away the underlying middleware infrastructure. For example, you describe a middleware topology with IBM HTTP Servers, IBM WebSphere Application Servers, and databases. The IBM PureApplication System then builds appropriate infrastructure and deploys the application into a cloud environment.

Virtual application patterns are highly optimized and are constructed solely for supporting a singular workload. This pattern requires the least amount of customization during deployment and provides the most direct method for obtaining a rapid return on investment. These patterns are implemented by using virtual application pattern types. These pattern types integrate the capabilities of multiple middleware software elements into a cohesive, built for purpose solution. By using this solution, you can represent your complete, often complex, environments as a single deployable unit.

Because it is an application-centric approach, you provide the application files and describe the characteristics of how the application should be run and managed by using policies. The appliance generates the middleware topology to meet your requirements, as shown in Figure 4-24. Also shown in Figure 4-24 is an overall picture of the relationship between virtual applications instance and virtual application patterns.

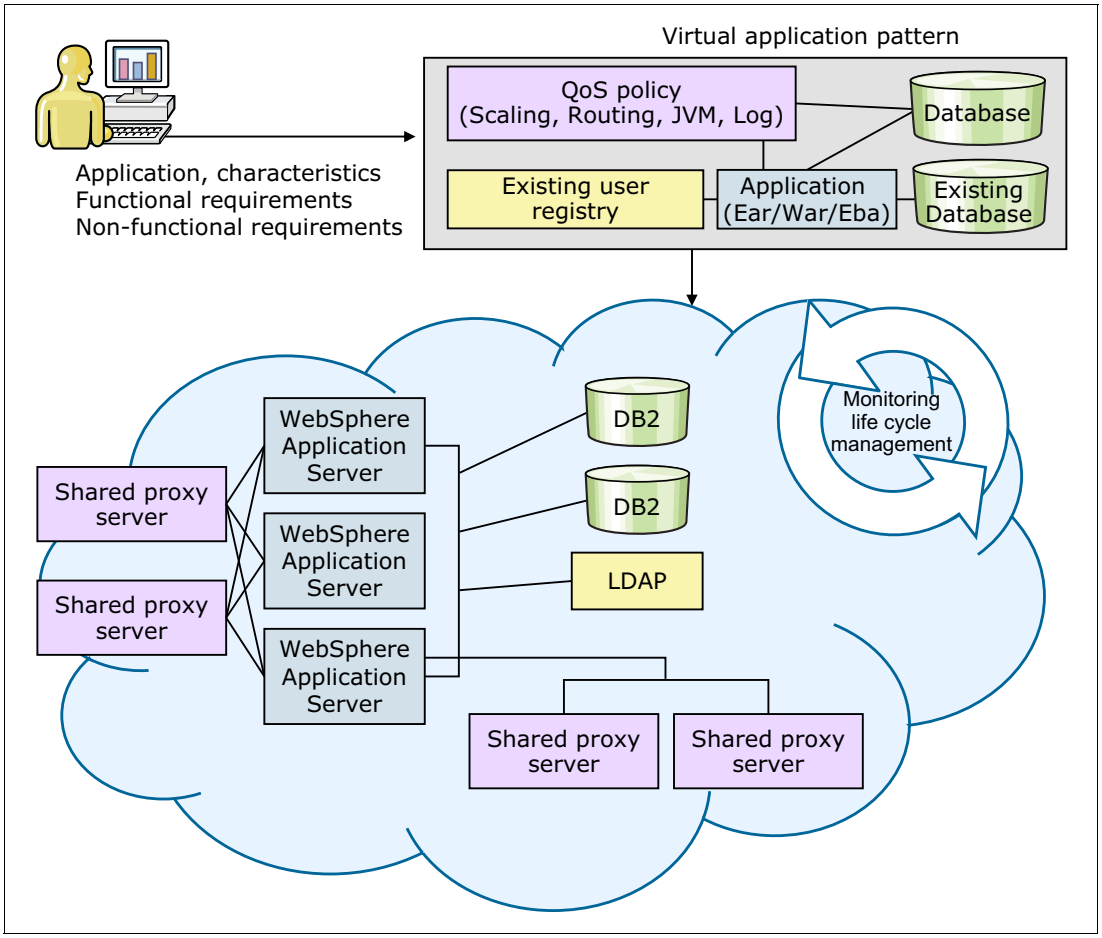


Figure 4-24 The overall picture: Virtual application pattern to virtual application instance

The virtual application deployment model is a platform as a service (PaaS) model in which your application is the focal point.

A virtual application pattern is the critical element that with which you rapidly set up and manage cloud application infrastructure. To create a virtual application pattern, you start with your application and define its specific requirements, such as what services it requires and the quality of service (QoS) that is needed. Based on the assets in your virtual application pattern, PureApplication System deploys and configures the appropriate middleware components in the background to run your application. This configuration simplifies the end-to-end process of creating, deploying, and configuring the middleware components for your applications. PureApplication System handles it all. After deployment, PureApplication System also takes care of monitoring the application for you, which adds resources as needed, for example, to meet the QoS requirements.

4.6.2 Virtual application patterns

Virtual application patterns represent a new cloud deployment model. The patterns are an evolution of the traditional topology patterns that are supported in virtual system patterns. Fundamentally, virtual application patterns raise the level of abstraction one notch higher than virtual system (topology) patterns and put the focus on the application. This difference means that when you use a virtual application pattern, the focus is on the application instead of the application infrastructure.

Virtual application patterns encapsulate the installation, configuration, and integration of middleware, and the installation and configuration of applications that run on that middleware. Most of this feature is hidden from you, the user, which means that you have less control over configuration and integration. However, you also reduced labor and increased agility. You can concentrate on the development of the application and its components and IBM PureApplication System can create and manage the infrastructure that services that application.

Reducing deployment time, increasing consistency, and fostering agility are benefits that you would likely expect when cloud-based approaches for your middleware application environments are explored. The IBM PureApplication System solution tackles these issues by making the deployment of cloud middleware environments fast, repeatable, and efficient.

The pattern-based approach is the foundation of IBM PureApplication System. It is consistent for virtual application patterns and virtual system patterns. By using the cloud appliance, you build and deploy patterns that represent your configured application environments. When you are ready to use a particular application environment, you pick a pattern and deploy it. IBM PureApplication System automates the deployment, configuration, and integration of the various virtual machines that make up your environment and delivers the completed product in a matter of minutes.

General features

Some of the virtual application patterns main features are shown in Figure 4-25.

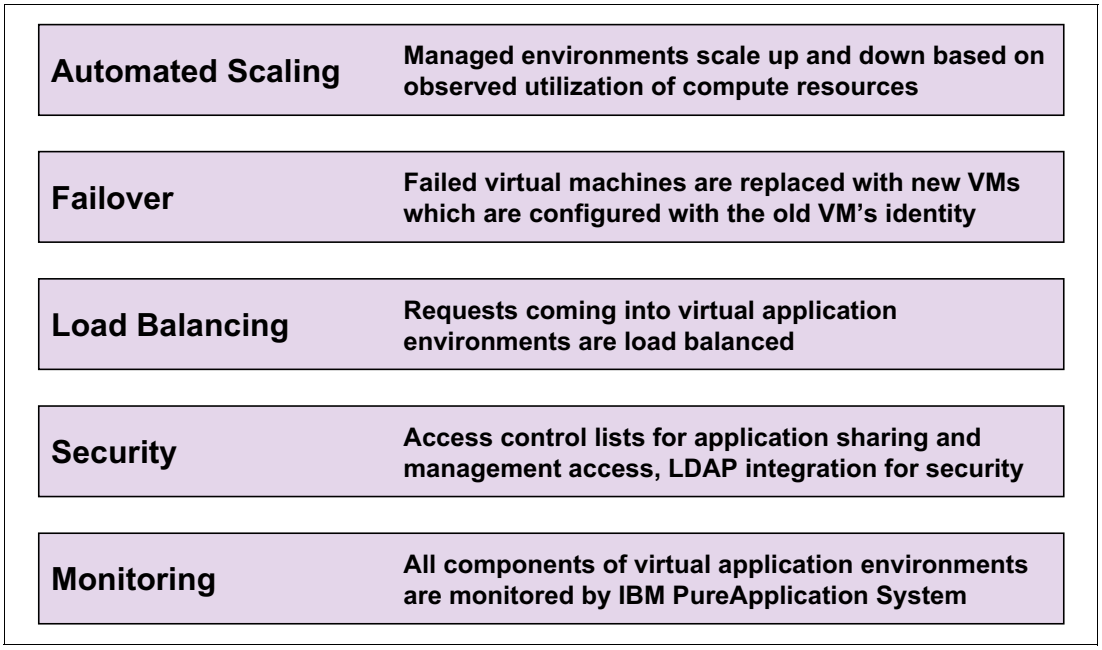


Figure 4-25 General features of virtual application patterns

Automated scaling

You can include a policy in your virtual application pattern that provides automated scaling, which is managed by PureApplication System. As PureApplication System monitors the resources on the system, it scales up and down based on the application load.

Failover

PureApplication System automatically replaces a failed virtual machine if a virtual machine has a problem at any time. This feature provides application failover automatically.

Load balancing

Load balancing is also done for you by PureApplication System when the proxy shared service in your cloud is used. Requests that are serviced by your virtual application automatically are distributed across available instances.

Security

Security is an important feature in virtual applications. Highly secure environments can be easily integrated with LDAP for application security.

Monitoring

All the various components of the virtual application environments are monitored for you by PureApplication System. Through this monitoring, you can gain quick access to the status, performance, and resource usage of all areas of your virtual application.

All of these features are built in by using IBM PureApplication System infrastructure without any other costs. The unique requirement is configuring the solution according to your needs; for example, to use LDAP as an application security endpoint.

Elements and functions of a virtual application pattern

There are five main elements in a virtual application pattern features the following main elements, as shown in Figure 4-26:

- Pattern types
- Plug-ins
- Components
- Links
- Policies

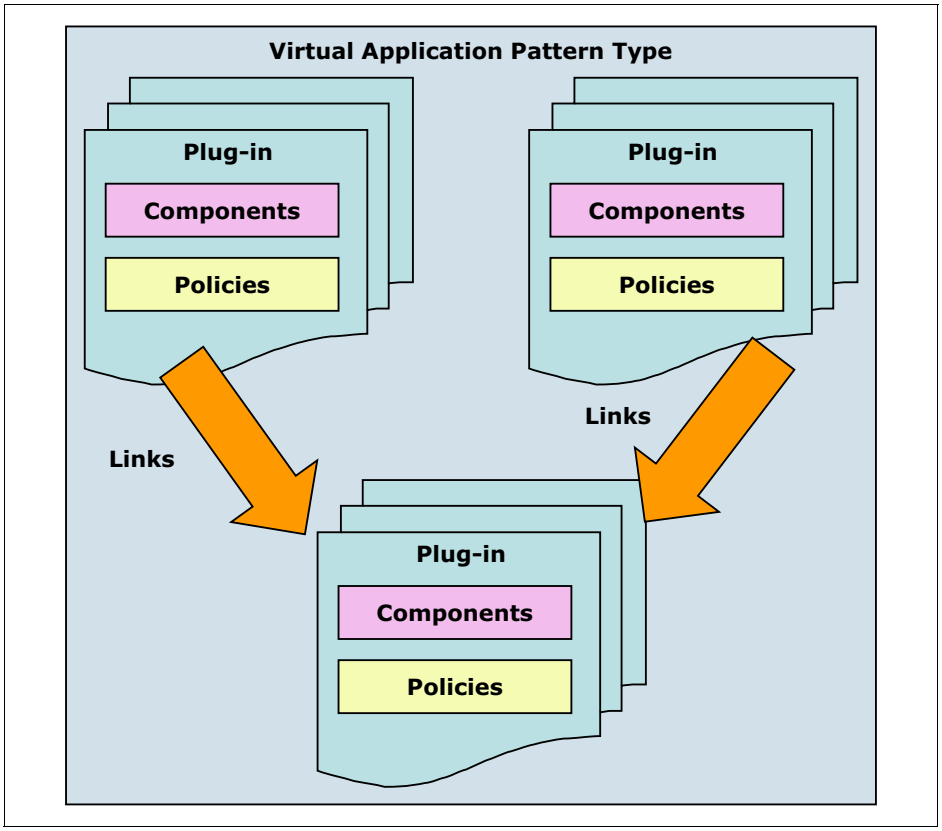


Figure 4-26 Virtual application pattern elements

Virtual application pattern type

A virtual application pattern type is a collection of plug-ins that define components, links, and policies, with configuration files, which are packaged in a `.tgz` file. The virtual application patterns are used to build a virtual application that includes these components, links, and policies.

Virtual application pattern types are the containers of solution-specific and topology-specific resources that are required for different types of virtual applications. Pattern types are really the aggregation of various capabilities for a specific type of application. The actual solution-specific intelligence is delivered via plug-ins. A plug-in can participate in multiple pattern types; however, a plug-in always has one primary pattern type. The pattern types also provide shared services that incorporate runtime services, such as caching services and elastic load balancing.

The following types of virtual application patterns included with the PureApplication System:

- IBM Foundation Pattern

  This pattern type is used to provide shared services for deployed virtual applications, such as monitoring and load balancing.

- IBM Web Application Pattern

  This pattern type is used to build and deploy web applications. The IBM Web Application Pattern provides a set of components that often are needed for online web applications. These applications include Java Platform Enterprise Edition applications, databases, Lightweight Directory Access Protocol (LDAP) servers, and messaging. These components are based on products such as WebSphere Application Server, and Tivoli Directory Server. By using the Web Application Pattern, you can incorporate connectors to remote systems, such as WebSphere MQ, CICS®, and IMS™ into your virtual application pattern.

- IBM Database Patterns

  This pattern type is used to build and deploy database instances. You can use IBM Database Patterns separately or you can incorporate them into a virtual application pattern that is based on the Web Application Pattern. The IBM Database Patterns provide support for DB2 in a database as a service (DBaaS) model, with which you can simplify and standardize the creation of databases. The following database patterns available that based on your needs:

  - The Transactional Database pattern is primarily used for online transaction processing and is optimized for transactional applications.
  - The Data Mart pattern is primarily used for data warehousing and is optimized for reporting applications.

- Application Pattern Type for Java

  This pattern type is used to build and deploy Java applications. The Java Pattern provides support for building Java applications. This pattern type provides an easy and fast mechanism for provisioning Java applications. It also includes components with which you can connect to network resources, such as databases and web services.

- Other patterns

  These patterns are the patterns that are included with PureApplication System and other patterns that you can import into the PureApplication System from the PureSystems Centre offering.

- Creating your pattern

  You also can create your own patterns by using the Plug-in Development Kit, which is available for download. For information about how to create your own pattern, see Chapter 7, “Integrating PureData for Transaction” on page 309.

Some topics, such as web application patterns, Java patterns, database patterns, and PureSystems Centre are described in other sections of this book.

Templates

Templates are previously created patterns that you save as application templates for reuse. You build a virtual application pattern according to a specific pattern type and an optional template. For instance, to build a standard Java Platform, Enterprise Edition web application, you can choose the Web Application Pattern Type 2.0 and the template Blank Java EE web application.

Plug-ins

A plug-in is the primary mechanism for delivering and installing extensions to PureApplication System in support of customer workloads and applications. For example, a plug-in is the basic unit of content for virtual application workloads. It generally implements a specific capability for an application such as the WebSphere Application Server plug-in, which provides components to host Web Archive (WAR), Enterprise Archive (EAR), or Enterprise Bundle Archive (EBA) applications in WebSphere Application Server. It also functions for DB2 or Tivoli Directory Server plug-ins, which provide a link to connect a WAR, EAR, or EBA file with a database, as shown in Figure 4-27.

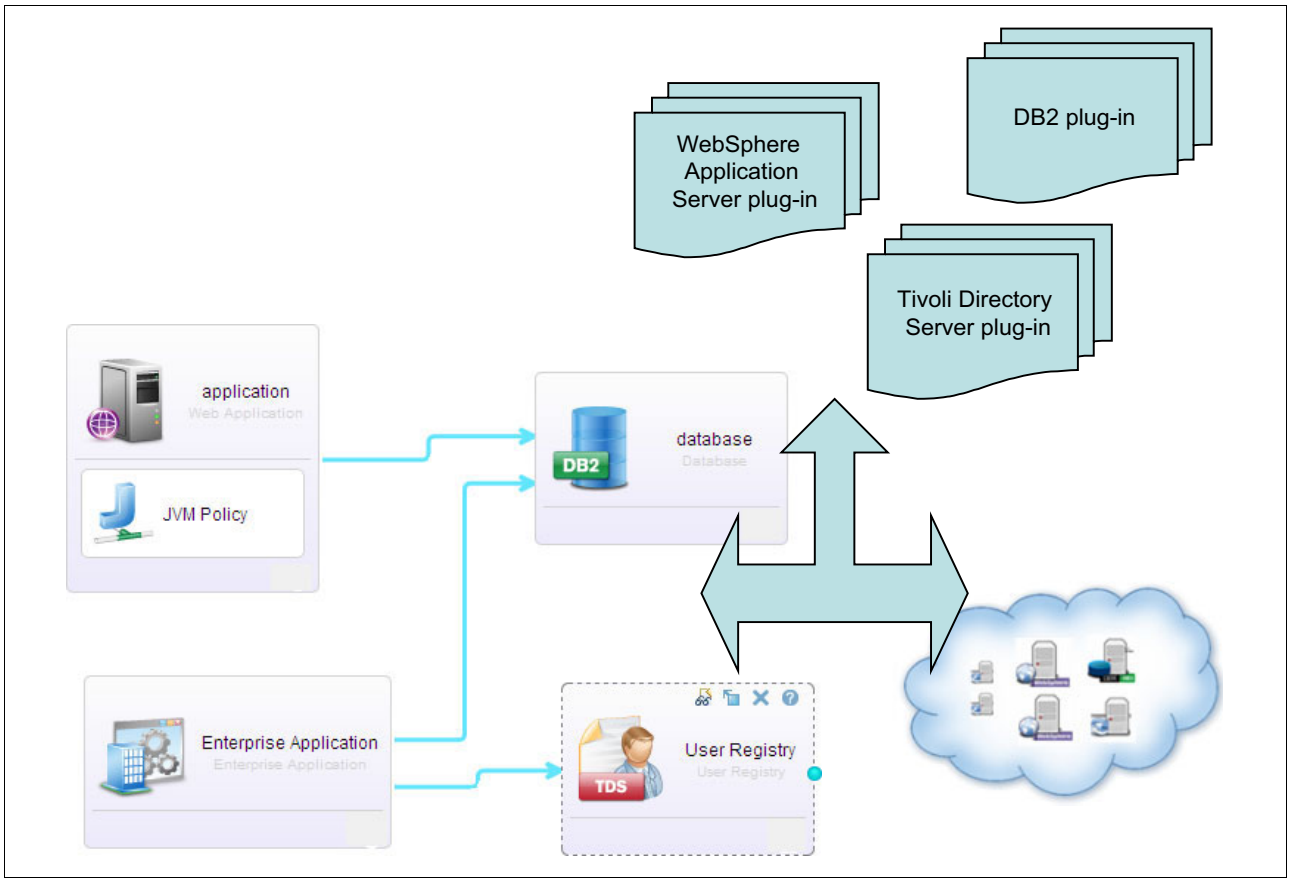


Figure 4-27 Plug-in functionality example

Plug-ins provide the key parts of a virtual application and the underlying implementation that makes the application deployable in the cloud.

Plug-ins are responsible for providing all of the necessary functionality to create and manage the real entities that are realized for the components, links, policies, services, and other features. Plug-ins first provide the visual elements that you see in the virtual application builder when your virtual application pattern is built. Plug-ins also are responsible for providing functions that are necessary to build the model of the system and eventually are deployed to the cloud. Plug-ins provide the necessary scripts to provision and configure the particular application elements. They also include the logic to federate the necessary elements to react to changes in the configuration, and to provide dynamic processing in support of policies. At deployment time, the plug-in provides implementation details and can further augment the deployed foundation image.

IBM PureApplication System provides many utilities to make this process easy and orchestrates the interaction with the plug-ins to deliver the necessary function in support of the application. This process is optimized and automated within the pattern so that a typical user of the pattern need not understand all of the mechanisms of the middleware. The user can instead focus on the wanted behavior of the application. For more information about how to use plug-ins, see 6.5, “Plug-in environment setup and creation of custom patterns” on page 286.

For a list of preinstalled plug-ins, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/apc_pluginov.html

Components

Components represent an application artifact such as a WAR file, and attributes such as a maximum transaction timeout. In terms of the order management application example, the components for the application are the WebSphere Application Server nodes and the DB2 nodes. The WebSphere Application Server components include the WAR file for the application and the DB2 components connect the application to the existing DB2 server.

The available components in the virtual application patterns that are provided with IBM PureApplication System are shown in Table 4-1. Some components can vary according to the standard application pattern that is selected, such as web application pattern, database pattern, or Java application pattern.

Table 4-1 Application components

Component Name	Description
Extra archive file (web application)	Specifies the external archive file that contains other files that are needed by the WAR or EAR file.
Extra archive file (Java application)	You can upload other archive files and your Java application archive file. You can use these archive files to deploy more resources, such as JDBC drivers or `.war` files, into an application server, or to overwrite parts of the deployed Java application, such as configuration files.
Enterprise Application Component	The enterprise application (WebSphere Application Server) component that represents an execution service for Java Platform, Enterprise Edition EAR files.
Existing Web Service Provider Endpoint	A web service provider that is provided by a remote server.
Java application (IBM Java Runtime Version 7)	The Java application component represents an execution service for the Java SE platform. You can use this component to deploy any application that requires a Java runtime environment.
Policy Set	A policy set is a component that is used to define QoS policies. It is a collection of assertions about how services are defined, which can be used to simplify security configurations.
Web application component	The web application component represents an execution service for the Java Platform, Enterprise Edition WAR files.

In Figure 4-28, you can see the Application Components palette in the IBM PureApplication System.

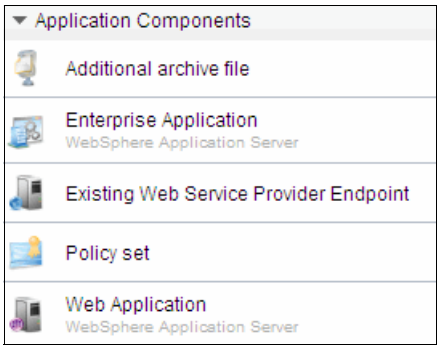


Figure 4-28 Application components palette

The database components are shown in Table 4-2.

Table 4-2 Database components

Component Name	Description
Database Studio web console	The Database Studio web console component is a database tool that is included with the IBM Database Patterns. This plug-in component is not available on the Virtual Application Builder unless you accept the license for the IBM Database Patterns.
Database (DB2)	The DB2 database component represents a pattern-deployed database service.
Existing database (DB2)	An existing DB2 database component represents a connection to a remote DB2 database instance that is running remotely outside of the cloud infrastructure. The configuration properties allow a connection to the remote DB2 database.
Existing database (Informix®)	An existing Informix database component represents a connection to a remote Informix database that is running remotely outside of the cloud infrastructure. The configuration properties allow a connection to the remote Informix database.
Existing database (Oracle)	An existing Oracle database component represents a connection to an Oracle database instance that is running remotely outside of the cloud. The configuration properties allow a connection to the remote Oracle database.
Existing IMS database	An Information Management Systems Database IMS DB component represents a connection to an IMS database instance that is running remotely outside of the cloud infrastructure. The configuration properties allow a connection to the IMS DB system.

The Database Components palette in IBM PureApplication System is shown in Figure 4-29.

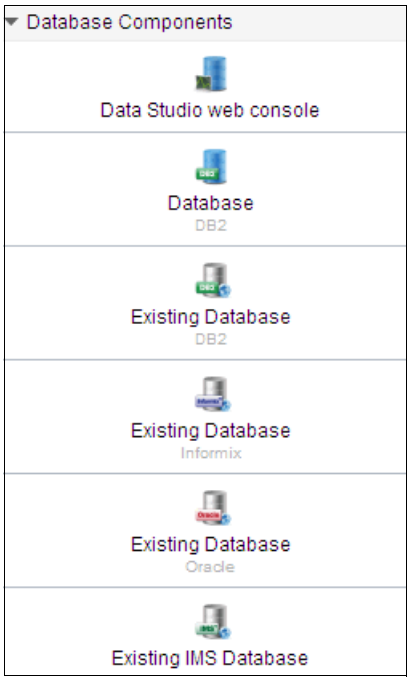


Figure 4-29 Database components palette in IBM PureApplication System

The Messaging components are shown in Table 4-3.

Table 4-3 Messaging components

Component Name	Description
Existing Messaging Service (WebSphere MQ)	An existing message service component represents a connection to an external messaging system, such as WebSphere MQ. The presence of a messaging system allows an enterprise application that is running on WebSphere Application Server to connect to the external messaging resource, such as WebSphere MQ.
Topic	A topic represents a message destination on a WebSphere MQ messaging service through which messages are published and subscribed. If you purchased and enabled the Messaging Extension for Web Application Pattern pattern type, you can connect to an external WebSphere MQ messaging service or a WebSphere MQ messaging service that is deployed by using the Messaging Extension for Web Application Pattern.
Queue	A message queue is a message queue on a WebSphere MQ service from which messages are sent and received. If you purchased and enabled the Messaging Extension for Web Application Pattern pattern type, you can connect to an external WebSphere MQ messaging service or a WebSphere MQ messaging service that is deployed by using the Messaging Extension for Web Application Pattern.

In Figure 4-30, you can see the Messaging components palette in IBM PureApplication System.

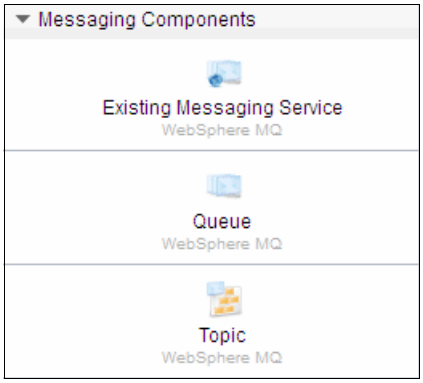


Figure 4-30 Message components palette in IBM PureApplication System

The OSGi components are shown in Table 4-4.

Table 4-4 OSGi Components

Component Name	Description
Existing OSGi Bundle Repository (WebSphere Application Server)	This component provides the URL of an existing WebSphere Application Server OSGi bundle repository.
OSGi Application (WebSphere Application Server)	This component represents the OSGi application on WebSphere Application Server.

In Figure 4-31, you can see the OSGi components palette in IBM PureApplication System.

Figure 4-31 OSGi components palette in IBM PureApplication System

The Transaction Processing components are shown in Table 4-5.

Table 4-5 Transaction Processing components

Component Name	Description
Existing CICS Transaction Gateway	An existing CICS Transaction Gateway (TG) component represents a connection to an existing CICS TG instance that is running remotely outside of the cloud. The configuration properties allow a connection to the CICS Transaction Gateway.
Existing IMS Transaction Manager	An existing Information Management Systems Transaction Manager (IMS TM) component provides an enterprise or web application that is running on WebSphere Application Server to connect to and submit transactions to an existing IMS system that is running remotely outside of the cloud.

In Figure 4-32, you can see Transaction Processing components palette in IBM PureApplication System.

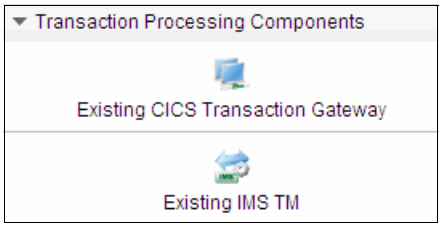


Figure 4-32 Transaction Processing components palette in IBM PureApplication System

The User Registry components are shown in Table 4-6.

Table 4-6 User Registry components

Component Name	Description
Existing User Registry (IBM Tivoli Directory Server)	An existing user registry cloud component represents an existing LDAP service (IBM Tivoli Directory Server) that can be attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security.
Existing User Registry (Microsoft Active Directory)	An existing user registry cloud component represents an existing LDAP service (Microsoft Active Directory) that can be attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security.
User Registry (Tivoli Directory Server)	A user registry (Tivoli Directory Server) cloud component represents a pattern-deployed LDAP service that can be deployed alone or attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security.

In Figure 4-33 on page 153, you can see the User Registry components palette in IBM PureApplication System.

Figure 4-33 User Registry components palette in IBM PureApplication System

The Other components that are available in the system are shown in Table 4-7.

Important: The components that are shown in Table 4-7 can appear or not depending on the virtual application pattern that is selected.

Table 4-7 Other components

Component Name	Description
Connect In	This component is used to open the firewall for inbound TCP connections from a specified address or range of addresses to a specified port in the target application component.
Connect Out	Specifies a component that is used to open the firewall for outbound TCP connections from a web or enterprise application to a specified host and port.
Monitored file	Use the monitored file component to specify a file, or collection of files, to monitor and be available in the logging view.

In Figure 4-34, you can see the Other components palette in IBM PureApplication System.

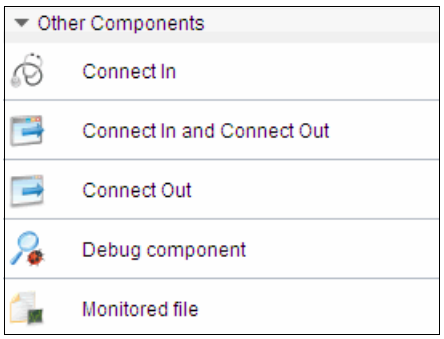


Figure 4-34 Other components palette in IBM PureApplication System

For more information about attributes and properties, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/apt_appov.html

Policies

A policy is a set of automated system processes that can perform actions, schedule work for users, or automate manual tasks. For example, you can attach an optional QoS policy to the virtual application. Two virtual applications might include identical components, but require different policies to achieve different service level agreements.

When policies are added to the application, you can extend the capability of the application. For example, if you want a web application to be highly available, you can add a scaling policy in the virtual application builder and IBM PureApplication System creates the application and topology to achieve that requirement.

The following common policies for the Web Application pattern type are available:

- Scaling policy

  Scaling provides runtime capability to scale the application platform as the load changes. A scaling policy component defines this capability and the conditions under which scaling activities are performed for your application, as shown in Figure 4-35.

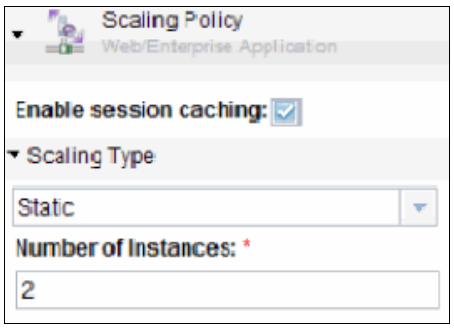


Figure 4-35 Scaling policy properties

- Routing policy

  Consider a routing policy that is a client policy for the proxy shared service. It provides routing and load balancing to multiple deployed web applications and supports HTTP and HTTPS requests. To enable an application to use the Elastic Load Balancing (ELB) shared service, you must add a routing policy to provide a virtual host name and a request protocol for the application, as shown in Figure 4-36.

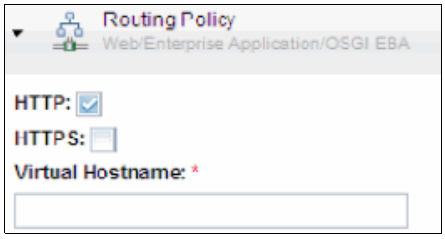


Figure 4-36 Routing policy properties

- Java virtual machine policy

  As shown in Figure 4-37, you can control the underlying Java virtual machine by using the JVM policy.

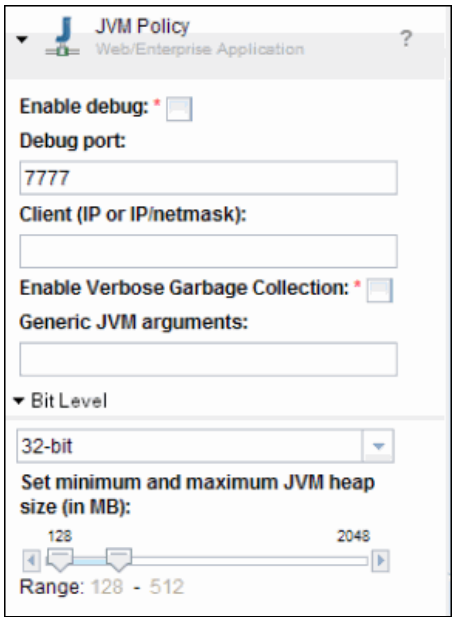


Figure 4-37 JVM policy

- Log policy

  The log policy specifies the configuration for log records. Figure 4-38 shows a Log Policy example.

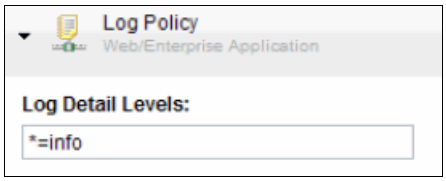


Figure 4-38 Log policy

Policies can affect the number of virtual machines that are started. For example, if you attach a scaling policy, multiple application server instances are connected with a load balancer and, optionally, an IBM WebSphere eXtreme Scale server for sharing sessions. Application artifacts are then deployed by starting the components and configuring them appropriately.

Policies can be applied globally at the application level or specified for individual components. For example, a logging policy defines logging settings. A scaling policy defines criteria for dynamically adding or removing resources from the virtual application. In terms of the order management application example, a Response Time Based scaling policy is applied. That policy scales the virtual application in or out to keep the web response time 1000 - 5000 ms.

When you deploy a virtual application, the virtual application pattern is converted from a logical model to a topology of virtual machines that are deployed to the cloud. Behind the scenes, the system determines the underlying infrastructure and middleware that is required for the application. It also adjusts them as needed to ensure that the quality of service levels that are set for the application are maintained. A deployed topology that is based on a virtual application pattern is called a *virtual application instance*. You can deploy multiple virtual application instances from a single virtual application pattern.

The components, links, and policies that are available to design a particular virtual application pattern are dependent on the pattern type that you choose and the plug-ins that are associated with the pattern type. Therefore, components, links, and policies are defined by plug-ins. When you create a virtual application pattern, the available components, links, policies, and configuration options are determined by the plug-ins that are included with the selected pattern type, as described in the following sections.

Links

Links connect components. Links represent some dependency or interaction between components. For example, certain components in the Web Application pattern type support links to other components in other pattern types, such as to a database component in the IBM Database Patterns type.

A link serves the following purposes:

- It ensures that the origin and destination components of the link are properly configured to support the connection.
- The link also ensures that the network and firewalls are configured appropriately to allow communication.
- The link ensures that dependencies are accepted and supported so that components can appropriately react to dependencies changes or failures.

Virtual Application Builder

The Virtual Application Builder in the IBM PureApplication System supports the application-centric approach for deploying applications to the cloud. This support provides the means of creating virtual application patterns.

A virtual application pattern consists of a combination of application components, links, and policies. The application component represents the middleware (such as WebSphere Application Server) to run the application instance. Links represent connections (such as JDBC), and policies represent the middleware configuration or quality of service.

To access Virtual Application Builder, you must have at least the Create new patterns permission.

Browse to the **Patterns** → **Virtual Applications** console page. By default, IBM Workload Deployer includes the following sample applications in the Virtual Application Patterns list:

- Sample Java EE web application
- Sample Web Application Only
- Secured Java EE web application

Figure 4-39 shows the Virtual Application Patterns sample applications pane.

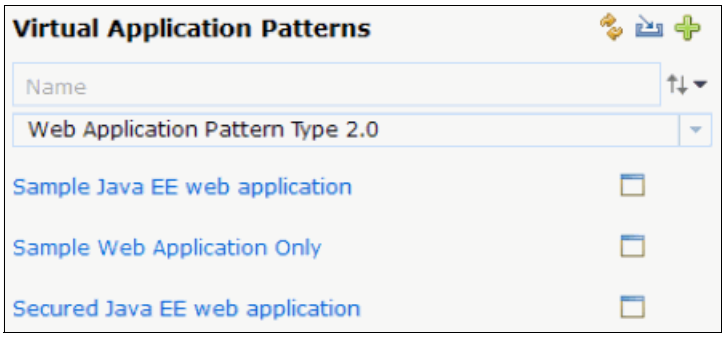


Figure 4-39 Virtual Application Pattern sample applications

Figure 4-40 shows the Virtual Application Builder.

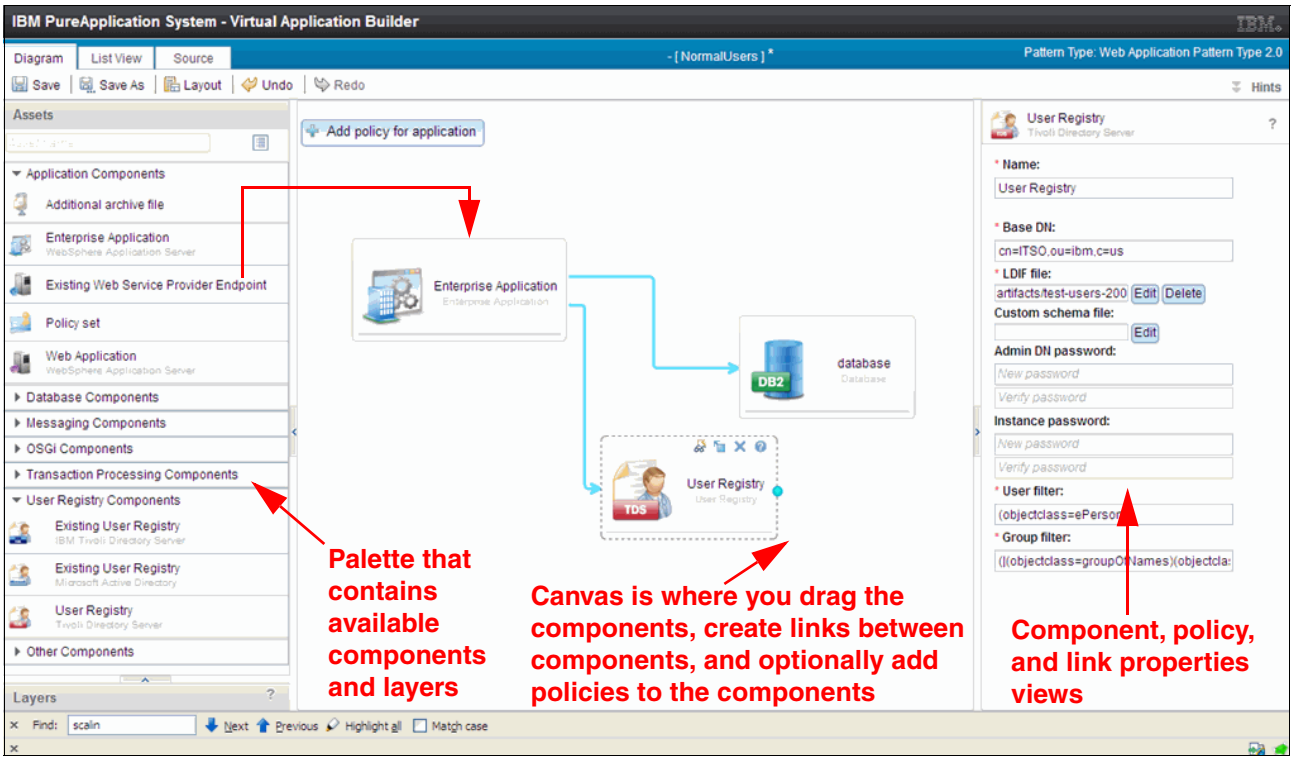


Figure 4-40 Virtual Application Builder

Use Virtual Application Builder to create your virtual application pattern. The palette on the left side of the pane shows the components that are available for your virtual application pattern. Figure 4-40 shows the Web Application Pattern so you see application components such as Enterprise application and Web application. You also see that the pattern contains Database Components, Messaging Components, and User Registry Components.

To build your pattern, you drag the needed components from the application onto the canvas in the middle of the interface. The canvas, as shown in Figure 4-40, is showing a pattern that includes an enterprise application that connects to a User Registry component and a Database component. On the right side of the pane, you define the components and tell PureApplication System what enterprise application to deploy. If you click a User Registry component, you see properties such as an LDIF file that defines which LDAP database to use.

Reference layering

Virtual Application Builder is used to create virtual application layers that provide a way to control the complexity of your virtual application and to reuse virtual applications. In Virtual Application Builder, the layering function is in the Assets panel of the Diagram view. It is at the bottom of the pane and under the components, as shown in Figure 4-41. It is collapsed by default.

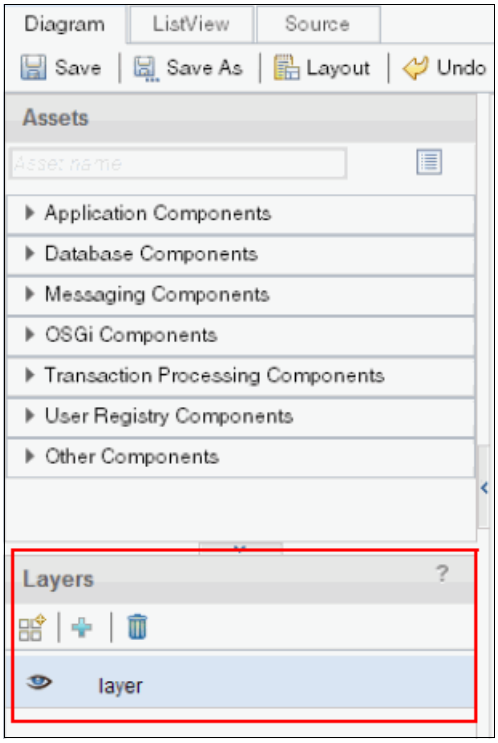


Figure 4-41 Layers capability in the Virtual Application Builder

The *layer* is a generic container in a virtual application for a collection of components. It helps you to control the complexity in the application diagram by disabling or enabling layers and to reuse the application by importing an existing application as a reference layer. By default, a virtual application consists of one layer when you first create it. When you use application layering, you can modify an existing virtual application by adding separate layers. A virtual application can contain multiple layers. A layer can contain component types of the virtual application, or the layer can reference another virtual application, which is called a *reference layer*.

For more information about creating, editing, importing, and deleting layers, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/workloadconsole/t_mngpatterns.html

Web Application Pattern

The IBM Web Application Pattern is a standardized application-centric pattern solution. It is based on common client experiences that can be reused to deploy and manage resources in the Web Application Pattern manages application deployment and lifecycle. The product extension sits on top of the IBM PureApplication System. Plug-in APIs run within the virtual application pattern to support models, patterns, binary files, and automation. A collection of existing services, such as DB2, WebSphere MQ, WebSphere Application Server, and CICS can be selected for the virtual application pattern, which allows for a customized environment.

Specifically, the Web Application Pattern provides a set of components that are typical for online web applications, like Java Platform, Enterprise Edition applications, databases, LDAP servers, and messaging. After the virtual application is built in the Virtual Application Builder, you can deploy the application and the system determines the underlying topology configuration.

The Web Application Pattern includes a scalable application server, a database, and an elastic caching component. These components are managed together as a single unit, which reduces the management and operational complexity of an end-to-end environment for hosting Java Platform, Enterprise Edition web applications. The following components are available:

- Scalable application server

  The Web Application Pattern includes a dynamically scalable application server that expands to meet the resource needs of the applications that are hosted within it. The expansion is based on processor consumption. Interaction with Web Application Pattern occurs through artifacts and policies. By providing artifacts, such as EAR files and data definition language (DDL) files, and specifying deployment and management policies, you are ready to use your application. The deployable artifact describes the operational policy for which the application server is managed. For example, you can specify that a WAR file is hosted in a highly available manner, with transactions tracked by using an external product.

- Database

  Web Application Pattern also includes a database. The use of the database component is as simple as providing a DDL file that describes the application schema. You can optionally define the amount of storage that is allocated to your database. It is also possible to connect to an externally managed database if you must use a database outside the scope of PureApplication System. However, the use of an external database removes the benefit of the solution, but the value from the remaining components still exists.

- Elastic caching

  In addition to the ability to dynamically scale the application server footprint in response to fluctuating processor consumption metrics, Web Application Pattern provides scalability by using an elastic caching component. This caching component is a distributed in-memory cache, which is highly scalable and fast.

IBM Database Patterns

IBM Database Patterns is a product extension that is used to build online DB2-based databases.

IBM Database Patterns manages a DB2 database deployment. IBM PureApplication System plug-in APIs run within the workload pattern to support models, patterns, and automation. You can select the database requirements to support typical departmental-style applications for a workload pattern. After a database is deployed, the IBM PureApplication System system determines the underlying topology configuration.

By using IBM Database Patterns, you can create and deploy DB2 databases in a database as a service (DBaaS) cloud environment. You select the database requirements that meet your needs and PureApplication System builds the underlying topology to meet those requirements.

Sometimes there is a need for databases that are independent of DBaaS offering. The following usages scenarios require an independent database:

- Separate management team
- Lifecycle independent of application lifecycle
- People or teams need database access across multiple applications

In many usage scenarios, the database is a distinct entity with its own administrative team and lifecycle. PureApplication System models this behavior with database patterns and database instances. You inform the PureApplication System of your database requirements in the form of a database pattern. Then, it builds and deploys a DB2 database instance for you, as shown in Figure 4-42.

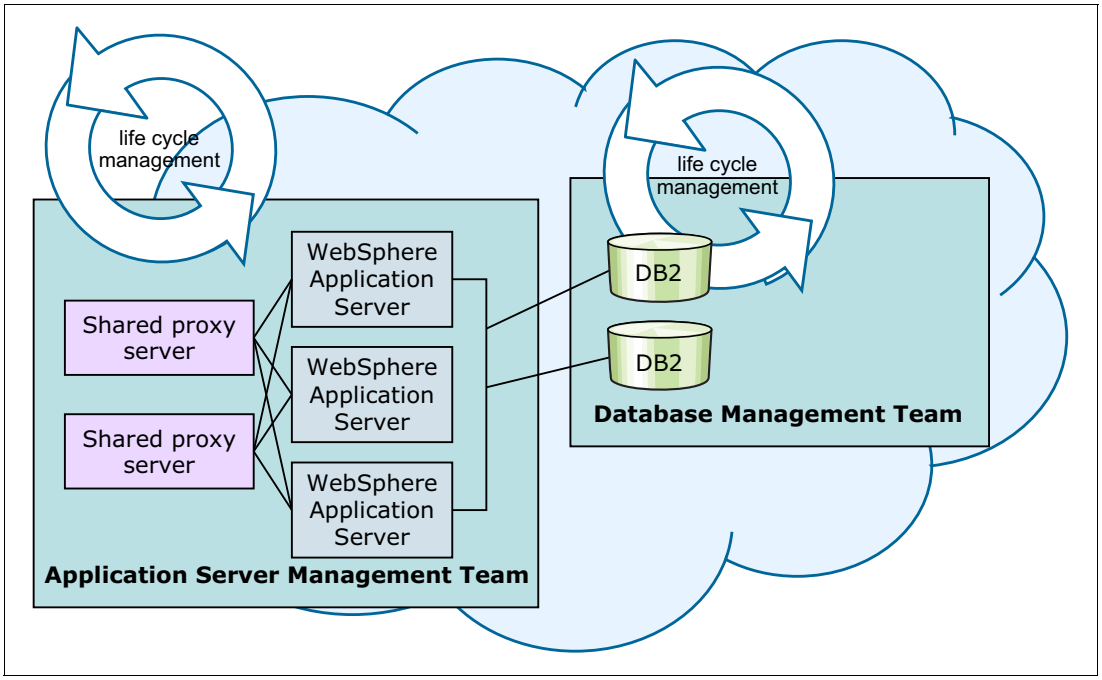


Figure 4-42 Database pattern overview in regards to separated administrative teams

Database patterns

In the context of PureApplication System, there are multiple ways to deploy or configure a database. As IBM DB2 software is integrated inside PureApplication System, the use of DB2 as a deployed application's database involves no extra costs. This configuration reduces overhead and other license tracking mechanisms and the total cost of ownership of the platform. The unified nature of DB2 within PureApplication System allows for best practices and expert-focused integration to be applied and followed throughout an application's use of DB2 as the database back end service. Like other applications that run on IBM PureApplication System, DB2 is available as a DB2 virtual system pattern or as part of a DB2 database workload pattern.

IBM Database Patterns can be created and managed independent of any virtual application pattern as DBaaS or as a Remote database (existing database component) that deploys a database pattern as part of the virtual application, as shown in Figure 4-43.

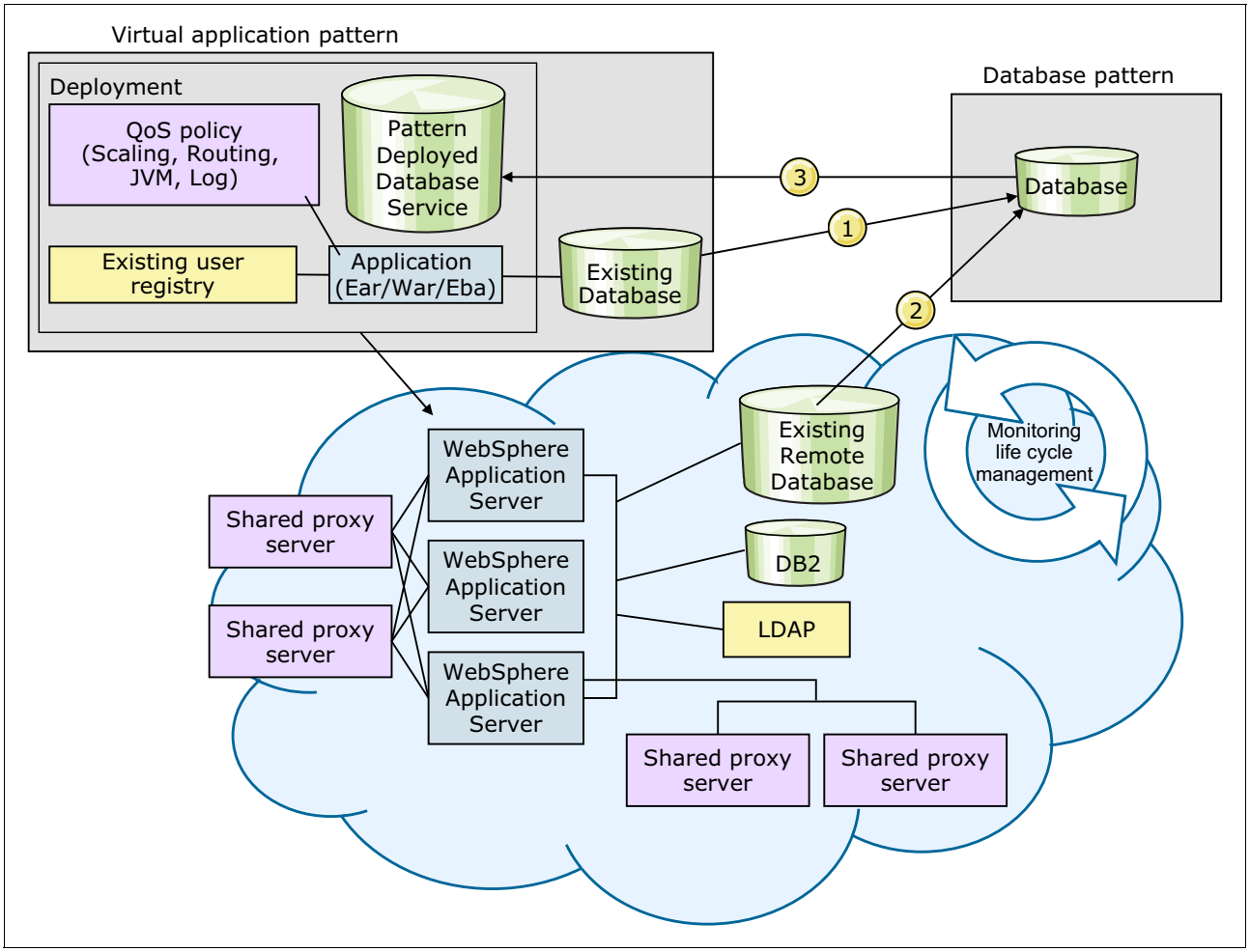


Figure 4-43 Deploy database pattern options

In most real world usage scenarios, databases are managed independently of any one application.

To more closely model this paradigm, database patterns were introduced. You can create, delete, update, backup, and restore databases that are created by using a database pattern. These management activities are independent of your virtual application. Deleting your virtual application has no impact on the deployed database or database pattern.

In Figure 4-43, label 1 shows that you can include an existing database as part of your virtual application pattern. This ability is considered an *existing database* component and is not deployed as part of your virtual application. The database exists and is used by the application that is deployed as part of the virtual application. It can be created as a database instance in PureApplication System.

Identified as label 2 in Figure 4-43, is an aExisting Remote Database that was created outside of the PureApplication System environment.

In these cases where an existing database component is being used, deletion of the database does not affect the state of the virtual application instance. Similarly, changes to the virtual application instance do not affect the database.

Label 3 in Figure 4-43 on page 162 shows it also is possible to deploy a database pattern as part of the virtual application. Instead of including an existing database component in the virtual application pattern, you can include a database component that becomes a pattern-deployed database service. In the case of a pattern-deployed database service, the database is deployed as part of the virtual application. When the virtual application is deleted, the database also is deleted.

DB2 virtual system patterns

The DB2 virtual system pattern allows for more flexibility in the control and configuration of the middleware environment. It is explained in more detail in 4.5, “Virtual Systems” on page 121.

DB2 database workload patterns

In addition to the DB2 virtual system patterns available, DB2 database workload patterns are found within PureApplication System where the configuration and best practices are applied for a specific context. The deployment of a DB2 database workload patterns is simple with the flexibility for changes to some of the configuration parameters within the database layer. The following database workload patterns are available for DB2:

- IBM Transactional Database Pattern
- IBM Data Mart Pattern
- Custom Standard

The IBM Transactional Database Pattern is designed to accommodate departmental online transaction processing (OLTP) applications that do not require high levels of database customization. This database workload pattern includes automated configuration for departmental OLTP deployment, virtual machine deployment sizing templates, and database backup scheduling. Within this pattern, the DB2 Enterprise edition is used with the Storage Optimization feature enabled for data compression.

The IBM Data Mart Pattern provides a set of capabilities that are essential to the provisioning and management of the data mart infrastructure for data-centric applications within PureApplication System. Tuned for the unique I/O throughput that is required of data mart workloads, the IBM Data Mart Pattern includes data compression capabilities and data movement tools. These tools are designed to help move the business forward with the information that is needed without delay. Within this pattern, the DB2 Enterprise edition is used with the Storage Optimization feature enabled for data compression. Included inside this pattern are the SQL Warehousing tools for creating and modifying physical models, control flows, and data flows of the target data marts.

By using the Custom Standards, you can define your own workload standard when you must customize the deployed database to meet some of the required tuning or corporate standards.

Clone support

Cloning is a provisioning approach that uses an existing database image as a model for creating database patterns. When an image is selected, the metadata that is stored during backup is retrieved. A new virtual machine is created with the same resource settings. The DB2 Restore command creates a database with the same license and configurations. This cloned database then sits on the new virtual machine. You should use manually created images in preference to automatically created backups for this task. You can manually create a database image with the Create a Database Image function in the Database Service Console.

IBM Database Patterns requires the IBM Tivoli Storage Manager server to store database images. If there is no available Tivoli Storage Manager server, database images cannot be created in IBM Database Patterns. Tivoli Storage Manager must be configured to use clone support function. Without it, you cannot take an existing database that was backed up to Tivoli Storage Manager and clone it (copying it and all its configurations exactly). For more information about how to configure IBM Tivoli Storage Manager for clone support, see this website:

http://www.ibm.com/developerworks/data/library/techarticle/dm-1206patterns2/index.html

Deployment and operations flow

Figure 4-44 shows the overall flow for deploying a database in PureApplication System.

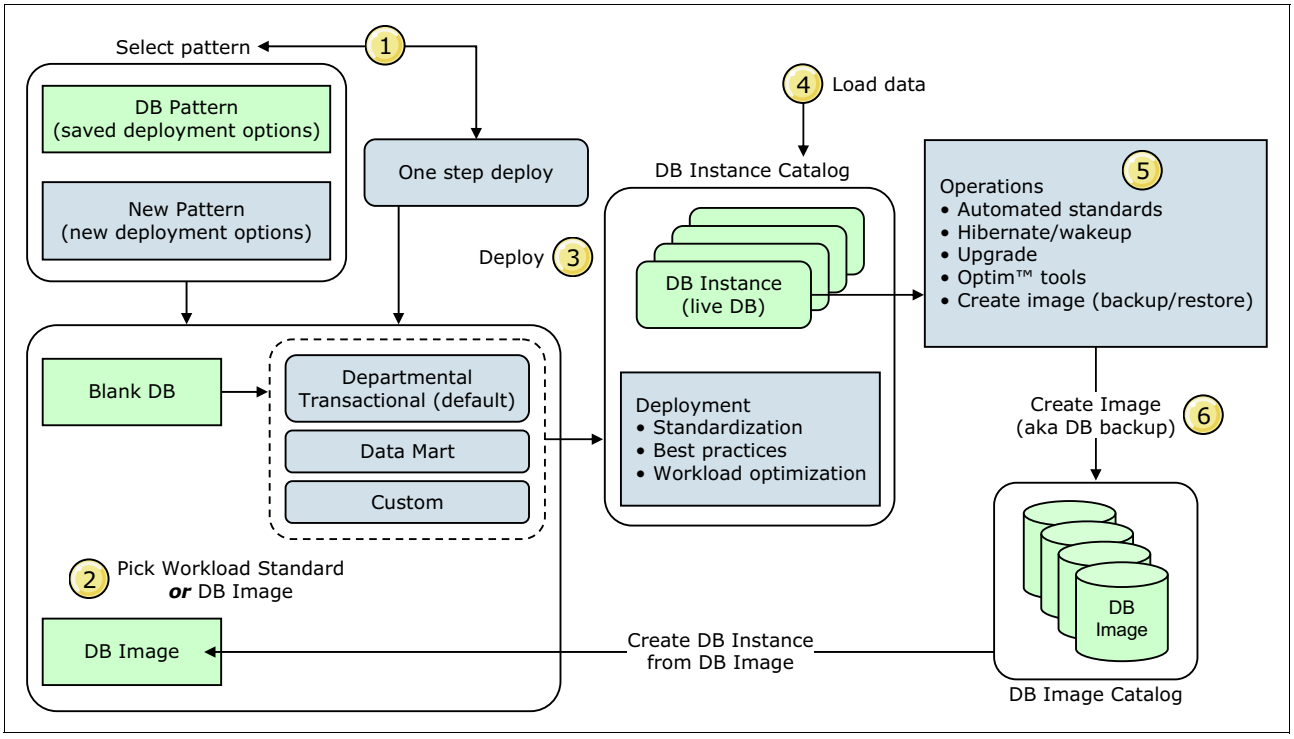


Figure 4-44 Deployment and operations flow

Label 1 in Figure 4-44, shows that you have two options to start. You can start a deployment from an existing pattern or you can perform a one step deployment by passing the pattern selection. If you use a pattern, you can use a pattern that was saved or create a pattern.

When you create a pattern you have two deployment options, as shown in label 2 in Figure 4-44 on page 164. You also are given these same two options for the one step deployment. You can select the type of workload standard to apply to the DB instance to be created or you can select to clone an existing DB image that was backed up to your DB image catalog repository. Tivoli Storage Manager is used by PureApplication System to store these DB images.

If a workload standard from which to create your pattern is selected, in the Blank DB option, you have the following options:

- Departmental Transactional

  This option is the default. By using this custom option, you can add an unlimited number of custom workload standards that define specific tuning requirements that are required for your particular applications or workloads.

- Data Mart

  The IBM Data Mart Pattern provides a set of capabilities that are essential to the provisioning and management of the data mart infrastructure for data-centric applications within PureApplication System. Tuned for the unique I/O throughput that is required of data mart workloads, the IBM Data Mart Pattern includes data compression capabilities and data movement tools that helps to move the business forward with the information needed without delay.

- Custom

  By using customized workload standards, you can define specific tuning requirements that might be required for mature applications or workloads.

After you have your pattern set up as you want, deploy the database pattern. Deploying causes a DB instance to be deployed, as seen in label 3 in Figure 4-44 on page 164. The deployment process makes many decisions for you to tailor your DB instance. The deployed DB instance includes standardization, best practices, and workload optimization that correspond to the workload standard selected.

After deployment, you have a fully functional robust database for your IT environment, as shown in label 4 of Figure 4-44 on page 164.

Label 5 in Figure 4-44 on page 164 shows some operations that are available to you from the PureApplication System Database Service Console. These options include the backup of your image to the DB Image catalog, as shown in label 6 of Figure 4-44 on page 164. This operation requires Tivoli Storage Manager.

DB2 service for cloud: Oracle or DB2 applications

IBM PureApplication System supports DB2 applications and Oracle applications, as shown in Figure 4-45.

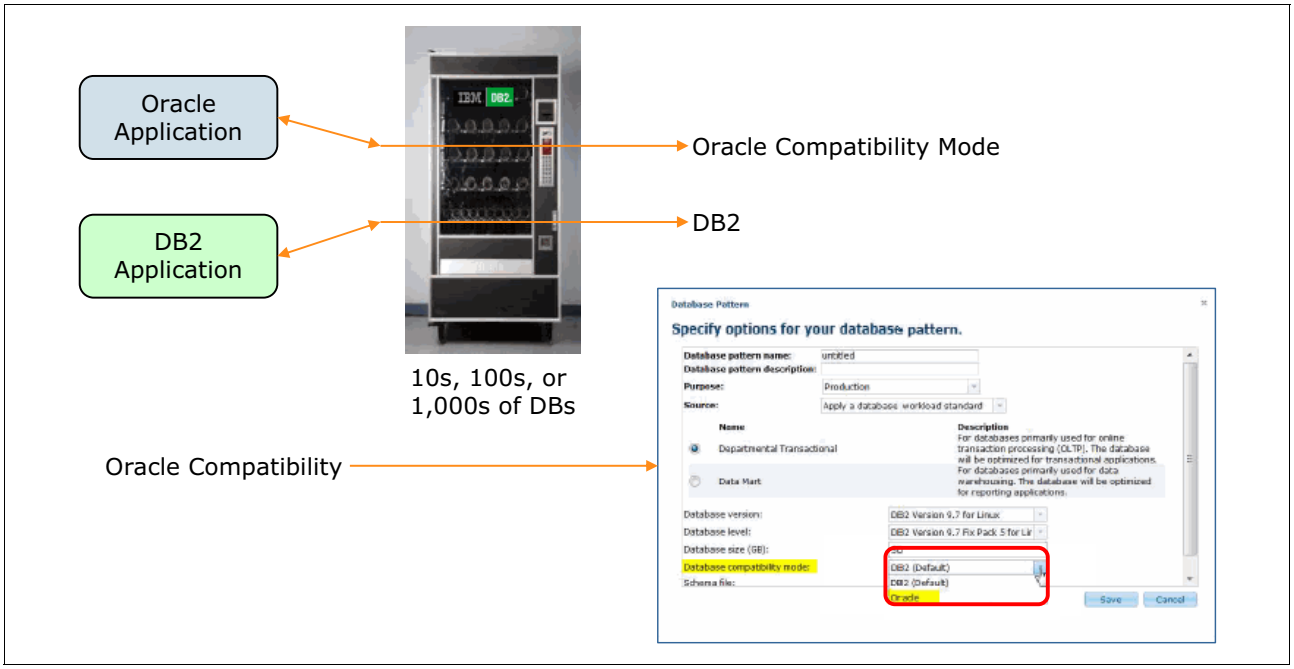


Figure 4-45 Oracle and DB2 support

In cases where you might need Oracle databases, PureApplication System supports an Oracle Compatibility mode, with which you can support your Oracle applications. When you are defining your database pattern, you can select the Oracle compatibility mode, as shown in Figure 4-45 on page 166. By using DBaaS for database deployment, a single skill set for your application group is possible.

Logical to physical mapping for database

In Figure 4-46, you can see the mapping from logical view to physical view in terms of the database patterns perspective.

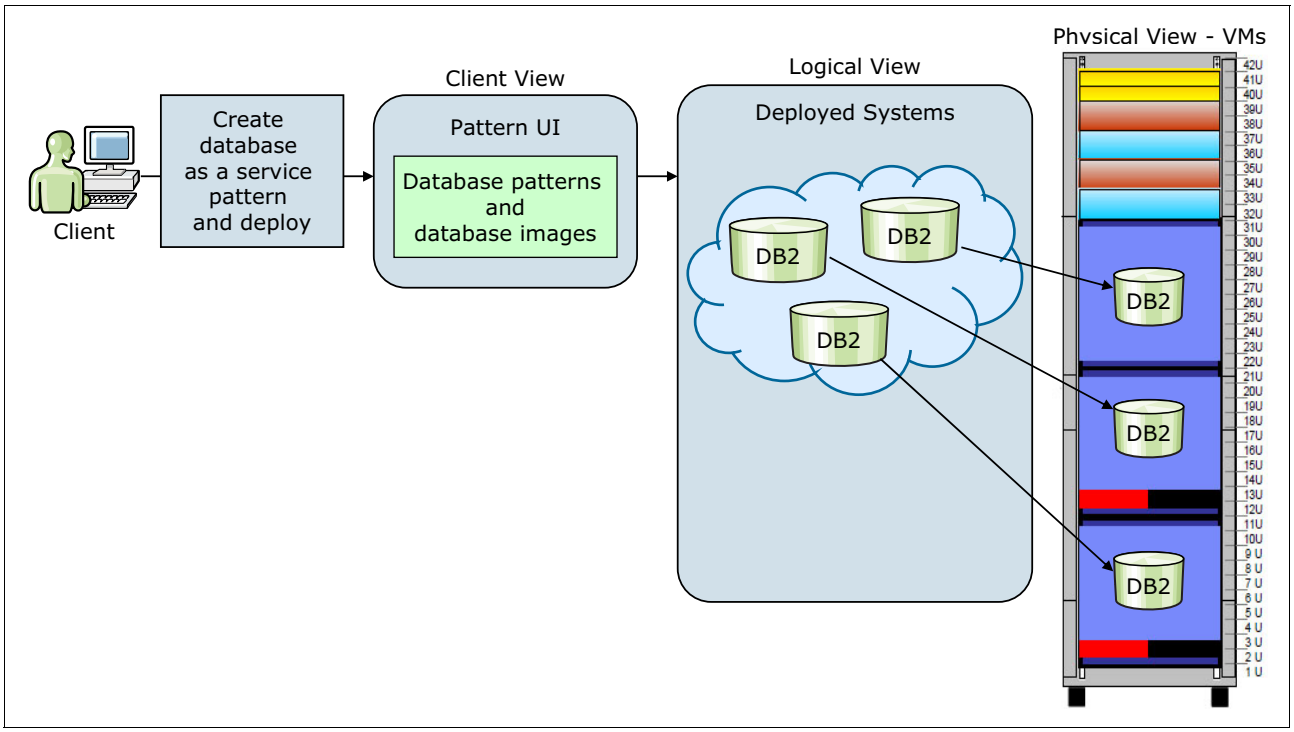


Figure 4-46 Logical to physical mapping in database patterns

Looking at DBaaS from the client view, you again must create a pattern (database pattern) and deploy it. One of the options that is available when you create your database pattern is pointing to an existing image to copy it for your new deployment. The logical view shows that a database is created for each deployment and each of those database deployments translates to a separate virtual machine that is created in the PureApplication System rack. After it is created, you can delete, update, backup, and restore the deployed databases.

Application Pattern Type for Java

The Application Pattern Type for Java is a virtual application pattern type that you can use to build Java applications. This pattern type provides an easy and fast mechanism for provisioning Java applications.

The Application Pattern Type for Java manages Java application deployment and lifecycle. This product extension sits on top of the IBM PureApplication System. Plug-in APIs run within the virtual application pattern to support models, patterns, binary files, and automation. A collection of other components allows connections to network resources, such as databases or web services, to deploy more files and to enable monitoring of log files. After the virtual application in the Virtual Application Builder is built, you can deploy the application. The system determines the underlying topology configuration.

The Application Pattern Type for Java provides an instance of IBM 64-bit SDK Java Technology Edition Version 7. By using the pattern, you can bundle an existing Java application with all of the resources it requires as a compressed archive file and deploy it into the cloud. In addition to providing the Java runtime environment, the Application Pattern Type for Java provides some pre-configuration, which implements best practices for performance and monitoring. The following monitoring and performance best practices should be considered:

- Enabling the use of compressed references
- Logging verbose garbage collection information
- Providing a mechanism for enabling the use of IBM Monitoring and Diagnostic Tools for Java - Health Center

Application Pattern Type for Java prerequisites

Before you use the Application Pattern Type for Java, verify that your hardware and software meet the minimum requirements.

The official set of hardware and software requirements is available on the System Requirements page of the product support site. If there is a conflict between the information that is provided in the information center and the information about the System Requirements page, the information about the System Requirements page takes precedence. The following requirements are for application pattern type for Java from PureApplication System information center:

- Hardware requirements:
  - IBM PureApplication System Version 1.0
  - IBM System x® servers, supporting VMware ESX Version 4.1, Version 4.0 U1 or higher, or Version 3.5 U5 or higher
- Software requirements:
  - IBM OS image for Red Hat Linux Systems (required for the use of Application Pattern Type for Java with VMware ESX hypervisors)
  - IBM Foundation Pattern
  - VMware ESX Version 4.1, Version 4.0 U1 or higher, or Version 3.5 U5 or higher

Building blocks of a Java virtual application

The IBM Application Pattern for Java provides five simple building blocks that you can use to create a Java virtual application. You can deploy your application and any other libraries or configuration files it might require, configure the incoming and outgoing firewall, and add your application log files in to the provided monitoring framework. The framework consists of the following components:

- Java application

  This application is the component that you use to supply and configure the main part of a Java application. You must provide the application itself as an archive file, such as a `.zip`, `TAR.GZ`, or TGZ file. This file must contain the compiled Java application code. After you supply the application, you declare the appropriate option for how the application runs. Then, you can use the main method (`com.ibm.sample.HelloWorld` along with any parameters and other class path entries) or a command line if a startup script (such as `/bin/start.sh`) starts your application.

  In addition to the configuration of the Java application you are deploying, you can optionally add a JVM Policy. By using this policy, you can alter the configuration of the underlying Java run time. You can set minimum and maximum Java heap sizes, enable debug mode, enable Health Center monitoring, or add other Generic JVM Arguments.

Enabling debug mode or Health Center monitoring also causes the necessary firewall ports to be opened. You can limit the range of IP addresses that are allowed to connect to the Java application.

Important: Health Center is a no cost, low-overhead diagnostic tool for monitoring an application that is running on an IBM Java virtual machine. It is available as part of the IBM Support Assistant. The IBM Application Pattern for Java uses its late attach capability so that you can start monitoring at run time.

- Additional archive file

  By using this component, you can add files to the virtual application by supplying an archive file that contains the other files. It can be a Java archive (a Java archive, WAR, or EAR file) that is deployed as-is into the specified location, or a normal archive file (`.zip`, `TAR.GZ`, or TGZ) that contains the other files that can be unpacked into the wanted location. The Java archive option is useful for adding components such as JDBC drivers or deploying a web application if the main Java application is an application server, such as Apache Tomcat. The normal archive option is useful for adding or overwriting default configuration files, such as a `server.xml` file, to change default port usage or add user IDs.

- Connect In

  By default, the deployed application does not allow any incoming or outgoing network traffic. By using the generic listener component, you can configure the firewall for incoming network traffic. This option configures only the firewall and does not cause an HTTP server to be deployed. However, you can use an IP/netmask to limit the range of IP addresses that can connect to the application.

- Connect Out

  The generic target is the counterpart to the generic listener but is used for configuring outgoing network connections. You specify an IP address or range of IP addresses to which the application can connect. For example, if the application connects to a database, a generic target is required to allow the connection.

- Monitored file

  By using the monitored file component, you can add one or more files into the Log Viewer of the workload console that is provided by the IBM SmartCloud, IBM Workload Deployer, and IBM PureApplication System user interface. The monitored file component allows the use of wildcards, and several monitored file components can be added if multiple files or multiple directories must be added to the Log Viewer. For example, `/logs/*.log` adds any file in the logs directory with a `.log suffix` to the Log Viewer.

Using Java application templates

The Application Pattern Type for Java provides some templates for deploying common applications. You must specify some information before you can deploy the application. For example, the Apache Tomcat template already contains the port that Tomcat listens on, the log directory that Tomcat uses, and the directory from which Tomcat automatically retrieves web applications. You must specify the web application to be hosted by Tomcat and the Tomcat file that you downloaded earlier.

Use these templates to quickly deploy the following applications: Apache Derby, Apache JMeter, and Apache Tomcat.

To get started, download the application that you want to deploy. The templates are set for specific versions of these applications. You can download different versions, but you must modify some of the template information. The following templates are the built-in versions:

- Apache Derby 10.8.2.2 (a Java database application)

  This application is a template for deploying the Apache Derby 10.8.2.2 database.

- Apache JMeter 2.7 (a load test application).

  This application is a template for deploying Apache JMeter 2.7 to conduct load testing. When you deploy this application to the same cloud as your Java application, you can test your application with much greater loads than if you used JMeter from outside the cloud.

- Apache Tomcat 7.0.27 (a web server application)

  This application is a template for deploying Apache Tomcat 7.0.27 and a Web Application. The required information is the Archive File entry when you specify the `.zip`, `TAR.GZ`, or TGZ file that contains the Java application that is uploaded.

For all of these templates, the following fields are required:

- Archive File: Specifies the `.zip`, `TAR.GZ`, or TGZ file that contains the Java application that is uploaded.
- Application Launch Type, one of two options:
  - Main Class: Specify the class that contains the `main()` method that uses a fully qualified class name, for example `com.ibm.myApp`.
    - Program Arguments: Specifies arguments to pass to the `main()` method.
    - Class path: Specifies other entries for the class path, uses ':' as a separator.
  - Command Line: Specifies the command line to start the Java application.

4.6.3 Simple topology instantiation

Simple topology instantiation represents a basic configuration of a virtual application. Figure 4-47 on page 171 shows that the simple virtual application topology is instantiated by PureApplication System. The virtual application pattern consists of an enterprise application component and a database component. Each of those components is deployed into a separate VM with the needed middleware and a deployer agent, which orchestrates the deployment. The middleware that is installed in the case of the Enterprise Application component is WebSphere Application Server. In the case of the database component, DB2 is installed. You see that an `.ear` file also is installed and a database is created in the respective VMs.

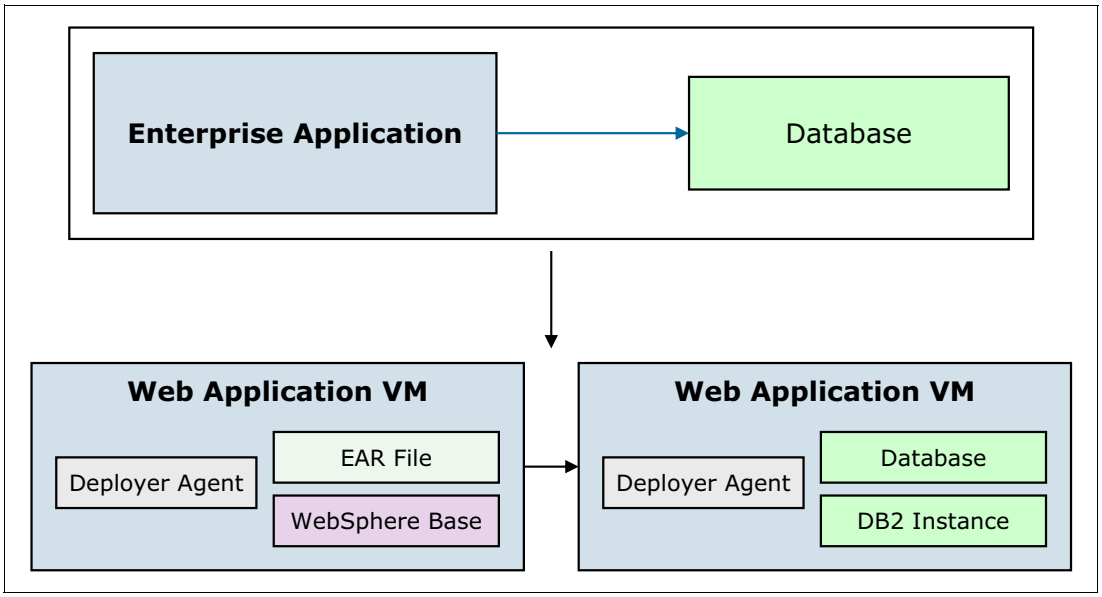


Figure 4-47 Simple topology instantiation for a Virtual Application

4.6.4 Topology with scaling policy instantiation

Shown in Figure 4-48 on page 172 is a simple topology instantiation. By adding a scaling policy to the virtual application pattern, you get a much more sophisticated instantiation. This enhancement is determined by PureApplication System that is based on your simple virtual application pattern. In this case, your application relies on two shared services: the proxy service and the caching service.

The proxy shared service provides routing and load balancing for the multiple web applications that are deployed on your behalf. The scaling policy also requires the use of the caching shared service, which allows for highly efficient caching that is based on WebSphere eXtreme Scale. The shared services are shared among all deployments within a cloud environment. They are not deployed when this pattern is instantiated. The shared services already are running and available for use, as shown in Figure 4-48 on page 172.

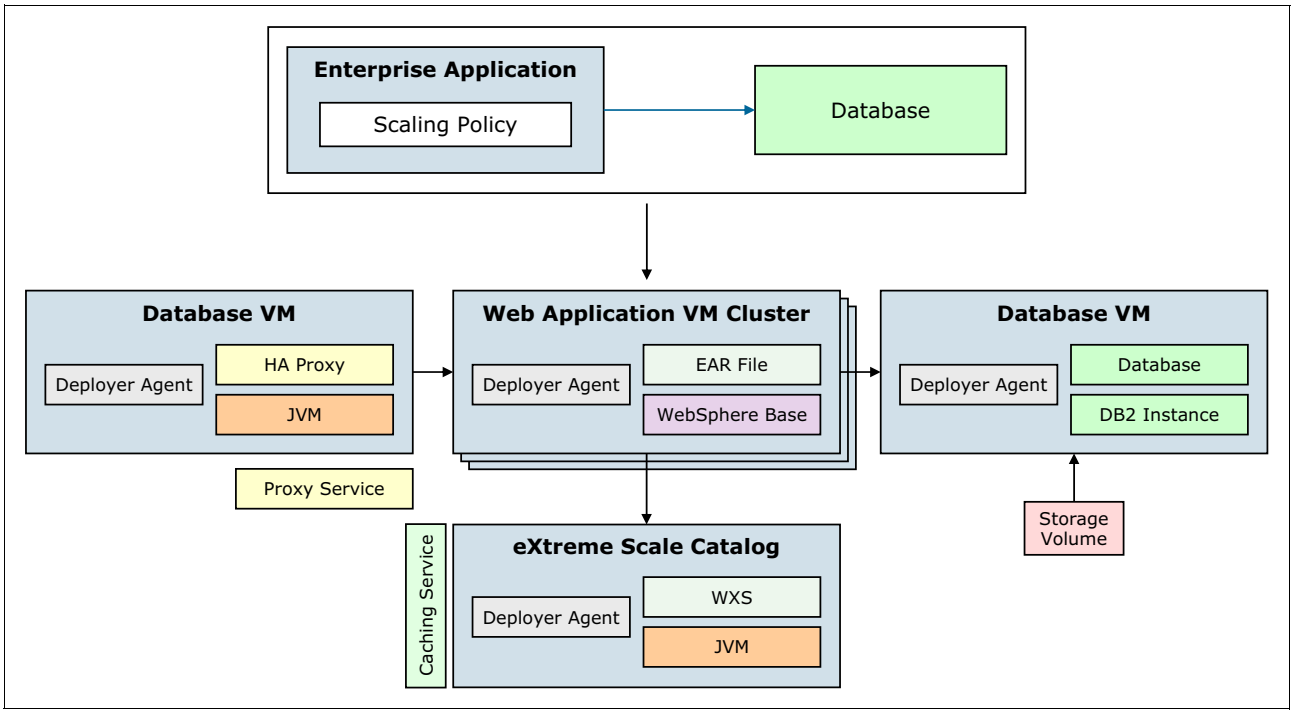


Figure 4-48 Virtual Application with Scaling policy instantiation

4.6.5 Shared services

As shown in Figure 4-49 on page 173, s*hared services* are predefined virtual application patterns that can be deployed and shared by multiple application deployments (virtual applications, virtual systems, and virtual appliances) in the cloud. They provide some runtime services to multiple applications, or services to users on behalf of multiple applications. Shared services create a simplified consumer (users/application deployments) and provider (implementation/shared service deployment) model. Shared services often are installed as part of the Foundation Pattern.

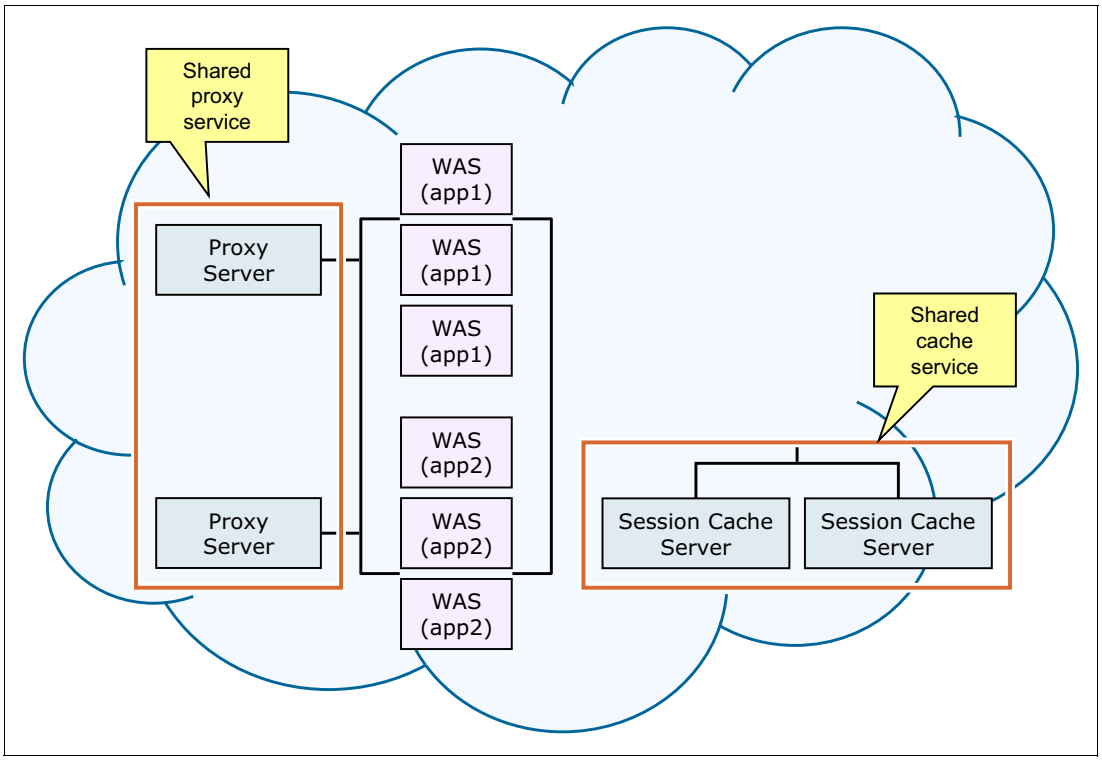


Figure 4-49 Shared services overview

Shared services provide a predefined virtual application pattern that is deployed and shared by multiple application deployments. These application deployments include virtual applications, virtual systems, and virtual appliances in the cloud.

To view shared services, you must be assigned the Workload resources administration with full permissions or the Cloud Administrator role.

IBM PureApplication System allows virtual applications to use a common (or shared) set of services. These services are used to proxy HTTP requests, cache session data, and monitor components of the virtual application. When they are deployed, these services are shared between all virtual applications within a cloud group. Each cloud group must have its own instance of a shared service for it to be available, which means that only one instance of a type of shared service can be deployed in a cloud group (physical group of hardware that defines a cloud). This shared service can be used by all application deployments in the cloud group, which means that the shared service must provide multi-tenant access. Virtual applications that enable a routing policy use the shared proxy service. Those services that enable session caching in a scaling policy use the shared caching service. These shared services offer automatic failover, a reduced resource footprint in the cloud, and improved performance.

Shared services provide the following features:

- A plug-in design service that provides full lifecycle management capabilities in the same way as a virtual application.
- Specific runtime services to multiple applications, or services to users on behalf of multiple applications.
- A simplified consumer (users/application deployments) and provider (implementation/shared service deployment) model.
- Viewed as a multi-tenant service.

Shared services are used in the following ways:

- Directly exposed by a resource in the Virtual Application Modeler by using routing policy.
- Indirectly exposed though a setting that implies the usage of a shared service, such as HTTP session caching inside scaling policy.
- Injected though transformation process, such as injecting the Logging service client.

Caching Service 2.0.x

The IBM PureApplication System hosted *caching service* is a shared service that is deployed in the cloud to allow other deployments from Workload Deployer to use common cached information. It enables in-memory cached objects in virtual applications.

Virtual applications share cache service in the cloud group to which they are deployed. Sharing the cache service function reduces the footprint of resources that are required for each virtual application. The smaller footprint is a result of the fact that applications do not have to maintain their own memory overhead to support the cache. Caching service also eliminates redundant virtual machines to support high availability.

The caching service is not only for session replication. A virtual system can use the caching service for sessions, as a dynamic cache or as a simple object grid. The caching service is based on WebSphere eXtreme Scale code and provides highly efficient caching. The caching service is self-managed and highly available, providing simple, and quick usage.

To deploy a caching service that is hosted as a shared service, complete the following steps:

1. Browse to **Cloud** → **Shared Services.** Click **Caching Service 2.0**, as shown in Figure 4-50.

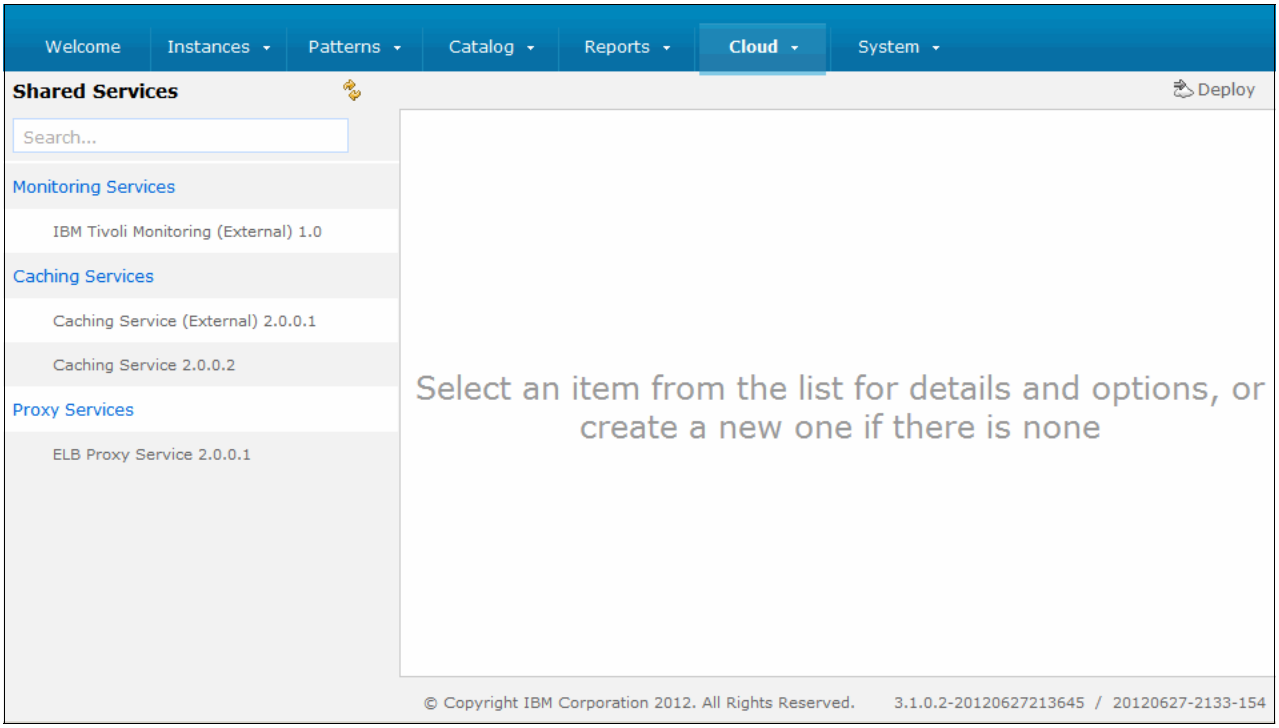


Figure 4-50 Caching Service 2.0.x

2. Click the **Deploy** icon.

3. Specify the instance size, number of instances, maximum number of instances, and the rules for auto-scaling (if automatic scaling is enabled), as shown in Figure 4-51.

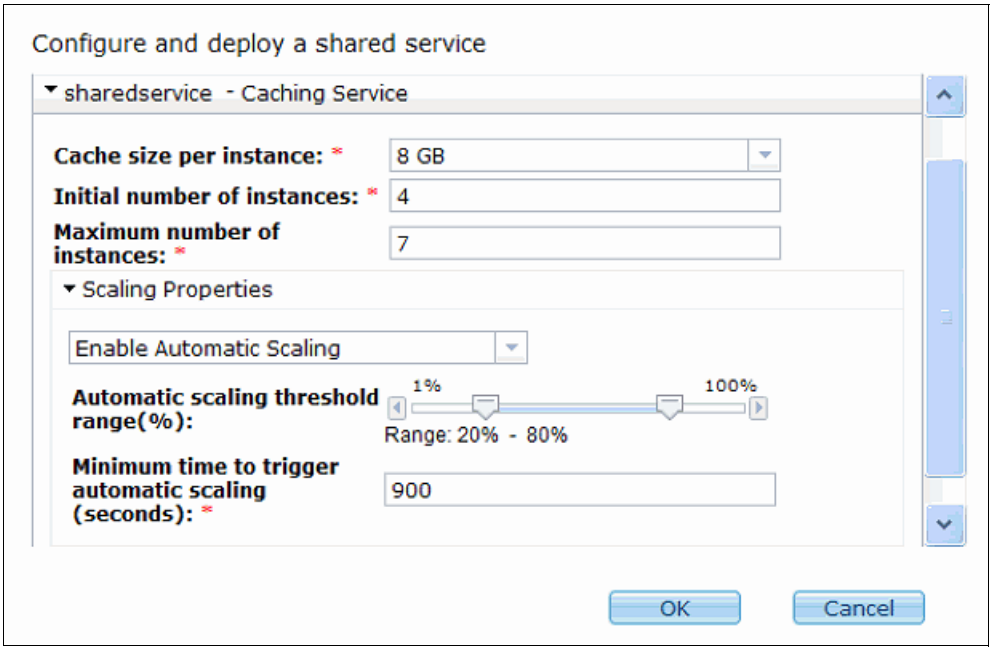


Figure 4-51 Caching service properties

The following settings determine the size and initial number of virtual machines in the cloud that are devoted to the caching service:

– Instance information

  The initial number of instances provides the minimum number of instances that share in session persistence and provide failover. For example, if you select 8 GB for size per instance and four initial instances, the caching service deploys with four virtual machines. Each machine can each handle 8 GB of caching information, for a total capacity of 32 GB. The information about each instance is replicated automatically to other caching virtual machines.

  The estimated cache grid provides a total of 32 GB in this example. The total virtual machine instance size is larger than 32 GB because of the administrative and OS memory requirement additions.

– Scaling information

  When you choose Enable Automatic Scaling, you must specify the following settings:

  - Automatic scaling threshold range percentage

    Use the slide rule to define the automatic scaling range when the capacity is outside the limits. The lower capacity limit is for scale down and upper capacity limit is for scale up.

  - Minimum time to trigger automatic scaling

    To trigger automatic scaling, the capacity must exceed a specified range. You specify the minimum range (in seconds) that must be exceeded to trigger the automatic scaling.

If you choose to Disable Automatic Scaling, the scaling properties are removed from the deployment pane, as shown in Figure 4-52.

- Automatic Scaling Disabled

  You must manually scale out and scale in the cache instances from the Virtual Application Console up to the maximum number of instances.

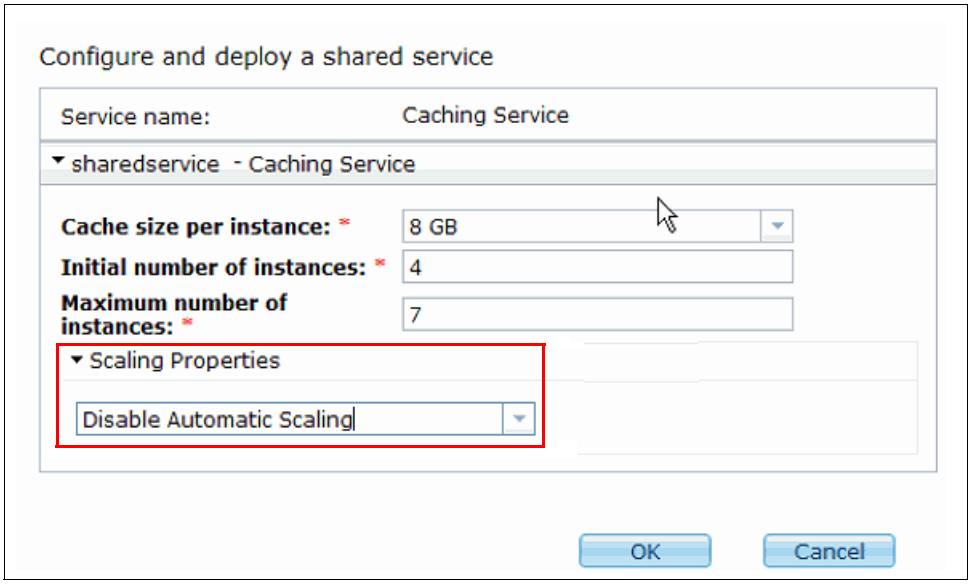


Figure 4-52 Disabling automatic scaling in the caching service

4. Complete the deployment by providing the cloud group information for the service and click **OK**.
5. You can view the new shared caching service virtual machines by selecting **Instances** → **Shared Services** and then selecting the shared service instance, as shown in Figure 4-53. You can monitor the deployment status in the virtual machine perspective section. The role status is caching when the instance is running.

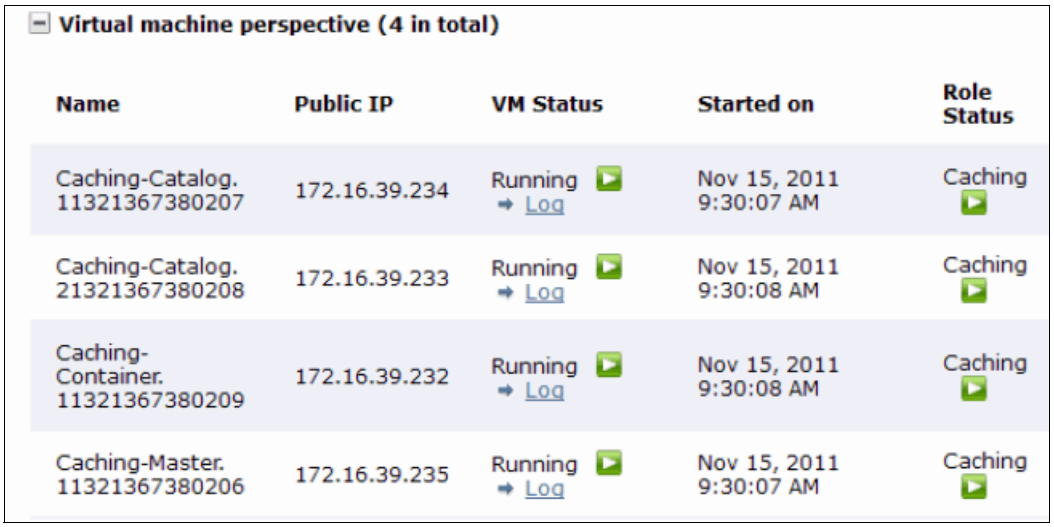


Figure 4-53 Caching service virtual machines

In Figure 4-53 on page 177, you see that the virtual machines have different functions as indicated by the name (catalog, master catalog, and container). Cached data is stored in the containers. The catalog service maintains topology information for the containers and controls balancing and routing for all clients. The last virtual machine to reach a running state is the master.

Managing the service

You can use the Virtual Application Console to manage the caching service. To open the console, click **Manage** from the shared service instance. If automatic scaling is disabled, use the scale out and scale in operations to increase and decrease the number of cache instances in the cloud. You cannot have automatic scaling enabled and also manually scale out and scale in because these operations conflict with the automatic scaling rules. You also can use the Virtual Application Console to manage grid caching directly. If you are using a virtual application pattern with a scaling policy and shared services, the grid is managed and configured for your WebSphere Application Server automatically. You can manage the grid in the cloud yourself from the Virtual Application Console in the following cases (see Figure 4-54):

- If you are not using a virtual application pattern (that is, you are deploying a WebSphere Application Server virtual system) and want to customize that installation for caching manually.
- If you have an application that has direct WebSphere eXtreme Scale appliance grid operations.

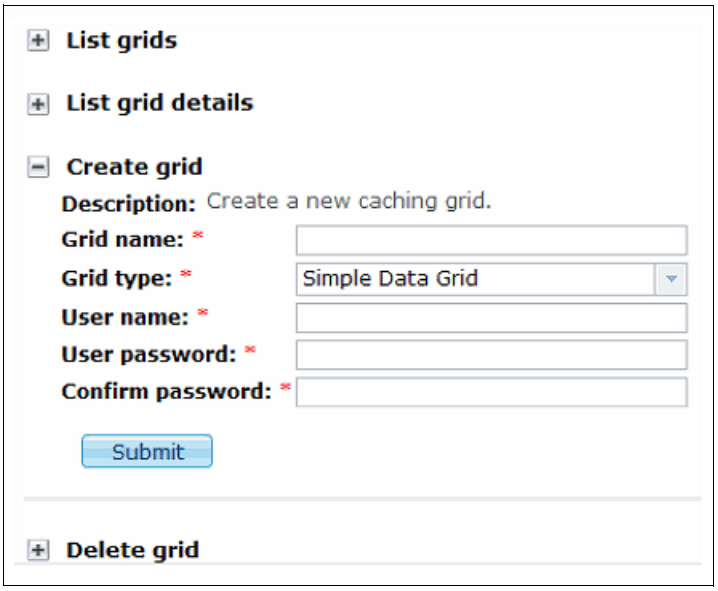


Figure 4-54 Virtual Application Console grid operations

Grid caching maintains data that can be accessed from multiple clients, which minimizes network latency and reducing bandwidth. You can set the following options when you are using the caching service to configure grid caching:

- Create grid: Create a grid to maintain cached data. When a grid is created, you must provide a name, specify the type of grid, and assign the grid an ID and password.
- List grid: Return a list of all of the grids that exist in the caching service.
- List grid details: Return the details of a specific grid.

- Delete grid: Delete the specified grid. If you choose to delete a grid, all of the cached data on that grid is deleted. This action cannot be undone. Deleting a grid also deletes the user ID that is associated with the grid.

Caching Service (External) 2.0.x

You can also connect to an external caching appliance collective with by browsing to **Cloud** → **Shared Services** → **Caching Service (External) 2.0.x.** Click **Deploy** to display the required parameters for the external cache, as shown in Figure 4-55.

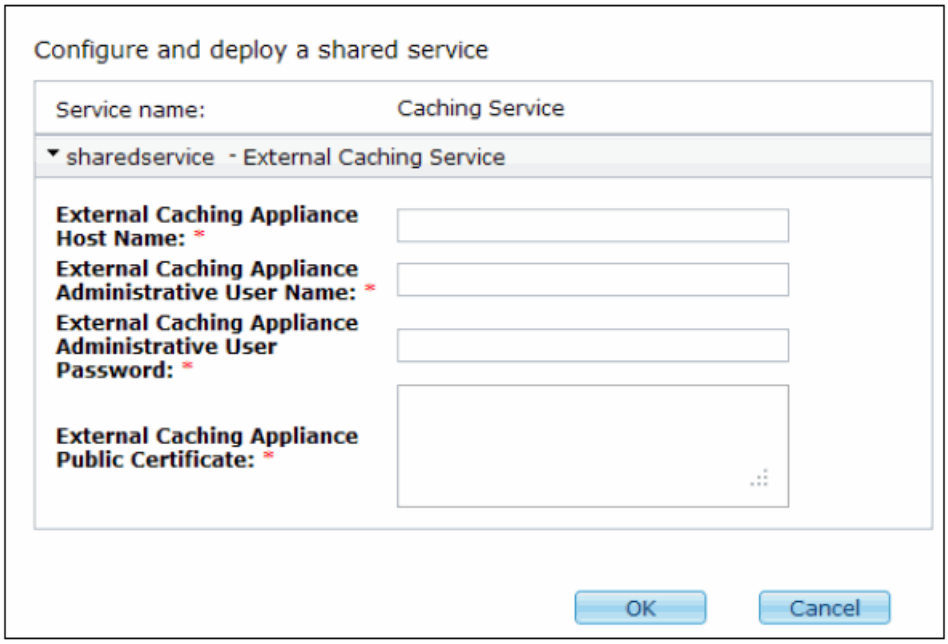


Figure 4-55 Configuring an external caching service

In this pane, you can point to an external WebSphere DataPower XC10 Appliance. The following parameters are required to point to an external appliance:

- External Caching Appliance Host Name
- External Caching Appliance Administrative User Name
- External Caching Appliance Administrative User Password
- External Caching Appliance Public Certificate

You can obtain the External Caching Appliance Public Certificate by using a browser to access the XC10 appliance and then downloading the public certificate. When you deploy an application with a scaling policy that is configured to a cloud with an external caching service, the application caches according to the configured external service settings.

ELB proxy service

The primary benefit of enabling a proxy service is that the IP addresses that are used internally are not visible externally on the web. A typical environment has a first tier (a well-known host such as `www.ibm.com`). The first tier sprays requests to a second tier of internal servers that host the application and content (protected and secured). The IBM PureApplication System ELB proxy service is on the second tier. The multiple ELB instances, set up by the service, are the internal IP addresses that the first tier sprays with requests.

The ELB proxy service provides a front end to virtual applications in the cloud by balancing the load across the instances of virtual applications. The ELB shared service is shared by virtual applications that are deployed to the same cloud group. Removing the redundant component from each virtual application that participates in sharing improves cloud density. Requests are routed to an application that is based on the protocol (HTTP and HTTPS) and the application's host name. Complete the following steps to enable the ELB proxy service:

1. Browse to **Cloud** → **Shared Services**, select **ELB Proxy Service 2.0**, and then click the **Deploy** icon.
2. Configure the number of initial instances, as shown in Figure 4-56. The initial number of ELB instances values is how many ELB instances should be created. This number indicates the number of virtual machines that share the responsibility of load balancing and provide failover. The default value is 2.

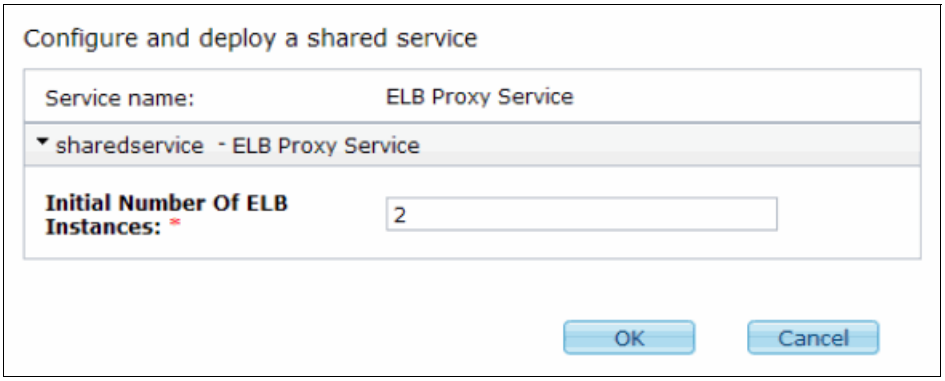


Figure 4-56 Configuring ELB proxy service

3. Select **Instances** → **Shared Services** and select the service to monitor the deployment. The Deploy Virtual Application page shows the target environment profile or target group, as shown in Figure 4-57. Wait for your ELB service to reach the running state to be ready for load balancing and fail over. If something fails, you can see in Log link the reason for failure.

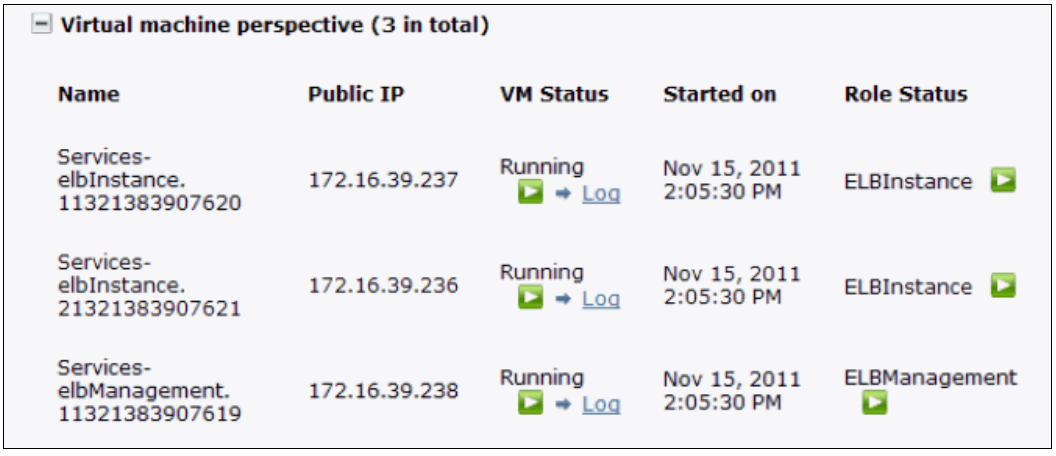


Figure 4-57 ELB service virtual machines

The ELB instance virtual machines are the actual load balancer virtual machines. The ELB manager is the virtual machine for all ELB-related operations and management. After the ELB proxy service is deployed, click **Manage** from the instance to open the Virtual Application Console, which is shown in Figure 4-58.

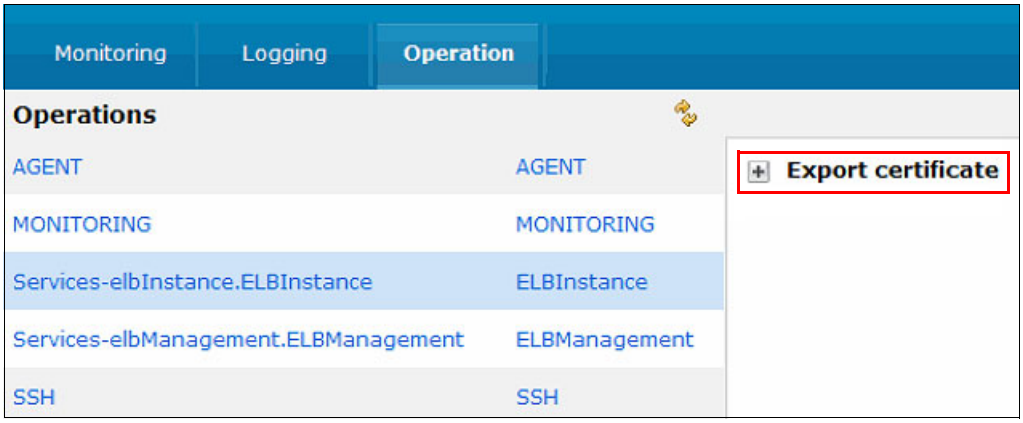


Figure 4-58 Management operation options

For the ELB instances, you can export the server certificate or root signer certificate, as shown in Figure 4-59.

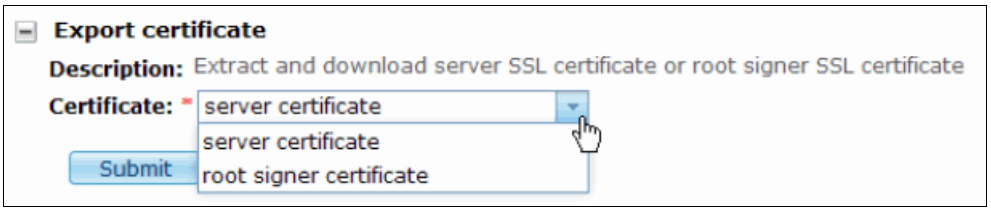


Figure 4-59 Performing operations on the ELB cache service

In the ELB management virtual machine, you can scale in or scale out the number of ELB instances you provisioned for the shared service, as shown in Figure 4-60.

Figure 4-60 Manual scaling the ELB instances

Monitoring

IBM PureApplication System allows virtual applications and systems to use a common, or shared, set of services to provide advanced monitoring capabilities. When deployed, these services are shared between all virtual applications and systems within a cloud group. Each cloud group must have its own instance of a shared service for it to be available.

Links for the system monitoring shared service open a new window by using Java WebStart. At this time, only 32-bit Java is supported. The default view for a virtual application provides a topology view of the components of the application. When you click the Endpoint link for the System Monitoring shared service, the monitoring service opens to the topology view for the shared service. From the topology overview, a deployer can drill into the operating system metrics for each virtual machine that is included in the virtual application.

For more information, see Chapter 8, “Troubleshooting PureApplication System Environment” on page 329.

Media shared service

Often a virtual application deployment requires some form of prerequisite binary media to install (such as binary files on a build server or ISO DVD images). There might be reasons not to package these binary files with the application pattern or components plug-ins. Some of these reasons are licensing, code distribution concerns, or the binary must be updated independently of the application pattern.

In a cloud environment, multiple application deployments might need to reference the same media files. If the required files are on a slower (remote) network outside the cloud, it can take considerable time to download the files into each deployed virtual application.

To solve these problems, we suggest creating a media shared service that hosts a binary file repository for use by any virtual application in the cloud group. The media service can be deployed and loaded with prerequisite files independently of the virtual applications and plug-ins.

Rather than copying files from something outside the cloud each time new application deployments are started, you can copy the files once into the media service VM, which makes them locally available in the cloud. By using this configuration, the administrator can manage this service in a more centralized design versus independent VMs that share no common framework.

The file copy speed from the media service to the virtual application is much faster. It is internal to the rack (the PureApplication System) and cloud group, which allows multiple deployments to quickly access the data.

The following example is a typical flow for the media service:

1. Deploy the shared service/media service.
2. The media service VM starts and is loaded with binary files to share via NFS on the cloud group.
3. Deploy a virtual application that uses the media service (component flag or policy in the pattern).
4. The virtual application asks the media service for the binary file locations or connection information.
5. The virtual application can network file system (NFS) mount the file repository from the media service VM.
6. Repeat the process for multiple virtual application deployments. Each deployment is given access to the shared files via the media service.

A Plug-in Development Kit (PDK) is the product in IBM PureApplication System, which provides a strong starting point to create shared services. For more information, see this website:

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

For more information about PDK usage, see Chapter 6, “Customizing Virtual Application Patterns” on page 267.

4.6.6 Benefits and trade-offs

To understand what is best for you or your organization, it is useful to review the following benefits and trade-offs of the use of PureApplication System:

- Simplicity: Virtual applications are simple to create, deploy, or monitor. Virtual applications are the quickest way to deploy applications without worrying about the infrastructure configuration. PureApplication System handles deploying applications for you, which reduces much of the complexity.
- Easy-to-use interface: PureApplication System provides an easy-to-use graphical interface to build virtual applications with drag-and-drop features and easy view of over all solution.
- Ease of configuration: PureApplication System handles the infrastructure configuration, set up, and tear down. Having PureApplication System handle the configuration also reduces configuration and deployment time errors, which helps reduce your application deployment time.
- Savings: Significant time and resource savings through the reduction in configuration and deployment times, middleware skill requirements, and configuration and deployment time errors. These savings reduce the required time that is needed to configure and deploy your applications.
- Scalability: Using policy-based features guarantees dynamic scalability. While the virtual application deployment model provides scalability for your web-based applications, it completes this task by using a single server topology.
- Built in shared services available for all virtual applications: Dynamic registration to shared services is a built-in feature in IBM PureApplication System. It provides a simplified consumer (users/application deployments) and provider (implementation/Shared Service deployment) model.

The use of PureApplication System includes the following trade-offs:

- Less control over your environment’s configuration: One potential trade-off with the virtual application deployment model is the lack of control over your environment’s configuration. If you require a specific configuration for your application, you might not be able to use the virtual applications deployment model because of the limited number of customizations available.
- Not ideal for Tier 1 applications: The PureApplication System is ideal for many Tier 2 and Tier 3 application scenarios, but not for Tier 1 applications. While there is some failover capability built into virtual applications, the database components feature limited failover. As a result, this deployment model is not ideal for Tier 1 applications at this time where a high degree of failover capability is required.
- Only web-based applications are supported: Currently, only web-based applications are supported in the virtual application deployment model. Because the full WebSphere Application Server programming model is not supported, you might not be able to use virtual applications for your deployments if your application uses some unsupported functions. For example, remote enterprise beans are not supported in the virtual application deployment model.

- Does not support full scalability and failover: While the virtual application deployment model provides scalability for your web-based applications, it provides this scalability with a single server topology. As a result, the full scalability that is available in the Network Deployment configuration is unavailable. This unavailability means that QoS features such as the HA manager, Core Groups, and Data Replication Services (DRS) are not supported in this environment.

However, the rule of thumb is to review your particular application to see whether you can use this particular deployment model. For more information about application migration, see 4.7, “Implementation strategy” on page 184.

4.7 Implementation strategy

When you consider migrating or creating an application, it is important to understand why you must put your application in a cloud infrastructure that is provided by the IBM PureApplication System. One of the reasons is to support fast business response and workload demands by using a cloud environment. With this reason in mind, you can enable cloud service providers to serve your application as PaaS by using infrastructure, platform, and application patterns in a fast deployment configuration with provisioning.

Based on a cloud environment approach to migration, whether you create an application or migrate an existing one (including code, middleware, and database) consider the following implementation strategy. The best approach for migration is to understand there can be changes in an application lifecycle. Therefore, one solution might not always be applicable.

You should base your decision on your requirements and management needs for the particular application that is considered. Virtual systems can be the best strategy.

For each application, you decide whether the middleware infrastructure-centric approach of virtual system patterns or the application-centric approach of virtual application patterns works best for your organization. You might be driven by the need to support specific configurations that do not easily fit into an existing virtual application pattern type. In that case, you might choose to create your own virtual application pattern type or use a virtual system pattern to create the exact topology that your application requires. This consideration could include replicating a physical environment that you previously implemented. In other cases, you might find that your application fits well into one of the virtual application pattern types that are already provided.

Whenever possible, use the optimization and convenience of a virtual application pattern because this pattern always provides the lowest total cost of ownership and shortest time to value. However, there are scenarios where you require detailed configurations and therefore decide to use the detailed control that is available with virtual system patterns.

The most important thing is to understand all of the options and make an informed decision. Review your use case, understand what is available to help you accomplish that use case, and then decide on what you want your user experience to be.

It is also important to note that IBM PureApplication System supports both of these models concurrently. You can have a mix of virtual applications, virtual systems, and even virtual appliances that are all deployed to the same pool of cloud resources. The robust capabilities that are built into IBM PureApplication System enable these various deployments. By using these capabilities, you can choose the deployment model for each application that delivers the best fit with the highest return on investment.

4.7.1 Questions for consideration in application migration or creation

There are some important questions to consider when you are deciding to use virtual application or virtual system in application migration or creation. This section reviews those questions and provides more information to help you frame what option might work best in your organization. Consider the following list of questions:

1. Are you building a new application?

   The simplest deployment model that is offered by PureApplication System is the virtual application. If you are building a new application and can influence the technology and design choices that are made in the application, choose the technology and design that makes an application compatible with a virtual application.

   However, in most cases, the applications that you deal with daily are not greenfield applications. Instead, you often deal with an existing pre-built application that runs in an existing environment. You then must consider the next question in this series.

2. Is this application a web application?

   What is meant by this question is: Does this application take requests on inbound HTTP or HTTPs only? This more specific question incorporates various patterns in application development. The answer can cover any of the following application types:

   – An application that provides RESTful services to a user interface that is written by using JavaScript and Ajax technologies.

   – A Web Services provider that implements SOAP services for external clients on the Internet.

   – A classic web application that is built with servlets and JavaServer Pages.

   However, this definition excludes some application types. For example, consider a Java client/server application that uses a Java thick client that connects through RMI or RMI/IIOP to Enterprise JavaBeans in a back end. This type of application is not defined as a web application that uses this definition. That consideration also leads to the next question.

3. Do you use remote Enterprise JavaBeans?

   Enterprise JavaBeans are a useful part of the Java Platform, Enterprise Edition programming model almost since its inception. However, the benefit of remote Enterprise JavaBeans is balanced by a trade-off in the complexity of your application topology. Your application servers must handle incoming HTTP traffic to your servlets, JavaServer Pages, and Web Services, and incoming RMI/IIOP traffic from the Enterprise JavaBeans clients. Usually, this configuration is accomplished through building two tiers of application servers. One tier is dedicated to handling HTTP traffic, and the other is dedicated to handling RMI traffic. As part of the simplification process that uses virtual applications, you must give up some of these topological options. If you need remote Enterprise JavaBeans, use virtual systems where these topology options are available for your use.

4. Is your application packaged in a standard way?

   In this case, standard packaging is considered packaging as an EAR file, WAR file, compressed archive, or an OSGi Enterprise Bundle Archive (EBA). Although the Java Platform, Enterprise Edition standard is to package applications as EAR or WAR files, and the OSGi Standard introduced EBA archives, many applications are still not packaged that way. Instead, they are shipped as exploded directory structures. While that format might work for simple servers such as Tomcat, a nonstandard way makes it complicated to move to new Java Platform, Enterprise Edition servers, such as the ones that support virtual applications. It is best to repackage your application in one of these standard formats. There are many other packaging strategies in WebSphere Application Server that you might want to use; for instance, server associated shared libraries. Again, to simplify the model, these strategies are not used in virtual applications. If you cannot avoid the use of these approaches, then consider the use of virtual systems instead.

5. Is your application using standard Java Platform, Enterprise Edition programming models?

   An old phrase states that “One of the great things about standards is that there are so many of them to choose from.” Unfortunately, this statement is true when you are speaking about programming models. New APIs are introduced at a rapid pace. The Java community process is littered with the remains of Java Specification Requests that were never approved or never gained wide enough acceptance to officially become part of the Java Platform, Enterprise Edition standard. The problem is that with virtual applications, you want to keep things simple. Therefore, you must restrict the set of APIs that are supported to a manageable set. If your application uses only standard APIs from JEE5, J2EE (1.4, 1.3, or 1.2), OSGi, JPA, JAX-RPC, JAX-WS, and JAX-RS, you should be fine.

   If you are writing to a newer Java Platform, Enterprise Edition level (such as Java Platform, Enterprise Edition 6), or are using some obscure API from deep within the bowels of the Java Community Process, your application probably does not work as a virtual application. IBM takes the approach to offer support for newer APIs through Feature Packs. If you are planning for a new API level, consider building a virtual system by using WebSphere Application Server V8 and incorporating the Feature Pack (and support for that API) when it becomes available.

6. Does this application run on WebSphere Application Server Version 7 or Version 8?

   There are different answers to this question to consider. If your answer to this question is yes, but you fall into one of the previous categories that addressed the programming model, packaging, or use of remote Enterprise JavaBeans, your decision is already made for you. Your application cannot be a virtual application. If you answer no, you might still be able to run as a virtual application. If your application is compliant with the previous programming model questions and packaging questions, there is a possibility to run as a virtual application. If your application is running on a much earlier version of WebSphere Application Server or on another application server, there is another consideration. You might have a migration effort to complete first before you can move to a virtual application or a virtual system.

7. Does your application require any WebSphere family products like WebSphere Portal Server or WebSphere Process Server?

   The virtual application approach is targeted at building Web applications. If your application type (or workload) is not a web application, the current virtual application approach is not appropriate for you. This statement is purely a point in time statement. New workload types will be added to PureApplication System over time. Also, even if you have a business process management application, you can take advantage of the higher levels of automation that virtual applications offer to web applications today. However, at this moment, if you require any of these products to support your business process application such as WebSphere Process Server or you web application such as WebSphere Portal, you must build a virtual system. These patterns can be obtained separately through PureSystems Centre.

8. Is your application ready to take advantage of session management with WebSphere eXtreme Scale?

   Essentially, you must first consider your use of the HttpSession API. Many applications are written in a stateless way and do not use the HttpSession API. Those applications are perfectly suited for virtual applications. However, if you are using HttpSessions in your application, you must consider how you use them.

   Review if all the contents of your HttpSession are declared to be `java.io.Serializable`. If not, you have an issue to manage. The model that scaling policies follow for virtual applications is designed so that application server instances can be dynamically created and deleted as needed, which is done to handle the amount of load that an application is taking on at any time. If you assume that your server is long-lived and its memory is a safe repository for session information if you try to implement a virtual application, your application cannot work as designed.

   Likewise, if your sessions are large (hundreds of megabytes), you can have problems by the time it takes to load a session over the network from the WebSphere eXtreme Scale grid. If you have small, serializable sessions that are compliant with HttpSession best practices, you can use virtual applications. Ensure that your HttpSession objects are small and serialized. If they are, you can use virtual application pattern. If they are not, you must change your code to use virtual application or you must use virtual system pattern.

9. Does your application use an external security product or does it use special security APIs such as Trust Authentication Interceptors (TAI) or Java Authentication and Authorization Service plug-ins?

   Consider the security requirements that are placed on your application. If your application does not have security requirements or uses WebSphere security such as local OS security, you can implement your system as a virtual application. Also, if your WebSphere security uses one of the supported user registries (IBM Tivoli Directory Server, Microsoft Active Directory), you can also implement your system as a virtual application. However, if you use a separate security product such as Tivoli Authentication Manager, a competitor product, or any of the special WebSphere Security features like Java Authentication and Authorization Service or TAIs, you must plan to build a virtual system.

10. Consider the scenario: Your application does not conform to the constraints of any PureApplication System virtual application patterns. Your application also does not have a complete, reusable, reliable set of deployment and configuration scripts. What is the best strategy?

    The best strategy for this situation is to use virtual system patterns because this pattern type allows a wide set of customization options. You must create scripts to complete your configuring work. Another option is to use Advanced Middleware Configuration to capture your environment and create repeatable and deployable virtual system patterns.

Picking the correct approach at the optimum time

Applications have lifecycles and a single deployment model might not hold for the entire lifetime of the application. For instance, you might want to deploy an application in your development and test environments as a virtual application. This deployment is the simplest model and ensures that you correctly capture the configuration parameters (like the Java virtual machine policy) in those environments. However, you might want to deploy as a virtual system in production to set up the most highly optimized environment for that application. Likewise, you might change the code in later versions of an existing application that is deployed as a virtual system to make it compatible for deployment as a virtual application.

4.7.2 Database best practices to deploy

In the context of PureApplication System, there are multiple ways to deploy or configure a database. Because IBM DB2 software is integrated inside PureApplication System, the use of DB2 as a deployed application's database involves no extra costs. Thus, there is a reduction in overhead and other license tracking mechanisms. This inherent benefit of DB2 in PureApplication System reduces the total cost of ownership of the platform. The unified nature of DB2 within PureApplication System allows for best practices and expert-focused integration to be applied and followed throughout an application's use of DB2 as the database back end service.

DB2 patterns in IBM PureApplication System

IBM PureApplication System features the following options for DB2 patterns:

- DB2 virtual system patterns that are described in “DB2 virtual system patterns” on page 128.
- DB2 database workload patterns that are described in “IBM Database Patterns” on page 160.

DB2 SQL compatibility feature

For users that have not used IBM DB2 software as a database solution before, PureApplication System is an excellent environment to evaluate DB2 for production deployments with the existing enterprise applications. By using the DB2 virtual system and database workload patterns, you can enable an SQL compatibility mode to assist with migrations of applications that are written to use other competing database software. With this feature enabled, native SQL that is written for other competing databases is compiled natively in the DB2 engine without the use of slow emulation software. A compatible data concurrency model also is available. DB2 also includes tools that are compatible with existing scripts and personnel skills, which simplifies the transition to DB2.

For more information about the DB2 SQL compatibility feature, see this website:

http://www.ibm.com/developerworks/data/library/techarticle/dm-0907oracleappsondb2/index.html

Using a remote database outside of PureApplication System

In some use cases, you might need a deployed application within PureApplication System to access and link to a remote database system. For example, perhaps performance and other criteria categorized a particular database workload to be in a mission critical tier one database category. Thus, a dedicated physical system is required to host such a database system.

When a virtual application pattern is defined, you can attach existing remote database components to the pattern. These databases are outside of PureApplication System. The configuration properties of these components define the connection parameters to the remote database.

Choosing a database

The following steps are a simplified procedure for choosing a database deployment for an associated application in PureApplication System:

1. Use the DB2 database workload patterns. These patterns already incorporate best practice guidelines in their implementation of DB2. If needed, create and reference a database workload standard to apply the changes in the configuration to an associated application.
2. Because of performance or other criteria, you might choose to have the database outside of PureApplication System. If the database is outside, use an appropriate interface to attach an existing remote database component into a virtual application pattern.
3. If DB2 database workload patterns are too restrictive to use with an application, use a DB2 virtual system pattern. By using these patterns, you have greater flexibility to control this middleware environment.

For more information about procedures for deployment of DB2 in PureApplication System, see this website:

http://www.ibm.com/developerworks/websphere/library/techarticles/1204_chan/1204_chan.html

4.8 PureSystems Centre

Use the PureSystems Centre site to access solutions from IBM and IBM Business Partners, updates to systems and solutions, and expertise for maximizing the benefit of systems and solutions.

The PureSystems Centre offers a simplified experience for PureSystems users to obtain PureSystems optimized content, fixes, updates, and access to IBM and IBM Partner expertise.

Accessing the PureSystems Centre

Complete the following steps to access information about PureApplication System in the PureSystems Centre:

1. Browse to this website:

   http://www-01.ibm.com/software/brandcatalog/puresystems/centre/

2. On the welcome page, select **PureApplication System,** then select **Browse Solutions,** as shown Figure 4-61 on page 190.

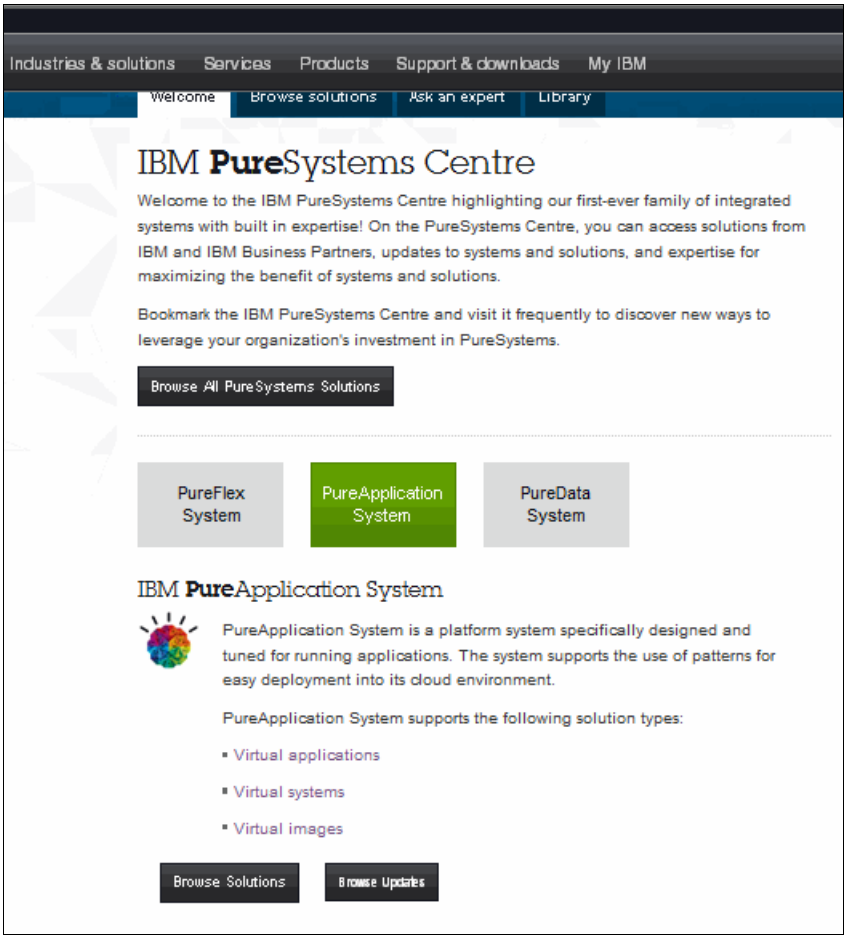


Figure 4-61 PureSystems Welcome page

You also can access the PureSystems Centre by completing the following steps:

1. From the IBM PureApplication System console, you can access PureSystems Centre, as shown in Figure 4-62.

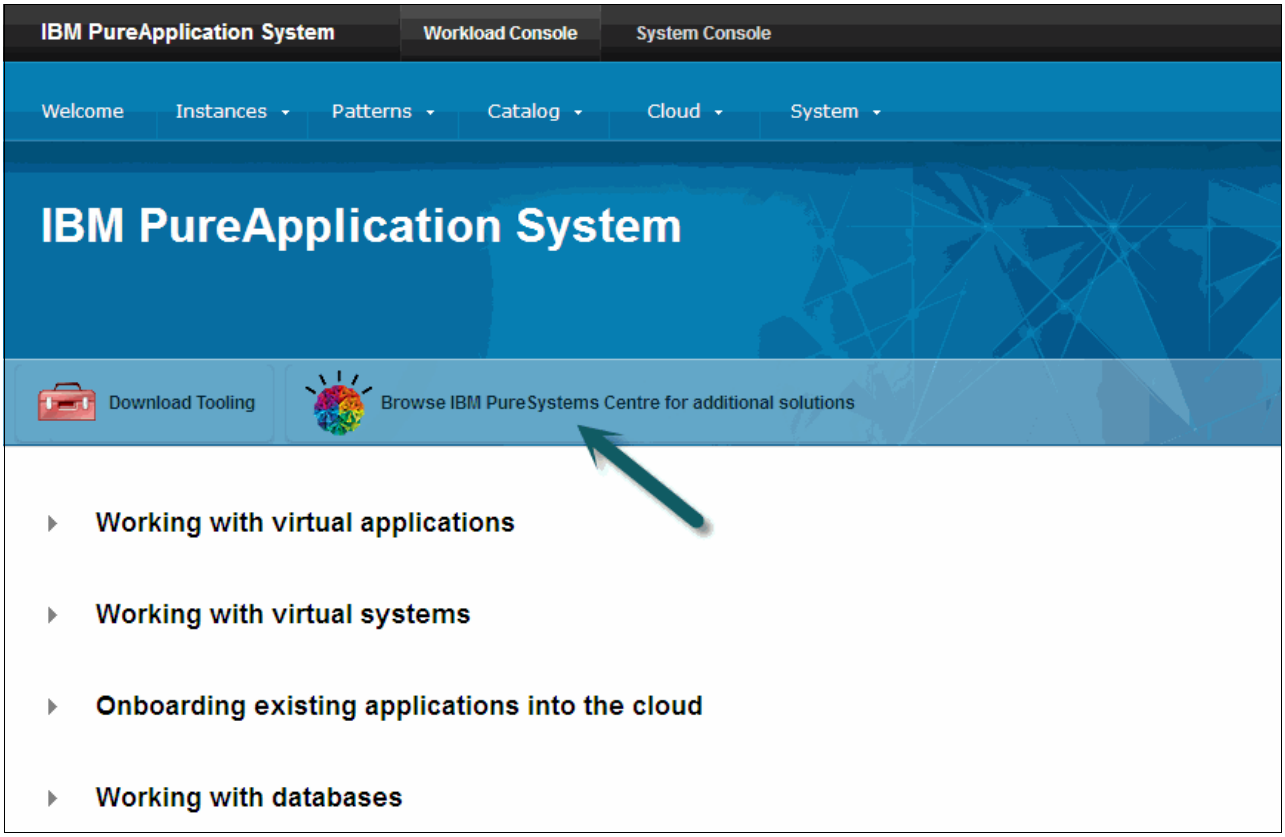


Figure 4-62 PureSystems Centre access from IBM PureApplication System console

2. Click the **Browse IBM PureSystems Centre for additional solutions** link to go directly to the Browse solutions tab, as shown Figure 4-63.

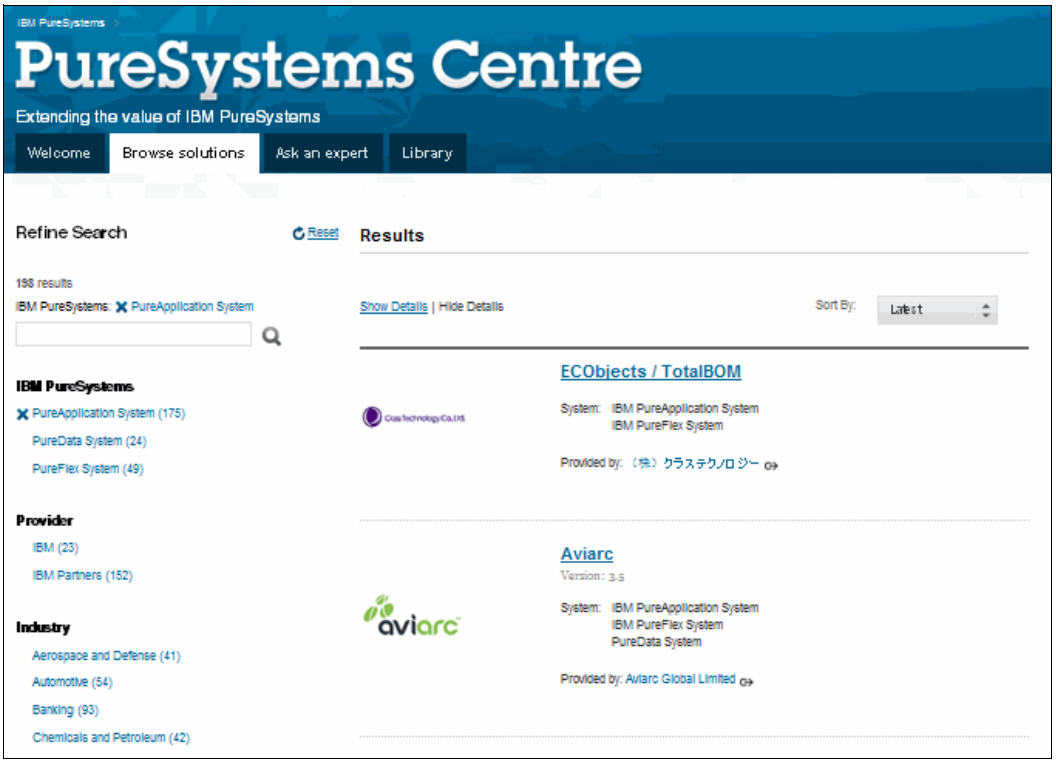


Figure 4-63 PureSystems Centre Browse solutions

Selecting a provider

From the PureSystems Centre main menu, select **PureApplication System → Browse Solutions**. There are two main providers, IBM and IBM Partners, as shown in Figure 4-64.

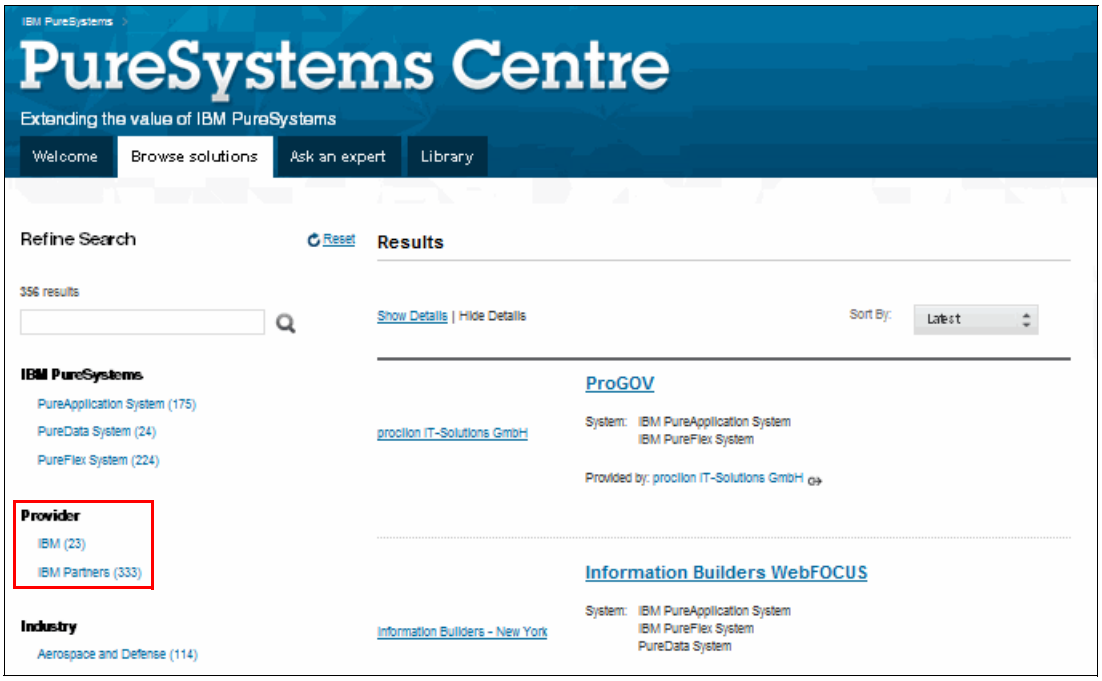


Figure 4-64 PureSystems Centre providers

IBM as the provider

When you select IBM as the provider, a list of offerings for IBM PureApplication System are displayed, as shown in Figure 4-65.

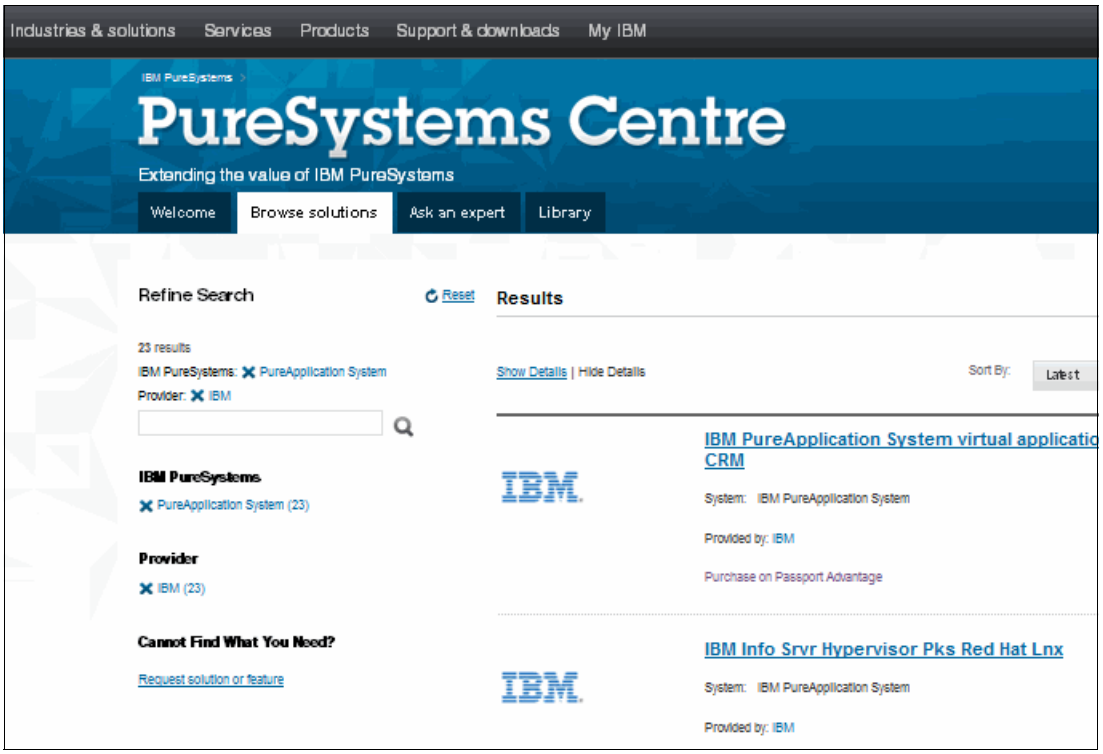


Figure 4-65 IBM Provider view

The list shows all of the IBM offerings that extend current IBM PureApplication System built-in virtual image offerings. To see the built-in images that are supported in IBM PureApplication System, see the following websites:

- http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS213-038&appname=USN
- http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/index.jsp?topic=%2Fcom.ibm.ipas.doc%2Fiwd%2Fpct_manage_cat_vi.html

For example, consider that you must get an IBM Business Intelligence Pattern for your solution. Select the IBM Business Intelligence Pattern, scroll down and select **Purchase**, as shown in Figure 4-66.

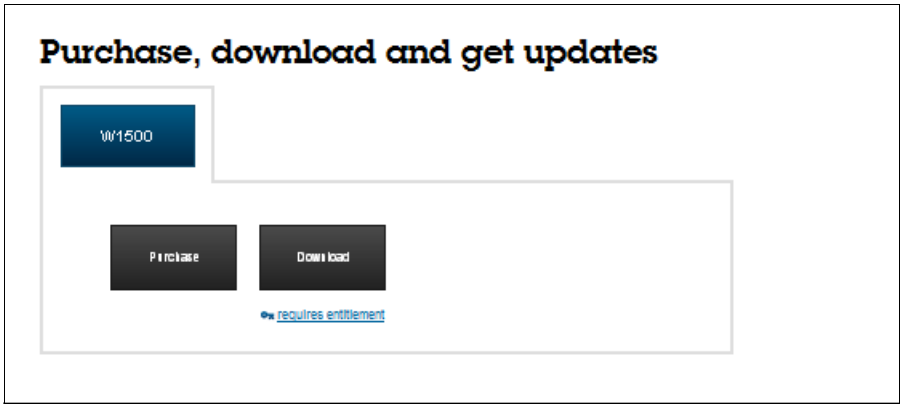


Figure 4-66 Selecting a pattern to purchase

Continue through the purchase process to the section where you select media packs (CD-ROMs, DVDs). Your order is processed.

IBM Partners

Selecting **IBM Partners** in the provider list shows a complete list of partners that produce patterns or solutions for IBM PureApplication System, as shown in Figure 4-67. You can select a Partner, learn more about the solutions, and follow the necessary links on that solution's page.

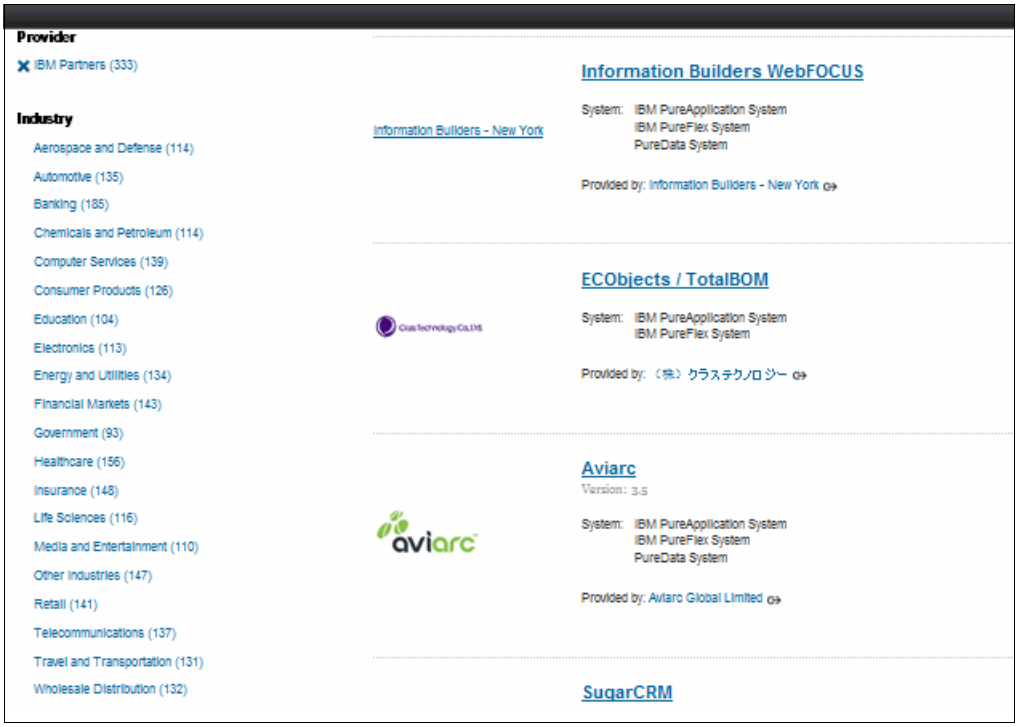


Figure 4-67 IBM Partners view

Asking an expert

From the main page, you can select the **Ask an Expert** tab (as shown in Figure 4-68) to get answers to your questions by using the knowledge base of community experts. This section offers the opportunity for you to contribute. To contribute on this site, you must be registered.

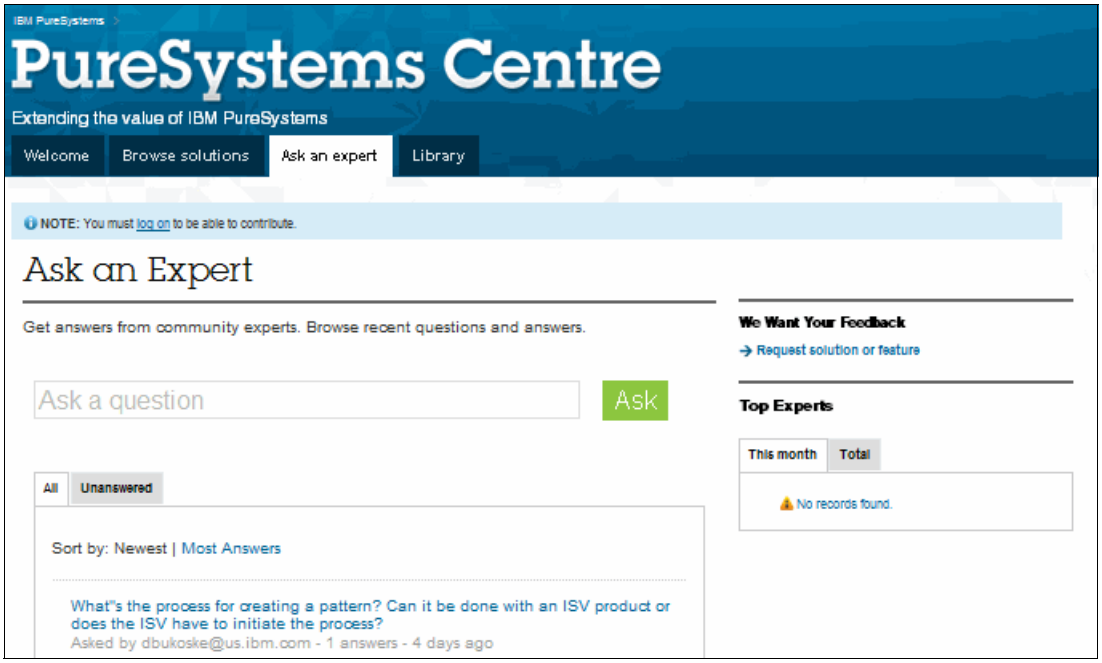


Figure 4-68 Ask an Expert view

Library

By selecting the Library tab on the main menu (Figure 4-69), you can search for topics about which you want to learn more information. This tab contains web articles, information center topics, and IBM Education articles. The library contains a breadth of knowledge, resources spanning initial information, overviews, and in-depth documentation and videos to help you learn about IBM PureSystems.

Figure 4-69 Library view

Browse updates

From the PureSystems Centre welcome page, select **PureApplication System**, in which there is an option is to view the updates. Select **Browse Updates, as** shown in Figure 4-70.

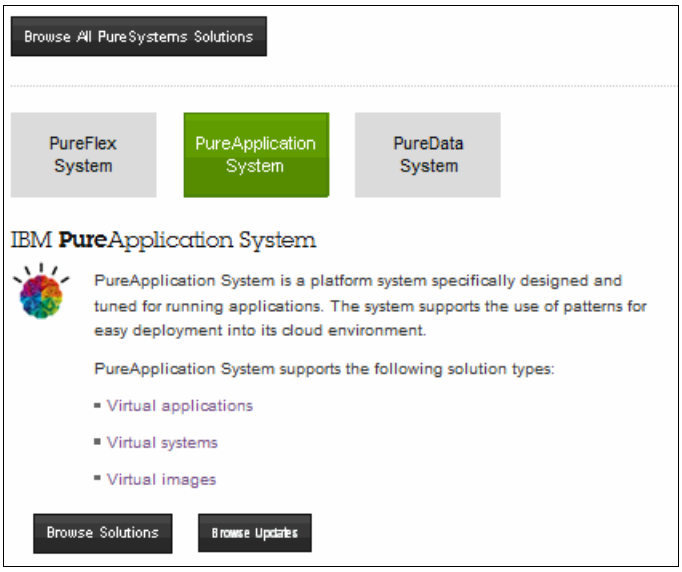


Figure 4-70 Browse updates

A list of PureApplication System fix packs (firmware and system updates for a specific version) and Group Fix Updates (updates for add-ons and shared services) are displayed, as shown in Figure 4-71.

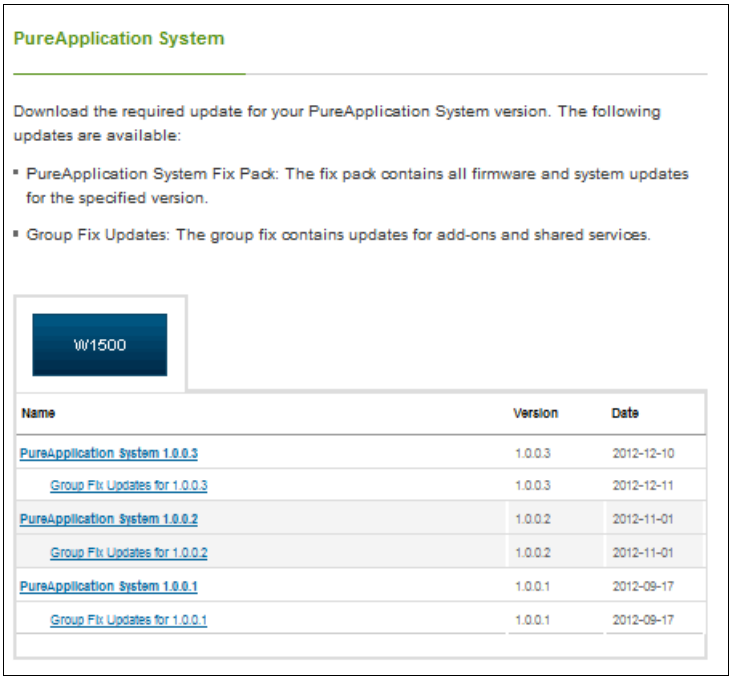


Figure 4-71 Available updates

5

Customizing Virtual System Patterns

This chapter describes the various available resources or tools for customizing virtual system patterns in PureApplication Systems, managing virtual images, script packages, add-ons, and advanced options. It also describes the usage of common tools to customize the virtual system patterns.

The following topics are covered in this chapter:

- Resources for customization
- Managing patterns by using Virtual System Pattern Editor
- Build virtual images by using the IBM Image Construction and Composition Tool
- Administering PureApplication System by using CLI
- Working with script packages
- Using Advanced Middleware Configuration
- Add-ons and advanced options

© Copyright IBM Corp. 2013. All rights reserved.

5.1 Resources for customization

Virtual systems form a part of the PureApplication System deployment model. It primarily consists of the virtual images and the virtual system patterns. The virtual images are included with many products and services that are provided by IBM. Some of these products and services are product images, sets of execution services for stand-alone virtual machines, image management capabilities, or library functions. The virtual system patterns are more focused on middleware and are components of IBM provided product images and patterns for commonly used topologies.

Virtual system patterns provide the following support:

- Repeatable system deployments
- Automated provisioning of systems as patterns
- Traditional configuration and administrative model (more control for the user)

Several virtual system patterns are shipped with PureApplication System. The following patterns are the default patterns:

- WebSphere Application Server (V7.0, V8.0, and V8.5)
- DB2 Enterprise (V9.7 and V 10.1)

The patterns for each of the shipped product patterns contain topologies that are tailored for best practices and intended for supporting user applications.

However, there might be certain situations where the default patterns need tuning to cater to business needs. Then, users one can use the customization options that are exposed by the virtual images and system patterns.

The following common use cases drive customization:

- Need for custom middleware topologies that are not available in the shipped patterns
- Introduce user-wanted tuning and company standards or policies
- Include custom scripts for modifying the middleware configuration
- Integrate custom products with default patterns to build your own configuration

The following scenarios exemplify those use cases:

- A company must enforce their own security policy for applications, which is not provided by the in-house patterns.
- The need to build a pattern by using WebSphere Application Server and Oracle database.

For more information about using available shipped virtual system patterns, see 4.5.2, “Planning and designing your virtual system pattern” on page 129.

5.1.1 Customization targets

Virtual Systems patterns are combinations (logical groupings) of parts that are grouped to create a specific topology for a deployment need. An example would be a WebSphere Application Server Cluster pattern that contains Deployment Manager, one or more Custom Nodes, IBM HTTP Server, and configuration scripts for installing applications to the topology.

To build virtual system patterns, you can use parts from one or more virtual images, add-ons, and script packages.

The following endpoints are standard for customization:

- Image customization: Customize the IBM provided Hypervisor images of IBM middleware products or Custom Hypervisor images.
- Middleware version number: Customize the virtual system pattern to include middleware versions of your choice. An example of this option is building a mixed cell WebSphere topology with deployment manager by using 8.0.0.0, custom nodes by using 7.0.0.25, and using IBM HTTP Server with version 7.0.0.23.
- Initial number of nodes: Include user-provided number of nodes instead of the default set available in the pattern.
- Arbitrary script package execution: Attach your own script packages that must be run as part of the pattern.
- Advanced Options & Ordering: Specify the order or sequence in which the parts of the pattern are deployed and advanced configuration settings for the pattern, such as messaging, session management, and security.

5.1.2 Tools for customization

After you have the wanted level of information about the business requirements and the applicable customizations, its time to begin with the task of customizing. Based on the components or parts that must be modified, IBM PureApplication System provides the following tools for customization:

- Virtual System Pattern Editor: Integrated into the PureApplication System console or GUI that you use to manage Virtual system patterns.
- IBM Image Construction and Composition Tool: Available as a separate downloadable tool, which is installed and configured to manage the virtual images part of the PureApplication systems environment.
- Command-line interface: Available as a separate utility that can be set up to manage the PureApplications Systems environment, including virtual images, patterns, deployed instances, and other entities.
- Script packages: Could be extended or added to support automated provisioning (installation and configuration) of the virtual system patterns.
- Add-ons to the catalog: Available as downloadable parts that must be included in the catalog.
- Advanced Options and Ordering: Composed of configuration settings for patterns, such as messaging, session management, and security. This option also works with defining the order of the parts in the pattern for deployment.

5.2 Managing patterns by using Virtual System Pattern Editor

Virtual system patterns implement deployment topologies from one or more virtual images and applications from the system catalog. Use the Virtual System Pattern Editor on the workload console of the PureApplication System to manage virtual patterns. The following common operations are performed by using the virtual system pattern editor:

- Creating new patterns
- Editing existing patterns
- Cloning or extending patterns
- Include parts from multiple images in a pattern
- Drag script packages and add-ons onto any part in the pattern

5.2.1 Editing virtual system patterns

A virtual system pattern can be edited to modify its configuration (associated parts or script packages). The following conditions apply for any user who is editing the virtual system pattern:

- Pattern is not read-only or locked
- Does the creator of the pattern have the required user permissions (write access)

The Figure 5-1 shows the panel for working with virtual system patterns on the Workload console.

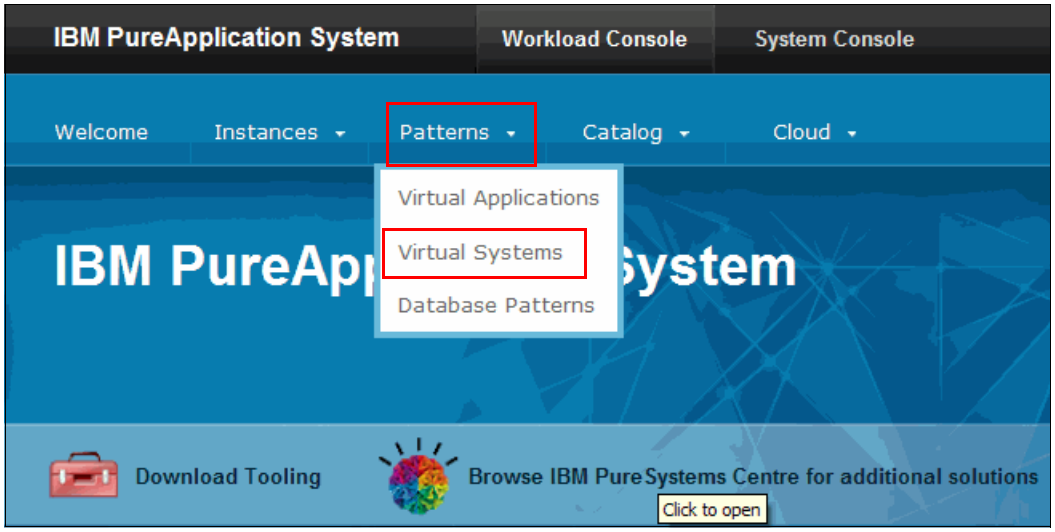


Figure 5-1 Link to Virtual Systems pattern on the Workload console

The Patterns and Pattern Editor panes provide fields to view and customize your virtual system patterns topology, which are described next.

Pattern pane view

The Patterns pane displays the applicable virtual system patterns for the available virtual images. You can search for or add virtual system patterns. It also includes information about the topologies that are associated with the patterns.

The Patterns pane provides the following interactive panes:

- The left pane that lists the virtual system patterns.
- The right pane that shows configuration and topology information for the selected virtual system pattern.

Figure 5-2 displays the sections that are available on the Patterns pane.

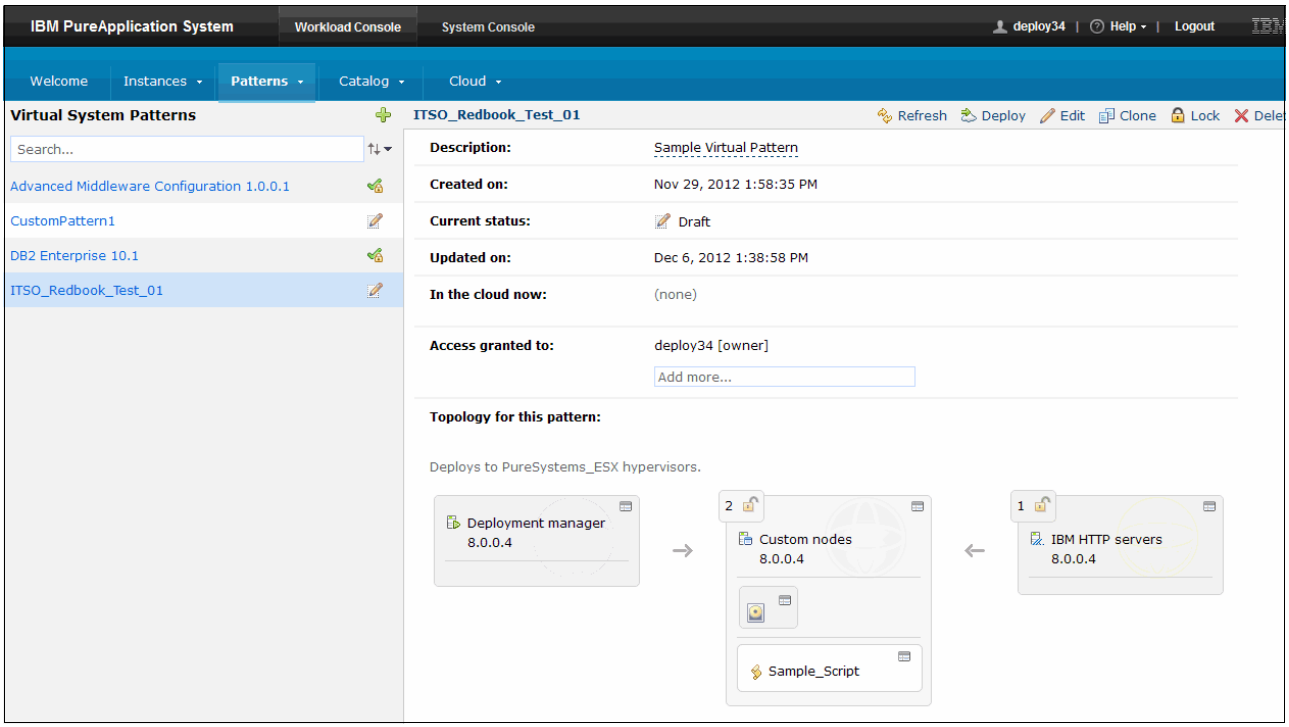


Figure 5-2 Pattern configuration panel

Pattern Editor panel view

The Pattern Editor panel contains information about specific virtual system patterns that are edited. Lists of scripts, parts, and add-ons are provided and can be added to the virtual system patterns topology or edited on the provided canvas. This panel is displayed when you select the Edit button in the upper right corner in the Pattern pane for the selected pattern.

Figure 5-3 shows the sections that are available on the Pattern Editor panel.

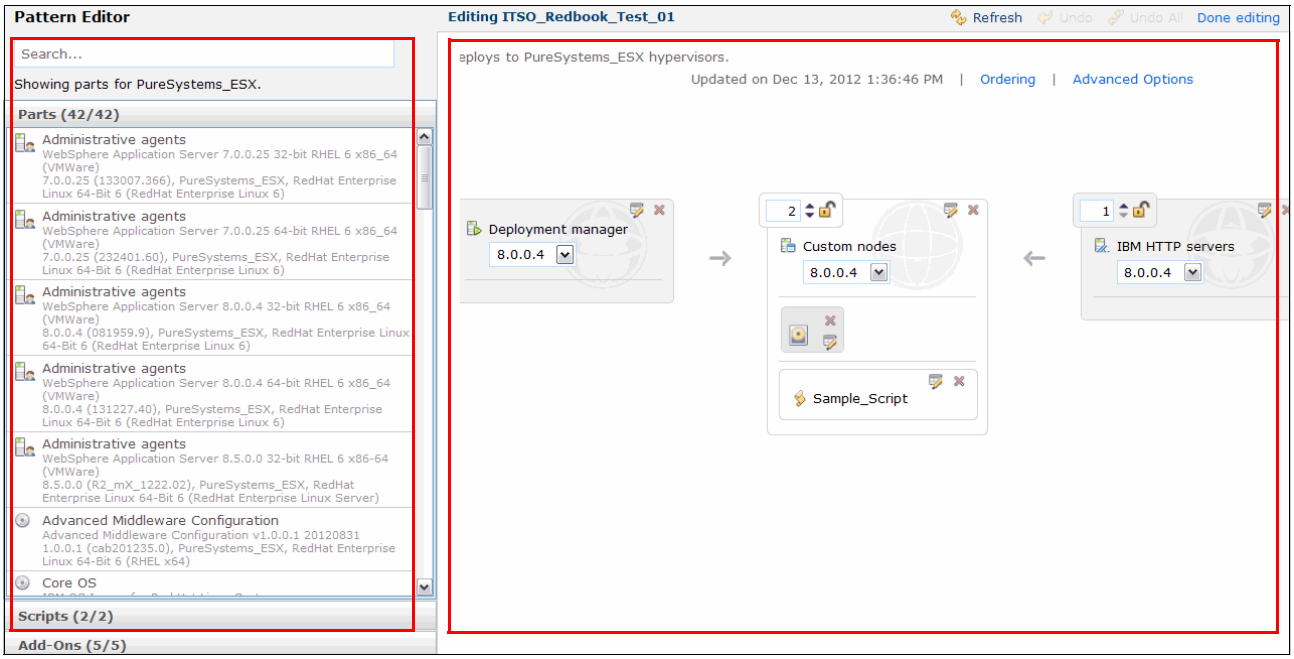


Figure 5-3 Pattern Editor panel

Parts

The Parts pane displays parts that are available to use in your virtual system pattern. It depends on the virtual images that are installed and the hardware types of parts that are already included in the virtual system pattern. Only parts with the same hardware type as parts that are already available in the virtual system pattern are listed.

Scripts

The Scripts pane lists the script packages that are available. It can contain any script packages that you provided for use with PureApplication System. Use this pane to add script packages to the parts on the editing pane.

To add parts, drag them onto the workspace on the right side canvas of the Pattern Editor panel and drop them onto the part objects.

The script packages section in the Pattern Editor panel is shown in Figure 5-4.

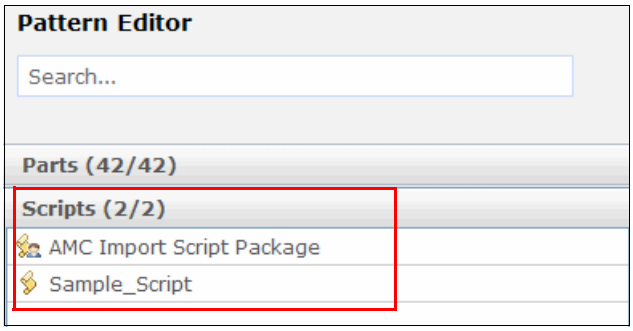


Figure 5-4 Script packages panel

Add-Ons

The Add-Ons pane lists the add-ons that are available.

To add add-ons to the nodes on the editing pane, drag them onto the workspace on the right side canvas of the Pattern Editor pane and drop them onto the node objects. Add-ons that can be added to the nodes for deployment on a virtual machine are types such as disk, network interface controller (NIC), and user.

Figure 5-5 shows adding a disk to the virtual system pattern as an Add-on.

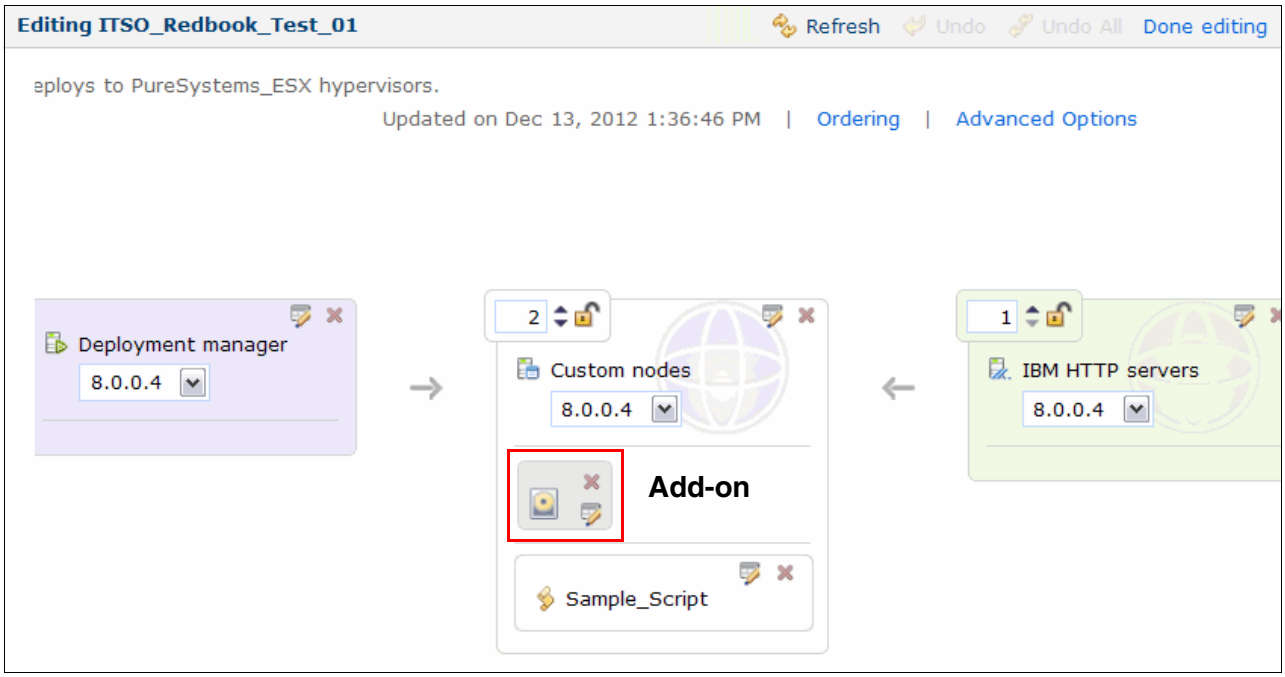


Figure 5-5 Adding disk as an Add-on to the virtual system pattern

Figure 5-6 shows setting properties for the add-on disk.

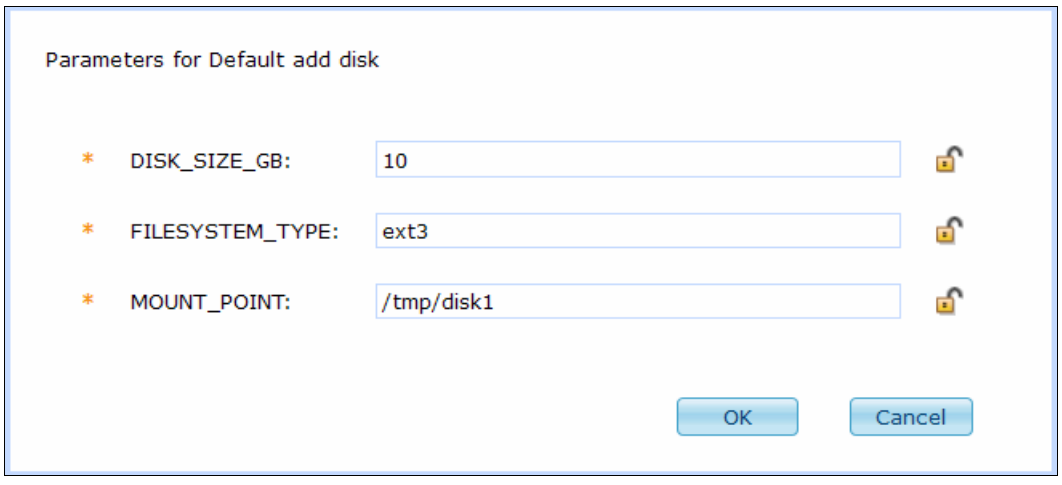


Figure 5-6 Setting parameters for the newly added add-on disk

When a specific virtual system pattern is being edited in the Pattern Editor pane, the graphical topology view is displayed in an editing canvas. The following options are available to view a virtual system pattern:

- Ordering
- Topology

These options are described next.

Ordering

The Ordering link shows the order in which the parts are started when the virtual system pattern is deployed. If you are working with a copy of a provided virtual system pattern, the default setting is the recommended order. This view displays the parts in the right column of the pane numbered in the order they are started. The left column provides a textual description of the order in which the parts are started with order constraints for parts and scripts.

Figure 5-7 shows the ordering panel with which you can specify or modify the deployment order of the parts of the virtual system.

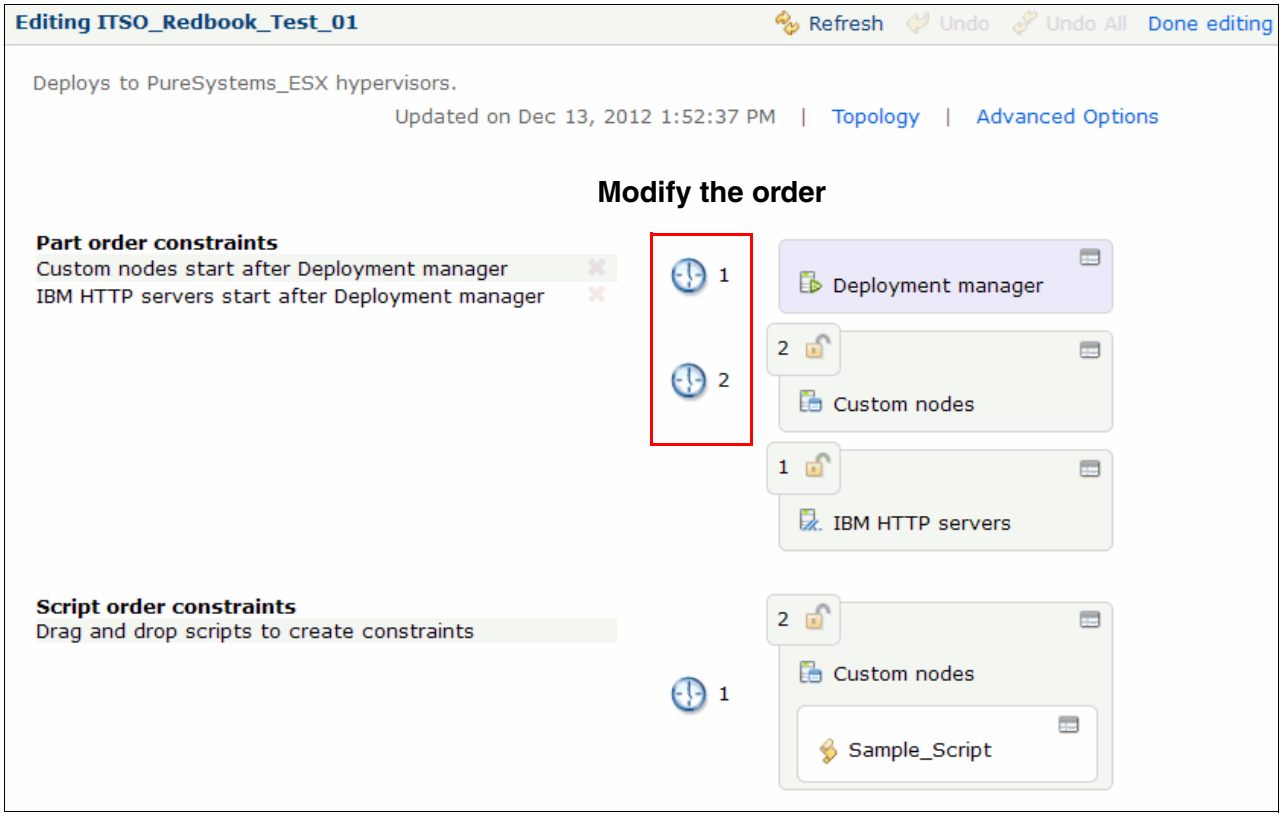


Figure 5-7 Ordering virtual system patterns

Figure 5-8 shows the application of a search filter to select parts.

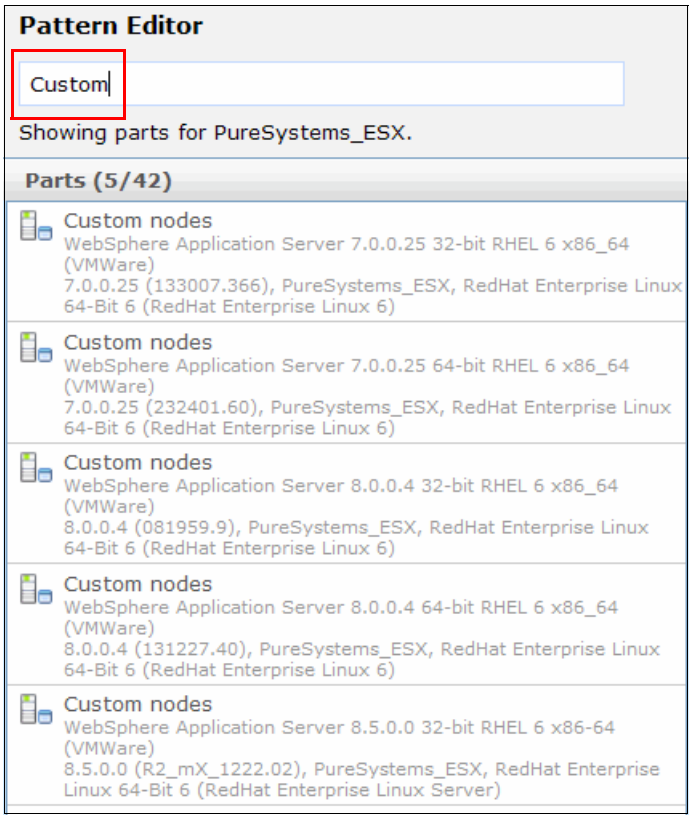


Figure 5-8 Applying search filter to choose parts

Topology

The Topology link is shown when you are in the Ordering view. Switch back to the Topology view to show the relationship of the parts.

Modify pattern configuration

By using the various configuration panels that are available in the virtual system pattern editor, you can customize configurations. Based on the customization that is needed, you must browse to the corresponding panel and modify the pattern settings.

Consider a simple example of modifying the middleware version number for custom nodes in the pattern. Complete the following steps to perform this task:

1. Click the **Workload Console** tab at the top of the Welcome page to open the workload console.
2. Click **Patterns** → **Virtual Systems**
3. Select a virtual system pattern. The example pattern that is used in this scenario is `ITSO_Redbook_Test_01`
4. Click **Edit** on the toolbar.

5. In the configuration pane on the right side under part Custom nodes, select a suitable middleware version from the drop-down list. Figure 5-9 shows how to modify the middleware version number of the custom nodes.

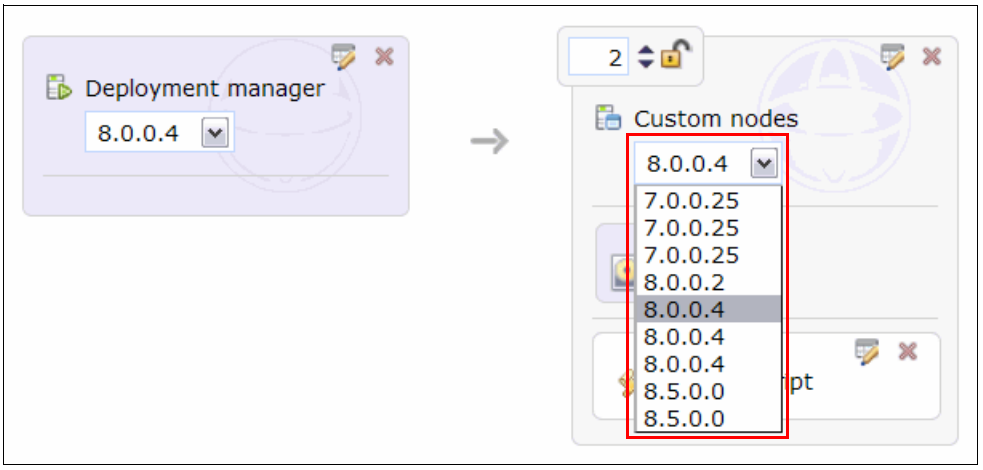


Figure 5-9 Modifying middleware version number for custom nodes

6. Select the **Edit** icon to modify the Properties for the custom nodes part, as shown in Figure 5-10.

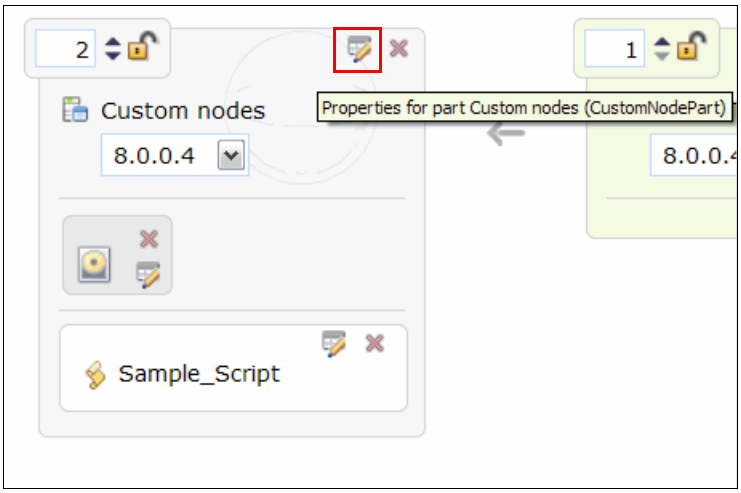


Figure 5-10 Editing properties for custom nodes

7. Enter the values for the WebSphere Repository location, user, password, and so on. Click **OK**. Figure 5-11shows the properties for custom nodes that are available for editing.

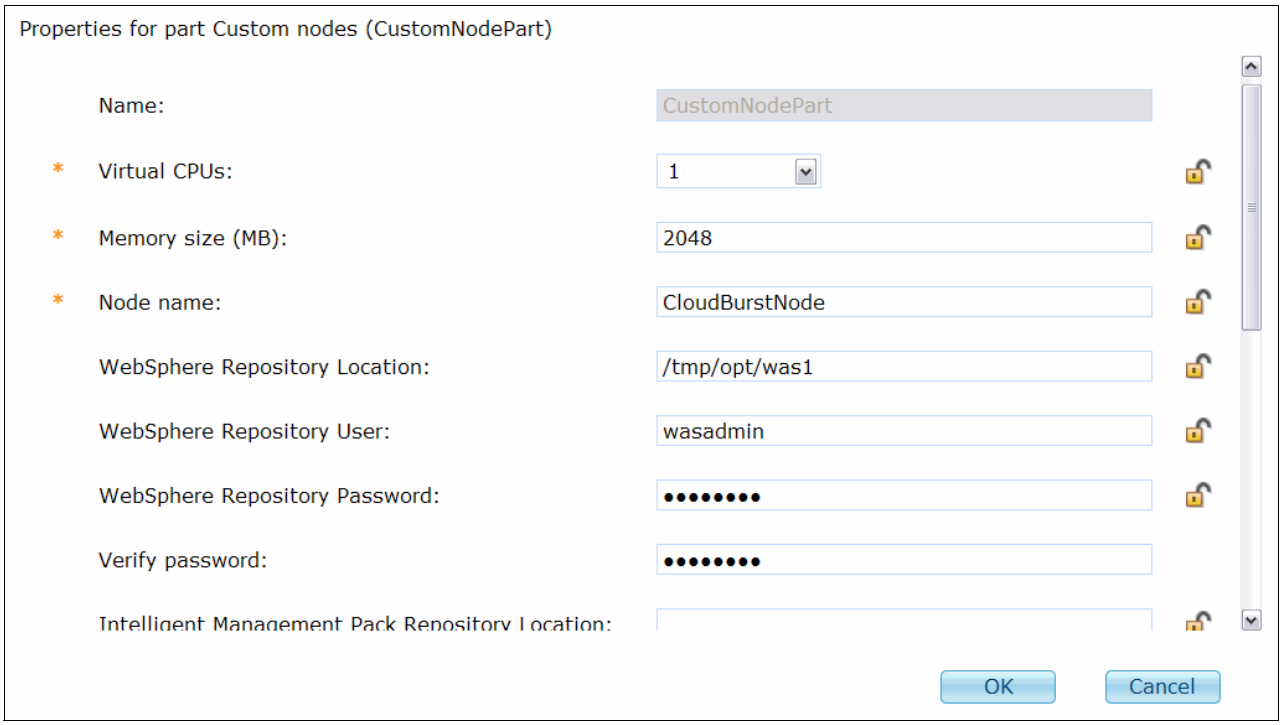


Figure 5-11 Properties panel for custom nodes

8. After all of the changes are made, click **Done editing link** to save the changes.

Similarly, various other configuration parameters such as script packages, add-ons, and advanced options can be edited or modified by using the virtual system pattern editor.

5.2.2 Cloning virtual system patterns

One of the commonly used ways to modify patterns is cloning. To customize patterns for your environment, clone one of the available default patterns. Select a pattern that closely meets the business needs (matches with the topology that is built) and then edit its settings.

Important: If cloning is not enabled, you must accept the license agreement for the virtual image that is associated with the virtual system patterns. To accept the license, click **Catalog** → **Virtual Images** on the workload console view and locate the image that is associated with the virtual system pattern.

Complete the following steps to clone a virtual system pattern:

1. Click the **Workload Console** tab at the top of the Welcome page to open the workload console.
2. Click **Patterns** → **Virtual Systems.**
3. Select a virtual system pattern.
4. Click the Clone icon on the toolbar.

5. Enter the following information about the new virtual system pattern:
   - Name: Unique name for the virtual system pattern in the Name field
   - Description: (Optional) A detailed description to identify the virtual system pattern
   - Virtual Image: Select a virtual image from the list to associate the virtual system pattern

Important: If the original virtual system pattern contains parts from different virtual images, the Virtual Image option is disabled. In this case, the parts in the new virtual system pattern are associated with the same virtual images as the corresponding parts in the original pattern.

Figure 5-12 shows the configuration settings for cloning virtual system patterns.

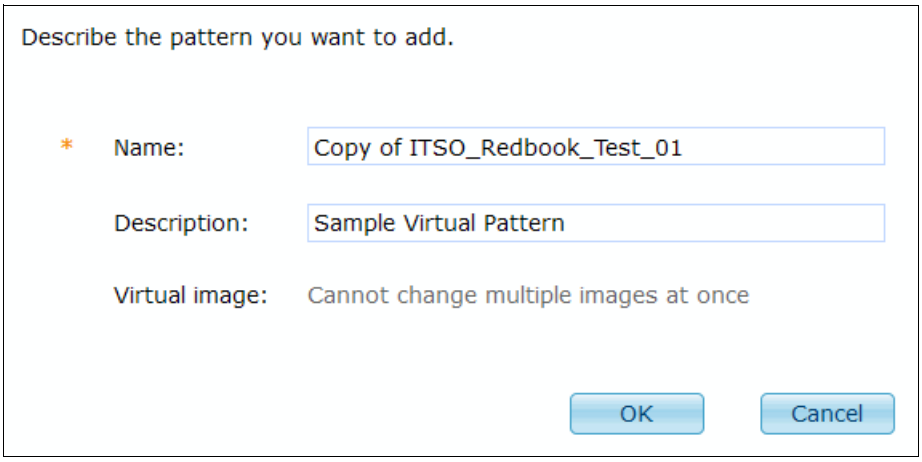


Figure 5-12 Cloning virtual system patterns

6. Click **OK**. When the information is processed, the Virtual System Pattern pane opens. The virtual system pattern that you cloned is added to the list.

Figure 5-13 shows the newly cloned virtual system pattern.

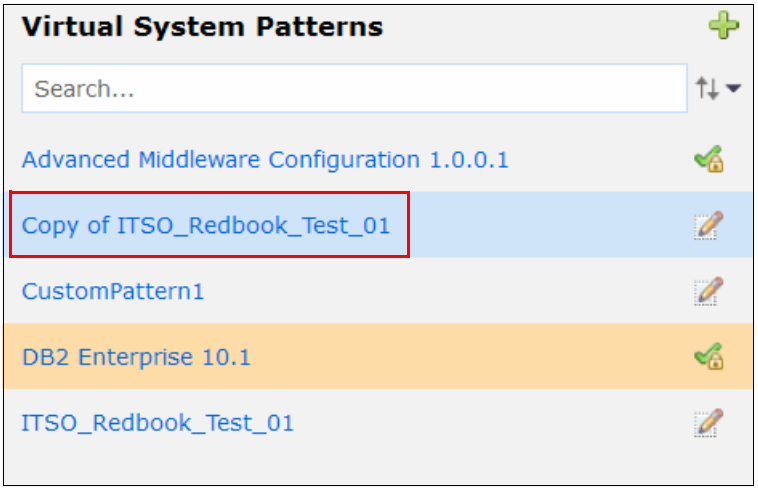


Figure 5-13 Cloned virtual system pattern

To make more changes to the pattern, select the pattern from the left side of the Patterns pane and click **Edit**. The Virtual System Pattern Editor pane opens, which shows the various fields on this pattern. The configuration can then be edited, such as adding or modifying script package properties, parts contents, add-ons, or advanced option settings for the patterns. Your business needs can drive these configurations.

For more information about editing virtual system patterns, see 5.2.1, “Editing virtual system patterns”.

5.2.3 Creating virtual system patterns

When the existing virtual system patterns do not provide the wanted environment, you can create a pattern to meet your business needs.

Complete the following steps to create a virtual system pattern:

1. Click the **Workload Console** tab at the top of the Welcome page to open the workload console.
2. Click **Patterns** → **Virtual Systems**.
3. Click the New icon on the toolbar
4. Enter the following information about the new virtual system pattern:
   - Name: Unique name for the virtual system pattern
   - Description: (Optional) A detailed description to identify the virtual system pattern
5. Click **OK**. The new virtual system pattern is created and its details are displayed in the Virtual System Pattern pane.

Figure 5-14 shows the panel that is used for creating a pattern.

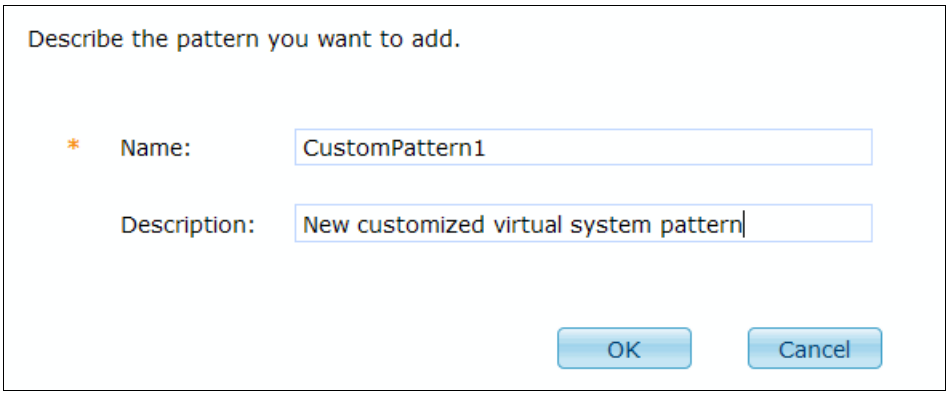


Figure 5-14 Creating virtual system pattern

You also can edit the pattern, add script packages, modify parts, configure properties for parts and script packages, include add-ons, or define advanced options in accordance with the wanted configuration.

Figure 5-15 shows the options and settings for further editing the created pattern.

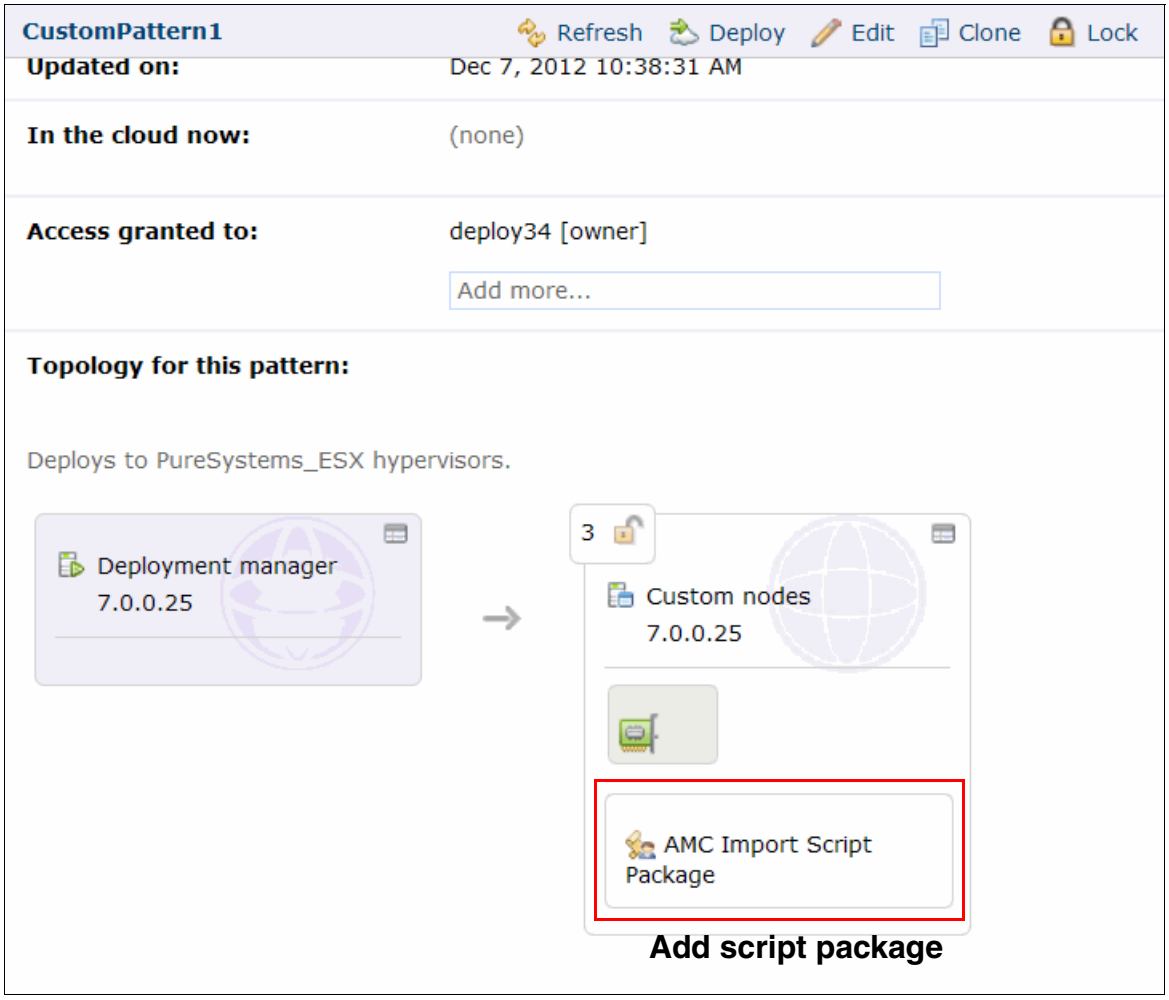


Figure 5-15 Editing newly created virtual system pattern

5.3 Build virtual images by using the IBM Image Construction and Composition Tool

Virtual images provide the operating system and product binary files that are required to create a virtual system instance. The IBM OS image for Red Hat Linux Systems virtual image, for example, is preinstalled on the PureApplication System. However, these images can be extended for customization.

Complete the following steps to see a list of all available virtual images in the system:

1. Click the **Workload Console** tab at the top of the Welcome page.
2. Click **Catalog** → **Virtual Images**.

Important: To view virtual images, you must be granted access to the virtual image or assigned Workload resources administration with full permissions.

Overview

The PureApplication System provides the Image Construction and Composition Tool to build virtual images for deployment into cloud environments. You also can build software bundles for reuse by others.

The use of Image Construction and Composition Tool (as opposed to traditional image creation process) includes the following benefits:

- Simplifies the complexity of image creation by using model-driven image creation.
- Reduces errors that are caused by automated installation process.
- Provides reuse of images and software bundles in another cloud environment.
- Standardizes the image creation process.
- Allows for high degree of customization.
- Enables image lifecycle management, such as image creation and activation.
- Provides interoperability with other IBM products and software.

Target audience

The primary users and the purpose of the use of Image Construction and Composition Tool are described in Table 5-1.

Table 5-1 Users of Image Construction and Composition Tool and their scope

Standard Users	Purpose of use
Operating system specialists	Create base operating system images for organizational use
Software specialists	Create software bundles encapsulating software content
Image builders	Build images for deployment by selecting the operating system and software
Cloud administrators	Provide users of their cloud systems with virtual machines that have preinstalled software

5.3.1 Installing Image Construction and Composition Tool

Image Construction and Composition Tool builds Open Virtualization Appliance (OVA) files that can be deployed into VMware clouds and IBM PureApplication System clouds.

Image Construction and Composition Tool can be installed and used as a stand-alone application in the PureApplication System. This section describes how to install and configure Image Construction and Composition Tool for stand-alone environments.

Important: If you are installing Image Construction and Composition Tool as a stand-alone application, you can create only a cloud provider that is based on the existing set of cloud providers. You cannot create a PureApplication System cloud provider.

Prerequisites

On the IBM PureApplication System Welcome page, click **Download Tooling** → **Download IBM Image Construction and Composition Tool** to download the compressed file that contains the installation binary files. The format of the compressed file is `ICCT_Install_<version>.zip`.

Important: Before you download Image Construction and Composition Tool, you must enable the Foundation Pattern type. To enable the Foundation Pattern type, on the IBM PureApplication System Welcome page, click **Cloud** → **Pattern Types** → **Select Foundation Pattern Type 2.0.0.0** and click **Enable**.

Figure 5-16 shows the panel that is used to download Image Construction and Composition Tool.

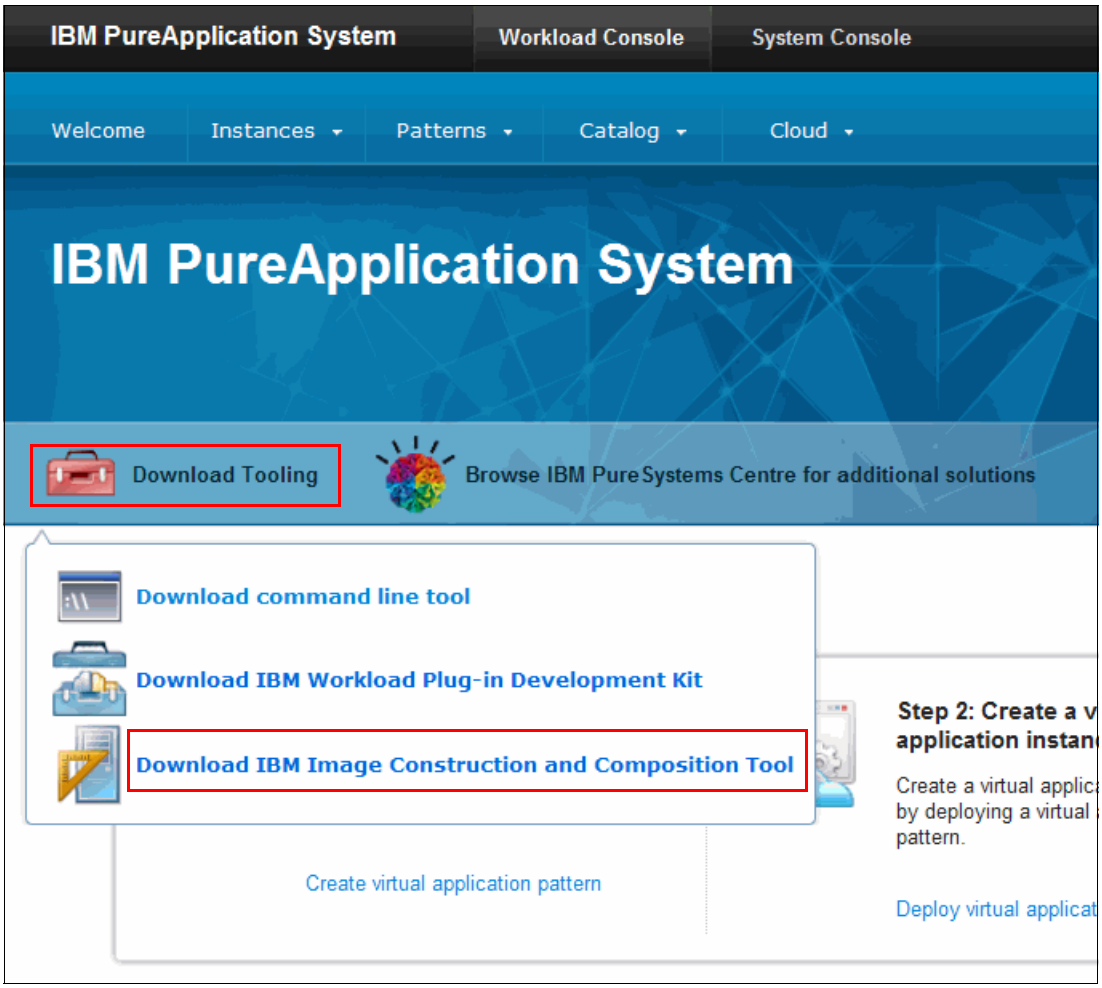


Figure 5-16 Downloading Image Construction and Composition Tool

When you are considering installing Image Construction and Composition Tool, the prerequisites for hardware requirements (physical or virtual) and supported operating systems are documented in the product information center, which is available at the following websites:

- Linux:

  `http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/ICON/topics/iwd_cicn_installationsteps.html`

- AIX:

  `http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/ICON/topics/cicn_installationsteps_aix.html`

Installing Image Construction and Composition Tool

Image Construction and Composition Tool can be installed by using normal (GUI) or silent installation procedures. For silent installation, extract the compressed file and run `./install` in the directory where you extracted the compressed file. IBM Installation Manager is installed if it is not already installed. After IBM Installation Manager is installed, it installs Image Construction and Composition Tool.

For more information about installing Image Construction and Composition Tool on a specific platform (Linux of AIX), upgrading from previous versions, or backing up and restoring the database, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/ICON/topics/iwd_cicn_installing.html

Post installation configuration

When you access Image Construction and Composition Tool for the first time, you are prompted to create a cloud provider definition. All actions that are performed are based on a cloud provider.

Ensure that your system meets the cloud provider and browser requirements for the use of Image Construction and Composition Tool. For more information about system requirements, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/ICON/topics/iwd_ricn_prereqs.html

The Welcome page contains links to each of the main Image Construction and Composition Tool pages that provide access to the key functions of the tool, as shown in Figure 5-17.

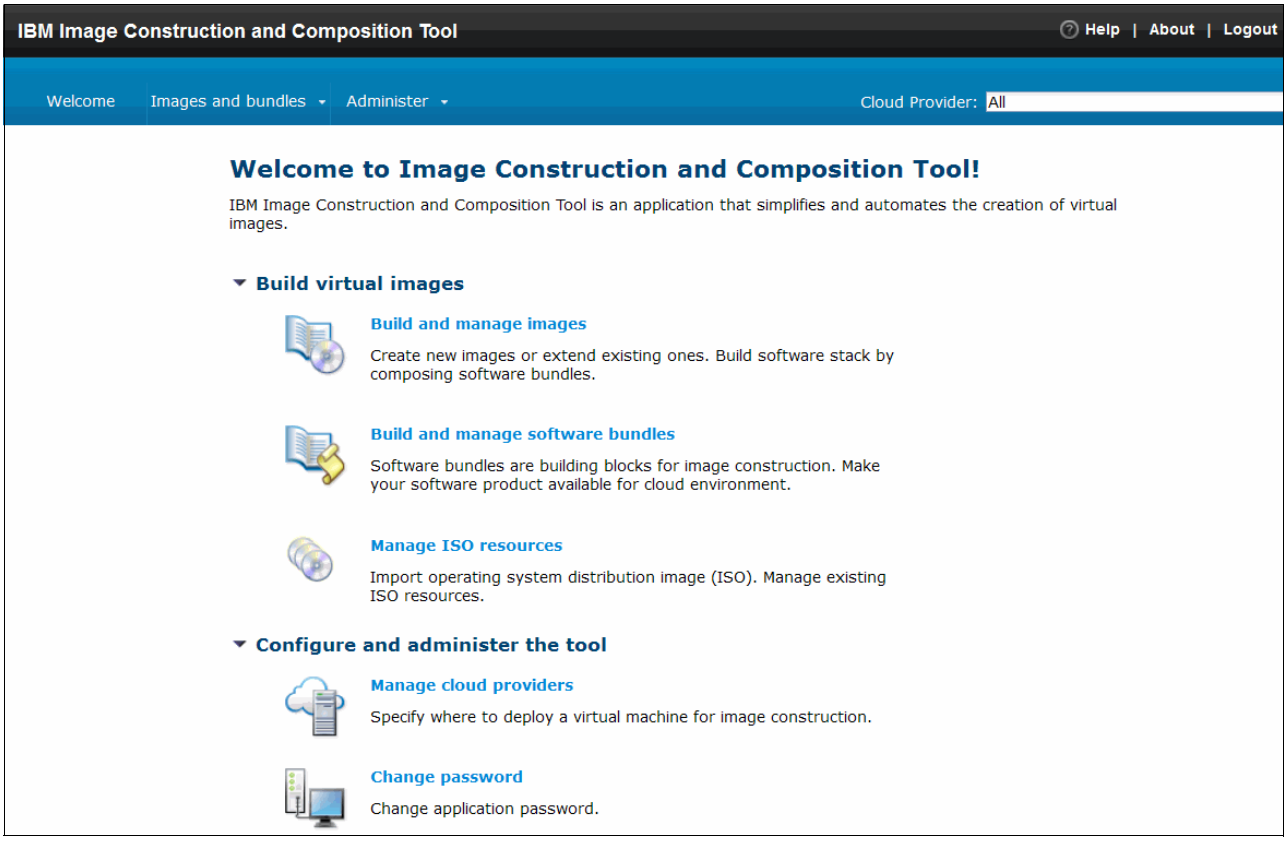


Figure 5-17 Image Composite Construction Tool Welcome page

Table 5-2 defines the pages that are available that use Image Construction and Composition Tool and the corresponding operations that can be performed in the respective sections.

Table 5-2 Image Composite Construction Tool pages

Page	Operations allowed
Images	► Import virtual images from your cloud provider ► Create virtual images ► Extend virtual images ► Capture virtual images ► Delete virtual images ► Search for virtual images ► Share software bundles by importing from or exporting to a remote system ► Create software bundles ► Extend software bundles ► Publish software bundles ► Delete software bundles ► Search for software bundles
Administer	► Configure cloud provider ► Change password ► Download log files

Universal Identifiers

When you create virtual images, bundles, and other assets, you enter a unique universal identifier (ID) for the asset. The universal ID uniquely identifies an Image Construction and Composition Tool asset across different repositories.

The universal ID uses an Open Service Gateway Interface (OSGI) notation, with a reverse domain name and a version in the format `major.minor.macro` and an optional qualifier; for example, `com.ibm.images.was_1.0.0` for a WebSphere Application Server base virtual image. In general, you can use any dotted notation to create a universal ID; for example `icct.image.base.rhel56_1.0.0`.

Configuring user password

Complete the following steps to change the user password when Image Construction and Composition Tool is installed as a stand-alone application outside of PureApplication System:

1. Click **Administer** → **Change password**.
2. Enter your new password (according to the password requirements) in the New password field and verify it.
3. Click **OK**.

Figure 5-18 shows the password change panel for Image Construction and Composition Tool.

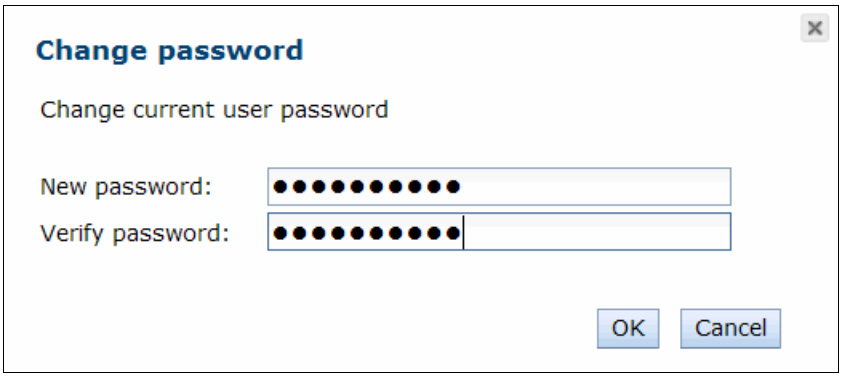


Figure 5-18 Password change panel in Image Construction and Composition Tool

Firewall settings

When an image is extended, if you enable the firewall on the virtual machine after synchronize and before capture, ensure that the ports that are required for SSH communication are open for inbound and outbound traffic. Access is required for the Image Construction and Composition Tool to complete the image building process.

Figure 5-19 shows information about opening ports for Image Construction and Composition Tool for the corresponding operating systems.

```
Redhat Linux
cat /etc/sysconfig/iptables | grep 22
cat /etc/sysconfig/ip6tables | grep 22

SUSE Linux
cat /etc/sysconfig/SuSEfirewall2 | grep 22

AIX
ls filt –v 4
Lists all of the ipv4 ports. Check if port 22 is in the list.

lsfilt -v 6
Lists all of the ipv6 ports. Check if port 22 is in the list.
```

Figure 5-19 Configuring ports for providing access to Image Construction and Composition Tool

Configuring the cloud provider

A cloud provider is a service provider that offers storage or software services on a private or public network, termed as a cloud. The following systems can be configured as cloud providers with Image Construction and Composition Tool:

- PureApplication System
- VMware ESX

The first time that you open Image Construction and Composition Tool, you see the Create a new cloud provider wizard. If the wizard does not open automatically, complete the following steps to configure the cloud provider:

1. Click **Administer** → **Manage cloud providers**.
2. Click the **Add new cloud provider** icon.
3. Enter the required information in the opened panel, such as cloud provider host name, connection credentials, and provider name to create the cloud provider.

Important: Cloud provider configuration cannot be edited after it is created.

5.3.2 Enabling Image Construction and Composition Tool as a virtual application

A virtual application pattern that is named Image Construction and Composition Tool with the type IBM Image Construction and Composition 1.2 is available for deployment, if you can access this pattern. This section describes how to create and deploy Image Construction and Composition Tool as a virtual application pattern.

Important: You must be granted access to view and create patterns, or have the Workload resources administration with full permissions to complete this task.

Enabling Image Construction and Composition Tool for first time

Complete the following steps to enable Image Construction and Composition Tool for first time (these steps are necessary only once):

1. Click the **Workload Console** tab at the top of the Welcome page to open the workload console in PureApplication System.
2. Enable the Foundation Pattern type by clicking **Cloud** → **Pattern Types**. Select **Foundation Pattern Type 2.0.0.0** and click **Enable**.

Base images: If you intend to use base images, import those images first from PureApplication System by using the steps that are provided at this website:

`http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/ICON/topics/ticn_importiwdimage.html`

3. Click **Cloud** → **Default Deploy Settings**.
4. Select the imported base image.
5. Click **Patterns** → **Virtual Applications**.
6. From the Virtual Applications pane, select the Image Construction and Composition Tool virtual application pattern and click **Deploy**. For more information about the deployment process, see 4.5.3, “Deploying virtual patterns” on page 132.
7. When the deployment process is complete, click **Endpoint** in the Application Detail pane.
8. In the Endpoint information pane, click the URL to start Image Construction and Composition Tool.

Important: The use of the URL that is provided in the Endpoint pane is the correct way to start Image Construction and Composition Tool. You cannot start Image Construction and Composition Tool by entering the URL in a browser or book mark it for later.

When the virtual application is deleted

If the Image Construction and Composition Tool virtual application pattern is deleted, you can create it again by using the steps in this task. After this task is complete, Image Construction and Composition Tool is available for use as a virtual application pattern. Complete the following steps to create the Image Construction and Composition Tool if the virtual application pattern was deleted:

1. In the Workload console tab of PureApplication System, click **Patterns** → **Virtual Applications**.
2. Click the **New** icon on the toolbar.
3. Select **IBM Image Construction and Composition 1.2** and click **Start Building**.

Figure 5-20 shows the panel that is used to select the Image Construction and Composition Tool virtual application pattern for building.

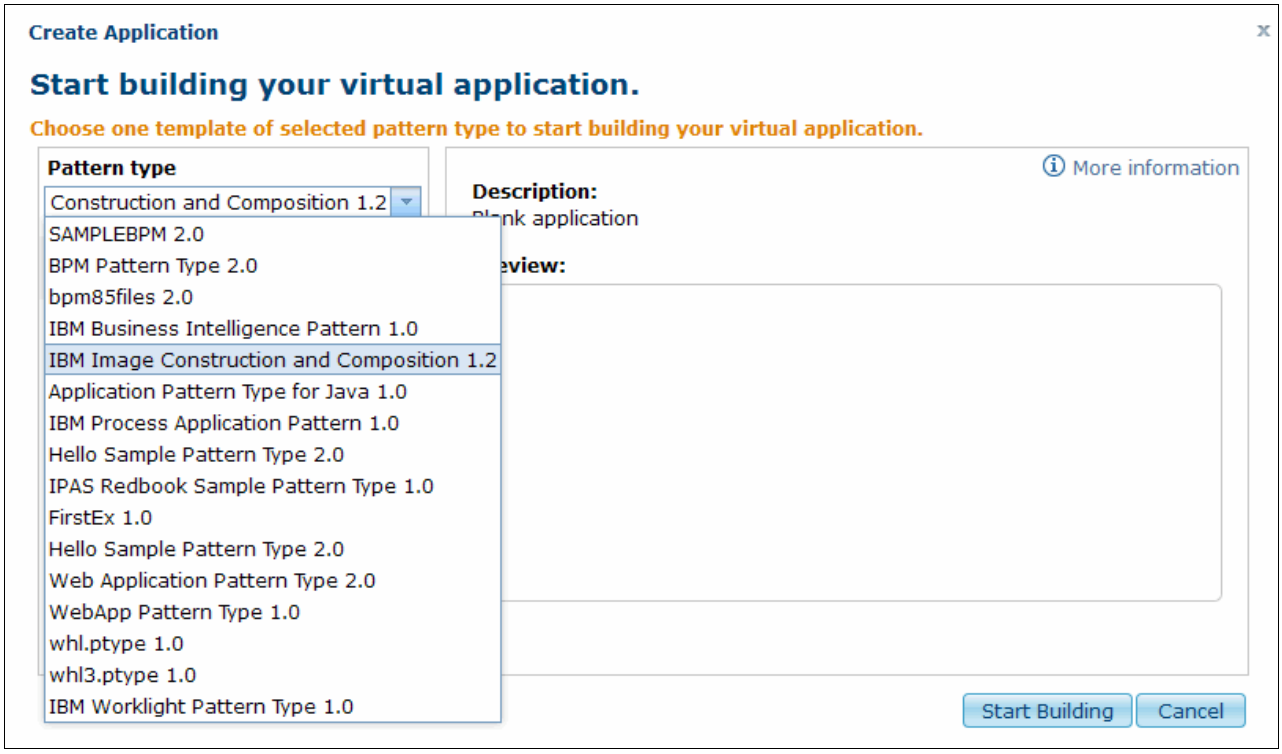


Figure 5-20 Building Image Construction and Composition Tool as a virtual application

4. Scroll down, open Other components, and drag Image Construction and Composition Tool onto the canvas.
5. Click **Save**.
6. From the Virtual Applications pane, select **Image Construction and Composition Tool virtual application pattern** and click **Deploy**. For more information about the steps to complete the deployment process, see 4.5.3, “Deploying virtual patterns” on page 132.

Figure 5-21 shows the panel that is used to deploy Image Construction and Composition Tool as a virtual application pattern.

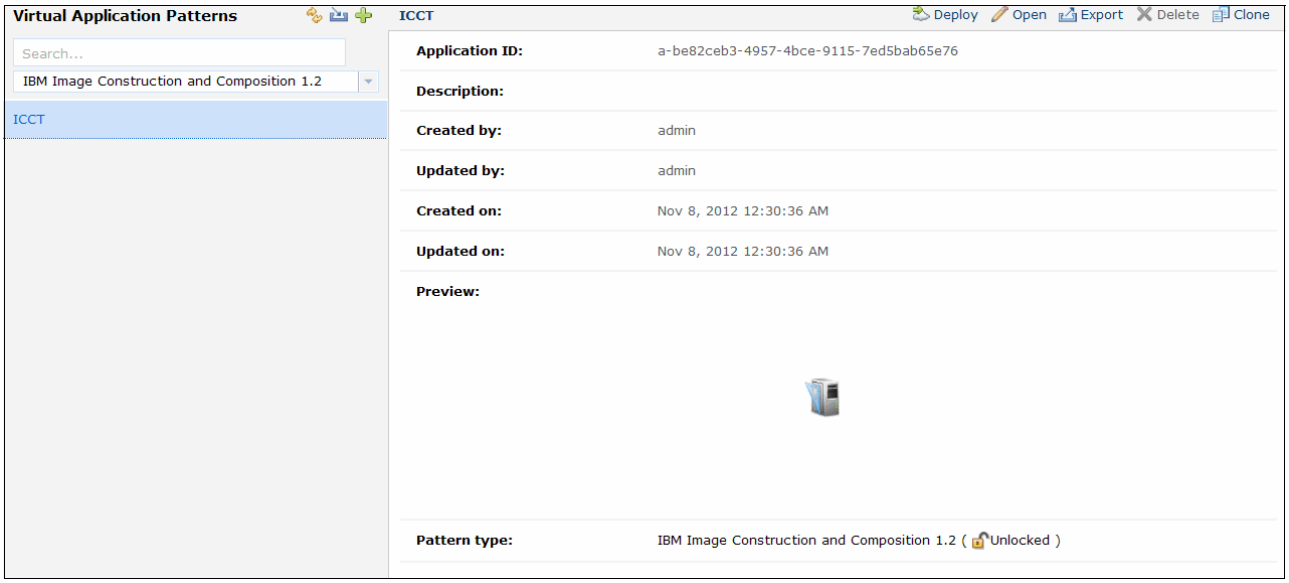


Figure 5-21 Deploying Image Construction and Composition Tool as a virtual application pattern

7. When the deployment process is complete, click **Endpoint** from the application detail pane.
8. In the Endpoint information pane, click the URL to start Image Construction and Composition Tool.

> **Important:** The use of the URL that is provided in the Endpoint pane is the correct way to start Image Construction and Composition Tool. You cannot start Image Construction and Composition Tool by entering the URL in a browser or book mark it for later.

5.3.3 Working with Image Construction and Composition Tool

This section describes how to configure PureApplication System as a cloud provider with Image Construction and Composition Tool. It also describes the common configuration and deployment tasks that must be performed with this setup.

Configuring IBM PureApplication System as a cloud provider

Complete the following steps to configure Image Construction and Composition Tool with PureApplication System as the cloud provider:

1. From the main menu of Image Construction and Composition Tool, click **Administer** → **Manage cloud providers**.
2. On the Cloud Providers page, click the **Add** icon.
3. On the Welcome page, click **Next**.
4. On the General page, specify the following details:
   - Name: Enter a name for the provider; for example, `PureApplication Cloud Provider`.
   - Description: Enter a suitable description.
   - Cloud Provider Type: Select **IBM Workload Deployer or IBM PureApplication System** and click **Next**.

Figure 5-22 shows the input panel for creating the PureApplication cloud provider.

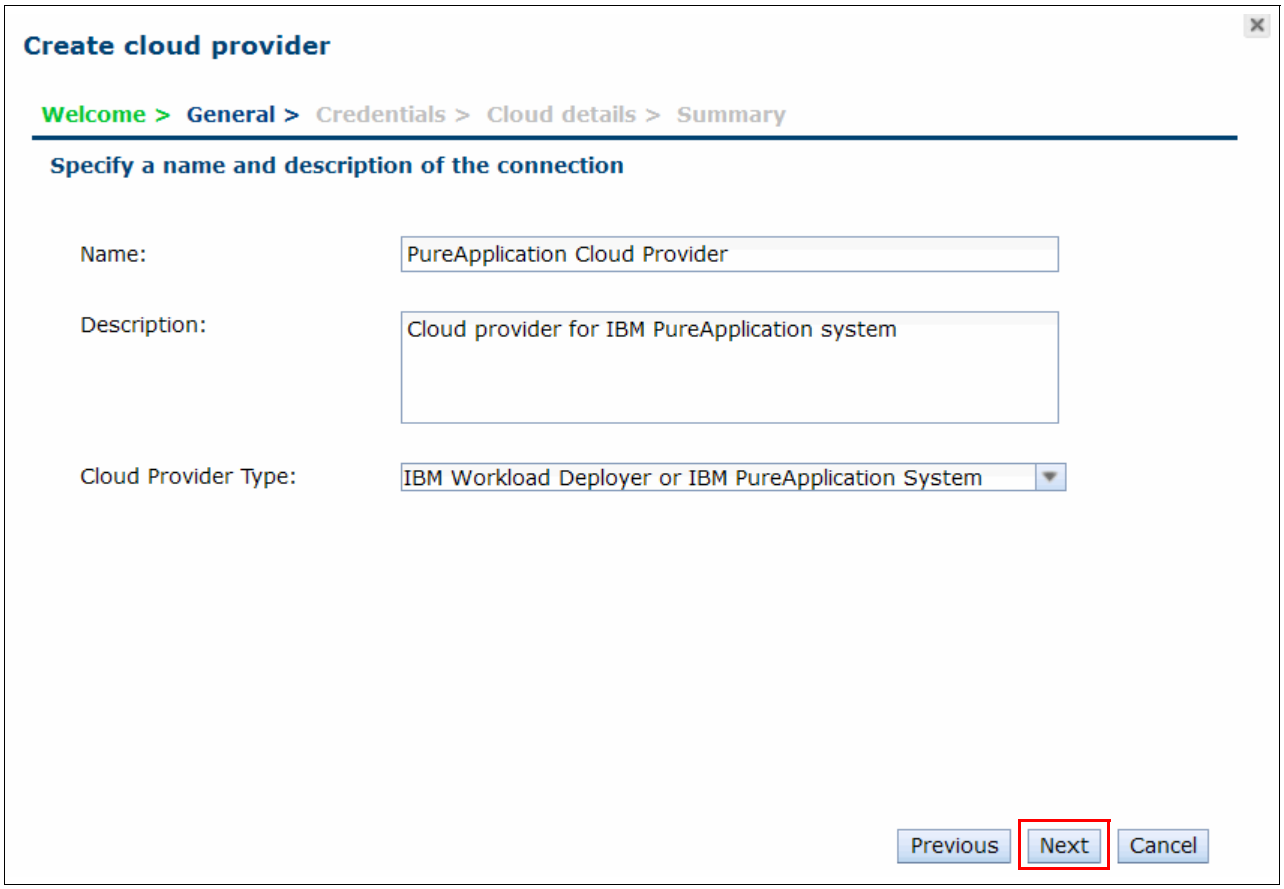


Figure 5-22 Creating PureApplication cloud provider

5. On the Credentials page, enter a user name and password. The user name must have access to the PureApplication System user interface and must have permissions to perform the intended actions.
6. On the Cloud details page, enter the host name or IP address of PureApplication System. Click **Done** to complete the process.

Figure 5-23 shows the submit panel when the PureApplication cloud provider is created.

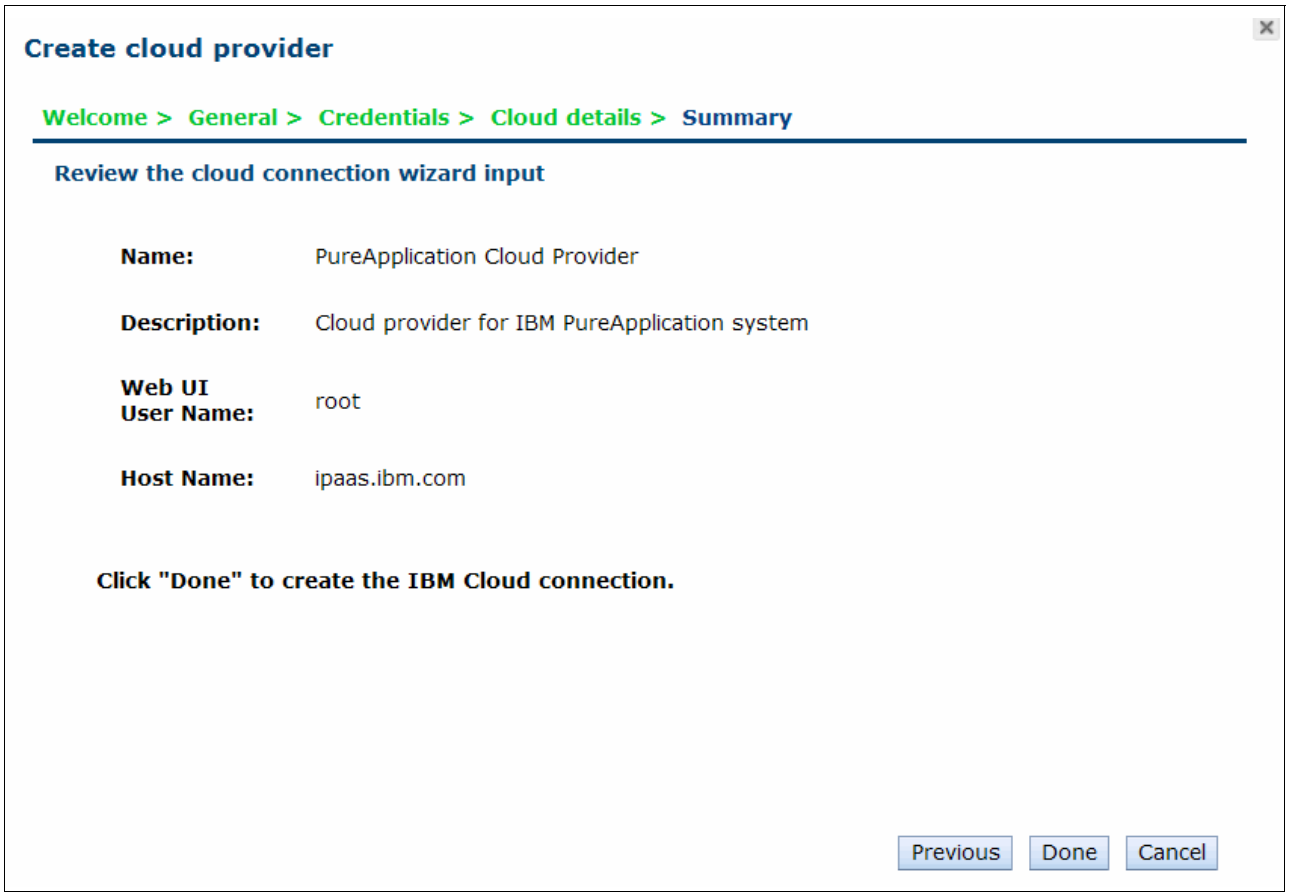


Figure 5-23 Submitting inputs for creating PureApplication System cloud provider

7. The user that is configured on the PureApplication System cloud provider needs permissions to complete the required tasks in PureApplication System. Complete the following steps on the PureApplication System user interface to ensure that the user is configured correctly:
   a. Click **System Console**.
   b. Click **System** → **User**.
   c. Select the user in the left navigation pane.
   d. In the Permissions section, ensure that the following options are selected:
      - Create new patterns
      - Create new catalog content
      - Cloud group administration: Read-only view

You successfully configured the cloud provider and made the assets from PureApplication System available for use in Image Construction and Composition Tool. Figure 5-24 shows the configuration of Image Construction and Composition Tool with PureApplication System.

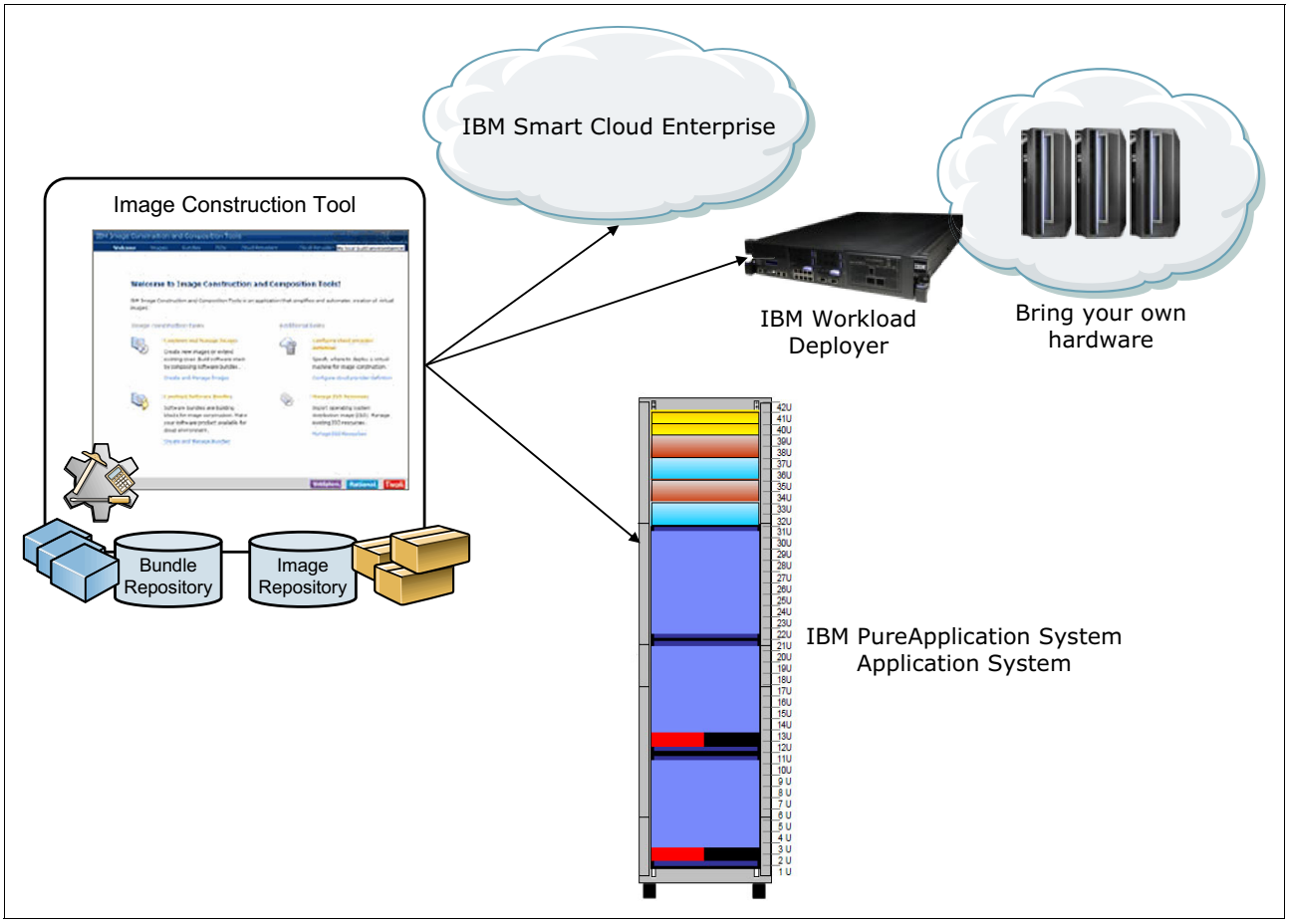


Figure 5-24 Configuring PureApplication System as cloud provider with Image Construction and Composition Tool

The following standard tasks are performed after the PureApplication System is configured as the cloud provider for Image Construction and Composition Tool:

- Import base images or other images (such as WebSphere) from PureApplication System catalog.
- Extend the PureApplication System images with software bundles.
- Synchronize or capture those images back into PureApplication System.
- Use images as part of virtual system pattern and deploy in PureApplication System.

The following sections describe each of these tasks and highlighting some of the important steps to be performed in each task.

Import and extend images from PureApplication System

Images can be imported from PureApplication System into Image Construction and Composition Tool and extended for customization.

Figure 5-25 shows the overall process of importing virtual images into the Image Construction and Composition Tool.

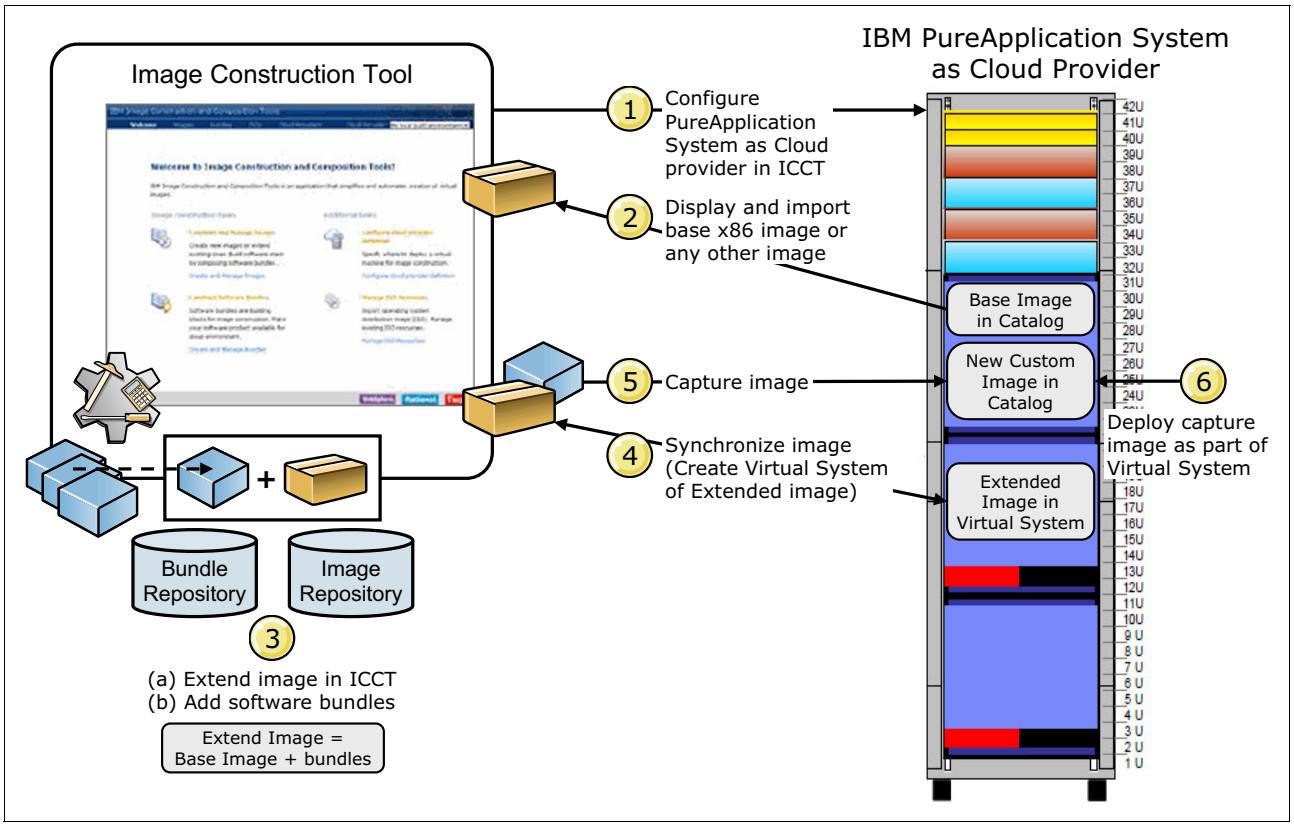


Figure 5-25 Importing or exporting images into Image Construction and Composition Tool from PureApplication System

Complete the following steps to import and extend images from PureApplication System by using the Image Construction and Composition Tool:

1. From the Image Construction and Composition Tool Welcome page, select **Images and bundles** → **Build and manage images**.
2. Select a cloud provider (in this case, **PureApplication Cloud Provider**, which corresponds to the PureApplication System). Click the **Import** icon to import the images from the PureApplication System.

Figure 5-26 shows how to import virtual images into Image Construction and Composition Tool.

Figure 5-26 Importing virtual images into Image Construction and Composition Tool

3. Filter the search category to narrow your search. Select images and add them to the import list, as show in Figure 5-27. Click **Import** to import the images.

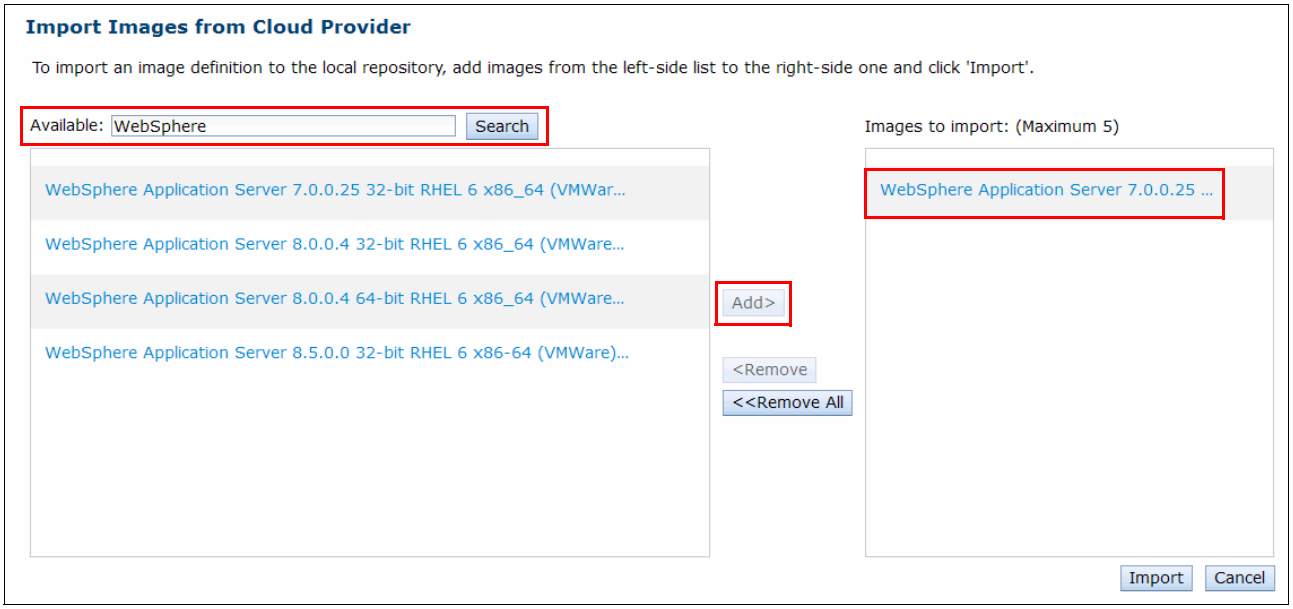


Figure 5-27 Filter and select virtual images from the list to import

4. Wait for a status update. When the image status changes to Completed with a green check mark beside it, the import is successful.

Figure 5-28 shows the image status upon completion of the import task.

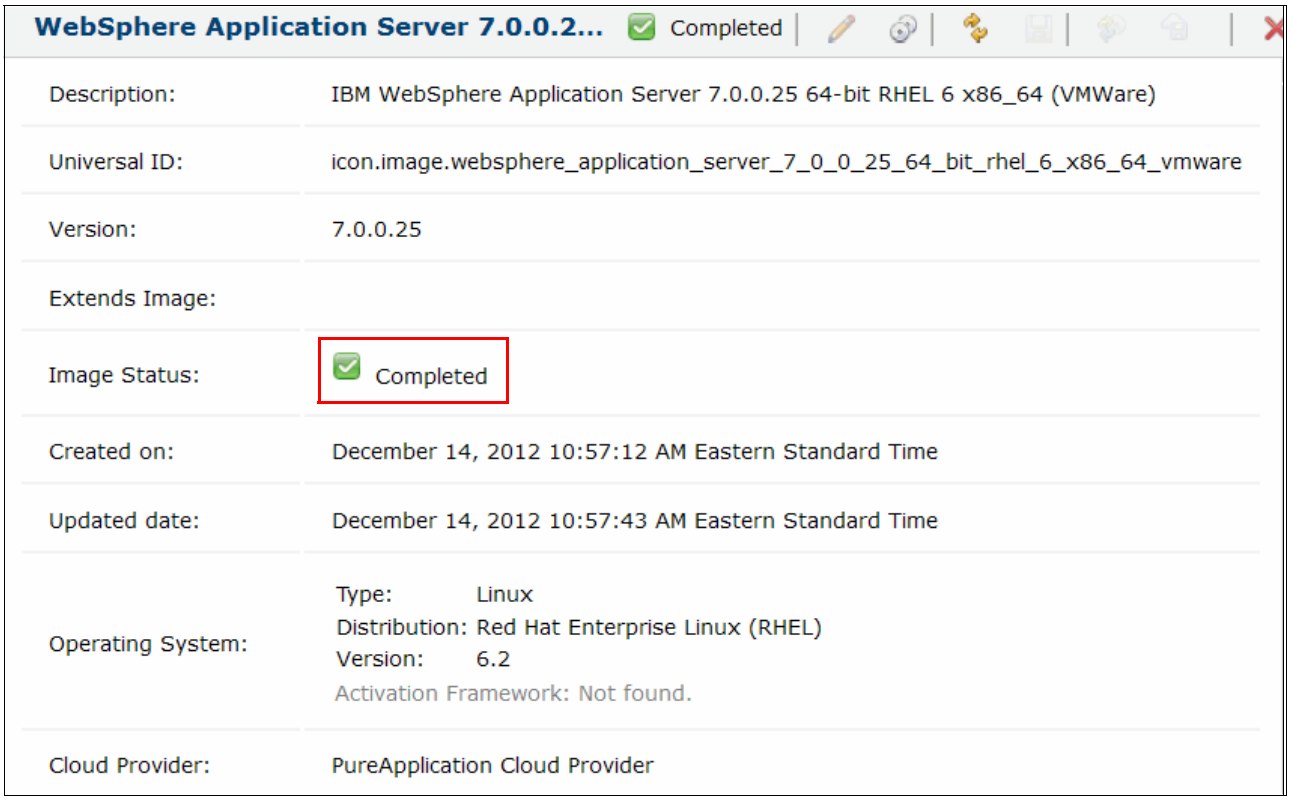


Figure 5-28 Virtual image import task status

Important: The actual image content is not transferred; only copy of metadata is saved.

5. Select the imported image. Click the **Extend** icon to extend this image.

   Figure 5-29 shows the panel for extending the imported virtual image.

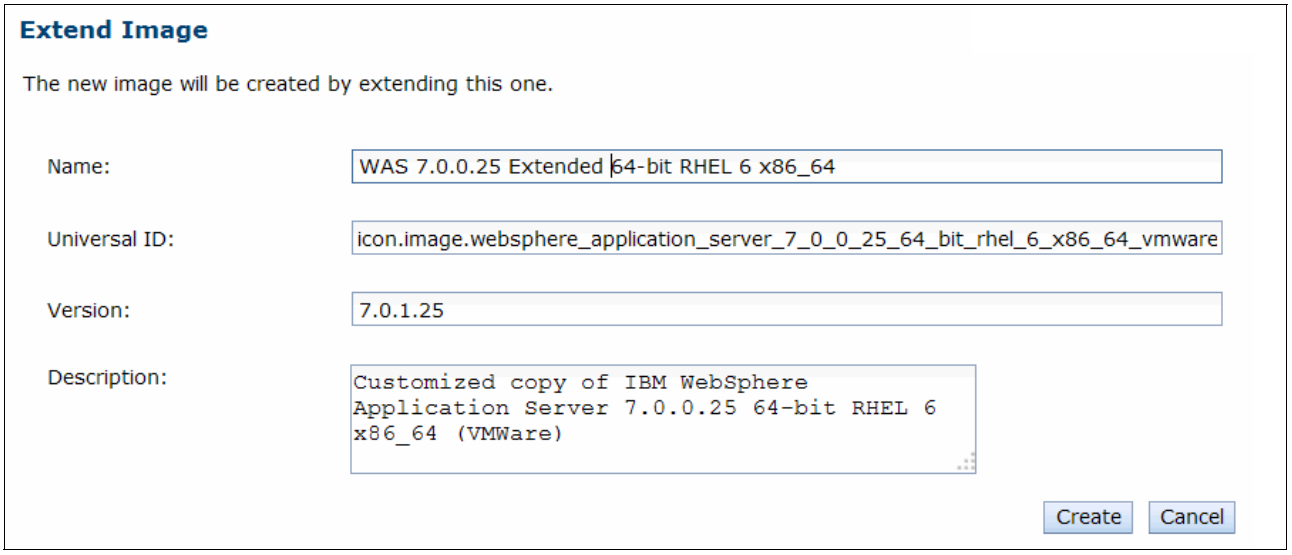


Figure 5-29 Extending a virtual image in Image Construction and Composition Tool

6. By using the Extend Image panel, enter a new image name, universal identifier, version, optional description, and then click **Create**. The new image is now listed under the Images panel.

   Figure 5-30 shows the status of the virtual image after the extend task is complete.

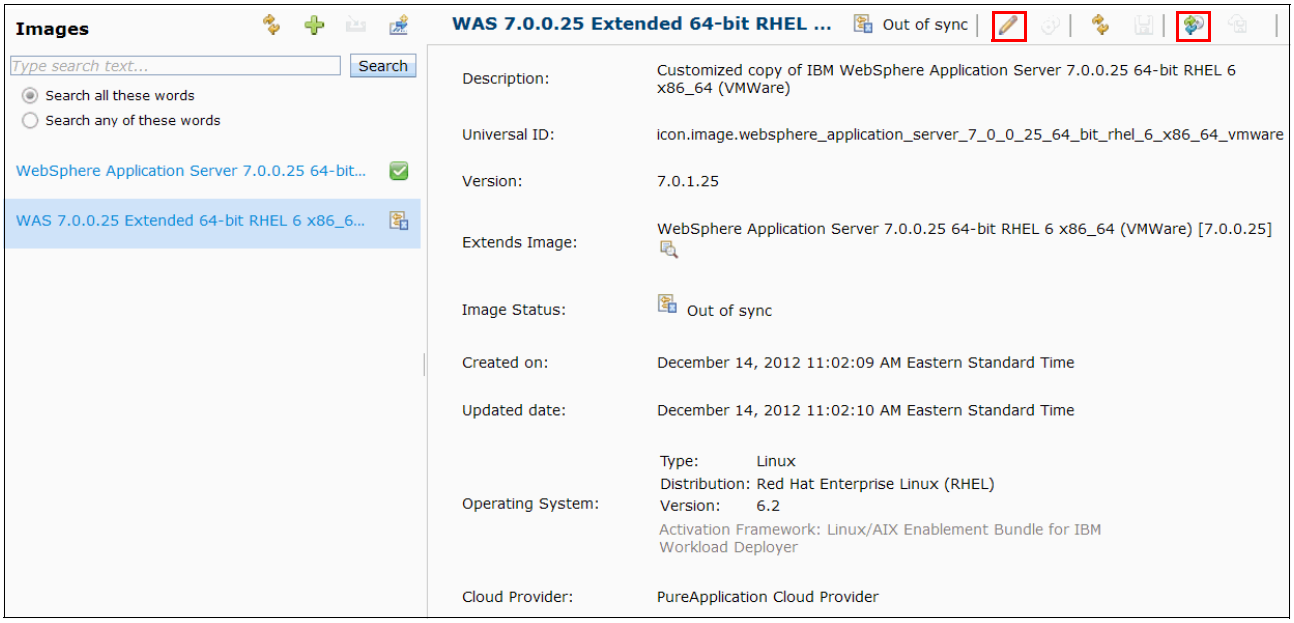


Figure 5-30 Extended virtual image status

After the extend task is complete, the virtual image status shows as Out of sync, as shown in Figure 5-32 on page 230. This chapter describes how to synchronize the virtual image with PureApplication System in a later section. The next section describers the process that is used to add software bundles to the virtual image.

Add software bundles to extended image

Now that the image is imported into Image Construction and Composition Tool, you can extend it by adding suitable software bundles. Complete the following steps to add a software bundle:

1. From the Image Construction and Composition Tool Welcome page, select **Images and bundles** → **Build and manage software bundles**.
2. Click the **import Image Construction and Composition Tool** icon.
3. A new panel opens in which for you enter information about the location of the software bundle and the credentials that are needed to connect to the path or location. Enter your information. Click **OK**, as shown in Figure 5-31 on page 229.

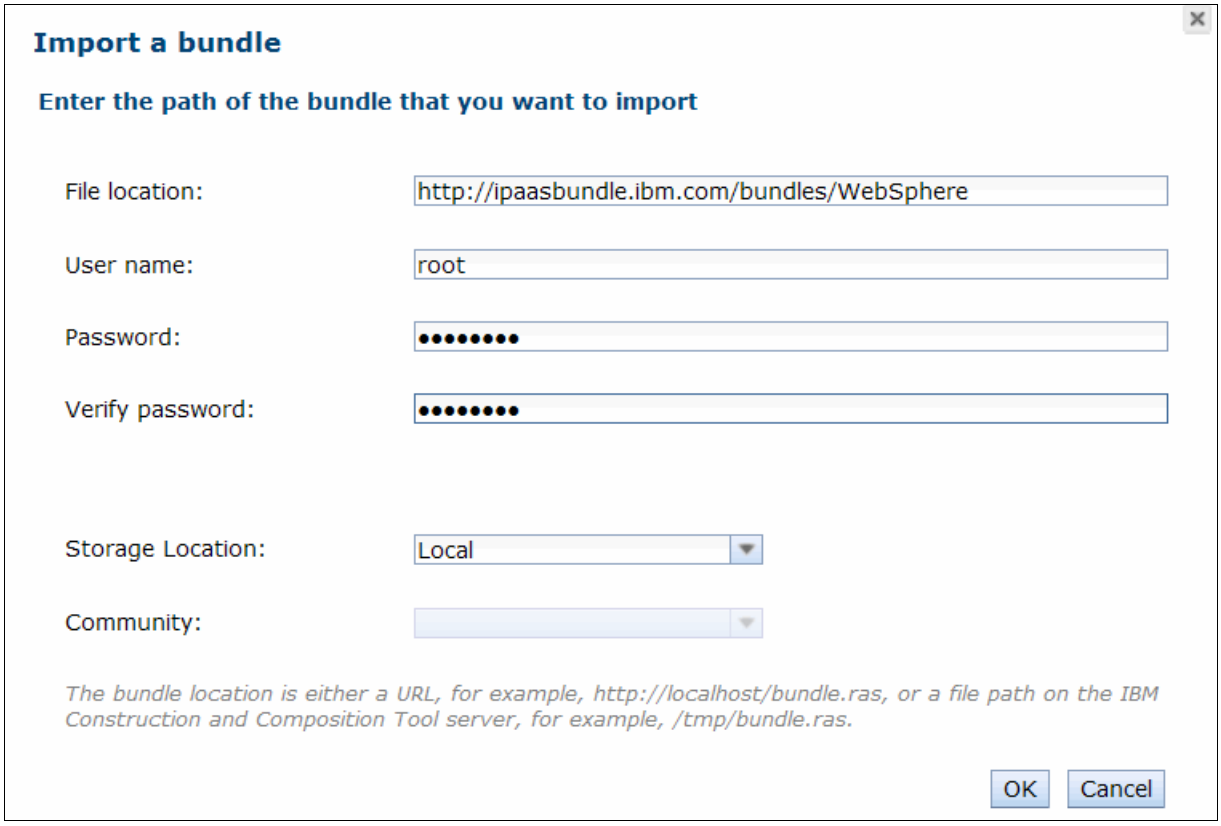


Figure 5-31 Importing software bundles in Image Construction and Composition Tool

All the software bundles from the location are imported into the Image Construction and Composition Tool.

4. To add a bundle to an extended image, browse to the **Images and bundles** → **Build and manage images** panel and select the image to extend.
5. Click **Edit image**.
6. Expand Software Bundles and click **Add**.

Figure 5-32 shows adding new software bundles.

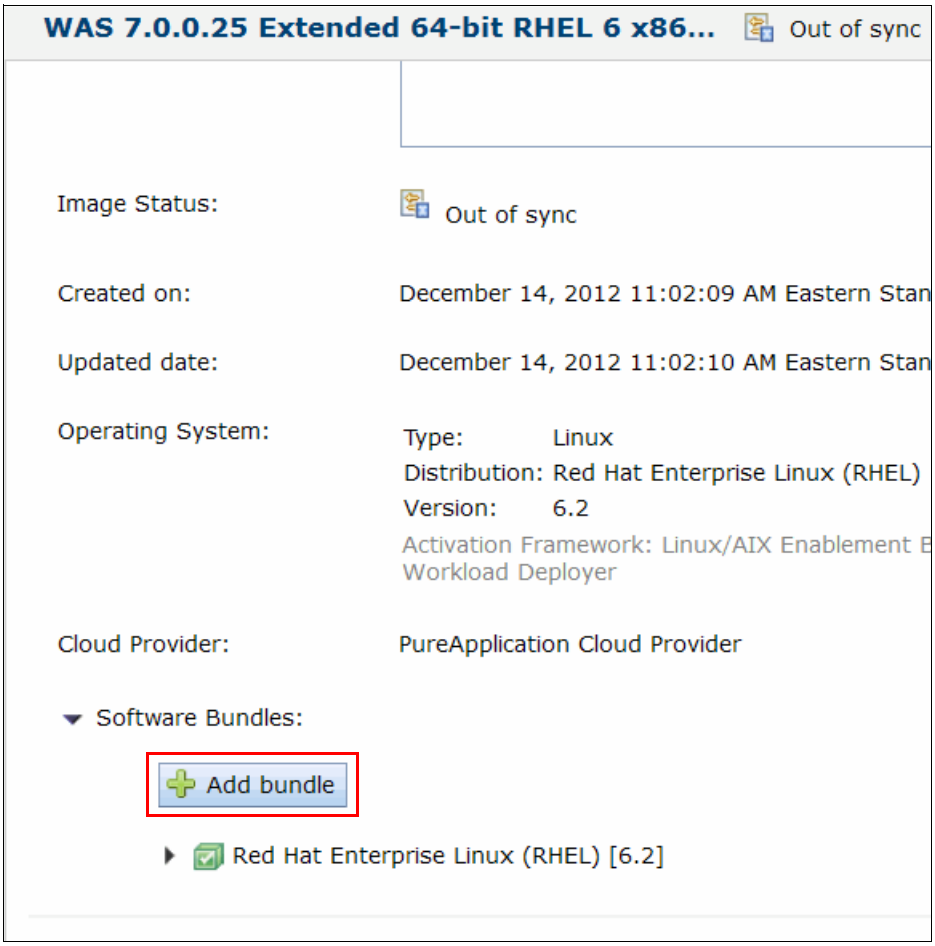


Figure 5-32 Adding software bundles to virtual image

7. Select the bundle from the list of bundles that is available in Image Construction and Composition Tool to add to the image.
8. Add more bundles as applicable. When you are finished with your selections, click **Save** → **Done editing**.

You configured the virtual image by adding suitable bundles from the bundles that are available in the PureApplication System. For more information about creating your own software bundles, see 5.3.4, “Creating software bundles with Image Construction and Composition Tool” on page 238.

Synchronizing virtual image to PureApplication System

After the image is customized by extending it with suitable software bundles, the next task is to synchronize the customized image with the cloud provider (PureApplication System). The synchronize process includes the following tasks:

- Transfers new image to the cloud provider (an image copy is made to the PureApplication System catalog).
- Creates a new virtual system pattern with the new image.
- Deploys the pattern and creates an instance (creates virtual machine for this image).
- Runs installation scripts in each of the bundles in the image.

Important: The installation scripts are run in the order the bundles are added into the image.

- Lays down configuration scripts for each of the bundles in the image.
- Prepares image for capture as a virtual image in the catalog.

Complete the following steps to synchronize the image:

1. Select the image in Image Construction and Composition Tool and click the **Synchronize** icon.
2. Enter the details, such as cloud groups to deploy the virtual system and the credentials for the virtual system. Click **Done**.

   Figure 5-33 shows the panel that appears for synchronizing the image.

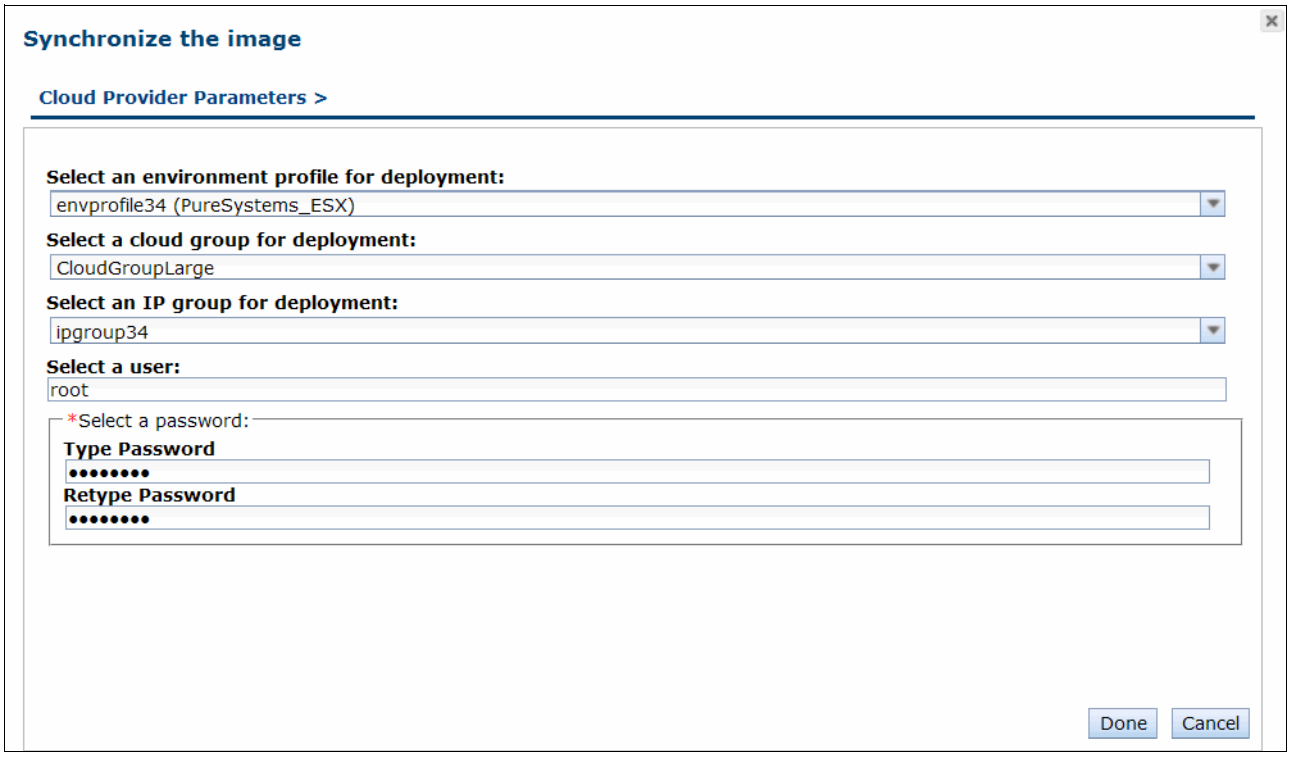


Figure 5-33 Synchronizing the virtual image in Image Construction and Composition Tool

The status changes to Synchronizing while Image Construction and Composition Tool communicates with PureApplication System to obtain the status of the deployed virtual machine of the image.

Figure 5-34 shows the status of synchronization of the virtual image.

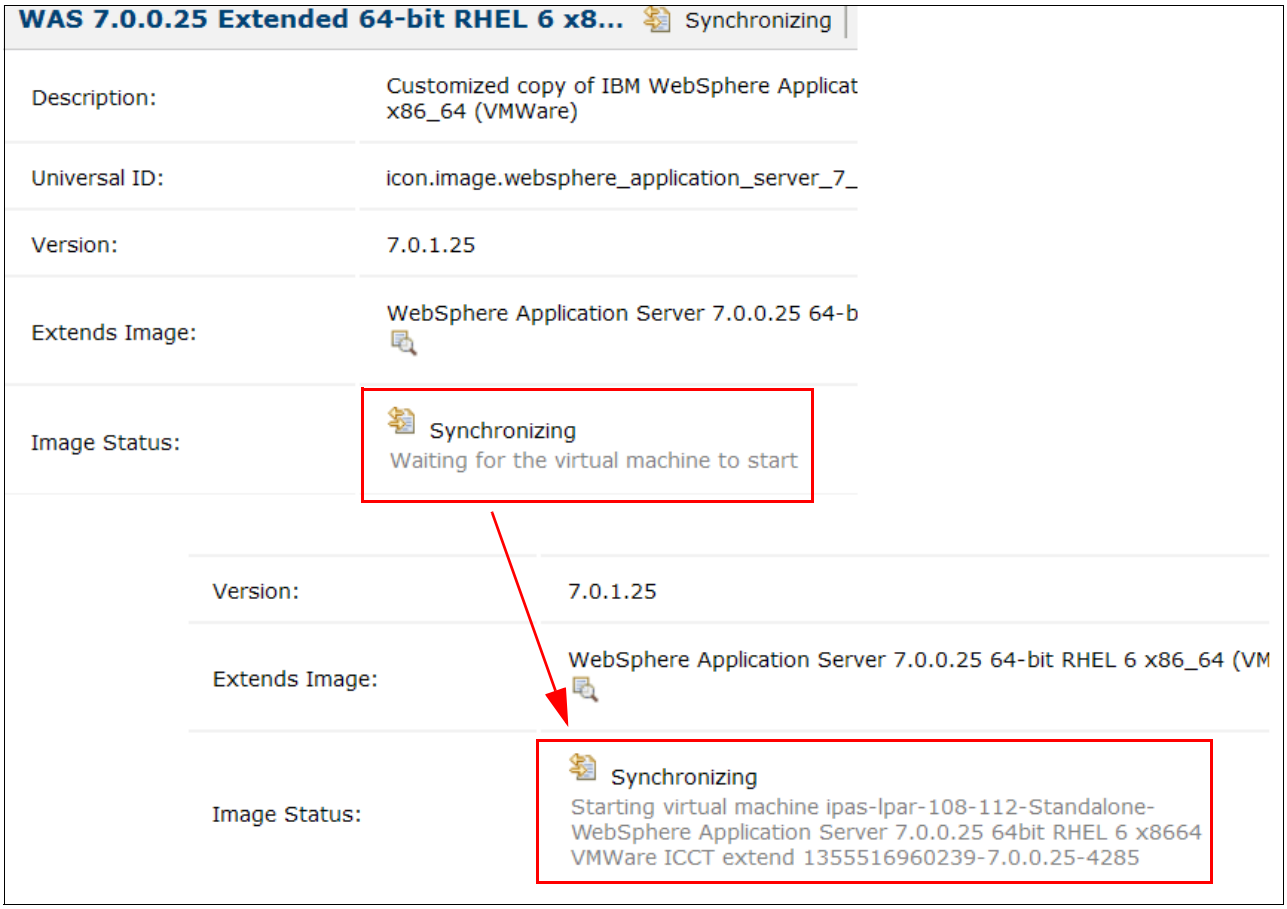


Figure 5-34 Virtual image synchronization status

3. Refresh the image in Image Construction and Composition Tool to monitor the synchronization progress. This process can take some time to complete.
4. A temporary virtual image is created in the PureApplication System. It can be viewed in the PureApplication System Workload Console by clicking **Catalog** → **Virtual Images**.

Figure 5-35 shows the newly created virtual image during synchronization.

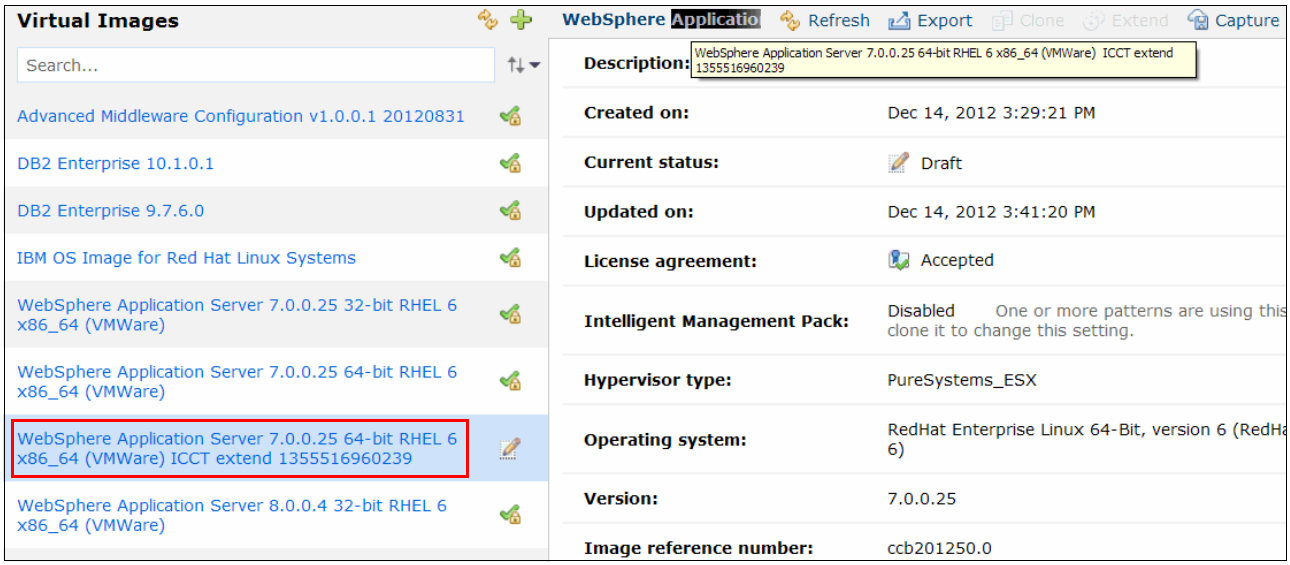


Figure 5-35 Virtual image during synchronization

5. A temporary Virtual system pattern is created and deployed as an instance. It can be viewed in the PureApplication System Workload Console by clicking **Instances** → **Virtual Systems**.

   Figure 5-36 shows the deployed virtual system pattern after synchronization.

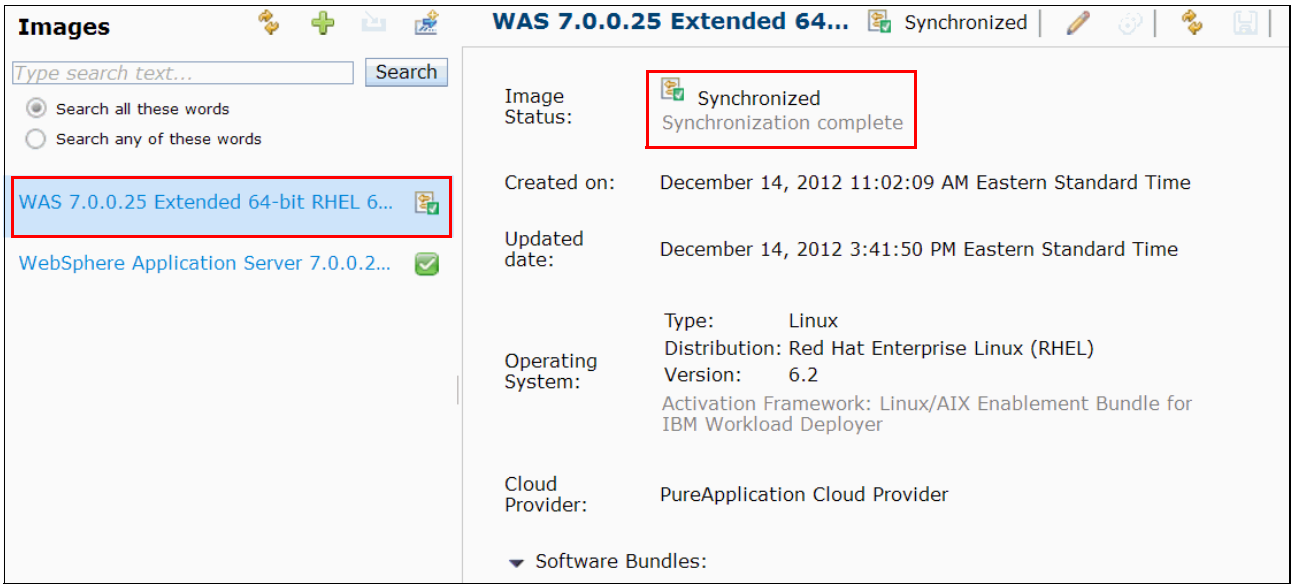


Figure 5-36 Virtual image after synchronization

After synchronization is complete, Image Construction and Composition Tool displays synchronization as complete and PureApplication System reflects the virtual machine as running.

Capture and construct virtual images

The deployed image can be captured from Image Construction and Composition Tool (after synchronization) and then used to create a virtual image in the PureApplication System.To capture the virtual image, select the synchronized virtual image in Image Construction and Composition Tool and click the **Capture** icon.

The capture feature creates a copy of the virtual image with the same name in the PureApplication System's catalog. The temporary virtual system pattern (which is accessed by clicking **Instances** → **Virtual Systems**) and virtual system instance (which is accessed by clicking **Patterns** → **Virtual Systems**) were created during the synchronize step and are deleted.

Figure 5-37 shows the process of capturing images, and its status.

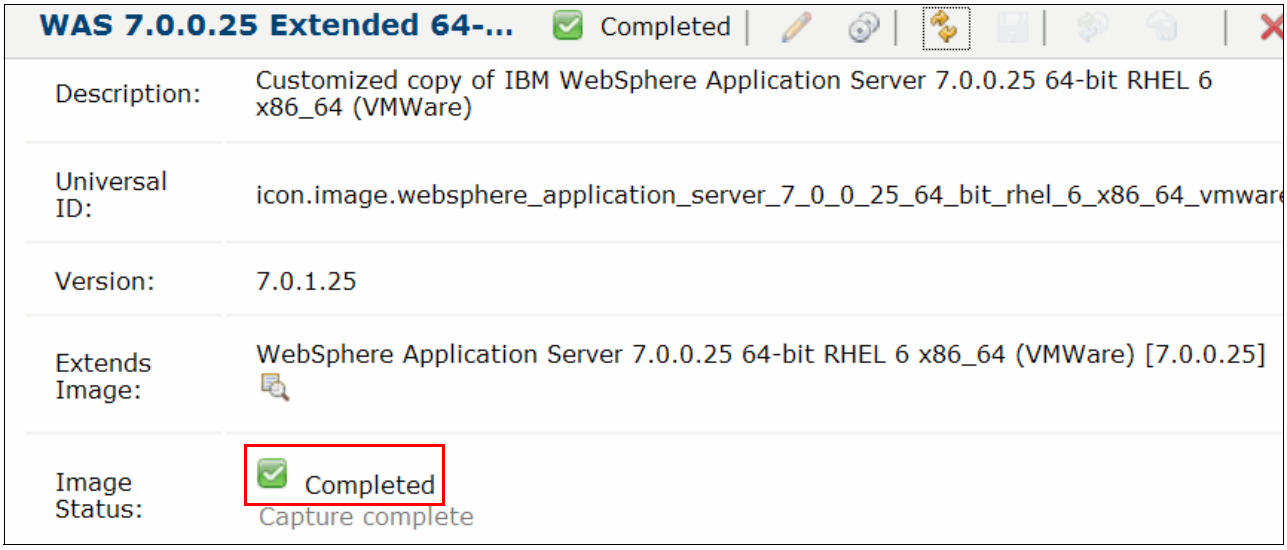


Figure 5-37 Virtual image capture status

However, the virtual image (which is accessed by **Catalog** → **Virtual Images**) remains. The name of the virtual image is changed to match the name of the image that was created in the Image Construction and Composition Tool.

Figure 5-38 shows the capture results in PureApplication System.

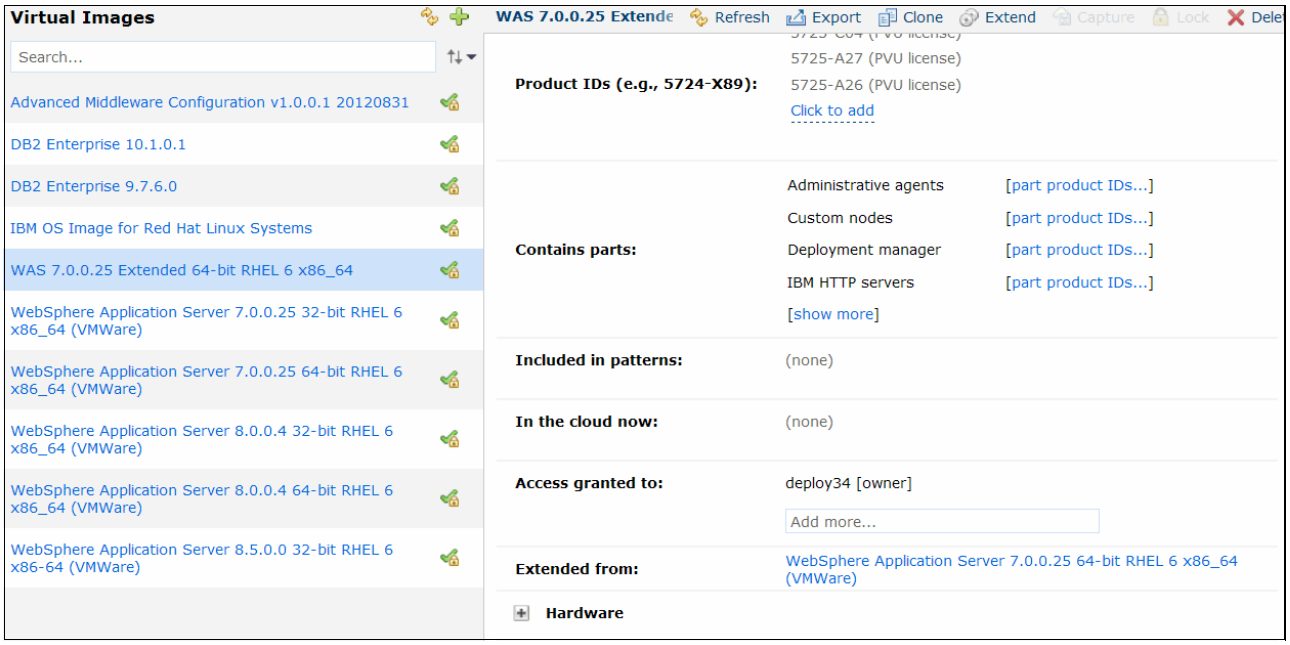


Figure 5-38 Captured virtual image in PureApplication System

Complete the following steps to use the new catalog image to create virtual system patterns:

1. In the pattern editor, select the part that corresponds to the new virtual image you recently captured.
2. Add this image part to the pattern.

   Figure 5-39 shows the pane that is used when a custom node virtual image is added to a pattern.

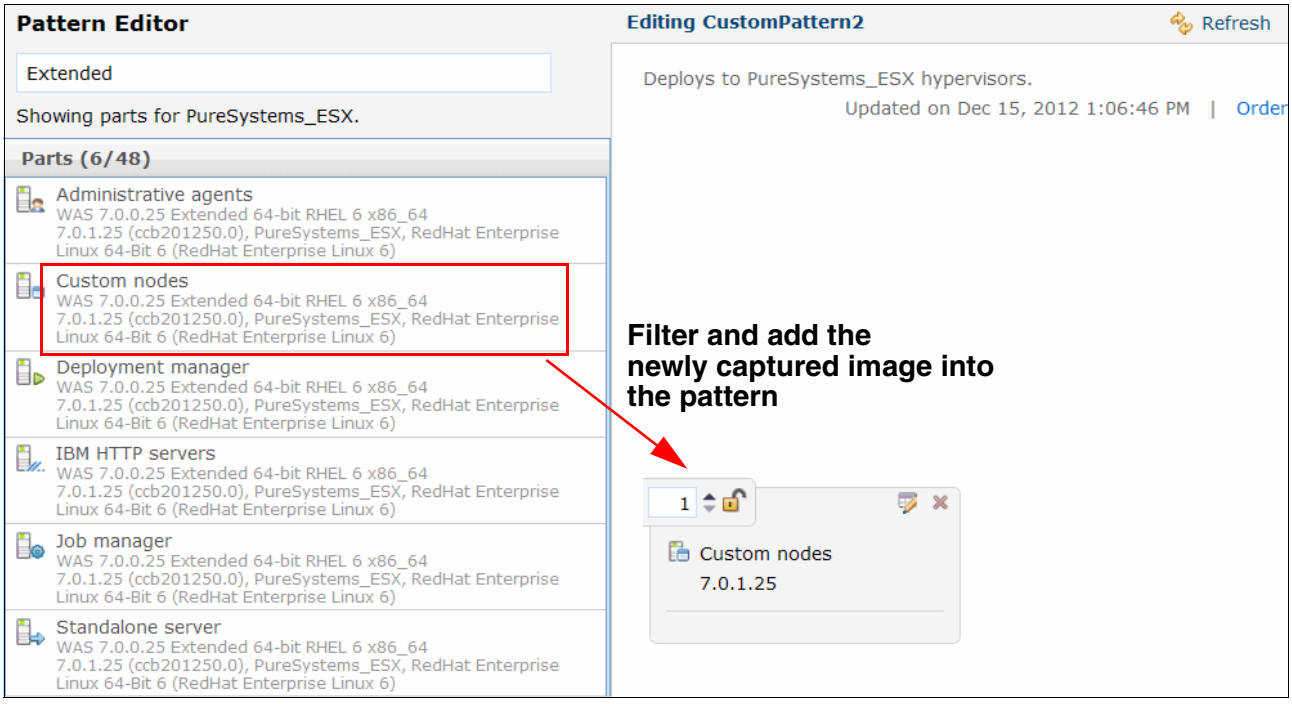


Figure 5-39 Adding virtual image to the virtual system pattern

3. Deploy the pattern into the cloud. For more information about deploying patterns, see Chapter 4, “PureApplication Deployment Models” on page 107.
4. During deployment, parameters that correspond to the software bundles that are included in the virtual image are prompted for your review and any changes.

You successfully deployed a pattern in the PureApplication System by capturing and constructing the virtual image by using Image Construction and Composition Tool.

Building your own operating system by using Image Construction and Composition Tool

There are situations where the available operating systems in the PureApplication System do not meet the business needs. In such cases, you can build your own operating system images by using Image Construction and Composition Tool.

Complete the following steps to build your own operating system:

1. Create a virtual machine in a hypervisor external to the PureApplication System. The following operating systems can be used with the virtual machine:
   - Red Hat Enterprise Linux v6.0
   - Red Hat Enterprise Linux v6.1
   - SUSE Linux Enterprise Server v11.1

   The virtual machine setup must meet the following requirements or restrictions:
   - No LVM is used
   - Disable Network manager
   - Install VMWare tools
   - Disable SELinux
   - Delete Snapshots
2. In Image Construction and Composition Tool, select **Create Image from running VM** to import the image from the running virtual machine. Enter the requested information and click **Create**.

   Figure 5-40 shows the panel that is used to create virtual images by using a running VM.

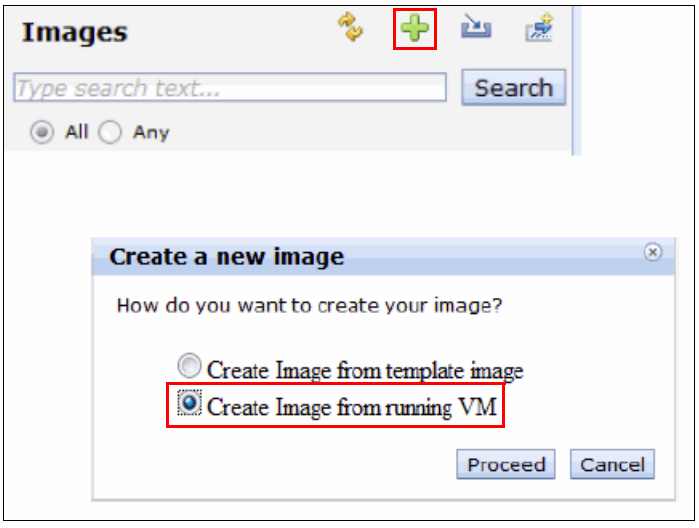


Figure 5-40 Creating a virtual image from a running VM

3. After the image import is complete, export the image as Open Virtualization Archive (OVA) to a host with SCP enabled. Provide information to the host on where SCP is installed and other necessary details.

   Figure 5-41 shows the panel that is used for exporting the virtual image as an OVA file.

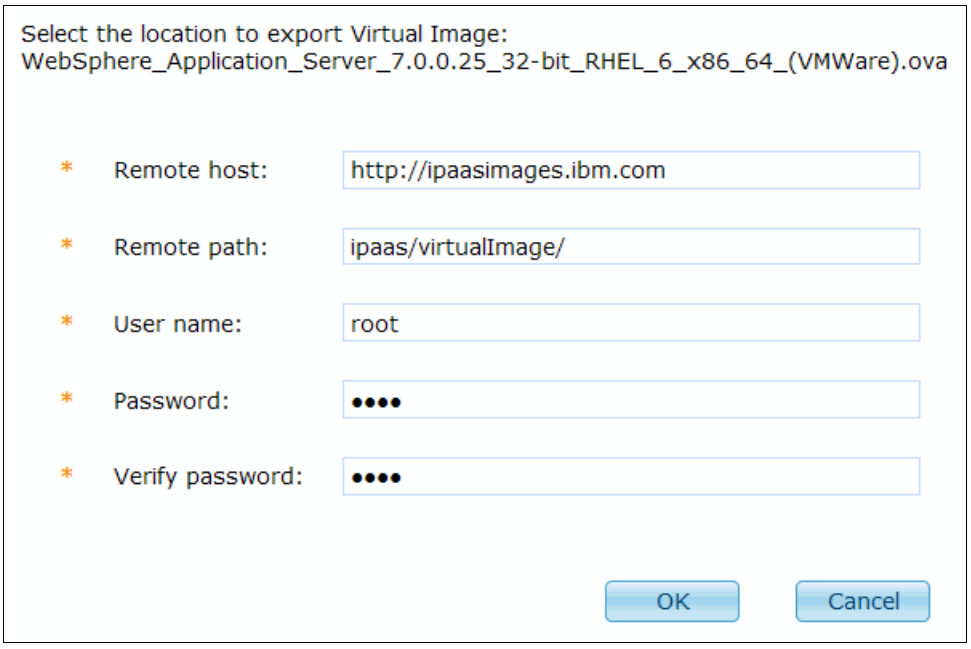


Figure 5-41 Exporting virtual image as an OVA file

4. Import the OVA file into PureApplication System. From the Workload console panel, select **Catalog** → **Virtual Images** → **Add**.

   Figure 5-42 displays the opened panel for importing an image from an OVA file.

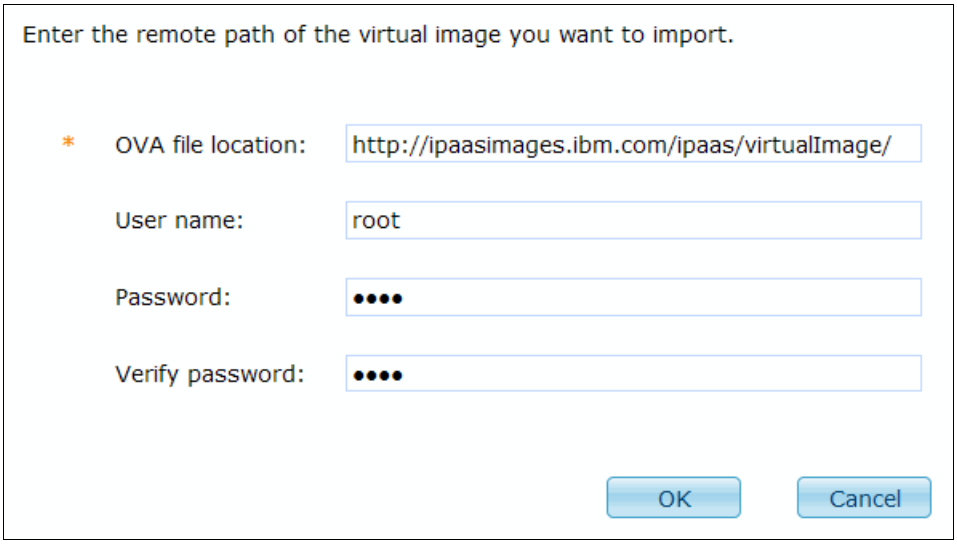


Figure 5-42 Importing a virtual image from an OVA file

5. Verify that the imported image is accurate.

5.3.4 Creating software bundles with Image Construction and Composition Tool

The software bundles represent the software that you want to install within the virtual image. Previous sections in this chapter described how to add software bundles and extend virtual images. However, for certain scenarios, you might have to build your own software bundles when the existing bundles do not completely meet the requirements. This section describes the process of creating your own software bundles by using Image Construction and Composition Tool.

Overview

The following primary parts are contained in software bundles:

- Install: Used to specify to Image Construction and Composition Tool the software that is installed. It also provides necessary scripts to install components (if applicable), and how to start the scripts. The installation task runs only once initially during the image creation. This task allows the inclusion of large binary files directly into the image, long-running installation tasks, or other actions.
- Configure: Used to specify the actions that configure the installed software on the image. The configuration scripts and corresponding script parameters are provided, which can be exposed so that users can provide inputs during deployment. The configuration scripts become part of the image's activation framework and runs during each image deployment process. This configuration supports multiple deployments for different scenarios by using a single virtual image.

For each software bundle, you can specify the following information or tasks:

- Requirements: When you add a software bundle to an image, Image Construction and Composition Tool checks that prerequisites such as type of OS and version are met.
- Installation: Defines the files to copy and commands to run to install and define parameters.
- Configuration (Activation): Define tasks that run during the virtual image activation process and make necessary configuration updates that are based on data, such as IP addresses, host names, and other deployment time variables.
- Firewall rules: Indicate network ports and port ranges that are open.
- Reset: Include scripts to clean up and reset any files that you do not want in the final image. Reset scripts are run before the image is captured to reset the image state to ensure that any unplanned content is not captured.
- Syntactic validation: Checks that all required fields have realistic values.
- Semantic validation: Performed by the server and run only when changes are saved. The validation report entries have three severities: info, warning, or error.

Bundle creation

Image Construction and Composition Tool provides a simple interface for managing the software bundles to easily create, import, or export those bundles. Complete the following steps to create a bundle:

1. From the Image Construction and Composition Tool Welcome page, click **Images and bundles** → **Build and manage software bundles**. Click the **New Bundle** icon. Figure 5-43 shows the panel that opens when a new bundle is created.

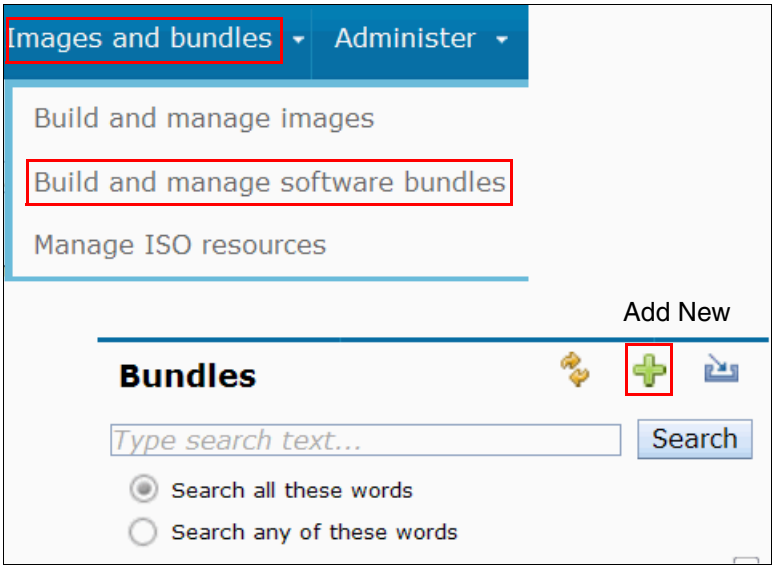


Figure 5-43 Adding a new software bundle

2. You must provide information such as bundle name, universal ID, version, storage location, and description. Then, select the option for Installation manager, if applicable. (This option is needed if the respective product images require Installation Manager for maintenance activities.) Click **Create**. Figure 5-44 on page 240 shows the bundle creation user input panel.

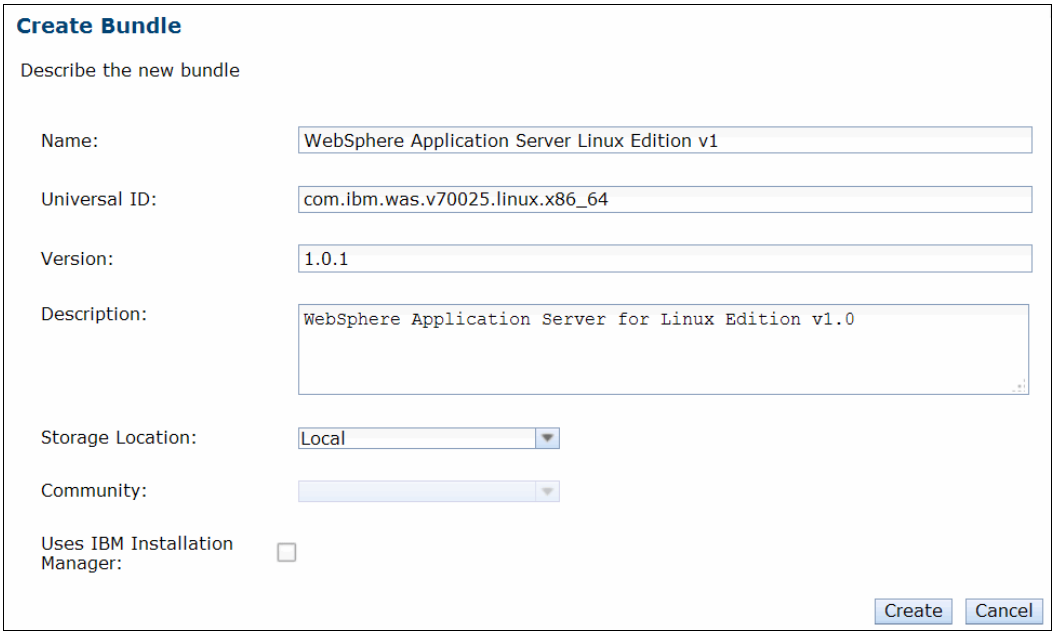


Figure 5-44 Add new software bundle properties panel

3. The General tab is displayed with the values that were added. Click the **Add** icon in this tab under the Products in the bundle section. Add the products to be included in the bundle. In this case, this bundle contains WebSphere Application Server and a Software Development Kit (SDK).

Figure 5-45 shows the General tab properties with the products included in the bundle.

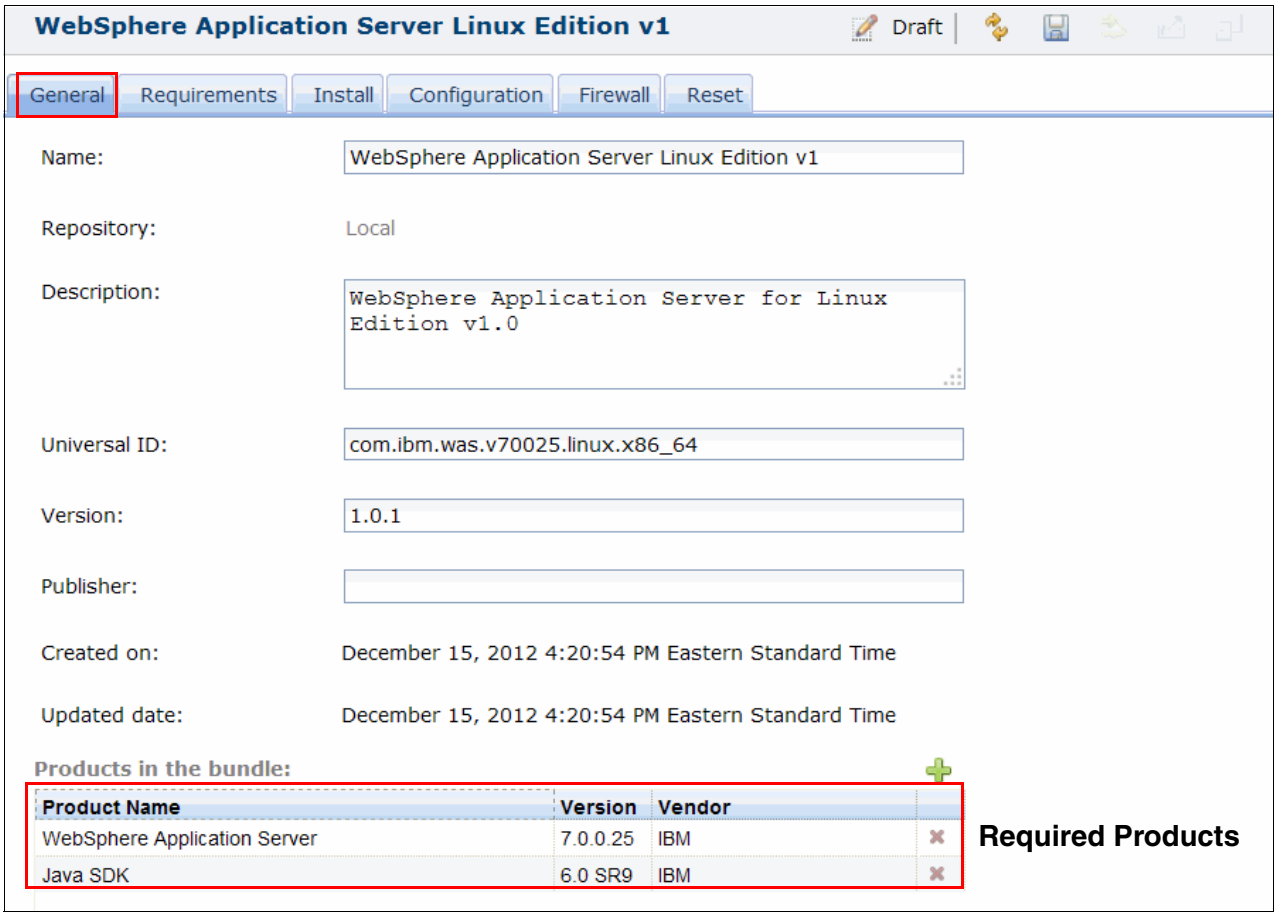


Figure 5-45 Configuring general settings for bundle

Important: After products are added, review the bottom of the page for Validation Status messages. You might see an error message that states the bundle must have at least one operation defined. Subsequent steps in this chapter cover defining an operation.

4. Browse to the Requirements tab and under the Supported Operating Systems panel, expand the plus (+) sign that reads Any and choose **Linux** from the drop-down list.

Figure 5-46 shows choosing an OS in the Requirements tab.

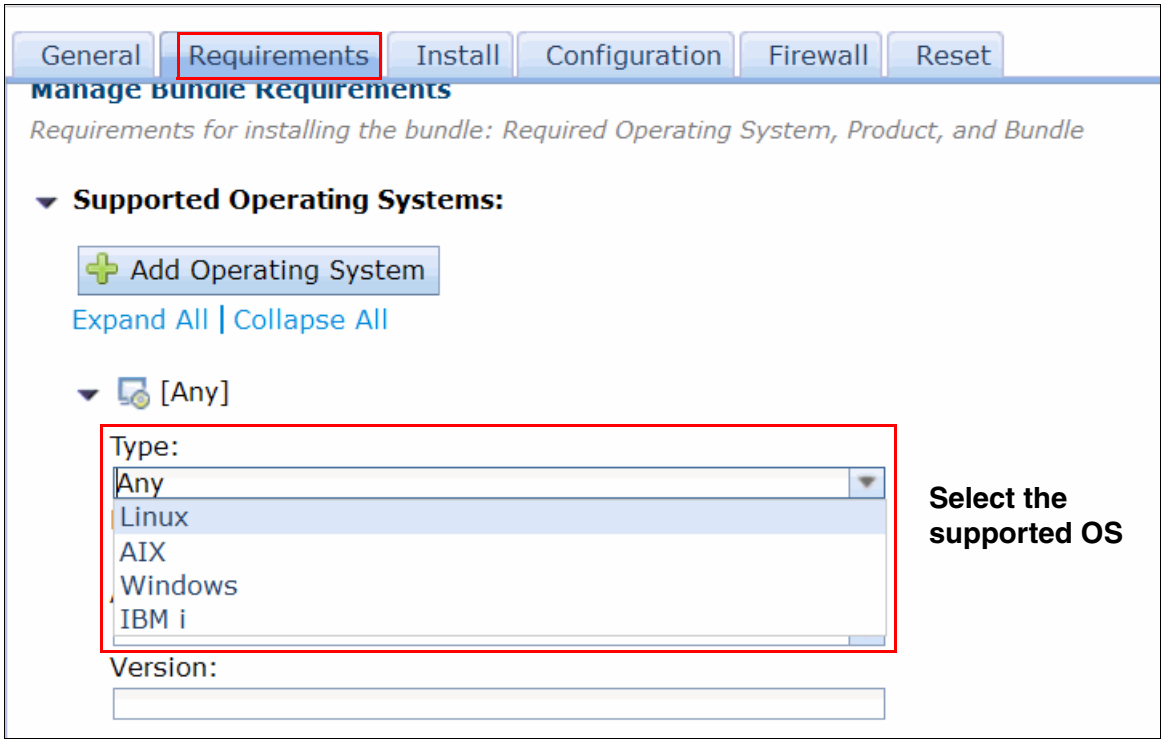


Figure 5-46 Selecting an OS for the bundle

5. Under the Required Software sections, define the software products that are required by the bundle (such as WebSphere MQ) and the corresponding version. If no version is specified, the bundle makes no distinction. Similarly, add the required bundles under the Required Bundles section.

   Figure 5-47 shows the bundle that contains OS and required products.

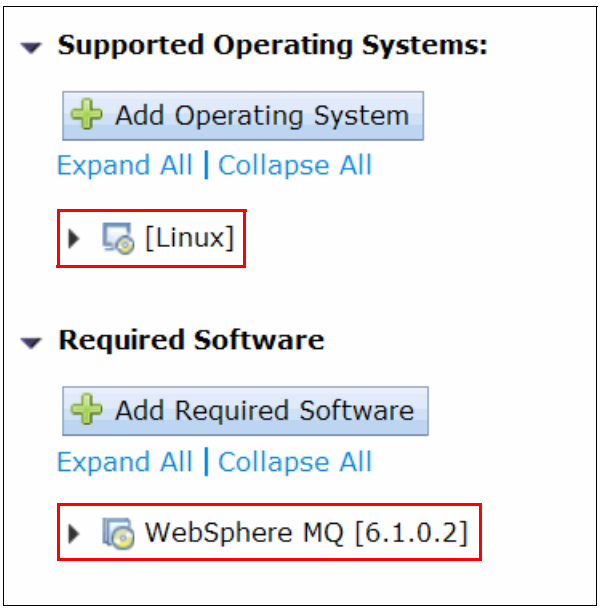


Figure 5-47 Add OS and Required Software for the bundle

6. Browse to the Install tab and under the Files to Copy section, click the **Add** icon. Choose the location of the script to upload and click the **Make executable** option. Enter the values for the Run Command and the Run As fields.
7. Under the Arguments section, enter the arguments or parameters that are used by the command to run. Figure 5-48 shows the panel with the configured installation properties.

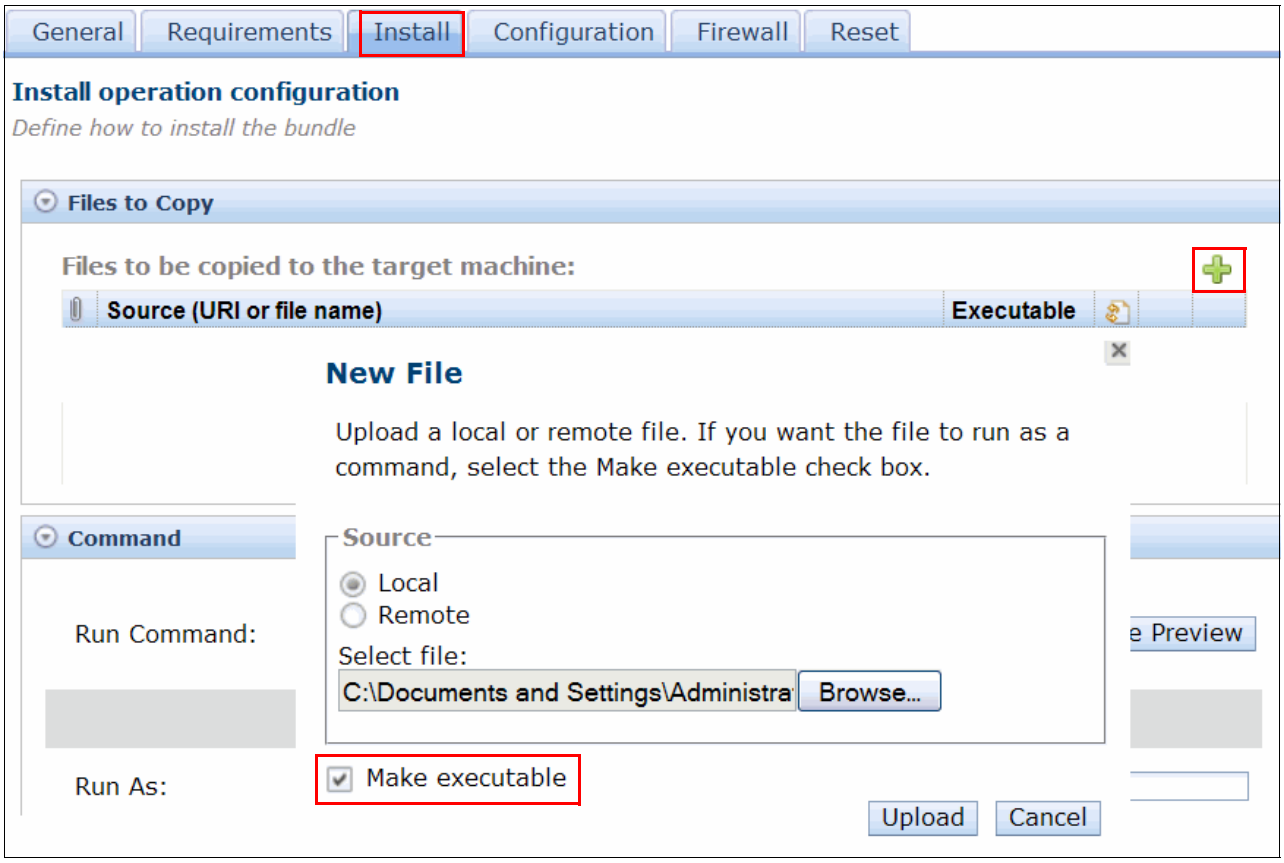


Figure 5-48 Provide arguments and commands to be run for installation script

Figure 5-49 shows the installation arguments for the software bundle.

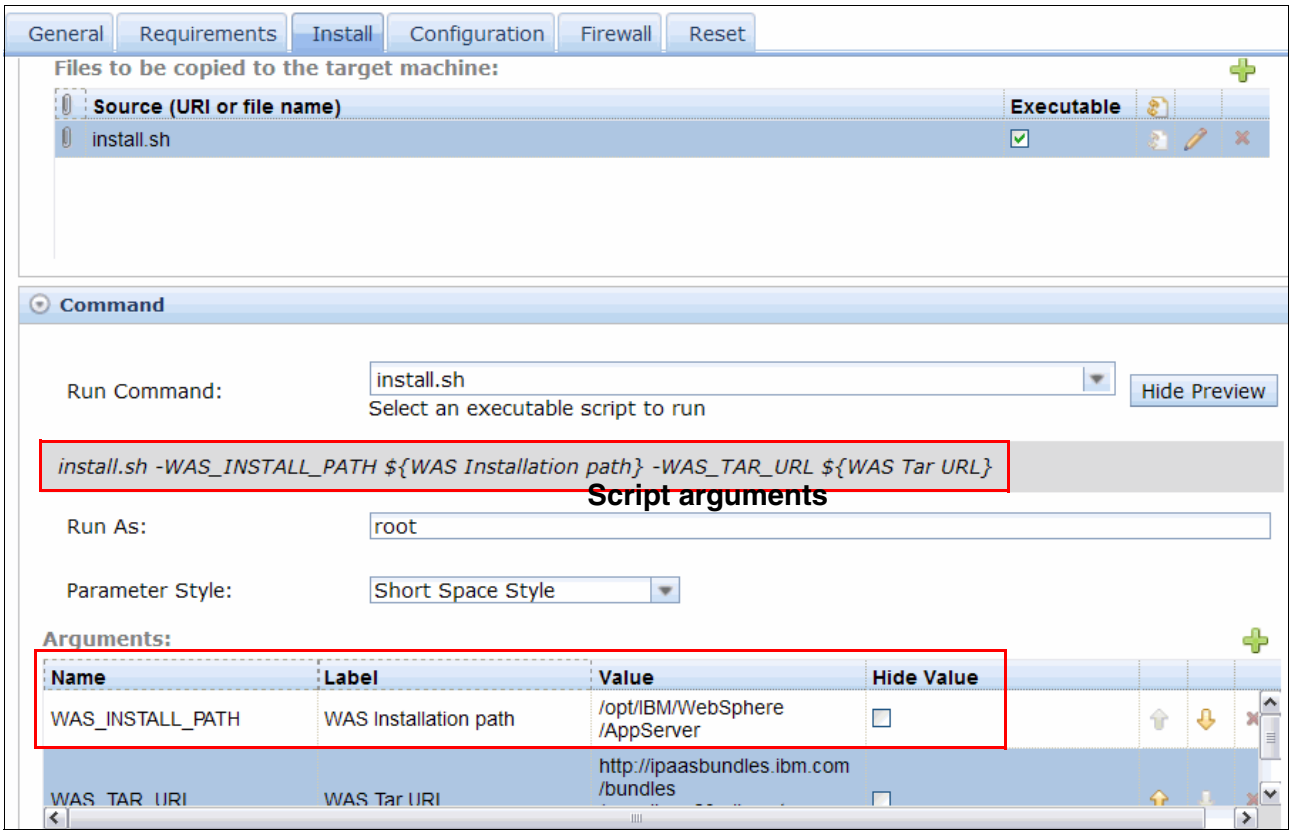


Figure 5-49 Install arguments for the software bundle

8. Browse to the Configuration tab and under the Files to Copy section, click **Add** and select the configuration script. Enter an Operation Name and click **Executable** option. Select the script from the Run Command drop-down list. Figure 5-50 on page 245 shows the configuration parameters for the software bundle.

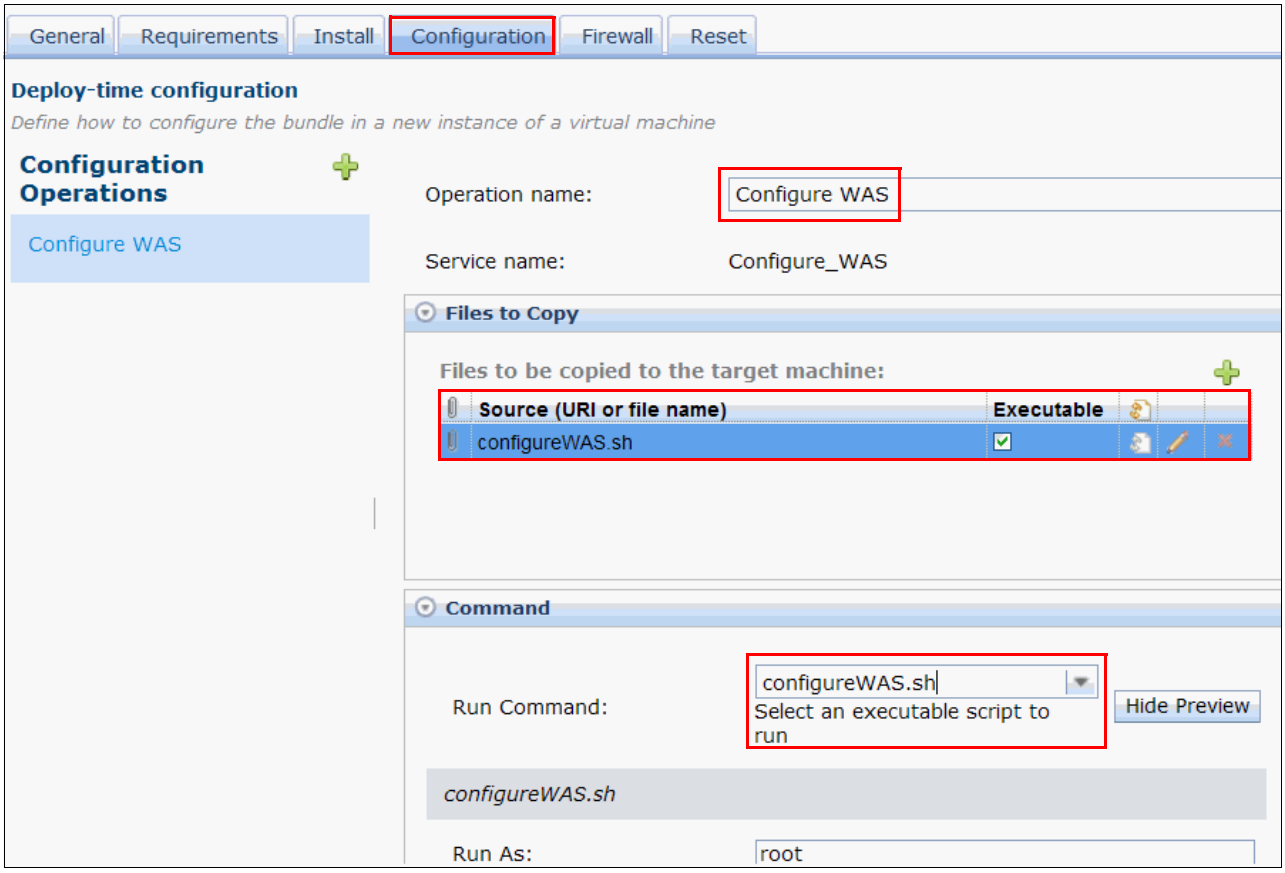


Figure 5-50 Provide configuration parameters for the software bundle

9. Under the Arguments section, enter the arguments that are necessary to run the configuration script (such as the number of servers and product installation root). Additionally, under the Dependencies section, add any operating system services on which the bundle depends (such as sendmail or syslog). Figure 5-51 shows the arguments section for the configuration script for the software bundle.

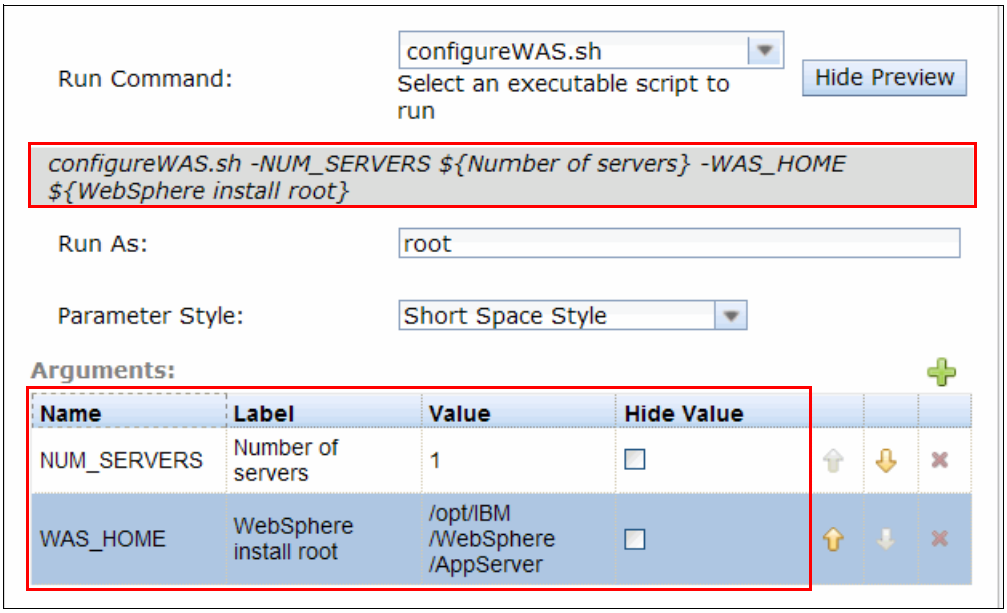


Figure 5-51 Arguments in the configuration settings for the software bundle

Important: Configuration scripts always are run as root; the value of Run As is ignored.

10. Browse to the Firewall tab. Click **Add** to specify ports that must be opened for the bundle software to work correctly.

Important: Opening ports is critical in environments such as the IBM public cloud (SmartCloud Enterprise), where all ports are closed by default.

At deployment time, a special Image Construction and Composition Tool service (part of all images that are created with Image Construction and Composition Tool) open the bundle ports. Specifying what ports to open can be done in the following ways:

- By entering a constant corresponding to the port number
- By referencing a bundle deployment parameter

Figure 5-52 displays the firewall settings for the bundle.

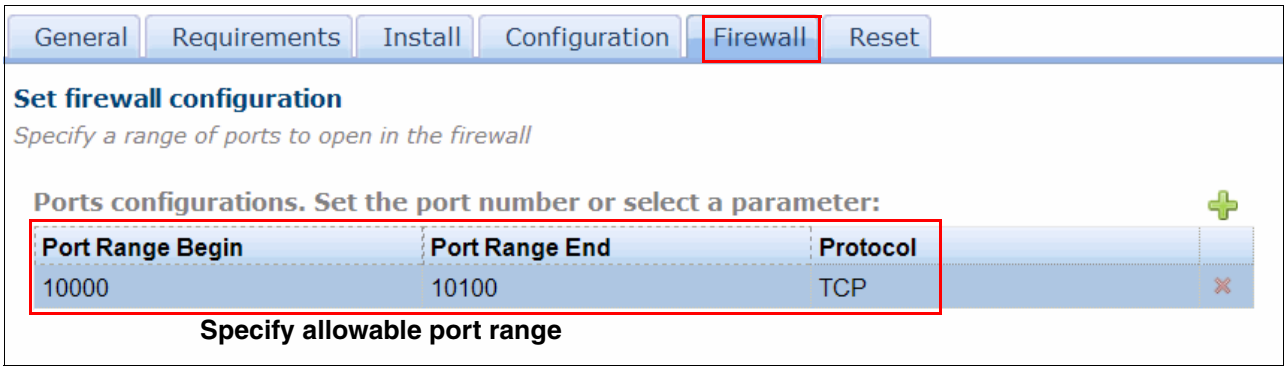


Figure 5-52 Firewall settings for the software bundle

11. Browse to the Reset tab. Click the respective **Add** icons for the Files to Copy and Commands sections to provide the scripts or commands to run as part of the reset operation. The objective is to provide a way to revert the image to a relevant state that was created during the bundle installation.

Important: Reset commands always are run as root; the value of Run As is ignored.

It is up to the bundle author to decide what to clean up, such as temporary files, log files, and security-sensitive information. The execution of the reset script is triggered directly before Image Construction and Composition Tool captures the image contents. Figure 5-53 on page 248 shows the scripts and commands that are configured for resetting the bundle.

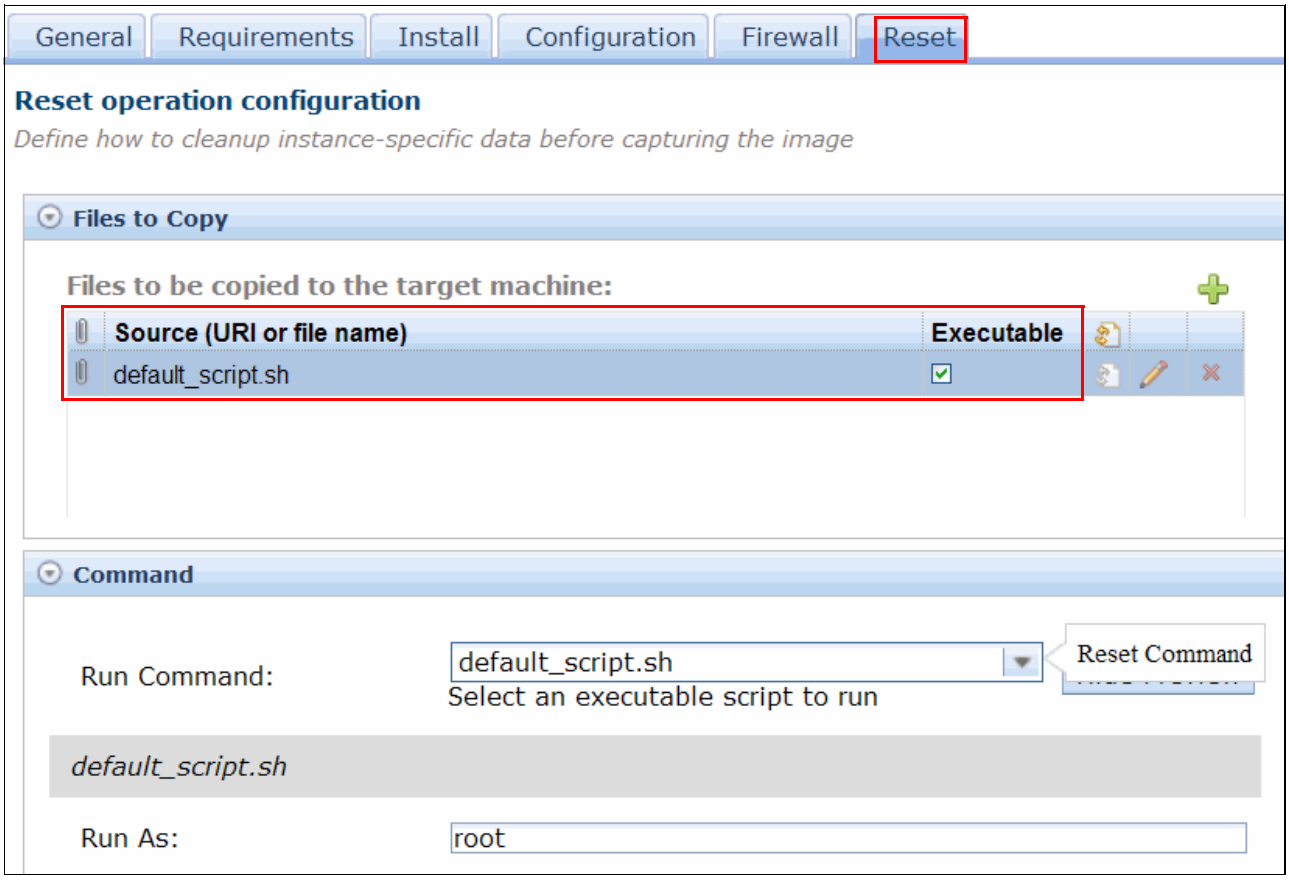


Figure 5-53 Reset script or command settings for the bundle

You completed the process of creating a software bundle. You can publish the bundle, in which case it cannot be edited further. Also, to create similar bundles, you can use the clone feature. This feature copies all of the settings from the current bundle and allows for further editioning, which facilitates customization. Figure 5-54 shows the panel for choosing the publish and clone options for the software bundle.

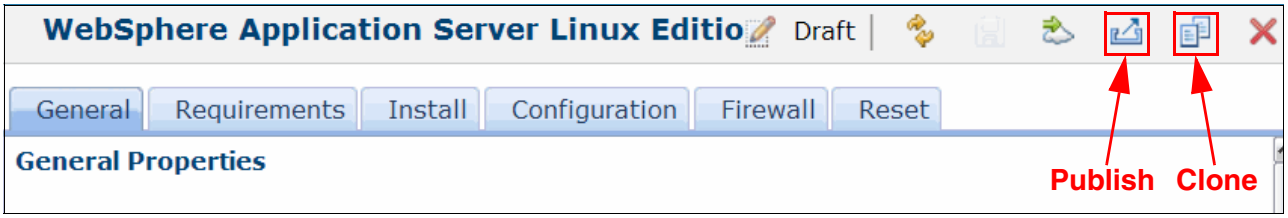


Figure 5-54 Edit panel settings for software bundle

Troubleshooting the bundle

Problems might occur in the following phases when the software bundles are configured:

- Bundle authoring
- Image construction (when the bundle is installed on a base image)
- Image deployment

The following points might help in analyzing issues during the installation or configuration software bundles process:

- Each bundle must have the installation, configuration, or reset operations defined. If the installation operation is chosen, it must include at least one product in the list of available products.
- When the bundle is edited in Image Construction and Composition Tool, be aware of the validation status at the bottom panel in the bundle editor's GUI to monitor success, warnings, or error messages.

  Validation warnings do not prevent a bundle from being used. However, it might identify issues that cause the software not to install or work correctly.
- An Image synchronization failure might indicate a problem in running the bundle installation script. Review for configuration settings or missing software (as detected by the bundle installation script) which might return a non-zero code to Image Construction and Composition Tool.
- Review Image Construction and Composition Tool's trace log for further analysis.
- For image deployment problems, check whether the bundle configuration operations failed. Analyze the scripts' standard output and error logs, activation engine logs, deployment parameter values, and configuration script's contents.

For more information about known problems in Image Construction and Composition Tool with software bundles and suggested solutions, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/ICON/topics/tricn_troubleshooting.html

5.3.5 Using command-line interface for Image Construction and Composition Tool

To perform administrative functions for Image Construction and Composition Tool, you can download and run the command-line interface (CLI) on a local machine. The CLI can be downloaded from the Welcome page of Image Construction and Composition Tool.

For more information about configuring CLI or performing administrative tasks for Image Construction and Composition Tool, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/ICON/topics/cicn_cli_using.html

5.3.6 Advanced configuration with Image Construction and Composition Tool

The previous sections in this chapter described the various options available in Image Construction and Composition Tool for customizing virtual images. The focus mainly was on PureApplication System as the cloud provider.

For more information about the topics in the following list, see the IBM Redbooks publication *Creating Smart Virtual Appliances with the IBM Image Construction and Composition Tool*, SG24-8042:

- Configuring Image Construction and Composition Tool with various other cloud providers such as KVM, PowerVM, or ESX
- Setting up virtual appliance build environments
- Constructing virtual appliances
- Customizing deployment options

5.4 Administering PureApplication System by using CLI

Similar to the GUI or PureApplication System console, you can use the CLI to administer PureApplication System. The CLI provides support to manage the system in a scripted, non-graphical environment. CLI includes the following features:

- Minimal functions are available without a network connection to the system
- Runs Python scripts and commands in a Jython scripting environment
- Can be started in interactive, command, or batch modes
- Built by using REST APIs
- Provides Workload console and System console support

CLI download and setup

Complete the following steps to download the CLU to your local system from the PureApplication System console:

1. Download the CLI from the user interface Welcome page. From the Welcome page, click **Download tooling** and then click **Download command-line tool**.

> **Important:** The CLI code from previous versions of IBM WebSphere CloudBurst® Appliance does not work with PureApplication System.

Figure 5-55 shows the panel for downloading the CLI from the PureApplication System Workload console section.

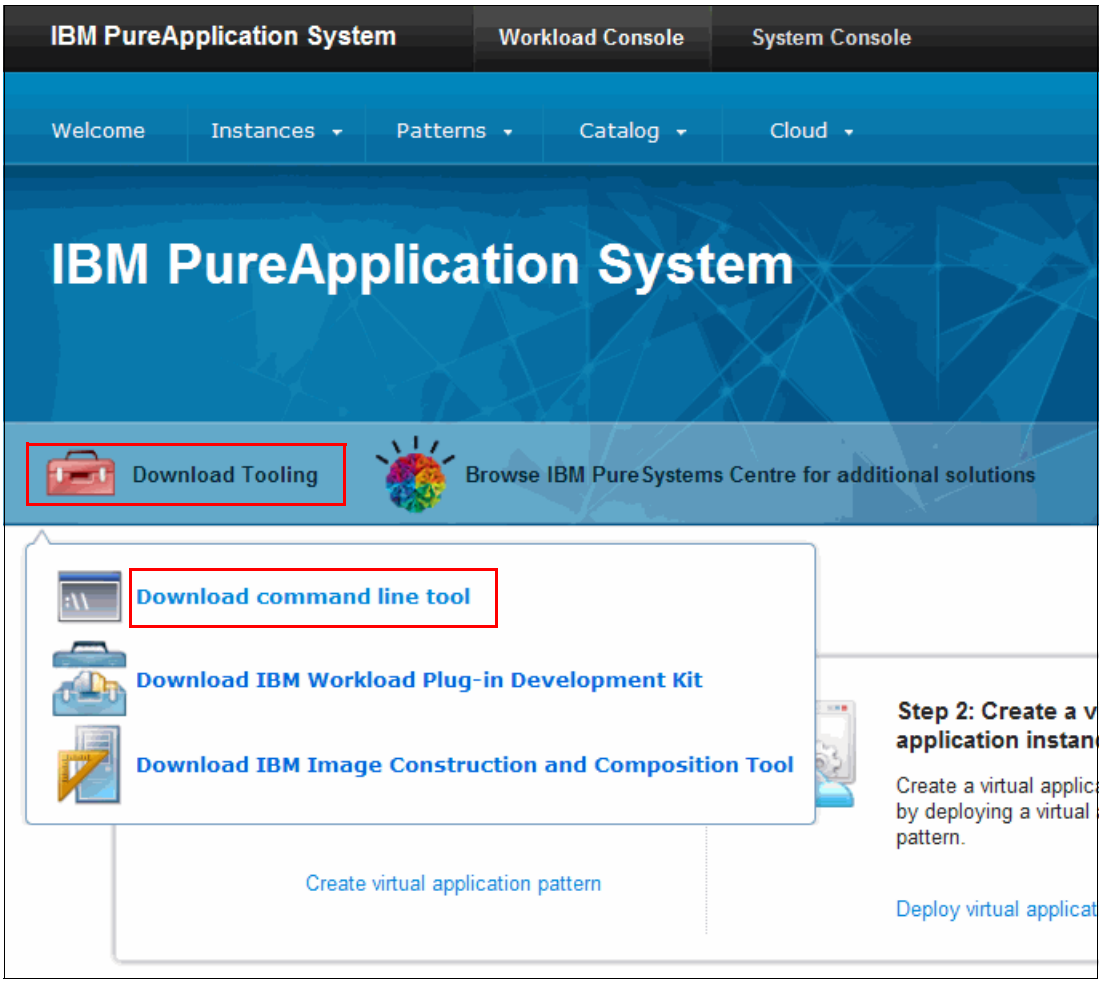


Figure 5-55 Downloading CLI

2. Save the `.zip` file to your local hard disk drive.
3. Extract the contents of the `.zip` file to a directory on your hard disk. When extracted, a single directory that is called `pure.cli` is created.
4. Ensure that Java Runtime Environment V6 or higher is installed on the system and the `JAVA_HOME` or the `PATH` environment variable points to the Java Runtime Environment location.

Optional: If you are using Windows Server 2003 or 2008, under the `pure.cli` directory, create a registry file in the `lib/<version>` with the following line:

```
python.os=nt
```

To manage PureApplication System with the CLI, you can download the CLI to any machine and then point to where PureApplication System is running.

Important: Run commands from `<CLI_HOME>/pure.cli/bin`, where `CLI_HOME` is the installation (extract) directory of CLI.

The CLI communicates with the PureApplication System over a hypertext transfer protocol secure (HTTPS) session. The CLI does not cache updates and has only minimal caching for reads. If the machine on which the CLI is running loses connectivity to port 443 of the system, you can perform only rudimentary operations.

Starting the CLI

The CLI can be started in following modes:

- Interactive mode: Supports command history and subset of Emacs commands with control-key bindings only (from JLine)
- Command mode: Run a single command
- Batch mode: Run a set of commands by using Python or Jython scripts

Example 5-1 shows the various modes of starting the CLI.

Example 5-1 Modes of operation in CLI

```
Interactive mode
   pure –h <host> -u <user> -p <password>

Command mode
   pure –h <host> -u <user> -p <password> -c “<command>”

Batch mode
   pure –h <host> -u <user> -p <password> -f “<filepath>” <script_args>
```

The CLI commands are written in Jython syntax with the dot operator. The dot operator denotes the member of (for example, parent-child) relationship between the two operands that are separated by the dot(.) symbol. The data that is passed into or displayed from the commands often are in the python data (dictionary) format.

Example 5-2 shows a sample code snippet that uses the CLI syntax and data formats.

Example 5-2 CLI syntax and data format

```
Command syntax
   deployer.<object_type>[.<command>(<arguments>)]
   deployer.<object_type>[.<command>[(<data_or_file>)]]

Data format
   [
     {
      “name1”:”value1”,
      “name2”:”value2”,
      …
     },
     {
      “name1”:”value3”,
      …
     }
   ]
```

Using the CLI

The CLI can be used to perform various operations that are possible by using the Workload and System console from the GUI mode. Example 5-3 shows how to obtain help with the list of commands that are available for the respective sections.

Example 5-3 Help commands for CLI

```
Workload Console commands
>>> help(deployer)

System Console commands
>>> help(admin)
```

Some of the examples for system commands are listed in Example 5-4.

Example 5-4 Sample system commands

```
Basic commands:
>>> admin.leds
>>> admin.events
>>> admin.systemlog

Download all log files:
>>> admin.diagnostics.get('c://temp/diagnostics.zip')

Display the last 20 lines of the error log:
>>> admin.errors.tail(20)

Display the last 1000 lines of the trace file and print to a file (in command
mode):
>>> pure –h <host> -u <user> -p <password> -c "admin.trace.tail(1000)" >
c://temp/trace1000.txt
```

The complete set of operations that are supported by CLI on the system console is listed in the product information center at this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/clihelp/cliForSystemConsole.html

For many of the commands, the CLI uses the terms *resources*, *resource collection*, and *methods*. A PureApplication System manages different types of resources; for example, hypervisors, patterns, virtual images, and virtual system instances. Within the CLI, Jython objects are used to represent the following resources and collections of these resources:

- Resources: Represent an individual object and have properties and methods.
- Resource collections: Represent the entire collection of one type of resource; have only methods but do not have properties.
- Methods: Represent the operations that can be performed on a particular resource.

For more information about the commands that are applicable for different resources, resource collections, and the corresponding methods, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/ccc_clires.html

Exporting and importing patterns

Exports and imports of virtual system patterns and database as a service (DBaaS) patterns can be done only by using CLI. Use the standard CLI parameters to specify the host name, user ID, and password to access the virtual system pattern. The parameters also are used to access the location of the script that is used for the import or export task. Specify the file name and virtual system pattern, as shown in Example 5-5.

Example 5-5 Import and export virtual system pattern tasks by using CLI

```
Virtual System Patterns export (interactive mode):
>>> deployer.patterns.list({"name":"My Pattern"})[0].toPython("c://temp/My
Pattern.py")

Virtual Systems Pattern export (batch mode):
> pure –h <host> -u <user> -p <password> -f .../samples/patternToPython.py -p "My
Pattern" -f "c://temp/My Pattern.py“

Virtual System Pattern import (batch mode only):
> pure –h <host> -u <user> -p <password> -f "c://temp/My Pattern.py“

DBaaS pattern export (interactive mode):
>>> deployer.applications.list({"app_name":"My DBaaS"})[0].download("c://temp/My
DBaaS.zip")

DBaaS pattern import (interactive mode):
>>> deployer.applications.create("c://temp/My DBaaS.zip")
```

CLI resource object reference

Anything that can be managed in PureApplication System is a resource object on the CLI. The PureApplication System CLI manages different types of resources; for example, patterns, virtual images, and virtual system instances.

A user without the Workload resources administration (full permission) role cannot set or modify the console resource field that is named `Access granted to`. This field controls items such as pattern resources and catalog resources. A user can modify the field by way of the command line, by the workload console with valid permissions, or by another user with such permissions.

The reference for the resource objects that are available in CLI can be found from the product information center, which is found at this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/ccr_availres.html

Problem determination by using CLI

You can perform the PureApplication System problem determination tasks by using CLI. The most commonly used objects for troubleshooting are diagnostics, traces, and errors.

The CLI provides various methods for each of these objects that can be used for configuring the PureApplication System to capture information for analysis. The CLI commands reference for debugging is provided in the product information center, which can be found at this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/tsr_cliprobd.html

For more information about various aspects of troubleshooting the CLI based on the operations that are performed in PureApplication System, see Chapter 8, “Troubleshooting PureApplication System Environment” on page 329.

5.5 Working with script packages

Script packages can be added to virtual system pattern topologies to customize the behavior of the parts. Virtual system patterns are used to define cells and they can include script packages to further define the behavior.

Script packages are simple containers that contain all the required artifacts to run a script. The script package is a directory that is compressed into a single file that is uploaded to the catalog and then associated with virtual system patterns. The code that is included in the script package can be as simple as a `.war` file or as complex as a complete product. The content of a script package is not defined by the product. Instead, the included scripts in the package define the required content for that package.

During deployment, script packages are transferred to the target virtual machines at a location you specify in the configuration. After they transfer, they are extracted in that same location. When the virtual machines successfully start and are federated (if applicable), script packages are extracted. The scripts are run by using the supplied command line.

Important: Files that are associated with script packages are written to the file system as the root user. If a different user requires access to these files, ensure that the correct user properties are set on the guest operating system.

The following scripts are included with the PureApplication System:

- Add IBM HTTP Server node
- WebSphere Application Server Samples

When multiple script packages are included with a virtual system pattern, by default, the scripts are run in the same order they were added to that pattern. However, you can change the order in which script parts are run in the pattern. For more information, see , “Ordering” on page 206.

A sample script package to create a server is shown in Example 5-6.

Example 5-6 Script package to create application server

```
[
     {
            "name": "Server creation",
            "version": "1.0.0",
            "description": "This script package creates a server on each node
      within the cell",
            "command": "${WAS_PROFILE_ROOT}/bin/wsadmin.sh",
            "log": "${WAS_PROFILE_ROOT}/logs/wsadmin.traceout",
            "location": "/opt/tmp/createserver",
            "timeout": "0",
           "commandargs": "-lang jython -f /opt/tmp/createserver/create_server.jy
      $SERVER_NAME",
            "type": "APPLICATION",
            "keys":
            [
               {
               "scriptkey": "SERVER_NAME",
             "scriptvalue": "",
              }
            ]
     }
]
```

This script creates an application server on all of the custom nodes in a virtual system pattern or on the node part for which it is included. The script package is intended to be used on a deployment manager or stand-alone server part in a virtual system pattern.

The name of the application server is specified by the user by using the script variable `SERVER_NAME`. The script package starts a Jython script to create the application server by using WebSphere scripting APIs, as shown in Example 5-7.

Example 5-7 Jython script to create the application server

```
serverName = sys.argv[0]

managedNodeStr = AdminTask.listManagedNodes()

if len(managedNodeStr) != 0:
   managedNodes = managedNodeStr.split("\n")
   i=1
   for managedNode in managedNodes:
      thisServer = serverName + "_" + str(i)
      AdminServerManagement.createApplicationServer(managedNode, thisServer,
'default')
      i=i+1


else:
   node = AdminControl.getNode()
   AdminServerManagement.createApplicationServer(node, serverName, 'default')

AdminConfig.save()
```

The following common configuration tasks are performed with script packages:

- Adding script packages
- Associating a script package
- Cloning script packages
- Deleting script packages
- Configuring script packages by using JSON object
- Script package environment variables and properties

For more information about each of these tasks, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/pct_scipacui.html

5.6 Using Advanced Middleware Configuration

By using Advanced Middleware Configuration (AMC), you can migrate your existing WebSphere applications to a new virtual system pattern in the PureApplication System. This process commonly is referred to as *onboarding applications on the cloud*.

5.6.1 Overview

If your application already has an existing automation in place, you can create the following options:

- Virtual system pattern that specifies the required topology (number of WebSphere nodes and IBM HTTP Server).
- Script package to run the automation and attach to the virtual system pattern.

AMC plays an important role when there is no existing or partial automation for your environment. It is delivered as a workload that can be used to automate the deployment of existing applications on a cloud. AMC automates the following important features:

- Installation of the product
- Topology configuration
- Application deployment on the configured topology

AMC features the following major components:

- Framework server: A single central server or system from which products are managed.
- Target system: One or more physical or virtual systems or cloud instances that are running the middleware products, such as IBM WebSphere Application Server. The target system can be used as source and destination. In source mode, the configuration on the target system is read and created or updated on the framework server. In destination mode, the configuration on the framework server is written (created or updated) to the target system.

Figure 5-56 on page 258 shows the components of AMC.

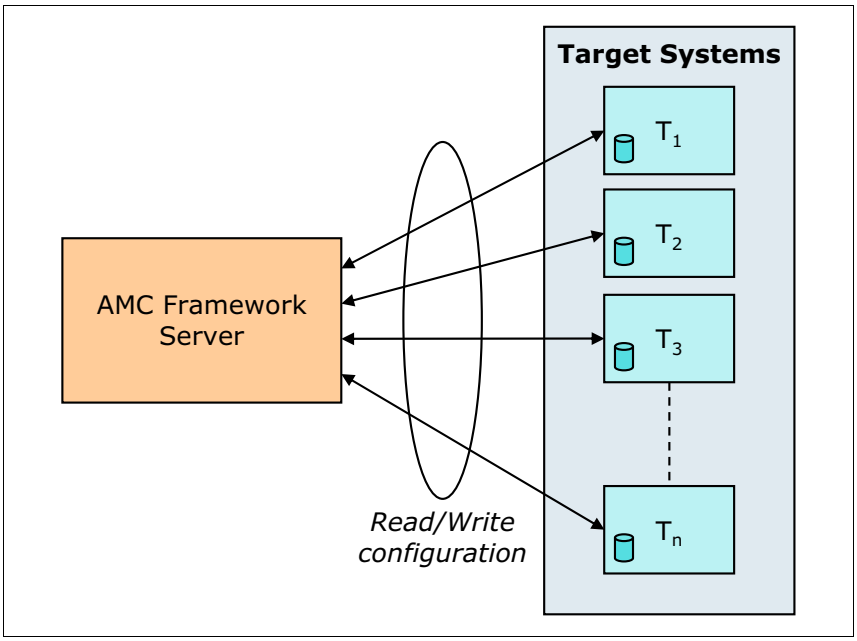


Figure 5-56 Components of AMC

For more information about the main processes that are involved in the use of AMC and the supported actions, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/RAF/com.ibm.raf.install.doc/topics/c_getting_started_tech_overview.html

5.6.2 AMC User interfaces for PureApplications System

The AMC for PureApplication Systems provides the following interfaces:

- Eclipse client: A graphical user interface that is based on the Eclipse integrated development environment. The Eclipse client supports the following perspectives:
  - Configuration: By using the Configuration perspective, you can view and edit configuration environments graphically. You also can compare the configuration environment data on the framework server with the configuration data on the corresponding target system. You also can run actions to synchronize or update the data in either location.
  - Automation: By using the Automation perspective, you can perform operational tasks to manage products. The tasks are defined in the framework server as automation plans, which perform actions on the specified target systems.

  For more information about configuring and the use of Eclipse client, see this website:

  http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/RAF/com.ibm.raf.common.doc/topics/c_getting_started_top.html
- WebClient: A browser-based graphical interface that is used by administrators and operators to work with automations.
- Command line: A CLI that you use to run actions, the Environment Generation wizard, integrate automations with a managed product, and manage libraries of actions that are used in automations.

In addition, you can call the Environment Generation wizard from any of the interfaces. The wizard is used for working with the WebSphere product family. You also can create configuration environments that are used to create deployments and read existing deployments from the configuration data on a target system.

5.6.3 On-boarding applications with AMC

If your existing application environment cannot be run as virtual applications and has no existing automation in place, AMC fits well in this scenario.

The following concepts regarding on-boarding applications to the PureApplication System as virtual system patterns by using AMC are pertinent:

- Deploying AMC framework server
- Creating virtual system pattern
- Setting up the application
- Capturing instance into AMC
- Deploying new application instances

These concepts are described next.

Deploying AMC framework server

Complete the following steps to deploy AMC Framework server as a virtual system pattern:

1. In the Workload Console, click **Patterns** → **Virtual Systems**.
2. From the Virtual System Patterns list on the left, click the **Advanced Middleware Configuration** pattern.
3. Click **Deploy** and the deployment pane is displayed, as shown in Figure 5-57.

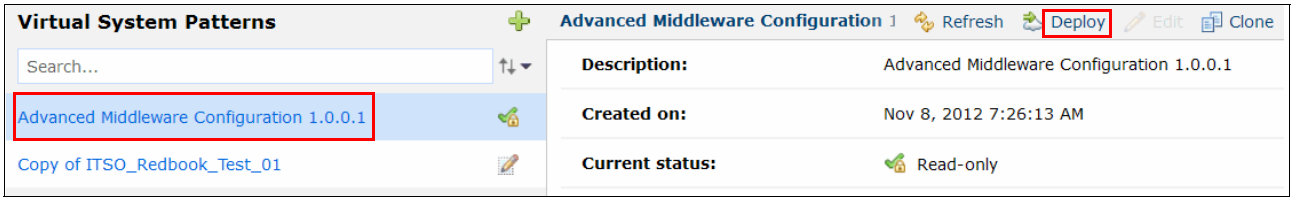


Figure 5-57 Deploying AMC

4. Enter a name for the new instance, such as `AMCFrameworkServer`. Enter the other necessary details in the deployment panel and then click **OK**. Figure 5-58 on page 260 shows the input panel for providing deployment properties.

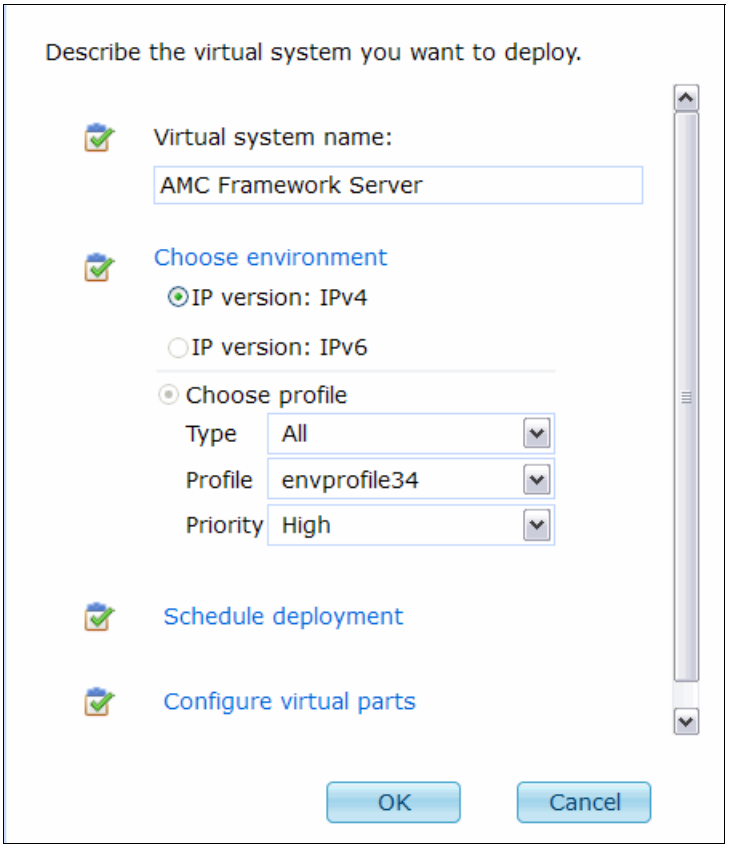


Figure 5-58 Deployment properties for AMC

5. After the pattern is deployed, the console automatically loads the new instance. When the Current status field shows Virtual system is ready, expand Virtual machines.

   Figure 5-59 shows the deployment status.

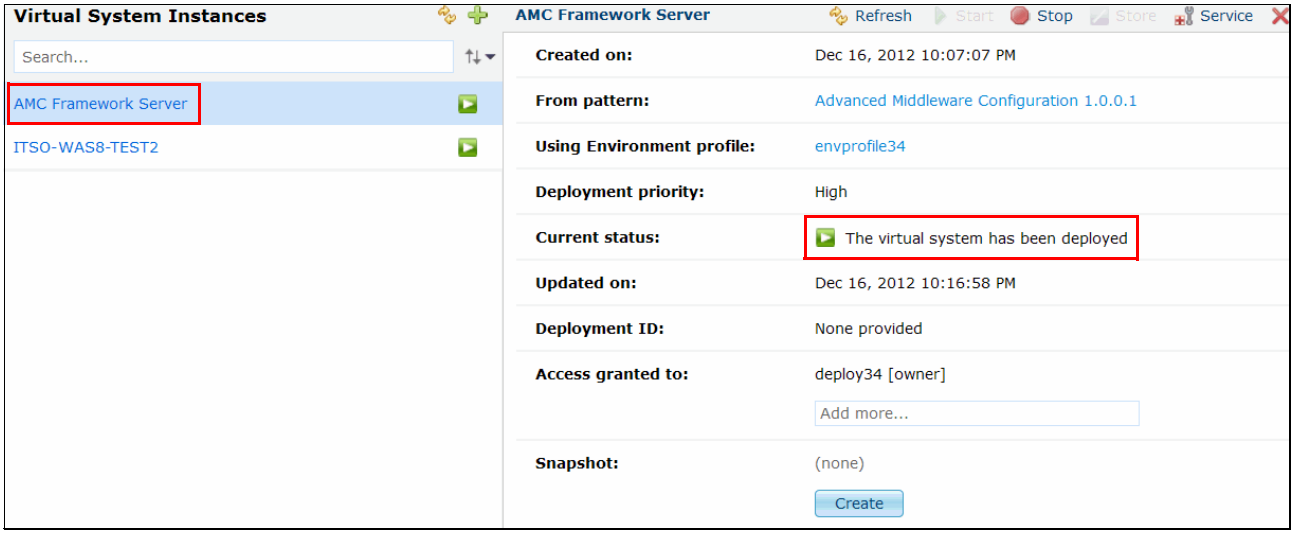


Figure 5-59 AMC successfully deployed as virtual system pattern

6. Expand the first machine that is displayed. Scroll to the bottom of the display for the machine, find the Consoles section, and click **AMC**. A browser window opens with the web client for Advanced Middleware Configuration that is running on the framework server.

   Figure 5-60 shows the link on the virtual system instance for the AMC Framework Server GUI.

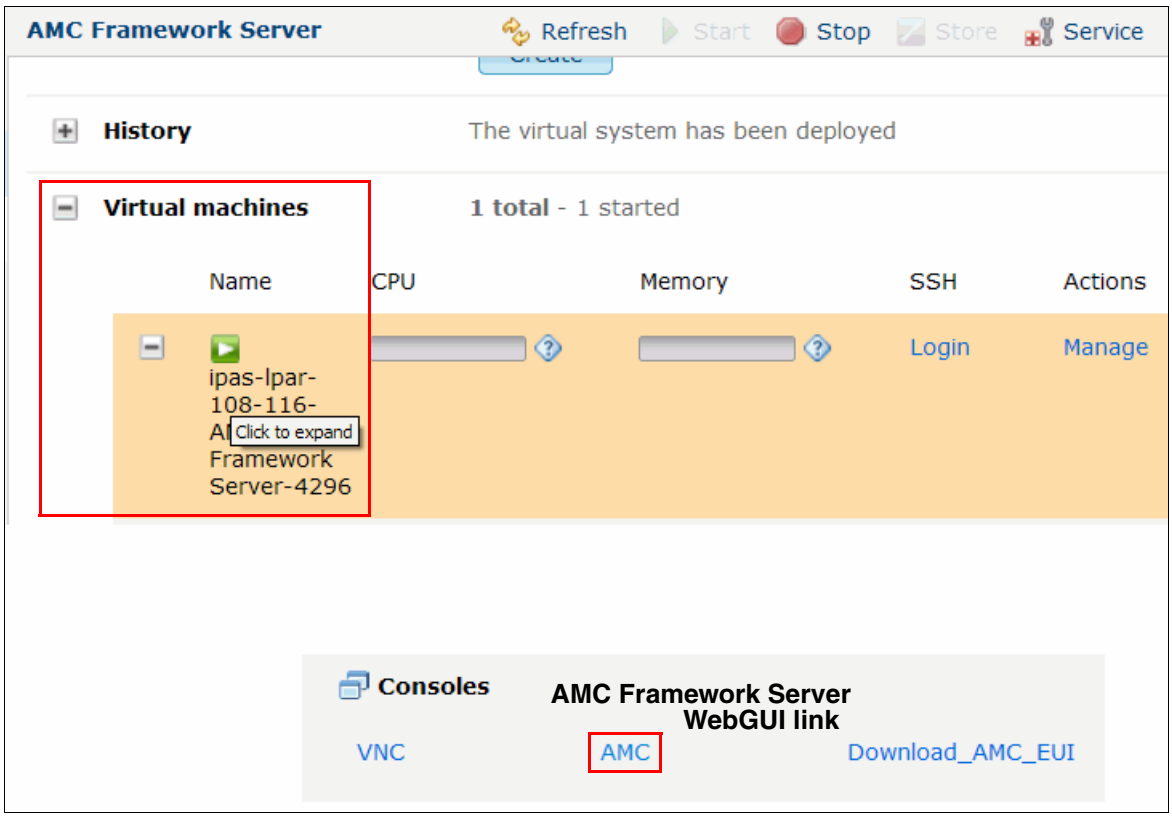


Figure 5-60 AMC virtual system instance

7. Log in to the web client. The following default credentials are available:
   - User name: root
   - Password: root
8. Set up a unique user ID and password for each model virtual system pattern that you plan to create. Complete the following steps to create users for virtual system patterns:
   a. Log on to the Advanced Middleware Configuration web interface.
   b. Select **Administration** → **Users**.
   c. Enter the name, password, and email address for the user. Click **Save**.
   d. Select the user from the list of users. Click **Change Groups**.
   e. Select **Build Engineer** from the list on the left, then click **Add**. Click **Save**.

Creating a virtual system pattern

Complete the following steps to import the topology for your environment as a virtual system pattern in PureApplication System:

1. In the web interface of AMC, click the **EnvGen** tab and then click **Read an Existing Cell Configuration**. The system displays the AMC EnvGen panel that is used to read existing cell configuration.
2. Complete the wizard fields. For more information about the wizard, see the Field reference for an existing WebSphere cell section at this website:

   http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/RAF/com.ibm.raf.common.doc/topics/c_checklist_existing_cell.html

   You must answer `yes` to the question Create a Pattern from this Environment?, and complete the other questions for VSP generation. After the wizard completes, a virtual system pattern is created in PureApplication System with the name that you specified in the wizard.
3. In the Workload Console, click **Patterns** → **Virtual Systems** and then click the name of the virtual system pattern that the Environment Generation wizard created.
4. Click **Deploy**.
5. In the deployment pane, assign a name for the model server instance that is created and then click **OK**. This instance is the instance where you install your application. The integration script package contains all the information that you entered in the Environment Generation wizard for the virtual system pattern questions section. When the deployment is complete, the console displays the new instance.

Installing and configuring the application

After the virtual system pattern is deployed, a WebSphere Application Server cell is running in the instance. You now install and configure the application in the cell by using the administrative console or by using any existing `wsadmin` scripts.

Use this instance as a template that can be duplicated each time that you deploy the server pattern in the future.

Capturing instance into AMC

Capturing the instance places configuration information into the AMC framework server. After it is captured, AMC can work with the integration script package to create duplicates of your model server instance that you can deploy repeatedly to the cloud.

Complete the following steps to capture the instance into AMC:

1. Customize the Import Automation Plan.

   In Advanced Middleware Configuration, open the Projects list and click the plan name. You can use the project as-is. However, you can save time by excluding any steps that are not used. The plan is used when the Import script package runs.

2. Run the Import script package:
   a. In the instance where you installed the application, expand the Deployment Manager part (or Server part if it is a stand-alone server).
   b. Scroll to the Script Packages section.
   c. Under AMC Import Script Package, click **Execute now**.
   d. Click **OK** in the login dialog. The login is not used.
   e. The job starts. To view its progress, switch to the Advanced Middleware Configuration web client, click **Jobs**, and then click your job in the list. The job has the same name as your Automation Plan plus a build number suffix.

Deploying new application instances

AMC works with the integration script package to create instances of your model server pattern that automatically include your application and any other specific configuration information.

This process works because the integration script package includes the name of an AMC environment. When that environment exists on the framework server, the Integration script package uses configuration information from that environment to update the new instance.

Complete the following steps to deploy application instances:

1. Customize the Integration Automation Plan.

   In Advanced Middleware Configuration, open the Projects list and click the plan name. You can use the script as-is. However, if you changed the Import Automation Plan, make the same changes in the Integration Automation Plan.

   Important: Excluding or removing unnecessary steps helps optimize the performance of the script package during deployment.

2. Deploy a new instance.

   Open the model server pattern and click **Deploy**. For each new virtual system that is deployed from the pattern, the following actions are performed automatically:
   a. Update the Advanced Middleware Configuration environment with the new host names for this virtual system.
   b. Push the WebSphere configuration data at all relevant scopes to the new cell.
   c. Deploy all applications for all relevant clusters to the new cell.
   d. Start all clusters or servers and applications in the cell.
   e. Generate and propagate the web server plug-in.
   f. Add anything custom into the Integration Automation Plan.

You successfully on-boarded the user application to the PureApplication System and enabled deployments of new instances as patterns.

5.7 Add-ons and advanced options

The Catalog add-ons and advanced configuration options that are available in PureApplication Systems provide extended customization capabilities for virtual system patterns.

5.7.1 Catalog add-ons to virtual system patterns

You can use the PureApplication System console to manage add-ons in the catalog and then add them to deployable patterns.

Important: Create catalog content permissions or the Workload resources administration with full permissions is needed to administer the add-ons.

The Figure 5-61 shows the Add-ons panel in the PureApplication System console.

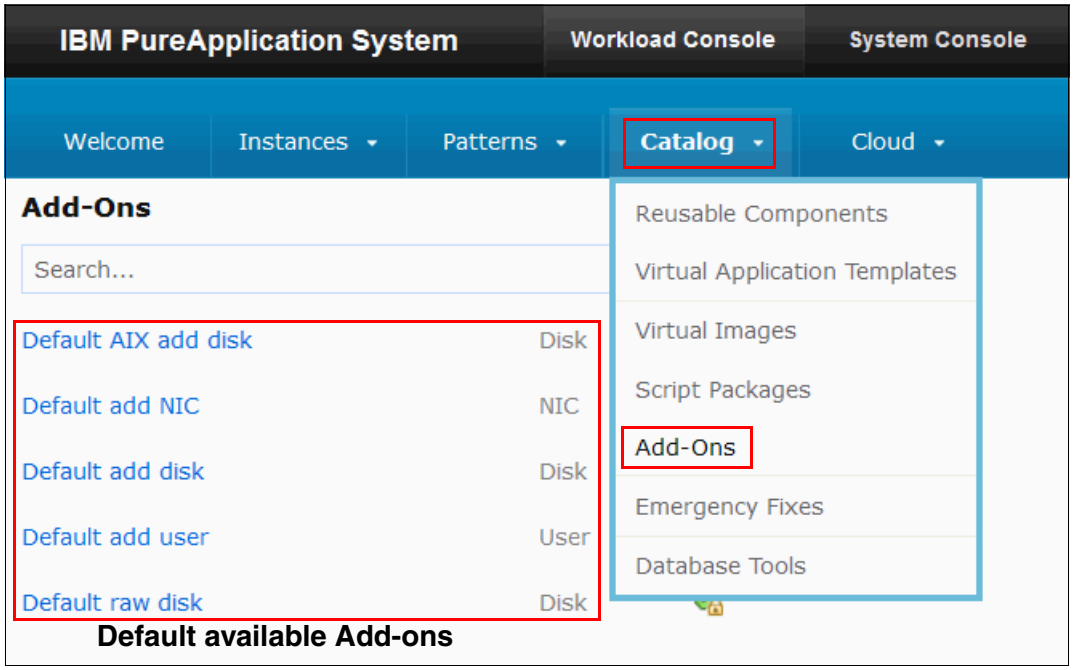


Figure 5-61 Catalog add-ons panel

The catalog add-ons that are available for creating, modifying, or deleting are disk, network interface controller (NIC), and user.

The following configuration options are available for add-ons:

- Adding add-ons to the catalog: You can use the console to create an add-on. An example of adding a disk add-on by using the PureApplication System console is described in, “Add-Ons” on page 205.
- Cloning an add-on: You can use the console to create an add-on that is based on an existing add-on. Default add-ons are provided, which can be added as they are or cloned and edited. The cloned add-on can be modified based on your specifications.
- Editing add-ons: You can edit any add-on that is not read-only. You can modify an add-on to suit the changing needs of your environment.

- Making add-ons read-only: Draft or read-only add-ons can be deployed for testing or production, but making an add-on read-only prevents further edits. Making add-ons read-only provides consistent reuse in the cloud.
- Deleting add-ons: You can use the console to delete an add-on and remove the association with a pattern.

For more information about managing the catalog add-ons with virtual system patterns, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/aot_addonsui.html

5.7.2 Advanced options

By using the PureApplication System Virtual System Pattern editor, you can define the advanced options for virtual system patterns that define your cell. The advanced options include configuration settings for patterns such as messaging, session management, and security.

For example, to further define single server virtual system patterns, the following advanced options are available:

- Enable session persistence
- Global security

The types of advanced options that are available can be classified based on the following type of virtual system pattern:

- WebSphere Application Server patterns
- Cluster patterns
- Intelligent Management Pack patterns

For more information about the steps for configuring the advanced settings for the respective pattern, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/mpt_advaopti.html

6

Customizing Virtual Application Patterns

This chapter describes the various interfaces that are available for customizing the virtual application patterns. There are two ways to create virtual application patterns: extend the available default patterns (Web Application Patterns) or create a plug-in and associate it with one or more pattern types, which are described in this chapter.

This chapter also describes different polices that can be attached to patterns to change the behavior of a deployed application.

The following topics are covered in this chapter:

- Prerequisites
- Virtual Application Pattern elements
- Creating virtual application patterns with the virtual application builder interface
- Lifecycle management of virtual applications
- Plug-in environment setup and creation of custom patterns

© Copyright IBM Corp. 2013. All rights reserved.

6.1 Prerequisites

This section describes the requirements that are needed before virtual patterns are created or customized. For more information, see this website:

http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/gsr_pattreqs.html

6.1.1 Hardware requirements

To use virtual application patterns, you must have the IBM PureApplication System W1500 hardware with the following software requirements:

- IBM PureApplication System W1500 V1.0.
- Configured NTP Server: Virtual machines that are provisioned as a part of a virtual pattern need an NTP server to set the system time.

6.1.2 Software requirements

The following requirements are necessary to start building new or customize existing patterns:

- IBM Foundation Pattern V2.0.0.3 or later (includes shared services). This pattern type is included with the product.
- Depending on the patterns you are using, there could be other requirements. For more information, see the Pure Application Information Center at this website:

  http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/iwd/apt_managepatterntypes.html
- To deploy a virtual application to the cloud environment, you must have one of the following roles or permissions:
  - Workload resources administration role with permission to manage workload resources (full permission)
  - Workload resources administration role with permission to view all workload resources (Read-only)
  - Create patterns
  - Create catalog content
  - Create environment profiles
  - IBM License Metric Tool user

6.2 Virtual Application Pattern elements

A virtual application pattern consists of plug-ins that define components, links, policies, and configuration files. By using virtual application patterns, components can be connected to show dependencies and policies that are attached during deployment to configure specific behavior.

6.2.1 Components

Components represent middleware services and are used as part of your pattern creation. Policies can be applied to these components to change the behavior of your pattern. You can create your own component as a part of creating virtual application pattern or use components that are provided by PureApplication System.

The following components are available in PureApplication System:

- Application:
  - Other archive file (web application)
  - Other archive file (Java application)
  - Enterprise application component
  - Existing Web Service Provider Endpoint
  - Java application (IBM Java Runtime Version 7)
  - Policy Set
  - Web application component
- Database:
  - Database Studio web console
  - Database (DB2), such as IBM DB2
  - Existing database (DB2)
  - Existing database (Informix)
  - Existing database (Oracle)
  - Existing IMS database
- Messaging:
  - Existing Messaging Service (WebSphere MQ)
  - Topic
  - Queue
- OSGi:
  - Existing OSGi Bundle Repository (WebSphere Application Server)
  - OSGi Application (WebSphere Application Server)
- Transaction processing:
  - Existing CICS Transaction Gateway
  - Existing IMS Transaction Manager
- User Registry:
  - Existing User Registry (IBM Tivoli Directory Server)
  - Existing User Registry (Microsoft Active Directory)
  - User Registry (Tivoli Directory Server)

- Other Components:
  - Connect Out
  - Connect In
  - “Connect in” (Java application)
  - “Connect out” (Java application)
  - Monitored file (Java application)

6.2.2 Policy types customization

Policies can be applied to virtual applications to attain specific behavior. For example, based on business criticality, some applications must be highly scalable and available. Policies can be applied to attain this goal. Figure 6-1 shows a sample web application pattern that is deployed. Figure 6-2 on page 271 and Figure 6-3 on page 272 show policy options that can be applied to this pattern.

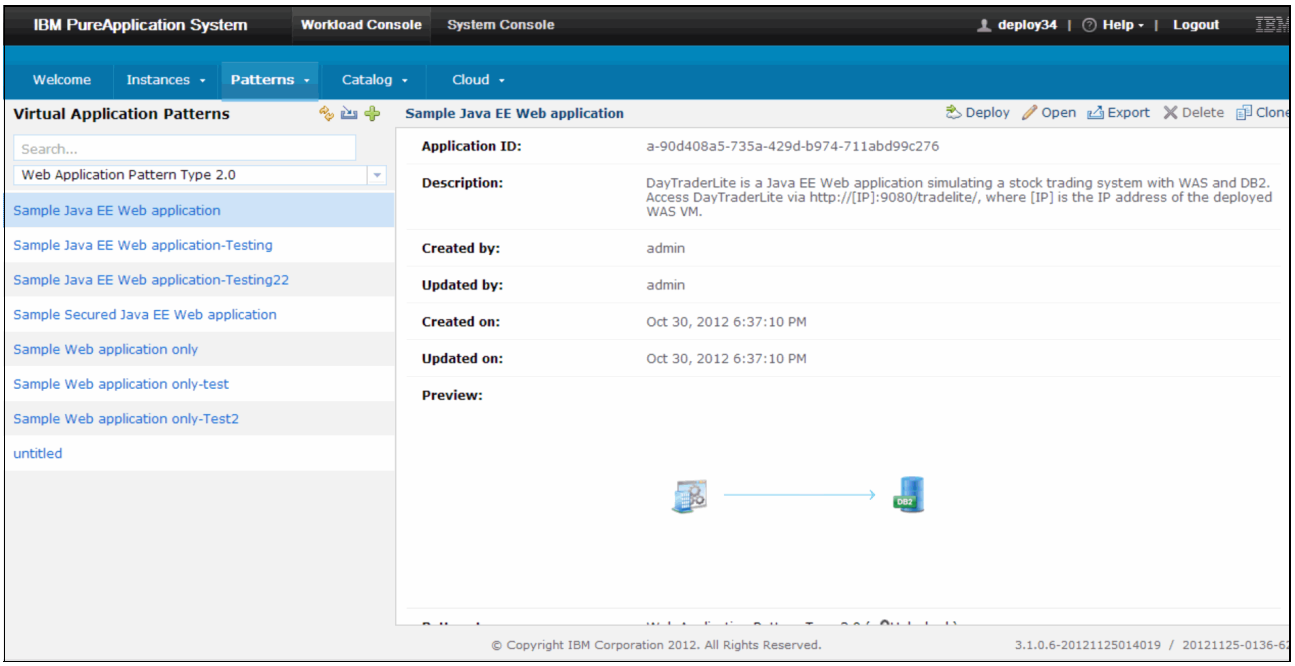


Figure 6-1 Sample Java EE web example

Scaling policy

A scaling policy is attached to a component to define the capability and conditions under which scaling activities are performed for your application.

The following attributes are available for a scaling policy:

- Session caching: Specifies whether to use session caching in your application.
- Scaling: There are several types of scaling, such as Static, CPU Based, Response Time Based, or Web to DB.
- Number of instances: Specifies the number of cluster members that are hosting the web application. The default value is 2, and an acceptable value range is 2 - 10. This attribute is required.
- Instance number, range of scaling in and out: Scaling range for instance members that host the web application. An acceptable value range is 1 - 50. This attribute is required.

- Minimum time (in seconds) to trigger an add or remove: Specifies the time duration condition to start the scaling activity. This attribute is required.
- Scaling in and out when CPU usage is out of threshold range (in percentage): Specifies the processor threshold condition that is necessary to start scaling activities. The platform is scaled after the processor usage exceeds the threshold range.
- Scaling in and out when web response time is out of the threshold range (in milliseconds): Specifies the web application response time condition necessary to start scaling activities. When the web application response time is out of this threshold range, your platform is scaled in or out. The acceptable values range 0 - 1,000 milliseconds.
- JDBC connections wait time is out of the threshold range (in milliseconds): Specifies the JDBC connection wait state that is necessary to start scaling activities. When the JDBC connections wait time is out of this threshold range, your platform is scaled in or out. The acceptable values range 0 - 10,000 milliseconds.
- JDBC connection pool usage is out of the threshold range (in percentage): Specifies JDBC connection pool usage necessary to start scaling activities. The platform is scaled after the threshold is exceeded.

Figure 6-2 shows the available scaling policy options.

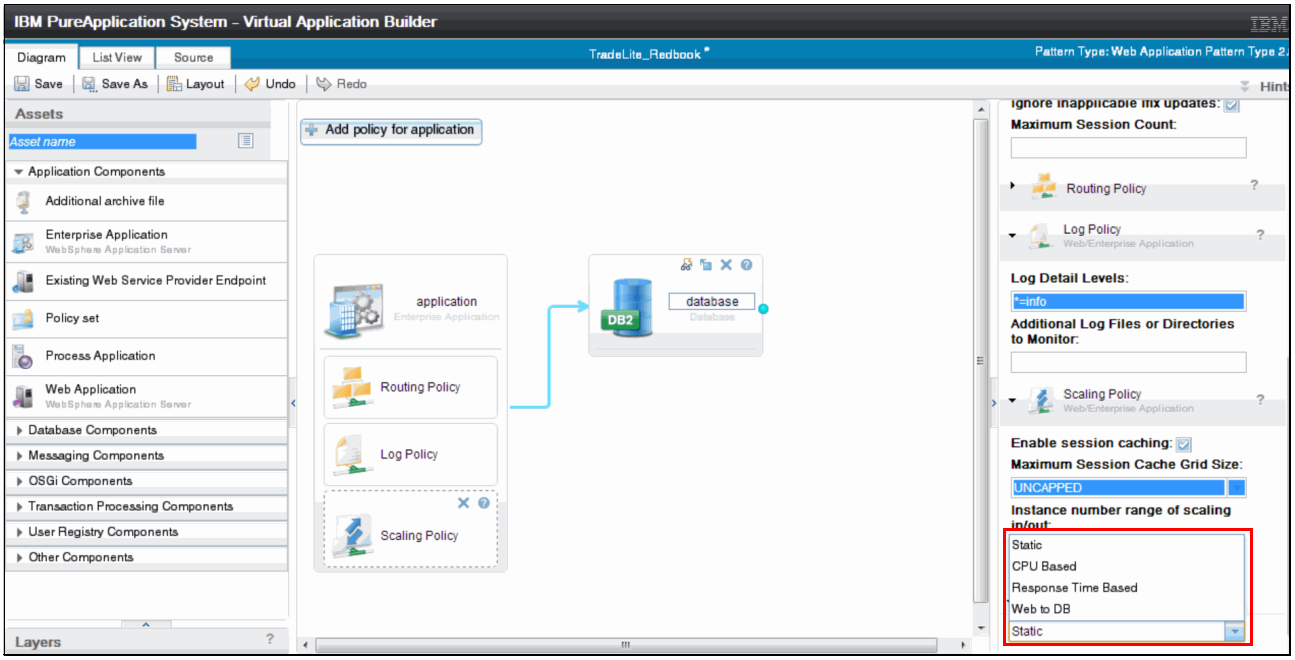


Figure 6-2 Scaling policy options

Figure 6-3 shows the options to enable session cache.

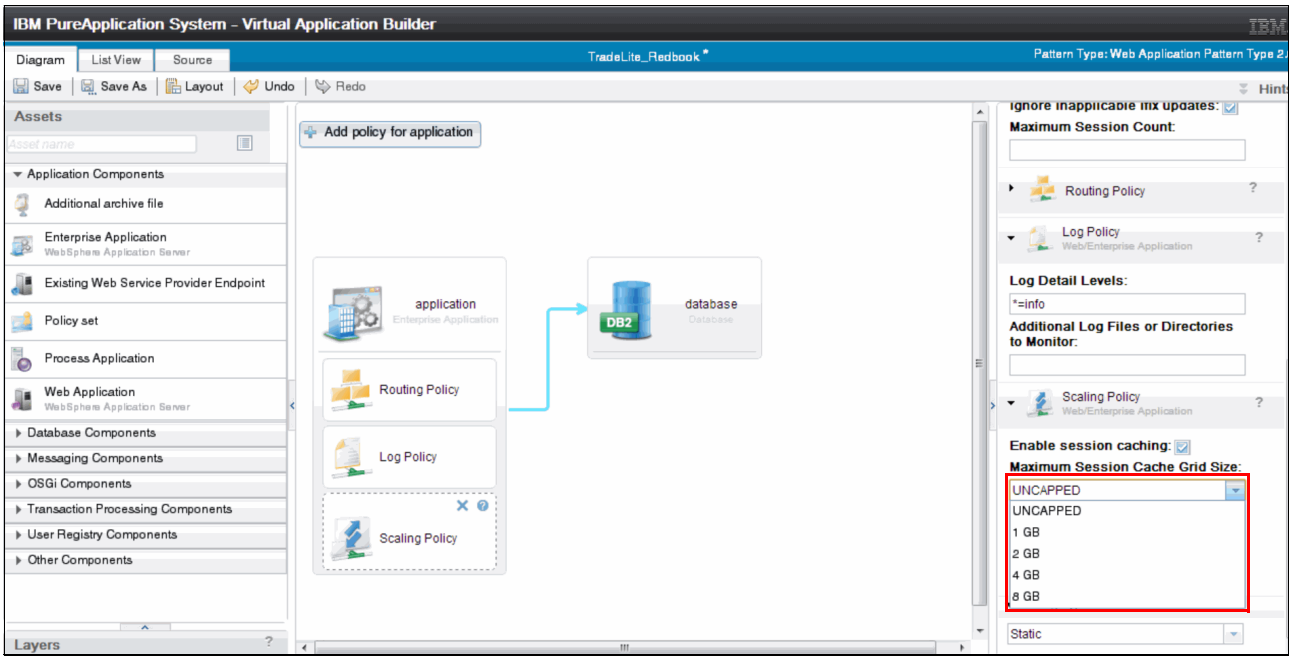


Figure 6-3 Options to enable session cache

Java virtual machine policy

A Java virtual machine policy controls the characteristics of the Java virtual machine.

A Java virtual machine policy features the following attributes:

- Minimum heap size: Specifies the minimum heap size of the Java virtual machine-specified size in megabytes (MB).
- Maximum heap size: Specifies the maximum heap size of the Java virtual machine-specified size in megabytes (MB).
- Enable debug: Specifies whether the Java virtual machine is in debug mode.
- Debug port: Specifies the port where the Java virtual machine listens for remote connections.
- Client (IP or IP/netmask): The IP address of the host that is used to debug.
- Client: Specifies an optional address of the debug client. This setting is used to restrict source access to the debug port. The value is an IP address, for example `1.2.3.4`, or IP/netmask `1.2.0.0/255.255.0.0`, which matches anything in the `1.2.` network.
- Enable verbose garbage collection: Specifies whether the Java virtual machine has garbage collection enabled.
- Generic JVM arguments: More Java virtual machine arguments can be added to customize how Java virtual machine runs.

- Bit level: Specifies bit level as 32 bit or 64 bit, as shown in Figure 6-4.

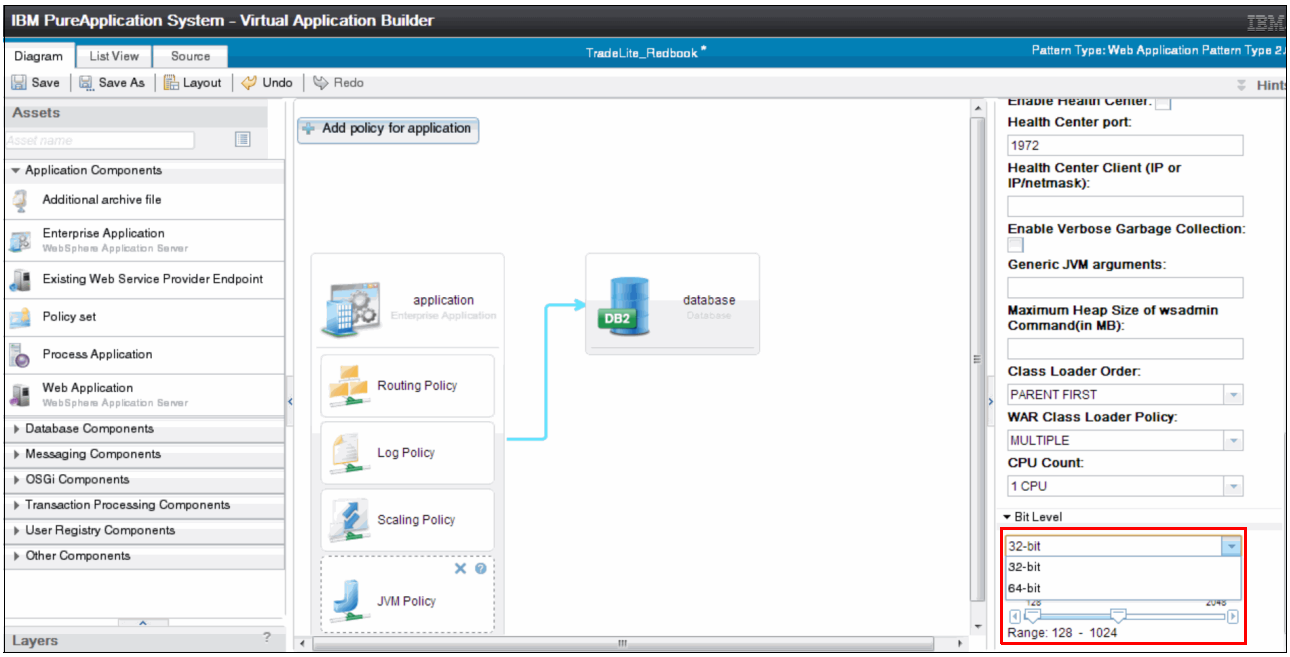


Figure 6-4 Java virtual machine options: Bit level customization

Routing policy

The routing policy can be applied to the application component parts of your virtual application pattern. By using the routing policy, you can customize the context root for your component. You also can specify the protocol for your component.

A routing policy includes the following attributes:

- Virtual host name: Name of the virtual host for the routing policy. This attribute is required.
- HTTP: Specifies support for HTTP schema with a routing policy.
- HTTPS: Specifies support for HTTPS schema with a routing policy.
- Customize prefix for context root: Used to specify custom context root.

Important: The routing policy is automatically applied to a web application when there is a proxy shared service that is running in the same cloud group into which it is deployed. Otherwise, the routing policy is not automatically added to the virtual application.

When elastic load balancing is enabled, the combination of the context root, context root prefix, and virtual host name must be unique to successfully deploy multiple virtual application instances from a virtual application pattern. If you do not manually add a routing policy to a virtual application pattern, the autowiring capability of the elastic load balancing service automatically generates a unique prefix for each deployment.

When you manually add a routing policy to a virtual application pattern, the context prefix is optional. If you do not specify a context root ID, the virtual host name and context root are reserved by the elastic load balancing service. Therefore, if you try to deploy another virtual application instance with the same values, an error message is displayed to indicate that there is a reservation conflict. Stopping the virtual application instance that originally used these values does not release the reservation. You must delete the original virtual application instance to enable another deployment to use the same virtual host name and context root or specify a different context root before a new virtual application instance is deployed.

Figure 6-5 shows the available routing options to customize for routing your component.

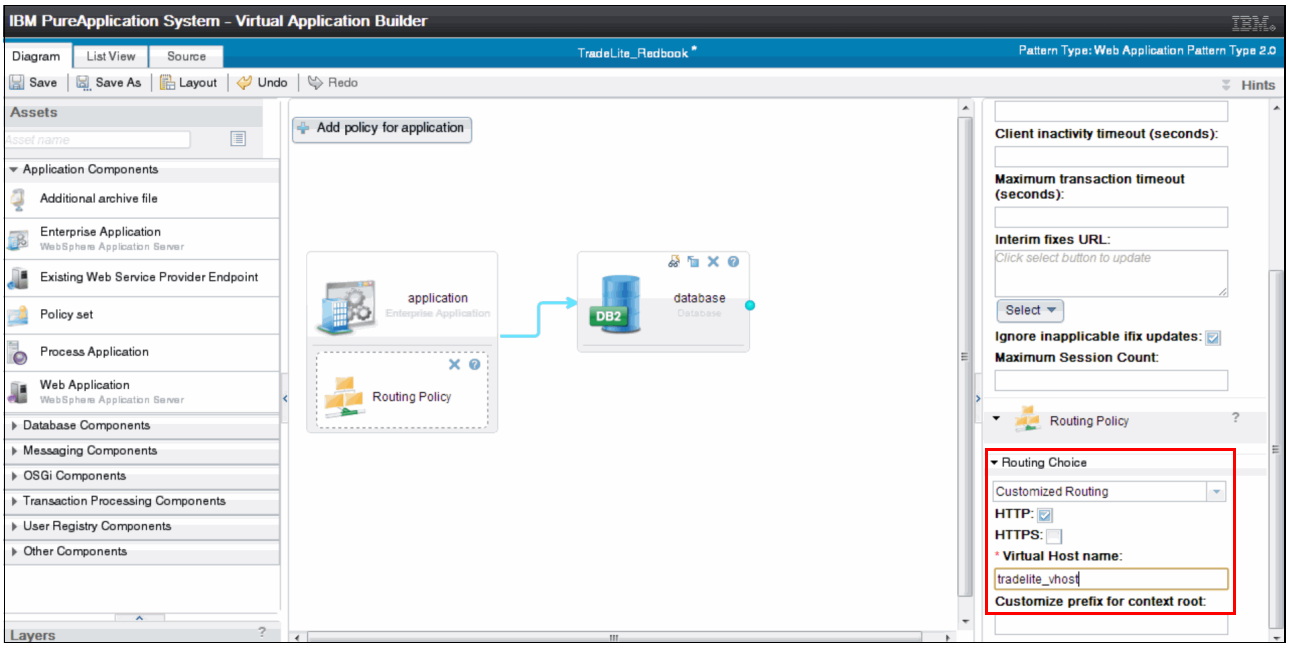


Figure 6-5 Routing options

Logging

A log policy can be added to your application component part to specify configurations for logging.

A log policy features the following attributes:

- Log detail levels: Specifies the usage of log levels to control which events are processed by Java logging.
- Additional Log Files or Directories to Monitor: Specifies a semicolon-delimited list of directories or files to monitor. To specify that an entry is a directory, add a suffix to the entry with a slash, for example, `/var/log/myApplication/`, or add a prefix to it with a string, such as, `dir:/var/log/myApplication`. You can use an asterisk wildcard in the file-specification only, for example, `/var/log/myApplication/*.log`. By using the wildcard in the following manner, `/var/log/*/my.log`, is invalid. Any directory that is specified is visible in the Log Viewer.

Figure 6-6 shows different levels of logging that can be applied to your component.

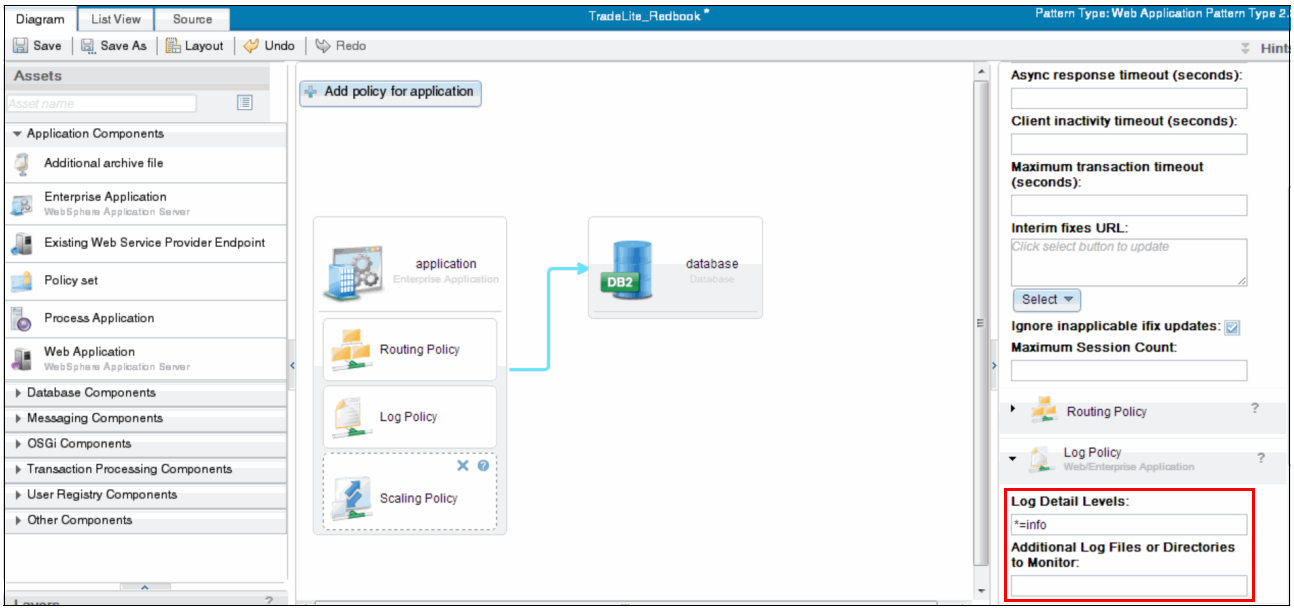


Figure 6-6 Logging options

Figure 6-7 shows the deployment overview for the web application pattern.

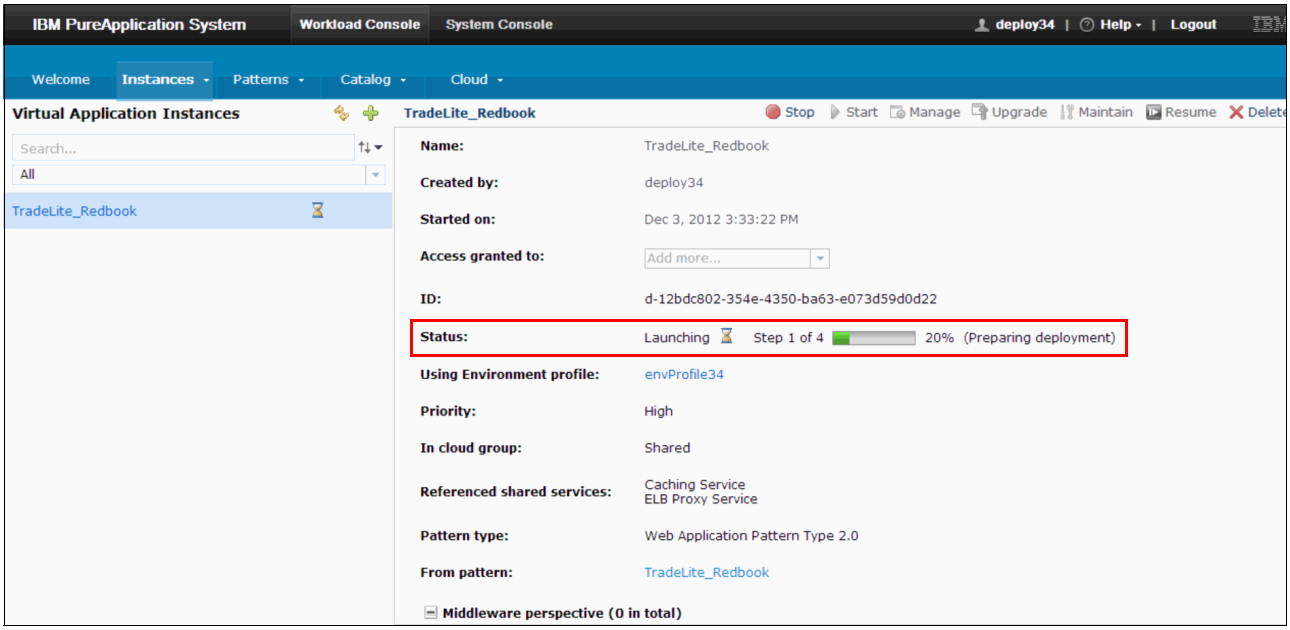


Figure 6-7 Deployment overview

Figure 6-8 shows the deployment process and virtual machine provisioning.

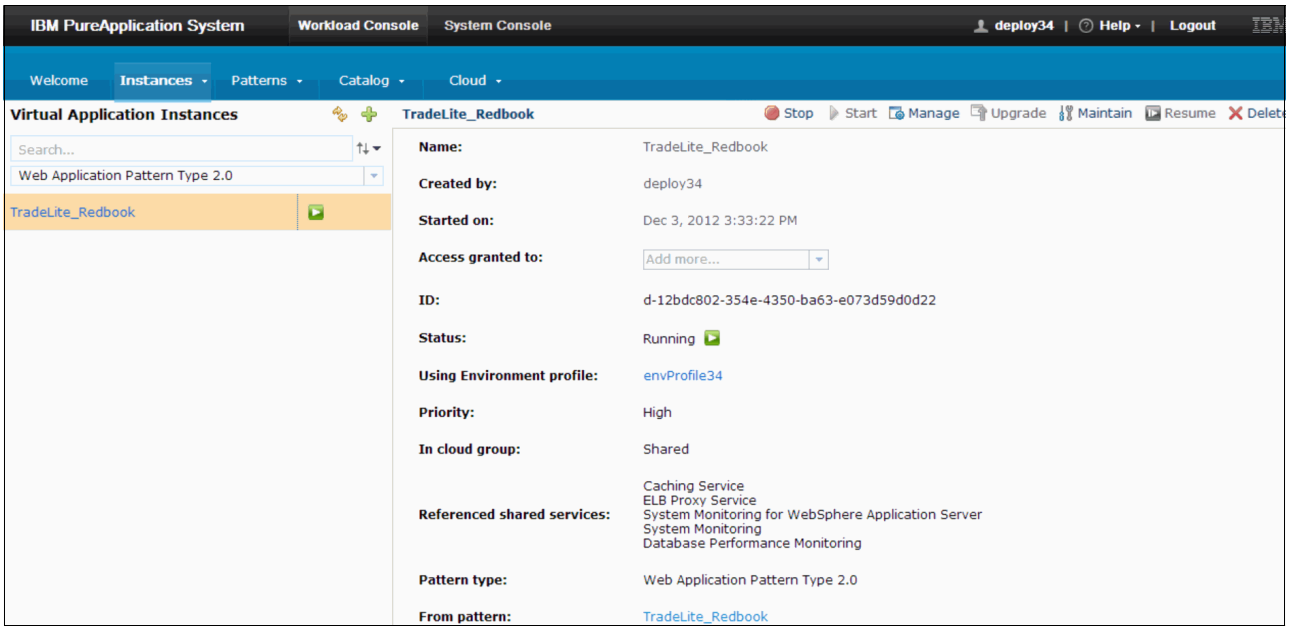


Figure 6-8 Virtual machine deployment

6.3 Creating virtual application patterns with the virtual application builder interface

Virtual application builder is the core to create a virtual application pattern. You can extend the functionality of default patterns that are provided by PureApplication System by using the following method:

- Editing virtual application pattern: By using the virtual application builder interface, you can edit existing preinstalled patterns to create your pattern and then deploy it. The following section shows how you can customize an existing pattern. Figure 6-9 shows the virtual application builder interface that is provided by PureApplication System.

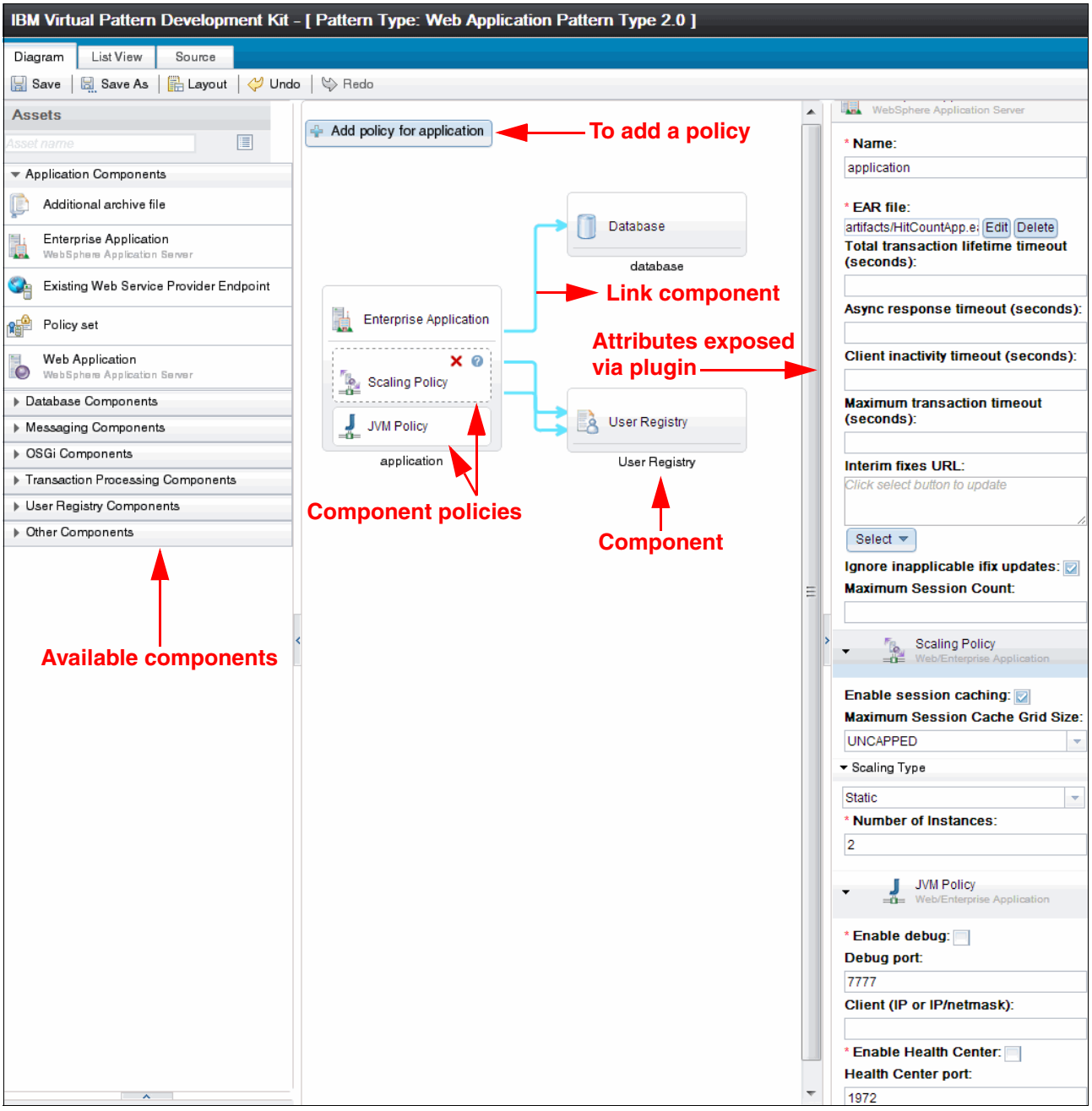


Figure 6-9 Virtual application builder

- Figure 6-10 shows an enterprise application component that is connected to a DB2 database component. It also shows a web application pattern, which has an application server and a database component that uses DB2.

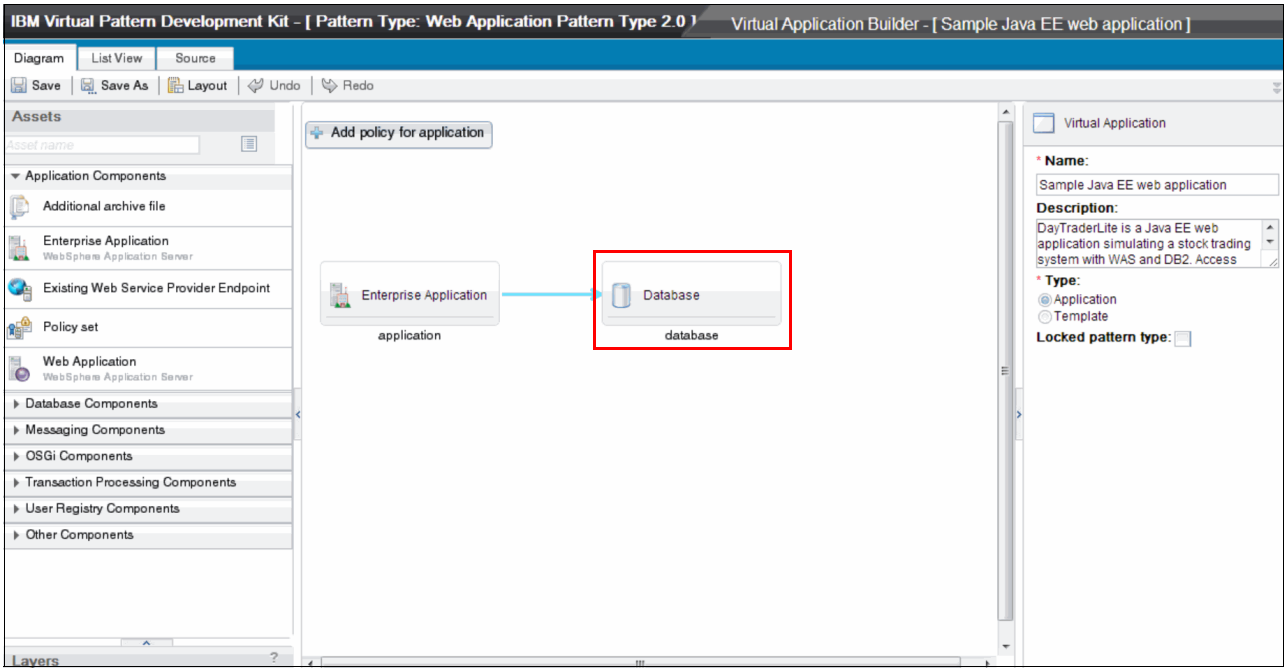


Figure 6-10 Generic Web Application pattern

By using the virtual application builder console, you can delete the DB2 component and add the Informix database component. This process is a simple drag with a link component connection. Figure 6-11 shows how the database is now changed to Informix and that the configuration of data sources is provided on the right side of the panel. You can select from the components that are listed to add into the palette by using the virtual application builder interface.

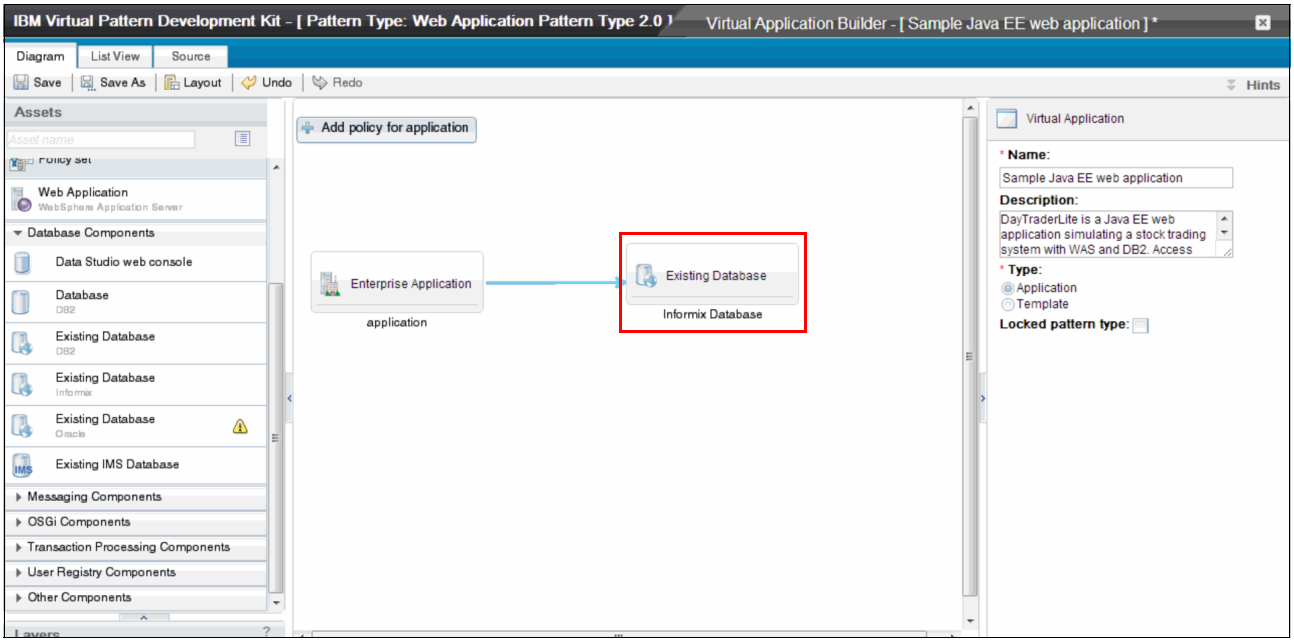


Figure 6-11 Database component replaced

6.4 Lifecycle management of virtual applications

A virtual application pattern is used to define and manage virtual applications. By using virtual applications, you define your application and non-functional requirements. The middleware that runs this application is addressed by the PureApplication System. Within your application, you define what policies are attached and the complete lifecycle of your application from inception through termination.

6.4.1 Virtual pattern development kit

The Virtual pattern development kit provides a development type virtual image of workload deployer and tooling that can be used to create custom virtual application patterns and deploy and test in this environment. The kit also includes web application and database pattern types, the IBM Image Construction and Composition Tool, the Plug-in Development Kit, and the Command Line tool. You can use this kit as a development or test platform to validate patterns and images before they are deployed.

When you are preparing to install the virtual pattern development kit, review the following link for a video demonstration that provides detailed information of how to install the kit:

`http://www.youtube.com/watch?v=nTpk-55kBTc`

A virtual application pattern consists of plug-ins that form the basis for creating virtual applications. Plug-ins are the core for virtual application patterns and are described next.

6.4.2 Plug-in overview

Plug-ins are used to define components, policies, and links. These plug-ins are used by virtual applications to create virtual application patterns. The IBM Workload Plugin Development Kit (PDK) allows developers to build their own plug-ins that are then deployed to the IBM PureApplication System.

Plug-ins consist of a package that includes the following configuration files and scripts.

- Contents of a Plug-in: Plug-ins consist of a configuration file, such as `config.json`. The following configuration and implementation extensions perform lifecycle management:
  - `config.json`: Required configuration file.
  - `appmodel/metadata.json`: Used to specify the components, links, and policies in the plug-in that are shown to users in the virtual application builder to build a model of a virtual application.
  - `appmodel/tweak.json` and `appmodel/operation.json`: Used to change deployed virtual application instances from the deployment inlet in the system console.
  - `bundles/{name}.jar`: The main file that contains the scanners, transformers, and provisioners of the plug-in.
  - `nodeparts/{name}.tgz`: Artifacts that are installed by the activation script.
  - `parts/{name}.tgz`: Extensions that are used to communicate with the system console about the lifecycle of a virtual application.

- Node parts, parts, and packages:
  - Node parts are installed by the activation scripts. Node parts contain the `setup.py` and can install **`start.py`** or **`start.sh`** scripts before the workload agent is started.
  - Parts contain scripts that manage the lifecycle of components by using roles. Roles are scripts that can be used by other scripts.
  - Packages are a collection of both node parts and parts. Packages should have a unique name.
- Roles: Roles provide the lifecycle scripts for the management of software and applications. They offer event notifications between components. Each role is described in a topology document by a JSON object, which is contained within a corresponding VM template. Roles include the following states:
  - INITIAL: Roles start in the initial state. The **`install.py`** script for each role is started and, based on the result, it is moved to the INSTALLED or the ERROR state.
  - INSTALLED: The **`configure.py`** script runs during this state, if it exists.
  - CONFIGURING: The **`start.py`** script runs during this state, if it exists.
  - STARTING: The automatic state setting stops. A lifecycle script must explicitly set the role state to RUNNING.
  - RUNNING: States the role the virtual machine is in.
  - Topology document: The topology document is a JSON object and VM templates element. In the topology document, is a JSON array of VM template elements. Each element in the array represents a virtual machine to deploy. Components correspond to VM templates, and links correspond to links or dependencies between components.

6.4.3 Virtual application lifecycle

Plug-ins are the core for creation, deployment, and management of virtual applications. This section describes the lifecycle of a virtual application from its creation to final provision and deployment.

- Creating an application model

  Virtual application builder is used to create or update virtual applications. You design a virtual application that is based on a virtual application pattern. This pattern is a collection of components, links, and policies. Virtual application builder scans for the artifacts to guide the modeling between components, links, and policies.

  Figure 6-12 shows the application modeling process

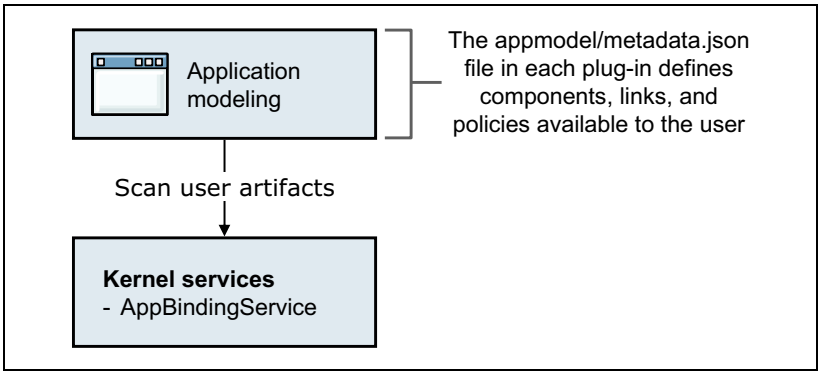


Figure 6-12 Application process

- Deploying the application

  Kernel services store the application model that is created by using the virtual application builder. When the virtual application is deployed to the target cloud, component transformation is started with link transformations. The kernel services convert the application model. The model is converted from logical description into a topology document or a physical description by using TopologyProvider and TopologyProcessor transformer implementation. TopologyProvider implementations are plug-in specific implementations that transform the application model into an unresolved topology. PureApplication System W1500 embeds Apache Velocity™ 1.6.2 as a template engine. Example 6-1 shows the sample transformer component and the corresponding TopologyProvider implementation it uses.

Example 6-1 Example for sample transformer component

```
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0"
immediate="false" name="SingleServer">
    <!-- implementation element determines whether trnasform is Java or
Velocity template. -->
   <implementation
class="com.ibm.maestro.model.transform.template.TemplateTransformer"/>
   <service>
      <provide interface="com.ibm.maestro.model.transform.TopologyProvider"/>
   </service>
    <property name="component.template" type="String"
value="templates/hello.vm"/>
</scr:component>
```

Figure 6-13 shows the deployment of an application.

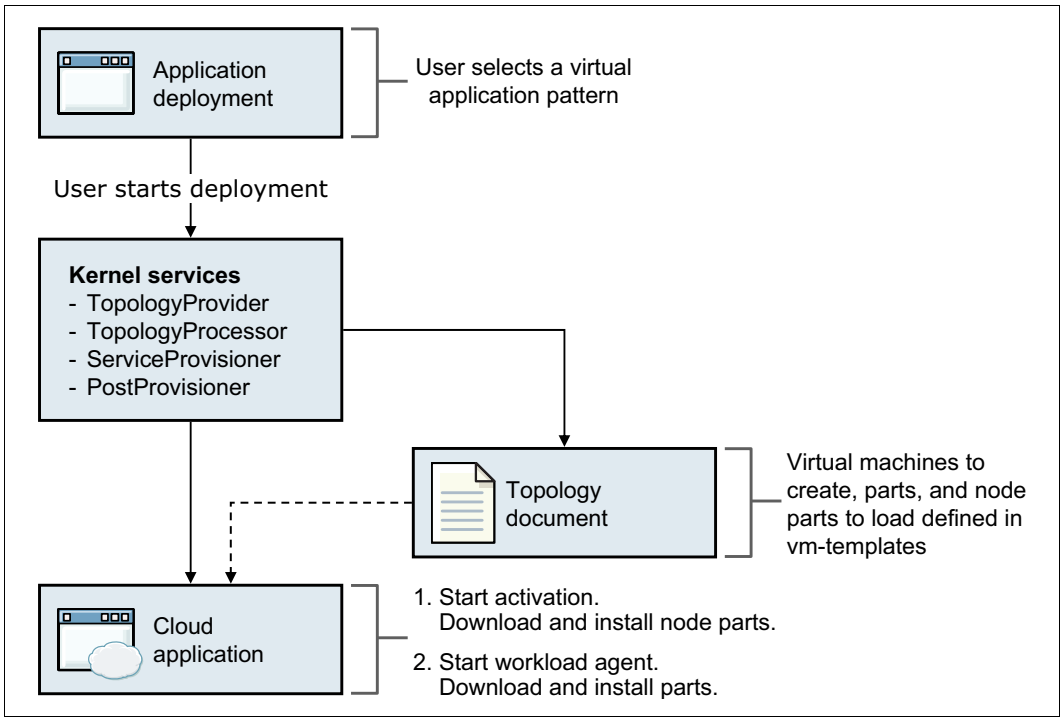


Figure 6-13 Virtual application process deployment

After the plug-in is built and packaged into `yourpluginame.tgz`, you extract this file and the contents should resemble the contents that is shown in Figure 6-14.

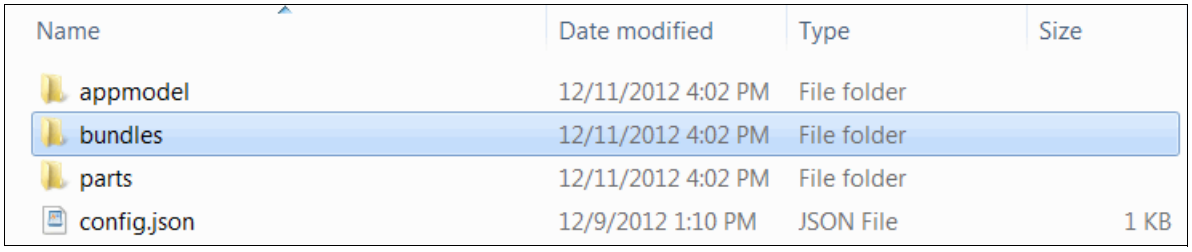


Figure 6-14 Plug-in contents

The packaged plug-in file includes the following contents:

- The transformers and provisioners are in the `bundles/{`*`name`*`}.jar` file.
- The plug-in `nodeparts/{`*`name`*`}.tgz` file, which contains the node parts that are downloaded and installed with the activation script on the virtual machine.
- The `parts/{`*`name`*`}.tgz` file contains the parts that are downloaded and installed with the workload agent on the virtual machine.

- End to End Deployment Process of virtual application

  Plug-ins use implementations of TopologyProvider that are provided by PureApplication System to convert instances of their components, links, and policies from the application model into an unresolved topology. The unresolved topology is generic. The transformers specify abstract package names rather than specific node parts or parts. The images and instance types are not yet specified.

  After the unresolved topology document is complete, the next step resolves the specific node parts and parts, and images and instance types, according to cloud details and plug-in configuration, as specified in the `config.json` file. The resolved topology is passed to a provisioning phase where resources are provisioned from shared services. The final plug-in developer exit point is the post provisioner, which is started after all services are provisioned and after the topology document is finalized and written to the storehouse.

  The topology document is written only once to the storehouse, and is never updated. A separate deployment document is written to the storehouse to represent the deployed virtual application. This deployment document is written and updated many times, and reflects the current state and status of the deployed virtual application. Example 6-2 on page 283 shows sample `config.json`, which includes parts and other information that are part of the package.

Example 6-2 Example description for config.json

```
{
   "name"         : "mywasplugin",
   "version"      : "1.0.0.1",
    "files":[
      "/PureApp_Examples/mytomcat-helloworld.war",
      "/PureApp_Examples/apache-tomcat-7.0.34.tar.gz",
      "/PureApp_Examples/ibm-java-sdk-6.0-12.0-linux-x86_64.bin"
      ],
   "patterntypes" : {
      "primary"   : {
         "ptype.firstEx" : "1.0"
      }
   },
   "packages": {
      "pkg1": [
         {
            "requires" : {
               "arch"  : "x86_64",
               "memory": 128
            },
            "parts": [
               {
                  "part": "parts/package_sample.tgz"
                  }
            ]
         }
      ]
   }
}
```

Deployment culminates in virtual machines that are deployed in the target cloud. The process of virtual application deployments includes the following tasks:

- As a part of the activation process, a script on each virtual machine is downloaded and parses the topology document. The required node parts that are specified in the topology also are downloaded and installed.
- The workload agent is a node part. The workload agent parses the topology document and downloads and installs the required parts. Finally, the workload agent initiates the lifecycle scripts for the specified roles and dependencies. The natural progress of the lifecycle scripts start and maintain the application through failure recovery.

Figure 6-15 shows the deployment steps of the virtual application.

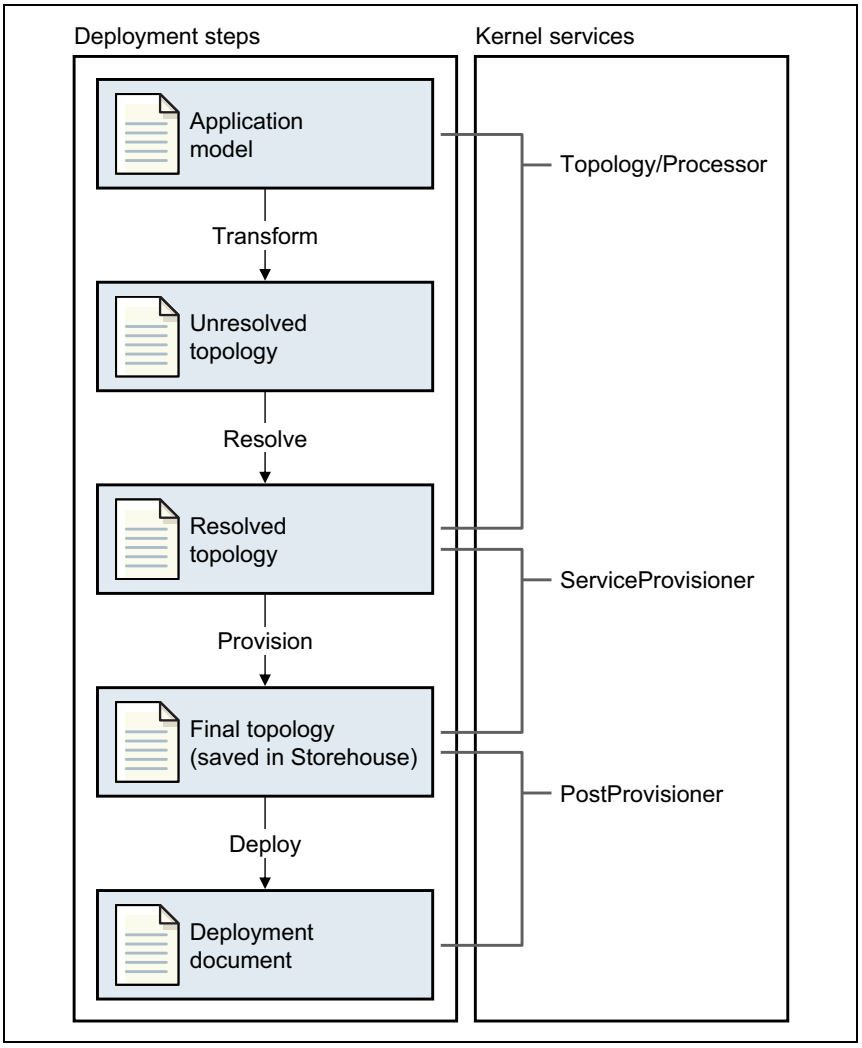


Figure 6-15 End-to-end deployment process

- Managing your deployed application

  When your virtual application is deployed, it becomes a virtual application instance. You can use the PureApplication System console to see the virtual machine to which your application is deployed. Figure 6-16 shows the process of viewing the deployed virtual application instance.

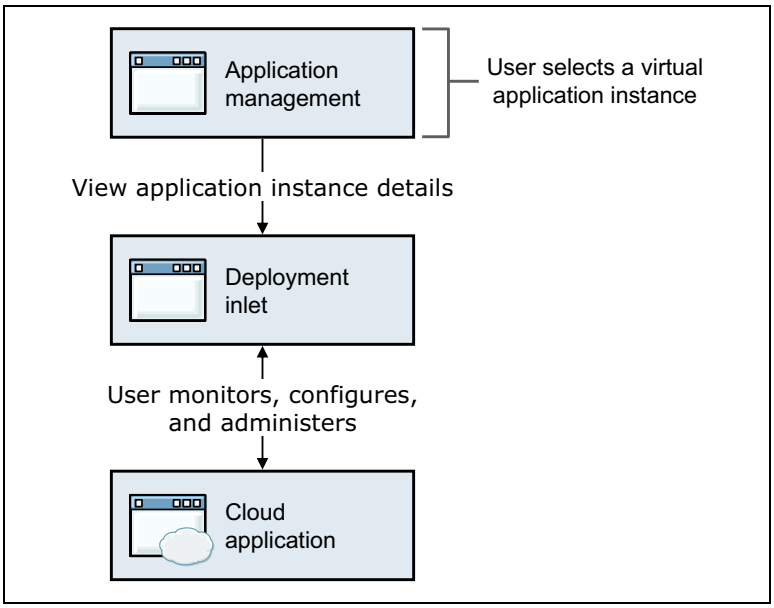


Figure 6-16 Managing applications

6.5 Plug-in environment setup and creation of custom patterns

This section provides information about setting the plug-in environment and development of custom patterns, including the deployment of the pattern into PureApplication System.

The PDK contains a build environment, samples, and tools to create plug-in projects. The PDK can be downloaded from the welcome window of the PureApplication center.

To download the PDK, use the available download link for PDK in the PureApplication console (as shown in Figure 6-17) or from this website:

https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-plugindekit

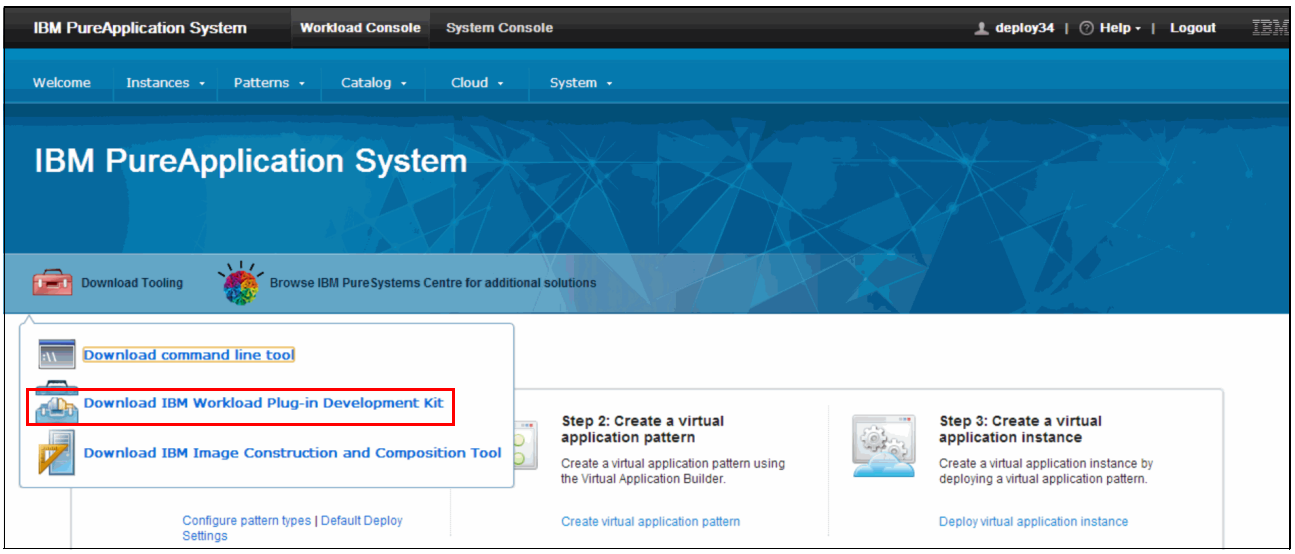


Figure 6-17 PDK download link

Complete the following steps to set up the plug-in development environment and create a sample plug-in:

1. Install and set up the environment.

   After you download the PDK and extract the file, you can view the directory structure of the extracted content, as shown in Figure 6-18.

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
bin			File folder	10/28/20...	
lib			File folder	10/28/20...	
license			File folder	10/28/20...	
build.xml	1,668	619	XML Docu...	10/28/20...	D6F9...
pdk-1.0.0.6.encoded	23,883...	23,881...	ENCODED F...	10/28/20...	B8C2...
README.txt	549	348	Text Docum...	10/28/20...	8F3F...

Figure 6-18 PDK extracted file

2. Set up the plug-in environment:
   - You can use Eclipse as a development environment (Eclipse V3.6.2, 32-bit). The Java Platform, Enterprise Edition version is recommended, but is not required. You also can use Apache Ant, which is included the Eclipse download.
   - Java Standard Edition 6, 32-bit, Apache Ant, 1.7 or later.

– Set ANTHOME so that Apache Ant can be located to run the build. You can use Apache Ant, which comes with Eclipse. Example location to find Ant binary:

```
C:\Users\IBM_ADMIN\Desktop\Eclipse\plugins\org.apache.ant_1.8.3.v20120321-1730
```

– Run Apache Ant

Figure 6-19 shows the successful build and sample plug-in examples are created.

Figure 6-19 Apache Ant build

3. All of the packages, samples, and plug-ins are expanded within the workspace.
4. The workspace can be imported into Eclipse for plug-in and pattern creation.

5. You can use the sample imported examples as a starting point to create custom patterns. Complete the following steps to create your own plug-in project:

   a. Copy sample hello pattern, `patterntype.hello`, and the `plugin-depends` projects from the `iwd-pdk-workspace`, as highlighted in the list of sample projects, which is shown in Figure 6-20.

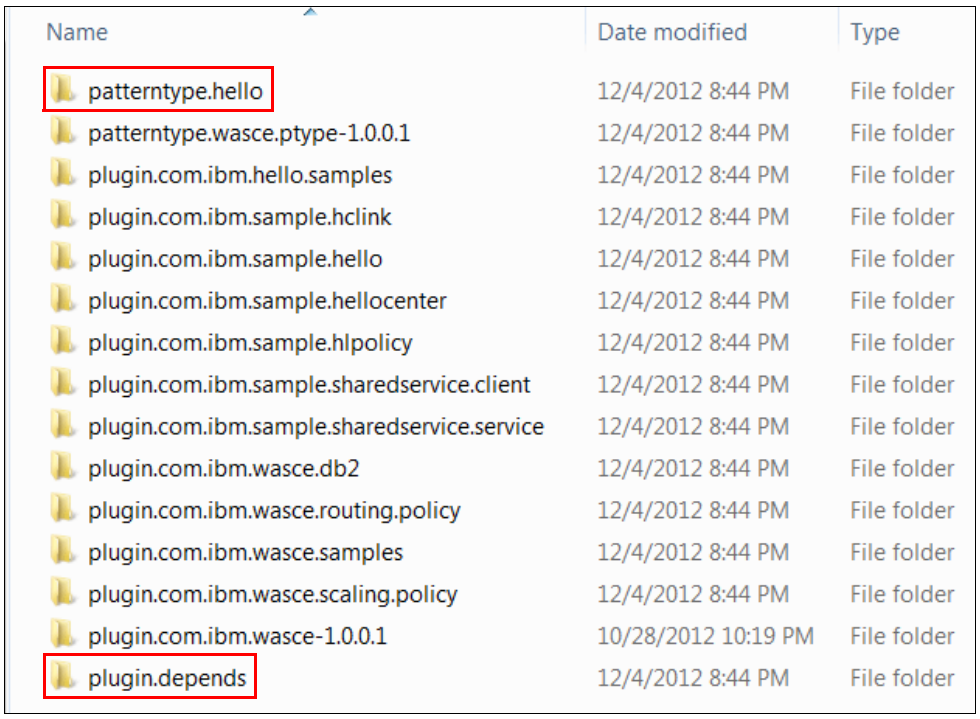


Figure 6-20 Sample example plug-in projects

b. Change the pattern name to your own pattern name, `patternType.firstEx`, as shown in Figure 6-21.

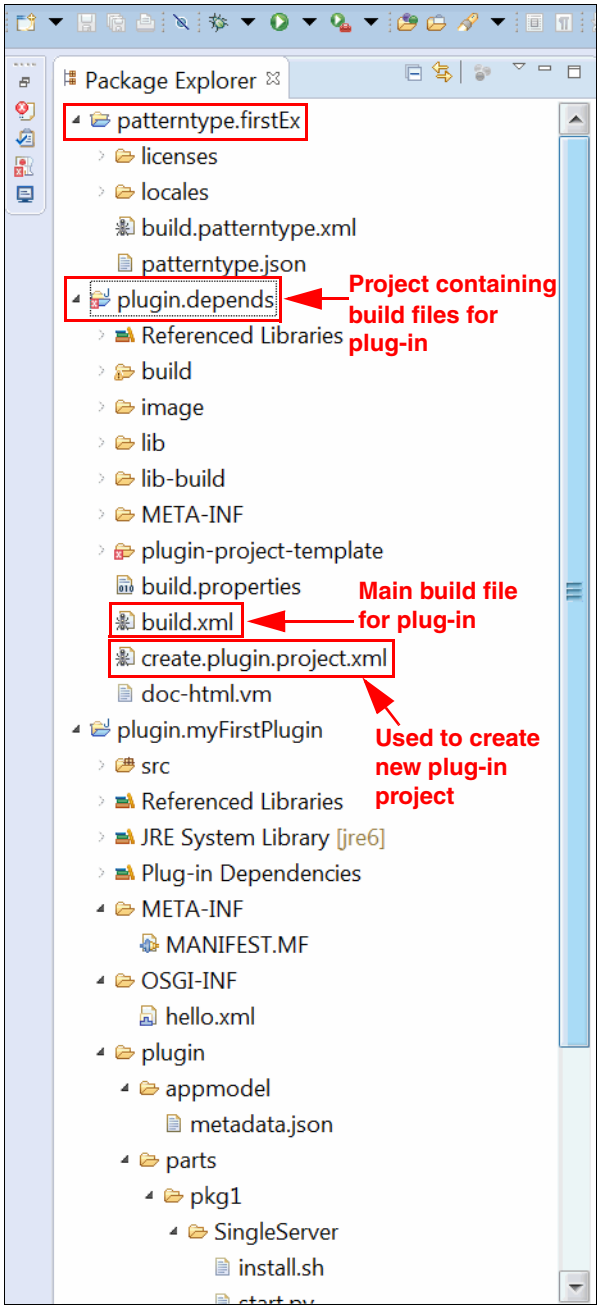


Figure 6-21 Eclipse sample project structure

c. Right-click **create.plugin.project.xml**.

d. Select **Run as** → **Ant Build** to create a plug-in project, as shown in Figure 6-22.

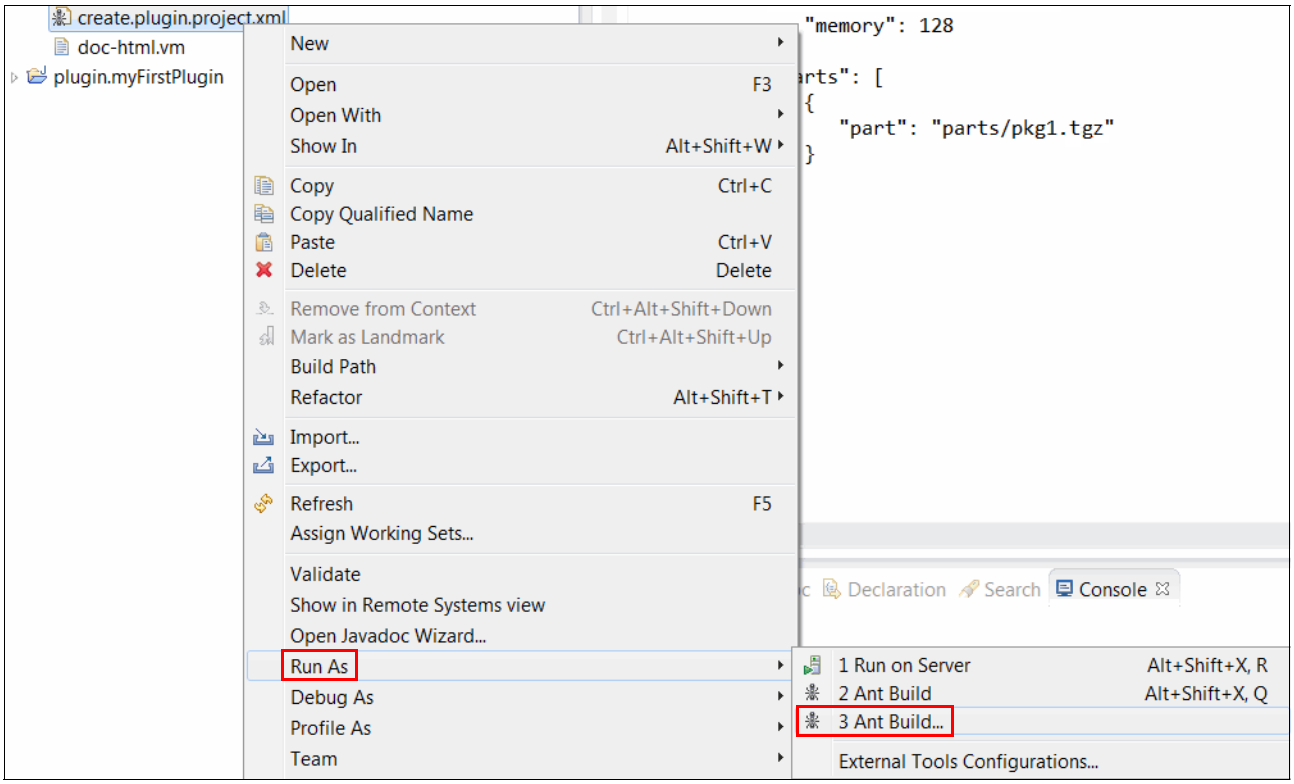


Figure 6-22 Create new plug-in project

e. Specify the plug-in project name and other configurations (such as `-Dproject.name=plugin.myFirstPlugin`, `-Dplugin.name=plugin.myFirstPlugin`, `-Djava.classname=com.ibm.redbook.sample.myFirstPlugin`), as shown in Figure 6-23.

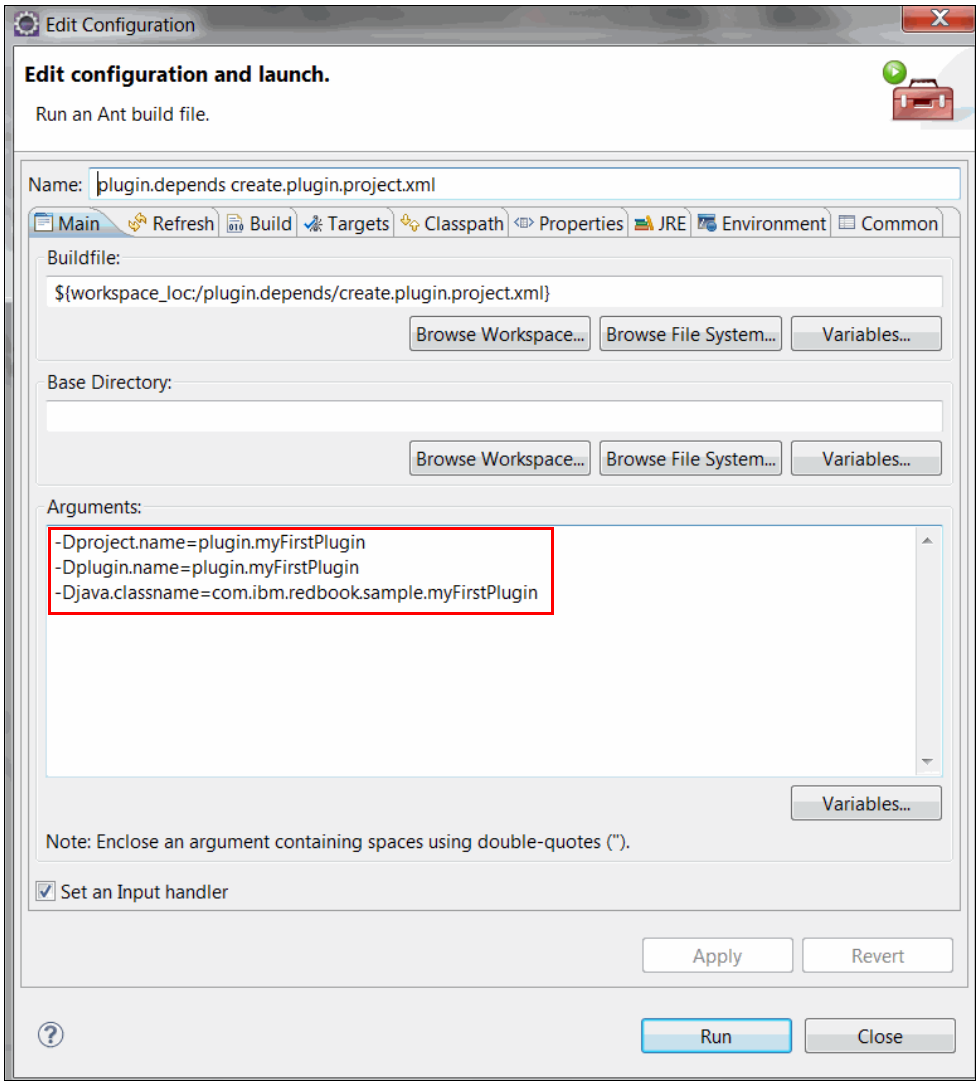


Figure 6-23 Plug-in name details

In Figure 6-24, the marked files are the files that must be edited or created for the sample plug-in.

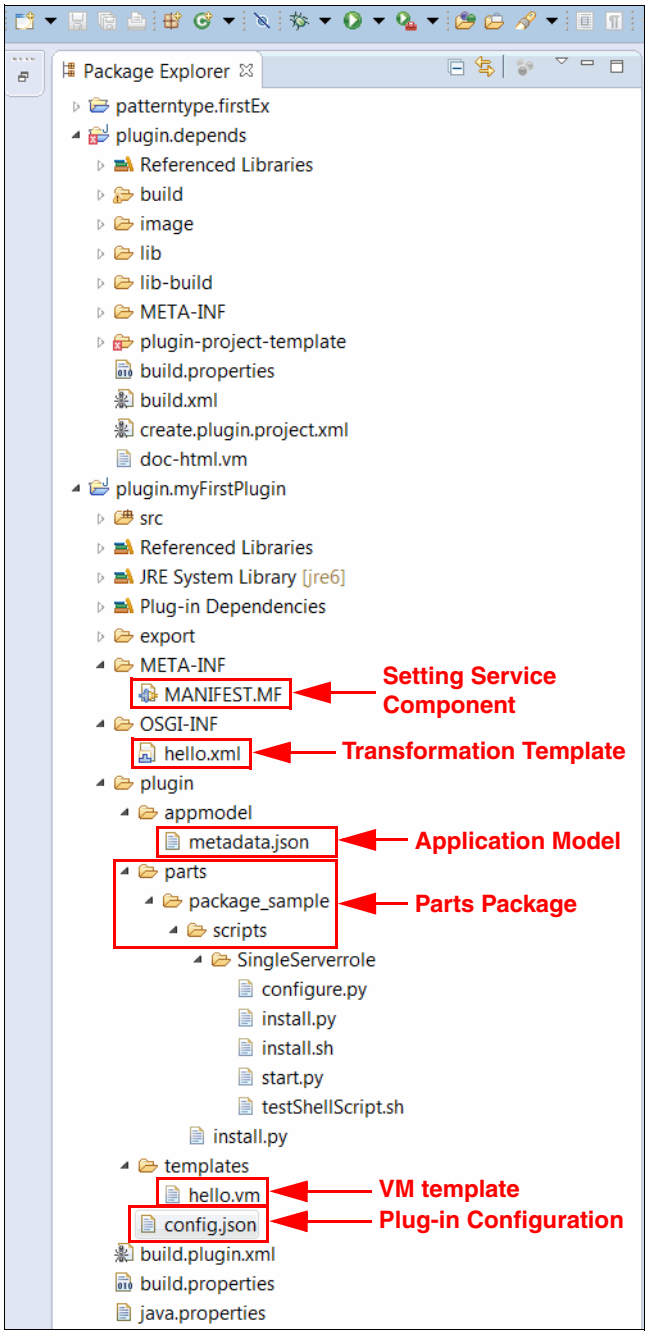


Figure 6-24 Eclipse directory structure

f. Edit the `patterntype.json` file, as shown in Figure 6-25.

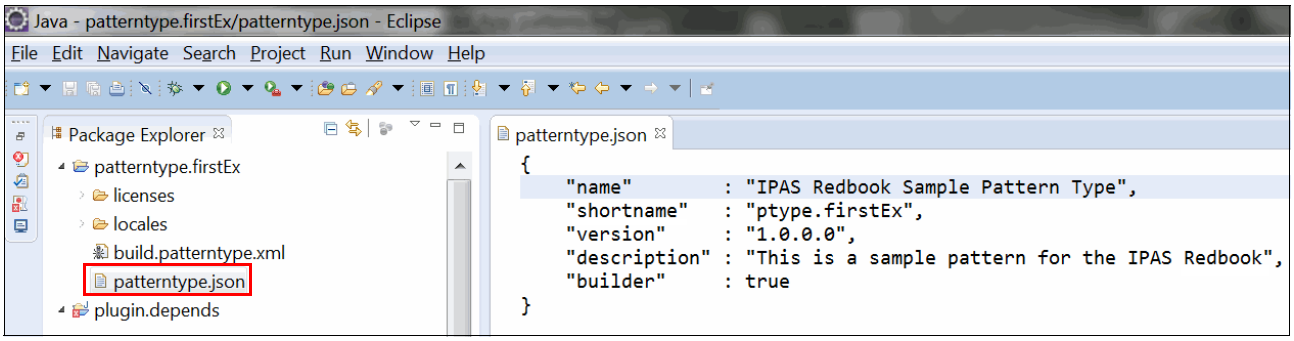


Figure 6-25 Sample patterntype.json

g. Edit the plug-in (`config.json`) file to have plug-in association with pattern type (`ptype.firstEx`), as shown in Figure 6-26.

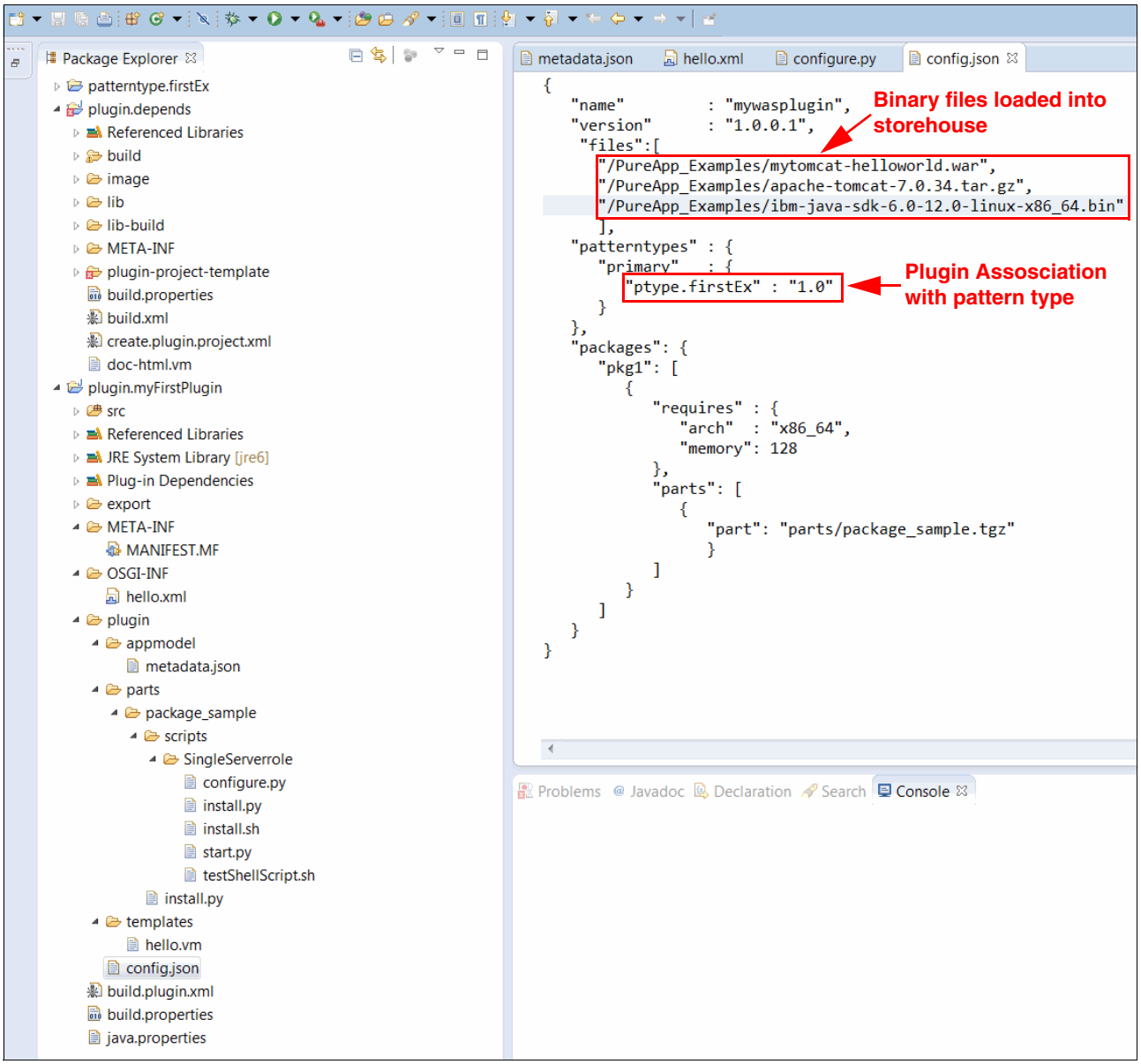


Figure 6-26 Plug-in Config.json

h. Edit the application model file (`metadata.json`) to include the component name (`Single Server`) and other attributes (`attr1`) that you want to make available to the user for customization, as shown in Figure 6-27.

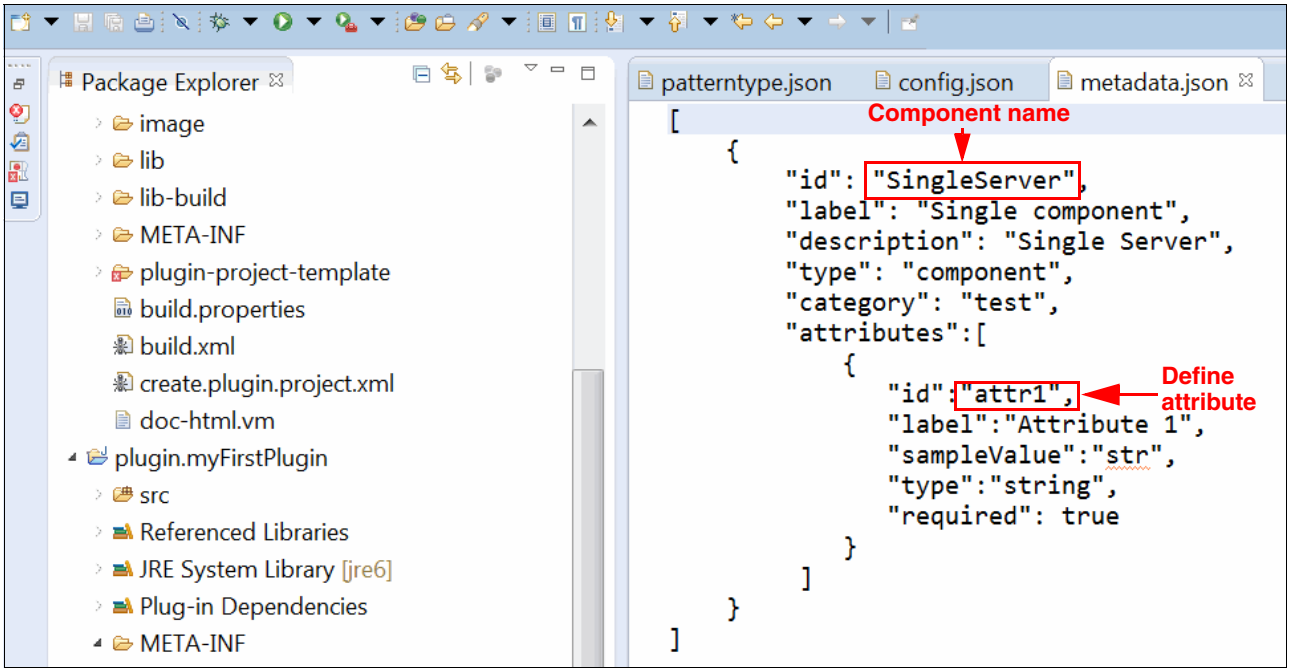


Figure 6-27 Application model metadata.json

i. Edit the transformer implementation class by using template topology. Associate the component (`SingleServer`) and the VM template (`hello.vm`), as shown in Figure 6-28. This process is how the transformation from application model to physical model is specified.

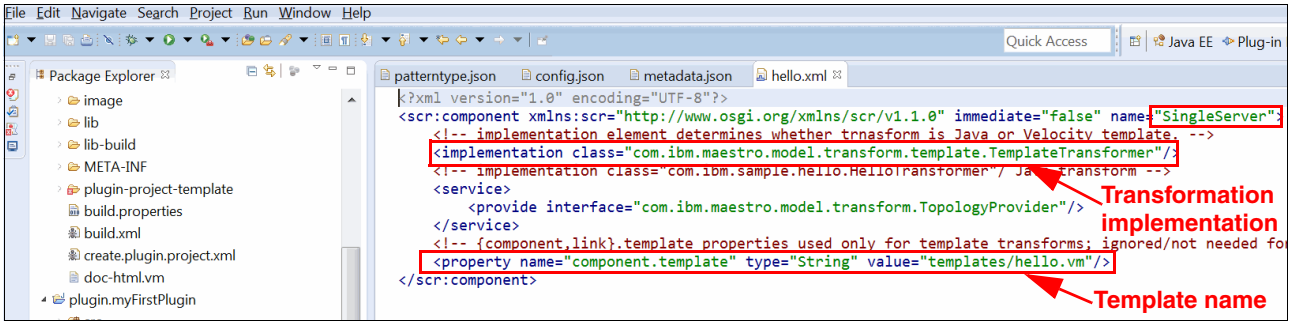


Figure 6-28 Topology hello.xml

j. Make sure that the service component includes the correct reference to the service registry in the “Manifest.MF” file, as shown in Figure 6-29.

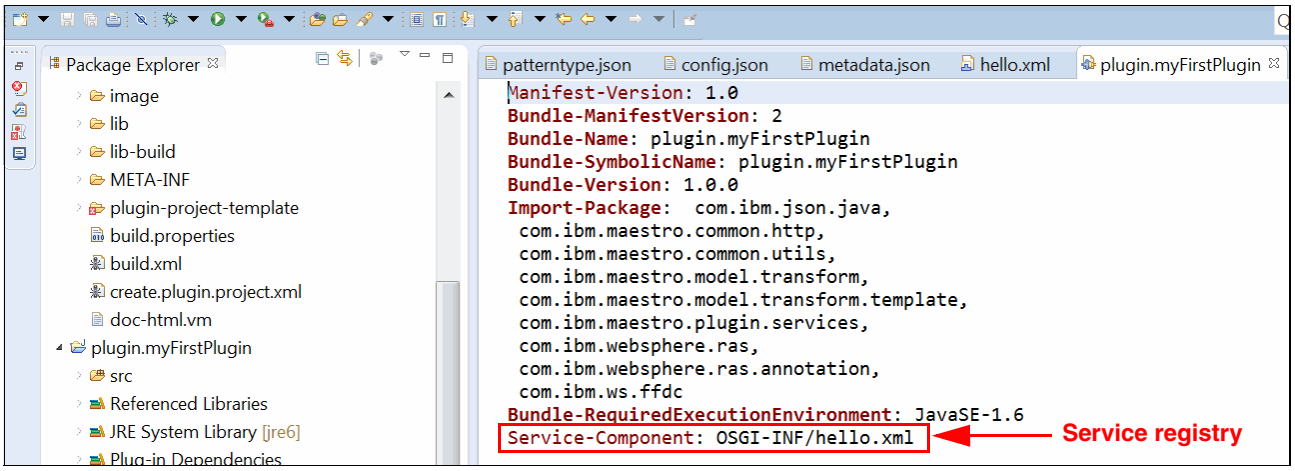


Figure 6-29 Sample Manifest file

k. Edit the template to specify the parts package for the plug-in, which you specified in `config.json`, as shown Figure 6-30.

Figure 6-30 Virtual template

l. Build the plug-in package:

   i. Right-click **build.xml** in the plug-in-depends project folder.

   ii. Select **Run as**.

   iii. Select **Ant build** to generate the plug-in package, as shown in Figure 6-31 on page 297.

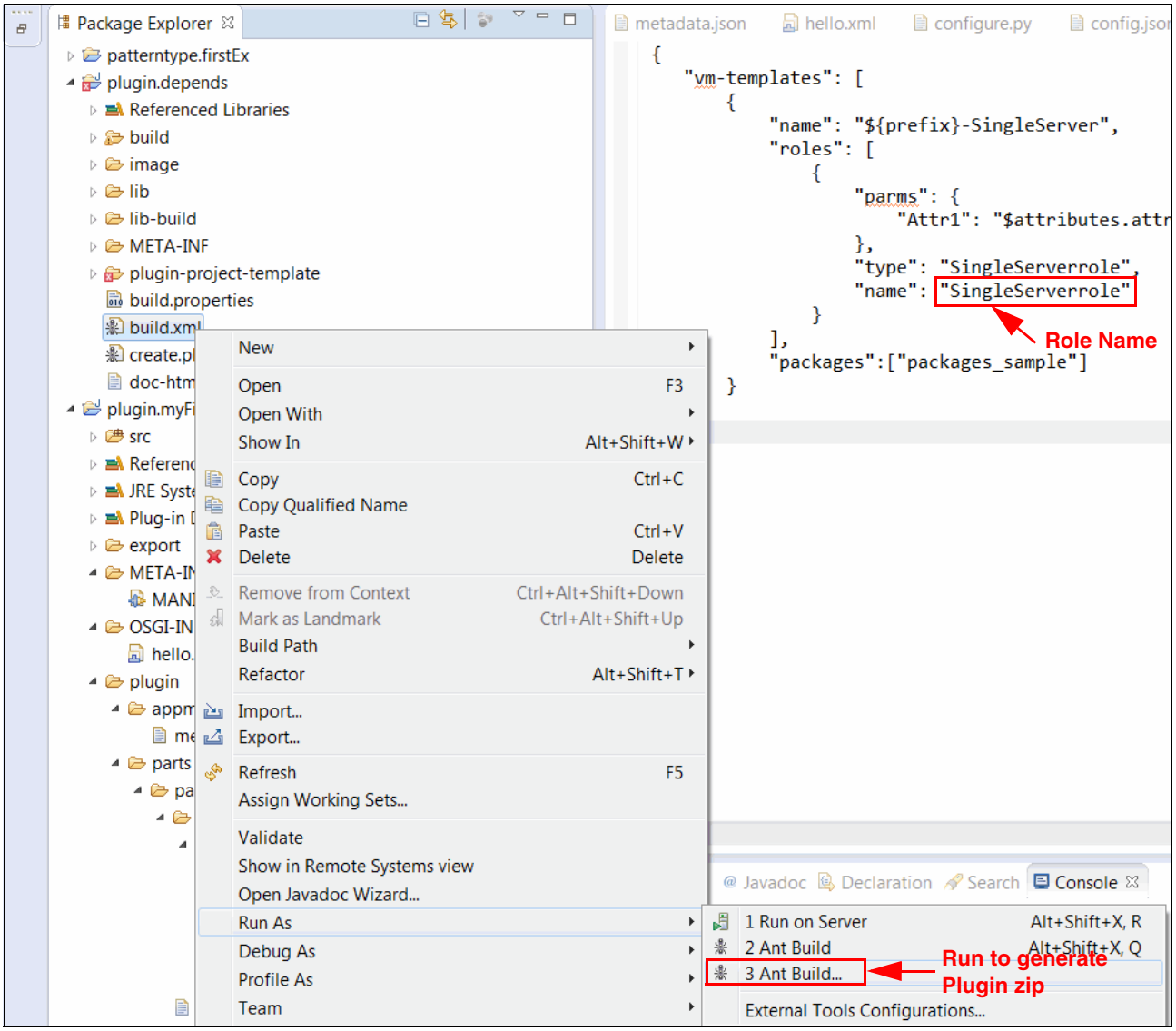


Figure 6-31 The build.xml in plugin.depends folder

m. Locate the plug-in build package in the image folder, as shown in Figure 6-32.

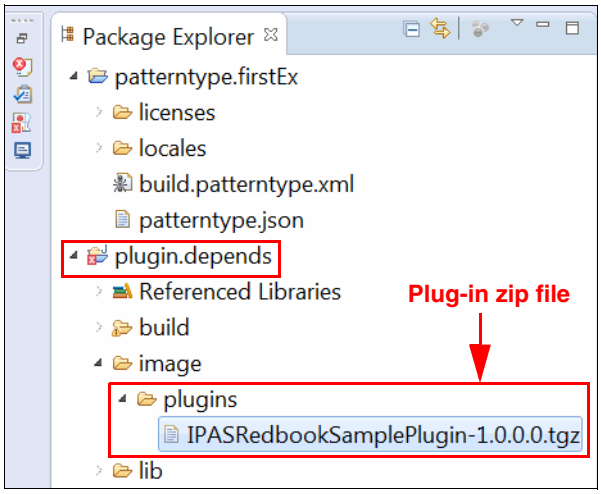


Figure 6-32 Plug-in exported .tgz structure in the image folder

Complete the following steps to build the pattern to include the plug-in package from the previous build:

1. Right-click **build-patterntype.xml**.
2. Select Ant build to generate the pattern package, as shown in Figure 6-33.

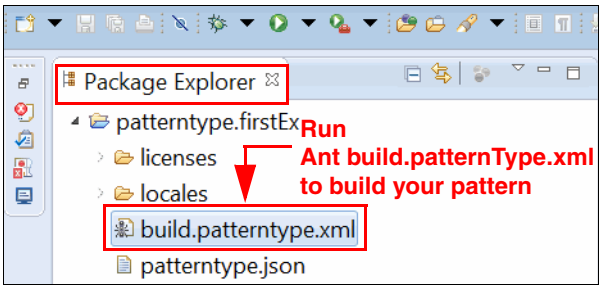


Figure 6-33 Pattern Build XML

The pattern package includes the plug-in package, as shown in Figure 6-34.

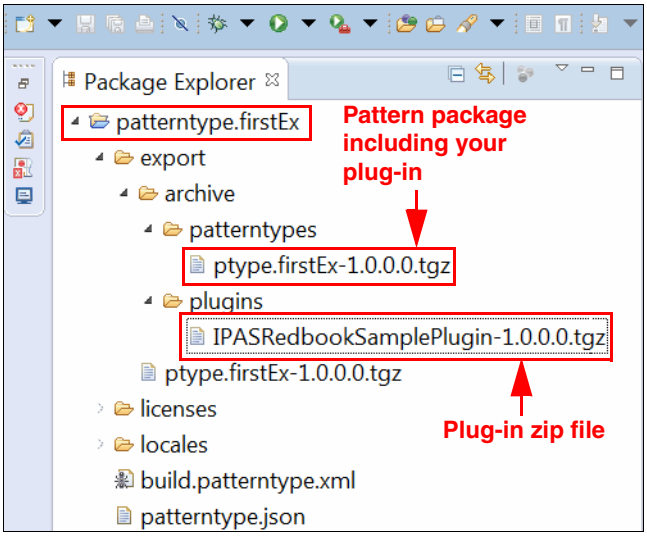


Figure 6-34 Pattern tgz file export location

3. Customize the pattern name. Edit the `message.json` file, as shown in Figure 6-35.

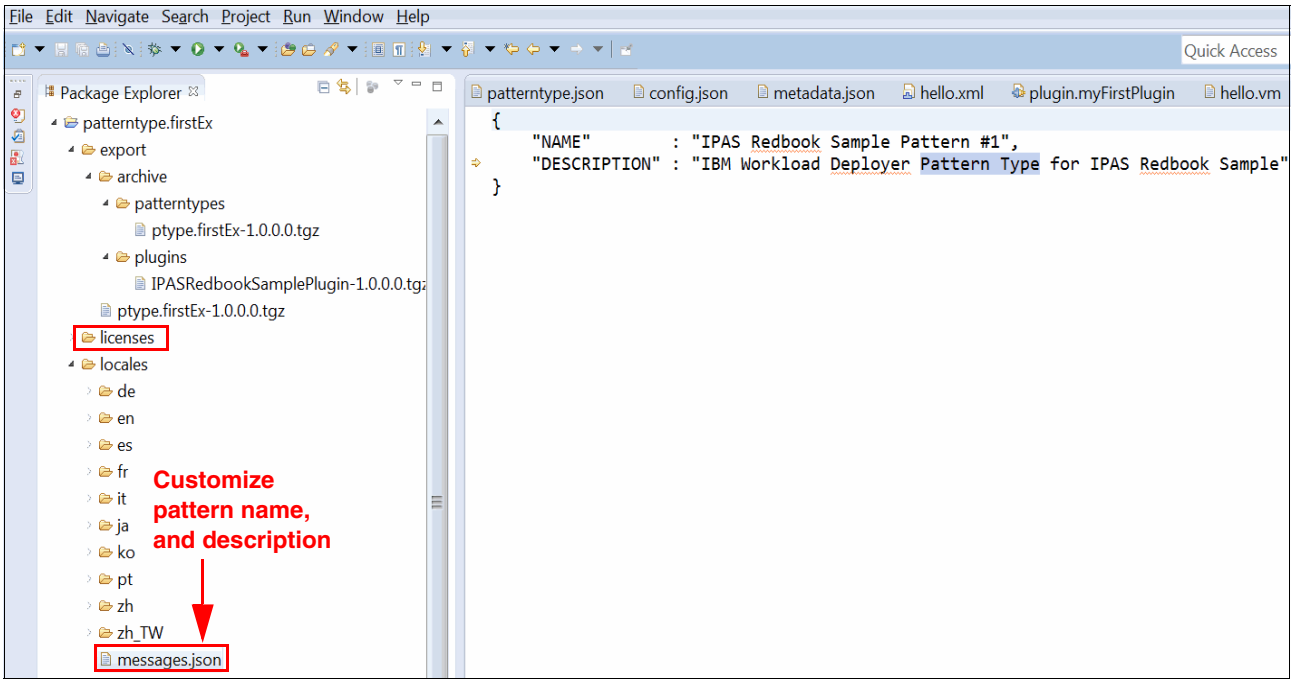


Figure 6-35 Messages.json

The next step is to import your custom pattern type. After you build your pattern type, you import the `.tgz` file, which is located under the export directory.

Click **PatternProjectName** → **Export** → **Archive** to access your pattern project. For example, `patterntype.firstEx` → `Export` → `Archive`.

Your custom pattern type is now created. The next step is to install your new pattern. Complete the following steps to install the new pattern:

1. Log in to the IBM PureApplication System Workload Console and click **Cloud** → **Pattern Type** and then click **+** to install the new pattern, as shown in Figure 6-36.

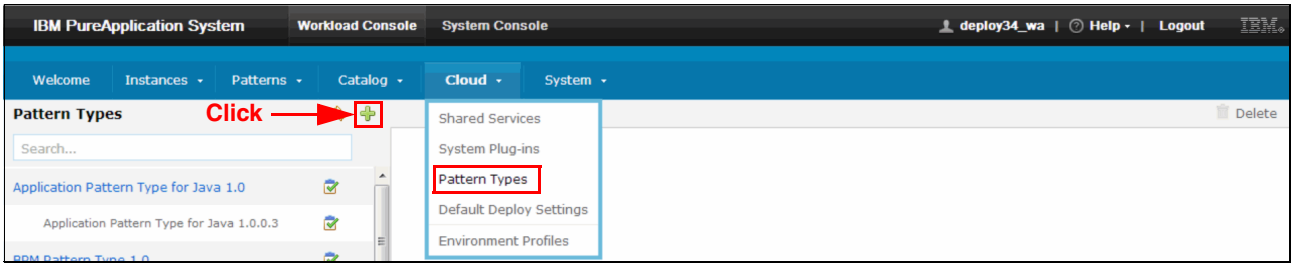


Figure 6-36 Install the virtual pattern

2. Figure 6-37 shows how to upload the new pattern. Click **Browse** to locate the directory where you saved the custom pattern you created. For example: `C:\Users\IBM_ADMIN\Desktop\pdk\iwd-pdk-workspace_Redbook\patterntype.firstEx\export\ptype.firstEx-1.0.0.0.tgz`.

   After you locate the file, you click **OK** to install the pattern.

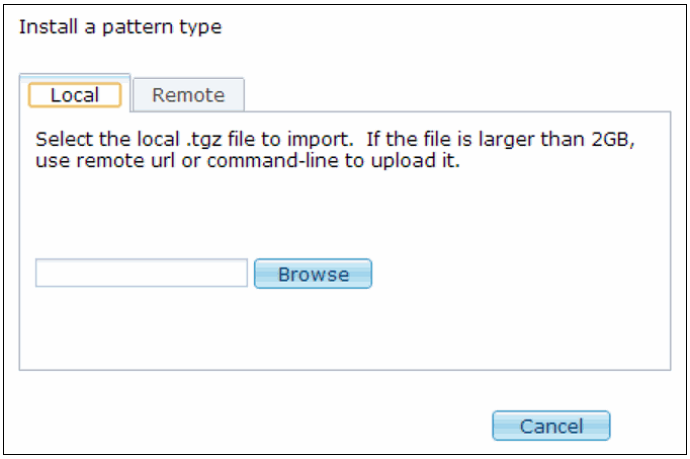


Figure 6-37 Upload .tgz file

3. After you import your pattern, the pattern is installed. You cannot create a virtual application unless you accept the license agreement, which is why you see a warning yellow sign, as shown in Example 6-38.

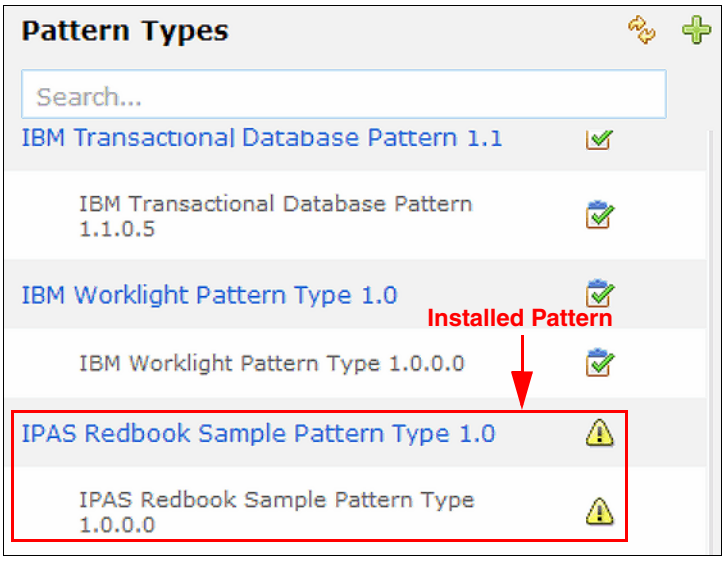


Figure 6-38 Installed pattern type after refresh

4. Click **View** to read and accept the license, which is required to enable the pattern, as shown in Figure 6-39.

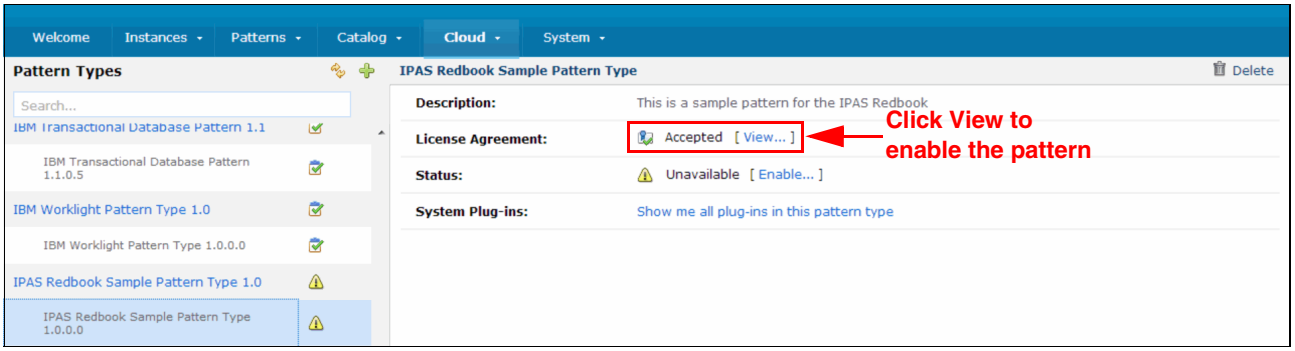


Figure 6-39 Enable pattern

5. Now you must create the virtual application that is based on the pattern you installed. From the Workload Console, select **Patterns** → **Virtual application** to create a virtual application, as shown in Figure 6-40.

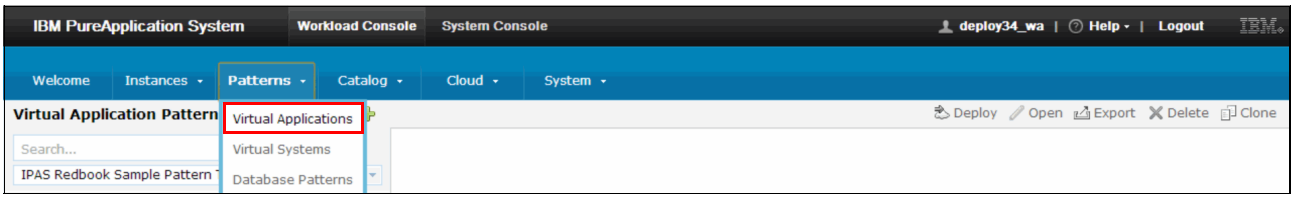


Figure 6-40 Create virtual application

6. Click **Start** to create a virtual application by using the pattern that you installed. The Virtual Application Builder interface of the PureApplication System opens in a separate window. In the left navigation panel, you find the example component that you created. Drag the Single Component into the editor, as shown in Figure 6-41. To the right side of the navigation panel, you enter the details about the name of virtual application.

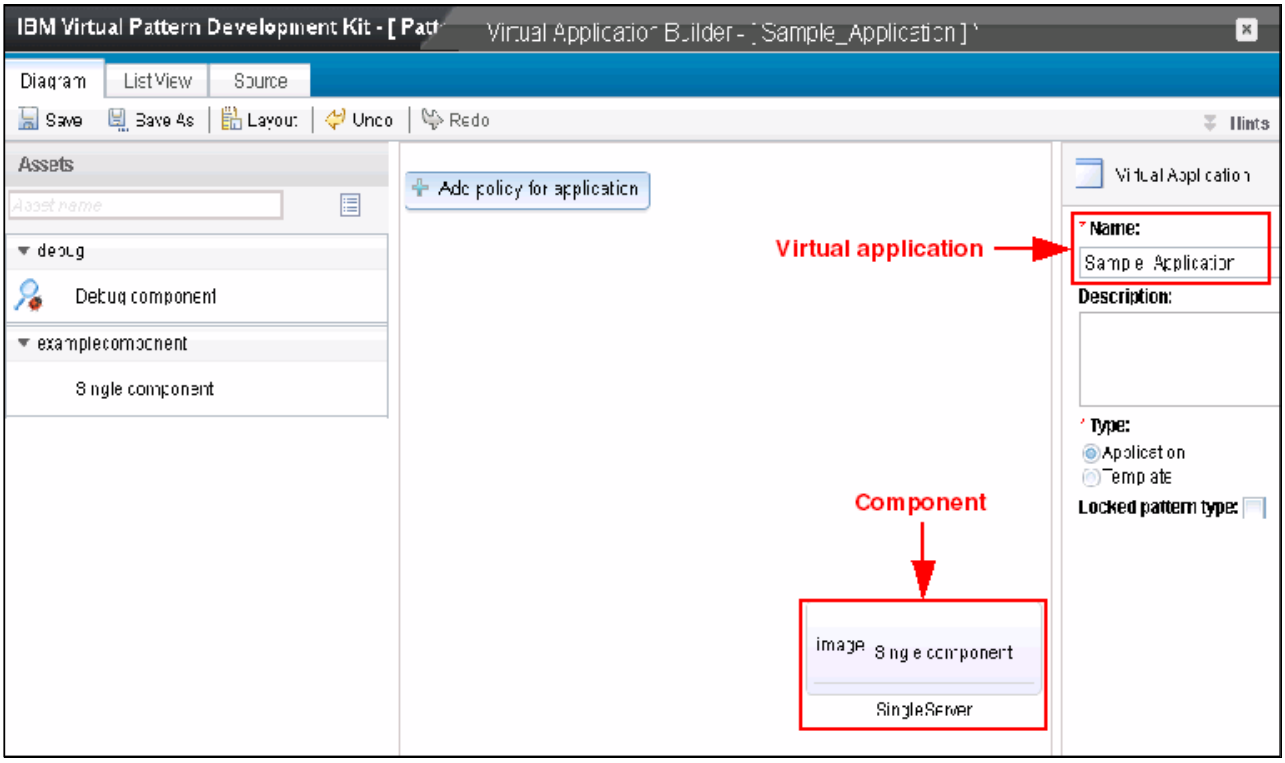


Figure 6-41 Virtual Application Builder for Single Component

7. Save the virtual application that you created by using the pattern that you imported, as shown in Figure 6-42. Click **Save As** and then click **OK** to save the application.

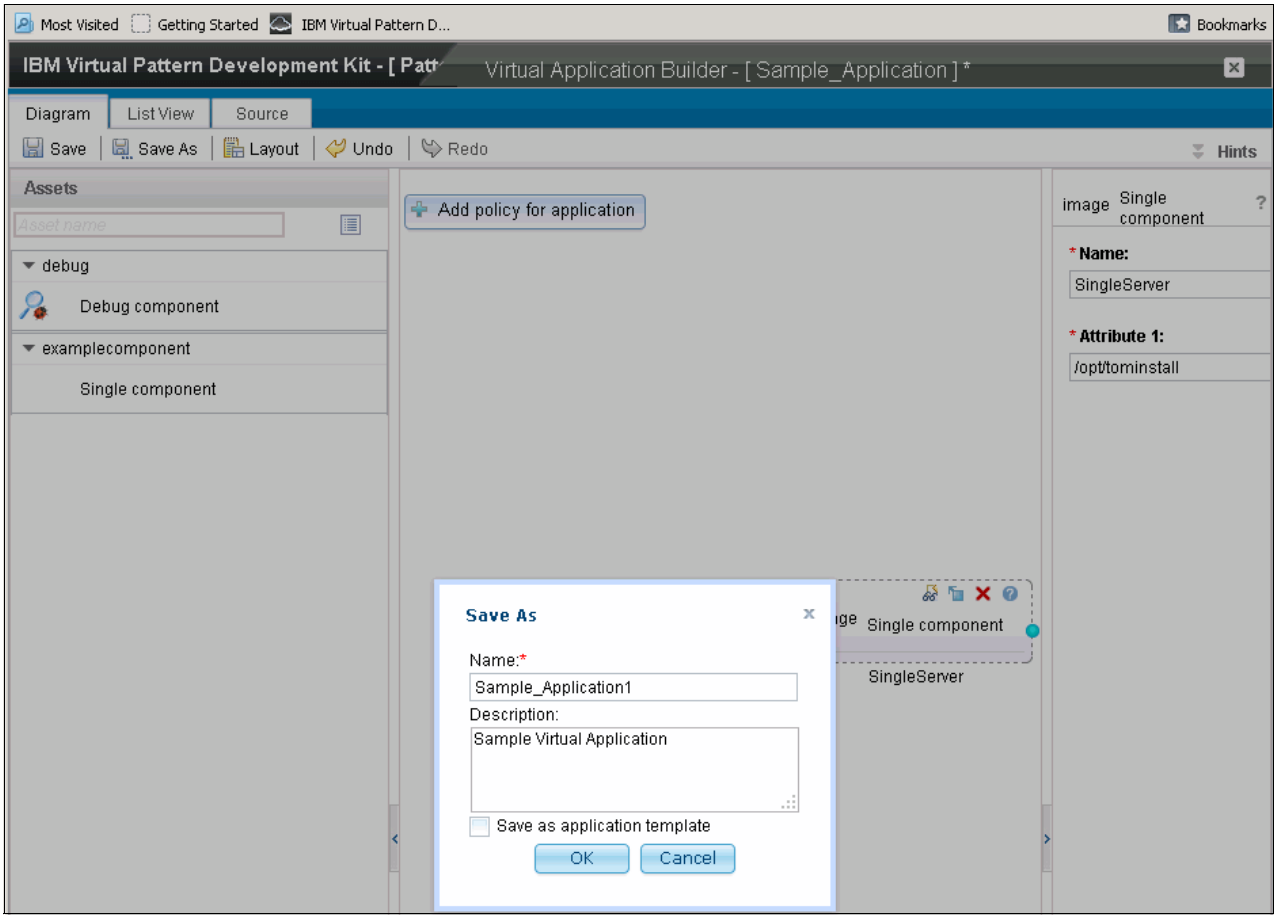


Figure 6-42 Save the virtual application

8. After the application is saved, deploy the virtual application. To deploy the application, from the workload console, select **Patterns** → **Virtual applications**. The window in Figure 6-43 on page 304 opens, which shows the name of the application. Select **Deploy** to deploy the application.

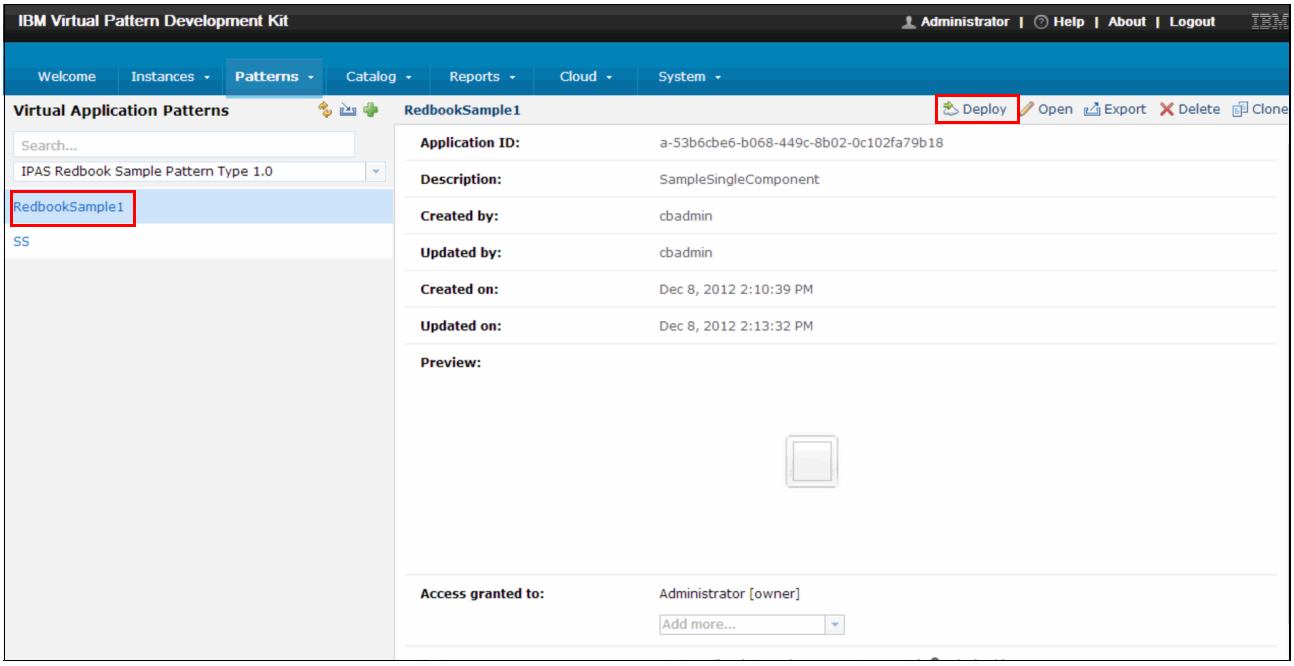


Figure 6-43 Deployment view of the sample virtual application

6.5.1 Troubleshooting and monitoring services

Plug-ins provide various operations that can be used to debug the deployed virtual applications. You can log in directly to the deployed virtual machine to run the scripts. The following websites provide more information about logging in to the deployed virtual machine:

- http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/apt_sshiwd.html
- http://www-01.ibm.com/support/docview.wss?uid=swg21614066
- http://www-01.ibm.com/support/docview.wss?uid=swg21614074

To monitor virtual application patterns, plug-ins provide operations that enable troubleshooting. The following methods are available for monitoring, logging, and troubleshooting:

- Troubleshooting services for plug-ins

  The troubleshooting service uses the deployment inlet operation capabilities with a recommended structure to provide consistency and reduce the work that is required by the plug-in to add troubleshooting operations. The troubleshooting python lifecycle script can start other scripts that the plug-in needs. It also can use helper methods that are provided by the troubleshooting service plug-in.

- Logging services for plug-ins

  The logging service provides general service to collect multiple types (text, binary) information and transfers from the virtual machine to a store for review. This service presents a subset of the collected information in the Log Viewer page of the workload console and the Virtual Application Console deployment Log Viewer tab. Log Viewer can display only information for requested the virtual machine.

- High-level design of the log service

  Plug-ins can use the generic logging service framework to specify what types of information must be collected and notify the logging implementation. You can use default log types to reduce the creation of numerous log types. Alternatively, you can create custom log files by specifying the type in the `logtype-config.json` file.

  Example 6-3 shows a sample logtype configuration file.

Example 6-3 Configuration file logtype-config.json example

```
{"types":[
   {
   "name": "adaptorName2",
   "description":"This is a new adaptor",
   "format":"text"
"start":
"\\[\\d{2}/\\w{3}/\\d{4}.*\\d{2}:\\d{2}:\\d{2}:\\d{3}.*\\-\\d{4}\\].*Start:.*",
"end":
"\\[\\d{2}/\\w{3}/\\d{4}.*\\d{2}:\\d{2}:\\d{2}:\\d{3}.*\\-\\d{4}\\].*End:.*"
   }
]}
```

- Plug-in interaction with the log service

  Plug-ins must notify the logging service with a list of directories and files to collect for the log viewer and logging service implementations.

  Plug-ins use the following methods inside the lifecycle scripts during the lifecycle execution:

  - `maestro.loggingUtil.monitor` (jsonData)
  - `maestro.loggingUtil.unmonitor` (jsonData)
  - `maestro.loggingUtil.registerPluginLogtype` (file)
  - `maestro.loggingUtil.isImplementationRegistered` (ImplName) [ImplName is the name of logging implementation class]

Important: First create a `JSON` file to list files and directories that must be monitored in the `start.py` script. Then, you can use any of the methods previously listed.

Example 6-4 shows example of calling logging service

Example 6-4 Calling for the logging service

```
listjson = '{ "role": "'+maestro.role['name']+'", "types": [ { "logtype":
"SingleLine", "type": "file",
"name": "/opt/plugin/log/instance.log"}, {"logtype": "BinaryFile", "type":
"dir", "name": "'/opt/plugin/log",
"pattern": "*.errlog"}, {"logtype": "File", "type": "dir", "name":
"/opt/plugin/log"}] }'

maestro.loggingUtil.monitor(listjson)
```

- Create a log service implementation

  A logging service supports the creation of custom implementations. These implementations act as the underlying process for a secure information transfer from the virtual machine and information storage for data review. Complete the following steps to implement a custom log service:

  a. Create a pattern type plug-in that contains the logging service implementation that is registered with the logging service.

  b. Register with logging service. Example 6-5 shows the method that is used to register and unregister logging implementation.

  Example 6-5 Register and unregister by using a logging service

  ```
  maestro.loggingUtil.registerImplementation(ImplName, ImplScript)
  maestro.loggingUtil.unregisterImplementation(ImplName)
  ```

  c. Implement methods such as `monitor`, `unmonitor`, and `registerPluginLogtype` in your custom code.

  Each custom logging service must provide a python script with a method (`monitor`, `unmonitor`, `registerPluginLogtype`) implementations. These methods are started after the implementation is registered.

- Monitoring service for plug-ins

  Plug-ins provide monitoring operations that collect and display deployment metrics for resource usage and performance at the virtual machine, middleware, and application levels. You can configure and register collectors for custom plug-in specific metrics at run time. Then, you can apply metadata to define the presentation of the monitoring metrics in the Virtual Application Console deployment panel.

6.5.2 Deploying and managing virtual applications

To build a virtual application, you can use a virtual application pattern or a virtual application template. You can use the virtual application template that is associated with a virtual application pattern to start building an application. After the virtual application is deployed, it becomes a virtual application instance.

You can deploy a virtual application by using a virtual application pattern or a virtual application template.

Virtual application pattern

For more information about deploying virtual application by using virtual application patterns, see 6.5, “Plug-in environment setup and creation of custom patterns” on page 286.

Virtual application template

Virtual application templates are designed for more flexibility and properties can be changed during deployment. Log in to the PureApplication System console and complete the following steps:

1. Click **Catalog** → **Select virtual application template**. Click the **Deploy** icon.

   After the virtual application is deployed, you can see it under the virtual application instances pane. To view virtual instances, click **Instances** → **Virtual applications**, then perform any of the following tasks:

   – To stop a virtual application instance: Select **Virtual application** → **Stop**.

   – To start a virtual application instance: Select **Virtual application** → **Start**.

   – To redeploy a virtual application: Select **Virtual application** → **Deploy** icon in virtual application builder pane.

   – To remove a stopped application: Select **Virtual application** → **Delete** icon.

2. Secure the virtual application.

   There are various security levels that are available to secure virtual application instances. Security can be set at the component level or at the instance level. The following security options are available:

   – User permissions: You can protect the cloud environment by applying various security roles. For more information, see this website:

     http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/index.jsp?topic=%2Fcom.ibm.ipas.doc%2Fiwd%2Faac_user_permissions.html

   – Securing web applications with secure sockets layer (SSL): Virtual instances that are based on web application patterns can be secured by using SSL certificates. For more information, see this website:

     http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/index.jsp?topic=%2Fcom.ibm.ipas.doc%2Fiwd%2Fapt_ssliwd.html

   – Configuring Secure Shell (SSH) key-based access: Configure SSH key-based access so that you can connect directly to virtual machines for troubleshooting and maintenance. For more information, see this website:

     http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/index.jsp?topic=%2Fcom.ibm.ipas.doc%2Fiwd%2Fapt_sshiwd.html

   – Lightweight Third-Party Authentication (LTPA) keys for web applications: Manage LTPA keys for a virtual application instance that is based on the web application pattern. For more information, see this website:

     http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/index.jsp?topic=%2Fcom.ibm.ipas.doc%2Fiwd%2Fapt_ltpakey.html

7

Integrating PureData for Transaction

This chapter describes the integration of PureData for Transaction with PureApplication System. This integration helps you to use the databases from PureData for Transactions.

The following topics are covered in this chapter:

- Advantages of integrating PureData for Transaction with PureApplication System
- Prerequisites for integrating the PureData for Transaction with PureApplication
- Registration of PureData for Transaction with PureApplication System
- Deploying pureScale Database from PureData for Transaction
- Deploying PureApplication System middleware

© Copyright IBM Corp. 2013. All rights reserved.

7.1 Advantages of integrating PureData for Transaction with PureApplication System

In this section, we describe the advantages of the use of PureData for Transaction system with PureApplication System. This integration provides you with rapid flexible data services solutions, data availability, on-demand resource allocation, and consolidation of database servers.

Integrating these systems includes the following advantages:

- Mission Critical Data

  IBM PureApplication System and IBM PureData for Transaction integration enables your business to deploy optimized and continuously available mission-critical databases when you use these databases in IBM PureApplication System middleware applications.

- High availability of transactional data for IBM PureApplication Systems System

  PureData for Transactions can provide built-in database clusters, which spread across multiple cluster nodes. DB2 pureScale® provides unlimited capacity, load balancing, and application transparency. This feature helps you keep your data available.

  IBM PureApplication System, as the solution front end, and IBM PureData System, for the back end, work as a combined platform. IBM PureApplication System provides a high availability (HA) proxy server, application server, and caching server for applications that are deployed in this platform. IBM PureData System provides a highly available database server and storage resources for the platform.

- Speed

  IBM PureApplication System is a highly efficient platform for applications deployment. When you use IBM PureApplication System and PureData system for transaction together, the user or customer can specify the requirements, such as when applications demand higher levels of scalability, availability, and performance. This solution allows the customer to easily and dynamically increase performance.

- On-demand resource allocation

  IBM PureApplication System is a complete system platform of hardware and middleware. The user or customer drops applications into the system's preconfigured middleware engine that includes IBM DB2 database and IBM WebSphere Application Server. You can dynamically adjust to demand spikes and reallocate the system resources automatically. Multiple applications and database patterns are provided (as part of the system) for optimally deploying and managing applications and resources in virtual and cloud environments.

- Consolidating the Database Server for Applications

  These systems provide database server consolidation. You do not have to be concerned about managing individual database servers. A single interface makes it easier to manage the databases that are used by many applications.

- Scalability

  This system integration also provides nondisruptive scalability. Administrators can add more nodes with no application changes required. The user or customer can start small and easily grow, with no need to over-purchase or over-provision.

7.2 Prerequisites for integrating the PureData for Transaction with PureApplication

In this section, we describe the prerequisites to integrate the PureData for Transaction system with the PureApplication System.

Integration requires the following prerequisites:

- Configure the connection between PureData System for Transaction and PureApplication System.
- Configure network communication between PureData System for Transaction and PureApplication System.
- Configure the trusted certificate between PureData System for Transaction and PureApplication System.
- Register the PureData System for Transaction with PureApplication System.
- Configure administrator access to the PureApplication System Workload Console.
- Configure existing user ID and password information for PureData System for Transaction. To use PureData System for Transaction, users must have the security administration role.

7.3 Registration of PureData for Transaction with PureApplication System

This section describes how to register the PureData System for Transaction with PureApplication System. Registration of PureData for Transaction with PureApplication System provides communication between PureData System for Transaction and PureApplication System.

Complete the following steps to register the PureData System for Transaction with PureApplication System:

1. Click the **Workload Console** tab at the top of the Welcome page to open the workload console on the PureApplication System.
2. Click **System**.
3. Click **PureData System registration**.

4. Complete the following fields, as shown in Figure 7-1:
   a. Enter the IP address of PureData System for Transaction. This IP address is the same address that is used to access the PureData System for Transactions system console.
   b. Enter the user name for this system; it must have the security administrator role.
   c. Enter a password that is associated with the user name.
   d. Click **Register**.

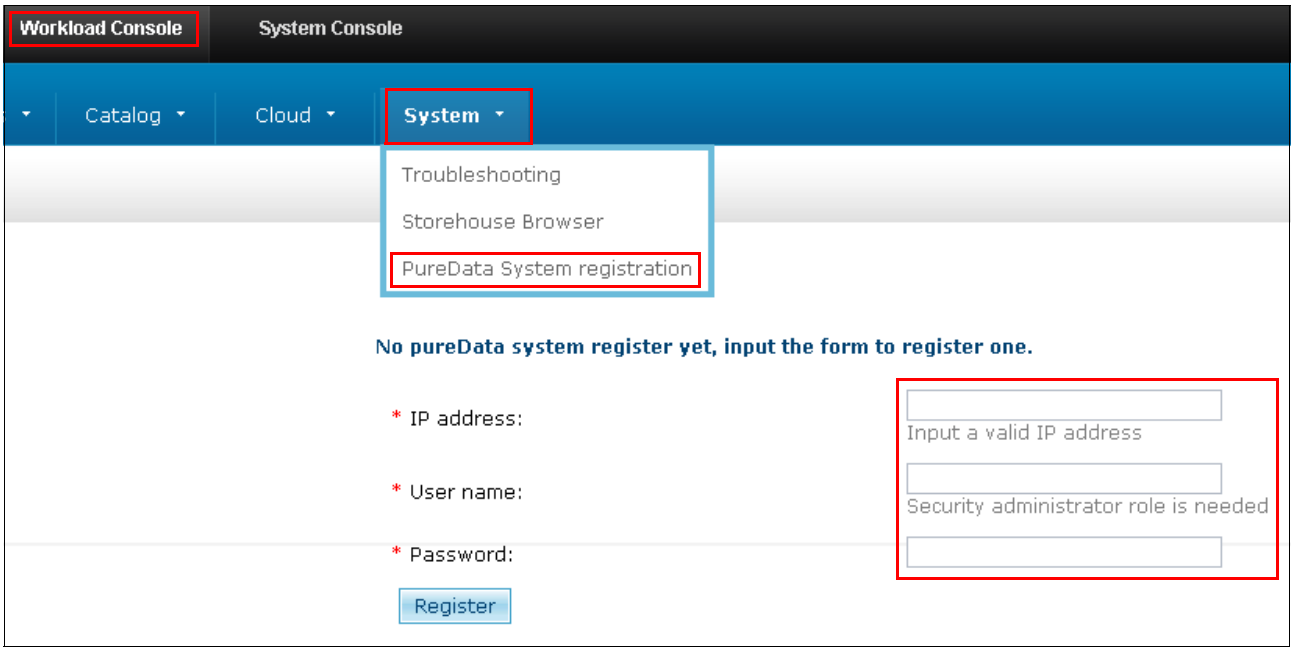


Figure 7-1 PureData system registration

Important: Only one PureData System for Transactions system can be registered with a PureApplication System at any time.

7.4 Deploying pureScale Database from PureData for Transaction

This section describes the steps to deploy the new pureScale database on PureApplication System.

You can create pureScale database patterns by using PureApplication System from PureData for Transaction. A new pureScale database is provisioned by specifying the provisioning options and deploying from PureData for Transaction.

Complete the following steps to provision a new pureScale database:

1. Click **Workload Console**.
2. From the Workload Console page expand the section Working with databases and click **Deploy database,** as shown in Figure 7-2.

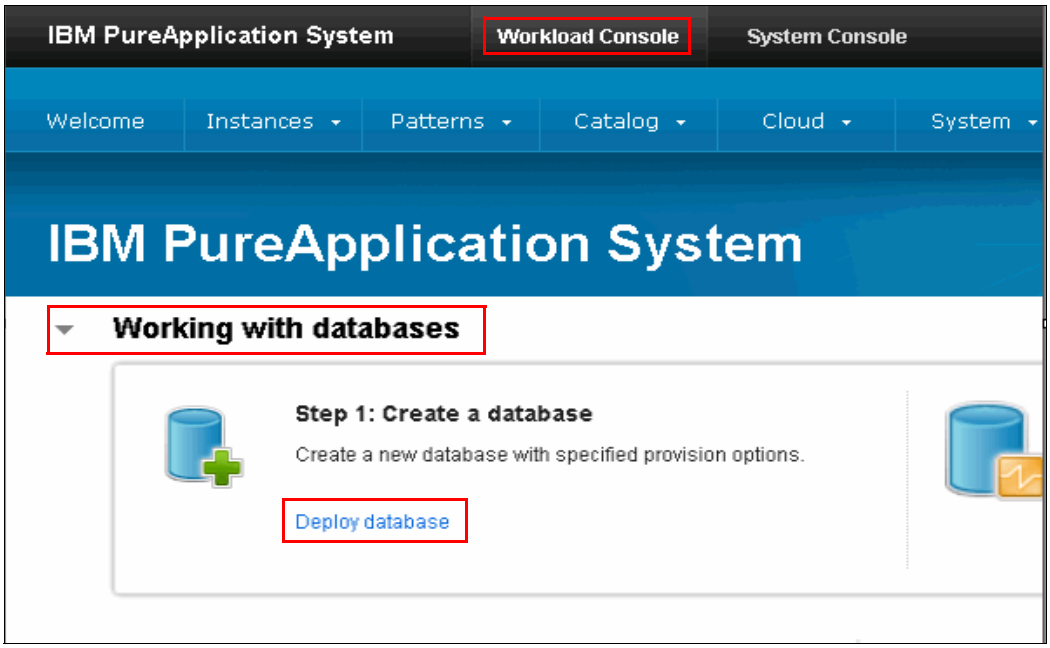


Figure 7-2 Deploy database

3. Configure and enter values into the following fields:
   - Database Name: Enter the database name

     The database name must be eight characters, begin with an alphabetical character, and contain an underscore (_). It does not have to include other special characters (such as ?, !, *, or @).
   - Description of Database: A description of the database is helpful but not mandatory.
   - Availability/Scalability: in this field, there are two options, standard and high. Select **High**, which is used for pureScale database from PureData for Transaction.

> **Important:** You must have a DB2 pureScale instance that is deployed on the PureData System for Transactions before you create the database.

- Source field: You must select **Apply a default database workload standard**.

  After you select **Apply a default database workload standard,** a default online transaction processing (OLTP) database is selected automatically, which is used for OLTP. The database is optimized for transactional applications.
- Default user: Enter the user name and in the Password field, enter the password for the default user.
- Database size: Enter the size of the database. In this example, 20-Gb size is used.
- Database compatibility mode: The default compatibility mode of database is DB2. You can choose the available database in list, for example DB2 (default) and Oracle.
- Database Version: You can select the version from the drop-down list. We used DB2 v10.1 for Linux. The version list is the database version and the supported operating system platform, for example, DB2 V10.1 for Linux.
- Database Level: You can select the database version and Fix Pack level with supported operating system from the drop-down list. We used DB2 V10.1FixPack Level 2 for Linux
- DB2 pureScale instance: Select the DB2 pureScale existing instance name, which is already created on PureData for Transaction system.
- Schema file: This field is optional. You can browse for your schema file if you choose to use one.
- Select **Advanced Options** to expand the field, and from the drop-down list you can select options for fields of Pagesize, Territory, Code set, and Collating sequence
- Pagesize: You can choose the page size for the database from the drop-down list. We used 4 as pagesize.
- Territory: Select the nation from the drop-down list, which can be used for database. We used US territory.
- Code Set: Select the standard character code set. We used UTF-8 code set.
- Collating Sequence: You can select the standard collating sequence of data source from drop-down list. We used SYSTEM.

4. Click **OK** after you enter the information in all of the required fields.

Figure 7-3 shows the fields and configuration page for the database.

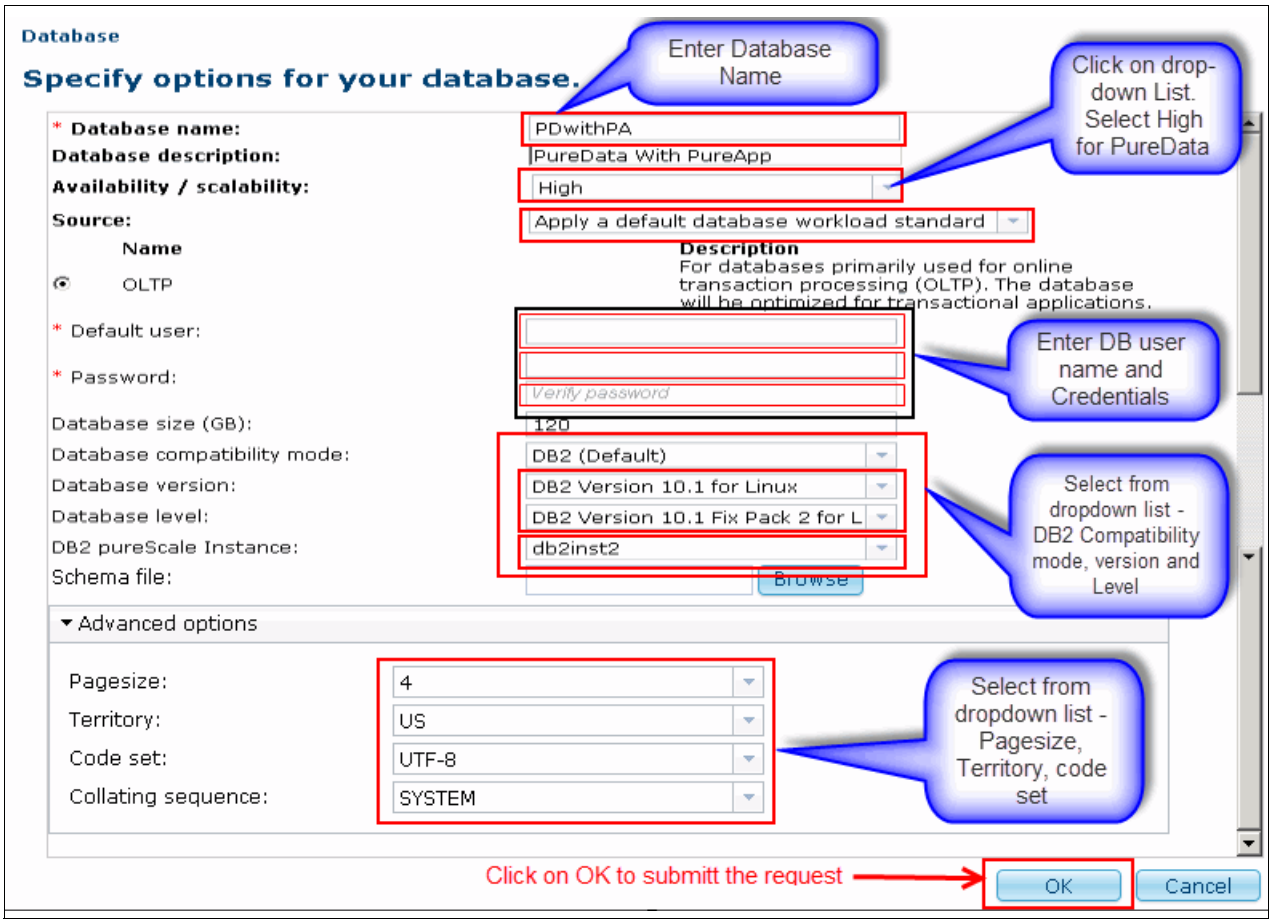


Figure 7-3 Deploy database fields

After you click **OK,** the database deployment process is started. You can see the status of the deployed database request, as shown in Figure 7-4.

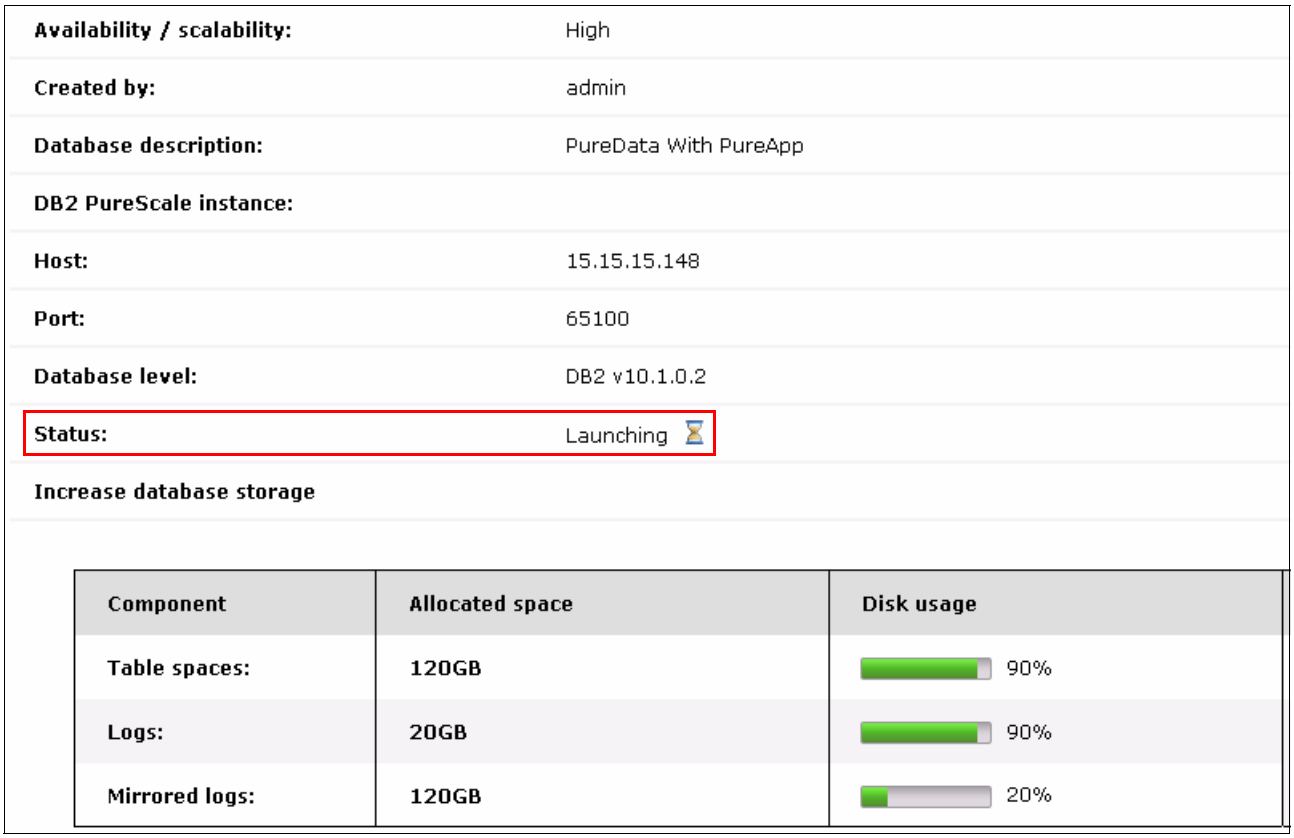


Figure 7-4 Deploy database status in Launching state

When the database instance status changes from Launching to Running, the following information about the database instance is shown:

- DB instance name
- DB instance IP address
- Port number

Click **log** to view the log.

Figure 7-5 shows the database instance display.

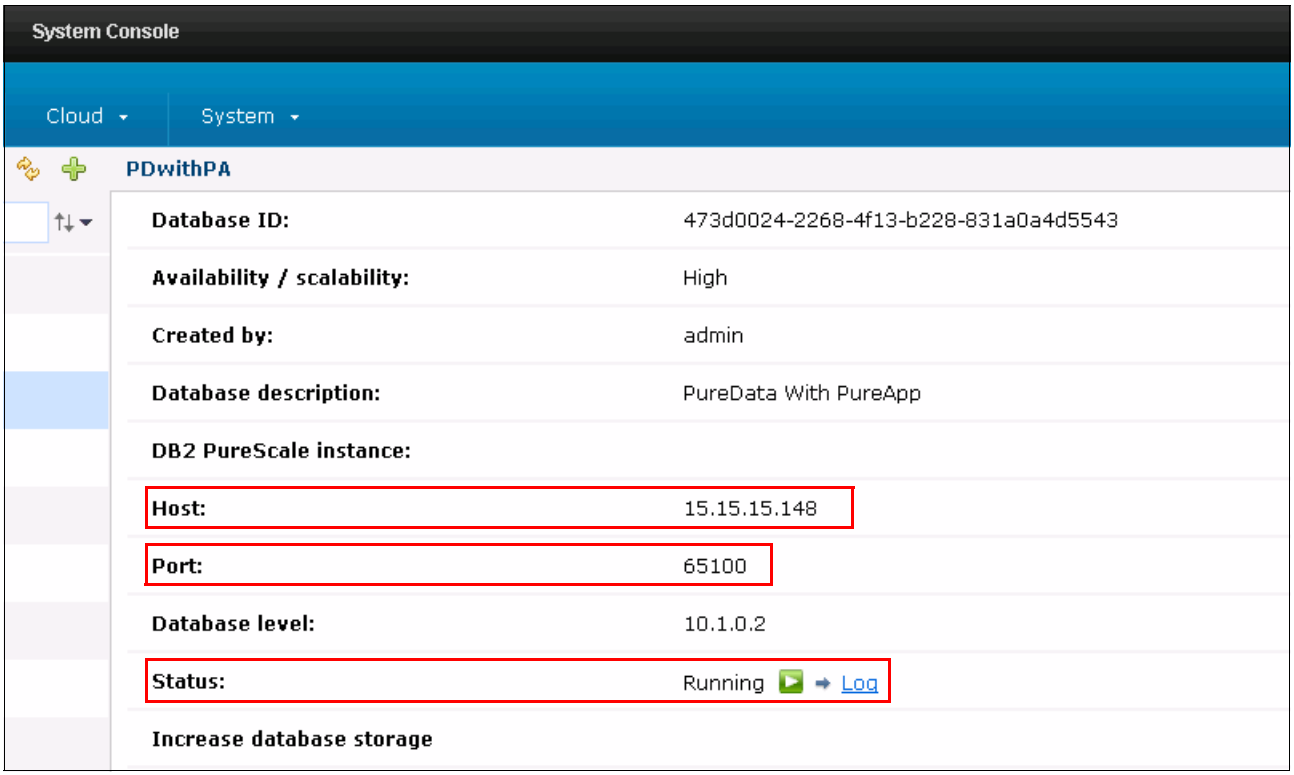


Figure 7-5 Database instance information

7.5 Deploying PureApplication System middleware

This section describes deploying an application from PureApplication System that fully employs the PureData for Transaction System. In this case, we are using existing data capabilities and we can constantly access the data. We are using the existing database instance, which was created in 7.4, “Deploying pureScale Database from PureData for Transaction” on page 313.

7.5.1 Creation of a virtual application pattern

To use the PureData for Transaction System, you must create a virtual application pattern to make the data services robust and more scalable. Complete the following steps to create the virtual application:

1. Click **Workload Console**.
2. On the Welcome page, click the **Patterns** drop-down list and select **Virtual Applications**.
3. Select the virtual application pattern type that you want to create, as shown in Figure 7-6. In this scenario, `Web Application Pattern type 2.0` is selected and used.

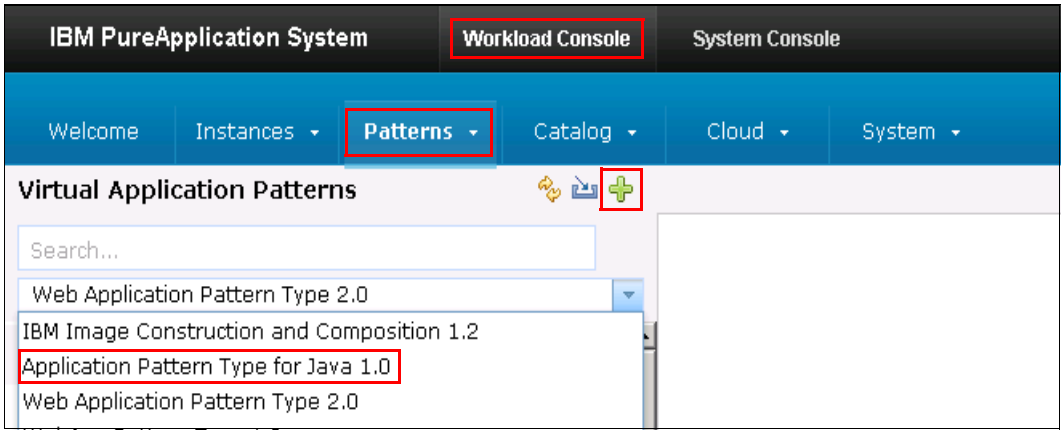


Figure 7-6 Application pattern selection

4. Click **+,** as shown in Figure 7-6, to start building new your virtual application.

5. When you click **+**, the Create Application window opens and you can create the customized application. Select **Blank application** and click **Start Building**, as shown in Figure 7-7.

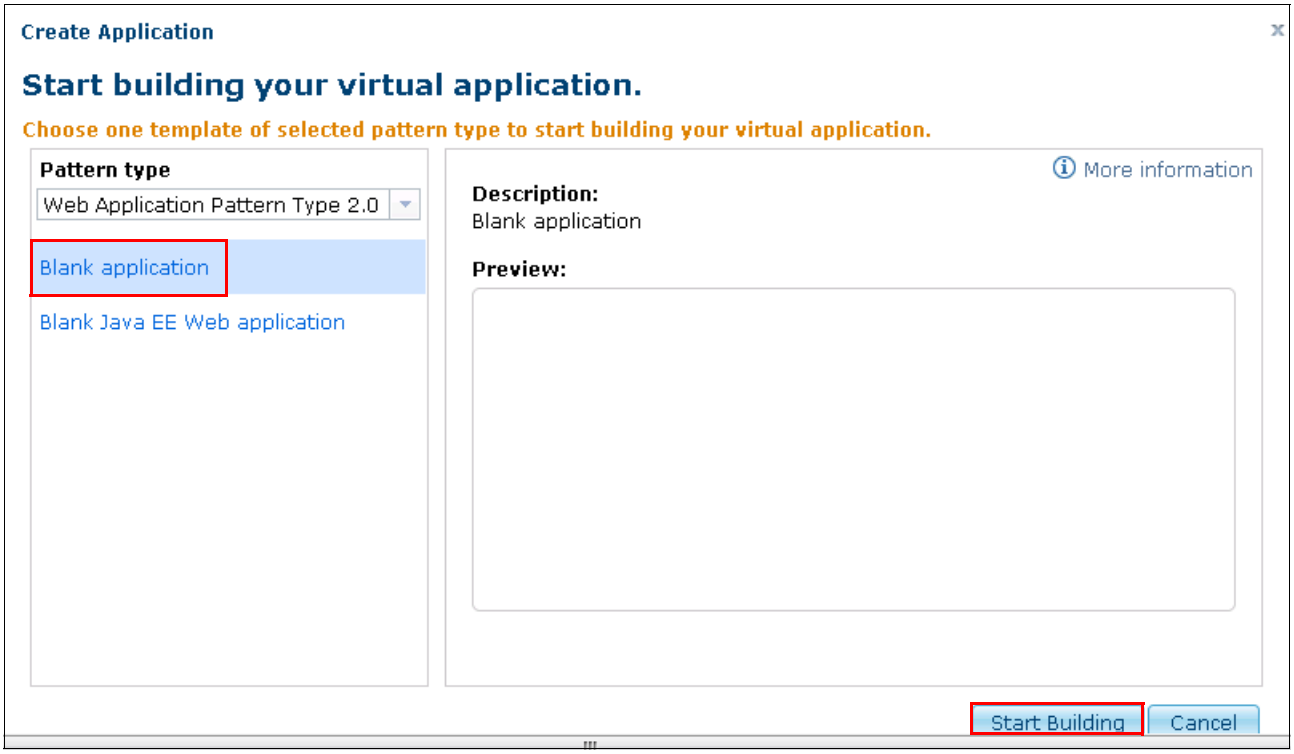


Figure 7-7 Pane to start building your virtual application

6. When you click **Start Building**, the Virtual Application Builder opens in another browser window, is shown Figure 7-8 on page 320. In this scenario, the decision is made to create a virtual application for Enterprise Application for WebSphere Application Server. The scenario also associates an existing DB2 database from PureData for Transaction system to that virtual application. Complete the following steps to re-create this scenario:

   a. Expand the Application components and drag the Enterprise Application component to the right side and onto the canvas, as shown in Figure 7-8 on page 320.

   b. Expand the Database components and drag the Existing Database component for DB2 to the right side and onto the canvas, as shown in Figure 7-8 on page 320.

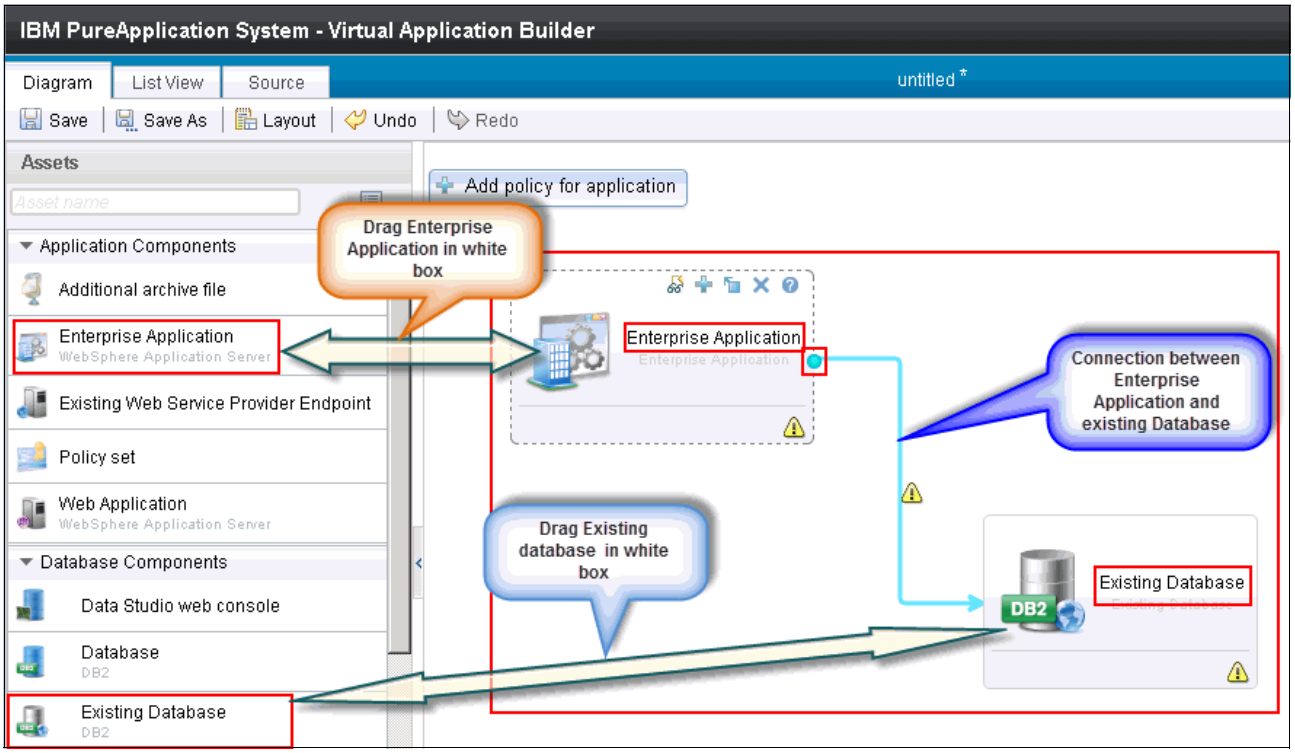


Figure 7-8 Virtual Application Builder window

c. Create the connection between the Enterprise Application component and the Existing Database component. The connection is made by dragging the blue dot (on the side of the component) from one component over to the other component, as shown in Figure 7-9.

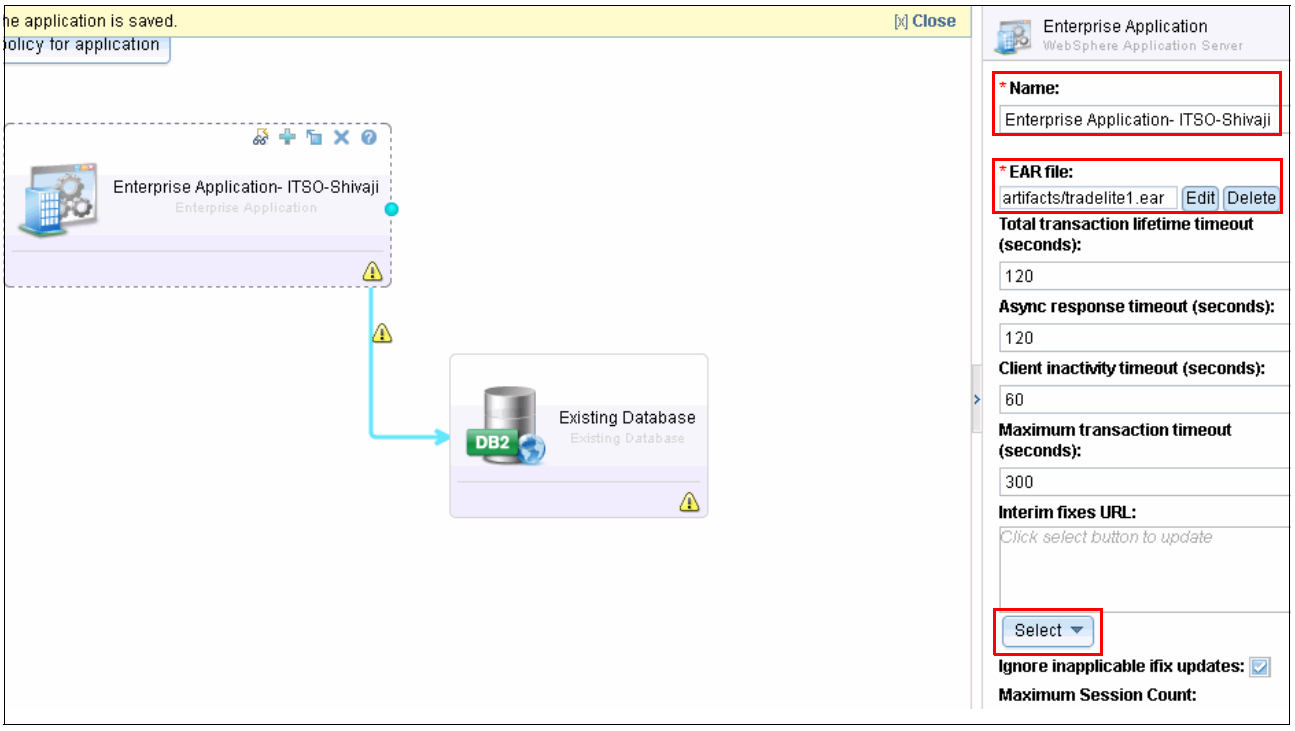


Figure 7-9 Application Components: Enterprise Application

d. Click **Application components - Enterprise Application**. After you click the component, the right side of the window gives you options for that component. You can then specify a name for the Enterprise Application, apply an EAR file, and select the interim fix pack from drop-down lists, as shown in Figure 7-9 on page 320.

e. Click the **Database component - Existing database** option. After you click this component, the right side of the window gives you options for that component. You can specify information in the following fields, as shown in Figure 7-10:

- Name: The name for the Existing database.
- Existing Database Name: The name for the database that you want to use from PureData for Transaction.
- The Server Hostname or IP Address, Server Port Number, User name, and Password fields are self-explanatory.

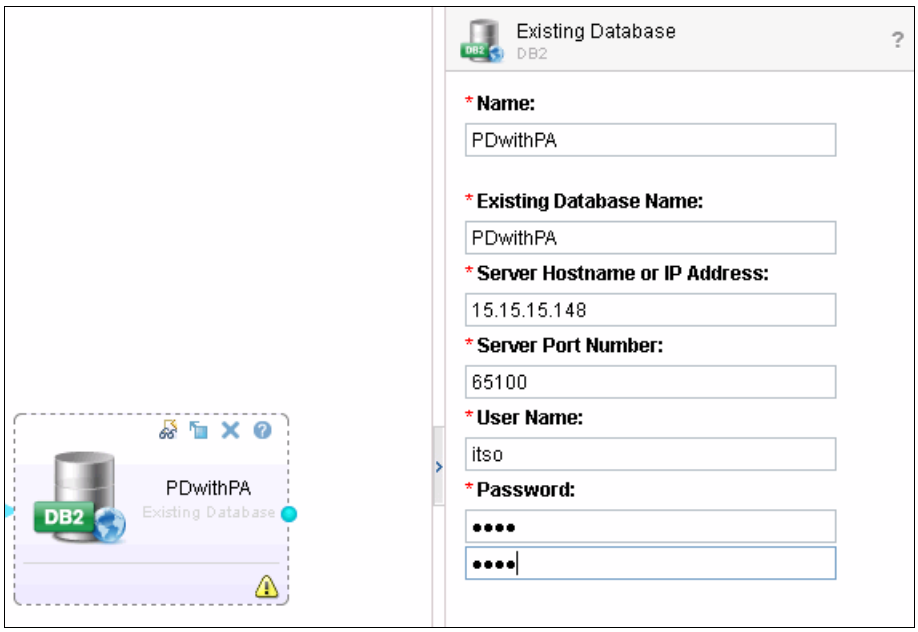


Figure 7-10 Database Components: Existing Database entry fields

f. Click the **connection bar** (which is made when you are connecting the components by using the blue dots). After you click the bar, the right side of the window gives you the options that are associated with the bar. You must specify the JNDI name of Data Source, as shown in Figure 7-11.

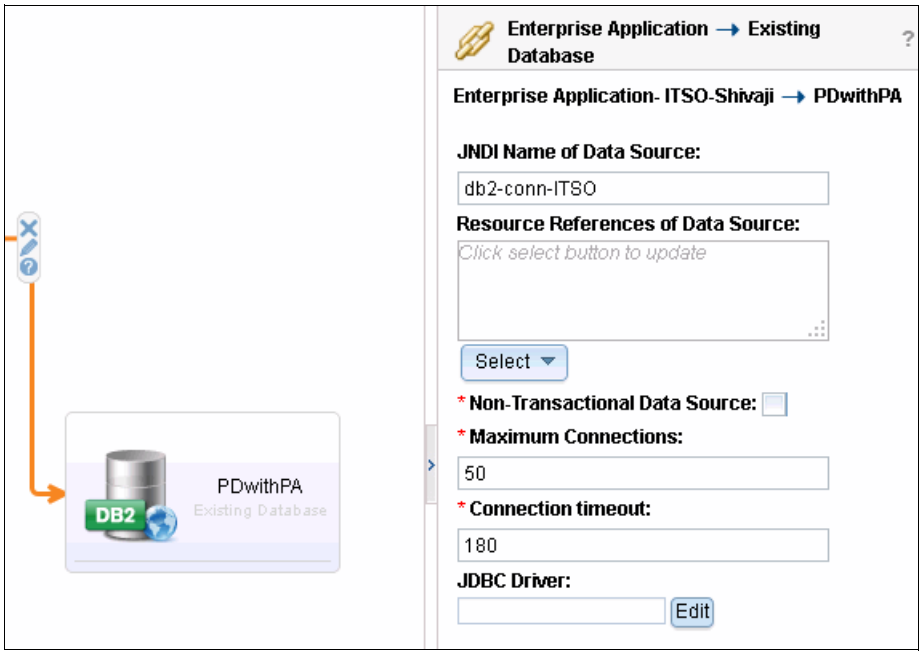


Figure 7-11 Field input for connection bar

7. Click **Save** to save the virtual application, as shown in Figure 7-12.

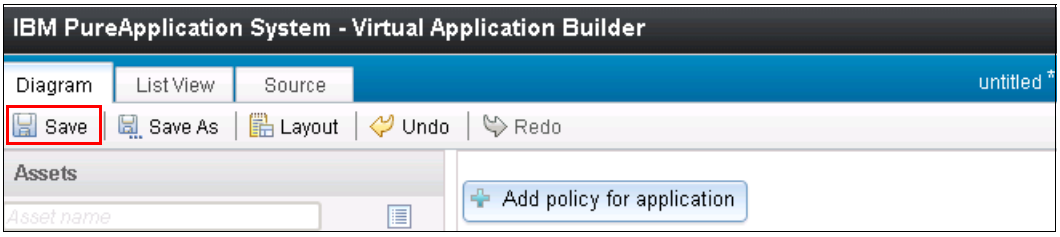


Figure 7-12 Save the virtual application

8. Specify the name of the custom virtual application in the Save As window, as shown in Figure 7-13.

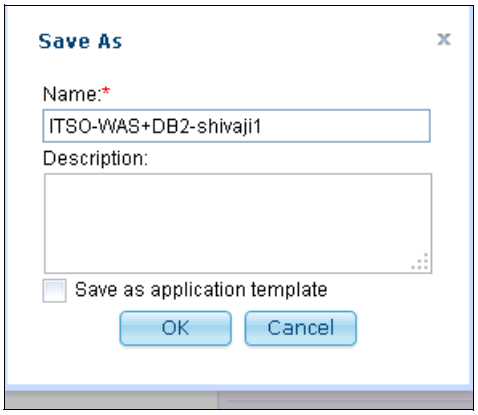


Figure 7-13 Saving your custom application

7.5.2 Deploying the virtual application

This section describes the process that is used to deploy the application that was created in section 7.5.1, “Creation of a virtual application pattern” on page 318. This scenario selects to deploy a virtual application for Web Application pattern type 2.0.

Complete the following steps to deploy the virtual application to deploy a new virtual application pattern:

1. Browse to Welcome page of the Workload Console panel.
2. Click **Instances**.
3. Select **Virtual Applications** to deploy a new virtual application pattern**,** as shown in Figure 7-14.

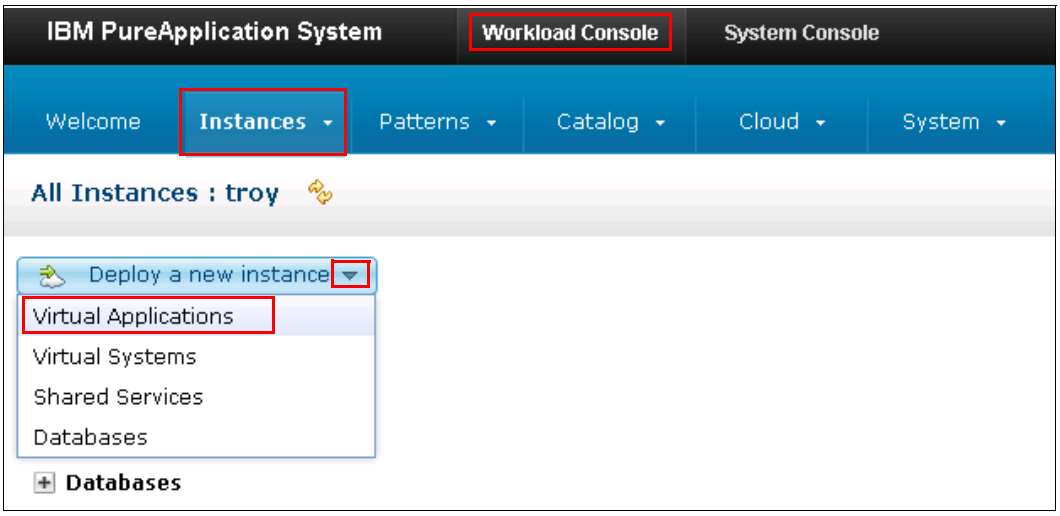


Figure 7-14 Workload Console panel

4. When you Select **Virtual Applications** in same pane, it opens Deploy Virtual Application pane. Select **Pattern Type** → **Web Application pattern type 2.0,** as shown in Figure 7-15.

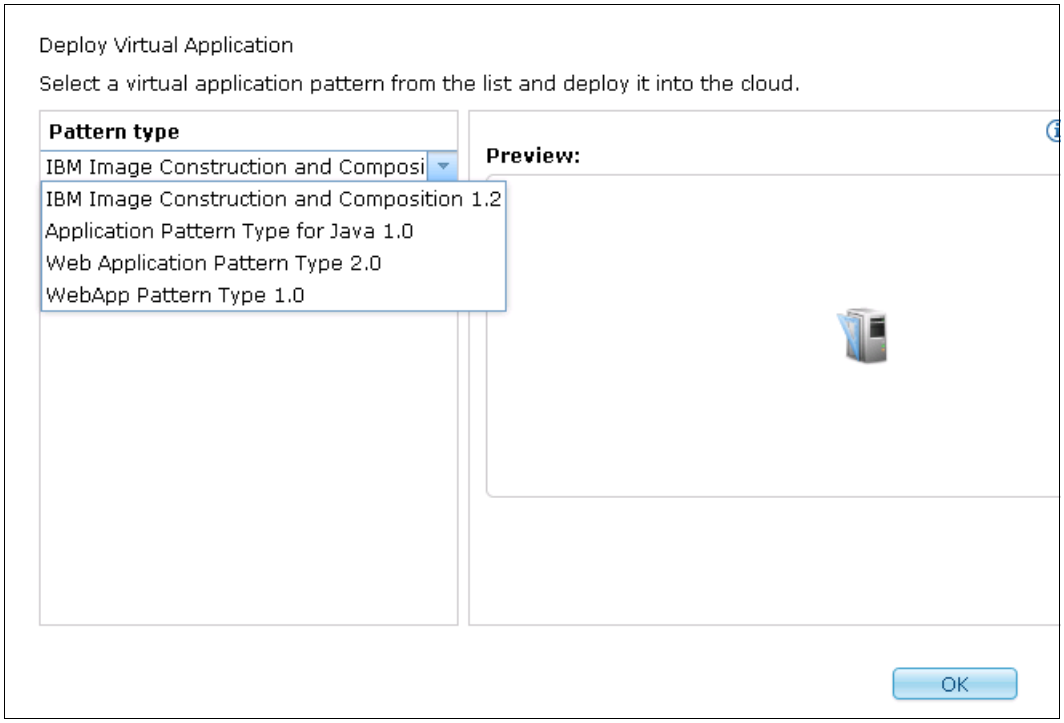


Figure 7-15 Deploy Virtual Application pane

5. Click **Ok**.

6. A pattern list is displayed, as shown in Figure 7-16. The pattern that was created in 7.5.1, “Creation of a virtual application pattern” on page 318 is available in the list.

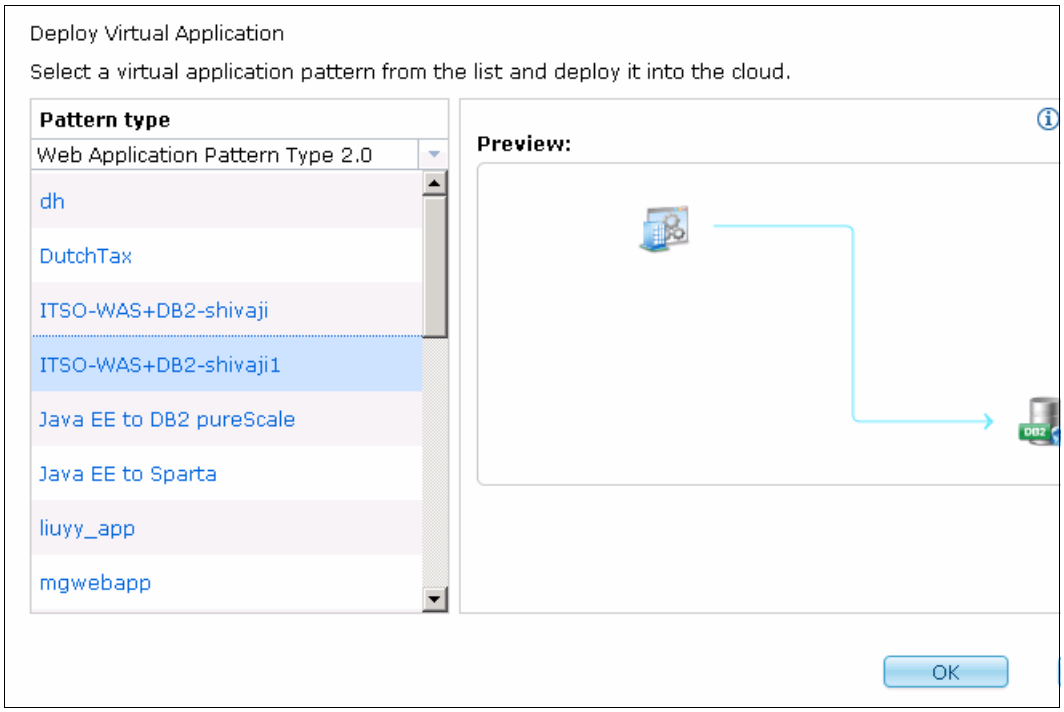


Figure 7-16 Pattern type list

7. Select the pattern that was created in the scenario or the pattern that you created when you complete the scenario, as shown in Figure 7-16.
8. Click **OK**.

9. The Deploy Virtual Application pane is displayed, as shown in Figure 7-17.

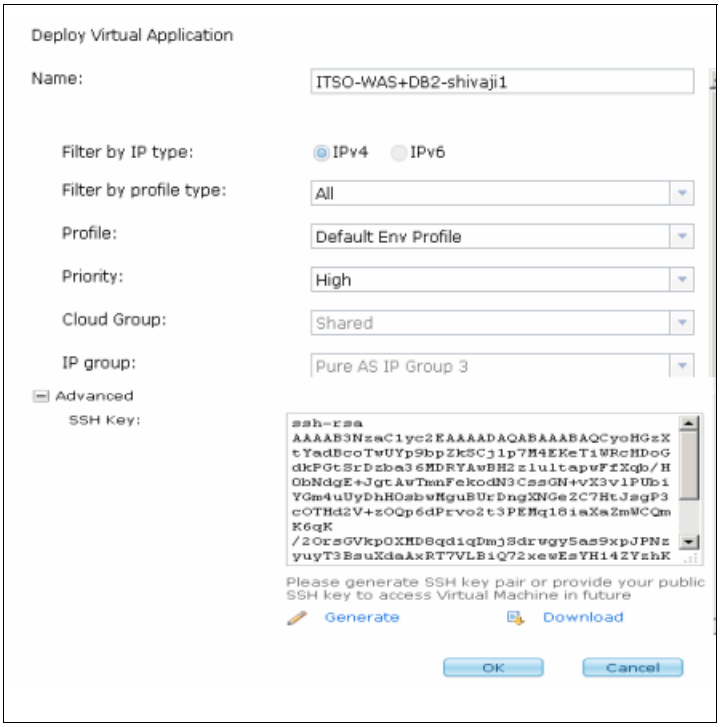


Figure 7-17 Deploy Virtual Application pane

10. Specify the fields for Name and Filter by IP type (IPv4 or IPv6).
11. Select the profile type from the Filter by profile type drop-down list, as shown in Figure 7-18.

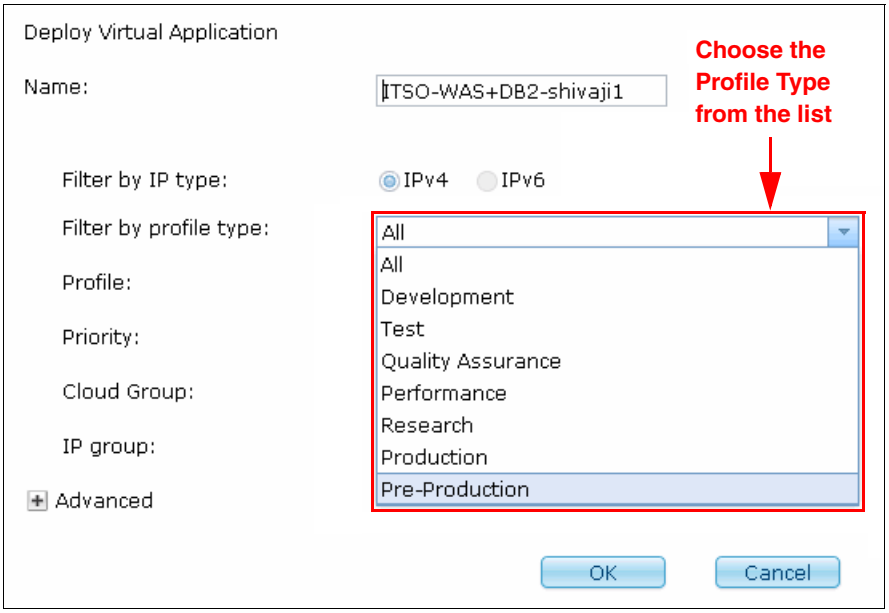


Figure 7-18 Filter by profile type list

12. Select the profile from the Profile drop-down list, as shown in Figure 7-19.

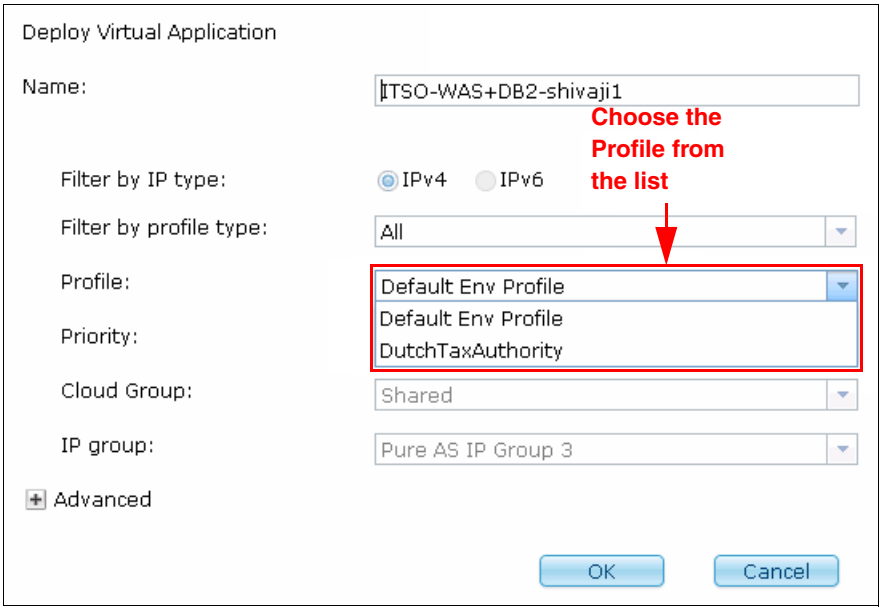


Figure 7-19 Profile list

13. Select the deployment priority from the Priority drop-down list, as shown in Figure 7-20.

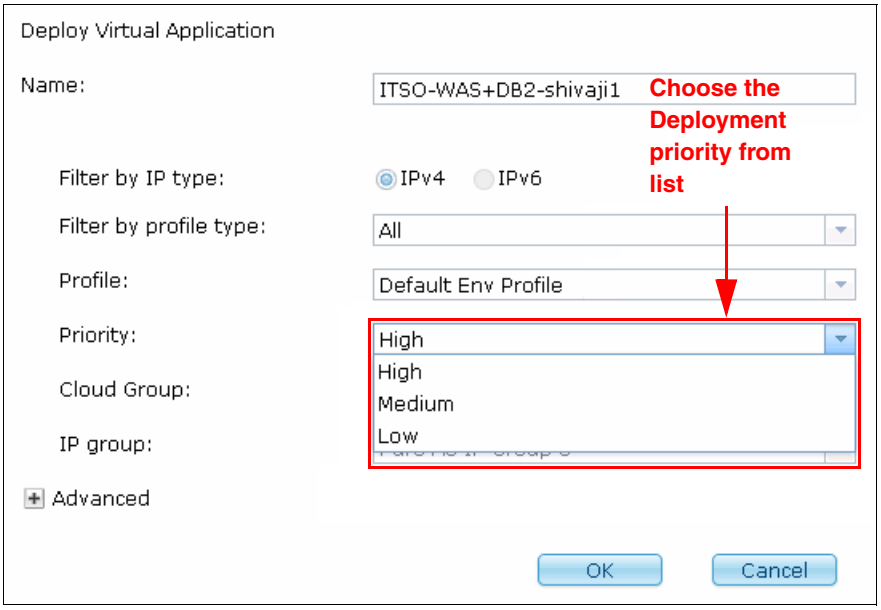


Figure 7-20 Deployment priority list

14. Select the Cloud group from the drop-down list. In this scenario, there is only one cloud group (Shared), as shown in Figure 7-17 on page 326.
15. Select the IP group from drop-down list. In this scenario, there is only one IP group (Pure AS IP Group 3), as shown in Figure 7-17 on page 326.
16. Expand the Advanced option to generate or download the SSH key, as shown in Figure 7-17 on page 326.
17. Click **OK** to deploy the virtual application, as shown in Figure 7-17 on page 326.

The deployment status of the virtual application can be checked by using one of the following methods:

- In section 7.5.2, “Deploying the virtual application” on page 323 step 17, when you click OK, you see another pane in which you can click **clicking here**, as shown in Figure 7-21.

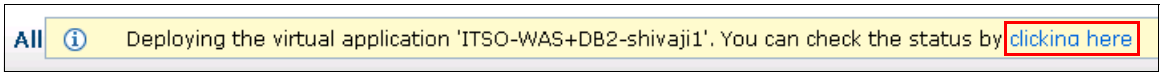


Figure 7-21 Check the status of deployment for the virtual application

- Click **Workload Console**. On the Welcome page of the Workload Console, click **Welcome Page → Instances,** then select the **Virtual application** to check its status, as shown in Figure 7-22.

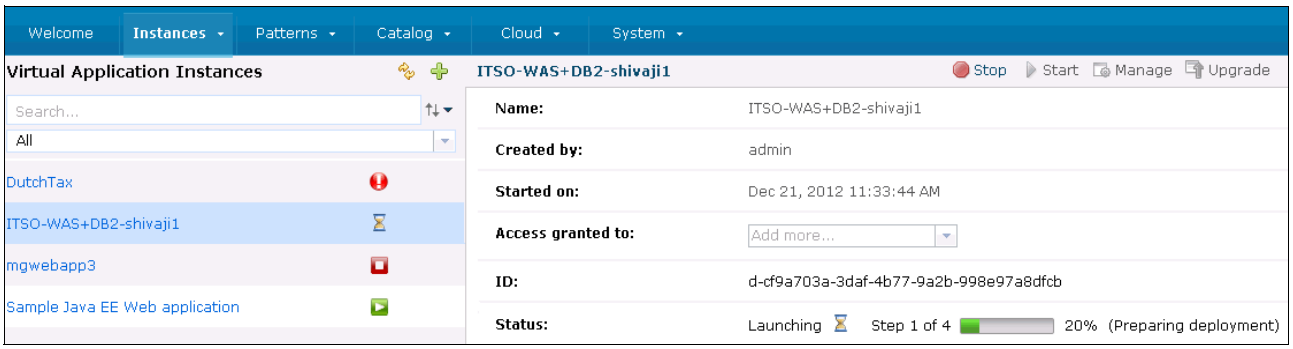


Figure 7-22 Check the status of deployment for the virtual application

Click **Workload Console**. On the Welcome page of the Workload Console**,** click **Welcome Page → Instances,** then select the **Virtual application**. When the deployment of the virtual application is completed and successful, a green arrow status displays, as shown in Figure 7-23.

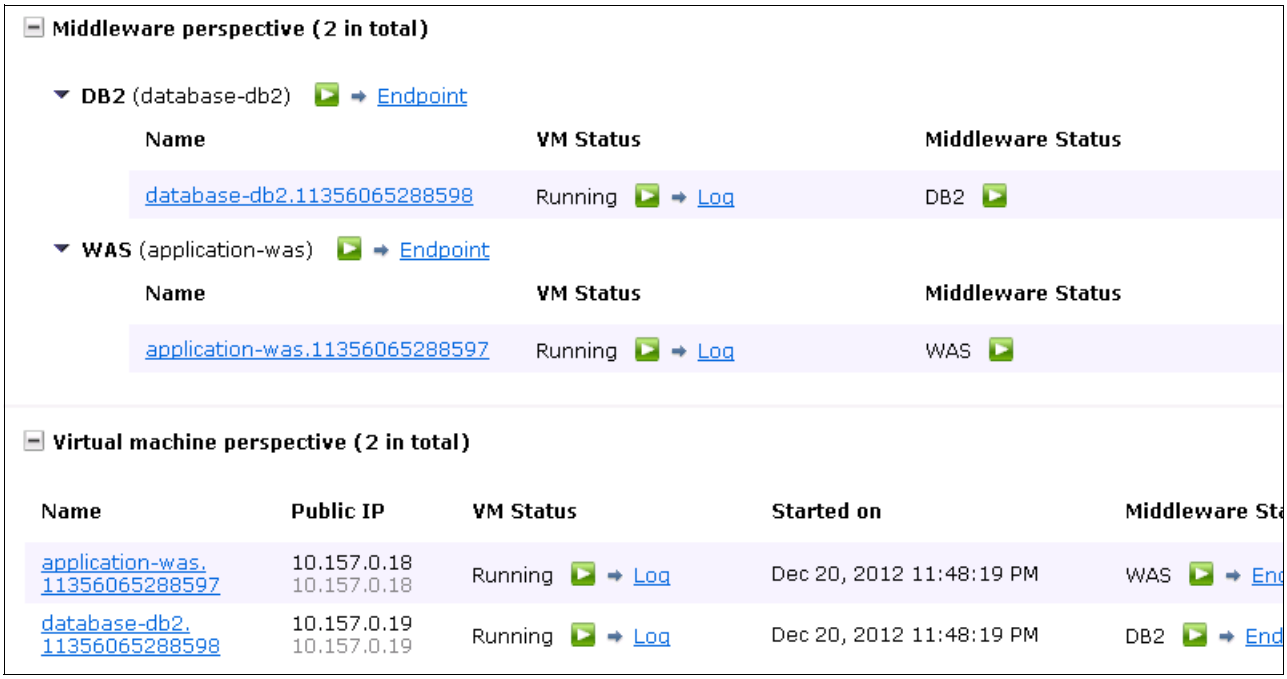


Figure 7-23 Successful deployment of a virtual application

8

Troubleshooting PureApplication System Environment

This chapter describes how to troubleshoot IBM PureApplication System. The concentration is on problems that can affect the user or customer in the following environments:

- PureApplication systems
- Guest virtual machine (VM)
- Virtual applications
- IBM support

Troubleshooting is a technique to solve the problem. The problem within a PureApplication System environment can take many forms, including PureApplication System, poor performance, virtual system and application, unavailability, and unexpected results. The first step in resolving a problem is to isolate and understand it.

The following topics are covered in this chapter:

- Troubleshooting overview
- Consoles available for PureApplication System troubleshooting
- Command-line interface
- Product logs and tracing
- Image Construction and Composition Tool
- Virtual image synchronization problems

© Copyright IBM Corp. 2013. All rights reserved.

8.1 Troubleshooting overview

IT infrastructure environments have advanced and there are many levels of redundancy that provide continuous availability of software, systems, and infrastructure. Even with all the advances, hardware and software components can unexpectedly stop functioning. When a failure occurs, it is necessary to collect as much information as possible to assist with problem determination and root cause analysis (RCA). IBM PureApplication System provides a set of troubleshooting functions to enhance the examination of failed hardware or software components. It automatically captures all logs from management software, firmware, and hardware in a rack.

For a user to troubleshoot a problem on PureApplication System, they must be assigned correct roles and permissions. To perform troubleshooting steps, the user needs the hardware administration role with permission to manage hardware resources (full permission).

PureApplication System automatically creates several log files to determine the problems in hardware and software components. Of the many logs that are created, the following types of logs are included:

- System Logs

  System logs consist of underlying hardware, firmware, system components, and management nodes. System logs can be accessed by using the PureApplication System console.

- Workload Logs

  Workload logs consist of operating system logs from virtual machines, middleware (such as WebSphere, DB2, and scripts), logs from within VMs that are related to middleware runtime components, and configuration scripts. Workload logs can be accessed by using the PureApplication Workload console.

Figure 8-1 shows the troubleshooting breakdown of System and Workload logs.

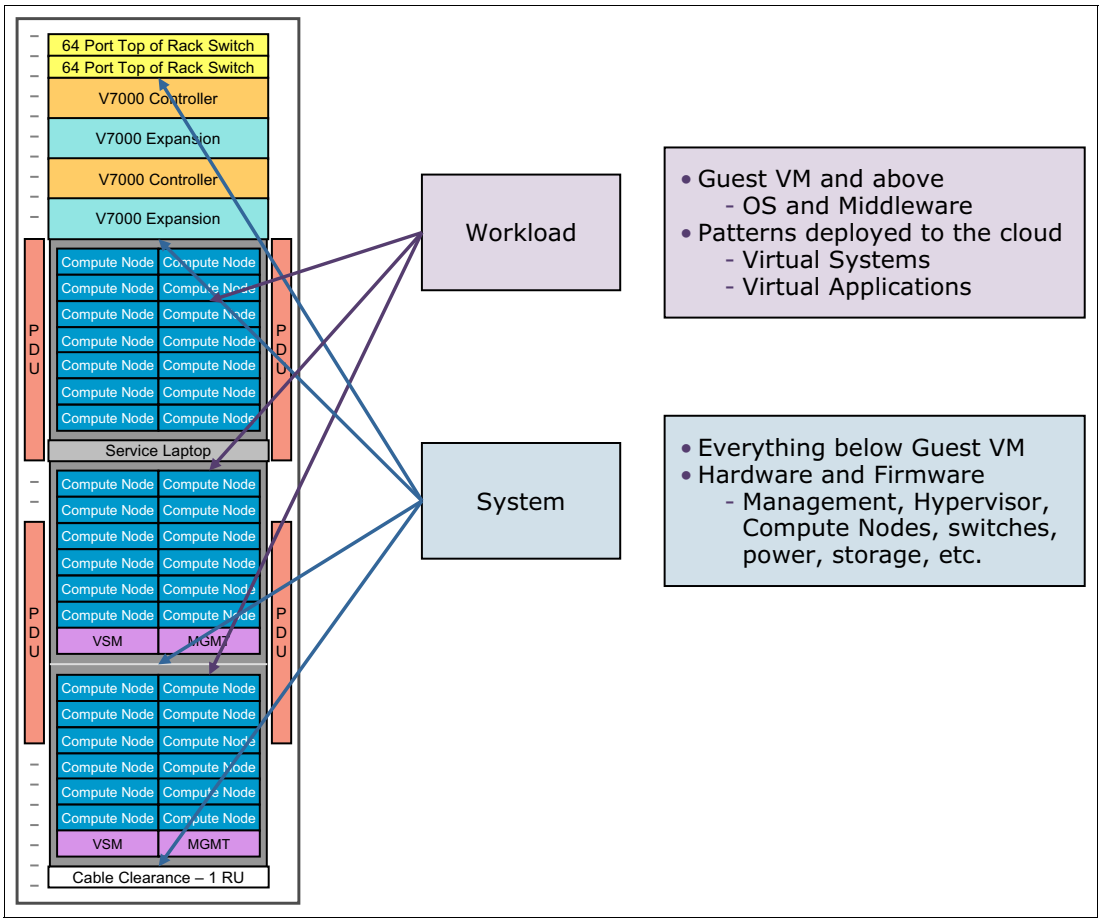


Figure 8-1 Troubleshooting breakdown logs

8.2 Consoles available for PureApplication System troubleshooting

The following sections describe the two consoles that are available for PureApplication System to troubleshoot a problem:

- System Console
- Workload Console

8.2.1 System Console

To troubleshoot system-related issues and diagnose problems, use the troubleshooting panel, as shown in Figure 8-2.

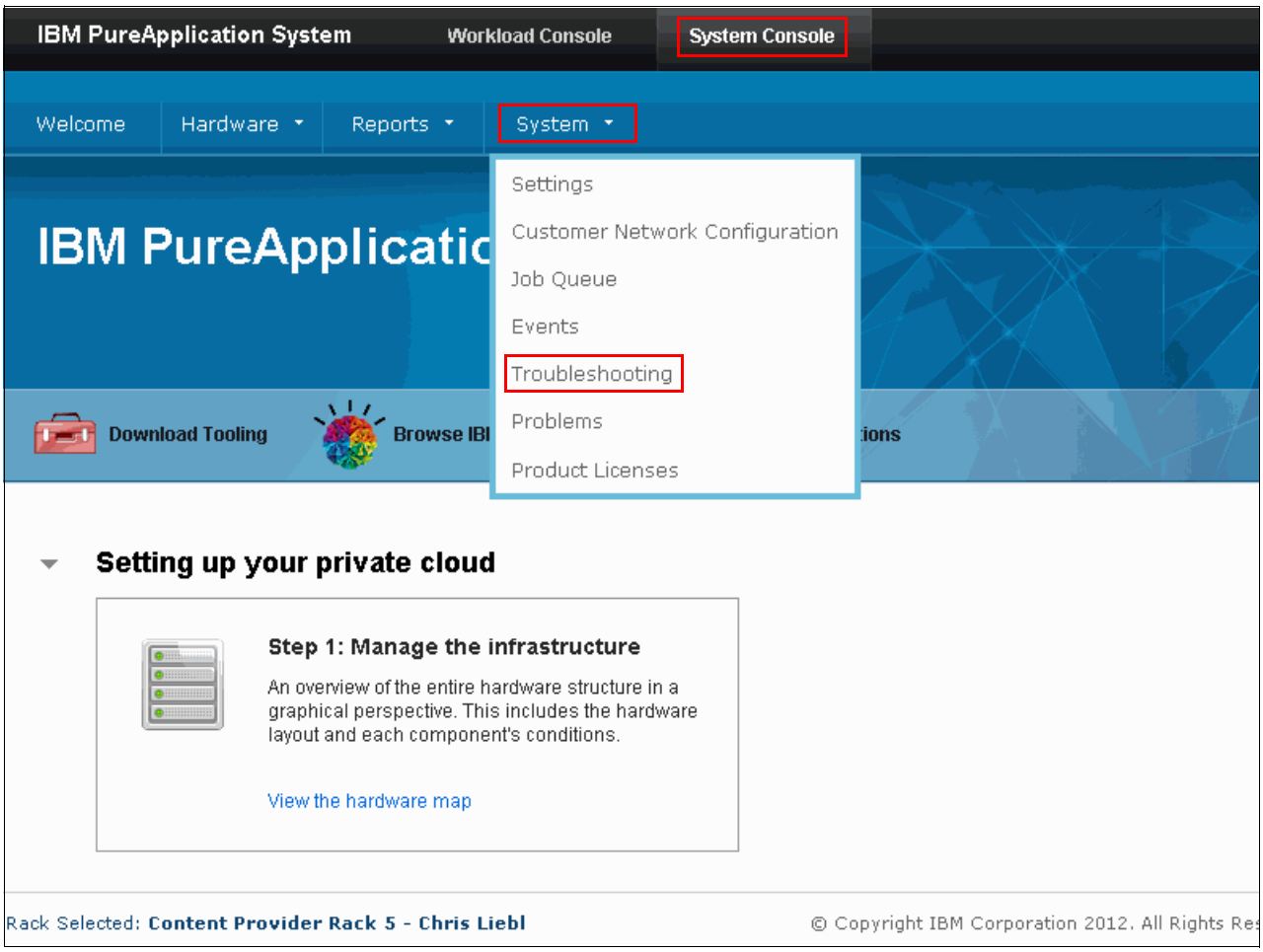


Figure 8-2 Troubleshooting panel

Enabling shell account

To troubleshoot the system features, you must enable the shell account to access the system's features. After you create the shell account, you can enter the **help** command in the shell to display the list of available commands.

Complete the following steps to create the shell account, as shown in Figure 8-2 on page 332.

1. Browse to **System Console** → **System** → **Troubleshooting**.
2. Expand the shell account.
3. Enter a password in the password of current user field.
4. Click **Create**.

System console user interface for user tasks

The system console user interface is used to access tasks that are related to troubleshooting information and actions.

There are several options for troubleshooting information, such as events, problems, and guided and service tasks, as shown in Figure 8-3.

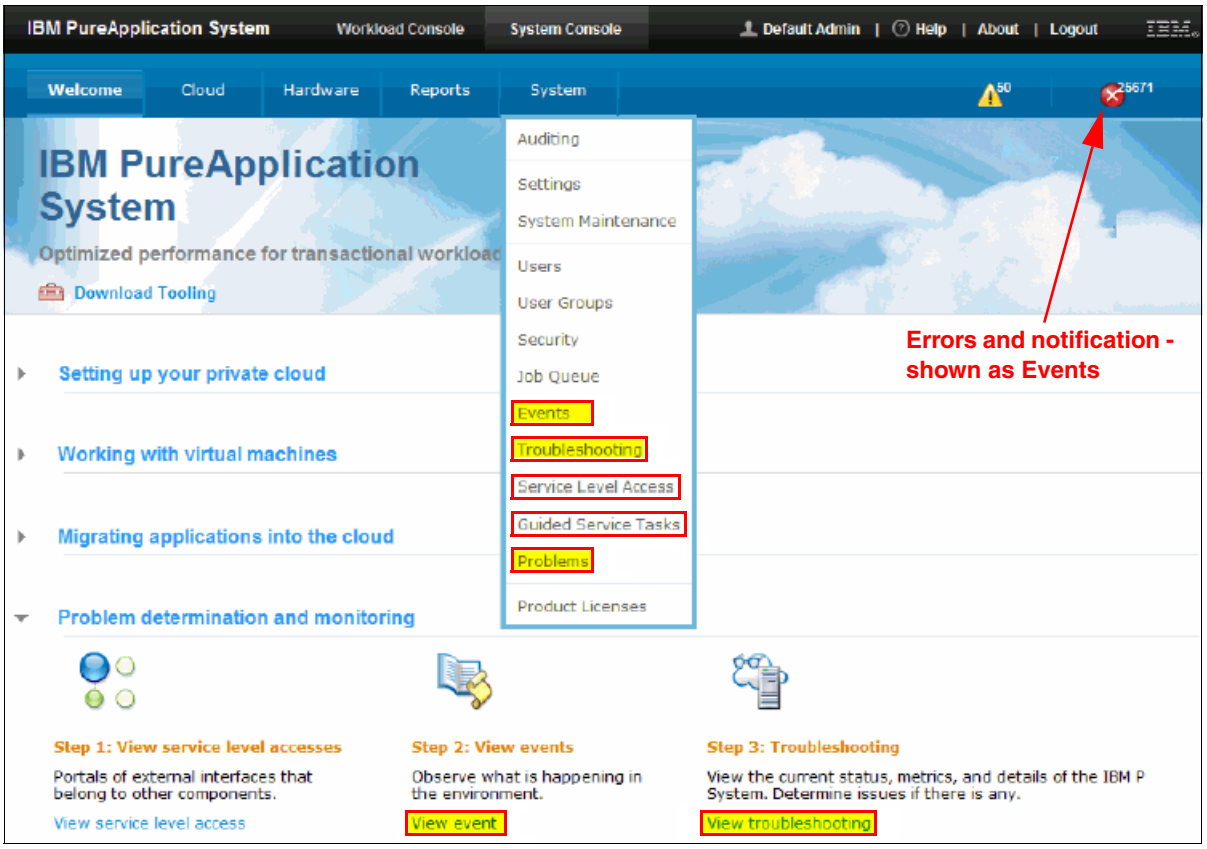


Figure 8-3 System console user interface for user tasks

The following options are available for troubleshooting:

- Events: A central view for events (errors and notifications) sent by different components.
- Troubleshooting: Contains system logs, vendor information, LEDs, and other information.
- Service Level Access: Access is restricted and is available only to IBM Customer Engineers. IBM Customer Engineers have access to complete all hardware maintenance and upgrades and updates to hardware firmware and system software.
- Guided Service Tasks: Access is restricted and is available only to IBM Customer Engineers. In Guided Services tasks, level access for IBM Customer Engineer provides tools to access advanced service level access panels (for access to underlying system panels).
- Problems: Contains a list of problems in the system for which the IBM PureApplication System customer might open a service request with the IBM support team.

System logs, collection logs, traces, and LED status

This section describes system logs, collection logs, traces, and LED status. It outlines the following features and important usage information:

- System logs: These logs come from underlying hardware components and system software.
  - The issues that are related to VM and include mainly hardware, system software, and storage networking.
  - Trace settings can be specified here as requested by from the IBM support team.
  - Actions, such as the collection of these logs, are accessed through System Logs, as shown in Figure 8-4.

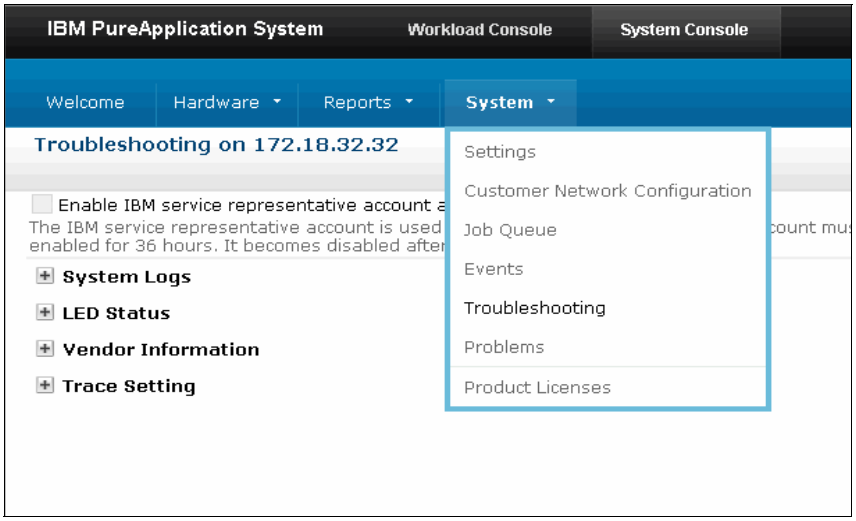


Figure 8-4 System logs pane

On the left side of the pane, System Logs has a + icon next to it. Click **+** to expand System Logs to access the function that are used to collect system logs, as shown in Figure 8-4.

Click **Collect System Logs** to access its options, as shown in Figure 8-5.

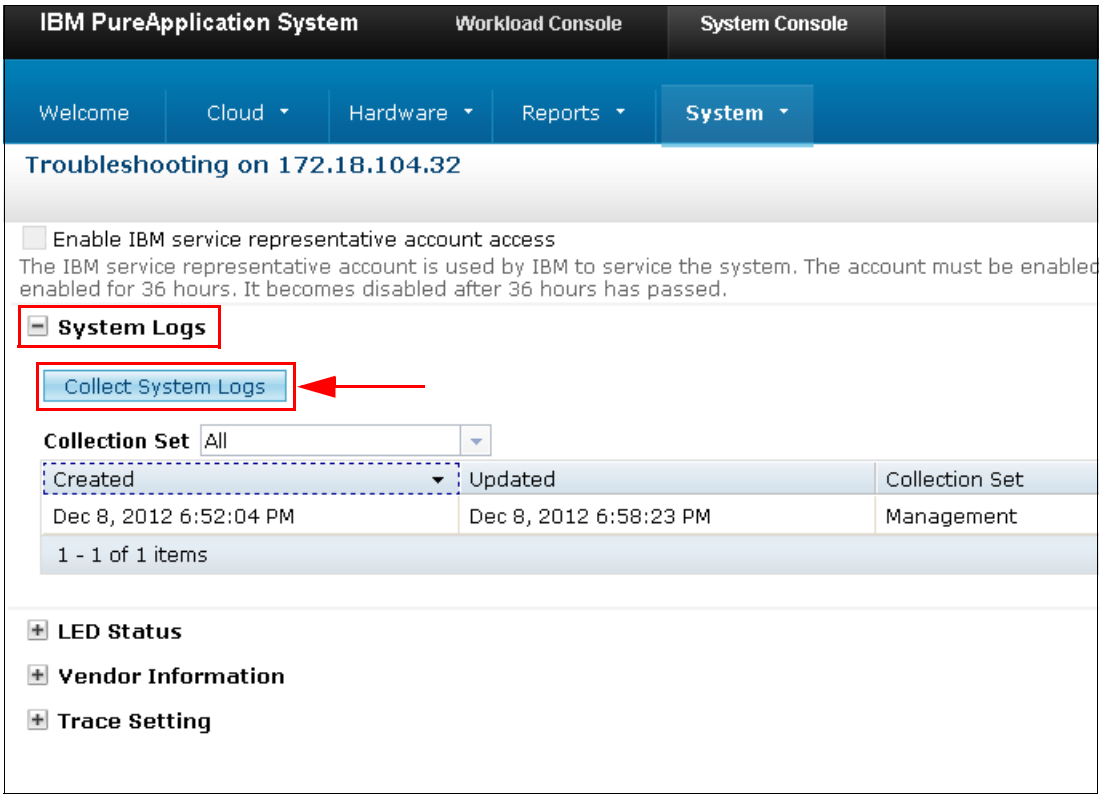


Figure 8-5 Collection System Logs

When the Request System Logs pane opens, select a collection that is set from the drop-down menu, as shown in Figure 8-6.

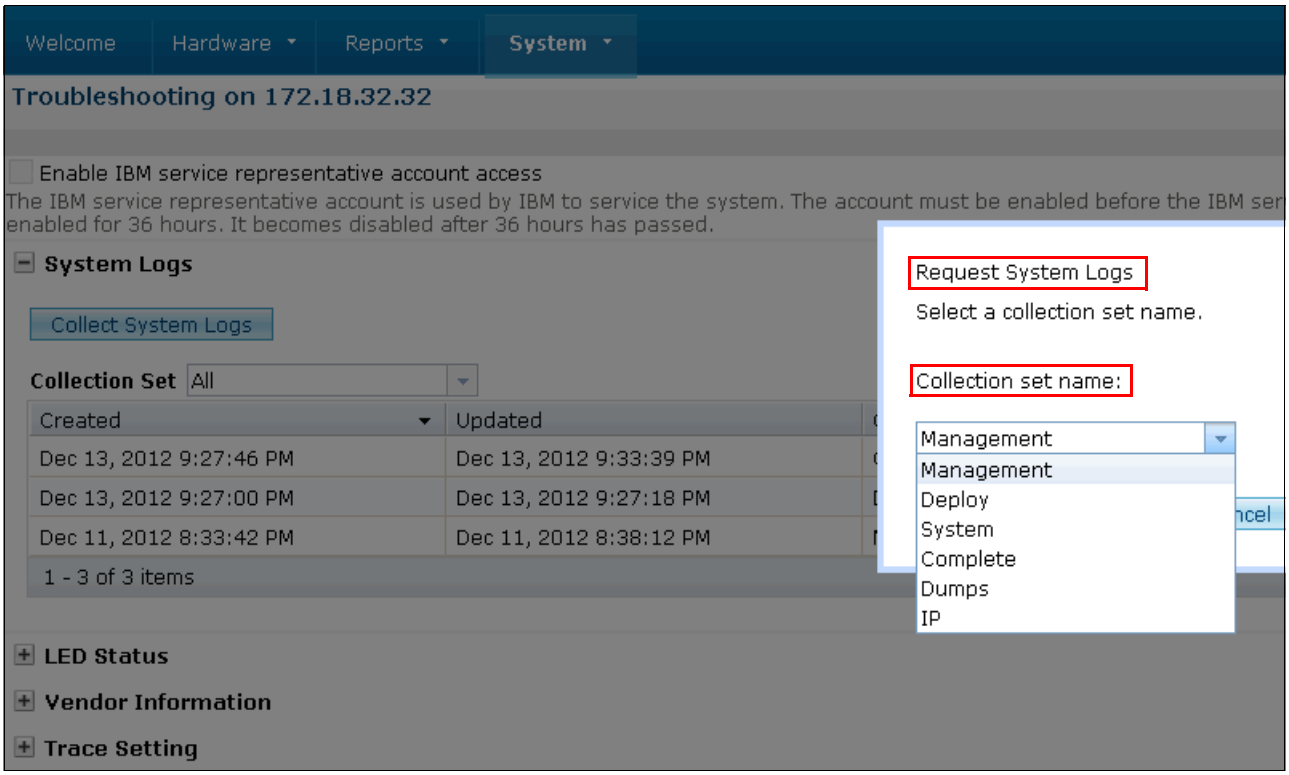


Figure 8-6 Request System Logs pane

The following capabilities are available for capturing system data for troubleshooting:

- Collection Logs:
  - Captures continuous collection logs and on-demand logs.
    - Continuous logs are gathered asynchronously. These logs rotate daily, weekly, and monthly to regulate disk usage.
    - On-Demand logs are collected when a user requests to extract the logs.
  - Supports compression and retrieval from either of two active PureApplication System Management nodes.
  - Client requested logs: These logs are collected into a single `.zip` file for the user and there is no built-in viewer of the log. The following options for logs are available from the user interface:
    - Management: Captures only PureApplication System software logs on the PureApplicaiton System management node.
    - Deploy: Latest version of workload console action logs.
    - System: The default logs that collect the management, deploy and syslogs from remote systems, and firmware syslogs.
    - Dumps: Contains the Java virtual machine dumps from PureApplication System node.
    - Complete: Collects system and historical deployment logs.
    - IP (IP address): Collects specific on-demand items from the IP, such as dumps and configuration.

An example of the collection set logs is shown in Figure 8-7.

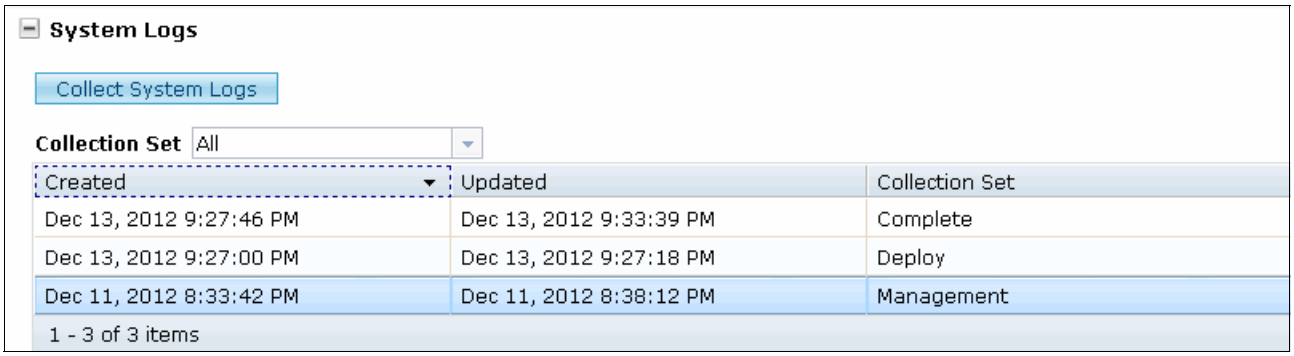


Figure 8-7 Collection set logs

- Trace Settings: Used to add details in the logs.

  There are multiple trace levels, such as OFF, SEVERE, WARNING, INFO, FINE, FINER, FINEST, and ALL, as shown in Figure 8-8.

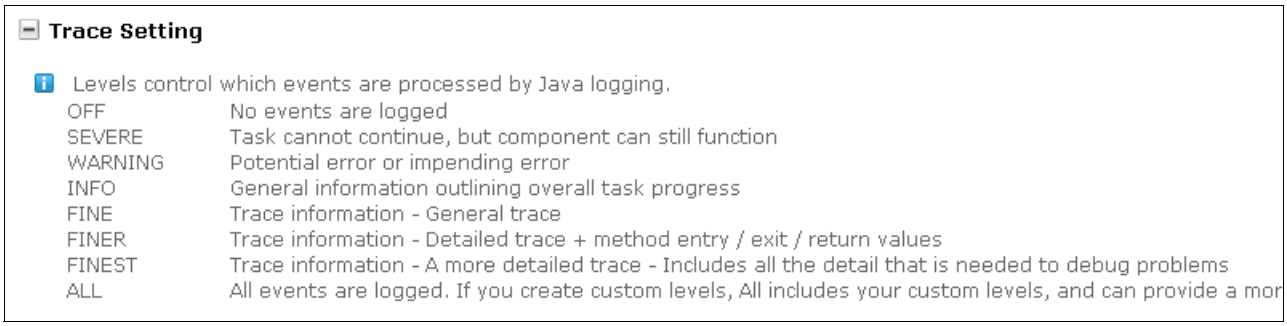


Figure 8-8 Trace settings

- LED Status

  LED status is ON or OFF. This status shows you whether a component is ON or OFF, as shown in Figure 8-9.

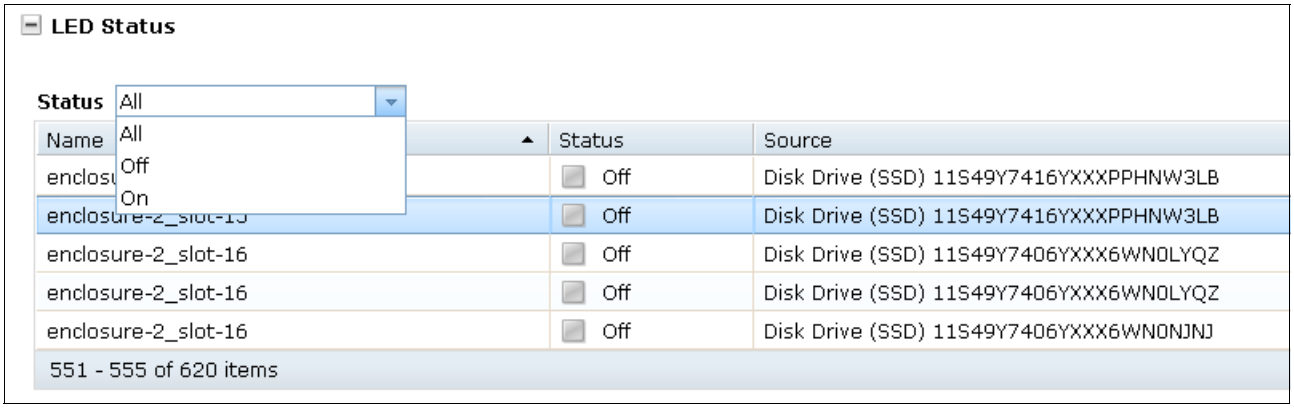


Figure 8-9 LED Status panel

- Vendor Information: This setting is an information only collection.

  Allows client or IBM SSR to see the information about a vendor, such as serial number and other details, as shown in Figure 8-10.

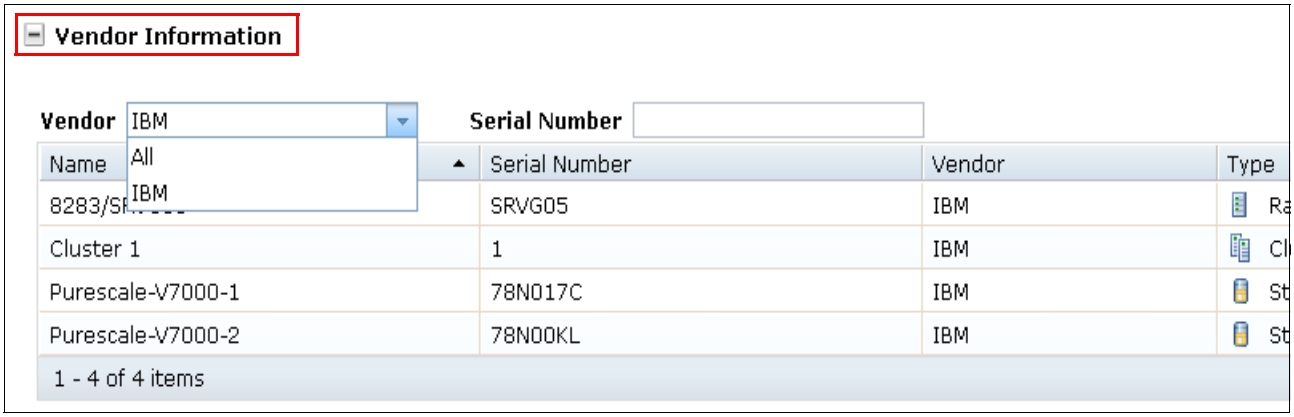


Figure 8-10 Vendor information

System events

System events generate the events of PureApplication System components. System events use Simple Network Management Protocol (SNMP) for generating these events for those system components. You can filter the system events that are based on type, severity, category, and time interval, as shown in Figure 8-11.

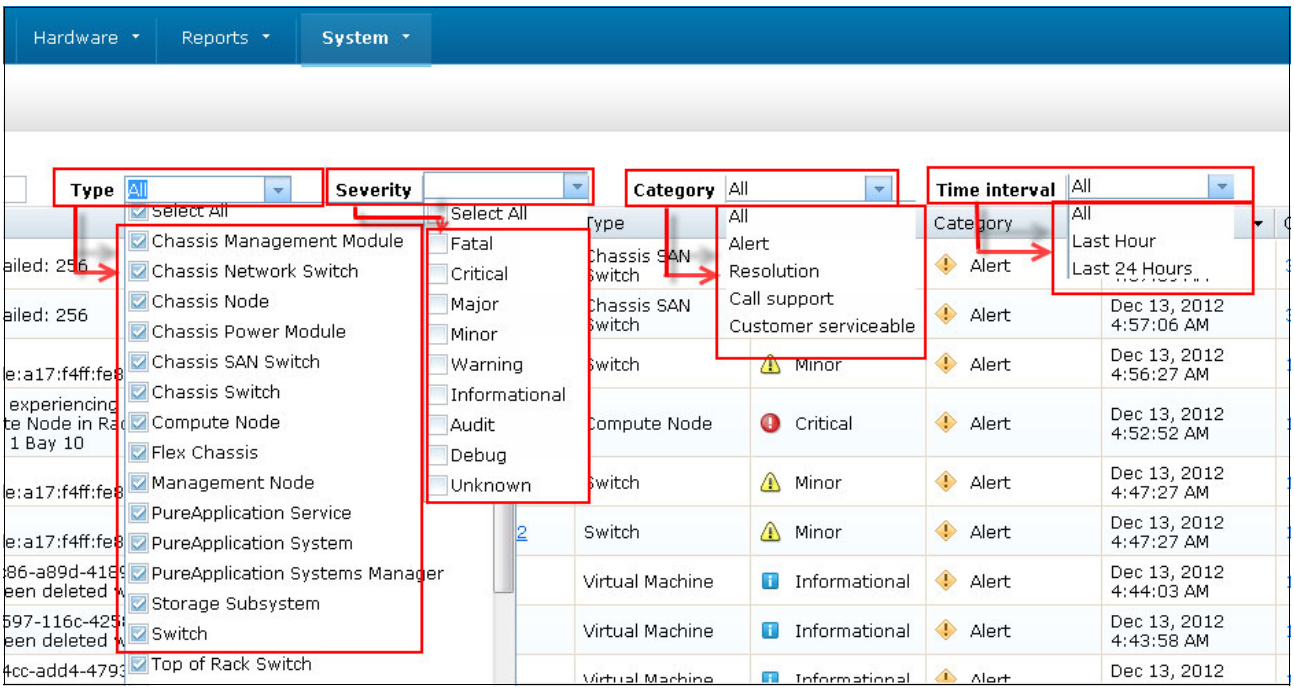


Figure 8-11 System events

Infrastructure map

The infrastructure map shows the entire rack and its components. This map contains important information to help you troubleshoot and with the performance of PureApplication System. There are two views to see the infrastructure map: Graphical and Tree.

The Graphical view is shown in Figure 8-12.

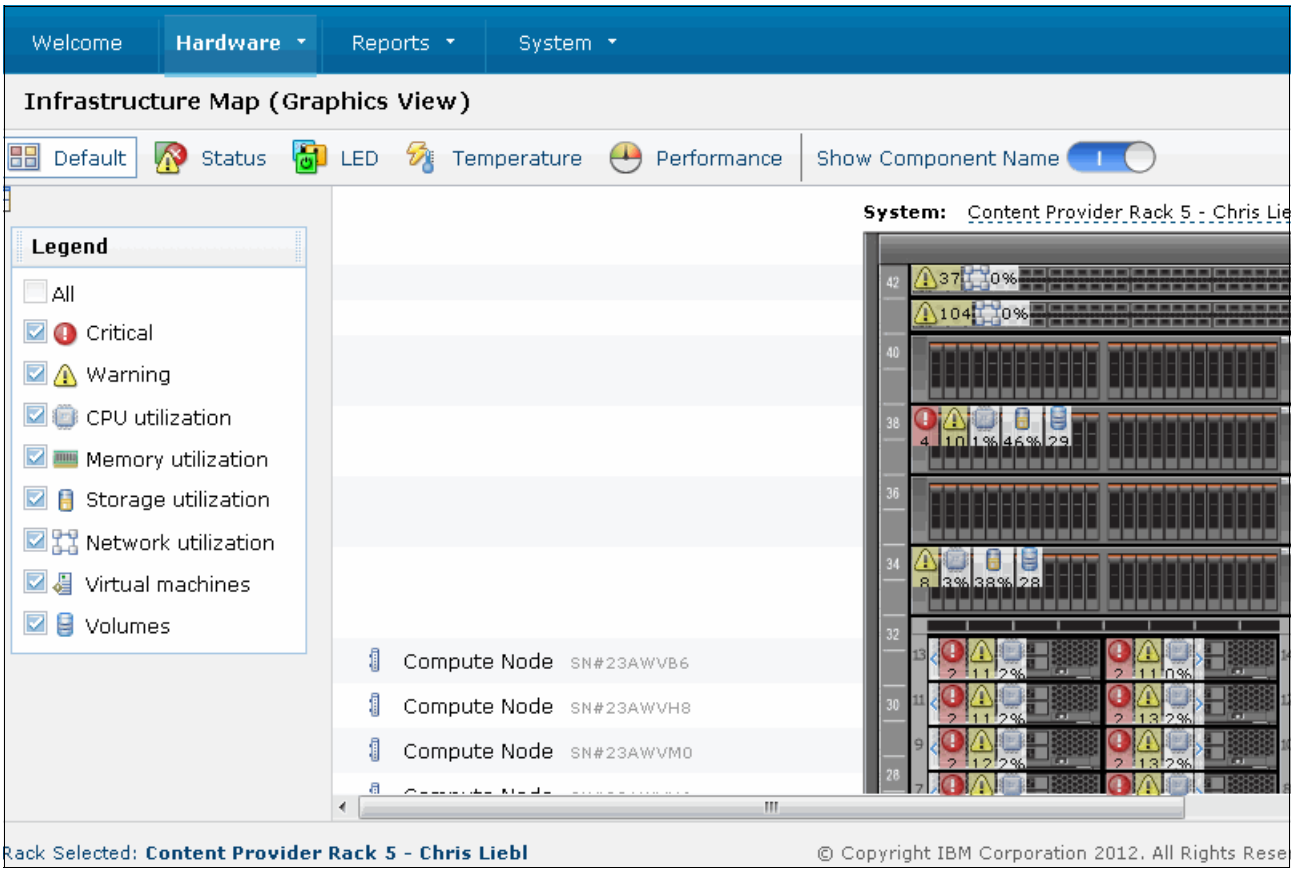


Figure 8-12 Graphical view of the infrastructure map

To switch your view of the infrastructure map from graphical to tree view, click **Switch to Tree View** from within the System Console pane, as shown in Figure 8-13.

Figure 8-13 Switch option for views of the infrastructure map

The infrastructure map tree view is shown in Figure 8-14.

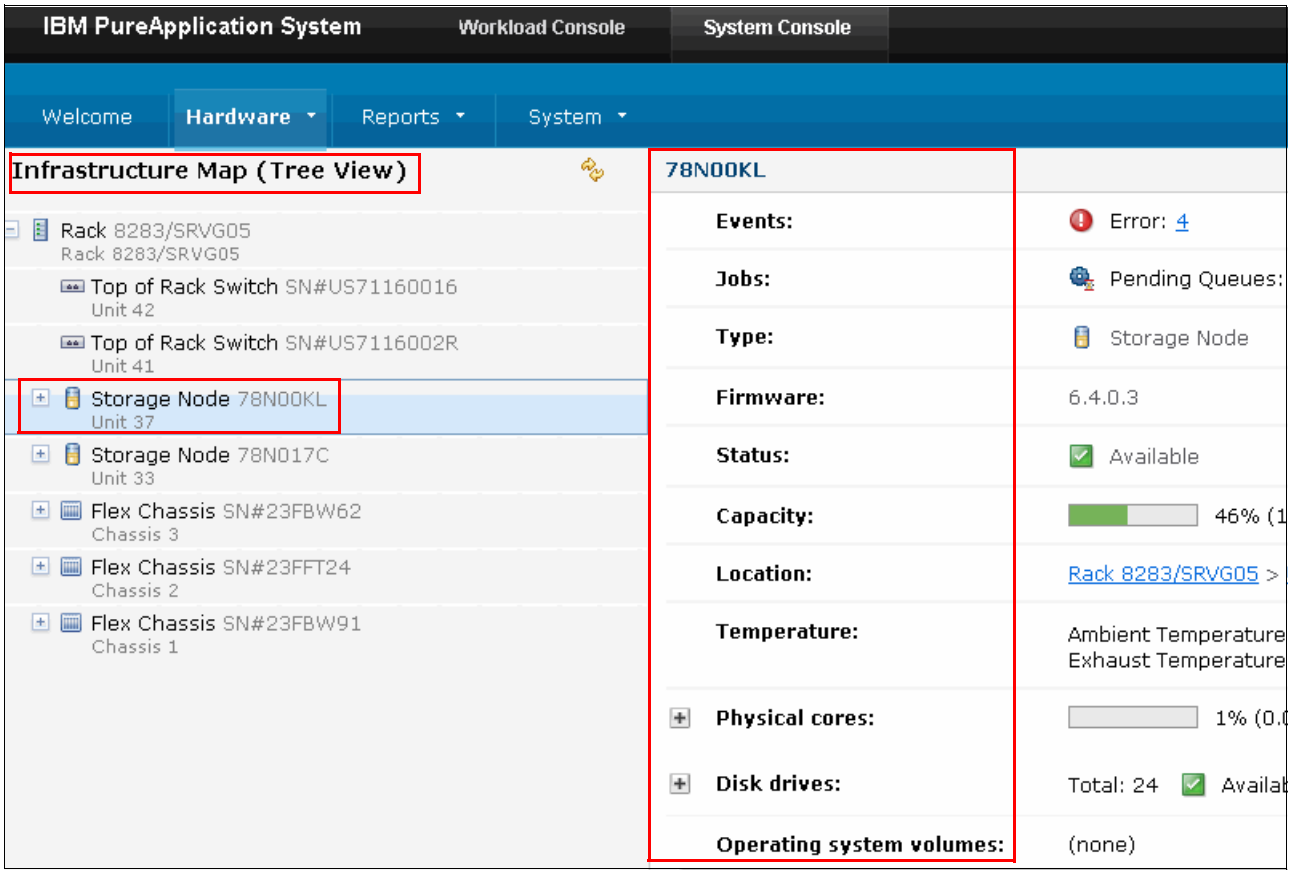


Figure 8-14 Tree view of infrastructure map

8.2.2 Workload Console

Workload Console is used to troubleshoot the deployed virtual application instances and virtual systems. Virtual applications have common troubleshooting capabilities that are provided by the foundation pattern type.

To troubleshoot issues that are related to virtual applications and virtual systems, use the Workload Console, as shown in Figure 8-15.

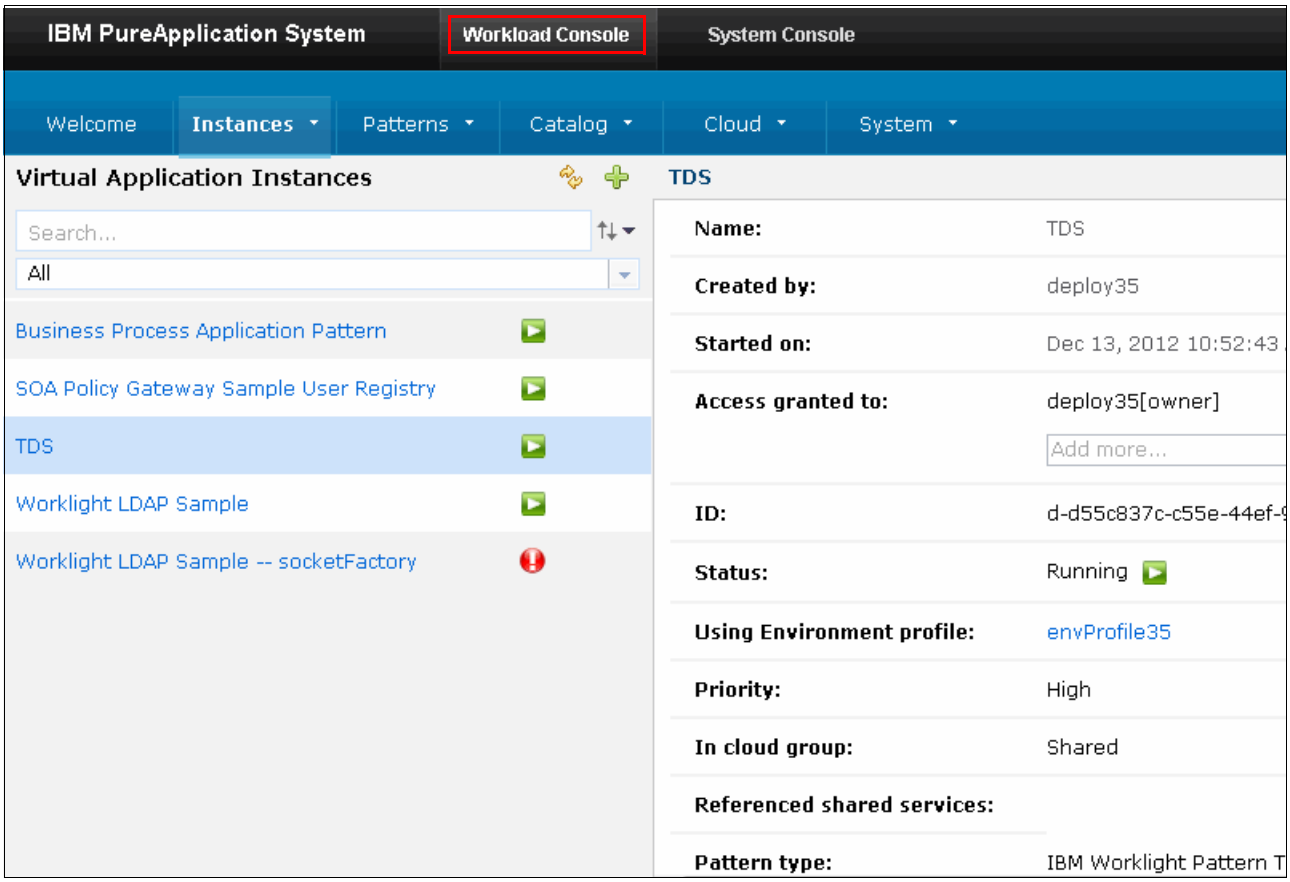


Figure 8-15 Workload Console

Virtual application instance

To troubleshoot the virtual application patterns that are based on the pattern types, enable the pattern type and configure the system plug-ins for those pattern types (where applicable). Enabling and configuring must be done to view or create the virtual application patterns.

To see the virtual application instance logs, click **Workload Console** → **Instances** → **All instances**. Select **Virtual Application** and double-click the selected Virtual Application.

You can see the components that are related to the virtual application instance view. These components include middleware logs and the operations menu, as shown in Figure 8-16.

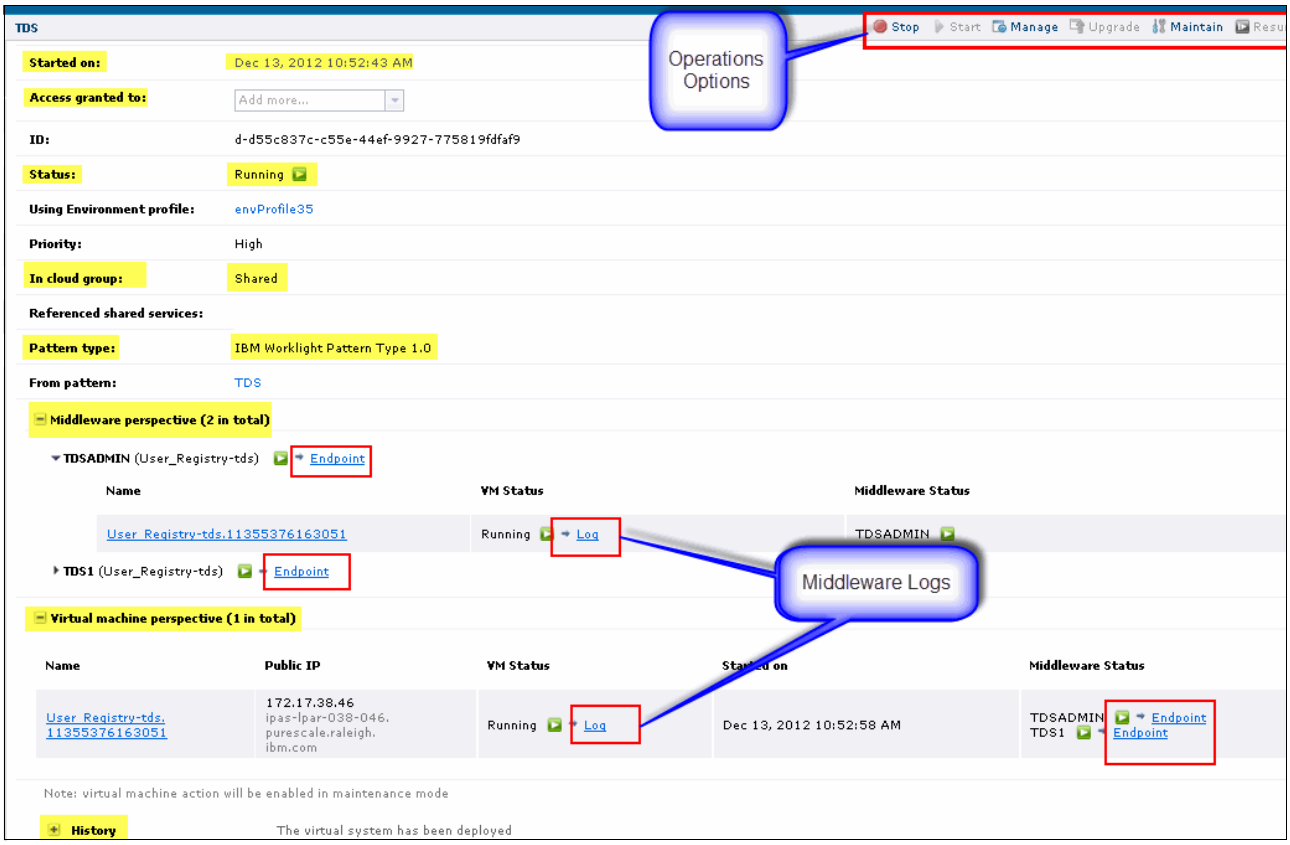


Figure 8-16 Virtual application instance view

As shown in Figure 8-16 on page 342, when the virtual application is started, then the status, type of cloud group, type of pattern that is used for this application, middleware logs, and endpoints are displayed. You can see the middleware log viewer by clicking **log** in the middleware prospective. The log then displays, as shown in Figure 8-17.

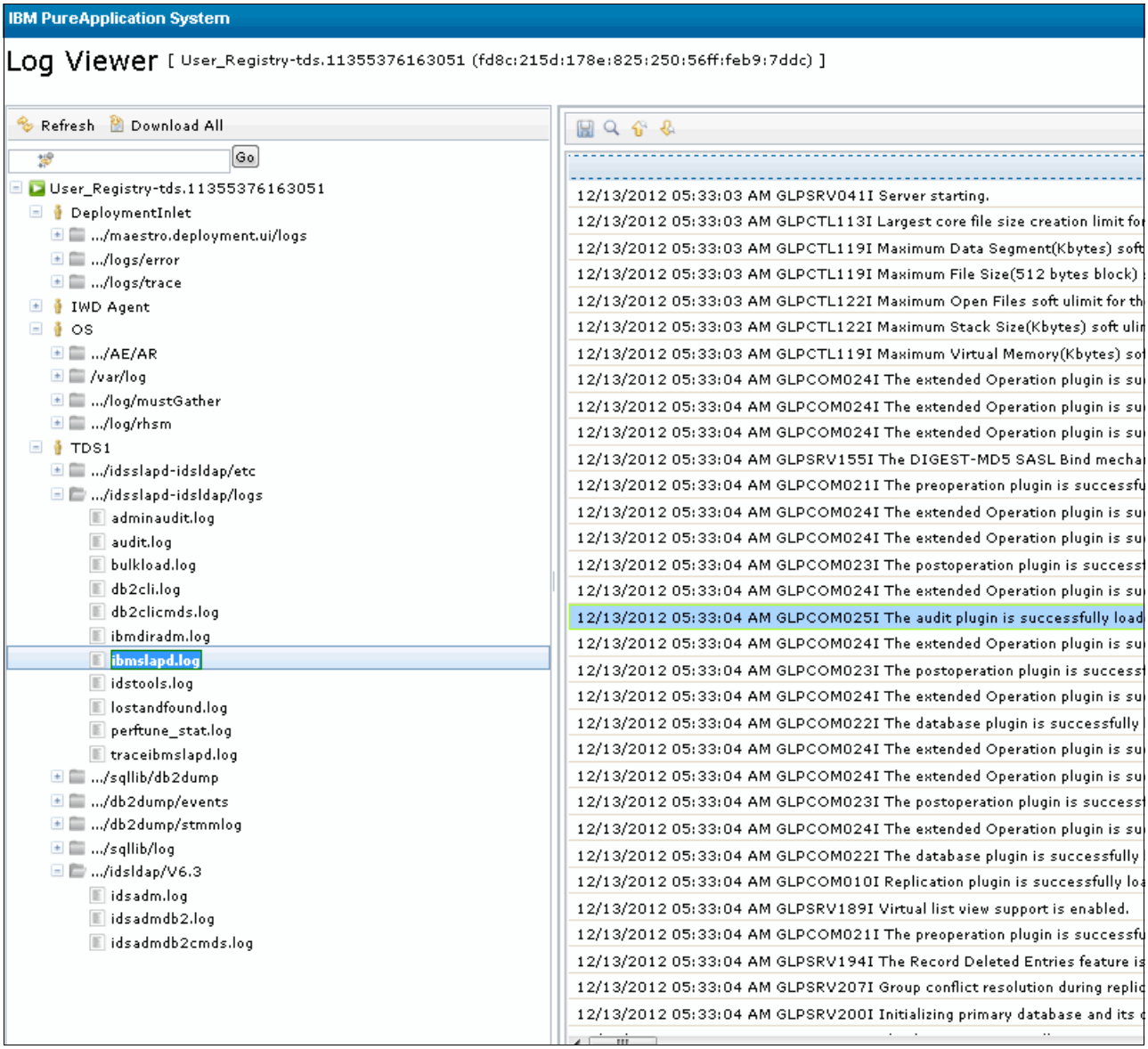


Figure 8-17 Virtual machine middleware log viewer

In Log viewer, you can download the logs or you can view the log in a browser. You can see logs from the operating system, agent, middleware, and user registries. Figure 8-17 shows those options and as an example, TDS1 was selected. TDS1 is a virtual application middleware instance. The pane shows the logs available for the IBM directory server.

Configure the trace for workload agent

Workload agent tracing must be enabled to debug and troubleshoot problems that are related to the workload agent process. The following methods are available to enable the tracing:

- Existing

  To enable tracing for existing virtual application instance workload agent trace settings, complete the following steps:

  a. Browse to **Workload Console** → **Instances** → **Virtual Applications,** then select the application instance.

  b. Click **Manage**, then click **Operation**, and select the agent role from the list of operations.

  c. The detail page displays the configuration options. Expand the Update agent trace settings by clicking the **+** icon, and specify the appropriate settings, as shown in Figure 8-18.

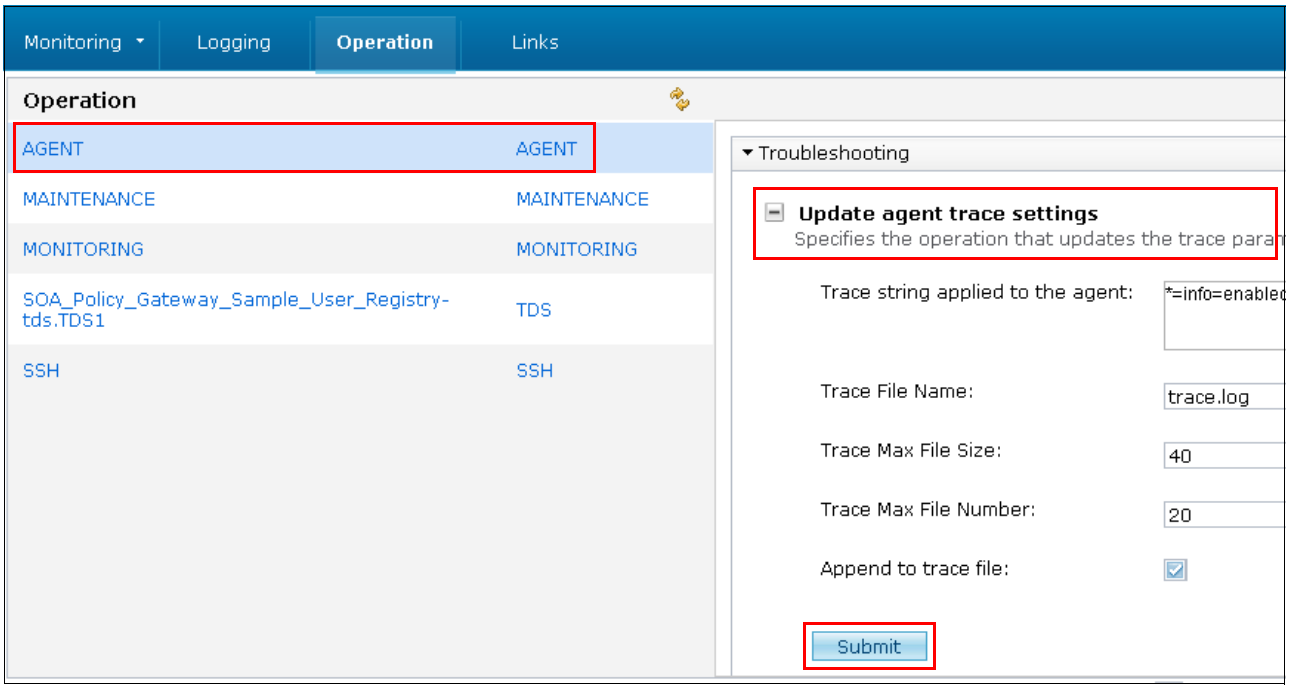


Figure 8-18 Existing virtual application instance workload agent trace settings

- New

  To enable tracing for all new virtual application deployments, configure the workload agent plug-in to enable tracing of the workload agent process. Complete the following steps to enable this type of tracing:

  a. Browse to **Workload Console** → **Cloud** → **System Plug-ins** → **Foundation Pattern Type**.

  b. From the drop-down list, select the agent plug-in that you want.

  c. Click **Configure toolbar**, and specify the settings, as shown in Figure 8-19 on page 345.

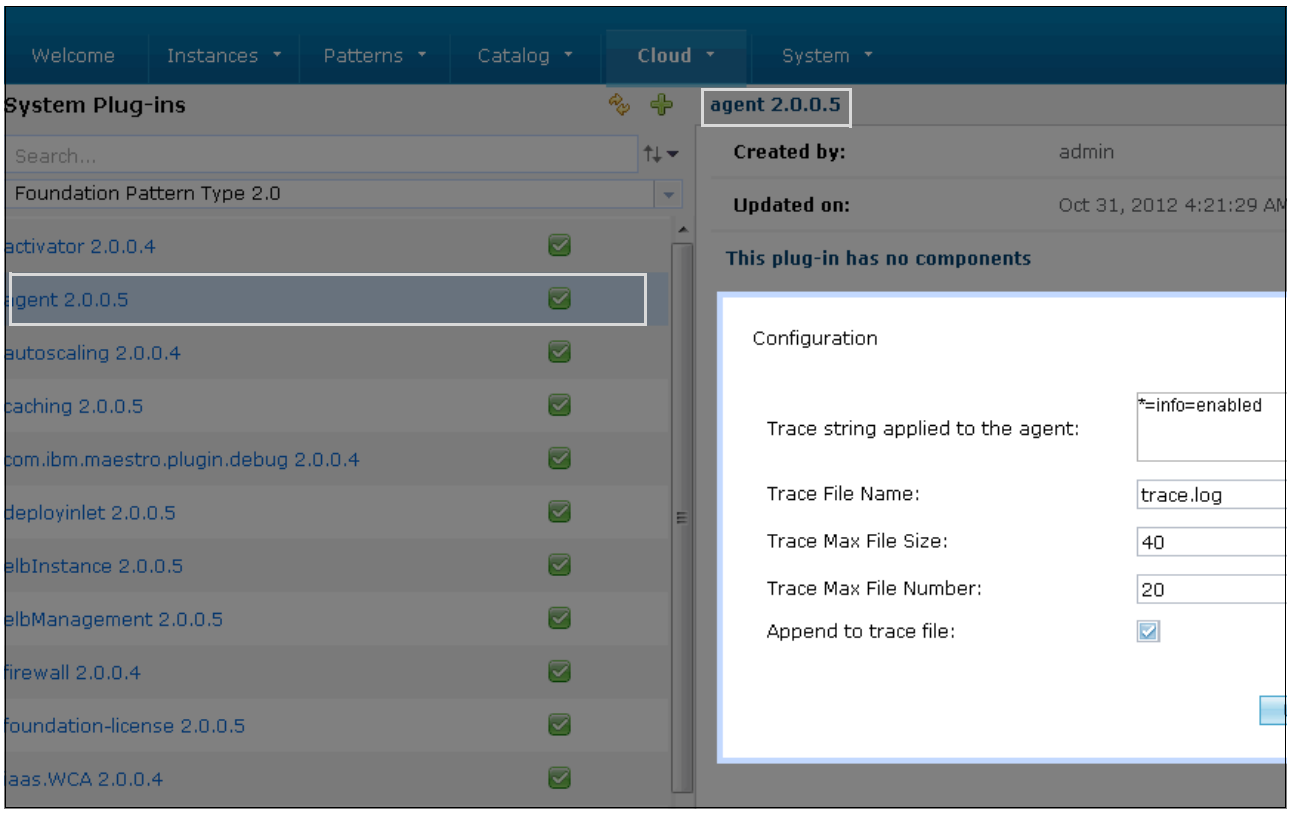


Figure 8-19 New virtual application deployments and workload agent trace settings

Virtual system instance

The virtual system instances panel is used to view and download the log files, access the administration console, and enable various traces.

To troubleshoot the virtual system instances problems, use the virtual system instances panel. You can access that panel by browsing to **Workload Console** → **Instances** → **Virtual Systems**.

During deployment of the virtual system instance, enable the Secure Shell (SSH) and virtual network computing (VNC) options to access the SSH and VNC session. SSH is a method of securely communicating to target or another computer. VNC is a method of communicating to target or another computer in graphics mode. You can enable traces by using the WebSphere console.

The virtual system instance panel for each virtual machine can be opened. You can log in to the virtual machines as a root user, as shown in Figure 8-20.

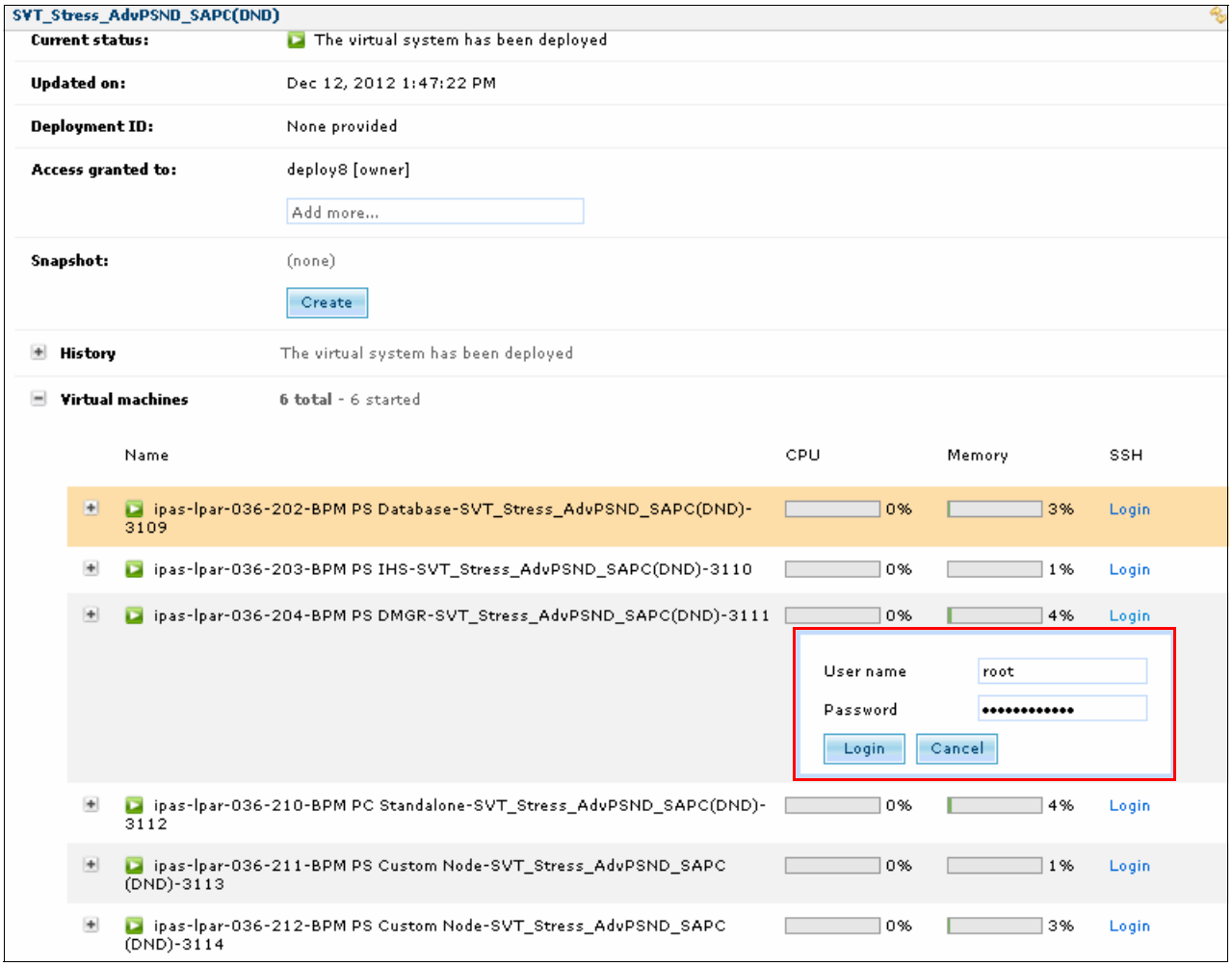


Figure 8-20 SSH login to virtual machine

You can download or view the logs of any virtual machine, as shown in Figure 8-21 on page 347.

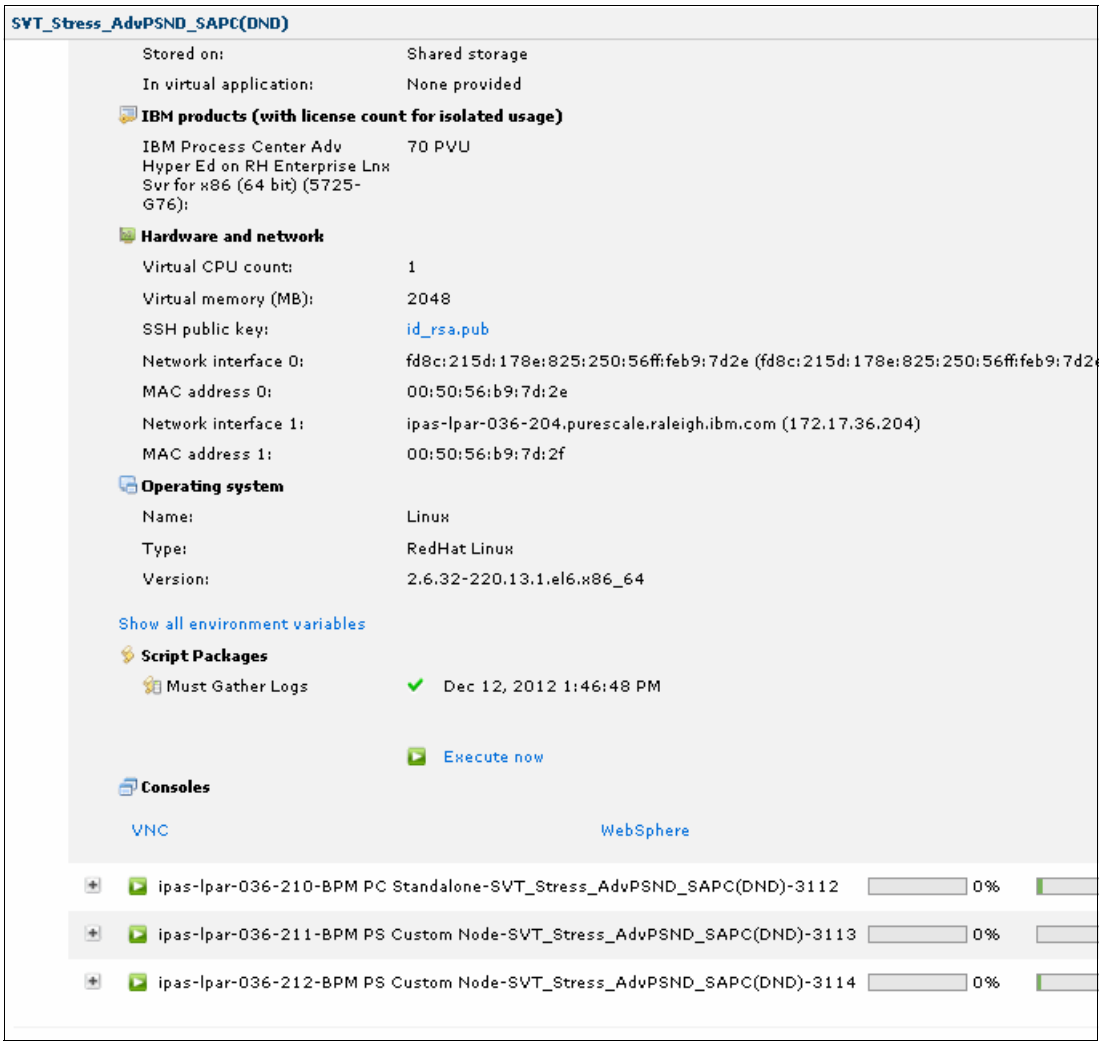


Figure 8-21 Download or view log files

8.3 Command-line interface

This section describes problem determination by using the command-line interface (CLI) and working with IBM PureApplication System W1500 and W1700. This section covers how CLI is used with objects for problem determination and multi-byte character sets.

The following types of objects are available for problem determination:

- Diagnostics objects
- Trace objects
- Errors objects

Diagnostics object

Diagnostics objects represent the diagnostics package for the IBM PureApplication System.

By using CLI, you can access help for a diagnostics object. To use help, pass the diagnostics object as an argument to the **`help()`** function, as shown in Example 8-1.

Example 8-1 Diagnostics object help by using a command line

```
>>> help(deployer.diagnostics)
```

The following diagnostics object methods are available:

- **`get`** method

  The **`get`** method downloads the diagnostics package as a compressed file. This method takes an optional path where the file is stored. The default path is `./diagnostics.zip`, as shown in Example 8-2.

  Example 8-2 Default path for get method

  ```
  >>> deployer.diagnostics.get()
  >>> deployer.diagnostics.get('/some/path/diagnostics.zip')
  ```

- **`getLatest`** method

  The **`getLatest`** method downloads a diagnostics package as a compressed file that contains only the latest version of the diagnostics files and is a much smaller file size. This method takes an optional path where the file is stored. The default path is `./diagnostics.zip`, as shown in Example 8-3.

  Example 8-3 Default path for getLatest method

  ```
  >>> deployer.diagnostics.getLatest()
  >>> deployer.diagnostics.getLatest('/some/path/diagnostics.zip')
  ```

Trace object

Trace objects represent running trace files on the IBM PureApplication System.

By using CLI, you can access help for trace objects. To use help, pass the trace object as an argument to the **`help()`** function, as shown in Example 8-4.

Example 8-4 Trace method

```
>>> help(deployer.trace)
```

The following methods are available for trace objects:

- **Add**: This method adds a logger and optional log level to the trace file specification. The default value is OFF. Logger names use Java package name syntax and the log levels are one of the following values:
  - OFF
  - SEVERE
  - WARNING
  - CONFIG
  - INFO
  - FINE
  - FINER
  - FINEST

  Example 8-5 shows the **add** method in use with a logger value.

  Example 8-5 Add method

```
   >>> deployer.trace.add('com.ibm.ws.deployer', 'FINE')
>>> deployer.trace.add('com.ibm.ws.deployer.not.interested')
```

- **Remove**: This method removes an existing logger from the trace file specification. Logger names use Java package name syntax, as shown in Example 8-6.

  Example 8-6 Remove method

```
   >>> deployer.trace.remove('com.ibm.ws.deployer.not.interested')
```

- **Set**: This method sets the log level for an existing logger in the trace file specifications. Logger names use Java package name syntax and the log levels are one of the following values:
  - OFF
  - SEVERE
  - WARNING
  - CONFIG
  - INFO
  - FINE
  - FINER
  - FINEST

  Example 8-7 shows the **set** method that uses logger values.

  Example 8-7 Set method with logger value

```
   >>> deployer.trace.set('com.ibm.ws.deployer', 'FINE')
   >>> deployer.trace.set('com.ibm.ws.deployer', 'SEVERE')
```

- **Spec**: This method returns a map with the trace file specification for the IBM PureApplication System. The map has key-value pairs in which the key is the package name and the value is the log level.
- **Tail**: In this method, **tail** prints the last <*n*> lines of the file, where <*n*> is an integer. The default value is 10. Example 8-8 shows the tail method with integer values.

  Example 8-8 Tail method with integer values

```
   >>> deployer.trace.tail(10)
   >>> deployer.errors.tail(100)
```

Errors object

Errors objects return an ErrorFile object that represents the running error file on the IBM PureApplication System.

Use the CLI to access help for errors objects. For help, pass the Errors object as an argument to the **help()** function, as shown in Example 8-9.

Example 8-9 Errors object

```
>>> help(deployer.errors)
```

The errors object has one method, the **tail** method. The **tail** method prints the last <*n*> lines of the file, in which <*n*> is an integer. The default value is 10. The **tail** method is shown in Example 8-10.

Example 8-10 Tail method

```
>>> deployer.trace.tail()
>>> deployer.errors.tail(100)
```

Resolving CLI errors when multi-byte character sets are used

Codecs are commonly used for languages with multi-byte characters. You can see the errors when you use the codecs. Jython 2.5.1 errors most commonly occur when the CLI is used in interactive mode. Errors usually occur as LookupErrors, which indicates that your encoding is unknown. If you are using Jython 2.5.1 in a multi-byte character set, you can see a LookupError error, as shown in Example 8-11.

Example 8-11 LookupError

```
LookupError: unknown encoding 'gb2312'
```

If you have LookupErrors by using the CLI, complete the following steps to resolve the issue:

1. Determine the firmware version of the particular system. Identify the `lib` directory that corresponds to the firmware version of the IBM PureApplication System. The `deployer.cli/lib` directory contains one subdirectory for each PureApplication System firmware version with which the CLI communicated. Use the following command format to determine the firmware version of particular system:

   ```
   $ pure -h <system_hostname> -u <username> -p <password> -c deployer.version
   ```

   Example 8-12 shows the command that is used to determine the version of firmware for system.

   Example 8-12 Command to check firmware version for system

   ```
   $ pure -h mysystem.foo.com -u cbadmin -p pw -c deployer.version
   ```

   The output of the command that is used in Example 8-12 returns the following information:

   ```
   System at mysystem.foo.com firmware version 3.0.0.0-30839
   ```

   The `lib` directory for the output from Example 8-12 is in the following location:

   ```
   pure.cli/lib/3.0.0.0-30839
   ```

2. Update the registry file. Complete the following steps to update this file:
   a. Uncomment the following lines:

      ```
      # python.console.encoding=iso-8859-1
      # deployer.console.encoding=gb2312
      ```

   b. Update `deployer.console.encoding` to specify the correct codec for your environment. As an example, the following list includes specific codecs for certain locales:
      - Simplified Chinese: `deployer.console.encoding=gb2312`
      - Traditional Chinese: `deployer.console.encoding=big5`
      - Japanese: `deployer.console.encoding=shift_jis`
      - Korean: `deployer.console.encoding=ks_c_5601-1987`
      - Western European: `deployer.console.encoding=ibm850`

   c. Ensure that you add a back slash, `(\)` ,as an escape sequence to these characters `(\a, \b, \f,\n,\r, \t,\x,\v)`. These characters also are known as string literals when directories or file paths are specified.

      An example of these string literals is shown in Example 8-13.

      Example 8-13 String literals

      ```
      c:\\foo.tgz
      ```

   d. Restart the CLI.

8.4 Product logs and tracing

Product logs and messages can be a good resource to troubleshoot the problems. This section describes the messaging framework and explains logging customization options.

8.4.1 Enable product tracing

Product tracing is enabled on the server and the client. These options are described next.

Server-side component of the Eclipse client

A trace log level is needed to set the global log level, an individual component, or logger. In `RAFW_HOME/logging.properties`, trace log levels are SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, and ALL.

To enable tracing for all components, complete the following steps:

1. Set the global logging level to one of the trace levels:

   ```
   # Default global logging level is ALL

   level=ALL
   ```

2. Run the command or action that caused the problem.
3. Review the messages in the product logs.
4. To see more messages, set the global logging level to a higher trace level and rerun the command or action.
5. Fix the problem.
6. Reset the global logging level to INFO.

Client-side components of the Eclipse client

To enable tracing on the client, complete the following steps:

1. Start the Eclipse client, and set the Java virtual machine property (in the `log4j.xml` configuration file or in the `rafui.ini` file). Use the following options to set the Java virtual machine property:
   - By using the CLI, start the Eclipse client by issuing the following command:

     ```
     RAF_CLIENT_HOME\rafui.exe -vmargs
     -Dcom.ibm.rational.raf.tracing.configfile=path_to_ log4j_config_file
     ```

   - You can modify the shortcut to start the Eclipse client. To do so, right-click the shortcut to the Eclipse client, and click **Properties**. In the Target field (after the path), add `-vmargs -Dcom.ibm.rational.raf.tracing.configfile=path_to_ log4j_config_file`. Start the Eclipse client by using the modified shortcut.
   - You can modify the `rafui.ini` file. To do so, open the `RAF_CLIENT_HOME\rafui.ini` file. After the vmargs section, add `-Dcom.ibm.rational.raf.tracing.configfile=path_to_ log4j_config_file`.
2. Save and close the file.

8.4.2 Product logs

Product logs contain generated messages from product commands and actions, which are sent to the standard output stream.

Message severity levels are determined where the messages are logged.

Standard output stream

In a standard output stream, messages are found from all components that are of the level of INFO and higher. The following rules apply to the standard output stream:

- When global logging is set to the default (`.level=INFO`), all messages of INFO level and higher are sent to the standard output stream.
- Commands or actions that are run on the framework server. Standard out messages are sent to the terminal.
- Commands or actions that are run in the web client. Standard out messages are sent to the step logs for the job.
- Actions that are run on the target system. Standard out messages are generated that are sent to an `action.log` file that is saved to the `RAFW_HOME/logs` directory on the framework server.

Important: If you decrease the global logging level (for example, from INFO to SEVERE), fewer messages are sent to the standard out.

You cannot increase the messages that are sent to standard out. If you set the global logging level to ALL, messages of severity ALL and higher are sent to product trace logs, but only messages of INFO and higher are sent to standard out.

Product logs

Product messages are written to standard output and product logs. The following types of product logs are described in this section:

- Product trace logs
- Action log
- Server and client log
- IBM Tivoli Remote Execution and Access trace log

Product trace logs

Product trace messages are written to different logs, as show in Table 8-1.

Table 8-1 Product trace logs

Log files	Content or significance
RAFW_HOME/logs/framework.log.x	Actions and other product commands that run on the framework server, or write messages to the `framework.log` files. The `framework.log` files are on the framework server.
RAFW_HOME/logs/log_scope/action.trace.x	Actions that run on target systems and write messages to the `action.trace` log files. The `action.trace` files are saved to the framework server. In the naming structure, action is the name of the action, and log_scope represents the scope or run path of the action. The following formats of the log_scope path components are available: - Cell path: `env_name/cell_name/` - Cluster path: `env_name/cell_name/cluster_name/` - Node path: `env_name/cell_name/node_name/` - Server path: `env_name/cell_name/node_name/server_name/`
RAFW_HOME/logs/wsadmin.trace.x	On target systems, actions run commands that are run by the WebSphere administrative (**`wsadmin`**) scripting program. The **`wsadmin`** program writes messages to `wsadmin.trace` files**.** The `wsadmin.trace` files are saved to the framework server.
RAFW_HOME/logs/log_scope/transfer.trace.x	Commands, that run a transfer of shared components and write messages to the `transfer.trace` file instead of an `action.trace` file. The `transfer.trace` file is on the framework server. Shared components are transferred from the framework server to a target system by running the **`rafw`** command without an action. The scope or run path of the **`rafw`** commands are represented by log_scope. The following formats of the log_scope path components are available: - Cell path: `env_name/cell_name/` - Cluster path: `env_name/cell_name/cluster_name/` - Node path: `env_name/cell_name/node_name/` - Server path: `env_name/cell_name/node_name/server_name/`

Action log

Actions are run in remote target systems. Messages that are directed to standard output on the remote target system are saved to the action log file, as shown in Table 8-2.

Table 8-2 Action log

Log file	Content or Significance
RAFW_HOME/logs/log_scope/action.log	Commands that are run on target systems write messages of INFO level and higher to the `action.log` file. The `action.log` file is on the framework server. The scope or run path of the action is represented by log_scope. The following format of the log_scope path components is available: - Cell path: `env_name/cell_name/` - Cluster path: `env_name/cell_name/cluster_name/` - Node path: `env_name/cell_name/node_name/` - Server path: `env_name/cell_name/node_name/server_name/`

Server and client logs

The location of server logs depends on the operating system. Client logs are available on Eclipse client installation directory, as shown in Table 8-3.

Table 8-3 Server and client logs

Log file	Content or Significance
UI_ROOT\workspace\.metadata\.log	Contains messages from the Eclipse client regarding client plug-ins and performance issues.
install_directory\Apache\tomcat\logs\raf.date.log install_directory/server/tomcat/logs/raf.date.log	Contains messages from the application server and shows the traffic between the Eclipse client and the framework server.

IBM Tivoli Remote Execution and Access trace log

The product uses the Tivoli Remote Execution and Access toolkit to authenticate and connect to remote target systems and to transfer files, as shown in Table 8-4.

Table 8-4 IBM Tivoli Remote Execution and Access trace log

Log file	Content or Significance
/tmp/rxa_trace.log	The Tivoli Remote Execution and Access toolkit writes messages to the `rxa_trace.log` file. The default location is `/tmp/rxa_trace.log` on the framework server.

Important: The `rxa_trace.log` cannot be in the `RAFW_HOME` directory or in any of its subdirectories.

By default, Tivoli Remote Execution and Access message logging is not enabled in `RAFW_HOME/logging.properties`. Do not enable message logging for the toolkit unless directed to do so by IBM support. Enabling messaging for the toolkit negatively impacts product performance.

Accessing product logs

Product log files are in plain text format. You can view the product logs by using any text editor. Log files are stored in the `RAFW_HOME/logs` directory.

Logging customization

For product trace logs, you can change the logging levels for components and change management options in `RAFW_HOME/logging.properties`. By using logging customization, you can modify the logging configuration.

The following logging configuration information is available for the server-side component of the Eclipse client:

- The global logging level is set in the `RAFW_HOME/logging.properties` configuration file.
- The `logging.properties` file controls the size of the logs that are generated by the interaction between the Eclipse client and the framework server.
- On Windows Systems, the file is in the `RAF_SERVER_HOME/Apache/tomcat/webapps/rafservices/WEB-INF/classes` directory.
- On Linux Systems, the file is in the `RAF_SERVER_HOME/server/tomcat/webapps/rafservices/WEB-INF/classes` directory.
- The logging file that controls the number of messages that are generated by the Apache file handler is in `org.apache.juli.FileHandler.level`.
- The logging file that controls the number of messages that are generated by the server-side component of the Eclipse client is in `com.ibm.rational.raf.services.level`.

Important: If you are running the product from WebSphere Application Server rather than from an Apache Tomcat server, use the administrative console to set logging levels.

Log level settings

Log level settings control the quantity of messages that are sent to the product logs. Log levels in the `RAFW_HOME/logging.properties` file specify the number of messages that are generated for a particular component.

The following supported level values are available:

- SEVERE (the lowest level that includes only severe error messages)
- WARNING
- INFO
- CONFIG
- FINE
- FINER
- FINEST
- ALL (includes all possible messages)

Use the following syntax formula to set the log levels:

```
property=level
```

For this formula, change `property` to the name of the specific logging component. Change `level` to one of the supported level values.

Components in the `RAFW_HOME/logging.properties` file are commented out by default, except for the global logging component, which is enabled.

Example 8-14 shows the default global logging level.

Example 8-14 Default global logging level information

```
# Default global logging level.
.level=INFO
```

Example 8-15 shows the log level parameters.

Example 8-15 Log Level parameters

```
# .level=INFO
# example.component.level=ALL
```

Example 8-16 shows the removal of the comment character (#). This removal is done to activate a specific logging level for a particular component. In this example, INFO level logs are generated for all components except the `example.component`. All messages are logged for the `example.component`.

Example 8-16 Activating a specific logging level by removing the # character

```
.level=INFO
example.component.level=ALL
```

Global logging component

The global logging component sets the logging level for all components in the `RAFW_HOME/logging.properties` file. The default level for global logging components is `.level`. The default log level is `INFO` in the `RAFW_HOME/logging.properties` file, as shown in Example 8-17.

Example 8-17 Default global logging level

```
# Default global logging level.
.level=INFOScope of global logging
```

Scope of global logging

The global logging level applies to all the components or loggers in the `RAFW_HOME/logging.properties` file.

Components or loggers are preceded by the comment symbol (#) by default, as shown in Example 8-18.

Example 8-18 Scope of global logging

```
# Default global logging level.
.level=INFO
# You can override the global level for various components of the framework
#jython.level=CONFIG
#com.ibm.rational.rafw.ant.tasks.RafwLogTask .level=ALL
```

Overriding the global logging level

For an individual component to override the global logging level, uncomment the component and specify a valid log level, as shown in Example 8-19.

Example 8-19 Overriding the global logging level

```
# You can override the global level for various components of the framework
jython.level=ALL
#com.ibm.rational.rafw.ant.tasks.RafwLogTask.level=SEVERE
```

Action trace logs and customize logging properties

This section describes customizing the logging property settings. The action trace logs are in the `RAFW_HOME/logs/log_scope` directory. The following logs are the action logs:

- `action.trace`
- `wasadmin.trace`
- `transfer.trace`

The `log_scope` represents the run path of the action or of the **`rafw`** command. The following formats are available for the `log_scope`:

- Cell path: `env_name/cell_name/`
- Cluster path: `env_name/cell_name/cluster_name/`
- Node path: `env_name/cell_name/node_name/`
- Server path: `env_name/cell_name/node_name/server_name/`

Table 8-5 on page 357 describes the customizable logging properties for action trace logs in the `RAFW_HOME/logging.properties` file.

Table 8-5 Customizable properties for action trace logs

Property and default settings	Description
actionlog.count=1	The number of logs (0 - 4) per log type in the rolling log sequence. Messages are written to the current log file of using `action.trace.0`, `wsadmin.trace.0`, or `transfer.trace.0`. Logs are stored at the `log_scope` that represents the run scope of the action or command.
actionlog.limit=1024000 bytes	The maximum number of bytes in the log file. When the approximate maximum byte number is reached, messages append or overwrite their log file. If `actionlog.append=TRUE`, messages append to the current log file. If `actionlog.append=FALSE`, messages overwrite previous messages in the current log file.
actionlog.append=TRUE	The current logs are `action.trace.0`, `wsadmin.trace.0`, and `transfer.trace.0`. If `actionlog.append=TRUE` (the default), messages append to the current log file. If `actionlog.append=FALSE`, each action or **`rafw`** command run overwrites messages in the current log file. actionlog.append=TRUE | FALSE (You can use TRUE or FALSE).

Customizing logging properties for framework server trace logs

You can customize logging for framework server trace logs by changing the property settings. Framework server trace logs are in the `RAFW_HOME/logs` directory. The format for naming individual logs is `framework.log.x` (where `x` is a number `0 - 4`). When the approximate maximum byte limit is reached, the content in `framework.log.0` rolls to `framework.log.1`. The `framework.log.0` log always contains the current message output.

Table 8-6 describes the customizable properties of framework trace logs.

Table 8-6 Customizable properties of framework server trace logs

Property and default settings	Description
java.util.logging.FileHandler.pattern=/logs/framework.log.x	The default location of the `framework.log.x` trace logs is in `RAFW_HOME/logs`. The variable `x` represents the log number in the rolling log sequence. Messages are written to the current log file, `framework.log.0`
java.util.logging.FileHandler.limit=10240000 bytes	The approximate maximum number of bytes in a `framework.log.x` file.
java.util.logging.FileHandler.count=5	The number of logs (0 - 4) in the rotating log sequence. The most current message output is always written to `framework.log.0`.
java.util.logging.FileHandler.append=true	If `FileHandler.append=true` (the default), messages are appended to the current log file until it reaches the approximate maximum number of bytes that is specified in `FileHandler.limit`. If `FileHandler.append=false`, messages overwrite previous messages in the current log file. append=true | false (true or false)

Customizing logging for Jython classes

This section describes customizing logging properties for Jython classes. Several components or loggers control log levels for Jython classes as provided by the product.

To diagnose a problem in the Jython code, you can modify log level for all Jython classes. You can remove the comment from the `jython.level` component and change the log level.

Example 8-20 shows the log level.

Example 8-20 Log level

```
# Default global logging level.
.level=INFO
# You can override the global level for various components of the framework
jython.level=ALL
```

Example 8-21 shows that the log level is modified for the ConfigReader.

Example 8-21 Log level for ConfigReader

```
############################################################
# Facility specific properties.
# Provides extra control for each logger.
############################################################
# For example, set the com.xyz logger to only log SEVERE
# messages:
#com.xyz.level = SEVERE
ConfigReader.level=ALL
#TransferClient.level=ALL
#manageServer.level=INFO
#XmlProperty.level=INFO
#SystemUtils.level=INFO
#FileHelper.level=INFO
```

8.5 Image Construction and Composition Tool

This section describes various problems that are related to the Image Construction and Composition Tool.

8.5.1 Troubleshooting the Image Construction and Composition Tool

This section focuses on the following possible problems with the Image Construction and Composition Tool install and upgrade:

- Install executable launcher error during installation
- Install executable Error 1 during installation
- ICYOIG0005E: Universal ID '<ID>_<version>' is not unique
- ICYOIG0005E: Universal ID '<ID>_<version>' is not unique
- Use the current file link, does not work correctly
- VMware ESX Server error when virtual machines are used with snapshots
- Failure during open virtualization appliance disk conversion
- CYOES0033E: Unable to import image, another image exists with the name
- Error occurs during open virtualization appliance (OVA) export
- Open virtualization appliance deployment to VMware vSphere Client fails
- Virtual image capture fails
- Files are not replaced correctly (software bundle)
- Unable to upload 2 GB or higher size file in the user interface
- Limitation when concurrent updates are performed
- Cannot remove software bundles from a personality
- Bundles are not displayed in the list of compatible images
- Limitation when bundles are added to a personality
- Limitation when images that are synchronizing are deleted

- Time lag when virtual images are deleted
- SSL exception error
- Cannot pass characters for arguments on the command line
- TCP/UDP firewall ports might appended incorrectly on a Red Hat system after synchronization

These problems are described next.

Install executable launcher error during installation

Sometimes upon installation, an installation executable launcher error occurs. Notice whether the Image Construction and Composition Tool fails with following error when the `./install` file runs:

```
The Install executable launcher was unable to locate its companion shared library.
```

This error indicates that a version of the Image Construction and Composition Tool exists on the system. To resolve this problem, upgrade the most recent version or install new most recent version instead of installing the same version of the Image Construction and Composition Tool.

Install executable Error 1 during installation

During the installation of the Image Construction and Composition Tool on a system, you might receive the following error:

```
‘exec error 1’
```

The problem is caused when port `443` is used by another application. To resolve the problem, issue the following command to check the port status:

```
# netstat -na | grep 443
```

If status returns a response that it is in `LISTEN` mode, the port is in use. You must ensure that port `443` is not used by another application. You also must make any needed configuration changes to free the port.

ICYOIG0005E: Universal ID '<ID>_<version>' is not unique

The version of every asset and combination of universal ID must be unique. If you extend an image and do not enter a unique universal ID and version number combination for the newly extended virtual image, you receive an error. The following error message is found in the `trace.log` and `error.log` files:

```
Cannot create the image: Error: HTTP response code:409
message: "CYOIGOOO5E: Universal ID '<ID>_<version>' is not unique."
```

To resolve this issue, enter a unique universal ID and version combination for the extended asset.

Use the current file link, does not work correctly

When you receive this error, the Image Construction and Composition Tool did not revert to the previous file correctly. The error might occur when the `Cancel - use the current file` is used in the Files to Copy section on the Install or Config tab for a software bundle. The occurrence is specific to an attempt to cancel the process and use an edited file.

Consider the following scenario. You upload a file to the Files to Copy section on the Install or Config tab for a software bundle, save that file, and edit it. Then, after the file is saved and edited, you select a new file to upload. Next, you click **Cancel - use the current file**. The problem occurs because the file is still updated with the new file, although you tried to revert to the previous file. This issue does not always occur, but can occur on occasion.

To resolve this issue, you have the following options:

- Click **Cancel** if the file is not updated correctly.
- If you click **OK**, and see the wrong file, refresh the software bundle to omit all the changes.

VMware ESX Server error when virtual machines are used with snapshots

If you use virtual machines with snapshots in VMware ESX, you receive an error in the `trace.log` file. The VMware ESX cloud provider does not support virtual images where the virtual machine has snapshots.

The error message in the `trace.log` file is similar to the following output:

```
java.io.IOException: disk conversion of /drouter/ramdisk2/mnt/
raid-volume/raid0/templates/2/base-rhel55-64-000001-delta.vmdk
from monolithicFlat to streamOptimized failed!
```

From that output, the `-delta.vmdk` syntax indicates that there are snapshots in the virtual machine.

To resolve this issue, you have the following options:

- Do not use virtual machines that have snapshots.
- Remove the snapshots from your virtual machine before you use the virtual machine in the Image Construction and Composition Tool.

Failure during open virtualization appliance disk conversion

This error occurs when the conversion of disk files that are contained in the open virtualization appliance (OVA) files fails. When the disk files are converted, an error occurs in the OVA import-to-export operations.

To resolve this issue, verify that the following conditions are met:

- The free disk space that is available in the `/drouter` directory. The free disk space must be at least twice the size of the disk that is converted.
- If you are getting errors similar to `"Bad ELF interpreter"`, verify the `ld-linux` shared library is available.
- In Red Hat Enterprise Linux V6.1, you must install the following packages:
  - `# yum install libz.so.1`
  - `# yum install ld-linux.so.2`

CYOES0033E: Unable to import image, another image exists with the name

Importing the same OVA image into multiple VMWare ESX server cloud providers causes a failure. The failure gives the following error message:

```
CYOES0033E: Unable to import image. Another image exists with the name:
CoC_RHEL62_64 and version: 1.0.6. If the image has already been
imported using another cloud provider, use the Import from
cloud provider option instead of the Import option.
```

To avoid this issue, verify that the image you want to use was not imported for this VMware ESX cloud provider. If the image was imported for a different VMware ESX cloud provider, you can still import the image into other VMware ESX cloud providers. To resolve the multi-import issue, instead of using the Import option, use the Import from cloud provider option.

Error occurs during open virtualization appliance (OVA) export

During OVA export operations, errors occur. You might see the following error message in the log files:

```
CYOCU0005
```

To resolve this issue, ensure that the user that is specified in the user interface has write permission for the specified target folder.

Open virtualization appliance deployment to VMware vSphere Client fails

This error can occur when an OVA template is deployed from VMware vSphere Client by using the **File** → **Deploy OVF Template**. If it occurs, the following error message is displayed:

```
This OVF package uses features that are not supported when deploying directly to
an ESX host.
The OVF package requires support for "OVF Properties" and "Unsupported element
Property"
```

OVA deployment directly to ESX or ESXi is not supported. To resolve this issue, you must use VMware vCenter Server (VirtualCenter) instead of VMware ESX/ESXi to deploy the OVA template. To verify the server type, use VMware vSphere Client to select the server in the left column. In the right pane above the Summary tab, check whether the server is a VMware ESX/ESXi server or a VMware vCenter Server.

If you do not have VMware vCenter Server (Virtual Center), you can install it in the license VMware vCenter Server and then deploy the OVA file to this VMware vCenter.

Virtual image capture fails

On occasion, the virtual image capture fails with the following error message:

```
Capture failed Failure to initiate capture. There are insufficient
resources to capture the image.
Check your private instance quota. Image cannot be saved.
```

To resolve this issue, use one of the following options:

- Enable the capture option again by expanding the virtual image capture error and clicking the option to try the capture again.
- Delete the failed virtual image and then re-create the virtual image and complete the capture process again.

Files are not replaced correctly (software bundle)

Sometimes the content of a software bundle file is not updated when a software bundle is added and replaced. Specifically, when a file is added to the Files to Copy section on the Install, Config, or Rest tab of a software bundle. After you add that file and then try to replace it, the content of the file is not updated but the name of the file is updated in the Image Construction and Composition Tool.

To resolve this issue, save the software bundle before you click the file to download the contents.

Unable to upload 2 GB or higher size file in the user interface

At times, when the user interface is used, the Image Construction and Composition Tool stops responding when you upload a 2 GB or higher size file.

To resolve this issue, upload the file by using the New file window. Uploads by using this window are asynchronous, while local uploads are synchronous. Complete the following steps to upload the 2-GB file:

1. In the New file window, select **Remote** and specify the URL for the file to be uploaded in the text box.
2. Verify that the file is uploading by checking the status indicator for the entry. The status locator is in the Files to Copy table.
3. Continue to refresh the software bundle definition until the upload is complete.

Limitation when concurrent updates are performed

The limitation is that you can use only a single mechanism of updating an object in Image Construction and Composition Tool. You can use the web interface in a single browser or the CLI.

Cannot remove software bundles from a personality

If a software bundle is added to a personality and then the virtual image is synchronized, the added software bundles cannot be removed from the personality.

To resolve this issue, delete the existing personality and create another personality where you add only the software bundles that you need.

Bundles are not displayed in the list of compatible images

Bundles are not displayed in the list of compatible images.

To resolve this issue, in the Add software bundles window, clear the check box for the **Show only bundles compatible with the image** option.

Limitation when bundles are added to a personality

For virtual images that have a single personality, all software bundles that are added to the image are automatically added to the personality. This behavior means that, in this case, the personality is identified with the image.

For virtual images that have multiple personalities, software bundles that are added to the image are not added automatically to any of the personalities in the image. For these images, you must explicitly add the software bundles to the personalities, as needed.

Complete the following steps to have specific control of bundles and personality association:

1. Extend the virtual image with the single personality.
2. Add software bundles, as required. Make sure these bundles are bundles that you want to be added to the personality because the bundles are added to the personality by default.
3. Add another personality to the image.
4. Add more bundles to the images. The bundles are not added to any personality by default because there is more than one personality.
5. If required, add the bundles to the appropriate personality.

Limitation when images that are synchronizing are deleted

Do not delete an image while it is synchronizing. Wait until the synchronization process completes and then delete it. If synchronization is active, it cannot delete the image. You might not able to delete image and it gives an error while the images are deleted.

Time lag when virtual images are deleted

If you delete a virtual image, it might take a while for the virtual image to be removed from view. There is a refresh issue that causes a deleted virtual image to remain in view until the deletion process completes. The deletion process can take a while to complete. After the deletion process completes, refresh your browser. The view should be correct.

Important: The final time that you refresh the resource and then select the virtual image, you receive an `unable to load resource error` message. This message arises because the virtual image was deleted. After this message, the view is refreshed.

SSL exception error

When the CLI of the Image Construction and Composition Tool is used in Linux, you might receive the SSL exception error `socket.sslerror: (-1, 'SSL exception').` This error occurs if the Java Runtime Environment version that you are using is incompatible.

To verify your current Java Runtime Environment version, use the **`# java version`** command.

Complete the following steps to resolve the issue:

1. Use a different Java Runtime Environment, which can be downloaded from this website:

   `http://www.ibm.com/developerworks/java/jdk/linux/download.html`

2. Install the new Java Runtime Environment version.

   You can set the following environment variables to allow this new Java Runtime Environment to be used by the CLI instead of the existing Java Runtime Environment:

   ```
   # export JAVA_HOME=/opt/IBM/icct/jre
   # export PATH=$JAVA_HOME/bin:$PATH
   ```

Cannot pass characters for arguments on the command line

You might not be able to pass certain characters for arguments on the CLI. For example, when you are starting the CLI on Windows and passing arguments that contain an exclamation point (`!`), this character is removed from the argument during processing. This condition occurs because the CLIU is based on Python. In certain cases, specific characters are used to indicate special characteristics.

To resolve this issue, when you want to pass the exclamation point as a command-line argument, use the caret character (^) to escape the exclamation point character. Also, enclose the argument in double quotation marks. For example, instead of entering `"password!"`, enter `"password^!"`.

TCP/UDP firewall ports might appended incorrectly on a Red Hat system after synchronization

If a software bundle has firewall rules that are defined when the bundle is added to a Red Hat image through the VMware ESX cloud provider, the TCP/UDP firewall ports might be appended incorrectly after synchronization.

The TCP/UDP firewall ports might not be appended correctly on a Red Hat Linux system after synchronization.

Complete the following steps to resolve the problem:

1. Run the **`# service iptables save`** command.
2. Run the **`# service iptables stop`** command.
3. Browse to the `# cd /etc/sysconfig` directory.
4. Open the `iptables` file.
5. If the new firewall rules are appended after the reject rules, move the new firewall rules before the reject rules. If the new firewall rules use a UDP port instead of a TCP port, change `tcp` to `udp` for the new firewall rule.
6. Run the **`# service iptables start`** command.

8.5.2 Image Construction and Composition Tool log files

This section describes Image Construction and Composition Tool log files and the settings that are required to troubleshoot problems. The following issues are described in this section:

- Accessing and downloading the logs
- Using log files to troubleshoot installation problems
- Synchronization log files
- Configuring timeout settings
- Configuring logging levels for troubleshooting
- Manually modifying the Image Construction and Composition Tool file
- Virtual machine activation troubleshooting

Accessing and downloading the logs

The Image Construction and Composition Tool server log files are accessed from the user interface. Click **Welcome** on the home page and then click **Download logs.** The logs are downloaded locally as compressed log files, as shown in Figure 8-22.

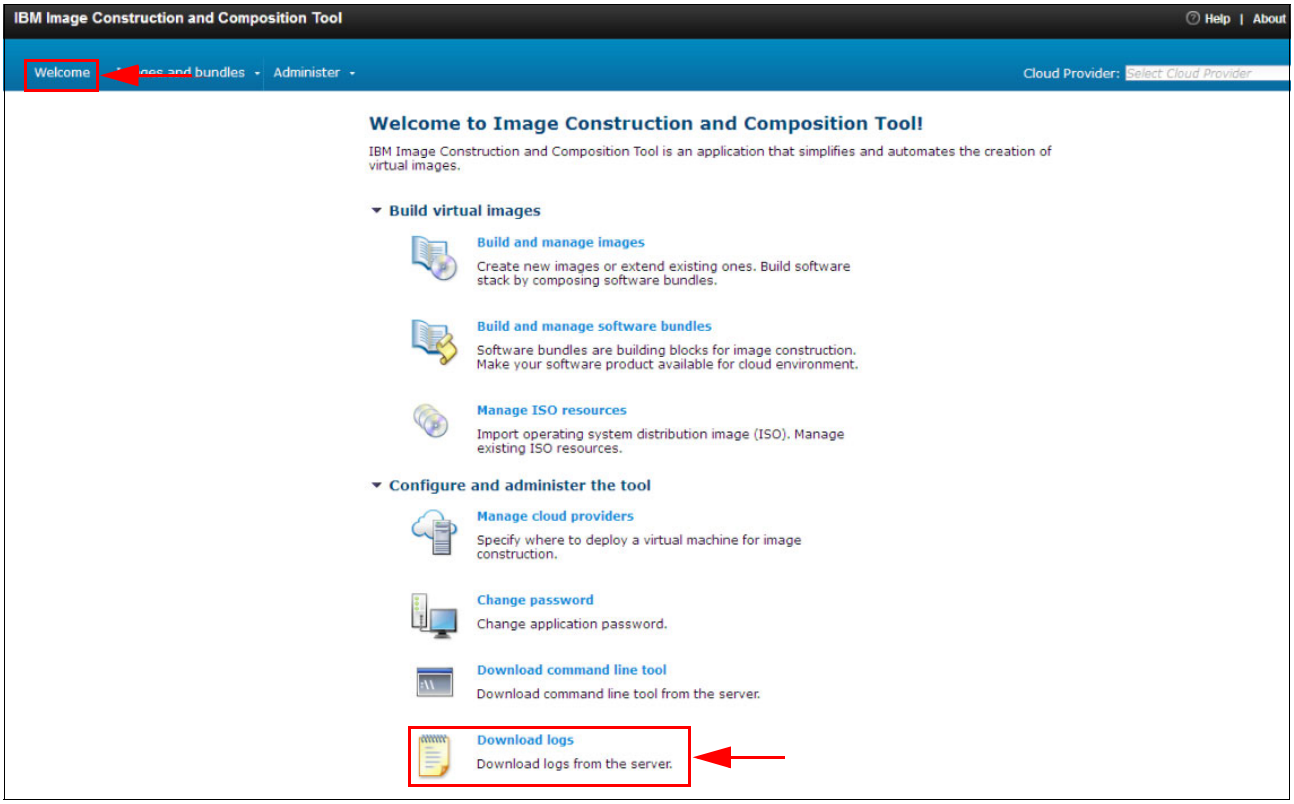


Figure 8-22 Image Construction and Composition Tool Server log download page

The downloaded compressed log file contains the following log files:

- Trace:
  - Location of the trace file is `/drouter/ramdisk2/mnt/raid-volume/raid0/logs`
  - The `trace.log` file is the most recent trace file.
  - Default size of the `trace.log` file is 100 MB
  - Nine levels of `trace.log` backup files
- Error:
  - Location of the trace file is `/drouter/ramdisk2/mnt/raid-volume/raid0/logs`
  - The `error.log` file is the most recent error file.
  - Default size of the `error.log` file is 2 MB.
  - Four levels of `error.log` backup files

Using log files to troubleshoot installation problems

During Image Construction and Composition Tool installation, if any problem occurs, you can see the following manager log files to troubleshoot the installation problems:

- `# /var/ibm/InstallationManager/logs/<date_time>.xml`
- `#/var/ibm/InstallationManager/logs/ant/<date_time>.log`

> **Important:** The date_time format is YYYYMMDD_HHSS. For example, 20121206_1139.

Important: In the log files, the Image Construction and Composition Tool message codes start with `CYO`.

Synchronization log files

Log files can help you troubleshoot the problems that are related to virtual image synchronization. If synchronization fails before the deployed virtual machine, no log files are available. If synchronization failed during or after the bundle installation, see the following log files for use with troubleshooting:

- VM logs: `error.log` and `trace.log`
- Bundle installation logs: `err.log` and `out.log`

Configuring timeout settings

Timeout settings are used to indicate how long they can wait to come out from session. You can set the timeout settings parameters in a configuration file, which is in `/opt/IBM/icon/icn.app/config`. The following timeout settings can be changed:

- ShimTimeoutInMinutes=120
- WindowsCommandTimeOut=3
- BundleExeuctionTimeout=360
- restReadTimeout=90000
- "maxIWDDeployWaitTime=0
- IWD_SYSTEM_CREATION_ATTEMPTS=120
- IWD_WAIT_FOR_SYSTEM_CREATION_RETRY_DELAY=30
- IWD_WAIT_FOR_DEPLOY_STATUS_DELAY=10
- IWD_WAIT_FOR_STARTUP_COMPLETE_DEALY=30
- IWD_MAX_SYSTEM_ACTIVATION_WAIT_TIME=300
- IWD_MAX_CAPUTRUE_ATTEMPTS=180
- IWD_WAIT_FOR_CAPTURE_STATUS_DELAY=60
- SCE_MAX_CAPTURE_VM_ATTEMPTS=240
- SCE_WAIT_BETWEEN_CAPTURE_ATTEMPTS=30
- SCE_MAX_CONNECTION_ATTEMPTS=120
- SCE_WAIT_BETWEEN_CONNECTION_ATTEMPTS=10
- SCE_WAIT_COMMAND_RETRY_DEPLAY=60
- SCE_MAX_DELETE_VM_ATTEMPTS=30
- SCE_WAIT_DELETE_VM_DELAY=60
- SCE_WAIT_DEPROVISION_DELAY=30

Example 8-22 shows the content of the configuration file and the settings that are available for change.

Example 8-22 TimeOut settings configuration file

```
## Timeout settings Config File.
# Configurable timeouts
    "ShimTimeoutInMinutes" : 120,

    # Enable Single sign-on
    "EnableSSO" : false,

    # ICCT instance user
    "InstanceUser": "",

    # When executing windows commands how long do we wait before we timeout
    # in minutes.  This default value for this timeout is 3 minutes.
```

```
    "WindowsCommandTimeOut": 3,
# When running the execution bundle during a synchronization,
    # how long do we wait before timing out in minutes.  Default
    # values is 6 hours.
    "BundleExeuctionTimeout": 360,
# IWD REST timeout
    "restReadTimeout": 90000,

    # IWD Max deploy wait time (minutes)  Default value is 0 which
    # disables the max Deploy time.  If set by default we have put in
    # 300 minutes (5 hours).  If this is set it will work in conjunction
    # with the following configuration parameters:
    #      IWD_SYSTEM_CREATION_ATTEMPTS
    #      IWD_WAIT_FOR_SYSTEM_CREATION_RETRY_DELAY
    #      IWD_WAIT_FOR_DEPLOY_STATUS_DELAY
    #      IWD_WAIT_FOR_STARTUP_COMPLETE_DEALY
    #      IWD_MAX_SYSTEM_ACTIVATION_WAIT_TIME
    "maxIWDDeployWaitTime": 0,

    # IWD Max number of creation attempts also the same
    # number of attempts for deployment status checks.
    "IWD_SYSTEM_CREATION_ATTEMPTS": 120,

    # IWD delay between creation retries (seconds)
    "IWD_WAIT_FOR_SYSTEM_CREATION_RETRY_DELAY": 30,

    # IWD delay between checks for deployment status in (seconds)
    "IWD_WAIT_FOR_DEPLOY_STATUS_DELAY": 10,

    # IWD wait for startup complete (seconds)
    "IWD_WAIT_FOR_STARTUP_COMPLETE_DEALY": 30,

    # Amount of time to wait for the system to become active.  This is checked
    # after system creation and deploy status checks.  Default value is 300
    # minutes.
    "IWD_MAX_SYSTEM_ACTIVATION_WAIT_TIME": 300,

    # How many times to check to see if the capture of an image
    # has completed before we fail the capture.  The default
    # value is 180 times.
    "IWD_MAX_CAPUTRUE_ATTEMPTS": 180,

    # When capturing an images in IWD.  How long to wait before
    # we check the status again in seconds.  Default value is 60 seconds.
    "IWD_WAIT_FOR_CAPTURE_STATUS_DELAY": 60,

    # Maximum number of time to attempt to capture a VM.
    # Default values is 240 times.
    "SCE_MAX_CAPTURE_VM_ATTEMPTS" : 240,

    # Time to wait between connection attempts in seconds.
    # Default value is 30 seconds.
    "SCE_WAIT_BETWEEN_CAPTURE_ATTEMPTS" : 30,

    # Maximum number of time to try to connect to a VM.
```

```
    # Default value is 120 times.
    "SCE_MAX_CONNECTION_ATTEMPTS" : 120,

    # Time to wait between connecting to the VM.
    # Default value is 10 seconds.
    "SCE_WAIT_BETWEEN_CONNECTION_ATTEMPTS" : 10,

    # When running commands on the VM.  Time to delay between checking the return
    # code of the command to see if it is done or not.  Default value is 60
seconds
    "SCE_WAIT_COMMAND_RETRY_DEPLAY" : 60,

    # Maximum number of time to try and delete a VM.
    # Default value is 30 times.
    "SCE_MAX_DELETE_VM_ATTEMPTS" : 30,

    # Time to wait between checking to see if the VM is deleted.
    # Default value is 60 seconds
    "SCE_WAIT_DELETE_VM_DELAY" : 60,

    # Time to wait after the VM's deletion state goes to deprovision.
    # Default value is 30 seconds.
    "SCE_WAIT_DEPROVISION_DELAY" : 30,
```

Configuring logging levels for troubleshooting

The trace and logs files are available to troubleshoot any problem and are required for many issues that arise. The content of these files and the logged information that is generated depends on definitions that follow the Java logging convention and WebSphere Application Server levels.

The following log level values are available for Java and WebSphere Application Server:

- OFF
- SEVERE
- WARNING
- INFO
- FINE
- FINER
- FINEST
- ALL

Manually modifying the Image Construction and Composition Tool file

Manually modifying the Image Construction and Composition Tool file helps you to set the logging levels in the configuration file to troubleshoot the problems. To update the file, access the /<ICCT install directory>/config/logging-default.config file. The changes that are made there take effect when you restart the Image Construction and Composition Tool. Complete the following steps to update the file:

1. Change to the directory <ICCT install_dir> by entering # cd /opt/IBM/icct in a command window.
2. Back up the file by entering the following syntax in a command window:

   ```
   # cp -p ./icn.app/config/logging-default.config
   ./icn.app/config/logging-default.config_currectdateandtimestamp
   ```

3. Update the following file per your requirements:

   ```
   # /opt/IBM/icct//icn.app/config/logging-default.config
   ```

4. Stop the Image Construction and Composition Tool by entering the following command:

   ```
   # /opt/IBM/icct/stop.sh
   ```

5. Start the Image Construction and Composition Tool by entering the following command:

   ```
   # /opt/IBM/icct/start.sh
   ```

The changes are now active in the system.

Virtual machine activation troubleshooting

When you are activating a virtual machine, problems can arise during deployment and synchronization during the activation of virtual image. Complete the following steps to delete the noap file and the contents of the `AR` directory:

1. Log in as root to the virtual machine by using the VMWare vSphere Client console.
2. Review the log files under the `/opt/ibm/ae/AR` directory for any errors.
3. Check whether a noap file exists in the `/opt/ibm/ae/AP` directory. If there is a noap file, complete the following steps:
   a. Delete the noap file.
   b. Remove all of the contents of the `AR` directory by entering the following syntax in a command window:

      ```
      # cd /opt/ibm/ae/AR/
      # rm -rf *.*
      ```

   c. Reboot the system.
   d. Check that the VM now activates successfully.

If the problem persists for virtual machine activation, complete the following steps:

1. In the user interface, press the ESC key.
2. The script that runs during activation is `/opt/ibm/ae/AL/master.al`. The scripts are in the `/opt/ibm/ae/AS` directory. Investigate the activation of each script and class to determine which line in each script is failing. After the failure is found, you must determine why the line in the script failed. For example, if **`ConfigLocale`** causes problems, run the following command once:

   ```
   # AE.sh -s ConfigLocale
   ```

3. Enter **`df -m`** to make sure that the file `activation.iso` is mounted. The file usually is in the `/tmp` directory.

8.6 Virtual image synchronization problems

Virtual images sometimes fail during synchronization. This section describes how to determine the cause. The following information consider the problem and the possible solutions that are related to the process of virtual image synchronization.

8.6.1 ESX VMWare Server and Microsoft Windows related troubleshooting

This section describes several possible issues and resolutions. To help you determine the cause of failing virtual images during synchronization, review the following list of issues that are related to VMware ESX:

- Virtual image synchronization failed when a trial version of VMware ESX was used
- Cannot synchronize Windows 2008 R2 image
- Timeout with Linux bundle in Windows image on VMware cloud provider
- Windows image synchronization failed

Virtual image synchronization failed when a trial version of VMware ESX was used

If you attempt to synchronize an image by using a trial version of VMware ESX, the synchronization fails when you attempt to copy the disk images to the VMware data store. When the image synchronization fails, the log contains messages that are similar to the syntax in Example 8-23 and Example 8-24. These logs can be found in the Image Construction and Composition Tool directory.

Example 8-23 Possible log message from a failed image synchronization

```
java.util.concurrent.ExecutionException: java.util.concurrent.ExecutionException:
com.ibm.venture.vm.VirtualizationShimException: Remote exception. No details
available.
shim=https://localhost/sdk_copy26 endpoint=https://localhost/sdk_copy26
localizedMessage="Remote exception. No details available.”
causeType=org.apache.axis.AxisFault cause=(0)nullCauses
```

Example 8-24 Possible log message from a failed image synchronization

```
Caused by: (0)null
at org.apache.axis.transport.http.HTTPSender.readFromSocket(HTTPSender.java:744)
at org.apache.axis.transport.http.HTTPSender.invoke(HTTPSender.java:144)
```

To resolve this issue, use a licensed version of VMware ESX for your VMware ESX cloud provider.

Cannot synchronize Windows 2008 R2 image

If an image is extended from another image that is imported from a Windows 2008 R2 virtual machine that is running, the image synchronization might fail.

When the synchronization fails, the image status is `virtual machine failed to start`. If you open the VMware vCenter console, a window is shown that includes the following error:

```
autochk program not found - skipping AUTOCHECK
```

The Image Construction and Composition Tool then rolls back the changes. The deployed virtual machine becomes Orphaned as a part of the rollback. The error pane then closes.

This problem is resolved by updating the VMware ESX 4.1 Update 2, and VMware ESX 5.0 Update 1. For more information, see the following article in the VMware Knowledge Base: *Cloned Windows 2008 R2 virtual machine fails to boot with the error: autochk program not found - skipping AUTOCHECK*, which is available at this website:

http://kb.vmware.com/selfservice/microsites/search.do?cmd=displayKC&docType=kc&externalId=2004505&sliceId=1&docTypeID=DT_KB_1_1&dialogID=675956945&stateId=1%200%2067596017

Timeout with Linux bundle in Windows image on VMware cloud provider

When a VMWare cloud provider is used, the provider might timeout with no error when a Linux bundle is synchronized inside a Windows image. This issue occurs because the installation script that is used in a software bundle is not recognized by Windows as a known executable file. However, the installation process that is run during synchronization runs without stopping. The synchronization task fails after the timeout.

The issue can be resolved when the timeout expires after six hours and the synchronization fails. You now can remove the incorrect Linux bundle and add the correct bundle.

Windows image synchronization failed

Windows image synchronization to the virtual machine can fail in relation to the `/etc/hosts` file. This file is required by the RXA protocol. If the `/etc/hosts` file is incorrectly configured in the Image Construction and Composition Tool server configuration, a failure occurs.

To resolve this issue, the machine where the Image Construction and Composition Tool is installed must contain an entry with the Image Construction and Composition Tool server IP, the related host name, and fully qualified host name. For example, the entry should look similar to the following `/etc/hosts` file entry:

```
9.10.10.235 ipas101 ipas101.in.ibm.com
```

8.6.2 Common problem troubleshooting

This section describes several possible issues and resolutions to help you determine the cause for virtual images failing during synchronization. The focus of this section is not on a specific piece of equipment but on issues that are more generic. To help you determine the cause of your issue, review the following common problems:

- Virtual image synchronization fails because the wrong key is specified
- Virtual image synchronization fails because of the return code value
- Virtual image synchronization failed and bundles that were used for synchronization are removed
- Synchronization of the virtual image takes a long time
- Virtual image synchronization fails after five hours for images
- Synchronization failed for a virtual image that was imported from a running virtual machine
- Synchronization fails after a new password is added
- By using the CLI, image synchronization does not work

Virtual image synchronization fails because the wrong key is specified

The virtual image synchronization process fails if it is configured with an incorrect private key file. When you are specifying your private key file and configuring IBM SmartCloud Enterprise as your cloud provider, you must specify a private key file with the `.rsa` extension. A common mistake is to specify a key file with the `.ppk` extension. If the wrong private key file is specified, the trace log file includes an error message similar to the following syntax:

```
Failed to connect to <IP_address>
```

To resolve this problem, ensure that you specify the private key file with the `.rsa` extension instead of `.ppk` and any other extension.

Virtual image synchronization fails because of the return code value

Virtual image synchronization fails if you use a script for the bundle installation that produces a return code value that is greater than 0 (zero). The Image Construction and Composition Tool checks the return value of the installation scripts used in bundles. A 0 return code indicates success and the synchronization process can proceed.

To resolve this issue, verify that the scripts that you are using to install the software return a 0 value to indicate successful installation. A value greater than 0 indicates a failure and your virtual image cannot be synchronized. The installation script checks for error conditions that are relevant to the installation of the bundle. For errors that might compromise the bundle installation, an exit code greater than `0` is returned, indicating failure. If a script returns no exit code explicitly, the exit code of the last command of the script is implicitly returned. Ensure that the bundle installation script can handle parameters that are passed in the following format:

```
scriptname -parameter1 <parameter1 value> -parameter2 <parameter2 value>
```

Example 8-25 shows a sample installation script.

Example 8-25 Sample script for installing software in a bundle with return code 0

```
#!/bin/sh
## Licensed Materials - Property of IBM
# (C) Copyright IBM Corp.  2011
# All Rights Reserved
# US Government Users Restricted Rights -Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
## ------------------------------------------------------------------
## Description: Sample install script for an ICON bundle
function usage ()
{
   echo ""
   echo " Usage :"
   echo "     installBundleSample.sh -parm1 \"parm1value\" -parm2 \"parm2value\""
   echo ""
   exit 1
}
echo "$0: === Starting Install of Bundle Sample === "
echo
echo "$0:    Arguments are: $* "
echo
#Init
MY_PARM1="Default parm1 value"
MY_PARM2="Default parm2 value"
echo "$0:    Parsing Arguments "
echo
#Read arguments
while [ $# -ne 0 ]
do
    case $1 in
    -parm1*)
       MY_PARM1=$2
       ;;
```

```
    -parm2*)
       MY_PARM2=$2
       ;;
    *)
       ;;
esac
shift 1
done
# Now perform the work and actually do the install stuff - check the return code
echo "$0:    About to do the real work..."
echo parm1 is "$MY_PARM1"      parm2 is "$MY_PARM2"
if [ $? -gt 0 ]; then
   echo "$0: echo $MY_PARM1 $MY_PARM2 failed" >&2
   exit 1
fi
# finish up and return zero to indicate everything went ok (non-zero return code
signifies an error)
echo " "
echo "$0: Done."
exit 0
```

Virtual image synchronization failed and bundles that were used for synchronization are removed

Sometimes when the synchronization fails, the bundles are removed. It is difficult to perform problem determination when the deployed image and the compressed package are removed. The difficulty is that you do not have access to the bundle installation scripts for troubleshooting.

To work around this issue, use the `ICCT_DEBUG` variable to troubleshoot the problem. Complete the following steps to enable the `ICCT_DEBUG` variable:

1. If the Image Construction and Composition Tool is started, use the following commands to stop the tool:

   ```
   cd <install_dir>
   ./stop.sh
   ```

2. Define the `ICCT_DEBUG` variable by using the following command:

   ```
   export ICCT_DEBUG=true
   ```

3. Restart the Image Construction and Composition Tool by using the following command:

   ```
   ./start.sh
   ```

Synchronization of the virtual image takes a long time

In this situation, you notice that the virtual image synchronization process takes a long time to complete, then it times out and fails. This condition occurs when cloud performance is poor. Cloud performance is not always consistent and can vary depending on usage and your connection. Depending on the cloud performance at any particular time, your virtual image synchronization can take longer to complete. The Image Construction and Composition Tool continues the synchronization process in case there are performance issues. If the synchronization process is slow and the performance is poor, the synchronization eventually times out. Then, you are notified that the synchronization failed because of performance issues.

To resolve this issue, after the virtual image synchronization process times out, you can retry the synchronization. If you retry and are still having synchronization issues, check the log files to determine the particular cloud provider issue that is affecting synchronization.

Virtual image synchronization fails after five hours for images

At times, the virtual image synchronization fails after five hours for images on the IBM PureApplication System and IBM SmartCloud Provisioning cloud provider. The synchronization can fail after five hours even if the image deployment is still running on IBM PureApplication System. This event happens because, by default, the Image Construction and Composition Tool waits up to five hours for the deployment of an image on IBM PureApplication System or IBM SmartCloud Provisioning. If the deployment does not complete within that time, the Image Construction and Composition Tool returns a failure.

To resolve this issue, make changes in the Image Construction and Composition Tool configuration file regarding the wait time. Complete the following steps to make these changes:

1. Open the `<ICCT_INST_DIR>/icn.app/config/configuration.config` file.
2. Edit the `maxIWDDeployWaitTime` parameter to increase the wait time. The default value is 300 minutes (5 hours).
3. If Image Construction and Composition Tool is started, stop it by using the following commands:

   ```
   cd <install_dir>
   ./stop.sh
   ```

4. Start the Image Construction and Composition Tool to apply the changes by using the following command:

   ```
   ./start.sh
   ```

Synchronization failed for a virtual image that was imported from a running virtual machine

When you are importing a virtual image from a running virtual machine, the virtual image does not synchronize if Network Manager is running on the virtual machine. Network Manager must be disabled on the virtual machine.

To resolve this issue, you must disable Network Manager on the virtual machine and import the virtual image again. Complete the following steps to disable the Network Manager:

1. From the CLI, run the following command:

   ```
   # service NetworkManager stop
   ```

2. Run the following command:

   ```
   # chkconfig NetworkManager off
   ```

3. Open the `/etc/sysconfig/network-scripts/ifcfg-eth0` file and verify that the interfaces are not controlled by Network Manager. Check the following parameter for this setting:

   ```
   NM_controlled = no
   ```

Synchronization fails after a new password is added

When you extend and synchronize an image by using a PowerVM Express cloud provider, the synchronization fails after you add a new password in the synchronization dialog.

To resolve this issue, when you extend and synchronize the image, you must enter the root user password of the original image. Do not enter a new password during the synchronization.

By using the CLI, image synchronization does not work

If you attempt to use the CLI to synchronize an image with an IBM PureApplication System or IBM SmartCloud Provisioning cloud provider, the operation fails.

Complete the following steps to resolve this issue:

1. To get a list of all parameters, run the following command:

   `allCloudSyncParams = image.getSynchronizeParametersFromCloudProvider()`

2. You receive an output from running the command in Step 1. The output is a list of parameters. Based on the retrieved list, build a new JSON object with a structure similar to the structure that is shown in Example 8-26.

   Example 8-26 JSON object structure

   ```
   myCloudSyncParams =
     [
       {'name':'profileList', 'type':'string', 'value':''},
       {'name':'envprofile', 'type':'string', 'value':'10'},
       {'name':'cloudgroup', 'type':'string', 'value':'12'},
       {'name':'ipgroup', 'type':'string', 'value':'1013'},
       {'name':'username', 'type':'string', 'value':'root'},
       {'name':'password', 'type':'string', 'value':'passw0rd'}
     ]
   ```

   This JSON object includes the following features:

   – The `profileList` element has an empty value.

   – You must add the user name and password elements to the JSON object. They are not returned by the **`getSynchronizeParametersFromCloudProvider()`** method.

3. To get a list of all parameters, run the following command:

   `myImgSyncParams = getSynchronizeParametersFromImage()`

4. To synchronize the image, run the following command:

   `image.synchronize(myImgSyncParams, myCloudSyncParams, True)`

9

High Availability and Disaster Recovery

This chapter describes high availability and disaster recovery aspects that are related to IBM PureApplication System. It includes different approaches and scenarios to demonstrate how high availability and disaster recovery can be achieved with IBM PureApplication System.

The following topics are covered in this chapter:

- High Availability and Disaster Recovery overview
- Workload failover and recovery scenarios
- Hardware high availability
- High availability scenarios
- Disaster recovery approaches
- Current limitations

© Copyright IBM Corp. 2013. All rights reserved.

9.1 High Availability and Disaster Recovery overview

As IT becomes more pervasive in businesses and our daily lives, the impact of system downtime is increasingly significant. Outages affect productivity, create inconveniences, and result in loss of business. The globalization of the economy places a new demand for systems to be highly available and for the elimination of unplanned outages. If failures happen, systems should be able to recover rapidly and without losing data.

The causes of downtime might result from planned or unplanned events. The following types of downtime are possible:

- Planned downtime: Includes hardware upgrades, software updates, fix packs installations, and other maintenance work. Planned downtime often can account for a significant amount of the total downtime of a system.
- Unplanned downtime: A result of hardware failures, software bugs, infrastructure problems, human errors, site disasters, and natural calamities. Such unplanned outages are costly with significant impacts and customers want to reduce these outages.

The availability of a system is measured by the percentage in time that the application is available to a user. The application is considered highly available as it approaches the 99.999% of availability, commonly referred to as *the five 9s of availability.* The resulting availability is the combination of the availability of all components of a solution, such as the infrastructure layer, hardware, operating system, middleware, and applications.

High Availability (HA) is required when applications cannot undergo an unplanned outage for more than a few seconds or minutes at a time, but can tolerate short periods of not being available, or can be down for a few hours for scheduled maintenance. The system or component is equipped to handle faults in an unplanned outage gracefully to continue providing the intended functionality.

Continuous Availability (CA) refers to the ability of a system or component to be operational and accessible always, not allowing any kind of outage. Although HA compensates for unplanned outages, CA handles planned and unplanned outages. Therefore, CA extends HA with continuous operation.

Disaster Recovery (DR) is the ability to quickly reconstruct and start the applications in an alternative physical site if the primary data center has some catastrophic loss and cannot continue to run the application for an extended period. It is the process of bringing the servers and applications online in a priority order to support mission-critical applications at an alternative site. The alternative site need not to be the same physical size as the primary data center because its role is to quickly get the mission critical application up and running.

HA and DR in IBM PureApplication System is accomplished by using tools and procedures. IBM PureApplication System changes the automation aspects that make these procedures repeatable and easier for customers. In the following sections, we describe some fundamental concepts with regards to HA and DR.

9.1.1 Failover strategies

The key for implementing an HA system is to identify and eliminate single points of failure through redundancies, clustering, and other failover mechanisms. Redundancies mask system and component failures from users. The level of transparency for failure masking depends upon the following failover strategies:

- Cold standby: In this failover strategy, the primary component runs actively while the secondary or backup component stays dormant. When the primary component fails, the secondary component is activated to assume an active role. The interruption is visible to users.
- Warm standby: In this failover strategy, the primary component runs actively with the secondary or backup component that is running without active participation in workload management. When the primary component fails, the secondary component assumes an active role. Because the secondary component was running passively with partially synchronized data, the fail over is faster than a cold standby. There is still a minimal interruption to users.
- Hot standby: In this failover strategy, the primary and secondary components are actively running as a single, unified system. Active data replication happens between the primary and secondary components. When the primary component fails, the secondary component continues functioning without interruption to users.

9.1.2 Software clustering modes

Clustering is a technique that is used to create multiple copies of software components that are running actively and collaborating seamlessly to present a single, unified system. IBM WebSphere Application Server, which is included on PureApplication System, has inherent support for clustering. The following modes of clustering are available:

- Vertical clustering*:* In vertical clustering, multiple copies of the components are running actively on the same physical machine. The components are hot standby or in active replication. This configuration means that failure of one component presents no visible interruption to users. This clustering technique optimizes usage of the machine, although the single physical machine presents a single point of failure.
- Horizontal clustering: In horizontal clustering, components are created on multiple physical machines. This clustering mode is optimized for performance because only one instance of the software is present on each physical machine. It achieves similar software redundancy as in vertical clustering, with the other benefit of hardware redundancy.
- Vertical and horizontal clustering: Combines vertical and horizontal clustering techniques and maximizing the use of individual physical machines to achieve HA, performance throughput, and scalability.

These modes of clustering are shown in Figure 9-1.

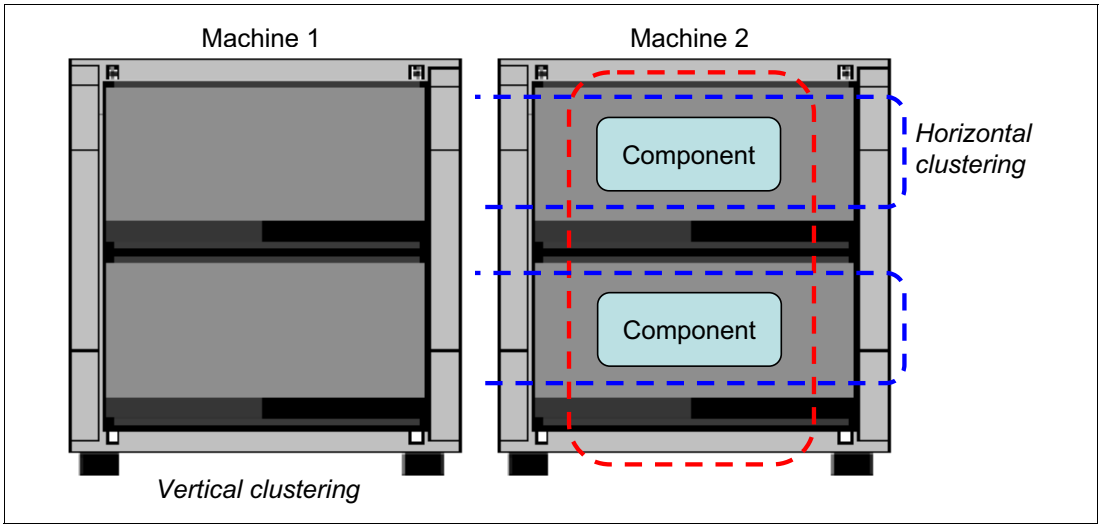


Figure 9-1 Vertical and horizontal software clustering modes

9.1.3 HADR solution components

A complete high availability and disaster recovery solution incorporates *data resiliency*, *infrastructure resiliency*, and *application state resiliency* into one integrated environment that addresses one or all of the outage types. The behavior of a solution to a customer depends upon the inclusion and implementation of these three basic elements into the clustering configuration. These elements are described next.

Data resiliency

Applications require access to data or copies of the data to perform business-critical operations. Therefore, data resiliency is the base or foundational element for a high-availability and disaster recovery solution deployment. Data resiliency across multiple nodes in a cluster is the foundation for building an effective high-availability solution.

Local storage replication is the most commonly used technique for deploying cluster data resiliency. The following general topologies are used for storage-based resiliency:

- Active storage sharing (Active-Active): In this arrangement, the nodes of the cluster have simultaneous (concurrent) read/write access to the shared data. The cluster management technology performs locking operations to ensure that only one node can perform an update or write operation at a time. The benefit of this approach is that no switching operation is associated with storage resiliency because all the nodes simultaneously own the shared resource. If a node outage occurs, another node in the cluster resumes production through a reassignment process.
- Shared-disk configuration (Active-Passive): In this arrangement, only one node in the cluster performs read/write operations to the disks. Ownership of those resources can be passed to other nodes in the cluster as part of a failover (or rollover) operation. The operating system, application, and data are all switched between nodes. The recovery point is established by applying the journals that are in the shared disk resources. The recovery time is associated with the time it takes to apply the journal.

These topologies can be extended further for geographic dispersion by using host-based or storage server-based replication technology. Data in a storage pool can be replicated in a synchronous manner for zero loss implementations. It also can be replicated in an asynchronous manner for geographically dispersed sites where latency might affect operations.

Infrastructure resiliency

Infrastructure resiliency provides the overall environment that is required to resume full production at a standby node. This environment includes the entire list of resources that the application requires upon failover for the operations to resume automatically.

For an automated or semi-automated failover operation to work, all of the resources that the application requires to function on the primary node also must be present on the secondary node. These resources include the following items:

- Dependent hardware
- Middleware
- Network connectivity
- Configuration files
- Attached devices
- Security profiles
- Application-specific resources
- Application data

The cluster constantly monitors the resilient infrastructure resources for changes that indicate a failure, a pending failure, or a possible configuration change that might cause a cluster operation (such as a failover) or an operator to take corrective action. An aspect of monitoring is performing periodic verification checks. In these checks, specific or custom scans of the cluster resources are conducted to assess status against the intended configuration.

In addition to real-time monitoring, these operations are performed as an integrity check that supersedes the real-time monitoring function. A modern high-availability solution automatically identifies changes and addresses them through auto-corrective features or notifications methods. In addition to the monitoring and verification capabilities, cluster-wide management functions should be available to operators. To maintain or update the infrastructure, it is important to enable the operators to perform various operations on behalf of the application and operating system.

Application state resiliency

Application state resiliency is characterized by the application recovery point as described when the production environment resumes on a secondary node in the cluster. Ideally, the application resumes on an alternative node at the last state where the application was on the primary system when a failure occurred. Practically speaking, the characteristic of the application to resume varies by application design and customer requirements.

While application state resiliency depends on many characteristics of the environment, a high-availability solution should aid in monitoring the health of the application stack. This solution also should provide for corrective actions in the cluster to reduce failover times and to accelerate recovery times. For example, some middleware in a high-availability configuration might create a cache of application state information about another node in the cluster apart from the active node, thus enabling a quicker failover.

9.1.4 Disaster recovery models

When a disaster recovery model is chosen, organizations traditionally relied on the level of service that is required, as measured by the following recovery objectives:

- Recovery Time Objective (RTO): The amount of time between an outage and the restoration of operations.
- Recovery Point Objective (RPO): The point in time where data is restored and reflects the amount of data that is ultimately lost during the recovery process.

Traditionally, the following main disaster recovery models are available:

- Dedicated model

  In a dedicated model, the infrastructure is dedicated to a single organization. This type of disaster recovery can offer a faster time to recovery compared to other traditional models because the IT infrastructure is mirrored at the disaster recovery site and is ready to be called upon in the event of a disaster. While this model can reduce RTO because the hardware and software are preconfigured, it does not eliminate all delays. This approach is also costly because the hardware sits idle when it is not used for disaster recovery. To mitigate the cost of ownership, some organizations use the backup infrastructure for development and testing, but that option introduces more risk into the equation.

- Shared model

  In a shared disaster recovery model, the infrastructure is shared among multiple organizations. Shared disaster recovery is more cost effective because the off-site backup infrastructure is shared between multiple organizations. After a disaster is declared, the hardware, operating system, and application software at the disaster site must be configured from the ground up to match the IT site that declared a disaster. This process can take hours or even days.

With dedicated and shared disaster recovery models, organizations are forced to make trade-offs between cost and speed to recovery. As the pressure to achieve continuous availability and reduce costs continues to increase, organizations can no longer accept trade-offs. Disaster recovery originally was intended for critical batch and back-office processes. Now, many organizations are dependent on real-time applications and their online presence as the primary interface to their customers. Any downtime reflects directly on their brand image. An interruption of key applications such as e-commerce, online banking, and customer self service is viewed as unacceptable by customers.

9.2 Workload failover and recovery scenarios

This section describes the workload failure and recovery scenarios for the different deployment models in PureApplication System.

9.2.1 Virtual applications: WebSphere Application Server

The basic assumption for applications that are running on WebSphere Application Server in a virtual application deployment model is that the application is considered *non-persistent*. This state means the application can run on any instance of WebSphere and there is no dependency that the state is locked in a specific instance. WebSphere Application Server instances can be started, stopped, created, and deleted as needed, sometimes without human intervention and with no impact on the running application.

For example, in a failure of a WebSphere instance, PureApplication System can provision another WebSphere instance in another virtual machine. If a WebSphere instance fails within a virtual machine and the VM is still running, PureApplication System monitors WebSphere process failures and restarts WebSphere once a day to prevent spinning of middleware. If scaling is enabled, PureApplication System starts another instance if the service level agreement (SLA) is not satisfied. There is no action that is needed by the deployer. At some point, the deployer can determine the cause of the failure by using the standard troubleshooting methods.

However, if the VM fails, PureApplication System detects the VM failure and respins another VM, assigning a new IP address. If scaling, routing, or caching policies are enabled, PureApplication System links the instance to the appropriate shared services. Again, there is no action that is required by the deployer. PureApplication System handles the recovery. The best practice is to have scaling enabled so that PureApplication System can manage the recovery to satisfy the SLAs that are required by the application.

9.2.2 Virtual systems: WebSphere Application Server without IMP

The failure and recovery of a WebSphere Application Server instance that is running in the virtual system deployment model without Intelligent Management Pack (IMP) is handled by the built-in capabilities of the middleware. PureApplication System does not monitor the WebSphere processes within the virtual machines in the virtual system deployment model. Normal troubleshooting and debug practices must be followed.

If a WebSphere Application Server node in Network Deployment topology fails, the Node Agent restarts the WebSphere node. This process is normal WebSphere function. If the Node Agent or the Deployment Manager fails, the deployer must follow normal WebSphere debug procedures to determine and fix the failure. The deployer can look at WebSphere log files, log in to the VM and restart WebSphere, debug from within the VM, or review the monitoring data for potential cause of failures.

If the VM that is running WebSphere fails, PureApplication System detects the VM failure, but there is no VM recovery. Deployers can try to restart the VM from the PureApplication System workload console and add more nodes by cloning the deployed instance while they debug the root cause of the failure. Table shows the different types of WebSphere failures for virtual system pattern scenario and the corresponding actions that are taken by PureApplication System and the deployer.

9.2.3 Virtual systems: WebSphere Application Server with IMP

The failure and recovery of a WebSphere Application Server instance that is running in the virtual system deployment model with IMP is handled by the built-in capabilities of the middleware. PureApplication System does not monitor the WebSphere processes within the virtual machines in the virtual system deployment model. Normal troubleshooting and debug practices must be followed.

The IMP provides dynamic clustering capability from the WebSphere Virtual Enterprise product. This capability allows WebSphere to provision more servers to satisfy SLAs, or when excess capacity is present, remove some servers and still maintain the SLAs when the load fails. For a dynamic cluster, if the SLA is not satisfied, IMP restarts WebSphere or creates more WebSphere cluster members, which are based on the settings. If a static cluster is used in WebSphere with IMP enabled, this scenario is the same as WebSphere with no IMP, as described in the previous section. For Deployment Manager failure, IMP can support backing up the Deployment Manager if the shared file system is configured, as explained in the WebSphere Virtual Enterprise information center, which can be found at this website:

http://pic.dhe.ibm.com/infocenter/wveinfo/v7r0/topic/com.ibm.websphere.virtualenterprise.doc/welcome_70_ve.html

If the VM that is running WebSphere fails, PureApplication System detects the VM failure but there is no VM recovery. Deployers can try to restart the VM from the PureApplication System workload console and add more nodes for static clusters by cloning the deployed instance. Adding nodes and restarting procedures can be done while they debug the root cause of the failure. Table shows the different types of WebSphere failures in the virtual system pattern scenario and the corresponding actions that are taken by PureApplication System and the deployer.

9.2.4 Virtual applications: DBaaS

The virtual machine for Database-as-a-Service (DBaaS) as part of a virtual application deployment is considered persistent. This configuration means that an individual database instance features non-replicated data and stores state. Therefore, in case of failure, PureApplication System cannot spin another database instance. Also, DBaaS as part of a virtual application is not scalable in the current release.

PureApplication System does not monitor DB2 failures. If there is a DB2 failure and the VM is still running, PureApplication System does not attempt to restart DB2. The deployer must review DB2 logs to identify the failure by accessing the VM by way of SSH and, if needed, restarting DB2.

If the VM that contains DB2 fails, PureApplication System detects the VM failure and restarts DB2 VM once at the same IP address because DB2 is considered persistent. If the VM does not come up after one retry, the deployer must create a DBaaS instance VM and then restore DB2 data from Tivoli Storage Manager backup. For use cases where data loss cannot be tolerated, periodic backups are essential. Table shows the different types of DB2 failures in the virtual application pattern scenario and the corresponding actions that are taken by PureApplication System and the deployer.

9.2.5 Virtual systems: DB2

The failover and recovery of DB2 that is running in virtual system deployment model is similar to traditional DB2 failure scenarios. It is handled like normal DB2 failures. PureApplication System does not take any special action and normal DB2 troubleshooting techniques should be applied.

If the DB2 failed but the VM is still running, PureApplication System does not monitor DB2 failures and therefore does not attempt to restart DB2. Clients must view the DB2 logs and log in to the DB2 VM to fix or restart DB2. Normal DB2 best practices and recovery processes apply to a DB2 that is running in PureApplication System. PureApplication System contains DB2 Enterprise and DB2 High Availability Disaster Recovery (HADR) images. For clients that need HADR capability, they can use the primary and secondary DB2 HADR virtual image parts.

If the VM containing DB2 instance failed, PureApplication System detects VM failure, but there is no restart of VM in virtual system. The client must review the logs, determine the failure, and restart the VM or create a VM. Standard DB2 backup and restore must be implemented. DB2 HADR can help here where the secondary DB2 can take over processing the requests. Table on page 385 shows the different types of DB2 failures in the virtual system pattern scenario and the corresponding actions that are taken by PureApplication System and the deployer.

9.2.6 Caching shared service

Caching shared service is provided through a pool of VMs for scale and redundancy. The number of VMs depends on the scaling policy of caching service, which includes auto-scaling that is based on the percentage of cache in use. Caching shared service uses WebSphere eXtreme Scale, which is robust. Internally, eXtreme Scale has its own processes that monitor the caching shared services.

If a caching shared service fails within a VM and the VM is still running, PureApplication System restarts the caching service. There is no action that is needed by the deployer. Caching VM is considered a persistent VM. Therefore, if a VM fails, PureApplication System detects the VM failure and restarts the caching service at the same IP address. There is no action that is needed by the deployer. If a primary or replica VM fails, it is restored and the cache is populated. PureApplication System handles the failures. Table shows the different types of caching shared service failures and the corresponding actions that are taken by PureApplication System and the deployer.

9.2.7 Proxy shared service

Proxy shared service supports scaling and the deployer can specify manual or auto scaling. These sets of proxy servers typically are front ended by an external load balancer. PureApplication System does not monitor proxy shared service failures. If the proxy shared service instance fails but the VM is still running, PureApplication System does not restart the proxy service.

However, if the client has multiple proxies that are defined through auto scaling, other proxy servers can be started that are based on the SLA. In the meantime, deployers can troubleshoot the problem and try to restart the VM. Because the proxy service is considered persistent, if the VM containing the proxy service fails, PureApplication System detects the VM failure and restarts the proxy service at the same IP address.

As a best practice, clients can use fixed IP address for the proxy services VM or have a separate smaller pool of IP groups for the proxy service. These sets of IP addresses then can be given to the front end load balancer to spray the requests. One way to achieve this configuration is to have a separate environment profile only for the proxy service and then use that for deployment of proxy shared services. Table shows the different types of proxy shared service failures and the corresponding actions that are taken by PureApplication System and the deployer.

9.2.8 Custom image

Custom images can be created by using the IBM Composition and Construction Tool or by using the PureApplication System extend and capture function. If the client uses one of the IBM supplied images as the base virtual image for creating a custom image, it includes the IBM Tivoli Monitoring OS agent by default to monitor the base OS.

All failover and recovery functions must be handled by the client in the image. If a custom image VM fails, PureApplication System does not monitor, respin, or restart the VM or any custom software within the VM. Clients are responsible for providing the failover and recovery mechanism for the software components that they add to the image.

9.3 Hardware high availability

Fault tolerance refers to the ability of a system to continue operation in case of failure of one or more of its components. This term often is referenced with high availability, particularly at the hardware level. Hardware redundancy (the addition of extra hardware for fault detection and toleration) is commonly employed in fault tolerant systems.

PureApplication System has built-in hardware redundancies for failover and recovery. So, if one of the following components fails, there is redundant hardware that assumes and prevents the system from interrupting the service:

- Management and virtualization nodes: Both of these nodes feature redundant backup servers that are continuously kept in sync with each other. PureApplication System has a floating IP address that is used to access the PureApplication System management functions. If one of the management nodes fails, the floating IP address is automatically assigned to the backup server and requires no change by the clients to access the management function.
- Network controllers: For the network controllers, there are redundant switches and cabling. Failure of one of the switches leads to reduced bandwidth; however, the rack continues to function.
- Storage and storage controllers: For storage, solid-state drives (SSDs) and hard disk drives (HDDs) are configured in RAID5, plus one spare. Therefore, RAID5 tolerates two concurrent drive failures without data loss. For the storage controllers, each controller has two canisters that can service all of the traffic to storage. If one of the canisters fails, the other handles the I/O.
- Compute nodes: If a compute node fails, PureApplication System attempts to move the virtual machine to another compute node that has free resources to accept the VM. The VM is moved within the compute nodes that belong to the same cloud group. This process is called *rebalance* or *workload evacuation*. If resources that are available on other nodes to handle the extra load is limited, the virtual machines are started based on their priorities. Appropriate messages and events are displayed for VMs that cannot be moved because of lack of space in other compute nodes.

9.4 High availability scenarios

IBM PureApplication System has built-in high availability features and is designed to not have a single point of hardware failure within the rack. However, to maximize these features, the software architectures and middleware topologies that are running on PureApplication System must be carefully planned and analyzed so that no aspect of the design is forgotten.

The following high availability features must be considered:

- Those features that are internal to the rack or intra-rack
- Those features that are across racks or inter-rack

In this section, we describe some of these scenarios for implementing high availability on PureApplication System.

9.4.1 Intra-rack high availability

In this scenario, a single PureApplication System rack is used to provide intra-rack high availability. A standard golden topology, including WebSphere and DB2 virtual system deployments that are running entirely inside the rack, is protected from failure by any one piece of hardware, such as a compute node, storage, or TOR switch.

As an example, when a compute node fails, the following actions occur as responses:

- The WebSphere middleware detects the failure of the Java VM and seamlessly routes the traffic to the other cluster members within the WebSphere cell.
- PureApplication System detects the compute node failure and moves the WebSphere VM to another compute node. After it is fixed, this node eventually is rejoined in the cluster by the plug-in and starts taking traffic again.
- For DB2 HADR deployments, if the primary database fails, the secondary DB2 is used to serve the requests.

The placement algorithms of PureApplication System are intelligent enough that, in most cases, two cluster members are never placed on a single compute node. This configuration is dependent on the configuration of the cloud group and the availability of compute resources within the cloud group. Figure 9-2 on page 388 shows a standard virtual system topology, including IBM HTTP Server (IHS), WebSphere Application Server, and DB2 HADR, which is enabled for intra-rack high availability.

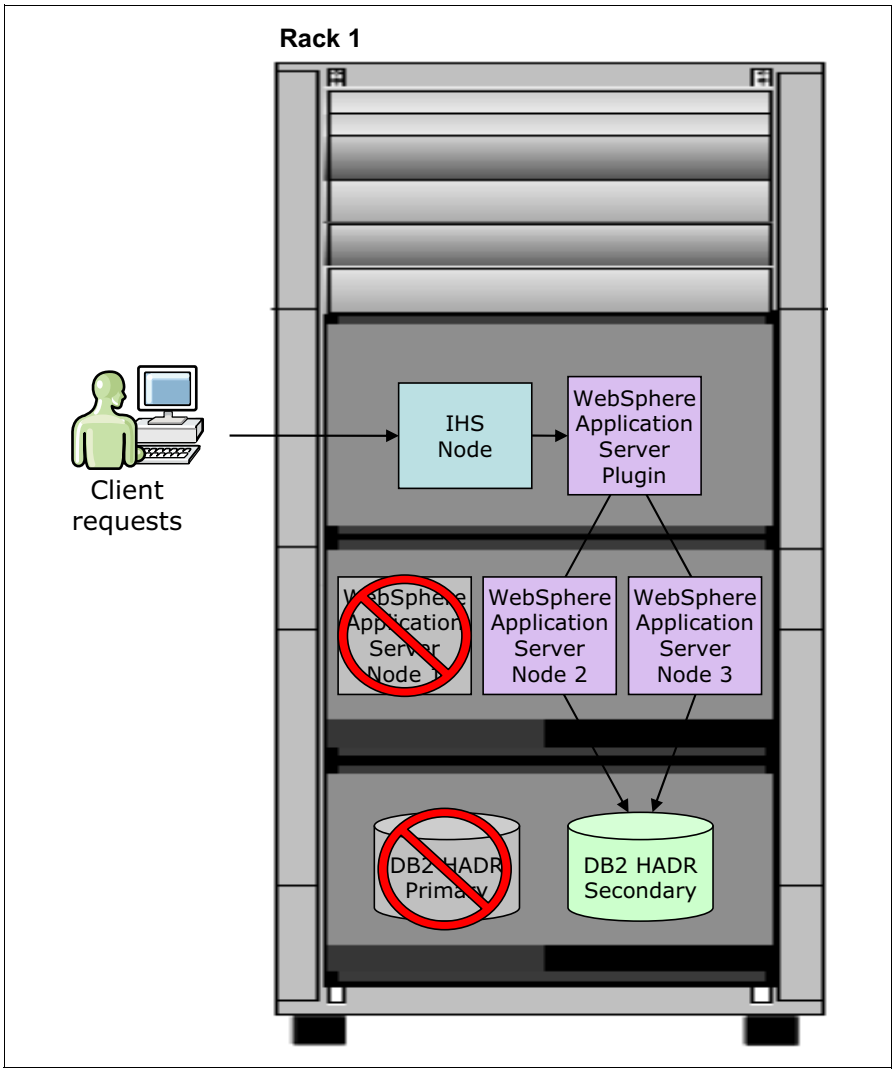


Figure 9-2 Standard virtual system topology that is enabled for intra-rack high availability

Intra-rack HA does not help if the entire rack fails because of some catastrophic conditions in the data center. That fact is where inter-rack high availability becomes important. You can again take full advantage of WebSphere, DB2, and other middleware facilities to provide this level of service.

9.4.2 Inter-rack high availability

This section describes some of the high-availability features across multiple racks. Standard middleware high-availability features are used to support high availability across racks. Many IBM middleware, such as WebSphere Application Server, DB2 Enterprise, and others include high-availability features that can be maximized in the same way as is traditionally done outside PureApplication System.

For example, to take advantage of the built-in HA features in WebSphere Application Server, you can provide one of the following failover topologies:

- Create WebSphere cluster nodes in the first rack and federate them to a Deployment manager (Dmgr) in the second rack, as shown in Figure 9-3.

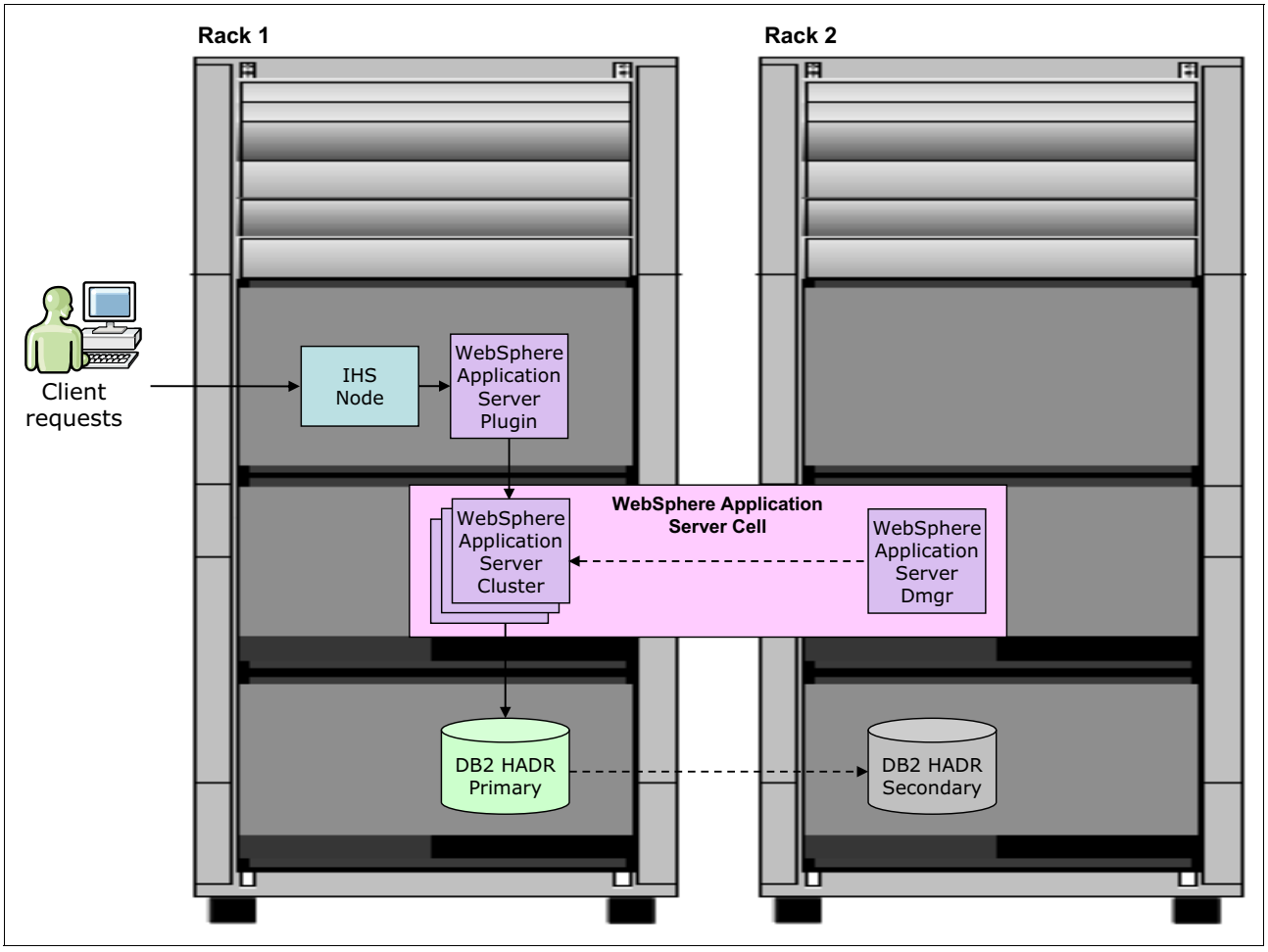


Figure 9-3 High availability topology with separate WebSphere Dmgr and cluster nodes

- Create two separate WebSphere cells (one per rack) and manage load distribution between the cells by using an on-demand router external to the two racks, as shown in Figure 9-4.

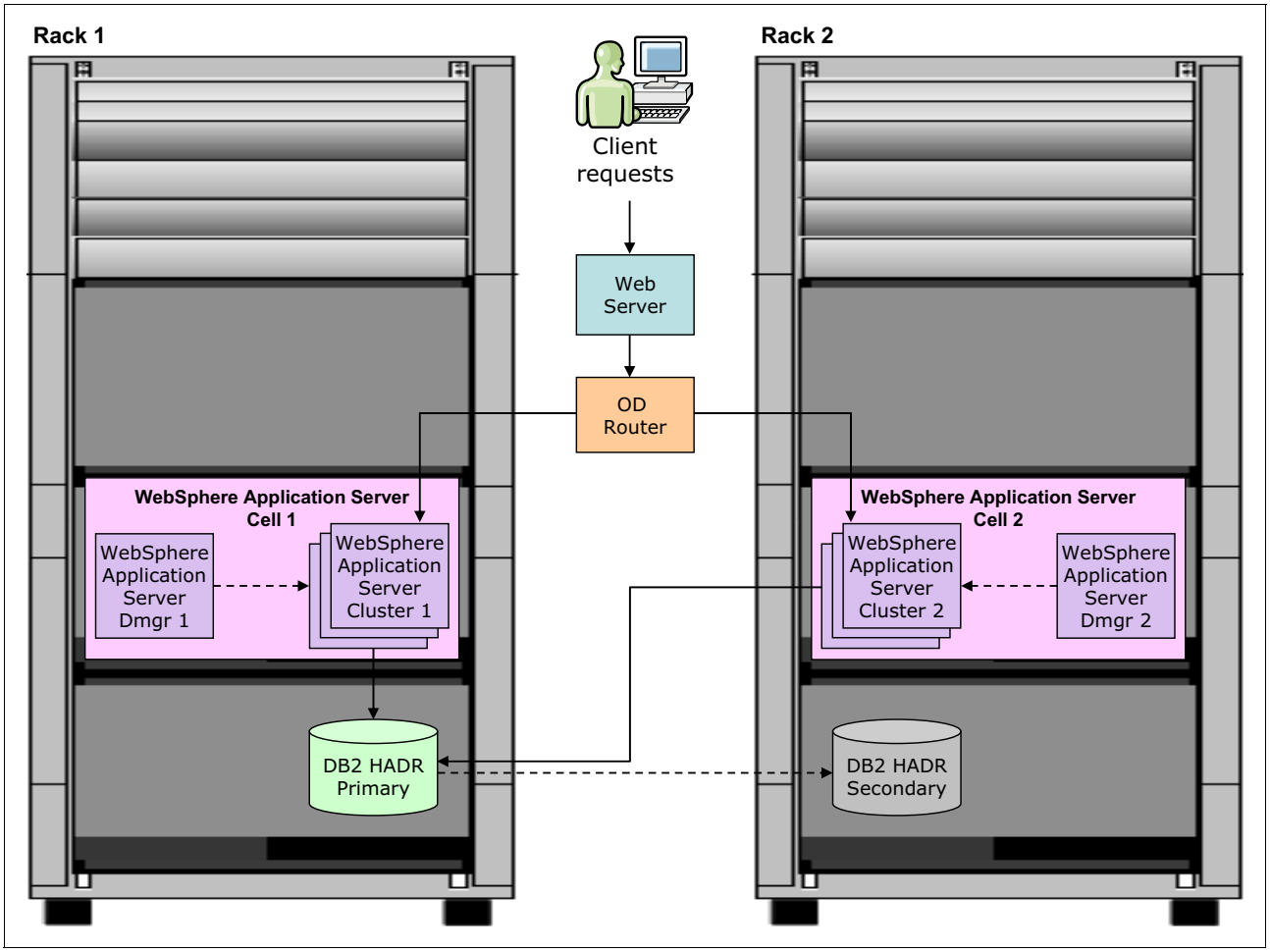


Figure 9-4 High availability topology with two separate WebSphere cells and external ODR

Both topologies must use standard DB2 HADR patterns to set up a primary database on a rack, with the backup database on the other (Active-Passive mode). For more information about these use cases, see 9.4.3, “Inter-rack high availability within the same data center” and Section 9.4.4, “Inter-rack high availability across distributed data centers” on page 392.

9.4.3 Inter-rack high availability within the same data center

In this high availability inter-rack scenario, which is referred to as the *single cell* model, a virtual system pattern that defines a WebSphere network deployment cell is created. It consists of one Deployment manager (Dmgr), one IBM HTTP Server (IHS) node, and WebSphere Application Server cluster nodes in the first rack, named Rack 1.

A virtual system pattern also is created on the second rack, named Rack 2. Rack 2 contains an IBM HTTP Server node and WebSphere Application Server cluster nodes that are federated to the Deployment manager that is running on Rack 1. The WebSphere Application Server HTTP plug-in for the IBM HTTP Server node is also present in the two racks. The DB2 HADR database is configured in an Active-Passive mode, with the primary (active) node on Rack 1 and the secondary (passive) node on Rack 2. This configuration defines the cell boundary as crossing both machines, as shown in Figure 9-5.

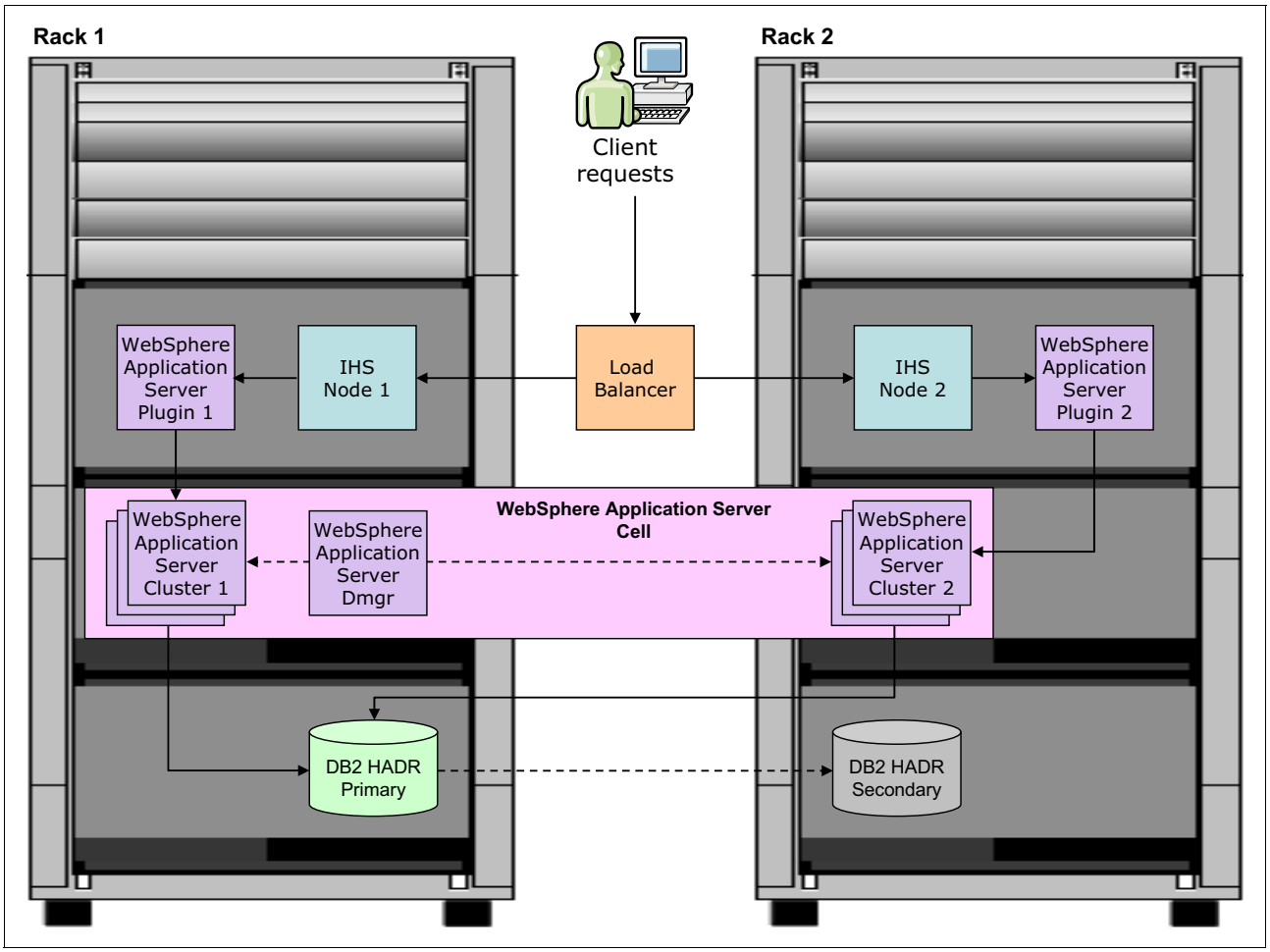


Figure 9-5 Single cell model across two racks in the same data center

It is necessary to create a virtual system pattern for the primary DB2 HADR node in Rack 1, and a second virtual system pattern for the secondary DB2 HADR node in Rack 2. For this configuration to work, it is necessary to configure an external load balancer to be aware of all of the IBM HTTP Server instances in the two racks. Deployers also must consider HTTP session management across the two racks. The simplest case in this approach is to enable database session persistence to the shared database.

In this configuration, IBM PureApplication System is now tolerant of a complete failure of either rack. The racks work in the following manner during a failover event:

- If Rack 1 fails, IBM HTTP Server instances and WebSphere Application Server cluster nodes on Rack 2 continue to take requests from the external load balancer. The DB2 HADR secondary instance takes over from the failed primary node. The only function that is lost is the ability to deploy changes to the cluster members on Rack 2 because the Deployment manager is no longer available.
- If Rack 2 fails, Rack 1 continues to function as normal and take requests from the external load balancer. Because the Deployment manager is in Rack 1, even the ability to deploy changes to the cluster members is kept.

9.4.4 Inter-rack high availability across distributed data centers

The case of two geographically separated PureApplication System racks is a bit more complicated in arrangement. The communication that is necessary between the Deployment manager and its cluster members (for management, code distribution, and so on) is inefficient over a wide area network. As a best practice, it is not recommended to federate cells across long distances. The recommendation is to create two cells, instead of joining all of the instances into a single cell, as shown in the previous scenario.

In this scenario, which is referred to as a *dual-cell* model, the deployer must create at least two individual cells by using a shared database, as shown in Figure 9-6 on page 393. As noted, the cell boundaries are entirely contained within each rack. In this way, the WebSphere cells are configured in an Active-Active mode, but the DB2 HADR database is configured in an Active-Passive mode (DB2 settings are same as arranged in the single cell model). The difference here is that the cells are independent from each other. Also, there is no inter-rack communication between the WebSphere Application Server cluster nodes, as shown in Figure 9-6 on page 393.

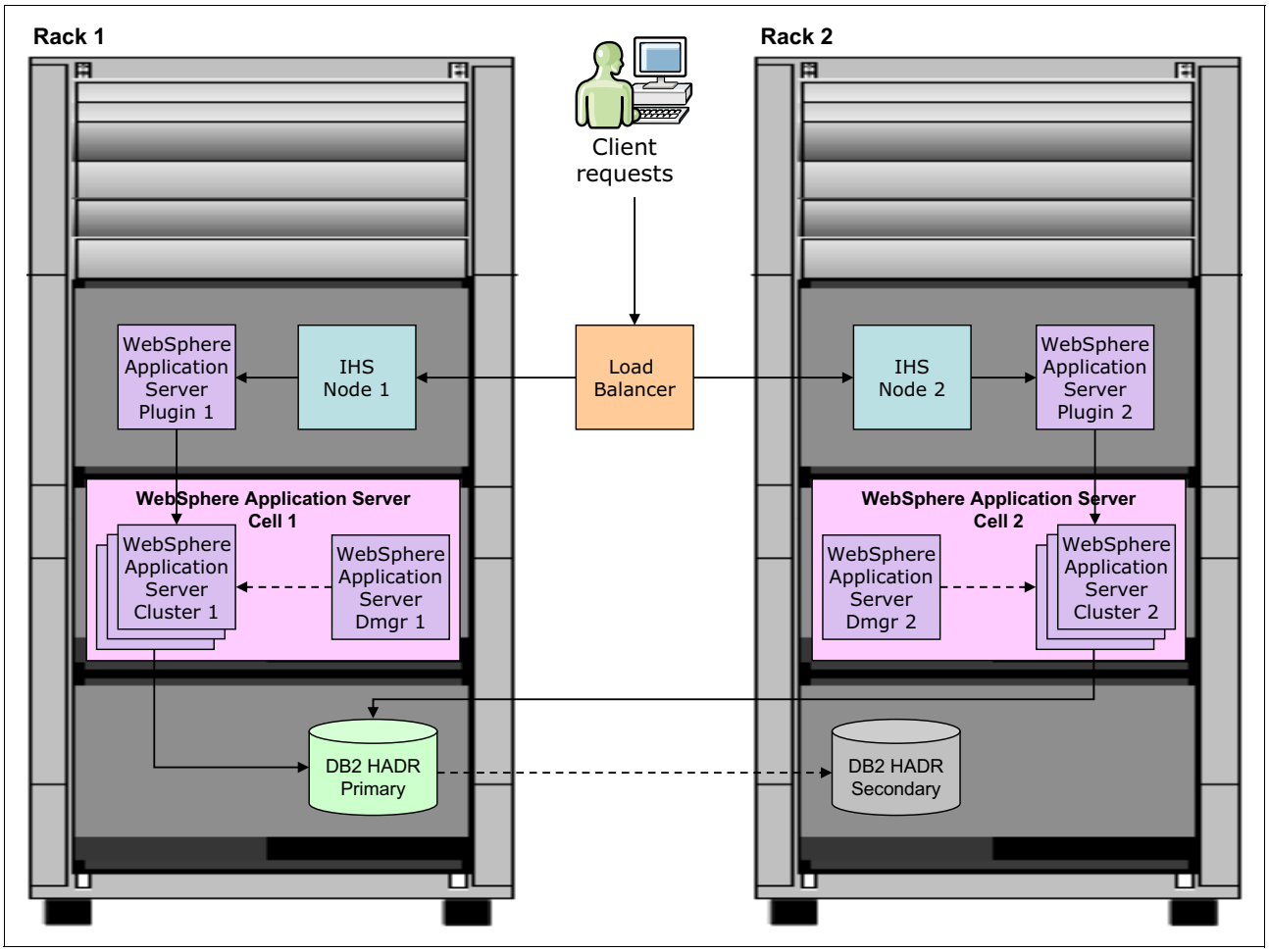


Figure 9-6 Dual cell model across two geographically distributed data centers

Complete the following steps to implement this approach:

1. Create the first cell with a virtual system in Rack 1 and export the virtual system pattern.
2. Import the virtual pattern into Rack 2 and create an instance of that pattern in Rack 2.
3. Use the deployer to create a DB2 HADR primary by using a virtual system pattern in Rack 1.
4. Create a DB2 HADR secondary with another virtual system pattern in Rack 2.
5. Set the external load balancer to feed traffic to the full set of IBM HTTP Server instances in both cells.

If either rack fails completely, the other rack continues to take traffic uninterrupted.

The use of HTTP session replication across two cells by using a shared database is possible, but it is rarely done. In most cases, session affinity is configured in the external load balancer. Requests for a session that was started in a particular cell always is outed to that cell. If you can tolerate lost session data in cases of a failover, you can set up a separate local database for session persistence.

9.5 Disaster recovery approaches

The following sections describe the different disaster recovery approaches that are available on PureApplication System for backing up and restoring virtual applications, virtual systems, and virtual appliance patterns.

9.5.1 Virtual application pattern backup and restore

Virtual application patterns can be exported and imported from the GUI console and the command-line interface (CLI), as shown in the following sections.

Back up by using the GUI console

Complete the following steps to back up a Virtual application pattern by using the GUI console:

1. Log in to the IBM PureApplication System GUI console.
2. Select **Workload Console**.
3. Select the virtual application pattern by clicking **Patterns** → **Virtual Applications**.
4. Click the **Export** icon from the toolbar, as shown in Figure 9-7.

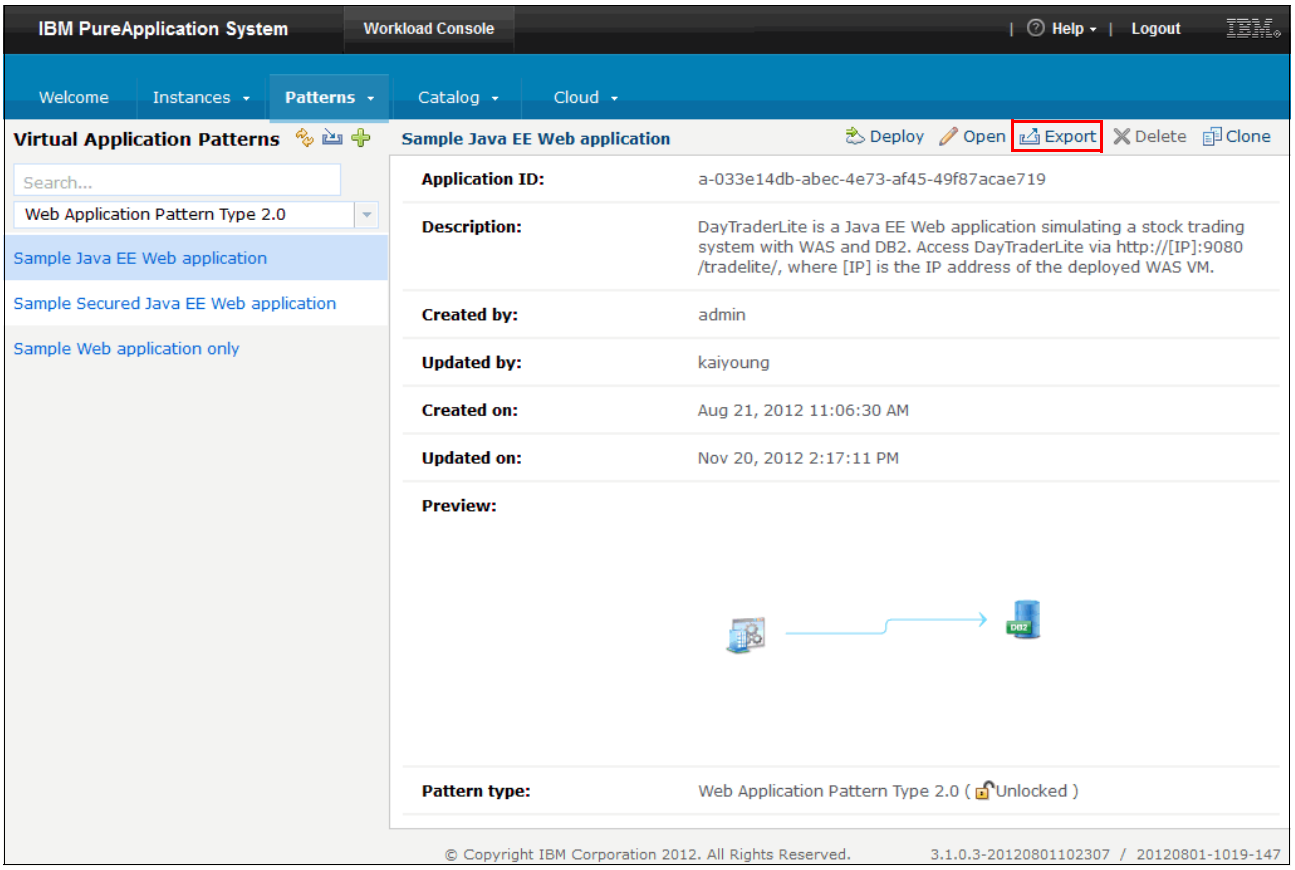


Figure 9-7 Exporting a Virtual Application Pattern by using the GUI console

5. Save the application `.zip` file to the local machine.
6. Extract the `.zip` file to get the application artifacts, such as EAR file and SQL schema.

Restore by using the GUI console

Complete the following steps to restore a virtual application pattern by using the GUI console:

1. Log in to IBM PureApplication System GUI Console.
2. Select **Workload Console**.
3. Select **Patterns** → **Virtual Applications**.
4. Click the **Import** icon from the left toolbar, as shown in Figure 9-8.

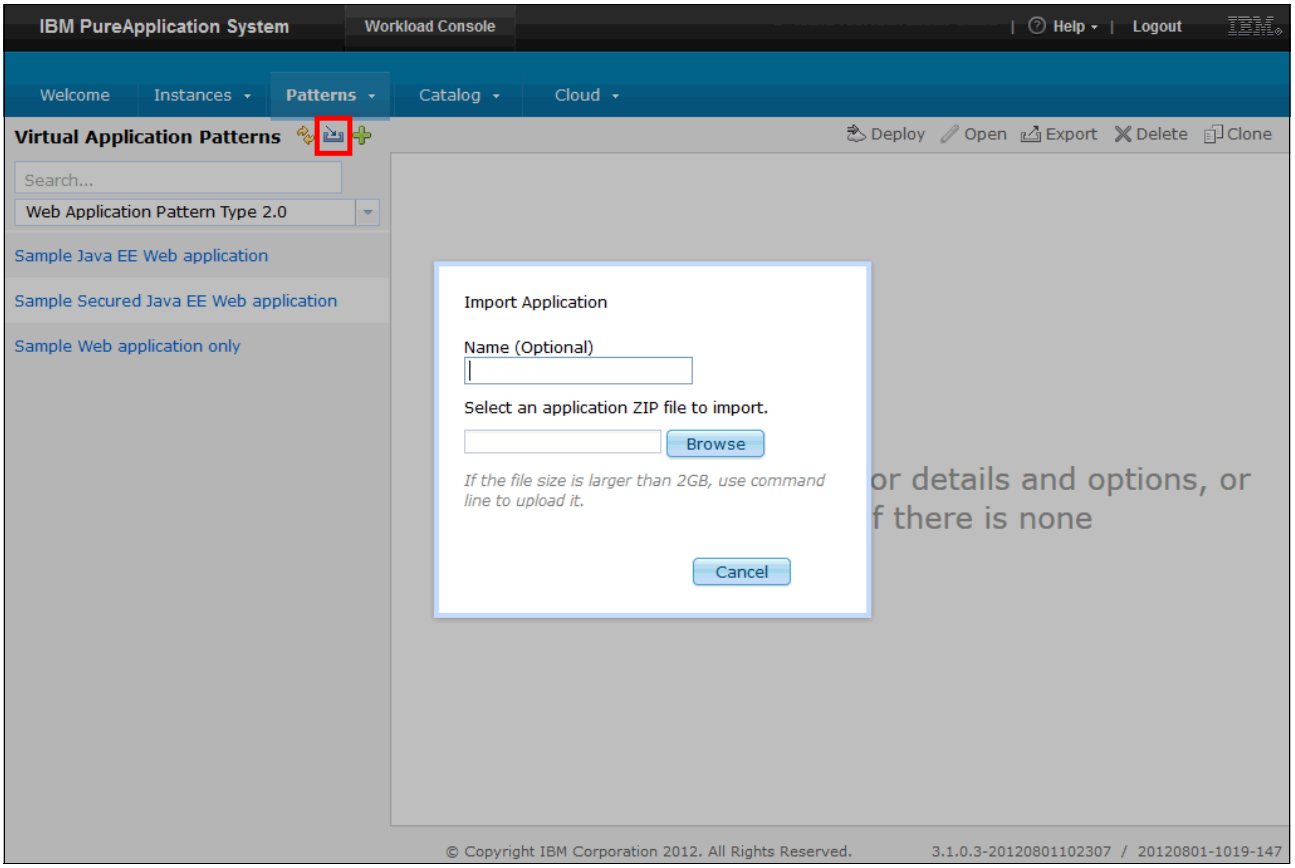


Figure 9-8 Importing a virtual application pattern by using the GUI console

5. The **Import Application** prompt window opens.
6. Enter the application name (optional).
7. Select the application `.zip` file to import. If the file size is larger than 2 GB, use the command-line tool to upload the file.

The `.zip` file is uploaded, validated, and if there is nothing wrong with the package, it is made available for use on the Virtual Application Patterns page.

Back up by using the CLI

Run the **pure** command (as shown in Example 9-1) to export a virtual application pattern. This process saves the application .zip file on the specified path. The .zip contains the application artifacts, such as the EAR file and database schemas.

Example 9-1 Command to export a virtual application pattern

Command:

```
C:\pure.cli\bin>pure -a -h <host> -u <user> -p <password>
Welcome to the IBM PureApplication System CLI. Enter 'help' if you need help
getting started.

>>> deployer.applications['<app_name>'].download('<local_file_path>')
```

Output: None

The following parameters are encountered:

- <host> is the IP address or host name of PureApplication System
- <user> is the user name for logging in to PureApplication System
- <password> is the user's password
- <app_name> is the name of the virtual application pattern
- <local_file_path> is the local path to the .zip file that is downloaded

The name of all available applications can be found by running the following command on CLI:

deployer.applications

Restore by using the CLI

Run the **pure** command (as shown in Example 9-2) to import a virtual application pattern. This process creates an application pattern from the uploaded `.zip` file. The `.zip file` should contain all the application artifacts, such as the EAR file and database schemas.

Example 9-2 Command to import a virtual application pattern

Command:

```
C:\pure.cli\bin>pure -a -h <host> -u <user> -p <password>
Welcome to the IBM PureApplication System CLI. Enter 'help' if you need help
getting started.

>>> deployer.applications.create('<local_file_path>')
```

Output:

```
{
  "access_rights": (nested object),
  "acl": (nested object),
  "app_id": "a-5a305cb1-03f2-4b31-bc69-0eaa096f8150",
  "app_name": "Sample Java EE Web application Test Import",
  "app_type": "application",
  "artifacts": (nested object),
  "content_md5": "57BD4C251935AAD000047AE6A343E0EEC",
  "content_type": "application/json",
  "create_time": "2012-12-21T01:03:16Z",
  "creator": "user51",
  "last_modified": "2012-12-21T01:03:18Z",
  "last_modifier": "user51",
  "patterntype": "webapp",
  "version": "2.0"
}
```

The following parameters are encountered:

- `<host>` is the IP address or host name of PureApplication System
- `<user>` is the user name for logging in to PureApplication System
- `<password>` is the user's password
- `<local_file_path>` is the local path to the `.zip` file that is uploaded

9.5.2 Virtual system pattern back up and restore

Virtual system patterns can be exported and imported only by using the CLI, as shown in the following sections.

Back up by using the CLI

A virtual system pattern can have virtual images, script packages, and add-ons. The `exportPattern.py` script, which is provided in the `samples` directory of CLI, can be used to download all the components from a virtual system pattern, as shown in Example 9-3.

Example 9-3 Command to export a virtual system pattern through CLI

```
Command:

pure -a -h <host> -u <user> -p <password> -f ..\samples\exportPattern.py --pattern
<vsp_name> --target <local_dir_path> --downloadAll

Example:

C:\pure.cli\bin>pure -a -h 9.38.12.55 -u admin -p admin -f
..\samples\exportPattern.py --pattern "vSysTest" --target ..\backup --downloadAll

Output:

Created a temporary directory
"c:\users\admin\ibm_ad~1\adddata\local\temp\exportin7pev" for
Exporting pattern "vSysTest"...
Exporting script package "Install TradeDB App" to
"script_packages/InstallTradeDBApp.zip...
Exporting script package "Install DB Drivers" to
"script_packages/InstallDBDrivers.zip...
Exporting add-on "Default add user" to "add_ons/defaultadduser.zip...
Exporting virtual image "WebSphere Application Server 8.5.0.0" to
"c:\users\admin\ibm_ad~1\adddata\local\temp\exportin7pev\images"...
Export virtual image "WebSphere Application Server 8.5.0.0" to "images/WebSphere
Application Server 8.5.0.0.ova" successfully
Exporting virtual image "DB2 Enterprise 9.7.5.0" to
"c:\users\admin\ibm_ad~1\adddata\local\temp\exportin7pev\images"...
Export virtual image "DB2 Enterprise 9.7.5.0" to "images/ DB2 Enterprise
9.7.5.0.ova" successfully
Export pattern "vSysTest" successfully.
```

The following parameters are encountered:

- `<host>` is the IP address or host name of PureApplication System
- `<user>` is the user name for logging in to PureApplication System
- `<password>` is the user's password
- `<vsp_name>` is the name of the virtual system pattern
- `<local_dir_path>` is the path to the local directory where the files are downloaded

You also can export specific components by creating filters that use the JSON notation. For example, the filter `exportAddOn.json`, as shown in Example 9-4, export add-ons only from the virtual system pattern.

Example 9-4 Sample JSON filter for exporting add-ons from a virtual system pattern

```
{
   "add_ons": [
      {"name": "Default add user"},
   ]
}
```

The virtual system pattern add-ons can be downloaded by using a JSON filter, as shown in Example 9-5.

Example 9-5 Using a JSON filter for exporting add-ons from a virtual system pattern

Example:

```
C:\pure.cli\bin>pure -a -h 9.38.12.55 -u admin -p admin -f
..\samples\exportPattern.py --pattern "vSysTest" --target vSysTest.tgz --download
exportAddOn.json
```

Output:

```
Created a temporary directory
"c:\users\admin\ibm_ad~1\adddata\local\temp\exportin7pev" for
Exporting pattern "vSysTest"...
Exporting add-on "Default add user" to "add_ons/defaultadduser.zip...
Export pattern "vSysTest" successfully.
```

Restore by using the CLI

The `importPatterns.py` script, which is provided in the `samples` directory of CLI, can be used to restore all components from a virtual system pattern, as shown in Example 9-6.

Example 9-6 Command to import a virtual system pattern through CLI

Command:

```
pure -a -h <host> -u <user> -p <password> -f ..\samples\importPatterns.py --source
<src_dir_path>
```

Example:

```
C:\pure.cli\bin>pure -a -h 172.24.70.10 -u admin -p password -f
..\samples\importPatterns.py --source ..\backup
```

Output:

```
Importing pattern "VSPTestRestore01" ...
Import pattern "VSPTestRestore01" successfully.
```

The following parameters are encountered:

- <host> is the IP address or host name of PureApplication System
- <user> is the user name for logging in to PureApplication System
- <password> is the user's password
- <src_dir_path> is the source directory from where the files are uploaded

9.5.3 Virtual appliance pattern backup and restore

Virtual appliance patterns can be exported and imported from the GUI console or the CLI. When you export a virtual appliance, an Open Virtualization Archive (OVA) file is saved to the local machine. This file can be imported later into another IBM PureApplication System.

Back up by using the GUI console

Complete the following steps to back up a virtual appliance pattern by using the GUI console:

1. Log in to IBM PureApplication System GUI Console.
2. Select **Workload Console**.
3. Select the virtual application pattern by clicking **Cloud** → **Virtual Appliance**.
4. Click the **Export** icon from the toolbar.
5. Save the OVA file to the local machine.

Backup by using the CLI

Run the **pure** command (as shown in Example 9-7) to export a virtual appliance pattern through CLI.

Example 9-7 Exporting a virtual appliance pattern through CLI

```
Command:

pure -a -h <host> -u <user> -p <password> -c
"admin.virtualappliances['<vapp_name>'][0].export('<local_dir_path>')"

Example:

C:\pure.cli\bin>pure -a -h 9.38.12.55 -u admin -p admin -c
"admin.virtualappliances['testAppliance'][0].export('C:\backup')"

Output: None
```

The following parameters are encountered:

- <host> is the IP address or host name of PureApplication System
- <user> is the user name for logging in to PureApplication System
- <password> is the user's password
- <vapp_name> is the name of the virtual appliance pattern
- <local_dir_path> is the path to the local directory where the files are downloaded

The name of all available appliances can be found by running the following command on CLI:

```
admin.virtualappliances
```

Restore by using the CLI

Run the **pure** command (as shown in Example 9-8) to import a virtual appliance pattern through CLI.

Example 9-8 Importing a virtual appliance pattern through CLI

Command:

```
pure -a -h <host> -u <user> -p <password> -c
"admin.virtualappliances.create({'name':'<vapp_name>', 'importurl':
'<import_url>'})"
```

Example:

```
C:\pure.cli\bin>pure -a -h 9.38.12.55 -u admin -p admin -c
"admin.virtualappliances.create({'name':'myName', 'importurl':
'file://C:/data/storage/templates/uploaded-template-134556812.ova'})"
```

Output:

```
{
  "acl": (nested object),
  "cloud": (nested object),
  "created_time": "Jan 22, 2013 9:29:10 PM",
  "description": None,
  "id": "b732dc2f-6dbe-42f6-844f-dbbe22d9b189",
  "importurl": "file:///data/storage/templates/uploaded-template-134556812.ova",
  "name": "myName",
  "state": "pending",
  "updated_time": "Jan 22, 2013 9:29:10 PM",
  "url": "/deployment/resources/images/b732dc2f-6dbe-42f6-844f-dbbe22d9b189",
  "vmconfigurations": (nested object)
}
```

The following t parameters:

- <host> is the IP address or host name of PureApplication System
- <user> is the user name for logging in to PureApplication System
- <password> is the user's password
- <vapp_name> is the name of the virtual appliance pattern
- <import_url> is the URL to the local file that is uploaded

This method returns a virtual appliance object for the new virtual appliance. Because of their size, importing virtual appliances can take several minutes. This method queues the operation and returns immediately. The returned object can be used to track the status of the import process.

9.5.4 Configurations backup

From IBM PureApplication System, it is possible to export other configurations such as DNS, cloud group, IP group, and user accounts. To export these configurations, run the **saveAll.py** command, as shown in Example 9-9.

Example 9-9 Command for exporting PureApplication System configurations

```
C:\pure.cli\bin>pure -h 9.38.12.55 -u admin -p admin -a -f ..\samples\saveAll.py
Generating report in 9.38.12.55 .txt...
Generating script in 9.38.12.55 .py...
Reading DNS data...
Reading NTP data...
Reading user data...
Reading user group data...
Reading IP group data...
Reading cloud group data...
Reading ipgroup data for Shared...
Reading compute nodes data for Shared...
Saved report to 172_18_24_12.txt.
Saved script to 172_18_24_12.py.
```

After the command is run, you have two files. One with the extension `.txt` and one with the extension `.py`. The `.txt` file contains information about appliance, DNS, NTP, users, user groups, IP groups, and cloud groups (IP groups and compute nodes). The `.py` file can be used to re-create the data that is found on the PureApplication System. The script creates users, IP groups, and cloud groups.

9.6 Current limitations

In this release of PureApplication System, there is no automatic process to set up another rack for disaster recovery. However, you can take the following steps to prevent system outages or minimize the service downtime:

- Create another rack that contains critical applications, either ready to run or already running in standby mode. This design allows a quick switch in a disaster.
- Set up an external sprayer to forward the requests. When the first rack is down, the sprayer can forward the requests to the backup system.
- At the core, continue to apply the normal disaster recovery mechanisms and standard procedures in use today.
- Back up critical databases, such as the stand-alone DBaaS or DBaas as part of Web Application patterns, by using backup solutions such as Tivoli Storage Manager. For DB2 VMs (as part of virtual system patterns), you can use the standard backup and restore mechanisms that are supported by DB2.
- Manage copies of image configurations, critical virtual applications, and virtual system patterns by exporting and importing to the secondary rack. This process can be done manually or automatically so that after the import, the scripts deploy the patterns and move them into a stable, configured state before they start taking traffic. You also can create the same users, groups, ACLs, and cloud configurations in the backup rack.

Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this book.

Online resources

The following websites are relevant as further information sources:

- IBM PureApplication Systems W1500 Information Center:

  http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.ipas.doc/welcome/welcome.html

- IBM developerWorks article, Managing application runtime environments in IBM PureApplication System:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1210_woolf/1210_woolf.html

- IBM developerWorks article, Security and trust in IBM PureApplication System:

  http://www.ibm.com/developerworks/websphere/techjournal/1212_chao/1212_chao.html

- IBM developerWorks article, A guided tour of IBM Database Patterns, Part 2: Database image management:

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1206patterns2/index.html

- IBM developerWorks article, Navigate a shared services workload in PureApplication System:

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

- IBM developerWorks article, DB2 SQL compatibility feature:

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0907oracleappsondb2/index.html

- Built-in images supported in IBM PureApplication System:

  http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS213-038&appname=USN

- Preparing for IBM PureApplication System, Part 1: Onboarding applications overview:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1204_abrams/1204_abrams.html

- Preparing for IBM PureApplication System, Part 2: Is your application ready to become virtual?:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1204_brown/1204_brown.html

- Preparing for IBM PureApplication System, Part 3: Choosing a database option:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1204_chan/1204_chan.html

© Copyright IBM Corp. 2013. All rights reserved.

- Design a virtual system pattern:

  http://www.ibm.com/developerworks/cloud/library/cl-puresystem-vspdesign/
- Manage application services with virtual application patterns:

  http://www.ibm.com/developerworks/cloud/library/cl-puresystem-vap/
- Navigate a shared services workload in PureApplication System:

  http://www.ibm.com/developerworks/cloud/library/cl-pureappsharesrvcs/
- Deploying into the cloud with the IBM Application Pattern for Java:

  http://www.ibm.com/developerworks/library/j-apppattern/
- Manage the topology with virtual system patterns:

  http://www.ibm.com/developerworks/cloud/library/cl-puresystem-vsp/index.html
- IBM PureApplication System Version 1.0 Information Center:

  http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/index.jsp?topic=%2Fcom.ibm.ipas.doc%2Fwelcome%2Fwelcome.html
- Education Assistance: IBM PureApplication System Deployment Models:

  http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp?topic=/com.ibm.iea.ipas/ipas/1.0/Patterns/IPASv1_VApp_DBaaSPatterns/player.html
- A guided tour of IBM Database Patterns, Part 2: Database image management:

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1206patterns2/index.html
- Preparing for IBM PureApplication System, Part 1: Onboarding applications overview:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1204_abrams/1204_abrams.html
- Preparing for IBM PureApplication System, Part 2: Is your application ready to become virtual?:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1204_brown/1204_brown.html
- Preparing for IBM PureApplication System, Part 3: Choosing a database option:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1204_chan/1204_chan.html
- Design a virtual system pattern:

  http://www.ibm.com/developerworks/cloud/library/cl-puresystem-vspdesign/
- Manage application services with virtual application patterns:

  http://www.ibm.com/developerworks/cloud/library/cl-puresystem-vap/
- Education Assistance: Virtual Application Shared Services:

  https://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.iwd/iwd/3.1/Patterns/WD31_VirtualApplicationSharedServices/player.html
- Education Assistance: Database-as-a-Service Patterns:

  http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp?topic=/com.ibm.iea.ipas/ipas/1.0/Patterns/IPASv1_VApp_DBaaSPatterns/player.html
- Education Assistance: Database-as-a-Service custom workloads:

  http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/topic/com.ibm.iea.ipas/ipas/1.0/Patterns/IPASv1_VApp_DBaaS_CustomWorkloads/player.html

- Navigate a shared services workload in PureApplication System:

  http://www.ibm.com/developerworks/cloud/library/cl-pureappsharesrvcs/
- Deploying into the cloud with the IBM Application Pattern for Java:

  http://www.ibm.com/developerworks/library/j-apppattern/
- Manage the topology with virtual system patterns:

  http://www.ibm.com/developerworks/cloud/library/cl-puresystem-vsp/index.html
- Education Assistance: IBM PureApplication System Deployment Models:

  http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp?topic=/com.ibm.iea.ipas/ipas/1.0/Patterns/IPASv1_VApp_DBaaSPatterns/player.html
- A guided tour of IBM Database Patterns, Part 2: Database image management:

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1206patterns2/index.html
- IBM PureApplication System cloud and isolation concepts, features and application management, refer to the following IBM Developerworks article:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1210_woolf/1210_woolf.html
- IBM PureApplication System security and trust refer to the following IBM Developerworks article:

  http://www.ibm.com/developerworks/websphere/techjournal/1212_chao/1212_chao.html

Help from IBM

IBM Support and downloads:

http://www.ibm.com/support

IBM Global Services:

http://www.ibm.com/services

Adopting IBM PureApplication System V1.0

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Adopting IBM PureApplication System V1.0

Learn the simplicity of PureApplication System bulit-in patterns

Discover IBM PureApplication System agility

Discover how to isolate applications

This IBM Redbooks publication introduces users to the concepts of the IBM PureApplication System V1.0. This book covers the most common problems, solutions, best practices, and use cases about adopting the IBM PureApplication System V1.0.

The target audience for this book is anyone from the IT industry who wants to acquire a better understanding of IBM PureApplication System, including technical consultants, business partners, and independent software vendors who are considering migrating to a cloud computing solution. This book also is application to system administrators, middleware specialists, and software engineers who need a more in-depth approach to PureApplication System features and capabilities.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-8113-00 ISBN 0738438014