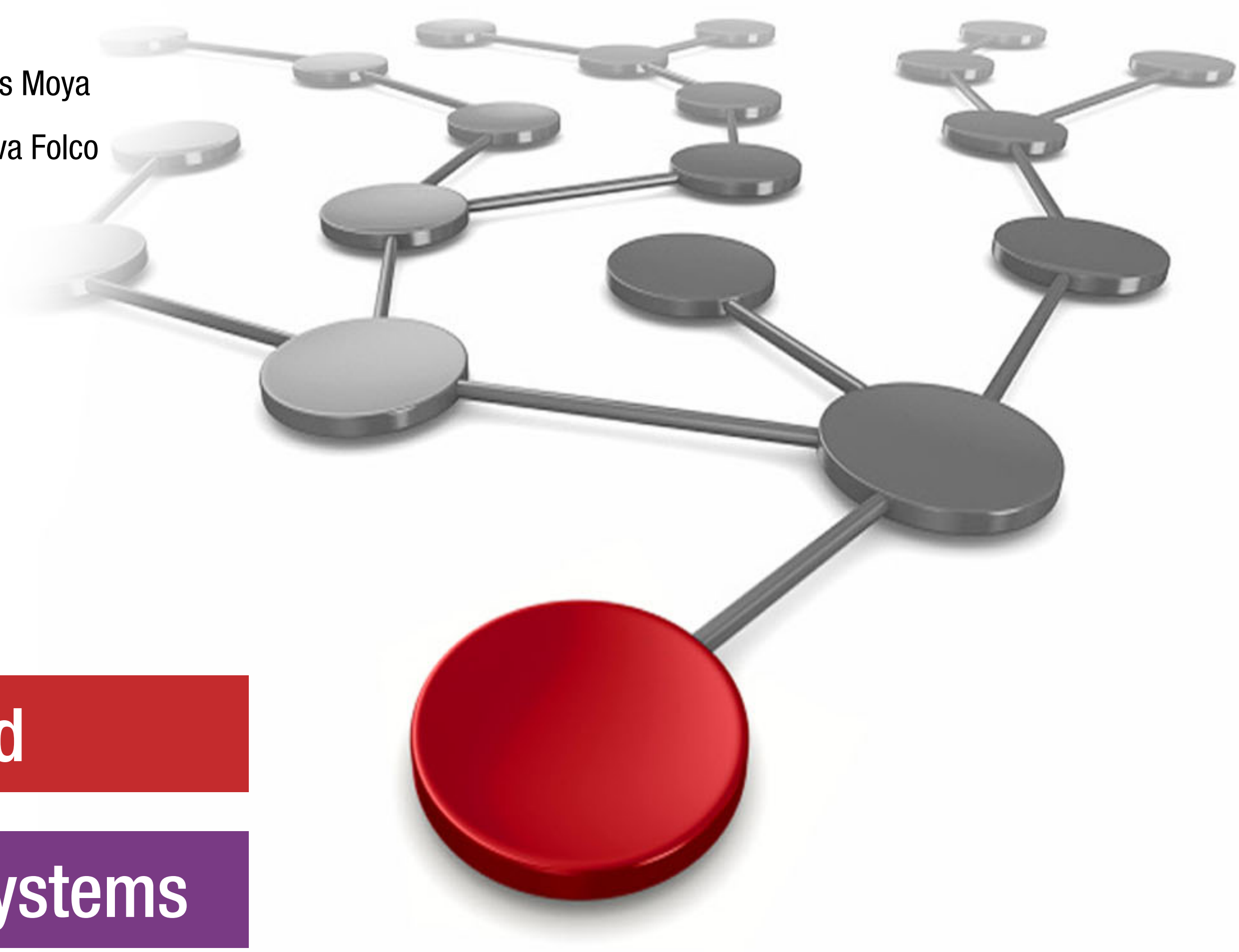
IBM Open Platform for DBaaS on IBM Power Systems

Dino Quintero

Fabio Martins

Eduardo Luis Cerdas Moya

Rafael Camarda Silva Folco



Cloud

Power Systems

Redbooks

ibm.com/redbooks

IBM

Redbooks

International Technical Support Organization

IBM Open Platform for DBaaS on IBM Power Systems

March 2018

SG24-8408-00

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

First Edition (March 2018)

This edition applies to the following versions:
GCC 4.53
Python 2.7 and 3
Ruby 1.2.3
Java7 2.6.8-1
LXC 2.0.7
Apache2 2.4.18
Pciutils 1.3.3.1
Sendmail 8.15.2-3
Sysstat 11.2.0
MySQL 5.7.19
Galera 25.3.19-Xenial
HAProxy 1.6.3-1
Keepalived 1.1.2.19-1ubuntu0.1
Memcached 1.4.25-2ubuntu1.2
Ceph 10.2.5-0ubuntu0.16.04.1
OpenStack: All Newton v14.1.1
Elasticsearch 2.4.1
LogStash 2.4.0-1
Kibana 4.6.4
Nagios 4.2.4
NRP and NagiosPlugins 2.15-1ubuntu1

Contents

Notices vii
Trademarks viii

Preface ix
Authors x
Now you can become a published author, too! xi
Comments welcome xi
Stay connected to IBM Redbooks xi

Chapter 1. Overview of the Open Platform for Database as a Service on IBM Power Systems solution 1
1.1 Introduction to the Open Platform for DBaaS on Power Systems solution 2
1.1.1 What is the Open Platform for DBaaS on Power Systems solution 2
1.1.2 Why use the Open Platform for DBaaS on Power Systems solution 3
1.1.3 Benefits of using the Open Platform for DBaaS on Power Systems solution 7
1.2 Open Platform for DBaaS on Power Systems components 9
1.2.1 The Open Platform for DBaaS on Power Systems hardware infrastructure 10
1.2.2 DBaaS elastic cloud infrastructure 13
1.3 Open Platform for DBaaS on Power Systems product delivery and support flow 20

Chapter 2. IBM DevOps concepts 25
2.1 IBM DevOps 26
2.1.1 DevOps lifecycle phases 27
2.1.2 DevOps practices 28
2.1.3 Cloud Infrastructure as a Service and DevOps 29
2.1.4 Infrastructure as code 31
2.2 Infrastructure as a Service+ 33
2.2.1 Open Platform for Database as a Service on Power Systems 34
2.2.2 Trove 36
2.3 Supported databases 39

Chapter 3. Architecture 43
3.1 Planning for the Open Platform for DBaaS on Power Systems solution 44
3.1.1 Physical integration with the customer's infrastructure 44
3.1.2 Security integration (LDAP and Keystone) 58
3.1.3 Custom database images 59
3.2 Hardware and software requirements 59
3.3 Infrastructure sizing 61
3.3.1 Starter 61
3.3.2 Entry (small) 63
3.3.3 Cloud scale (medium) 65
3.3.4 Performance (large) 67
3.4 Networking 68
3.4.1 General requirements 69
3.5 Solution components and roles 70
3.5.1 Compute nodes 71
3.5.2 Controller nodes 74
3.5.3 Block storage nodes 83
3.5.4 Object storage nodes 91

Chapter 4. Usage 97
4.1 Get and build images 98
4.2 Deploying and maintaining instances 98
4.2.1 Launching an instance 99
4.2.2 Checking the instance information 103
4.2.3 Maintaining an instance 105
4.2.4 Restarting an instance 106
4.2.5 Resizing an instance 107
4.2.6 Deleting an instance 108
4.2.7 Resizing a volume 109
4.2.8 Renaming an instance 110
4.2.9 Creating a database 111
4.2.10 Deleting a database 112
4.3 Backup and recovery 113
4.3.1 Creating a backup 114
4.3.2 Restoring from backup 115
4.3.3 Deleting a backup 116
4.3.4 Backup containers 117
4.4 Security 117
4.4.1 Creating a user 118
4.4.2 Deleting a user 119
4.4.3 Managing root access 120
4.4.4 Managing user access 121

Chapter 5. Monitoring and troubleshooting 123
5.1 Introduction to cluster monitoring and troubleshooting 124
5.2 Accessing the operations management tools 127
5.3 Nagios 129
5.3.1 Nagios Core basic monitoring concepts 131
5.3.2 Nagios Core deployment in Open Platform for DBaaS on Power Systems 131
5.3.3 Nagios Core configuration files 132
5.3.4 Nagios Core usage examples 137
5.4 Elastic stack (Kibana) 149
5.4.1 Using the Kibana Dashboard 153
5.4.2 Selecting other Dashboards that are available in the Open Platform for DBaaS on Power Systems solution 158
5.4.3 Performing searches with Kibana 159
5.4.4 Viewing saved searches 164
5.4.5 Using Kibana to create a graph that is based on a search 164
5.4.6 Viewing saved visualizations 168
5.4.7 Adding the graph visualization to the Dashboard 168
5.4.8 Using Kibana to view metrics 169
5.4.9 Using Kibana for troubleshooting 178

Chapter 6. Scaling 189
6.1 Scaling up your cluster 190
6.1.1 Adding a compute node 190
6.1.2 Adding a storage node 197
6.2 Horizontal scaling 198

Appendix A. Servers provisioning and deployment 201
Baremetal provisioning 202
OpenStack deployment 206
Alternative deployment 208

Image building 212
The dbimage-builder charm 212
The dibimage-builder scripts 214

Related publications 215
IBM Redbooks 215
Online resources 215
Help from IBM 215

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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, MD-NC119, Armonk, NY 10504-1785, US

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 jurisdictions 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 provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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 actual people or business enterprises 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

Preface

This IBM® Redbooks® publication describes how to implement an Open Platform for Database as a Service (DBaaS) on an IBM Power Systems™ environment for Linux, and demonstrates the open source tools, optimization, and preferred practices for the implementation. The Open Platform for DBaaS on Power Systems solution is an on-demand, secure, and scalable self-service database platform that automates provisioning and administration of databases to support new business applications and information insights.

This publication addresses topics to help sellers, architects, brand specialists, distributors, resellers, and anyone offering a secure and scalable Open Platform for DBaaS on Power Systems solution with APIs that are consistent across heterogeneous open database types. An Open Platform for DBaaS on Power Systems solution can accelerate business success by providing an infrastructure, and tools that use open source and OpenStack software that is engineered to optimize hardware and software between workloads and resources so you have a responsive, and an adaptive environment. Moreover, this publication provides documentation to transfer the how-to-skills for cloud-oriented operational management of an Open Platform for DBaaS on Power Systems service and its underlying infrastructure to the technical teams.

The Open Platform for DBaaS on Power Systems mission is to provide scalable and reliable cloud database as a service provisioning function for both relational and non-relational database engines, and to continue to improve its fully featured and extensible open source framework. For example, Trove is a database as a service for OpenStack. It is designed to run entirely on OpenStack, with the goal of enabling users to quickly and easily use the features of a relational or non-relational database without the burden of handling complex administrative tasks. Cloud users and database administrators can provision and manage multiple database instances as needed. Initially, the service focuses on providing resource isolation at high performance while automating complex administrative tasks, including deployment, configuration, patching, backups, restores, and monitoring.

In the context of this publication, the monitoring tool that is implemented is Nagios Core, which is an open source monitoring tool. Hence, when you see a reference to Nagios in this book, Nagios Core is the open source monitoring solution that is implemented. The implementation of Open Platform for DBaaS on IBM Power Systems is based on open source solutions.

This book is targeted toward sellers, architects, brand specialists, distributors, resellers, and anyone developing and implementing Open Platform for DBaaS on Power Systems solutions.

Authors

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

Dino Quintero is a Solutions Enablement Project Leader and an IBM Level 3 Certified Senior IT Specialist with the IBM Redbooks organization in Poughkeepsie, New York. Dino shares his technical computing passion and expertise by leading teams developing content in the areas of enterprise continuous availability, enterprise systems management, high-performance computing, cloud computing, and analytics solutions. He also is an Open Group Distinguished IT Specialist. Dino holds a Master of Computing Information Systems degree and a Bachelor of Science degree in Computer Science from Marist College.

Fabio Martins is a Senior Software Support Specialist with IBM Technical Support Services in Brazil. He has worked at IBM for 13+ years. His areas of expertise include IBM AIX®, IBM PowerVM®, IBM PowerKVM, IBM PowerHA®, IBM PowerVC, IBM PureSystems®, IBM DS8000®, IBM Storwize®, Linux, and Brocade SAN switches and directors. He is a Certified Product Services Professional in Software Support Specialization and a Certified Advanced Technical Expert on IBM Power Systems. He has worked extensively on IBM Power Systems for Brazilian customers, providing technical leadership and support, including how-to questions, problem determination, root cause analysis, performance concerns, and other general complex issues. He holds a bachelor degree in Computer Science from Universidade Paulista (UNIP).

Eduardo Luis Cerdas Moya is a Cloud Infrastructure Specialist who works for IBM Cloud Managed Services® in IBM Costa Rica, which he joined in 2015. He is CompTIA Cloud Essentials and CompTIA Cloud + certified. His areas of expertise include Power Systems, AIX, Linux, PowerHA, PowerVM, software-defined infrastructures, OpenStack, and containerized environments with Docker, Mesos, and Kubernetes.

Rafael Camarda Silva Folco is a Software Engineer at IBM Brazil. He holds a postgraduate degree in software engineering. He has 15+ years of experience in software development, and has worked for IBM for 11 years. His areas of expertise include Linux, virtualization, cloud computing, continuous integration, and performance. He has written extensively on IBM PowerVM and Kernel-based Virtual Machine (KVM) on Power.

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

Wade Wallace
International Technical Support Organization, Austin Center

Jeffrey Benson, Luke Browning, Gary Elliott, Benjamin Kreuz, Manoj Kumar, Samuel Matzek, Marcelo Perazolo, Andrew Solomon, Eric Trevino
IBM US

Ricardo Barros and Antonio Moreira de Oliveira Neto
IBM Brazil

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:

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:

  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

Overview of the Open Platform for Database as a Service on IBM Power Systems solution

This chapter introduces the Open Platform for Database as a Service (DBaaS) on IBM Power Systems solution and provides an overview of its objectives, components, and functions. This chapter also explains the goals that this platform aims to achieve, which includes delivering a quick deployment of the most commonly used open source databases, which enables you to speed up the development of applications that depend on a database, and introducing DevOps concepts to your environment. The Open Platform for DBaaS on Power Systems solution provides a deployment mechanism and tools for all the lifecycle maintenance of the environment.

This chapter contains the following sections:

- Introduction to the Open Platform for DBaaS on Power Systems solution
- Open Platform for DBaaS on Power Systems components
- Open Platform for DBaaS on Power Systems product delivery and support flow

1.1 Introduction to the Open Platform for DBaaS on Power Systems solution

This section introduces the Open Platform for DBaaS on Power Systems solution, and explains the reasons to consider using it in your environment. This section shows the components that are involved in this solution, how they interact, and the final objective that they achieve together. This section also shows the benefits of using this solution, including total cost of ownership (TCO) comparisons.

1.1.1 What is the Open Platform for DBaaS on Power Systems solution

The Open Platform for DBaaS on Power Systems solution is a solution that integrates several components, including software and hardware, and implements a complete environment that is easy to use and fast for deploying open source databases such as MariaDB, MongoDB, MySQL, PostgreSQL, and Redis. This solution provides all the necessary components to create quickly a database instance within minutes, providing you with an interface to connect to such a database and start developing your application.

The Open Platform for DBaaS on Power Systems solution integrates OpenPower system servers, software-defined storage (SDS) (by using Ceph), operating systems (OS) (Ubuntu is used in the physical servers that are part of the solution and on the virtual machines (VMs) that are deployed to run the open source databases), databases, physical network switches, and OpenStack as a cloud OS with all the necessary components to assemble Infrastructure as a Service (IaaS) and deliver an agile deployment methodology. The OpenStack Trove project is used in this solution for implementing DBaaS, and the OpenStack Swift project is used for implementing object storage for backup purposes.

This hardware and software solution enables rapid deployment of virtual databases on a private cloud. Figure 1-1 shows an overview of the components that are involved in this solution.

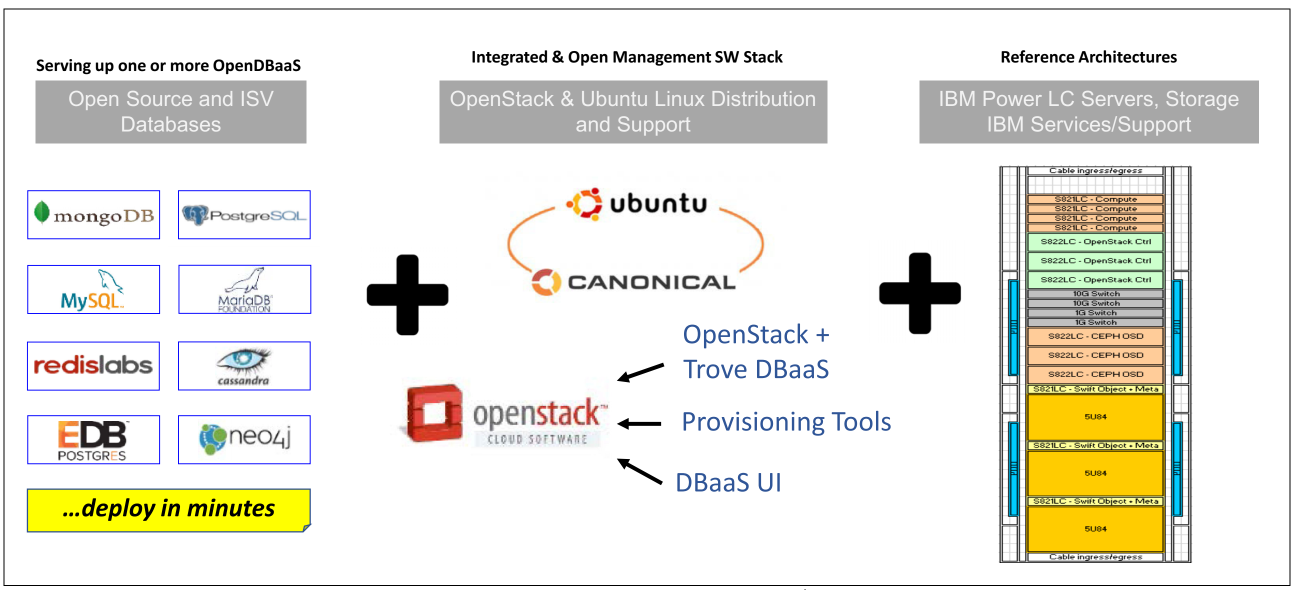


Figure 1-1 The Open Platform for DBaaS on Power Systems components[1]

[1] The source for this figure is *Open Platform for Database as a Service on IBM Power Systems Solution Brief*.

1.1.2 Why use the Open Platform for DBaaS on Power Systems solution

There are many sources of information that are used by businesses, including internal input (information that is entered by the employees), external input (information that is obtained from other companies, sometimes contracted to provide insight and market research, for example), social networks, and mobile input and sensors that provide data when certain events happen, which causes a constant increase in the amount of information that a business receives and needs to store, analyze, and evaluate.

Some of this information is structured, and other information is unstructured data, and each type of data requires a specific database engine to better store the data and provide a method of organizing and using such information. Conventional database engines that are used widely in traditional enterprise computing and data centers are optimized for structured data, and cannot store and use unstructured data in an optimal fashion.

Open source software, including open source database engines, enable companies to benefit from the expertise of the whole community behind the development of such software. There are many database engines that are developed by the community that are highly optimized for certain types of data. Sometimes (and more often), a single database engine is not enough to store the different sources and types of data, requiring various engines, and optimized for each of the data sets that it stores.

The Open Platform for DBaaS on Power Systems solution provides a solution for quick deployment and lifecycle maintenance of different database engines. The solution uses a single interface, so besides quick deployment that can be performed by DevOps (the developers can quickly perform IT operations actions, without waiting for another department to provide a database that must be used during the software development), it also provides a single interface to interact with many different engines so that the operator does not need to learn the particularities of the different engines. For example, the operators do not need to know how to install, create an instance, and then create a database with MySQL, MongoDB, and MariaDB because the operator uses only a single interface to perform the same actions on all three different engines, which certainly have several particularities that must be addressed to accomplish the same goal.

Besides speeding up deployment and maintenance, and making it easy to use different database engines (enabling you to select the most appropriate and optimized database for each type of data and application), the Open Platform for DBaaS on Power Systems solution also reduces the costs of licensing and infrastructure by using open source software and hardware solutions.

Figure 1-2 shows several sources of data and how an environment with multiple database engines, specially open source databases, can benefit and take advantage of a mix of structured and unstructured data.

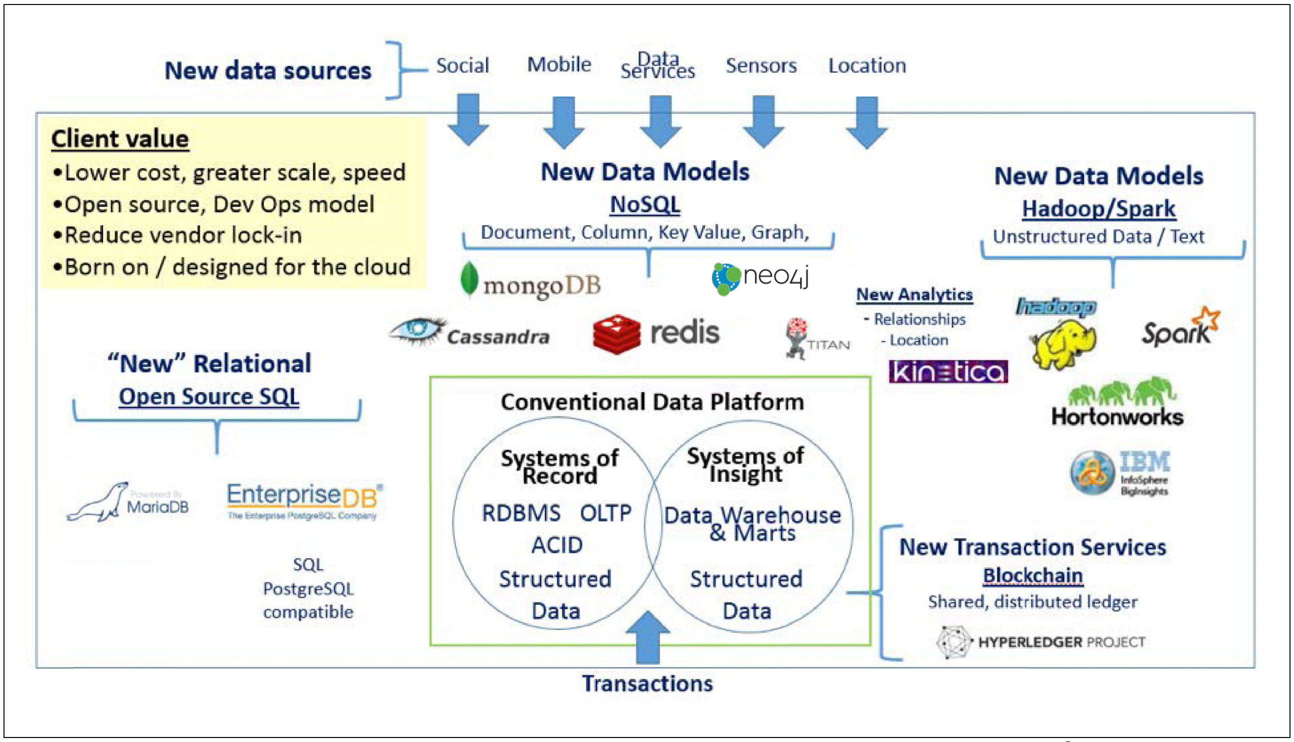


Figure 1-2 Different types and sources of data benefit from mixed and optimized database engines[2]

[2] Some of the database engines that are shown in this figure are not based on open source software.

The adoption of open source database engines has constantly increased in the past years. Businesses are seeing the benefits from optimized and open source database engines and using them in their development and production environment. Figure 1-3 shows a graph demonstrating how the adoption of open source database engines, such as MariaDB, Cassandra, MongoDB, and Redis, have increased.

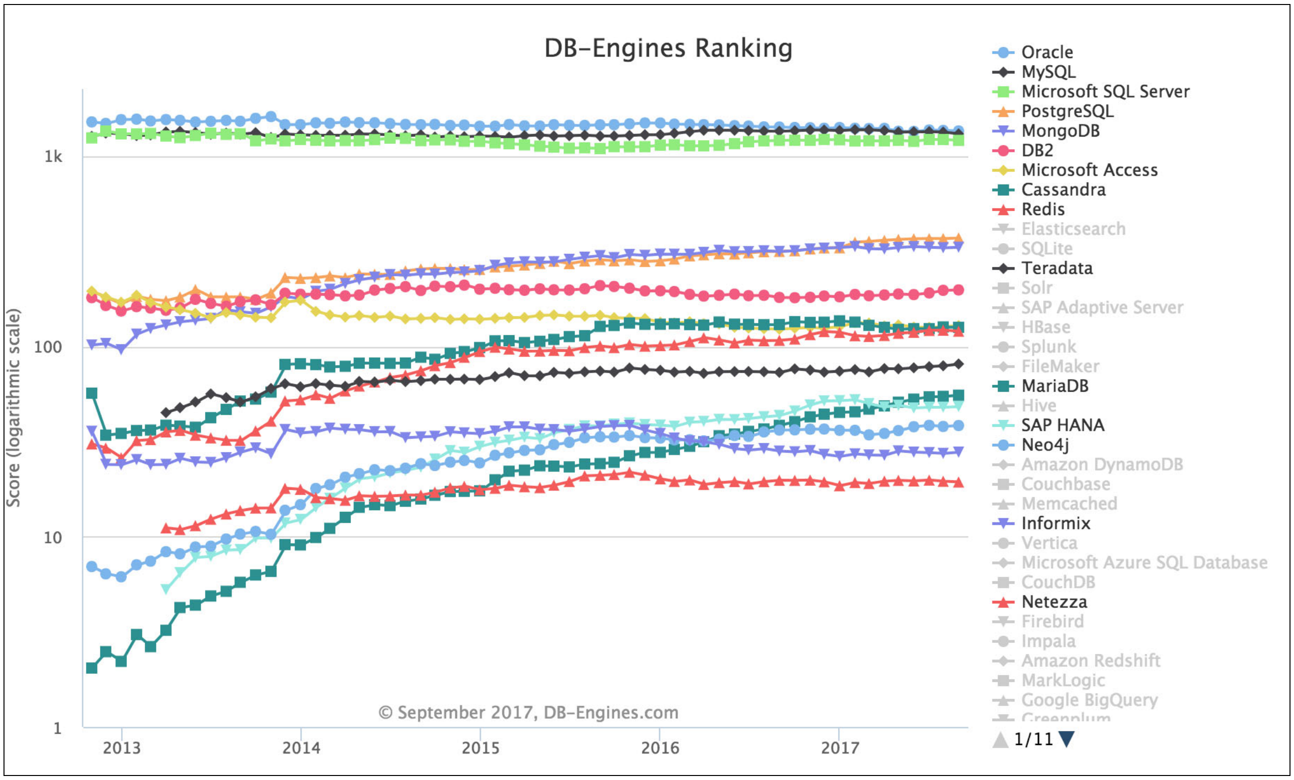


Figure 1-3 Database engines ranking graph

Note: The source of the information in Figure 1-3 is Historical trend of the popularity ranking of database management systems. This information was obtained at the time this publication was written.

Figure 1-4 shows a table comparing the popularity of several database engines. You can see that PostgreSQL, MongoDB, Redis, and MariaDB are increasing in popularity when you compare September 2017 with September 2016, which demonstrates a constant adoption of such database engines.

Rank			DBMS	Database Model	Score		
Sep 2017	Aug 2017	Sep 2016			Sep 2017	Aug 2017	Sep 2016
1.	1.	1.	Oracle	Relational DBMS	1359.09	-8.78	-66.47
2.	2.	2.	MySQL	Relational DBMS	1312.61	-27.69	-41.41
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1212.54	-12.93	+0.99
4.	4.	4.	PostgreSQL	Relational DBMS	372.36	+2.60	+56.01
5.	5.	5.	MongoDB	Document store	332.73	+2.24	+16.74
6.	6.	6.	DB2	Relational DBMS	198.34	+0.87	+17.15
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	128.81	+1.78	+5.50
8.	8.	↓ 7.	Cassandra	Wide column store	126.20	-0.52	-4.29
9.	9.	↑ 10.	Redis	Key-value store	120.41	-1.49	+12.61
10.	10.	↑ 11.	Elasticsearch	Search engine	120.00	+2.35	+23.52
11.	11.	↓ 9.	SQLite	Relational DBMS	112.04	+1.19	+3.41
12.	12.	12.	Teradata	Relational DBMS	80.91	+1.67	+7.84
13.	13.	↑ 14.	Solr	Search engine	69.91	+2.95	+2.95
14.	14.	↓ 13.	SAP Adaptive Server	Relational DBMS	66.75	-0.16	-2.41
15.	15.	15.	HBase	Wide column store	64.34	+0.82	+6.53
16.	16.	↑ 17.	Splunk	Search engine	62.57	+1.11	+11.28
17.	17.	↓ 16.	FileMaker	Relational DBMS	61.00	+1.35	+5.64
18.	18.	↑ 20.	MariaDB	Relational DBMS	55.47	+0.78	+16.94
19.	↑ 20.	↓ 18.	Hive	Relational DBMS	48.62	+1.31	-0.21
20.	↓ 19.	↓ 19.	SAP HANA	Relational DBMS	48.33	+0.36	+4.91
21.	21.	21.	Neo4j	Graph DBMS	38.42	+0.42	+2.06

Figure 1-4 Database engines adoption table

> **Note:** This source of the information in Figure 1-4 was obtained from DB-Engines Ranking - popularity ranking of database management systems. You can also understand how this popularity score is calculated by seeing DB-Engines Ranking - Method.

With this information, a database as a service platform provides key value to your enterprise, as it is an on-demand, secure, and scalable self-service database platform that automates provisioning and administration of databases to support new business applications and information insights.[3] The following elements can be highlighted as the fundamentals of a database as a service platform:

- Provisioning:
  - Quickly provisions database instances
  - Includes an operating environment that backs all the necessary components to run the instance
  - Database automation
  - Provides a broad choice of database engines

[3] *Database-As-A-Service Saves Money, Improves IT Productivity, And Speeds Application Development*

- Management:
  - Operational management (view logs, performance, and usage statistics)
  - Complete lifecycle management of the database instances
  - Backup and restore functions
  - Integrated software updates
  - Security functions
- Integration:
  - Uses and provides a standardized REST API (regardless of database engine), enabling better integration with other applications
  - OpenStack infrastructure integration
  - Provides integrated deployment and maintenance of the whole solution
- Configuration:
  - Provides predefined database images to use for deployment
  - Enables optimization and tuning
  - Uses scale-out configurations, enabling you to scale the solution when needed
  - Enables the management of database-specific parameters

These characteristics enable a database as a service platform to provide speed and agility for database provisioning, thus reducing costs in comparison to traditional database engines and enabling the usage of optimized databases according to the data type and source, enabling you to select the best database engine to cover your application needs. This cost reduction enables you to shift investment to other initiatives, including growth of your lines of business (LOBs) and improving your IT efficiency.

The Open Platform for DBaaS on Power Systems solution delivers all these benefits in addition to the advantage of using OpenPOWER as the hardware backing up this solution. All the leading databases that are available (most of them Open Source Initiatives) are optimized for Linux on Power, including MongoDB, EnterpriseDB (EDB), Neo4j, MariaDB, Spark, Hadoop, Redis, PostgreSQL, Cassandra, MySQL, IBM BigInsights®, and Hortonworks. The Open Platform for DBaaS on Power Systems solution provides all the functions and benefits of an Open Platform for DBaaS on Power Systems solution with the performance, optimization, and cost reduction that is delivered by OpenPOWER servers.

1.1.3 Benefits of using the Open Platform for DBaaS on Power Systems solution

The Open Platform for DBaaS on Power Systems solution provides many benefits to your enterprise:

- The Open Platform for DBaaS on Power Systems solution is *fast*, providing DBs in minutes, not hours or days. You can automate provisioning of the VMs, database instances, and infrastructure, including scaling infrastructure, and also deliver self-service provisioning and lifecycle management tools.
- The Open Platform for DBaaS on Power Systems solution is *flexible*. Developers can choose the best open database for their application by selecting from a menu of popular DB types, including options of SQL and NoSQL databases, with clusters or not. The Open Platform for DBaaS on Power Systems solution also provides tools for clients to add customer DB images to the image repository, expanding the available options of databases that the developer can use.

- The Open Platform for DBaaS on Power Systems solution is e*nterprise grade* because it offers options to improve continuity of the client business, offering clustering and replication for database high-availability, auto-DB backup and restore with disaster recovery options, and strong security capabilities.
- The solution offers *competitive cost, performance*, and *density* by delivering the best performance for client applications, taking advantage of Power Systems servers, and implementing open source SDS to deploy more databases in fewer servers.
- It provides a *foundation for compliance* by helping you keep your IT under control, providing centralized control over key database elements (DB types, security, resource usage, and so on), and providing tools to control and monitor the entire solution.
- The Open Platform for DBaaS on Power Systems solution is also *modern*, designed for the as-a-service model, with self-service capabilities enabling DevOps for DBaaS.
- It is a complete *open* solution that uses OpenPOWER technology, open industry DBaaS services APIs, OpenStack, Kernel-based Virtual Machine (KVM), Linux DBaaS infrastructure, and open database engines.
- It i*mproves the IT productivity* (of developers and operators, or DevOps) with full automation of the database functions, providing better IT resource usage through consolidation and efficient cloud-oriented operational management.
- It is also *integrated*, fitting the existing environment. It is an engineered, optimized, and tested scale-out solution, integrating with existing on-premises infrastructure and with existing cloud bare metal environment, supporting Windows, Linux, and Mac OS database clients to connect to the deployed database instances.

The OpenPOWER hardware that is used in this solution offers excellent cost and proven performance for open databases. Figure 1-5 shows a comparison of running Open Platform for DBaaS on Power Systems servers versus running on x86 either on-premises or in a public cloud.

Competitive differentiation versus x86 - for private (on-premises) or public cloud

	Feature	Open DBaaS on Power Systems	x86 on-premise	Benefits
Versus on-premises	Open hardware infrastructure	✓	✗	CAPI enabled, 300+ OpenPower members
	Reference architectures – engineered, pre-built, turnkey	✓	✗	Dedicated engineering team: experts in Cloud/OpenStack, Open DBs, Ceph block storage, Swift object store, OpenPOWER hardware
	Open source based DBaaS Toolkit	✓	✗	Downloadable via GitHub - built on OpenStack, Open Tools
	Guaranteed performance	✓	✗	1.8x and 2.0x price/performance guarantees
	Superior economics	✓	✗	Up to 6:1 server consolidation versus x86
	Feature	**Open DBaaS on Power Systems**	**x86 in public cloud**	**Benefits**
Versus public cloud	Speed, agility, and scale	✓	✓	Lines of business and developers can respond quickly and scale to meet demands.
	Superior economics	✓	✗	Open DBaaS beats AWS with **35% lower 3 year TCO**.
	Secure and local data solution	✓	✗	Data stays on-premises and fully secure to satisfy regulatory requirements. Also stays in country to meet government laws.
	Control governance and compliance	✓	✗	IT can be responsive and ensure business controls.
	Reduce or eliminate external network latency	✓	✗	Sub-second response times, complete data intensive jobs faster.

Figure 1-5 Competitive differentiation of running the Open Platform for DBaaS on Power Systems solution

You can engage the IBM IT Economics team and request a detailed TCO analysis of your environment at no cost by sending an email to `IT.Economics@us.ibm.com`.

1.2 Open Platform for DBaaS on Power Systems components

The Open Platform for DBaaS on Power Systems solution uses several software and hardware components to deliver its functions. IBM has contributed to many OpenStack projects because they are optimized for usage in OpenPOWER hardware, and all contributions were upstreamed and made publicly available at GitHub - open-power-ref-design/dbaas.

Among the components that are used in this solution are Power Systems hardware, OpenStack cloud software, Swift, which is used as object storage for backup purposes, Ceph, which is used as block storage (SDS), DBaaS services (OpenStack Trove), databases (MariaDB, MongoDB, Cassandra, Redis, and MySQL) and operational management tools (Kibana, ELK, and Nagios).

Figure 1-6 shows the architecture of the Open Platform for DBaaS on Power Systems solution and how each component interacts.

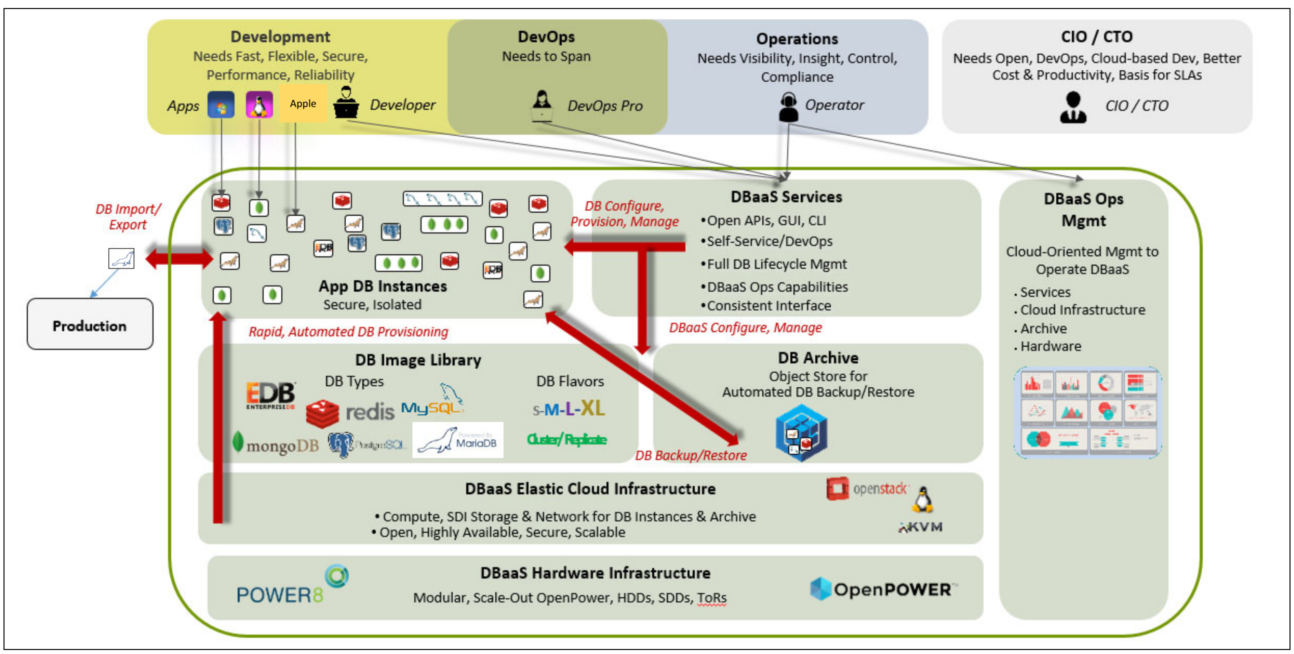


Figure 1-6 The Open Platform for DBaaS on Power Systems solution architecture

There are three main personas that interact directly with the Open Platform for DBaaS on Power Systems solution: development, DevOps, and operations.

- *Developers* need fast, flexible and secure deployment of the database engines to use during development of their application, so they use the DBaaS services (through REST APIs, GUI, or CLI) to deploy the database by using the type and image (selected from the OpenStack Cinder image library).
- The *operations* department needs visibility and control of all the involved components, including the VMs that are deployed in this solution to provide appropriate maintenance. Operations use the DBaaS services and DBaaS operations management (OpsMgr) to deploy database instances, maintain its lifecycle management, coordinate and administer the security policies of the VMs, maintain the projects that have access to the environment (including controlling its quotas), monitor the whole infrastructure and hardware that is involved in the solution, and analyze logs and monitor statistics.
- *DevOps* must span across the development and operations areas to access the tools and services that are used by both.

The following topics describe some of the components that are available in this solution:

- The Open Platform for DBaaS on Power Systems hardware infrastructure:
  - OpenPOWER
  - Metal as a Service (MAAS)
- The Open Platform for DBaaS on Power Systems elastic cloud infrastructure:
  - KVM
  - Juju
  - OpenStack:
    - DB image library (Cinder)
    - DB archive (OpenStack Swift)
  - SDS (Ceph)
  - DBaaS services (Trove)
  - DBaaS Ops Mgmt

1.2.1 The Open Platform for DBaaS on Power Systems hardware infrastructure

The Open Platform for DBaaS on Power Systems solution is deployed on Power Systems servers, uses the Mellanox network switches to interconnect with each other and to the network infrastructure from the data center. The Power Systems servers perform the following roles:

- DB compute: Where the VMs of the database instances run
- DB archive storage: Where the object storage (OpenStack Swift) runs to back up the database instances
- DBaaS control plane: The controller nodes, where most of the OpenStack components run, and where all the intelligence behind the DBaaS happens
- DB block storage: The nodes that are used as SDS running Ceph to deliver block storage for the VMs

Figure 1-7 shows the Open Platform for DBaaS on Power Systems physical components.

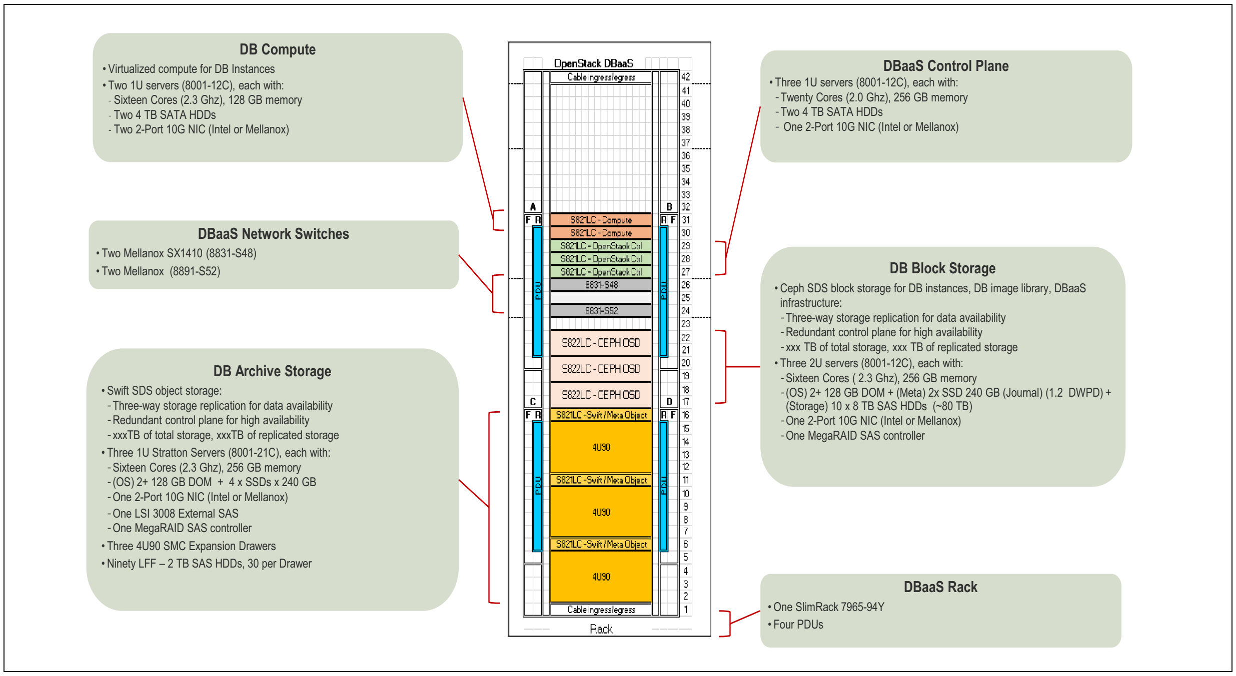


Figure 1-7 The Open Platform for DBaaS on Power Systems physical components

The solution is scalable, so the size and the number of servers vary according to your environmental; needs. For more information about the hardware components of the Open Platform for DBaaS on Power Systems solution, including the available sizes, see Chapter 3, “Architecture” on page 43.

OpenPOWER

The servers that are used in the Open Platform for DBaaS solution are IBM Power Systems S822LC and S821LC servers. They are flexible and powerful servers that use IBM POWER8® processors that are optimized for data-intensive workloads. These servers have an OpenPOWER foundation, and are an OEM between IBM and SuperMicro. They are powerful POWER8 servers with an attractive cost, optimized for Linux workloads (these servers do not have PowerVM hypervisor, hence do not use a Hardware Management Console (HMC)) with KVM virtualization features. For more information about IBM OpenPOWER LC servers, see IBM PowerLinux servers - OpenPOWER LC servers.

The Power Systems servers are a key component of the solution because all open databases are optimized for IBM POWER® processors, and these servers have a price/performance relationship that provides the best performance for data-centric workloads for open databases (Figure 1-8).

Figure 1-8 OpenPOWER advantages for running open databases[4]

Metal as a Service

The Open Platform for DBaaS on Power Systems solution uses the Canonical tool MAAS to provision the OS (Ubuntu) on the physical servers. MAAS manages the IBM Power Systems servers by using its management interface (baseboard management controller (BMC)) with Intelligent Platform Management Interface (IPMI) functions, and installs Ubuntu during the deployment steps, including a complete recipe to prepare the Open Platform for DBaaS on Power Systems solution.

The MAAS tool also calls Juju to deploy the OpenStack components of Open Platform for DBaaS to automate the deployment. For more information about MAAS, see What is MAAS?

Although the initial deployment by using MAAS is performed by IBM Lab Services, MAAS and Juju are installed on the Open Platform for DBaaS OpenStack controller node, enabling you to use it for scaling or recycling the solution if needed.

[4] The 80 - 100% price-performance advantage is based on the average of IBM internal measurements of Power System S822LC for Big Data relative to comparable x86 E5-2600v4 (Broadwell) 2-socket offerings on open source databases MongoDB and EDB. Comparisons use current pricing as of August 24, 2016. For more information, see IBM developerWorks: IBM Power Systems performance claims and proof points.

1.2.2 DBaaS elastic cloud infrastructure

The Open Platform for DBaaS on IBM Power Systems Software™ infrastructure relies on state-of-the-art open source software that is widely adopted in cloud environments and used by many customers worldwide. With these characteristics and standardized REST APIs, the Open Platform for DBaaS on Power Systems solution can be integrated with existing cloud platforms or cloud environments, as shown in Figure 1-9.

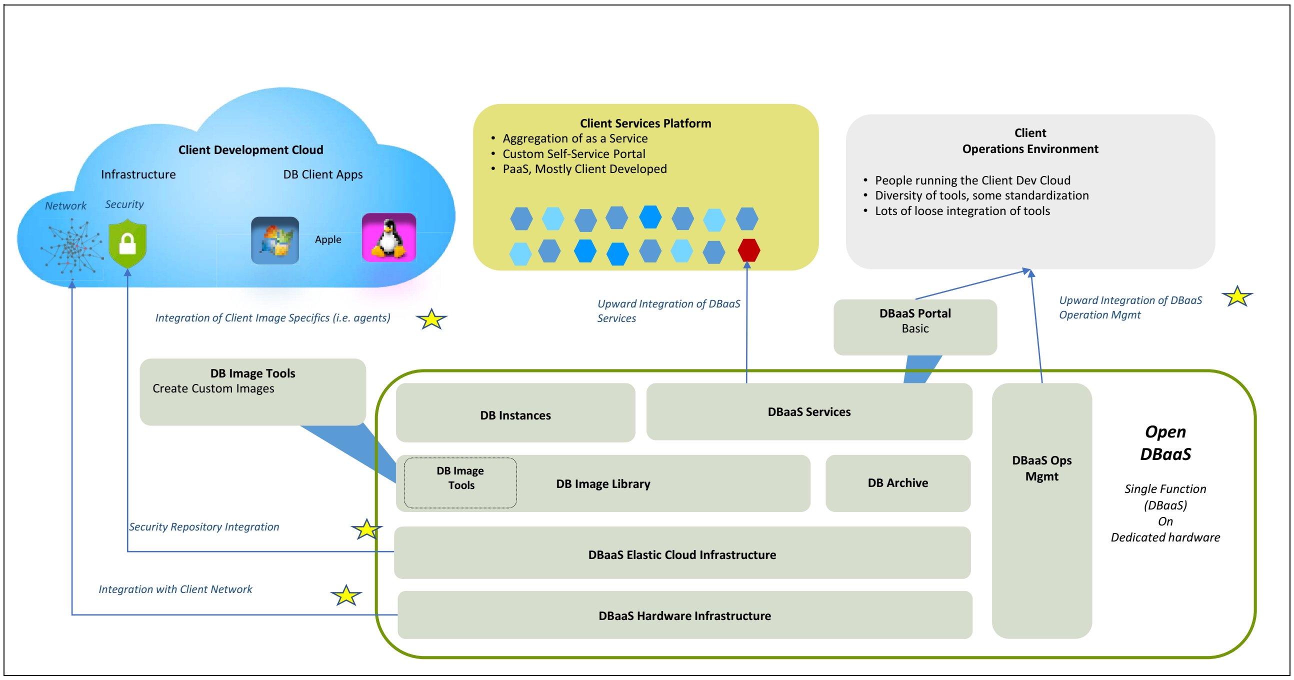


Figure 1-9 DBaaS integration on existing cloud environments

The following sections describe some of the open source software components that are used to implement the Open Platform for DBaaS on Power Systems solution.

Kernel-based Virtual Machine

The KVM virtualization feature runs on Linux, and transforms the Linux OS into a hypervisor, enabling it to run multiple VMs. KVM provides an open source virtualization choice for scale-out Power Systems servers, taking advantage of the performance, scalability, and security features of Power Systems servers, as illustrated by Figure 1-10.

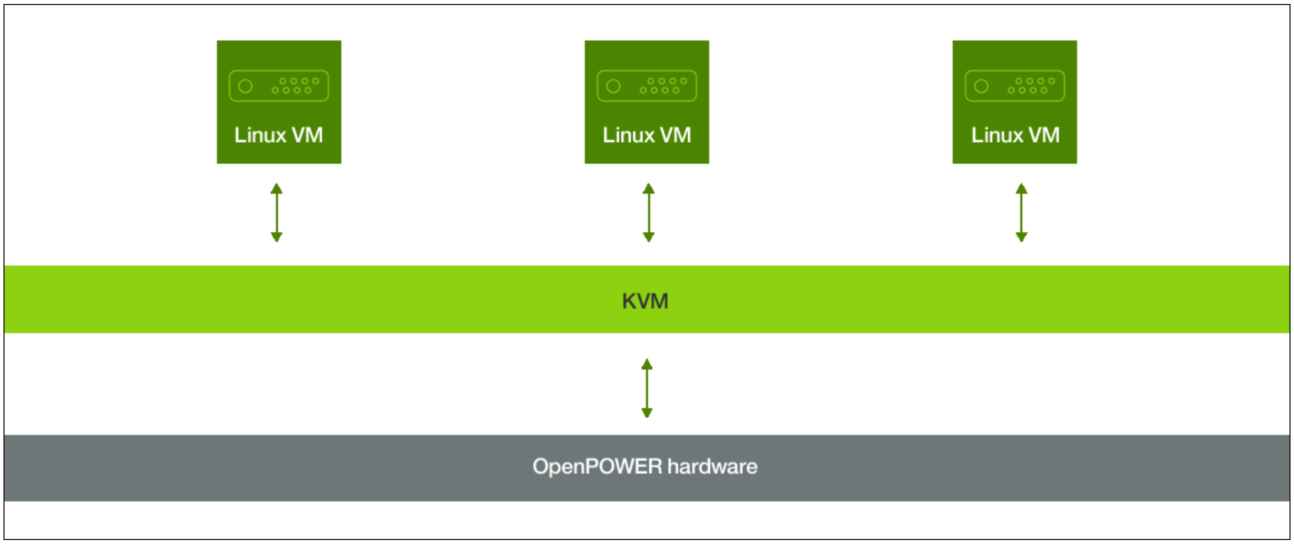


Figure 1-10 KVM on OpenPOWER hardware

Although running on Power Systems servers, KVM still uses the industry-standard features, so IT professionals who are already running KVM on x86 do not need additional training for working with KVM on Power Systems.

In an Open Platform for DBaaS on Power Systems environment, the KVM virtualization feature runs in an Ubuntu server on the OpenPOWER servers. For more information about KVM on Power Systems, see KVM on IBM Power Systems.

Juju

Juju is a service orchestration tool that is developed by Canonical and runs on Ubuntu. Juju is an open source tool that provides service modeling, application deployment, and application relationship management. Software can be quickly modeled, with application relationships defined and deployed in a cloud.

Juju uses Charms, which are structured bundles of YAML configuration files and scripts for a certain software component that enables Juju to deploy and manage software. For more information about Juju, see IBM Knowledge Center.

For sample Charms from IBM that are available on Ubuntu for Power Systems, see Search results for ibmcharmers.

The Open Platform for DBaaS on Power Systems solution uses Juju to deploy all the OpenStack software components to enable a cloud-based solution for open DBs. Juju charms are used to install OpenStack projects in the controller nodes, including Nova, Cinder, Neutron, Glance, and also Ceph (for block storage) and Swift (for object storage). Juju also automates the deployment of OpsMgr tools, such as Nagios and Elastic Stack (Kibana).

OpenStack

OpenStack is the OS for a cloud environment. Using multiple projects working together, OpenStack offers control for large pools of servers, storage, and networks in a data center, which builds a cloud. Together with such capabilities, OpenStack offers a Dashboard (called Horizon) for managing the resources. All these projects are standardized and constantly updated by the OpenStack community. IBM also contributes to the OpenStack development and provides compatibility with Power Systems.

By using OpenStack core projects, DBaaS delivers IaaS. The core projects are:

- Nova (compute): The compute service of OpenStack provides provisioning and management services of VMs by using supported Hypervisors, including KVM on Power Systems.
- Cinder (block storage): Cinder is the OpenStack project for block storage provisioning and management of VMs. Cinder provides virtual storage for the VMs in the cloud. Cinder integrates with the OpenStack Nova project and with many other storages, including IBM Storwize, SDSs, IBM Spectrum™ Scale, and Ceph. Cinder can be managed through the Horizon Dashboard.
- Neutron (network): Neutron is the network project for OpenStack that provides integration and management of networks, VLANs, and IP addresses (static or DHCP).
- Glance (images): Glance is the image repository project that is used by OpenStack. Glance provides an API-accessible service for handling images that are deployed in a cloud.
- Keystone (identity and security): Keystone is the project that is used by OpenStack to provide authentication and authorization mechanisms, controlling, for example, users and the projects that they can access within a cloud.

Figure 1-11 shows the OpenStack components and how they integrate with each other.

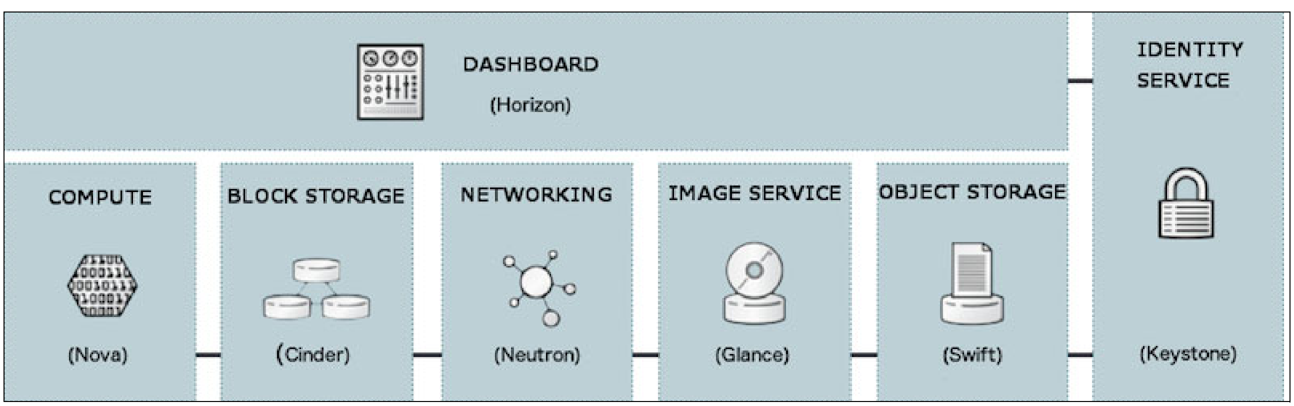


Figure 1-11 OpenStack components

The Open Platform for DBaaS on Power Systems solution uses OpenStack to provide a cloud environment and infrastructure. The DB image library (on top of Glance) and the DB archive (on top of OpenStack Swift) are described in more detail in the following sections.

DB image library (Glance)

The DB image library in Open Platform for DBaaS is the Glance image repository that contains the database images that the developer or DevOps professional can deploy. At the time this publication was written, the DB image repository comes with MariaDB, MongoDB, MySQL, PostgreSQL, and Redis database images, but more images are planned to be added to this repository. Also, more images can be customized and imported in the Open Platform for DBaaS on Power Systems solution, making it available to the users to deploy database instances. To prepare and import database images in the Open Platform for DBaaS on Power Systems solution, you need to use Juju Charms.

For more information about the databases that are supported in the Open Platform for DBaaS on Power Systems solution, see 2.3, "Supported databases" on page 39.

An alternative method for preparing the database image is by using the dbimage-builder project.

DB archive (OpenStack Swift)

The Open Platform for DBaaS on Power Systems solution uses the OpenStack Swift project, also known as the Object Storage project, which offers cloud storage software for handling distributed, non-structured data through a simple API. OpenStack Swift provides a mechanism for backup, archiving, and data retention.

The OpenStack Swift project is used by the Open Platform for DBaaS on Power Systems solution to automate database backups and restores.

Software-defined storage (Ceph)

The Open Platform for DBaaS on Power Systems solution uses Ceph as the SDS platform to provide block storage devices to the VMs that are deployed in the cloud, which is where the database instances run.

Ceph is an open source, SDS platform that implements object storage in a cluster by using commodity hardware. Ceph also provides interfaces for object, block, and file-level storage. The data that is stored in Ceph storage is striped and replicated between the Ceph nodes, ensuring reliability and performance.

The block storage mechanism Ceph uses relies on a technology that is called Reliable Autonomic Distributed Object Store (RADOS) Block Device (RBD). Ceph's client VMs view Ceph as a thin-provisioned block device, and when data is recorded in such block devices, Ceph replicates and stripes it across the Ceph cluster nodes, storing block device images as objects.

Because Ceph provides a Linux Kernel client and the KVM/Quick Emulator (QEMU) driver, the Open Platform for DBaaS on Power Systems solution takes advantage of it as a block storage provider to the database instances. The Ceph architecture, including the RBD mechanism is shown in Figure 1-12.

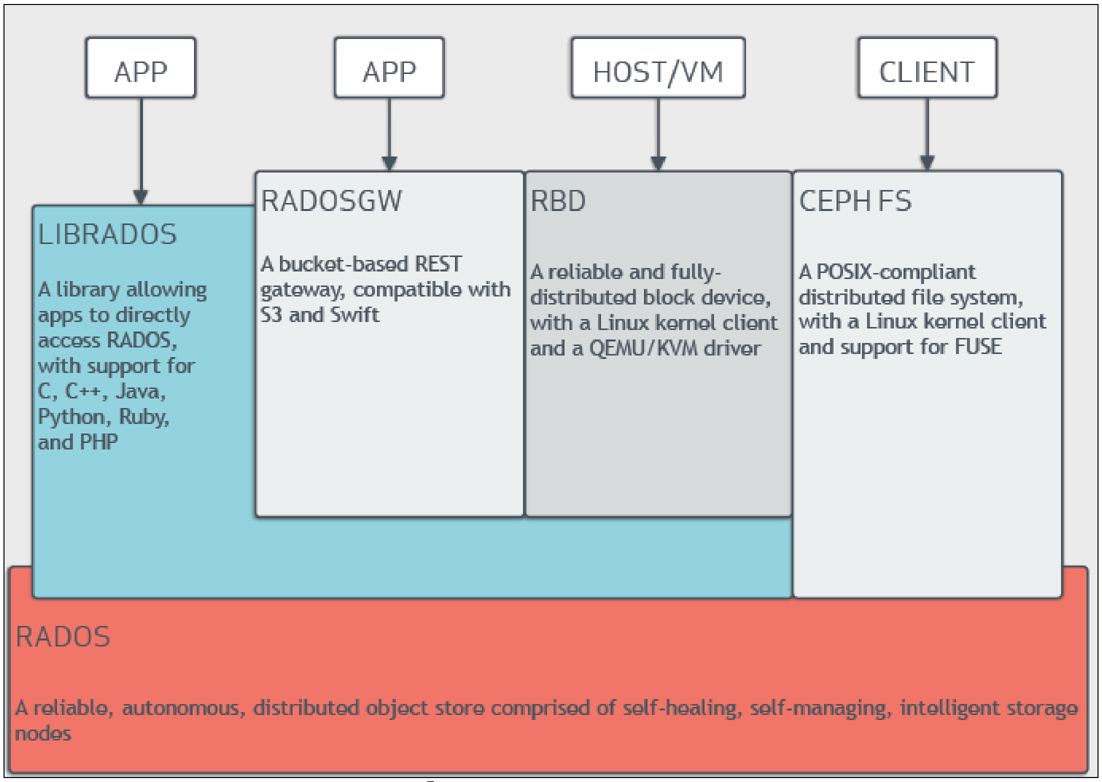


Figure 1-12 Ceph storage architecture[5]

DBaaS services (Trove)

The Trove OpenStack component is the core of this solution. It provides scalable and reliable provisioning functions for database engines (relational and non-relational). The Trove component is integrated with the remaining OpenStack projects, and provides tools to provision and manage many kinds of database engines, removing the complexity of handling all the particularities of a specific database.

For example, handling database operations on MongoDB is different than it is on MariaDB. The database engine MongoDB has its own concepts and particularities, so creating a database on MongoDB involves different actions than creating a database on MariaDB. With the Trove component, the Open Platform for DBaaS on Power Systems user does not need to know all these particularities because Trove offers a single interface to perform such operations.

[5] Source: Architecture - Ceph Documentation

With the Open Platform for DBaaS on Power Systems solution, the developer, operator, or DevOps professional can provision and manage multiple database instances with different engines by using Trove tools. The services that are provided by Trove include provisioning of database instances, creation of databases, instance resizing, instance restart, instance deletion, and user management, as shown in Figure 1-13.

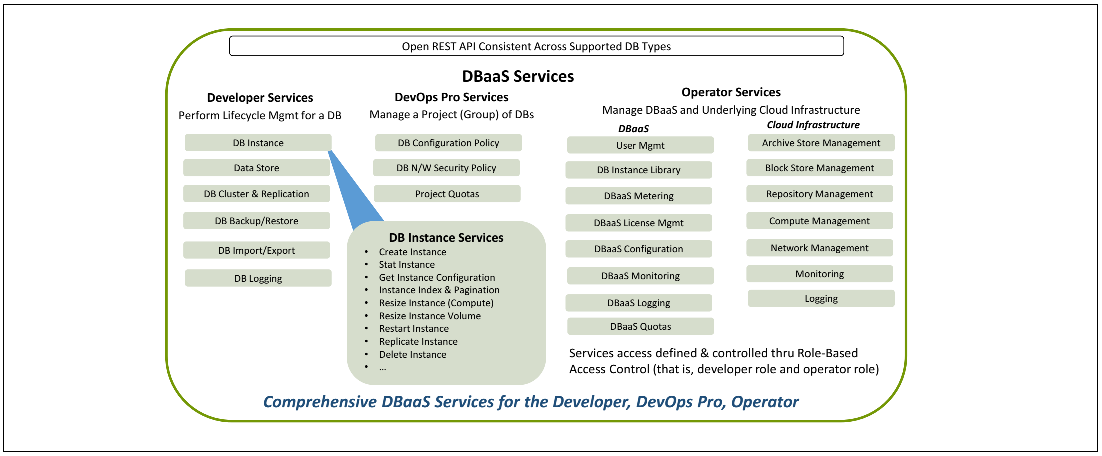


Figure 1-13 The Open Platform for DBaaS on Power Systems Trove services

Figure 1-14 shows the Open Platform for DBaaS Trove services that are categorized by areas of interest. It contains service, availability, security, and lifecycle management tools, which help development and DevOps professionals.

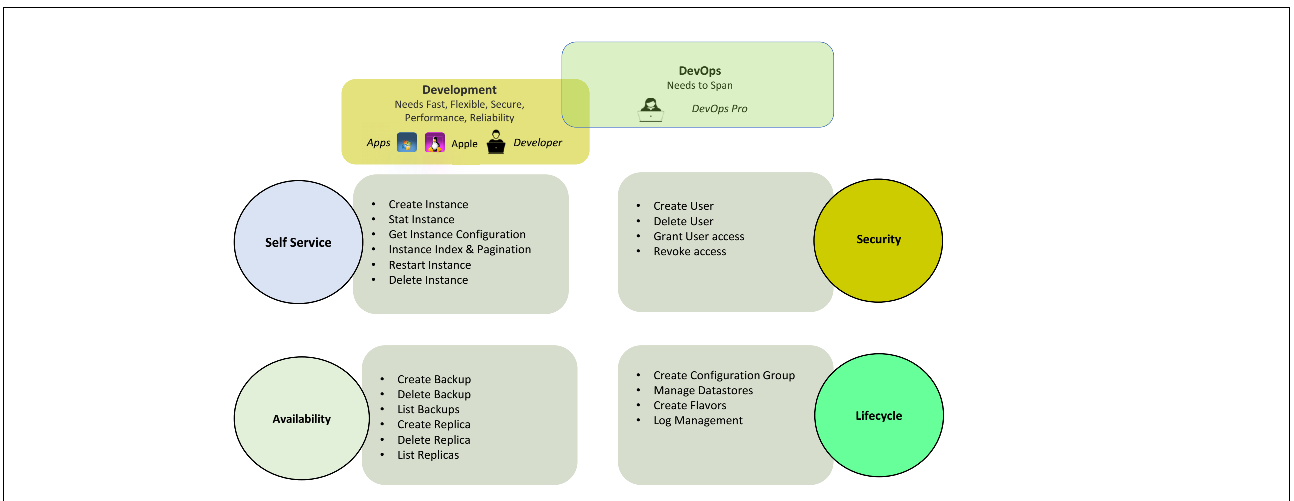


Figure 1-14 Trove services categorized

Most of the Trove operations are performed by a Trove conductor running in the Open Platform for DBaaS controller nodes, which communicates with the Trove Agent that runs inside each of the VMs that are deployed by the Open Platform for DBaaS on Power Systems solution, as shown in Figure 1-15.

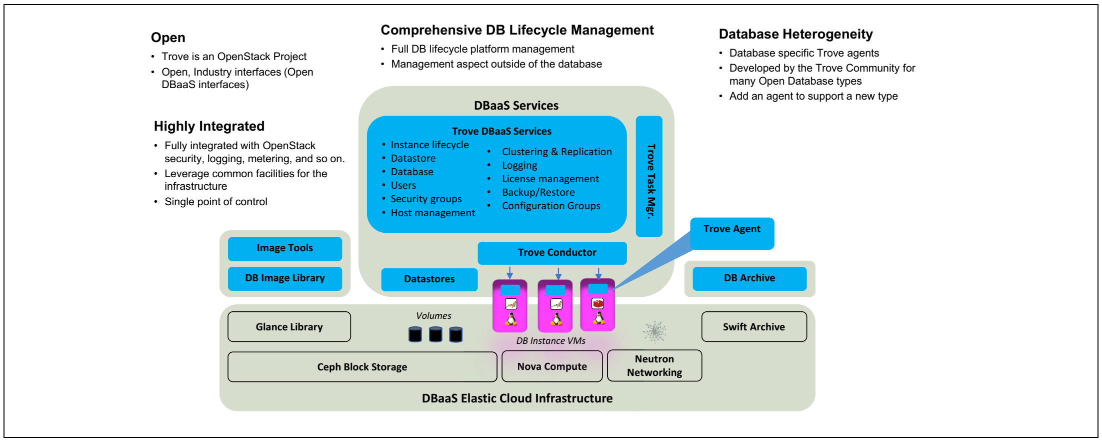


Figure 1-15 DBaaS Trove architecture

For more information about the Trove component, see Chapter 2, “IBM DevOps concepts” on page 25. The OpenStack Trove components that are used by the Open Platform for DBaaS on Power Systems solution are open source software that receives many contributions from the open source community and IBM. All development and contributions are upstreamed and available at GitHub.

DBaaS Ops Mgmt

The operators and administrators also need tools to manage the entire Open Platform for DBaaS on Power Systems solution. They must see logs and review statistics to ensure that Open Platform for DBaaS is healthy or to detect the source of any problem. The Open Platform for DBaaS on Power Systems solution comes with tools that are used and developed by the open source community that enable the operators to have good control of the entire solution. The components that are provided for OpsMgr purposes in the Open Platform for DBaaS on Power Systems solution are the following:

- Elastic stack**:** An integrated solution that is composed of three open source projects that provide a tool to retrieve, search, and analyze logs from an environment:
  - Elastic search: This is the search engine that is used by the elastic stack. It provides a full-text search engine with an HTTP web interface, enabling the operator or administrator to perform text searches by using the logs of the involved components in the Open Platform for DBaaS to detect and review error messages.
  - Logstash: Logstash is the open source software that receives data from all involved components in the Open Platform for DBaaS on Power Systems solution, processes and transforms it, and then sends it to the elastic search engine, enabling you to use it for the multiple logs (with different formats) by using a single interface.

– Kibana: This is the user interface where you take advantage of the elastic stack. It is an open source project for data visualization that is used with elastic search. Kibana provides visualization capabilities for the data that is obtained and indexed by Logstash in an elastic search environment. This is the interface that you use to analyze the logs, and generate and review graphics with the data.

- Nagios**:** This is the monitoring tool that is used in the Open Platform for DBaaS on Power Systems solution. It is an open source application that monitors the servers and network switches from the Open Platform for DBaaS on Power Systems solution and sends alerts to the operators and administrators to act so that you know when something is wrong with your environment.

For more information about the Kibana and Nagios user interfaces, see Chapter 5, “Monitoring and troubleshooting” on page 123.

1.3 Open Platform for DBaaS on Power Systems product delivery and support flow

The initial deployment of the Open Platform for DBaaS on Power Systems solution (using MAAS and Juju) is performed by IBM Lab Services before shipping the solution to the customer. Then, after receiving the system, IBM Lab Services is reengaged onsite to perform the initial start, network configuration, and validation, as shown in Figure 1-16.

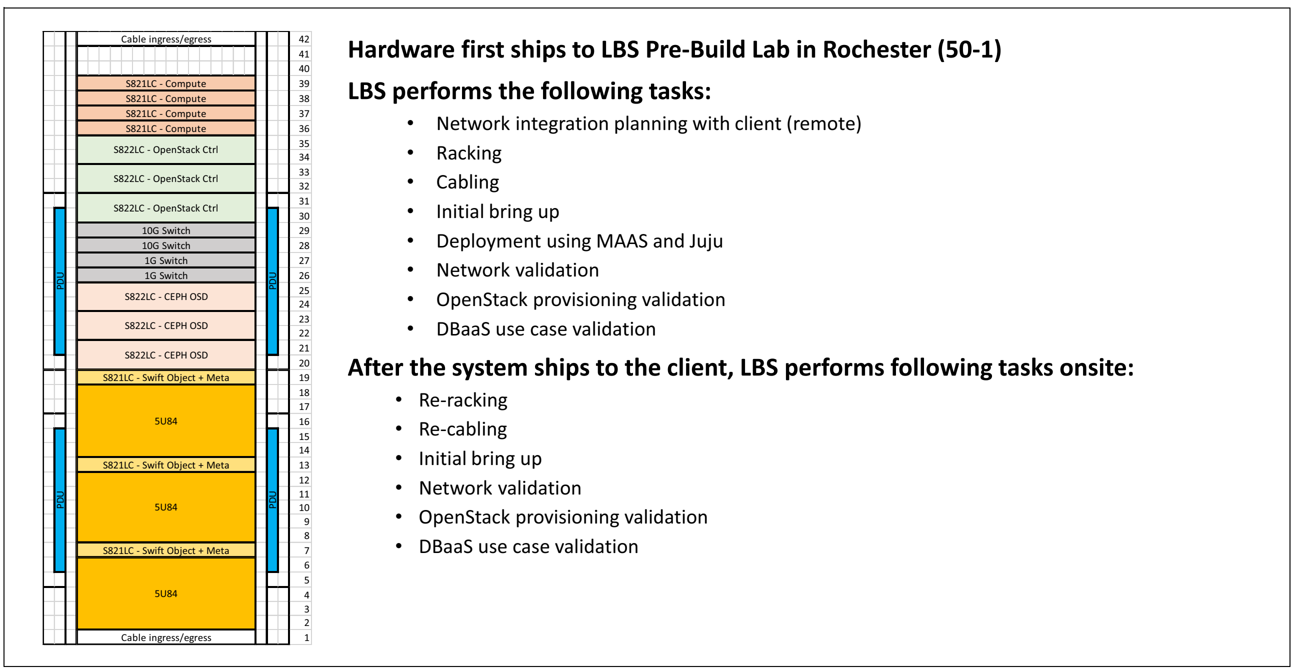


Figure 1-16 IBM Lab Services deployment of the Open Platform for DBaaS on Power Systems solution

The following task list details the activities that are performed by IBM Lab Services at either IBM or at the customer's location:

- Pre-installation and planning:
  - Planning session and logistics (onsite or remote).
  - Create the implementation documentation (racking diagram, cabling diagram, network topology, IP table).
- Hardware racking and cabling - IBM location:
  - Validate the bill of materials.
  - Install switches in racks and label them.
  - Install systems in racks and label them.
  - Install network cables between systems and switches, and label them.
  - Box systems after testing.
- Network switch preparation - IBM location:
  - Discover switches.
  - Set the switch IP addresses.
- Systems preparation - Rochester:
  - Update the firmware levels on all nodes as required.
  - Install and configure the encryption keys (optional).
  - Set BMC on all nodes to obtain the IP address through DHCP.
  - Set the IP address of the management node where the provisioning tool will be run.
  - Manually provision the management node for the provisioning tool.
  - Install the provisioning tool.
  - Edit the provisioning tool configuration file.
- OS provisioning - IBM location:
  - Run the provisioning tool.
  - Provision the OSs.
  - Provision the OpenStack environment (management, compute, and network nodes).
  - Customize the network configuration for items that are beyond the capability of the provisioning tool.
  - Perform the postinstallation cleanup and augmentation (additional packages are required).
  - Validate the provisioning tool results.
  - Deploy OpenStack and delete the VMs.
- Base Trove/DB provisioning - IBM location:
  - Validate the base Trove installation (performed by tools in a previous stage).
  - Add DB-specific Trove templates.
  - Validate the Trove functions.
  - Provision/deprovision the DB instance.
  - Provision the DB user.

- Onsite installation at the client site:
  - Rerack and recable systems (cannot be shipped with the rack).
  - Integrate the network.
  - Validate the OpenStack VM provisioning (by using client images).
  - Validate the Trove DB provisioning (by using client images).
  - Transfer client skills.

For the Open Platform for DBaaS on Power Systems implementation to succeed, you must have an internet connection at the customer's location for the Open Platform for DBaaS cluster (NAT is acceptable). Also, the IBM Lab Services Specialist must connect the IBM notebook to the Open Platform for DBaaS on Power Systems cluster. The customer must be available for 1 or 2 days for an onsite planning session (servers and network teams). At least one person from the customer must be available full time to work with the IBM Specialist during the implementation and testing phase.

The Open Platform for DBaaS on Power Systems solution receives contributions from several sources, which ensure quick development, bug fixes, new features, and improvements in general. The OpenStack community developers add features to the OpenStack projects, Canonical improves Ubuntu, MAAS, Juju and its Charms infrastructure, IBM develops and ports such features to the Open Platform for DBaaS on Power Systems solution, and upstream its contributions, so users benefit from all sources working on the open source solution.

IBM Technical Support Services is also engaged to provide technical support for questions, how to information, usage, problems, defects, and critical situations that can happen with this solution. Level 1 and Level 2 technical support (development support) work with Rogue Wave Software to fix issues in the code in case of any problem, even with the open source components of the solution. Such fixes are also incorporated by Open Platform for DBaaS on Power Systems developers and then upstreamed to the code.

Figure 1-17 shows the offering flow from the many contributors for the Open Platform for DBaaS on Power Systems solution and how you can benefit from it.

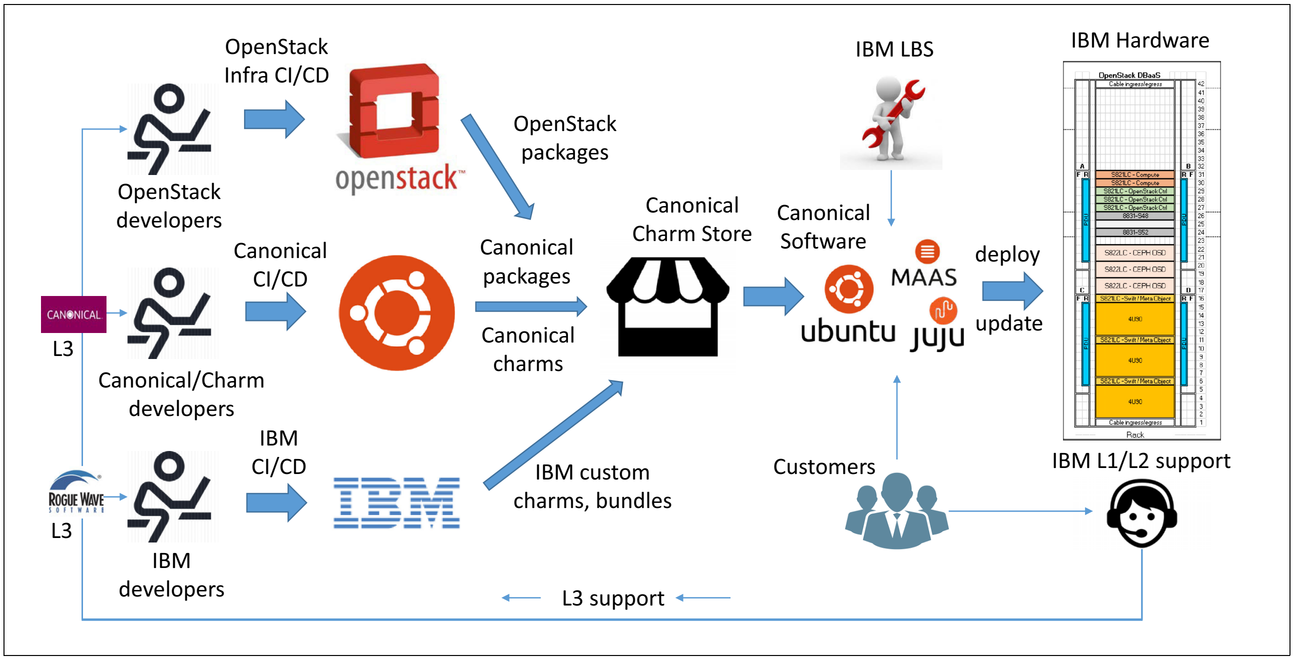


Figure 1-17 Open Platform for DBaaS on Power Systems offering flow

2

IBM DevOps concepts

IBM DevOps is an approach that promotes closer collaboration between lines of business (LOBs), development, and IT operations. DevOps is an enterprise capability that enables the continuous delivery, continuous deployment, and continuous monitoring of applications. This chapter provides information about DevOps concepts and approaches, and describes how it applies to Infrastructure as a Service (IaaS).

This chapter contains the following sections:

- IBM DevOps
- Infrastructure as a Service+
- Supported databases

2.1 IBM DevOps

The business changes that are driven by cloud, analytics, mobile, and social technologies are unprecedented in their speed and scope. In the current business environment, product and service delivery processes must be optimized for innovation and time-to-market.

Organizations are embracing approaches to software development that focus on the customer. By increasing the frequency of software delivery and reducing the time-to-feedback from customers, organizations can respond faster to shifts in the market and keep customers happy.

The increased release frequency demands tighter alignment and collaboration than are seen traditionally between LOBs, development, and IT operations, which drives the requirement for enhanced collaboration, automation, and information transparency among these groups. To achieve this seamless internal cooperation and promote sustained innovation across the enterprise, IBM recommends the adoption of DevOps.

DevOps is the practice of bringing together process and stakeholders in an organization who develop, operate, or benefit from the business' software systems, enabling continuous delivery of value to users by applying agile and lean thinking principles.

By extending lean principles across the entire software supply chain, DevOps capabilities enable businesses to maximize the speed of delivery of a product or service, from initial idea to production, release to customer feedback, and to enhancements based on that feedback.

DevOps also helps the unification of measurements and collaboration across the organization, reducing expenses, duplication, and rework, and replacing the isolated development and operations silos to create a multidisciplinary, well-integrated team participating in the entire application or service lifecycle.

Organizations that practice DevOps successfully tend to adopt the following processes and technologies:

Design thinking Focusing on delivering exceptional user experiences and increasing user conversion.

Lean startup Validating ideas and testing possible solutions before committing significant personnel, and helping organizations to stay focused on solving the problems that matter.

Agile As the development methodology for fast feedback cycles through early customer involvement.

Continuous security Eliminating security vulnerabilities from applications before they reach production.

Delivery automation Removing the silos between development and IT operations, and enabling the continuous delivery of changes.

Application monitoring Quickly detecting and addressing software application issues in test and production environments.

Application and user analytics Continuous learning is used to improve the quality and value of applications.

Squads Small, cross-functional, self-organizing teams that own end-to-end responsibility.

The value of DevOps can be illustrated as an innovation and delivery lifecycle, with a continuous feedback loop to learn and respond to customer needs. IBM has identified six phases in the DevOps lifecycle and six main DevOps practices for successful implementation of a DevOps approach, as shown in Figure 2-1 and Figure 2-2 on page 28.

2.1.1 DevOps lifecycle phases

The following list describes the DevOps lifecycle phases (Figure 2-1):

- Think: Conceptualization, refinement, and prioritization of capabilities.
- Code: Generation, enhancement, optimization, and testing of features.
- Deliver: Automated production and delivery of offerings.
- Run: Services, options, and capabilities that are required to run.
- Manage: Ongoing monitoring, support, and recovery of offerings.
- Learn: Continuous learning based on outcomes from experiments.

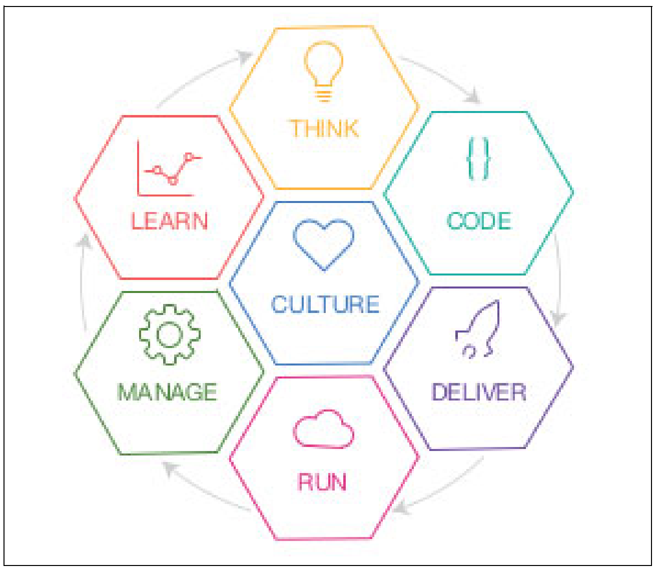


Figure 2-1 IBM DevOps lifecycle phases

2.1.2 DevOps practices

This section describes the DevOps practices that are shown in Figure 2-2.

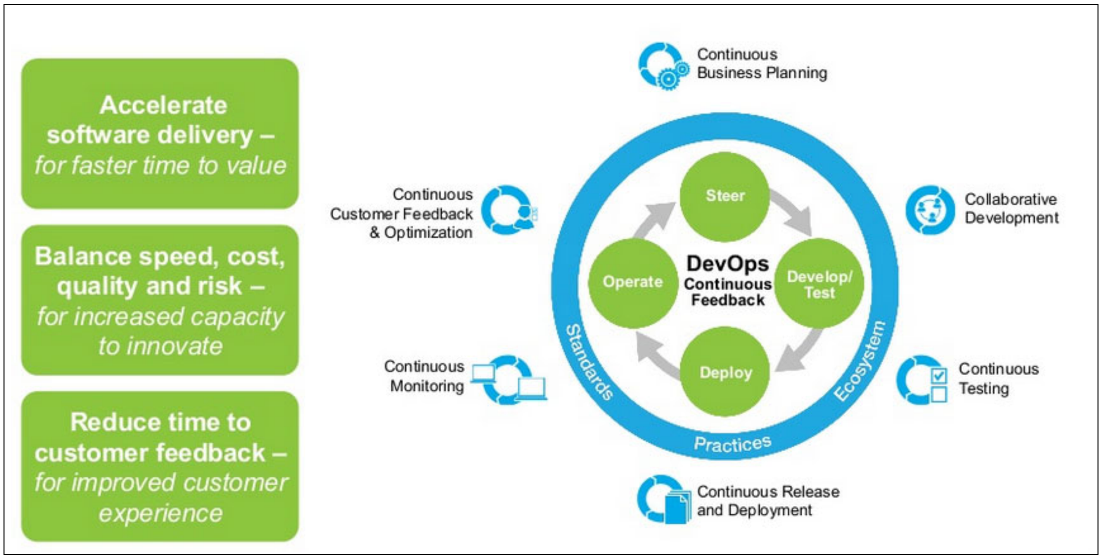


Figure 2-2 IBM DevOps practices

Continuous business planning

This practice employs lean principles to start small by identifying the outcomes and resources that are needed to test the business vision and value, to adapt and adjust continually, measure current progress, and learn what customers really want and shift direction with agility and update the plan.

Collaborative development

This practice enables collaboration between business, development, and quality assurance (QA). Includes support for multiple programming languages, and supports multiplatform development and elaboration of ideas.

Collaborative development also includes *continuous integration*, which promotes frequent team integrations and automatic builds. By integrating the system more frequently, integration issues are identified earlier when they are easier to fix, and the overall integration effort is reduced by continuous feedback because the project shows constant and demonstrable progress.

Continuous testing

This practice eliminates testing bottlenecks through virtualized dependent services, and simplifies the creation of virtualized test environments that can be easily deployed, shared, and updated as systems change.

This practice also reduces the cost of provisioning and maintaining test environments and shortens test cycle times by allowing integration testing earlier in lifecycle while helping development teams balance quality and speed.

Continuous release and deployment

This practice provides a continuous delivery pipeline that automates deployments to test and production environments. It reduces the amount of manual labor, resource wait-time, and rework by using push-button deployments that enable higher frequency of releases, reduced errors, and end-to-end transparency for compliance.

Continuous monitoring

This practice offers easy-to-use reporting that helps developers and testers understand the performance and availability of their application, even before it is deployed to production.

The early feedback that is provided by continuous monitoring is vital for lowering the cost of errors and changes, and for steering projects toward successful completion.

Continuous customer feedback and optimization

This practice provides the visual evidence and full context for analyzing customer behavior and difficulties. Feedback can be applied during both pre- and post-production phases to maximize the value of every customer visit and ensure that more transactions are completed successfully. This practice enables immediate visibility into the sources of customer struggles that affect their behavior and impact business.

2.1.3 Cloud Infrastructure as a Service and DevOps

Cloud and DevOps are not mutually exclusive, as both of them are catalyzers for each other. Cloud brings the ability to dynamically provision and scale test production environments so that the DevOps workloads hosting is easier, faster, and flexible. In return, DevOps brings agile transformation for cloud services, reducing deployment times for services and infrastructures, improving the lifecycle management, and delivering continuous improvement processes.

When adopting DevOps on cloud, IBM identifies two general profiles:

- Cloud-native profile
- Cloud-enabled profile

Cloud-native profile

This profile is characterized by small teams working to shorten delivery cycles, and are focused on effectiveness, and user or business outcomes. These teams focus on engaging users across multiple touch points, including mobile platforms and social media.

The cloud-native workloads are often based on a microservices architecture to enable agility in change and deployment, and reuse existing web services (data management, analytics, cognitive processing, Internet of Things, and so on) to speed their development time.

Figure 2-3 summarizes the main characteristics of a cloud-native profile.

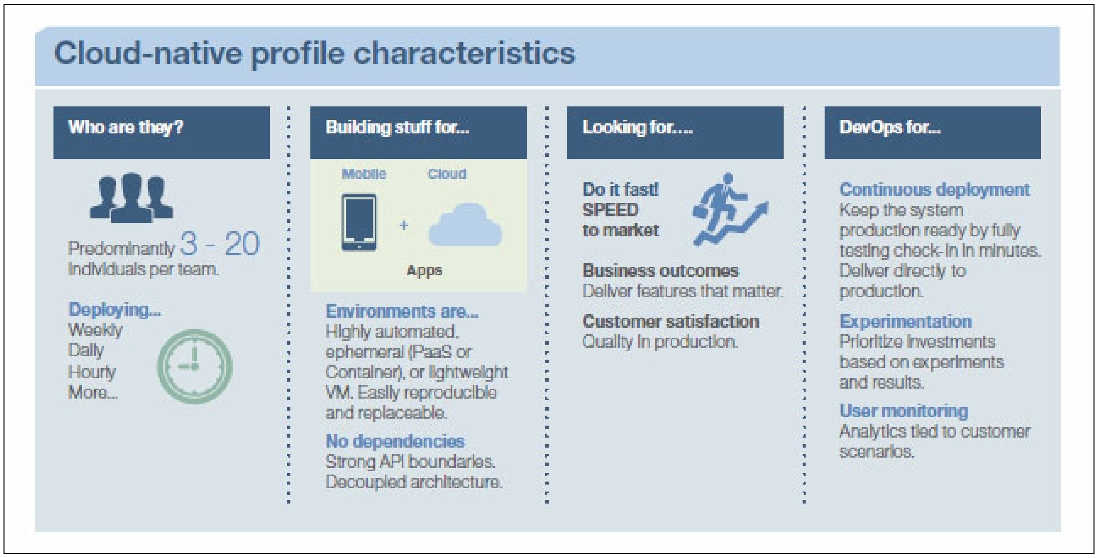


Figure 2-3 Cloud-native profile

Cloud-enabled profile

This profile is characterized by teams of teams working on longer delivery cycles who are focused on quality improvements, faster time-to-market, and balancing cost and value.

These teams manage different architectures that tend to be complex due to many dependencies, and APIs are used to bridge the pre-cloud environments and the new cloud-based environments. The workloads can run across multiple environments: on-premises, private cloud, or hybrid clouds.

Figure 2-4 summarizes the main characteristics of a cloud-enabled profile.

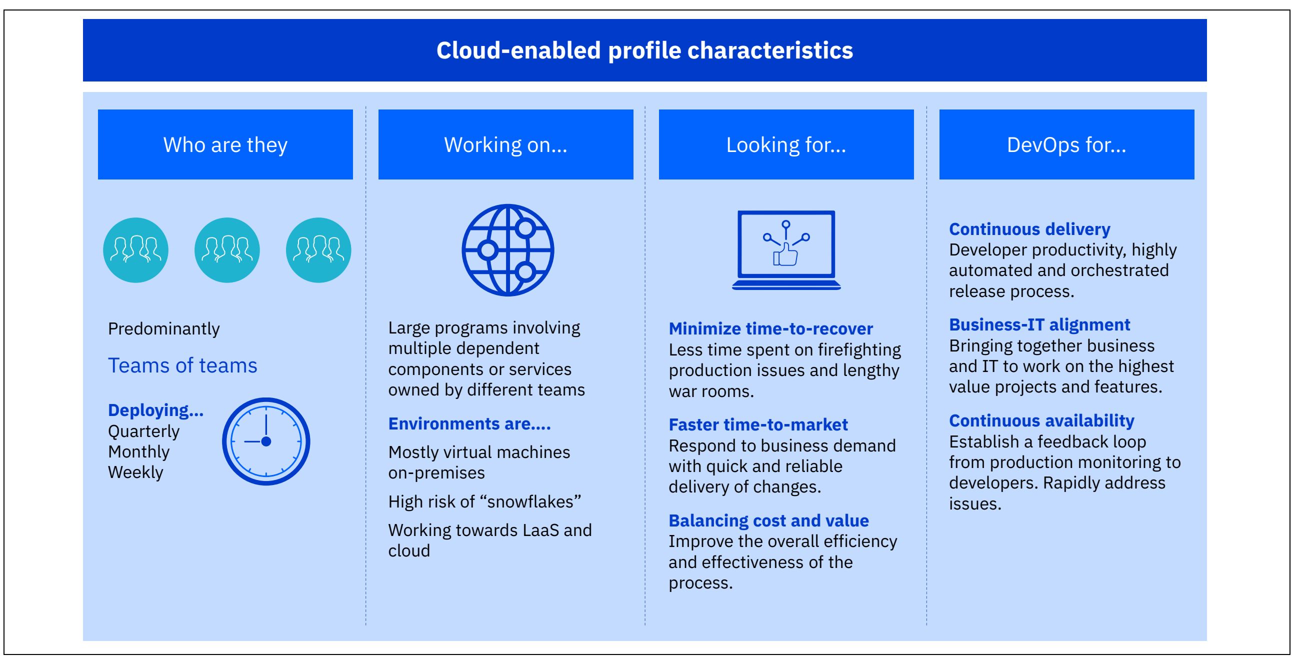


Figure 2-4 Cloud-enabled profile

IBM provides the infrastructure to host DevOps workloads and offers a toolset to help customers implement DevOps, as shown in Figure 2-5.

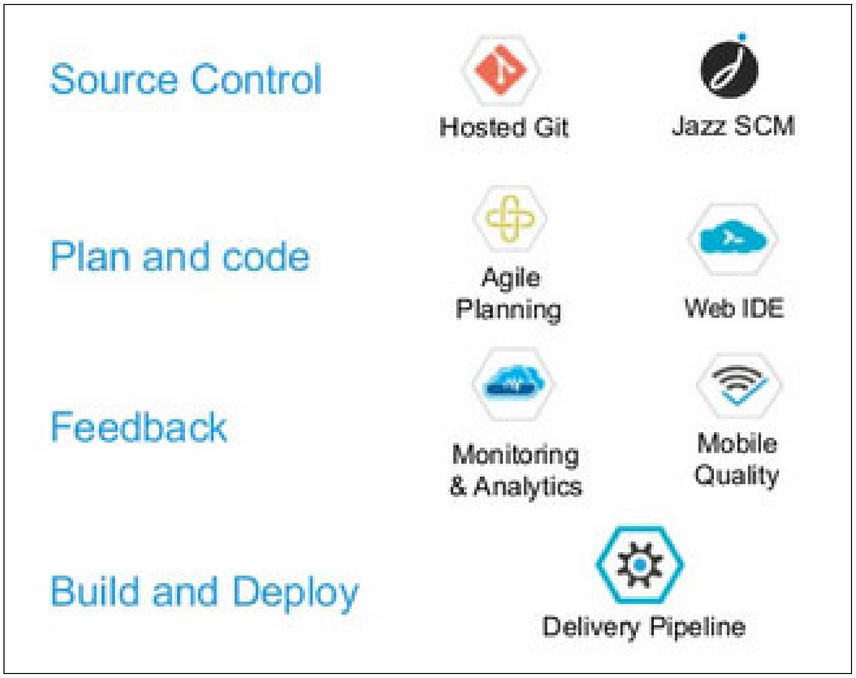


Figure 2-5 IBM DevOps Services

> **Note:** For more information about IBM DevOps Services, see DevOps toolchains.

2.1.4 Infrastructure as code

When an IaaS cloud model is deployed, the cloud provider is responsible for all the underlying infrastructure: virtualization, servers, networking, and storage.

DevOps can be an enhancer for this underlaying infrastructure. Through different tools, DevOps provides a new, agile way to provision and manage infrastructures automatically through code rather than manual or hardware-centric processes. This process is known as *infrastructure as code (IaC)*.

IaC or programmable infrastructure is the process of configuring and managing IT infrastructures, both physical and virtual, through definition files that are based on descriptive language, which provides versatile and adaptive provisioning, updating, and deployment processes.

When the infrastructure is conceptualized through code, it achieves a high level of automation, which offers the following advantages:

- Cost reduction: By eliminating manual processes, people can focus their efforts on important tasks, and avoid repetitive tasks or duplicate efforts. Thus, cost reduction in terms of effort and people is achieved.
- Faster execution: Through automation, a faster execution of infrastructure configuration is achieved. IaC provides visibility and traceability to all the involved equipment, enabling them to work quickly and efficiently.

- Human error reduction: Automation eliminates the risk of misconfigurations that are associated with human error, which reduces downtime and increases the reliability of deployments.

Figure 2-6 shows how the IaC approach automates infrastructures.

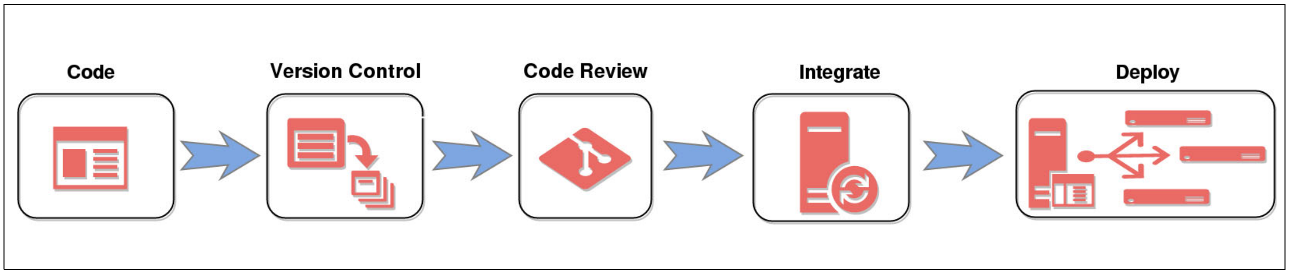


Figure 2-6 Infrastructure as code

IaC has three key elements:

- Approaches: The approaches define how the configuration, changes, management, and deployment of the infrastructure are addressed. There are three main approaches:

  Declarative Focuses on what is the wanted state for the infrastructure, and what can be done to achieve the final wanted state.

  Imperative Focuses on how the infrastructure is modified to achieve something by defining the specific commands and their order.

  Intelligent Focuses on understanding why the infrastructure must be a certain way based on the co-relationships and dependencies of the infrastructure elements. Determines the correct state of the infrastructure before running what is necessary to reach that state.

- Methods: The methods define how the different elements of the infrastructure are configured. There are two basic methods:

  Put The system to be configured extracts its configuration from a control server through an agent that is installed on the system.

  Push A control server pushes the configuration to the target system, usually through an ssh service. This is an agent-less method.

- Tools: Tools change, configure, and automate the entire infrastructure. They provide visibility and traceability of infrastructure configurations and deployments, making management more flexible. Tools can offer one or more of the following functions:

  Code Develops the code that is responsible for configurations and deployments.

  Version control Enables traceability to different versions of the created code, giving visibility to teams about changes and improvements that are made.

  Code review Reviews code to ensure it does not contain errors and to identify improvements.

  Integrate Continuous integration of the developed code, and the projects of the teams that are involved.

  Deployment Provides configuration and continuous deployment.

Table 2-1 on page 33 shows a comparison of the main infrastructure automation tools.

Table 2-1 Comparison of the main infrastructure automation tools

Tool	Approach	Method	Language
Ansible	Imperative	Push (can be configured to work under Pull method)	Written in Python and enables users to script commands in YAML.
Chef	Imperative	Pull	Written in Ruby-DSL (Domain-Specific Language).
Puppet	Declarative	Pull	Written in Ruby and offers custom domain-specific language (DSL) and Embedded Ruby (ERB) templates.
SaltStack	Imperative	Push	Written in Python and allows users to script commands in YAML.

Important: IaC does not replace IaaS, and it is not a new cloud service model. It makes IaaS more agile through DevOps tools.

2.2 Infrastructure as a Service+

When talking about cloud-based services, IaaS has been one of the most implemented services models. But, the ability to offer everything as-a-service has led the Platform-as-a-Service (PaaS) model to being deployed over an existing IaaS layer, which results in the need for both layers of the stack to be efficiently integrated.

Figure 2-7 shows how IBM Bluemix® is the result of the integration of a PaaS layer with an IaaS layer.

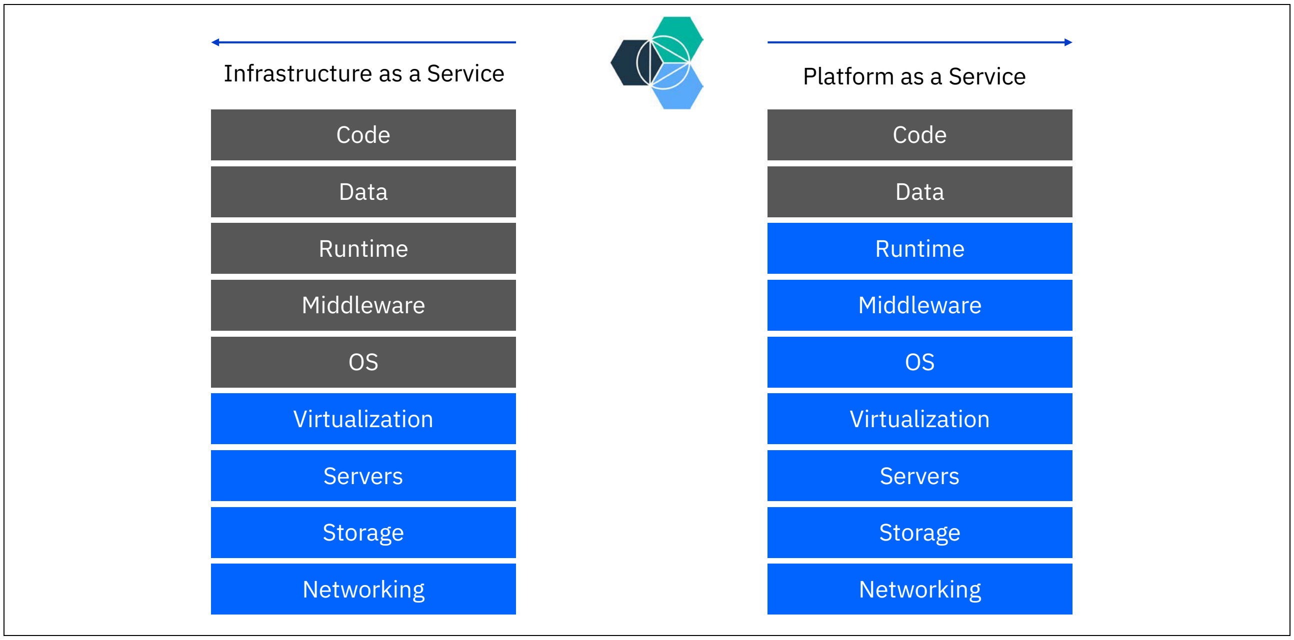


Figure 2-7 IBM Bluemix layers

IaaS+ is based on a set of modular services that are started in the underlying infrastructure, which are in charge of complementing the services that are started in the PaaS layer. IaaS+ provides three important features for IaaS and PaaS integration:

1. Abstraction: It offers a level of abstraction for the development layer, so that PaaS users need not worry about complex implementations and configurations of basic components for the development and hosting of applications, such as databases, message queues, and DNS.
2. Modularity: Each component is implemented as a module that can be easily added to the infrastructure, and can be updated and improved independently.
3. As-a-service: The modules have an as-a-service nature, which ensures multi-tenant deployments and on-demand scalability.

Some of these modular services are the following:

- Database
- DNS
- Cache
- Big data
- Messaging
- Telemetry

2.2.1 Open Platform for Database as a Service on Power Systems

The Open Platform for Database as a Service (DBaaS) on Power Systems solution refers to a cloud service where cloud consumers can provision, manage, consume, configure, and operate open source databases software without having to take care of the entire implementation and configuration process for the specific open source database. Users can deploy an open source database of their choice from a catalog of supported databases, and be charged according to their usage of the service.

This service model offers enough abstraction to manage complex database operational tasks, such as storage allocation, backups, recoveries, database upgrades, and cluster configuration and resizing.

The Open Platform for DBaaS on Power Systems solution has unique advantages over traditional database deployment models:

- Improves speed and agility with on-demand and agile provisioning of databases, which are requirements for today's cloud workloads.
- Helps reduce licensing and infrastructure costs.
- Eliminates database sprawl that is generated by the hundreds or thousands of underutilized or unutilized databases that have accumulated in organizations.

Figure 2-8 shows the categories of service that the Open Platform for DBaaS on Power Systems solution offers.

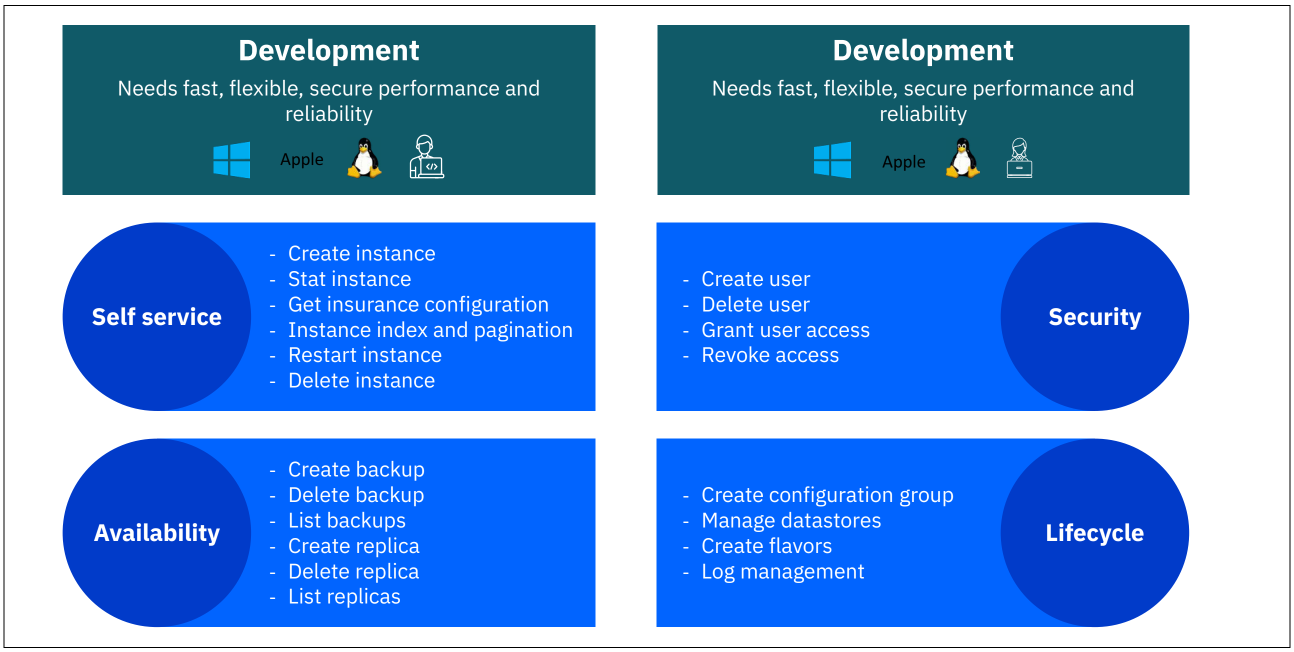


Figure 2-8 The Open Platform for DBaaS on Power Systems solution categories

2.2.2 Trove

Trove is the database as a service OpenStack IaaS+ module that provides resource isolation, abstraction, and automation of complex database administrative tasks of multiple database instances, either relational or non-relational.

Trove is based on a *pluggable approach* that provides a unique framework in which many different databases are supported. This framework provides a set of APIs that manage the native processes for each supported database, so the management and operations are unified through standard interfaces: CLI, RESTful APIs, and a web GUI.

Trove also enables you to directly control the cluster where the database is deployed, enabling users to create and resize database clusters directly through Trove interfaces.

Trove abstracts the entire database deployment by interacting directly with the OpenStack IaaS core components. Trove communicates with Nova compute service to create virtual machines (VMs) on which to run database servers, with Cinder to connect to the Ceph block storage back end to provide database storage, and with Swift's object storage to capture backups.

Figure 2-9 shows how Trove abstracts and handles the database instances management lifecycle.

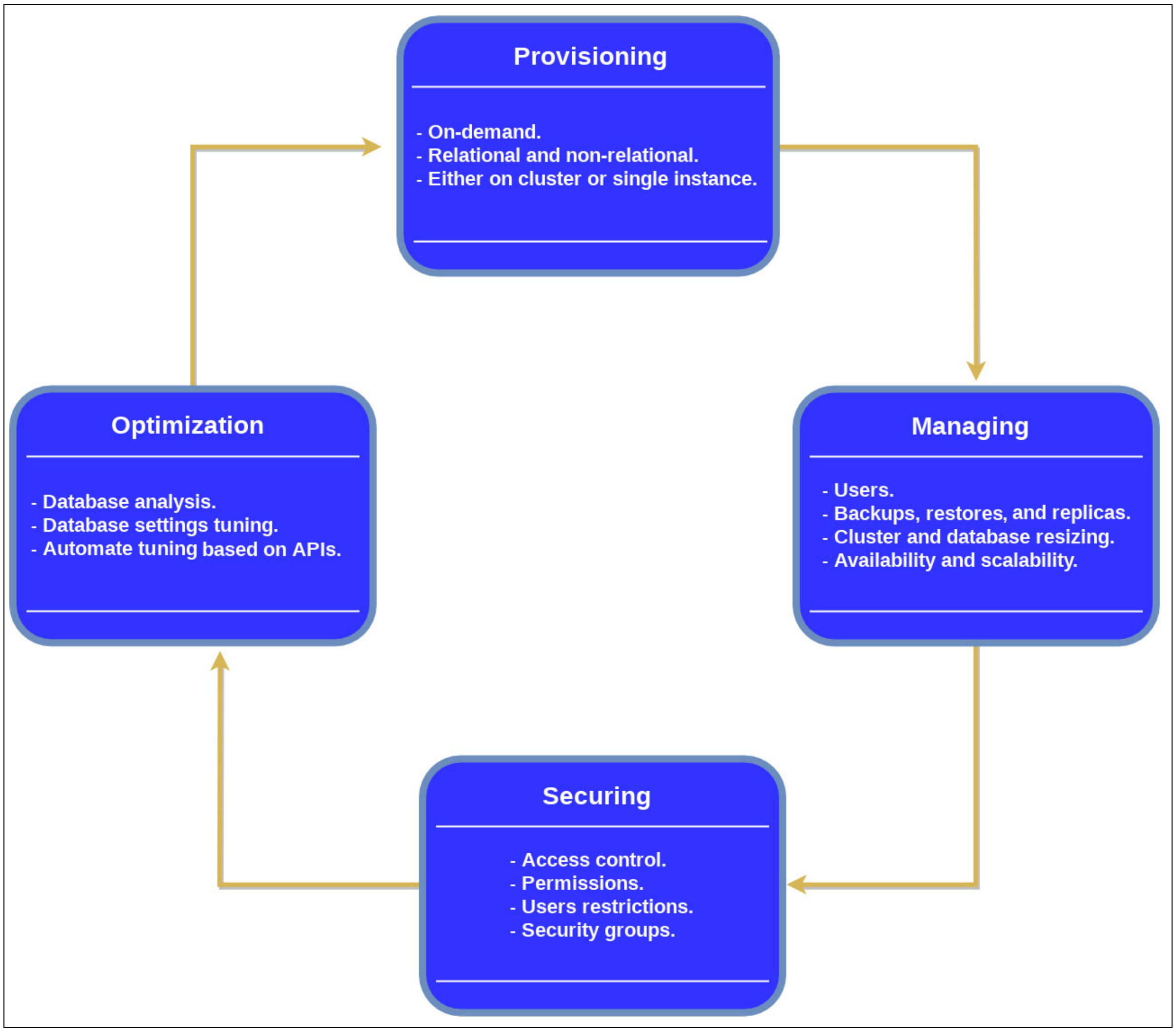


Figure 2-9 Database instances management lifecycle

Architecture

Trove architecture is based on a share-nothing messaging system over HTTP, where each component communicates to others over a message bus. Trove's components can be differentiated between those running on the server side (Trove-api, Message bus, and Trove-taskmanager), the virtual resource side (Trove-guestagent and guest image) and the host side (Trove-conductor).

Trove-api

Trove-api is an API server that handles the authentication, authorization, and control of basic functions that are related to guests agents and data stores. It focuses on taking requests from users, converting them into messages, validating them, and forwarding them either to the *task manager* to handle complex management tasks, or to the *guest agent* to manage simple database tasks. It supports JSON and XML APIs.

Message bus

This is an Advanced Message Queuing Protocol (AMQP) message bus that handles the interaction between the API end points, the task manager, the conductor, and the guest agent.

Back-end database

The back-end database is used by Trove to store the state of the system. This database can be MySQL or MariaDB.

Trove-taskmanager

Trove-taskmanager is a database-neutral service that runs complex tasks that are related to the provisioning and the administration of an instance lifecycle.

This component abstracts heavy database tasks such as backup, replicas, or database resizing so that users do not need to worry about the specific calls that are required to perform such tasks for each particular database.

Trove-guestagent

Trove-guestagent is the service that runs within the guest database instance. It converts the commands from APIs into the language of the specific database, and is responsible for managing and performing operations on data stores. Each guest agent is different and performs tasks according to the particular data store they manage.

Guest image

The guest image is the VM image that bundles the database server along with the proper guest agent. This ready-to-use bundle avoids the need to create database instances from scratch.

Guest images can be pre-built and configured, and can be downloaded from external sources.

Trove-conductor

Trove-conductor is a service that runs on the host, collects information and statuses from guest agents, and writes them into Trove's back-end database. By using conductor services, guest instances do not need a direct connection to the back-end database.

Figure 2-10 shows the high-level interaction of Trove's architectural components with IaaS core components.

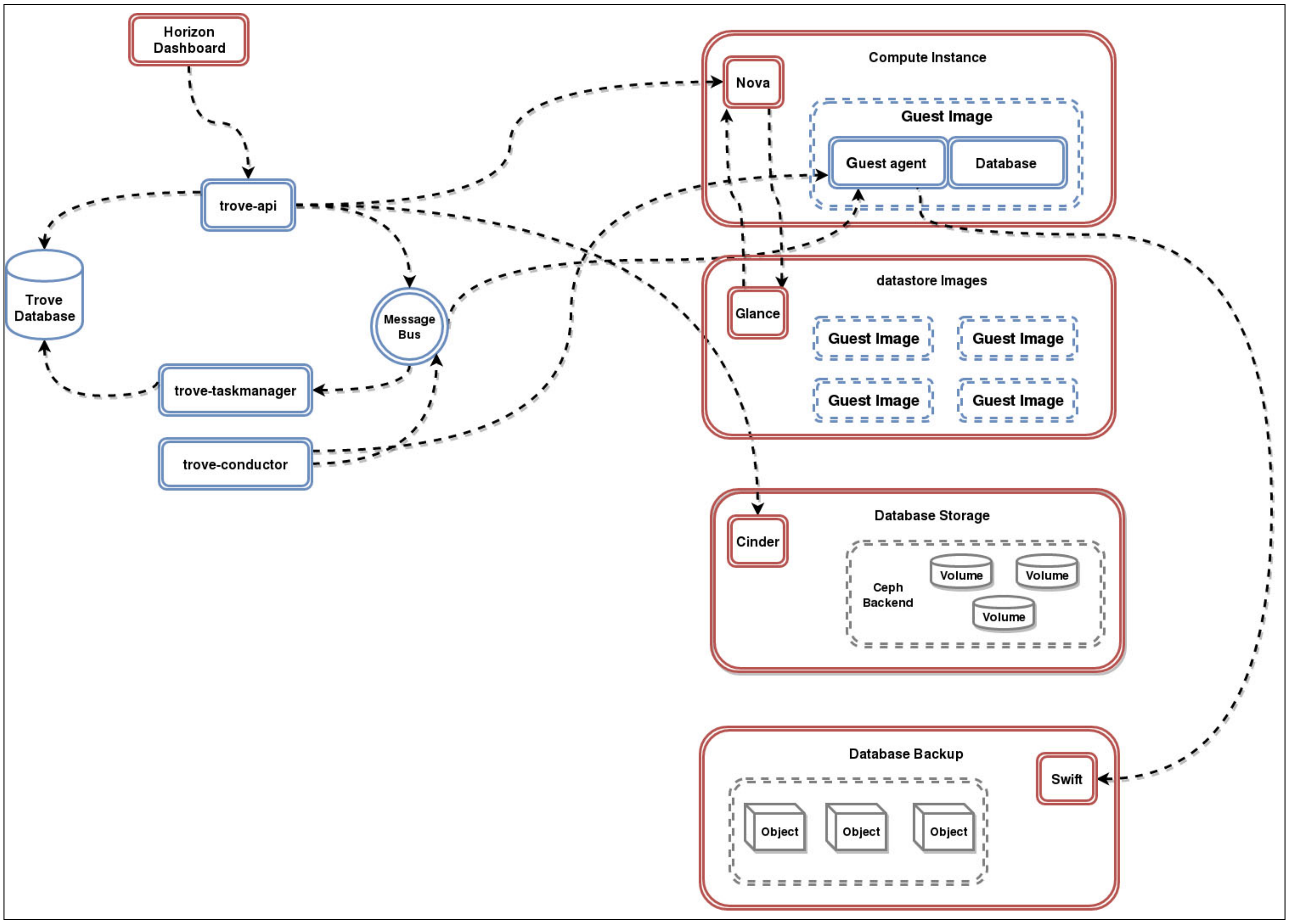


Figure 2-10 High-level Trove workflow

2.3 Supported databases

Mobile applications, Web 2.0, social networks, Internet of Things, and big data generate large amounts of new data that companies want to absorb and use to stay competitive. These various data sources present as two types of data: structured and unstructured.

Much of the new data is unstructured, which means that it cannot be stored in traditional tabular database schemas.

Relational and non-relational databases

Traditional relational database management systems (RDBMS), such as MySQL, MariaDB, and PostgreSQL, have formal schema defining tables and fields, and unique numeric values, which are known as primary keys, that enable related tables. These databases also contain indexes that enable the sorting of information and making certain queries. Thus, these relational databases, generally known as SQL databases, handle data in a structured way.

The non-relational or not-only-SQL databases such as MongoDB or Redis have a flexible schema because they handle data that does not have some kind of order that enables them to do a categorization like SQL databases do. Also, they have a distributed nature and are easily scalable.

Unstructured data can be emails, images, sound files, chats, tweets, PDFs, and so on. The main non-relational database types are detailed in the following sections.

Document-based

Designed for semi-structured data, documents use JSON-like field-value pairs, and are equivalent to the *object concept* in object-oriented programming.

Documents are organized through collections. For example, a document can be in a single collection or in several collections, depending on the implementation. Also, documents can be organized in a tree-like structure.

Column-based

This database type stores data in columns, and each storage block contains data from only one column.

By storing data in columns, the database can more accurately access the data that it needs to respond to a particular query instead of scanning and discarding unwanted data in rows, which increases query performance, particularly in sets of large data.

Key:value

This database type uses a hash table of keys or value pairs that are known as dictionaries. These dictionaries contain a collection of objects, which are formed by multiple fields; each field contains data.

Objects are stored and retrieved by using a key that uniquely identifies them, and is used to quickly find data within the database.

Graph-based

This database type is based on using graph structures to perform semantic queries by using edges and nodes that enable the storing and representation of data.

The relationships that are established in the graph enable the data in the database to be linked directly to each other, making queries and retrieval of information fast. There is no standard query language for this type of database yet, so each one uses its own language.

Table 2-2 shows the comparison between supported databases.

Table 2-2 Comparison between supported databases

Database	Version	Type	Links of interest
MongoDB	3.4 Community Edition 3.4 Enterprise Edition	Document-based scalable high-performance database that stores data in JSON like structures	IBM Power Systems and MongoDB High Performance MongoDB Applications with IBM POWER8
Redis	3.0 Ubuntu 16.04 Source 3.2 Community Edition	Key:value databases where the *value* holds more complex data structures, such as binary-safe strings, linked lists, sets, sorted sets, hashes, bit arrays (bitmaps), and HyperLog logs	IBM POWER8 Redis Labs
MariaDB	10.1 Community Edition	Relational drop-in replacement for MySQL, which maintains high compatibility through libraries and binary equivalence	MariaDB and IBM POWER8
PostgreSQL	5.7 Ubuntu 16.04 Source 9.6 Community Edition	Object-oriented relational database	PostgresSQL and IBM POWER8
MySQL	5.7 Ubuntu 16.04 Source	Relational	None available

3

Architecture

This chapter describes the architecture of Open Platform for Database as a Service (DBaaS) on IBM Power Systems, showing the hardware components that are involved in this solution and describing its roles (including Power Systems nodes and network switches). This chapter also explains the information that you need to prepare when receiving the Open Platform for DBaaS appliance so that IBM can help you integrate it into your existing environment, including network infrastructure and (optionally) Lightweight Directory Access Protocol (LDAP) integration.

This chapter contains the following sections:

- Planning for the Open Platform for DBaaS on Power Systems solution
- Hardware and software requirements
- Infrastructure sizing
- Networking
- Solution components and roles

3.1 Planning for the Open Platform for DBaaS on Power Systems solution

The deployment, installation, and configuration of the Open Platform for DBaaS on Power Systems solution is performed by an IBM Lab Services team, as described in 1.3, “Open Platform for DBaaS on Power Systems product delivery and support flow” on page 20. When you receive the appliance, customizations must be performed to integrate it with your existing infrastructure. The next sections explain the configuration process and the information you must provide to IBM to succeed in this step. The following sections also show customizations that can be performed to integrate the Open Platform for DBaaS on Power Systems solution to your LDAP server for user authentication and projects authorization process, and customizations to the database images that can be deployed in the Open Platform for DBaaS on Power Systems solution.

3.1.1 Physical integration with the customer’s infrastructure

The Open Platform for DBaaS on Power Systems solution is formed by Power Systems server nodes, network switches, and racks. The number of nodes that are involved in this solution depends on the size of the Open Platform for DBaaS on Power Systems solution that you choose for your environment. For more information about the number of nodes and components that are involved in each of the available sizes, see 3.3, “Infrastructure sizing” on page 61.

In the Open Platform for DBaaS on Power Systems solution, the Power Systems nodes develop different roles, including controller nodes (where the OpenStack application runs), compute nodes (where the virtual machines (VMs) and the database instances run), block storage nodes (that provide block disks to the VMs) and object storage nodes (that enable backups of the database instances). These nodes are all interconnected by data switches (where the data of the application traffic to the database instance) and management switches.

Figure 3-1 shows the high-level components of the Open Platform for DBaaS on Power Systems solution, with its components assembled in a rack.

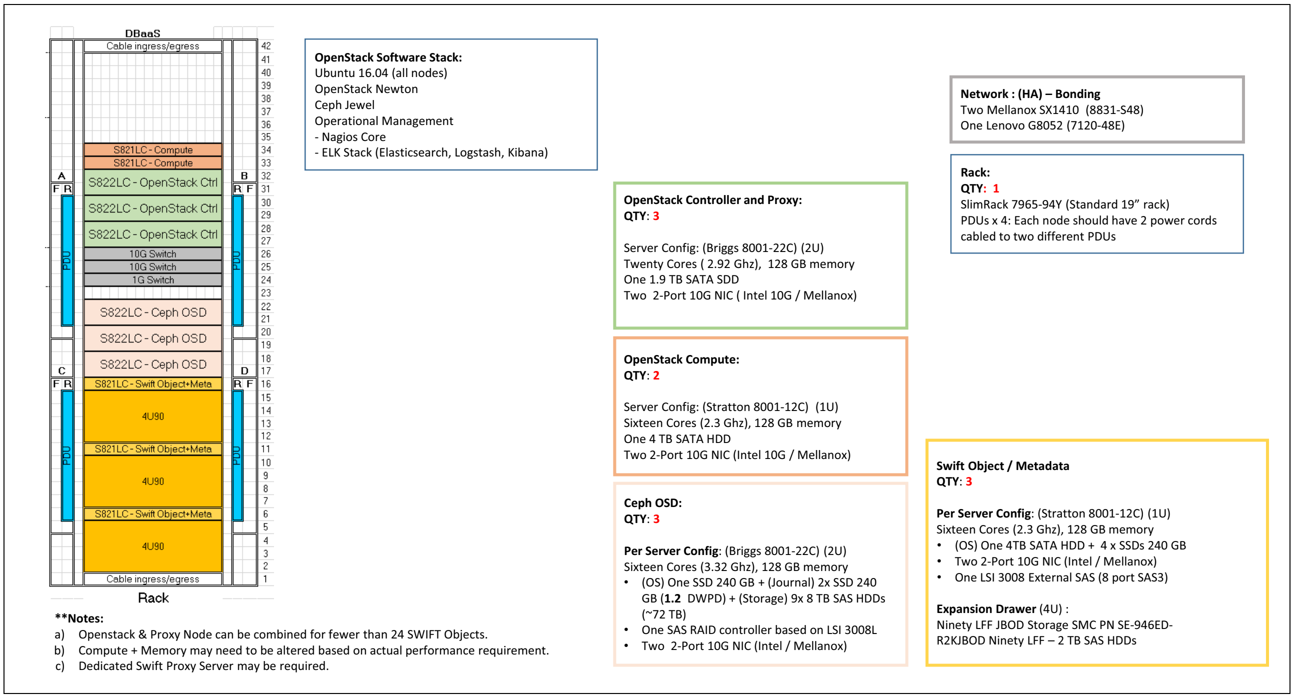


Figure 3-1 High-level component architecture diagram: DBaaS

Each node has several network ports, which are used for the following purposes:

- Management / IPMI: The purpose of this network interface is to communicate with the Power Systems nodes by using the Intelligent Platform Management Interface (IPMI) to manage and monitor the nodes, and also to install the bare metal operating system (OS) during the deployment phase of the solution. In the OpenPOWER LC servers, this network port is an interface to the baseboard management controller (BMC). This interface enables the Metal as a Service (MAAS) to perform management operations such as turn on, turn off, and restart a node when needed.
- Management / Preboot Execution Environment (PXE): This interface is used for the PXE, which enables the nodes to be started remotely by the MAAS to load a kernel and install the OS on the servers.
- Data: This is the data network, which the applications use to communicate with the database engines that are deployed within the Open Platform for DBaaS on Power Systems solution.

Figure 3-1 on page 45 shows network switches that are dedicated for management purposes and other switches that are dedicated for data purposes. The management switches are used internally by the cluster, and the data switches must be connected to your network infrastructure through uplink ports, enabling the databases to reach the remaining components of your environment. Figure 3-2 shows a high-level network architecture diagram that is used in the Open Platform for DBaaS on Power Systems solution.

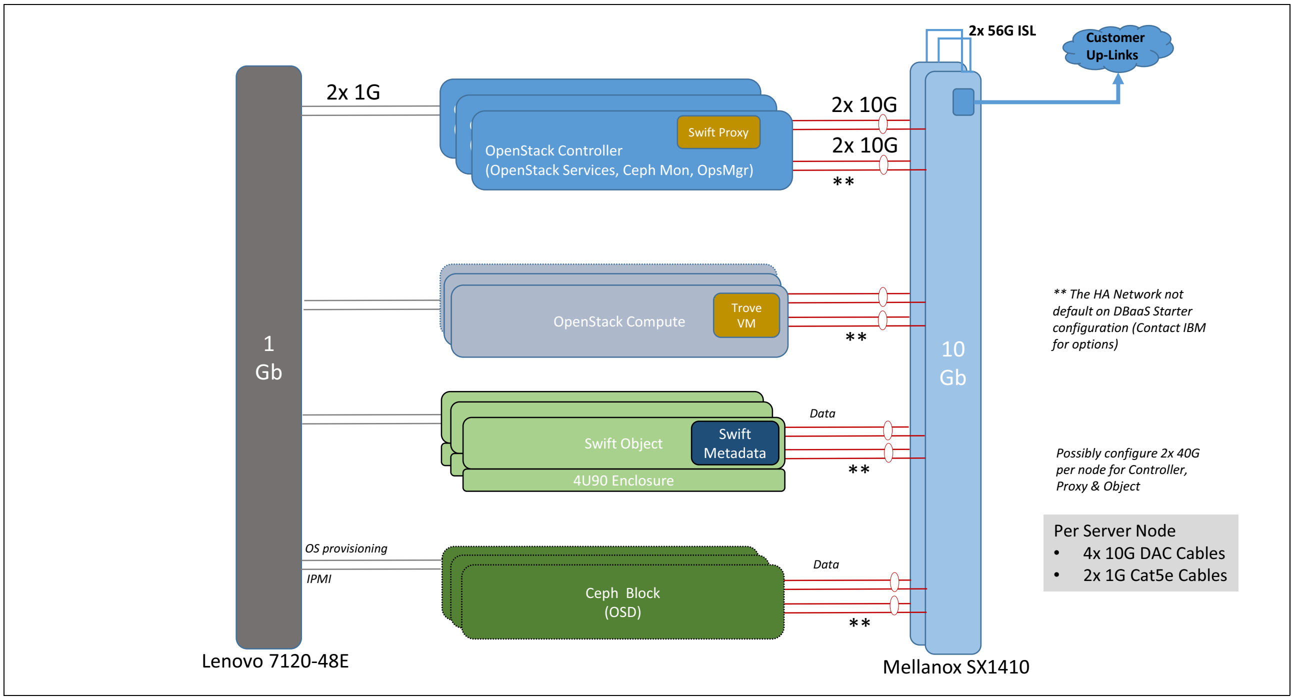


Figure 3-2 High-level network architecture diagram

As the development of the Open Platform for DBaaS on Power Systems solution continues, an updated version of its design is maintained at GitHub.

Figure 3-3 shows the network topology that is used in the Open Platform for DBaaS on Power Systems solution with the VLAN configuration.

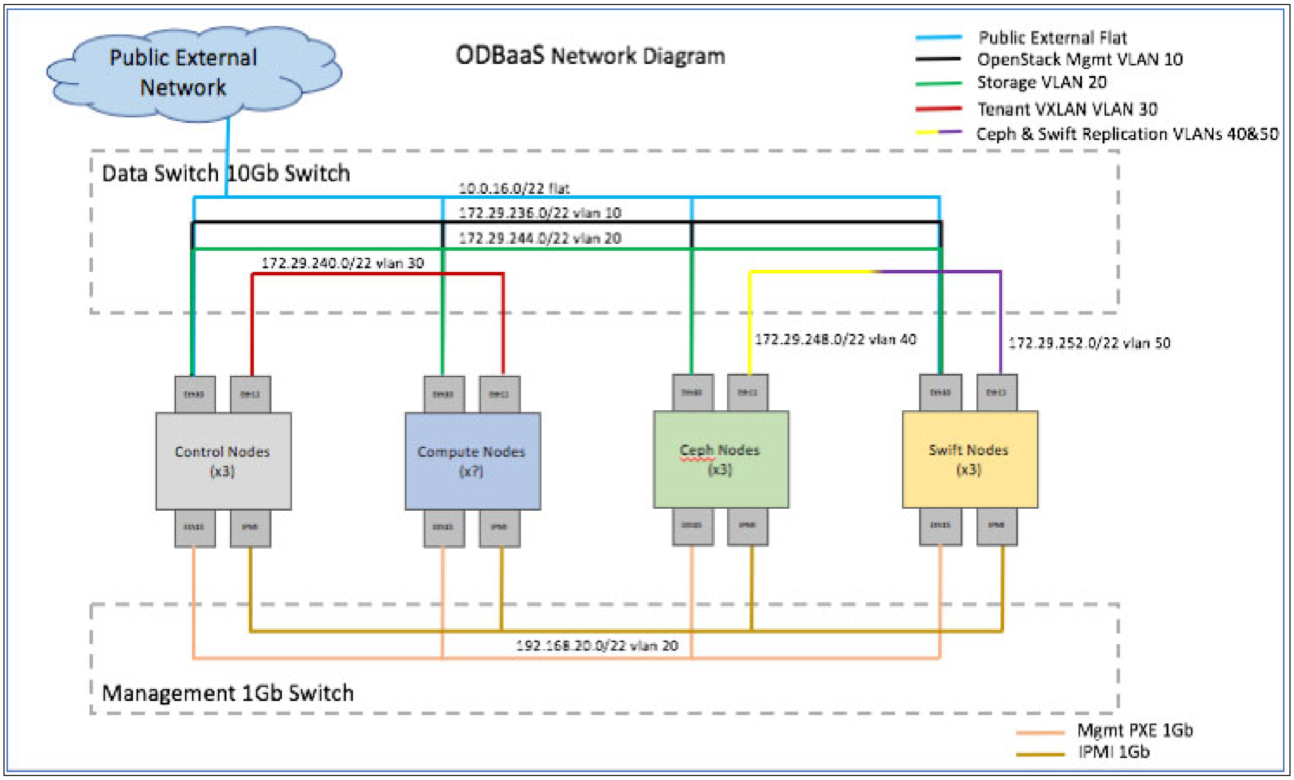


Figure 3-3 Open Platform for DBaaS on Power Systems network diagram

The *Bill of Materials* shows the list of servers and switches, referenced as the bill of materials, which are currently supported in the Open Platform for DBaaS on Power Systems solution

These links are provided as a reference for the architecture of the Open Platform for DBaaS on Power Systems solution, given that all these details are taken care by the IBM Sales Representative when you acquire the solution.

Information that is required from the existing infrastructure

When the Open Platform for DBaaS on IBM Power Systems is acquired, IBM assembles, installs, and configures all the involved components in its facility before shipping it to the customer. These steps are performed by using MAAS and Juju.

During the delivery, implementation and integration of the Open Platform for DBaaS on Power Systems solution to your environment, IBM Lab Services is also engaged to perform the required configuration. There are several environment-specific pieces of information that must be provided to properly configure the appliance, such as information that is detailed in Table 3-1 on page 48, Table 3-2 on page 48, Table 3-3 on page 49, Table 3-4 on page 49 and Table 3-5 on page 50.

Table 3-1 Logistics, floor space, and power requirements

Information	Description	Example
Primary point of contact (PoC)	Primary PoC at client location to be available while onsite.	Fabio Martins (fabio@client.com)
Data center location	Address of data center for receiving shipment.	11501 Burnet Rd., Austin, TX 78758
Rack floor location	Identify floor space for racks.	Building 100, Room 10, K1-AA2
Review power requirements	Provide power requirements information and obtain date of readiness.	Four Power Distribution Units (PDUs) ordered with L60A single phase connector, 25,000 W max total consumption, whips ready to connect by 6/1/2017
Node and switch names	Provide the first draft of servers and switch names in racks and get feedback and changes documented. Each node in the cluster needs a host name to be associated with the management network IP. The prefix can be modified and node numbering is applied by MAAS.	See rack layout tab: compute-1, control-1, ceph-1, switft-1, mgmt-eth-sw1, and data-eth-sw1

Table 3-2 General networking information

Information	Description	Example
Domain name	Obtain client domain name for system.	ibm.com.
Upstream DNS servers	Although a DNS server is configured within the cluster, upstream DNS servers must be defined for names that cannot otherwise be resolved.	4.4.4.4 and 8.8.8.8 as default public upstream DNS servers
NTP	NTP Time Server host name or IP address.	10.0.16.99.
External Internet access	How will each node get to the internet? Is there a NAT on the data or management networks?	A router on 10G data network at 10.0.0.1 provides NAT.
Switch configuration mode	The deployment tool uses passive mode when the customer is doing all switch configurations. Active (the default) mode is when the solution includes the switches and the deployment tool is configuring the switches automatically.	Active.

Table 3-3 Management network switch information (1 Gbps network switch)

Information	Description	Example
Management switch external IP address	1G management switch IP address that is manually configured on default VLAN 1 for deployer access before building other VLANs.	192.168.1.20
Management switch internal VLAN and IP address range	Private management network VLAN and IP range for the deployer and network switches on internal 1G network.	VLAN 16 - 192.168.16.0/24 (254 addresses)
Management switch internal IP address	IP address of the management switch. Labeled ipaddr-mgmt-switch in `config.yml` in the example.	192.168.16.20
Management client network VLAN and IP address range	Private management network VLAN and IP range for the cluster nodes on internal 1G network. The deployment tool auto assigns from this range to each nodes PXE interface (eth15).	VLAN 20 - 192.168.20.0/24 (254 addresses)

Table 3-4 Data network switch information (10 Gbps network)

Information	Description	Example
Data switch IP address	Management IP address of the data switch in the rack. Labeled ipaddr-data-switch in `config.yml` in the example.	192.168.16.25.
Network switch uplinks	Which ports and VLANS are used for uplinks on the management and data network switches.	Ports xx link agg on each 10G switch to customer switch xx ports xx with port VLAN ID xx, Port xx on management switch uplink to customer port xx on customer switch xx.
Network node bonding	Will nodes have single eth10 or eth11 interfaces with no bonding, or will there be two 10G ports bonded for osbond0 (eth10 and 11) and osbond1 (eth12 and 13)?	Single eth ports for eth10 and eth11 10G interfaces.
Data network VLAN and IP address range	Private data network VLAN and IP range for the cluster nodes on the 10G network. The deployment tool auto assigns an IP from the range to each node's data interface (individual or bonded).	VLAN 20 - 10.0.16.0/22 (1022 addresses).
OS management network	OpenStack private management network VLAN and IP range.	VLAN 10 - 172.29.236.0/22 (1022 addresses).

Information	Description	Example
Open Platform for DBaaS on Power Systems client management reserved IPs	Range of IP addresses within the OS infrastructure management network for deployed databases. This is a subset within the OS management network range.	172.29.236.100 - 150.
OS storage network	OpenStack private storage network VLAN and IP range.	VLAN 20 - 172.29.244.0/22 (1022 addresses).
OS tenant VXLAN network	OpenStack private tenant VXLAN network VLAN and IP range.	VLAN 30 - 172.29.240.0/22 (1022 addresses).
Open Platform for DBaaS client VXLANS	How many tenant VXLANs are needed if data network separation is necessary for the deployed databases?	Three VXLANS, 1 for development, 1 for test, and 1 for quality assurance (QA).
OS VXLAN range	Range that OpenStack can use when creating client VXLANS (1 - 1000).	300 - 500.
External connections into databases	Will VXLANs need to be accessible from outside networks (for example, floating IPs)?	Development VXLAN needs access from all external networks (for example, developer notebooks). Plan for 30 floating IPs on the data network.
OS VLAN range	Range that OpenStack can use when creating client VLANS (1 - 4094).	150 - 200.
Controller (horizon) external and internal floating IPs	An internal (OS management) and external (also for Horizon GUI) floating IP is assigned to the master controller for high availability.	10.0.16.50 and 172.29.236.50.

Table 3-5 Databases and use cases

Information	Description	Example
Certificates	Are signed certificates provided or are self-signed certificates required?	None available.
Types of databases	What open source databases, including versions, are required.	MongoDB x.x.x or Redis x.x.x.
Database sizes	What templates are needed (small, medium, or large), and what CPU, memory, and storage is required for each database?	► Small: One CPU, 4 GB memory, and 10 GB disk ► Medium: Two CPUs, 8 GB memory, and 30 GB disk ► Large: Four CPUs, 16 GB memory, and 50 GB disk

Information	Description	Example
Use cases	What use cases are needed?	Build image, deploy, resize, back up, destroy, restore, and user management.
Projects and user IDs	Are there any specific projects or users that must be created?	► ProjectA with users testa1 and testa2 ► ProjectB with users testb1 and testb2
Deployer node	(Optional) Will there be a requirement to have the deployer node at the client location?	The client provides an x86 server with Ubuntu 16.04.01 that has Internet access and can connect to the management switch and optionally the data switch.
Default node login data	Node IDs and passwords.	► BMC/IPMI network (Admin/Admin) ► OS management network (ubuntu/passw0rd)
Timeline	Set the target timeline to achieve the start of PoC.	► June 1 - 14: Assembly in Rochester pre-build lab ► June 14 - 21: Tear-down and ship to client location ► June 21 - 25: Assembly, reinstall, and load images in client location ► June 26: PoC start

All the information that you provide to IBM is added to a configuration file that is used by MAAS and Juju to deploy the solution, and to customize it in your environment. An example of this configuration file is available at GitHub.

Deployment node consideration (MAAS and Juju server node)

During the deployment of the Open Platform for DBaaS on Power Systems solution (still at an IBM location), a server is used as the deployment node, where the MAAS and Juju server is installed. This node is used to implement or deploy the whole environment, including activities such as installation of the OS on the bare metal servers, configuration of the network switches, installation and configuration of the OpenStack components, installation and configuration of the Ceph software-defined storage components, and installation and configuration of the operational management tools.

The deployment node is a separate server, not included in the Open Platform for DBaaS on Power Systems solution. It can be a POWER8 server or an x86 server. The node that is used inside IBM to deploy the solution is not shipped with it, so you must have an additional server at the customer's location to use as the MAAS and Juju server for adding nodes (scaling the solution in the future) and maintaining software levels (upgrade Ubuntu, OpenStack, Ceph, and so on). Here are the minimum requirements for the deployment node:

- Two cores and 32 GB RAM
- One network interface connection of 1 Gbps (connect to the management network)
- Ubuntu 16.04 LTS

If you already have a MAAS and Juju server, you can use it; otherwise, you can acquire a POWER8 server for this purpose.

Racking considerations

The Open Platform for DBaaS on Power Systems solution is composed of different Power server models and types, which also perform distinct roles. The details of the components and their roles are described in 3.5, "Solution components and roles" on page 70. In this section, you get an overview and a description of the nodes and its roles, as a quick reference for the racking considerations:

- Compute nodes: The nodes where the VMs (and the database instances) run. These are Power System S812LC (8001-12C) servers.
- Controller nodes: The servers where the OpenStack components run. These are Power System S822LC (8001-22C) servers.
- Ceph nodes: The server where the Ceph components run and provide block storage to the VMs. These are also Power System S822LC (8001-22C) servers.
- Swift nodes: The servers where the Swift components of the OpenStack run. These servers are used for backup purposes. They are Power System S812LC (8001-12C) servers. IBM 5U84 JBOD disk drawers are shipped with the Swift nodes. These drawers are connected to the Swift nodes by using SAS connections to expand their storage capacity.

When the components of the Open Platform for DBaaS on Power Systems solution arrive at your location, they can be racked and cabled. There is a suggested layout for racking the nodes, which varies according to the size of the Open Platform for DBaaS on Power Systems solution that was chosen. The racking description in this section considers the *Cloud Scale* size of the solution (for more information about available sizes, see 3.3, "Infrastructure sizing" on page 61), which has all the possible components. If your Open Platform for DBaaS on Power Systems solution has fewer components, they are placed in the same locations in the rack, leaving empty spaces for the missing components (compared to the Cloud Scale size), which will be available for future expansion of the cluster. The racking suggestion is as follows:

- Data network switches: The data switches are placed in positions U25 and U26 of the rack.
- Management network switches: The management switches are placed in positions U23 and U24 of the rack.
- Swift nodes and storage drawers: The three Swift nodes and the three storage drawers use 15 Us of the rack. They are positioned at the bottom of the rack, using positions U2 - U16.
- Ceph nodes: The Ceph nodes use 2U each, and they are positioned on top of the Swift nodes, using U17 - U22.
- Controller nodes: The controller nodes use 2U each, and they are placed on top of the network switches, using U27 - U32.
- Compute nodes: All the compute nodes are placed on top of the controller nodes. They are 1U node and use the remaining space according to the number of compute nodes that your solution has (varies according to the size that is chosen).
- PDUs: The PDUs are usually placed vertically at specific locations of the rack.

Figure 3-4 shows the suggested layout for racking a Cloud Scale size Open Platform for DBaaS on Power Systems solution.

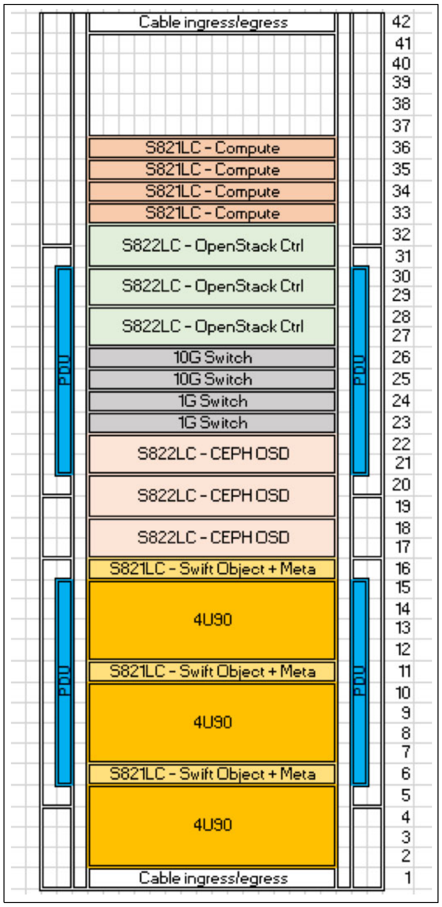


Figure 3-4 Cloud Scale size racking suggestion

Cabling considerations

The Open Platform for DBaaS on Power Systems cabling considerations cover the following components that requires cabling:

- Storage cabling
- Network cabling
- Power supply cabling

Storage cabling

The only components that require storage cabling are the Swift nodes because they are connected to a JBOD storage drawer to expand their storage capacity. Figure 3-5 shows the rear view of the Swift nodes and where the storage connections are. Figure 3-6 shows the rear view of the storage controller and where the SAS connection goes from the Swift nodes.

Figure 3-5 Rear view of the Swift node: Storage connections

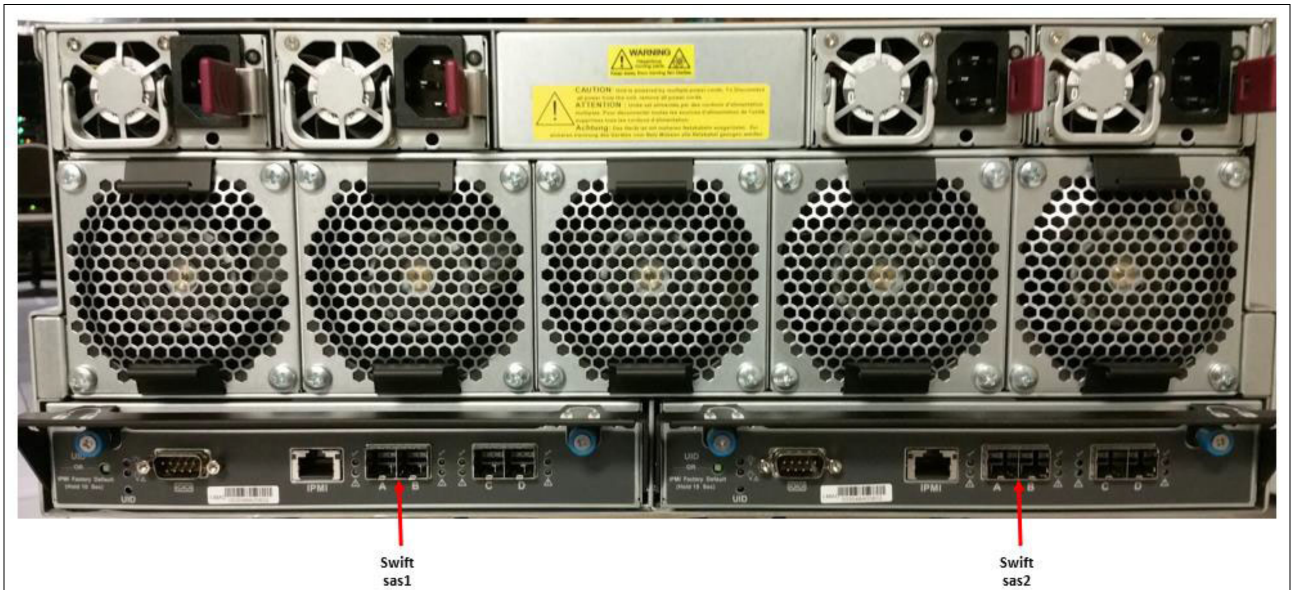


Figure 3-6 Rear view of the storage controller: Storage connections

Network cabling

For the network cabling, there are two types of networks that must be considered: management (1 Gbps network) and data (10 Gbps network). The management network is used for IPMI and PXE purposes, so the adapters that are labeled *Mgmt IPMI* and *Mgmt PXE* can go to the management network switches, and the adapters that are labeled *data* can go to the data network switches. There are two different Power Systems server models in this solution: S822LC (8001-22C) and S821LC (8001-12C), and both must be considered. Figure 3-7 shows the S821LC server network cabling layout. Figure 3-8 shows the S822LC server network cabling layout.

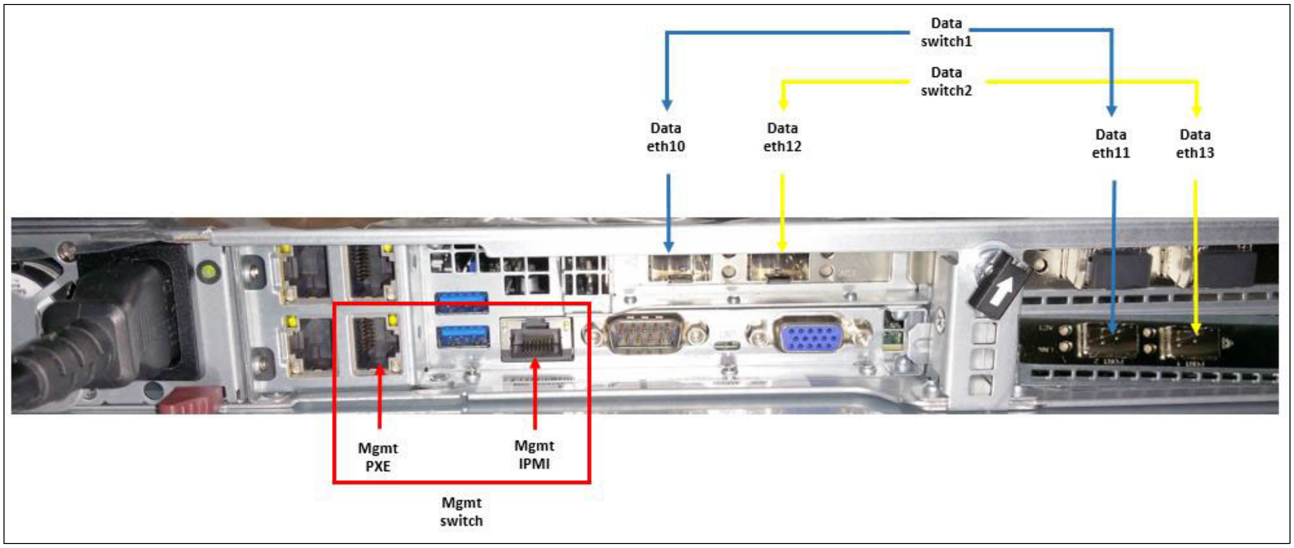


Figure 3-7 S821LC (8001-12C) network cabling layout

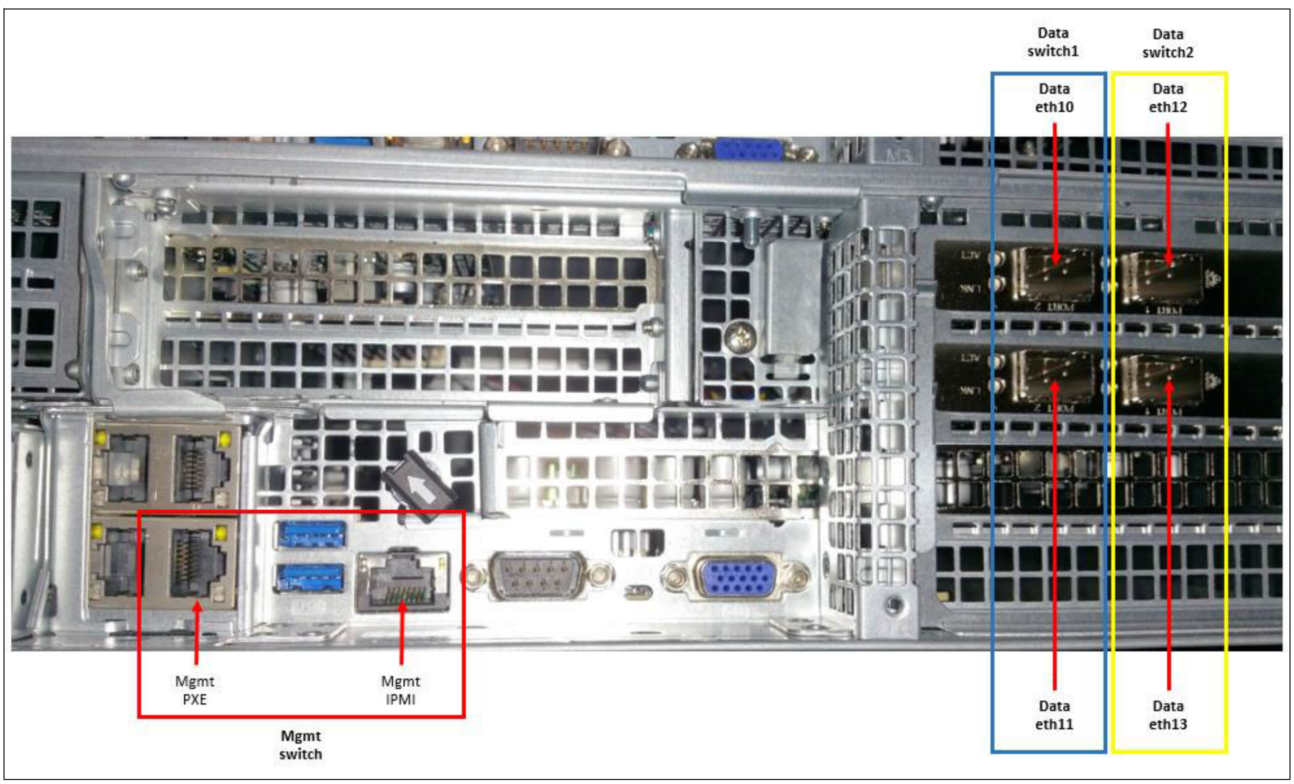


Figure 3-8 S822LC (8001-22C) network cabling layout

These cabling rules are valid for all types of nodes (compute, controller, Ceph, or Swift). Figure 3-9 shows the network cabling layout from the management switch perspective. The prefix Ctrl-X represents cabling coming from the controller nodes. The prefix Comp-X represents cabling coming from the compute nodes. The prefix Ceph-X represents cabling coming from the Ceph nodes. The prefix Swift-X represents cabling coming from the Swift nodes. The deployer node (where MAAS and Juju server run) also needs a network connection to the management switch.

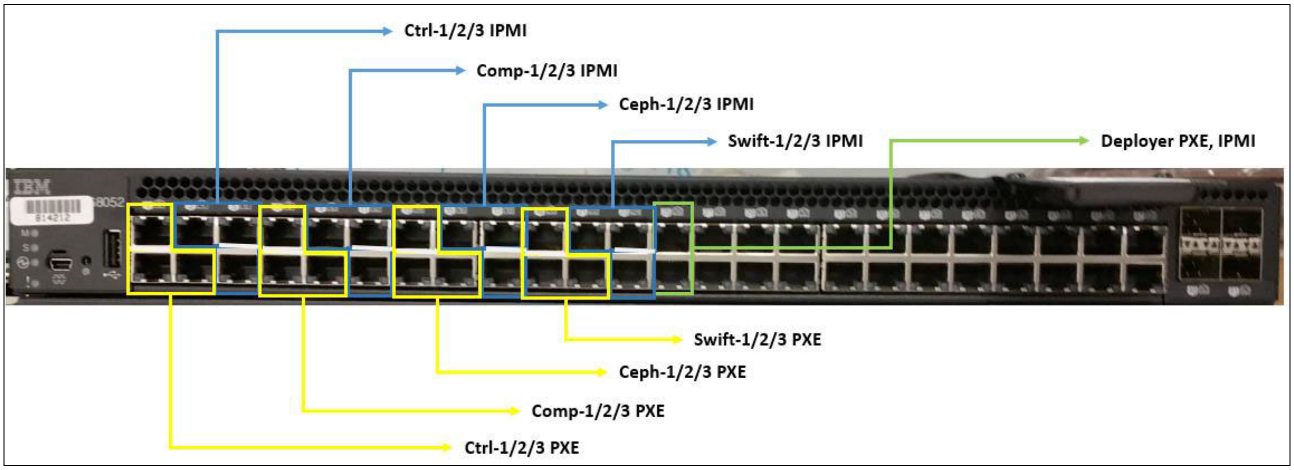


Figure 3-9 Management network switch cabling considerations

The same prefixes apply to the data network switches that are shown in Figure 3-10.

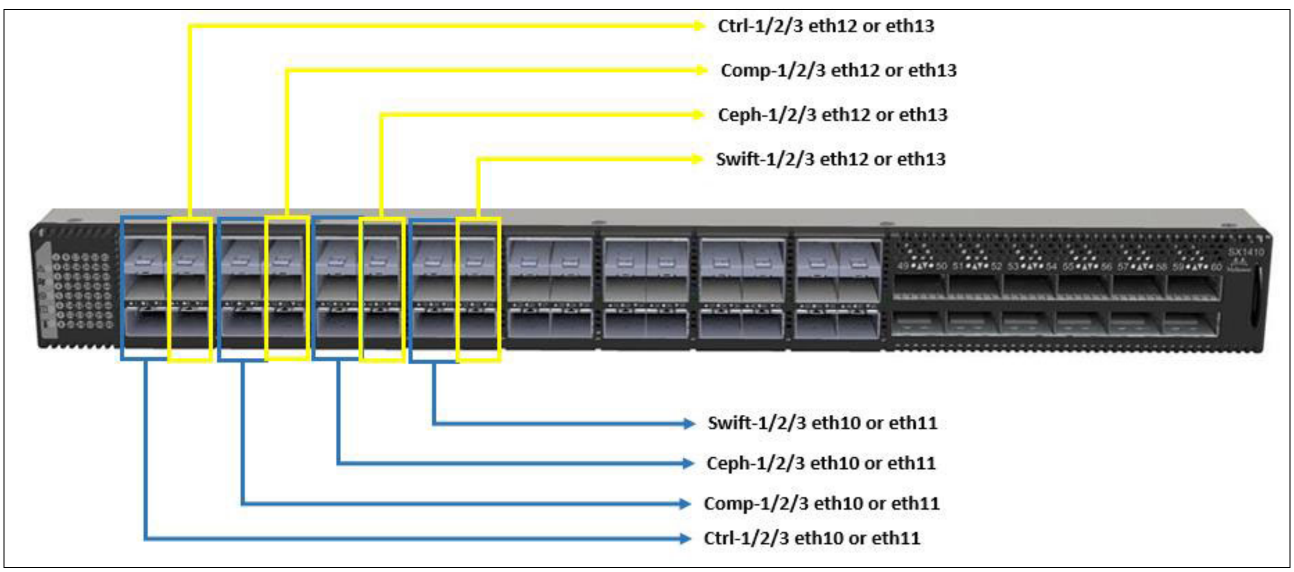


Figure 3-10 Data network switch cabling considerations

Power supply cabling

The racks that are used for the Open Platform for DBaaS on Power Systems solution usually have four PDUs for power connections to the cluster nodes and components. It is preferable to have redundant power lines to feed these PDUs, and connect two PDUs per power line. With such a layout, the servers with redundant power supplies can be connected to PDUs going to different power lines (for example, PDU1 goes to power line 1 and PDU3 goes to power line 2, so in a server with two power supplies, one can go to PDU1 and the other can go to PDU3). Both the S821LC and S822LC Power System servers have redundant power supplies, so consider connecting each power supply to PDUs going to different power lines for redundancy purposes.

If you cannot connect both power supplies, or if a component with a single power supply is inserted in the cluster, you can consider distributing the connection of nodes from the same role between different PDUs and power lines. For example, if you have three Ceph nodes, do not connect all to the same PDU and power line, but two servers in a power line (through two different PDUs) and the third server to a different PDU or power line so that you have some level of redundancy. If there is a PDU or power line failure, at least one Ceph node still is available to attend the cluster workload.

Cluster and OpenStack deployment

All the information that is provided to IBM during the initial phase of the implementation of the Open Platform for DBaaS on Power Systems solution (see the example of information that you must provide in “Information that is required from the existing infrastructure” on page 47) is used during the deployment phase of the cluster (which installs and configures all the components, including OS on the nodes and the OpenStack software). Such information is added to a configuration file in the YAML format (YAML is a data serialization standard that is used by many applications), which is provided to MAAS for the deployment of the environment.

MAAS automates the deployment by installing and configuring components (Power System servers and network switches). A sample of this configuration file that is prepared for the Open Platform for DBaaS on Power Systems solution is available at GitHub.

After the work is run by MAAS, the Juju server is also used to install and configure the OpenStack components on the nodes. Additional parameters can be required by Juju during this configuration step, as shown in Table 3-6.

Table 3-6 Additional parameters that are required for OpenStack deployment

Parameter	Description	Example
Keystone password	The password that is used for the OpenStack authentication services.	passw0rd
Virtual Router Redundancy Protocol (VRRP) ID	Virtual Router ID that is in the range 1 - 255 and must be unique across the network. This ID is scoped to a network, considering it is based on a broadcast protocol, so the network administrator must make sure that there is no other cluster or network services (such as routers) that are using this VRRP ID.	202

Juju deploys the OpenStack components by using containers. The container technology that is used for the OpenStack components is *LXD*, which is a lightweight container hypervisor that is based on *LXC*, which is an application that creates and maintains containers on a local node, and provides an API for its management. For more information about LXC and LXD, see the LXD documentation at Ubuntu.

The deployment is performed at an IBM location before shipping the equipment, and the integration is done at the customer's site by an IBM Lab Services Specialist.

3.1.2 Security integration (LDAP and Keystone)

As an optional step, you can integrate your Open Platform for DBaaS on Power Systems solution with an existing LDAP or an Active Directory server for authentication and also authorization (determines what a user can do in the Open Platform for DBaaS on Power Systems solution after authentication).

The OpenStack component that handles authentication services is *Keystone*, and it supports integration with LDAP, including multiple domains (for example, multiple tenants, or customers, with different LDAP servers authenticating and using the same Open Platform for DBaaS on Power Systems solution). *Configure OpenStack Keystone support for domain-specific corporate directories* describes the authentication process and support for Keystone and LDAP.

The OpenStack Keystone configuration for integration with LDAP is explained in Integrate Identity with LDAP.

And, in case you want to enable multiple domains, there are additional steps that must be performed, as described in Domain-specific configuration.

Note: These documents describe the configuration that is performed after the deployment of the OpenStack components. If you want to integrate the Open Platform for DBaaS on Power Systems solution with your LDAP server, pass this requirement along to the IBM Lab Services representative during the planning phase of the solution so that the LDAP configuration is included in the configuration files that are used for the deployment (YAML). Implementing LDAP (or Active Directory) backends contains an example of the LDAP information being configured in the files that are used during the deployment of the Open Platform for DBaaS on Power Systems solution.

If you are planning to integrate the Open Platform for DBaaS on Power Systems solution with an existing LDAP server, work with IBM during the planning and implementation phase and engage your LDAP administrator to provide the information, which is needed for Keystone to authenticate by using your LDAP, as shown in Table 3-7.

Table 3-7 Information that is required for Keystone integration with LDAP

Information	Description	Example
LDAP server IP address	The IP address of the LDAP server	`9.3.18.57`
LDAP admin account	The administrator account for the LDAP server	`cn=admin,dc=ibm,dc=com`
LDAP password	The password for the admin account provided	`Passw0rd#`
User search tree	The user search base for users in this domain	`user_tree_dn: "ou=Users,o=MyCorporation"`

Information	Description	Example
Group search tree	The group search base for groups in this domain	`group_tree_dn: "cn=openstack-users,ou=Users,o=MyCorporation"`
General search tree	The general search base for this domain	`ou=Openstack,dc=ibm,dc=com`

3.1.3 Custom database images

Several open database images are available as soon as the Open Platform for DBaaS on Power Systems deployment is complete, so you can deploy and start using them immediately. If you want to customize the images, for example, by including some software that your company uses to track the VMs in a standardization process, you can use the Open Platform for DBaaS on Power Systems solution to build your own images, customizing them and importing them into the Glance repository so that you can deploy your customized VMs later.

Section 2.3, “Supported databases” on page 39 has a list and description of the databases that are supported in the Open Platform for DBaaS on Power Systems solution, and the ones that are available as soon as the solution is deployed. In “Image building” on page 212, you can find the steps for customizing and building your own image.

3.2 Hardware and software requirements

The hardware that is required for running the Open Platform for DBaaS on Power Systems solution depends on the size of the appliance you choose. A list of the model, type, and the quantity of servers, and network switches that are required for each of the available sizes is in *Bill of Materials*.

Section 3.3, “Infrastructure sizing” on page 61 provides details about the available sizes and components of the Open Platform for DBaaS on Power Systems solution. Sections 3.4, “Networking” on page 68 and 3.5, “Solution components and roles” on page 70 provide detailed information about each physical component.

In terms of software requirements, all the software components are installed and configured in the nodes by MAAS and Juju during the deployment phase. Table 3-8 shows the software components that are used in the Open Platform for DBaaS on Power Systems environment.

Table 3-8 Software components used in the Open Platform for DBaaS on Power Systems

Main Component	Object storage	DBaaS
OS and hardware infrastructure	Ubuntu 16.04 LTS GCC 4:53 Python 2.7 or 3 Ruby 1:2.3 Java7 2.6.8-1 LXC 2.0.7 Apache2 2.4.18 Pciutils 1:3.3.1 Sendmail 8.15.2-3 Sysstat 11.2.0	Ubuntu 16.04 LTS GCC 4:53 Python 2.7 or 3 Ruby 1:2.3 Java7 2.6.8-1 LXC 2.0.7 Apache2 2.4.18 Pciutils 1:3.3.1 Sendmail 8.15.2-3 Sysstat 11.2.0
Basic infrastructure (deployed with OpenStack)	Memcached MySQL / Galera RabbitMQ HAProxy Keepalived	Memcached MySQL / Galera RabbitMQ HAProxy Keepalived
OpenStack (all Ocata)	Keystone Horizon Swift	Keystone Glance Nova API, Scheduler, Compute Neutron + Agents Cinder API Heat Horizon Swift Trove
Ceph (all Jewel)	Ceph Mon Ceph object storage daemon (OSD) Ceph MDS Reliable Autonomic Distributed Object Store (RADOS) Block Device (RBD) / Client	Ceph Mon Ceph OSD Ceph MDS RBD / Client
Operations	Operations management (OpsMgr) Elasticsearch Logstash and FileBeat Kibana + Dashs Nagios, Nagios Remote Plugin Executor (NRPE) Nagios Plugins	OpsMgr Elasticsearch Logstash and FileBeat Kibana + Dashs Nagios, NRPE Nagios Plugins
Databases	Not applicable	Redis MySQL MariaDB MongoDB PostgreSQL

3.3 Infrastructure sizing

The Open Platform for DBaaS on Power Systems solution is a scalable solution and can be resized according to your needs. There are initially four reference configurations (or four different sizes), but these configurations can be increased according to specific needs. Figure 3-11 shows the current reference configurations for the Open Platform for DBaaS on Power Systems solution.

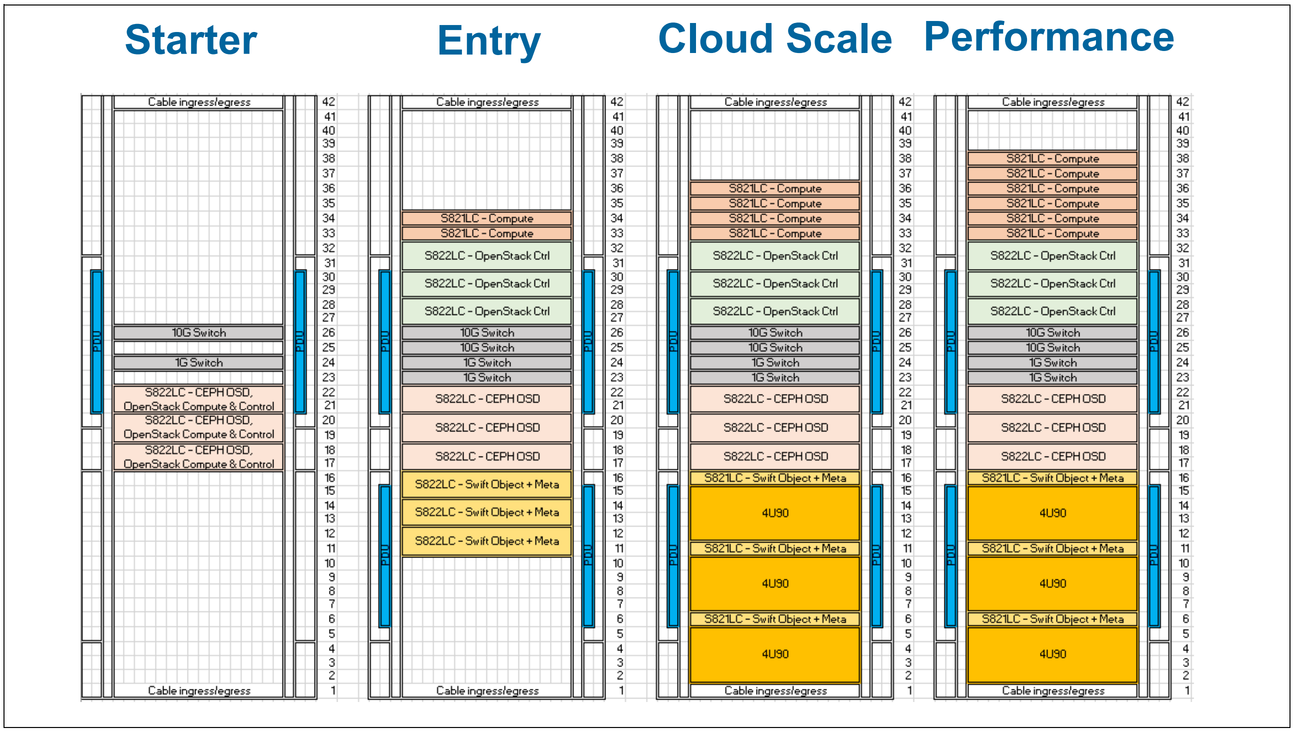


Figure 3-11 Reference configurations for the Open Platform for DBaaS on Power Systems solution

The next sections specify the number of components for each size. All this information is *Bill of Materials*.

3.3.1 Starter

The *starter* configuration is a simple environment that requires only three servers to deploy an Open Platform for DBaaS on Power Systems solution. This configuration enables the Open Platform for DBaaS on Power Systems features so that you can get used to its functions. The same features from all other sizes are available, except for the Swift Object Storage, and can be used for backups (which is not deployed in this scenario).

In the starter reference configuration, the controller, compute node, and software-defined block storage (Ceph) functions are performed by the same three nodes, which have the three functions. Even the starter configuration provides some level of redundancy, given that three nodes work as Ceph (maintaining copies of the data), providing fault tolerance. Figure 3-12 shows the Open Platform for DBaaS on Power Systems starter size components.

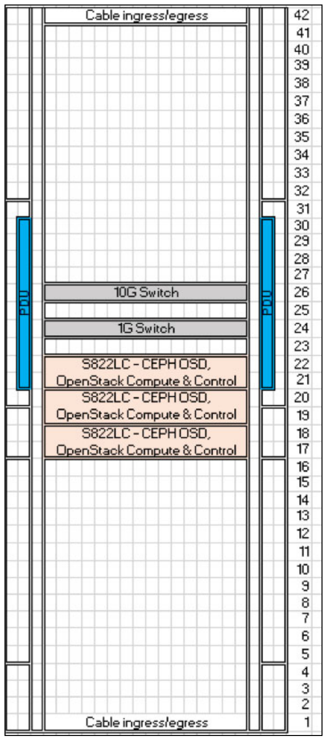


Figure 3-12 Starter Open Platform for DBaaS on Power Systems solution

Table 3-9 shows the hardware components of the starter Open Platform for DBaaS on Power Systems solution.

Table 3-9 Starter Open Platform for DBaaS on Power Systems hardware components

Function	Quantity	Hardware description
Compute, OpenStack controller, and Ceph node	3	2U 8001-12C servers, each with: ► Sixteen cores (2.3 GHz) and 256 GB memory ► (OS) 2 SSDs 240 GB + (Meta) 2 SSDs 240 GB (Journal) (1.2 DWPD) + (Storage) eight 8 TB SAS HDDs (~80 TB) ► One 2-Port 10G NIC (Intel/Mellanox) ► One MegaRAID SAS controller
Management switch	1	Mellanox (8891-S52)
Data switch	1	Mellanox SX1410 (8831-S48)

3.3.2 Entry (small)

The *entry* configuration provides all the functions of the Open Platform for DBaaS on Power Systems solution, including the Swift Object Storage, which is used for backup purposes. The controller, compute, and Ceph functions are performed by different nodes. In this scenario, you have two compute nodes, three controller nodes, three Ceph nodes, three Swift nodes, and redundant data and management network switches, as shown in Figure 3-13.

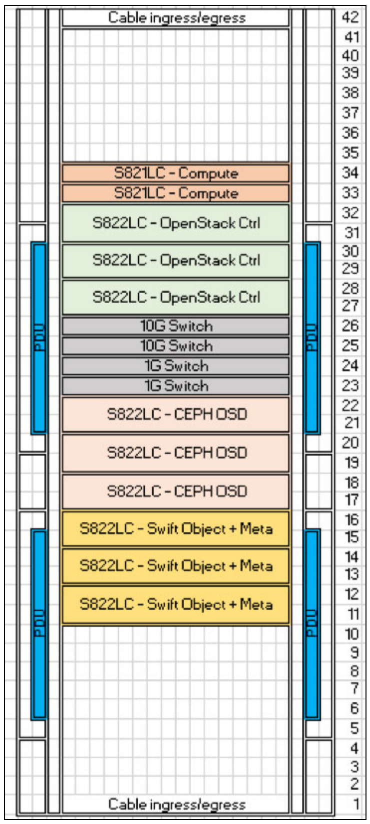


Figure 3-13 Entry Open Platform for DBaaS on Power Systems solution

Table 3-10 shows the hardware components of the entry Open Platform for DBaaS on Power Systems solution.

Table 3-10 Entry Open Platform for DBaaS on Power Systems hardware components

Function	Quantity	Hardware description
Compute node	2	1U 8001-12C servers, each with: ► Sixteen cores (2.3 GHz) and 128 GB memory ► Two 4 TB SATA HDDs ► Two 2-Port 10G NIC (Intel 10G/Mellanox)
OpenStack controller node	3	1U 8001-12C servers, each with: ► Twenty Cores (2.0 GHz) and 256 GB ► Two 4 TB SATA HDDs ► One 2-Port 10G NIC (Intel 10G/Mellanox)

Function	Quantity	Hardware description
Ceph node	3	Ceph software-defined storage (SDS) block storage for DB instances, DB image library, Open Platform for DBaaS infrastructure: ▶ Three-way storage replication for data availability ▶ Redundant control plane for high availability ▶ 192 TB of total storage and 64 TB of replicated storage 2U 8001-12C servers, each with: ▶ Sixteen cores (2.3 GHz) and 256 GB memory ▶ (OS) two SSDs 240 GB + (Meta) two SSDs 240 GB (Journal) (1.2 DWPD) + (Storage) eight 8 TB SAS HDDs (~80 TB) ▶ One 2-Port 10G NIC (Intel/Mellanox) ▶ One MegaRAID SAS controller
Swift Object Storage node	3	Swift SDS object storage ▶ Three-way storage replication for data availability ▶ Redundant control plane for high availability 1U 8001-21C servers, each with: ▶ Sixteen cores (2.3 GHz) and 256 GB Memory ▶ (OS) Two SSDs + (Meta) Two SSDs x 240 GB ▶ One 2-Port 10G NIC (Intel/Mellanox) ▶ One LSI 3008 External SAS ▶ One MegaRAID SAS controller
Data network switch	2	Mellanox SX1410 (8831-S48)
Management network switch	2	Mellanox (8891-S52)

3.3.3 Cloud scale (medium)

The *cloud scale* configuration is an upgrade to the entry configuration. The cloud scale size receives two additional compute nodes and three additional IBM 5U84 JBOD disk drawers, which are connected to the Swift nodes by using SAS connections to expand their storage capacity. Figure 3-14 illustrates the cloud scale configuration size.

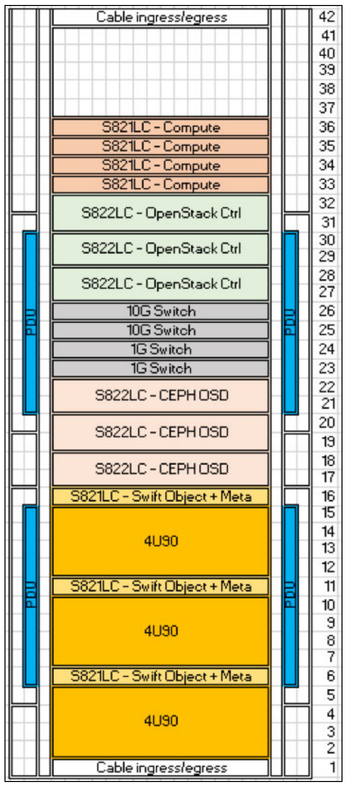


Figure 3-14 Cloud scale Open Platform for DBaaS on Power Systems solution

Table 3-11 shows the hardware components of the cloud scale Open Platform for DBaaS on Power Systems solution.

Table 3-11 Cloud scale Open Platform for DBaaS on Power Systems hardware components

Function	Quantity	Hardware description
Compute node	4	1U 8001-12C servers, each with: ► Sixteen cores (2.3 GHz) and 128 GB memory ► Two 4 TB SATA HDDs ► Two 2-Port 10G NICs (Intel 10G/Mellanox)
OpenStack controller node	3	1U 8001-12C servers, each with: ► Twenty cores (2.0 GHz) and 256 GB ► Two 4 TB SATA HDDs ► One 2-Port 10G NIC (Intel 10G/Mellanox)

Function	Quantity	Hardware description
Ceph node	3	Ceph SDS block storage for DB instances, DB image library, and Open Platform for DBaaS infrastructure: ► Three-way storage replication for data availability ► Redundant control plane for high availability ► 192 TB of total storage and 64 TB of replicated storage 2U 8001-12C servers, each with: ► Sixteen cores (2.3 GHz) and 256 GB memory ► (OS) two SSDs 240 GB + (Meta) two SSDs 240 GB (Journal) (1.2 DWPD) + (Storage) eight 8 TB SAS HDDs (~80 TB) ► One 2-Port 10G NIC (Intel/Mellanox) ► One MegaRAID SAS controller
Swift Object Storage node	3	Swift SDS object storage: ► Three-way storage replication for data availability ► Redundant control plane for high availability 1U 8001-21C servers, each with: ► Sixteen cores (2.3 GHz) and 256 GB Memory ► (OS) two SSDs + (Meta) two SSDs x 240 GB ► One 2-Port 10G NIC (Intel/Mellanox) ► One LSI 3008 External SAS ► One MegaRAID SAS controller Three 4U90 SMC expansion drawers: ► Ninety LFF – 2 TB SAS HDDs, 30 per drawer
Data network switch	2	Mellanox SX1410 (8831-S48)
Management network switch	2	Mellanox (8891-S52)

3.3.4 Performance (large)

The *performance* configuration is an upgrade to the cloud scale configuration. This configuration adds two more compute nodes to the solution, for a total of six compute nodes, together with the other remaining components. Figure 3-15 shows the performance size of the Open Platform for DBaaS on Power Systems solution.

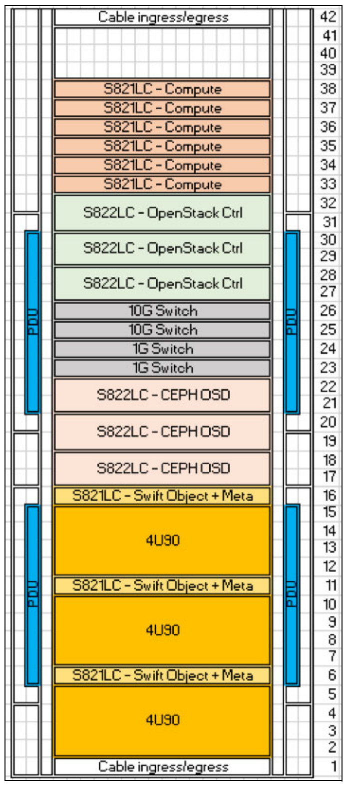


Figure 3-15 Performance size Open Platform for DBaaS on Power Systems solution

Table 3-12 shows the hardware components of the performance size Open Platform for DBaaS on Power Systems solution.

Table 3-12 Performance Open Platform for DBaaS on Power Systems hardware components

Function	Quantity	Hardware Description
Compute node	6	1U 8001-12C servers, each with: ► Sixteen cores (2.3 GHz) and 128 GB memory ► Two 4 TB SATA HDDs ► Two 2-Port 10G NICs (Intel 10G/Mellanox)
OpenStack controller node	3	1U 8001-12C servers, each with: ► Twenty cores (2.0 GHz) and 256 GB memory ► Two 4 TB SATA HDDs ► One 2-Port 10G NIC (Intel 10G/Mellanox)

Function	Quantity	Hardware Description
Ceph node	3	Ceph SDS block storage for DB instances, DB image library, and Open Platform for DBaaS infrastructure: ► Three-way storage replication for data availability ► Redundant control plane for high availability ► 192 TB of total storage and 64 TB of replicated storage 2U 8001-12C servers, each with: ► Sixteen cores (2.3 GHz) and 256 GB memory ► (OS) two SSDs 240 GB + (Meta) two SSDs 240 GB (Journal) (1.2 DWPD) + (Storage) eight 8 TB SAS HDDs (~80 TB) ► One 2-Port 10G NIC (Intel/Mellanox) ► One MegaRAID SAS controller
Swift Object Storage node	3	Swift SDS Object storage ► Three-way storage replication for data availability ► Redundant control plane for high availability 1U 8001-21C servers, each with: ► Sixteen cores (2.3 GHz) and 256 GB memory ► (OS) two SSDs + (Meta) two SSDs 240 GB ► One 2-Port 10G NIC (Intel/Mellanox) ► One LSI 3008 External SAS ► One MegaRAID SAS controller Three 4U90 SMC expansion drawers: ► Ninety LFFs – 2 TB SAS HDDs, 30 per drawer
Data network switch	2	Mellanox SX1410 (8831-S48)
Management network switch	2	Mellanox (8891-S52)

3.4 Networking

The network layer is architected over two networks: the management network and the data network.

The management network is how the JuJu orchestrator accesses and configures each one of solution nodes: compute nodes, control nodes, block storage nodes, and object storage nodes, either to deploy the solution, run administrative tasks, or expand the solution.

The data network is how the nodes communicate with each other and handle the internal processes: instances creation, data storage and retrieval, backups, and so on.

Both the management network and the data network must implement VLANs for the logical segmentation and isolation of nodes subnetworks. Table 3-13 shows the basic VLANs that must be configured.

Table 3-13 Basic VLANs

VLAN name	Description	VLAN ID
OSM	OpenStack management	Value according to the environment
OSS	OpenStack storage	Value according to the environment
TVX	Tenant VXLAN	Value according to the environment
CR	Ceph replication	Value according to the environment
SR	Swift replication	Value according to the environment

3.4.1 General requirements

Here are the general network requirements:

- At least one data switch and one management switch.
- All the nodes must support IPMI and PXE boot.
- All the nodes require one BMC and one PXE port connection.
- The BMC ports of all the nodes must be configured to obtain an IP address through DHCP.
- Each node requires at least four 10 Gb connections in its data network.[1]
- Each node requires 1G connections for BMC and PXE ports.
- Single racks with single or redundant data switches with MLAG are supported.
- Multiple racks can be interconnected with traditional two-tier access-aggregation networking.[2]

Switch support considerations

The Open Platform for DBaaS on Power Systems solution includes a Lenovo G8052 as the management switch and Mellanox SX1410 and SX1710 data switches.

If different switches are used, they must be configurable either by cluster-genesis or by Juju OPNFV Infrastructure Deployer (JOID).

[1] Node ports are bonded (LACP) in pairs.

[2] Two-tier leaf-spine networks with L3 interconnect capable of supporting VXLAN are intended to be supported in the future.

3.5 Solution components and roles

The Open Platform for DBaaS on Power Systems cluster is built on top of IBM S822LC and S821LC servers, which are flexible systems that incorporate many innovations that are developed by the OpenPOWER Foundation. These are also low-cost servers, but still provide all the features and performance that are delivered by the POWER processor. In 1.2, “Open Platform for DBaaS on Power Systems components” on page 9, you can find more information about the components that are used in the Open Platform for DBaaS on Power Systems solution.

The OpenPOWER Foundation is a collaboration between many companies. By opening up the IBM POWER architecture, other vendors can develop and customize their own hardware and software, bringing many innovations to Power Systems servers. For more information, see OpenPOWER Foundation.

You can access the shell of your nodes (compute, controller, block storage, or object storage nodes) by checking the Horizon Dashboard to see which IP the node is using. You can accomplish this task by accessing the Inventory area, under the Operational Management tools, as shown in Figure 3-16.

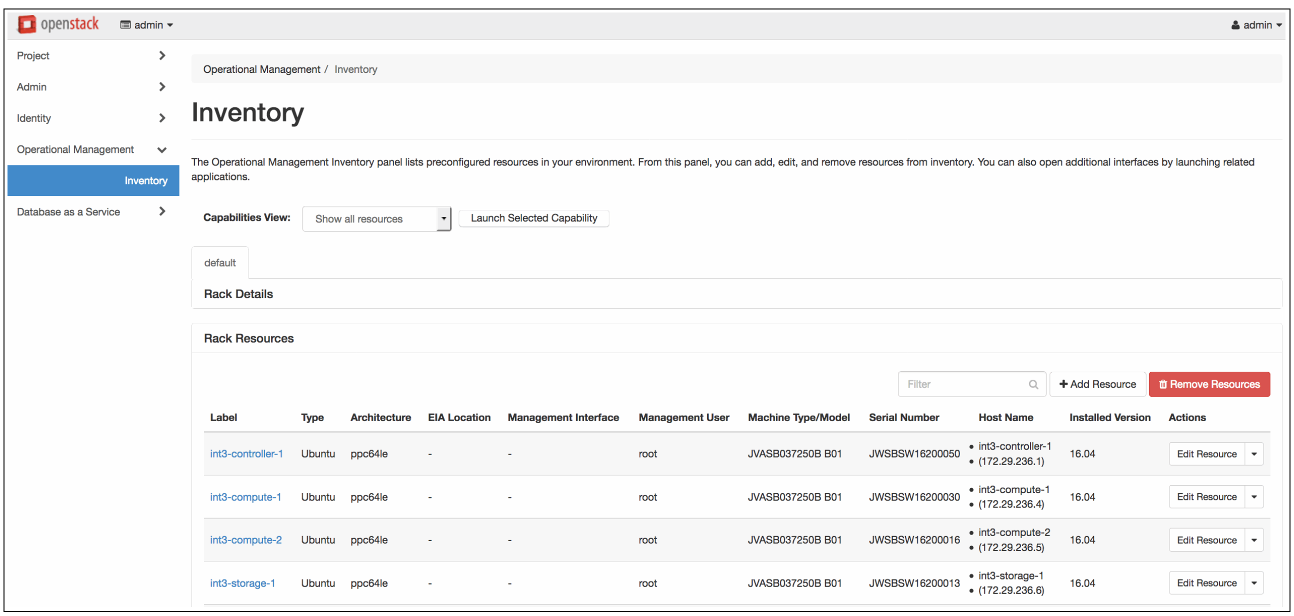


Figure 3-16 Checking the IP addresses of the nodes

You notice that the nodes are using IPs in the private network, which might not be routable from your computer. You can ssh to the cluster floating IP (the one that is used to access the Horizon Dashboard through the web), which takes you to one of the controller nodes (the workload from the controllers is balanced by using haproxy), and then you can ssh to the wanted node by using the IP that you obtained from the Inventory, as shown in Example 3-1.

Example 3-1 Accessing a compute node by using a controller node as a bridge

```
fabios-mbp:~ fabiomm$ ssh ubuntu@9.3.80.139
ubuntu@9.3.80.139's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-87-generic ppc64le)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
```

```
 * Support:        https://ubuntu.com/advantage
Last login: Tue Oct 17 13:11:28 2017 from 9.18.156.128

ubuntu@int3-controller-1:~$ ssh 172.29.236.4
The authenticity of host '172.29.236.4 (172.29.236.4)' can't be established.
ECDSA key fingerprint is SHA256:GCDHx+jXmRKjf8CyQJk5McyvmadAOQnHzV7hSYE1rJ8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.29.236.4' (ECDSA) to the list of known hosts.
ubuntu@172.29.236.4's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-87-generic ppc64le)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Tue Oct 17 12:12:19 2017 from 172.29.236.1
ubuntu@int3-compute-1:~$
```

3.5.1 Compute nodes

In the Open Platform for DBaaS on Power Systems solution, the compute nodes are deployed on IBM S821LC servers with the following configuration:

- Model and type: 8001-12C
- Two 8-core POWER8 2.328 GHz processors
- Eight 16 GB DDR4 memory DIMMs
- One integrated SATA controller
- One 4 TB 3.5” SATA HDD
- Two PCIe3 2-port 10 GbE SFP+ adapters
- Two power supplies

For more information about the Power System S821LC server, see *IBM Power System S821LC Technical Overview and Introduction*, REDP-5406.

The compute nodes run Ubuntu 16.04.3 LTS OS, as shown in Example 3-2.

Example 3-2 Operating system running in the compute node

```
root@int3-compute-1:~# cat /etc/*release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.3 LTS"
# Ansible managed:
/etc/ansible/roles/openstack_hosts/templates/openstack-release.j2 modified on
2017-03-01 03:41:06 by root on int3-controller-1

DISTRIB_ID="OSA"
DISTRIB_RELEASE="14.1.1"
DISTRIB_CODENAME="Newton"
DISTRIB_DESCRIPTION="OpenStack-Ansible"
NAME="Ubuntu"
VERSION="16.04.3 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.3 LTS"
VERSION_ID="16.04"
```

```
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial
```

The compute nodes also run the nova-compute service, so they can be added to the OpenStack cluster as a compute node, as shown in Example 3-3.

Example 3-3 The nova-compute service running in one of the compute nodes

```
root@int3-compute-1:~# systemctl status nova-compute.service
* nova-compute.service - nova openstack service
   Loaded: loaded (/etc/systemd/system/nova-compute.service; enabled; vendor
preset: enabled)
   Active: active (running) since Thu 2017-10-05 16:08:16 CDT; 1 weeks 4 days ago
 Main PID: 54055 (nova-compute)
   CGroup: /system.slice/nova-compute.service
           |- 54055 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-compute
--log-file=/var/log/nova/nova-compute.log
           |-135651 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/privsep-helper --config-file /etc/nova/nova.conf
--privsep_context vif_plug_linux_bridge.privsep.vif_plug --pri
           `-137452 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/privsep-helper --config-file /etc/nova/nova.conf
--privsep_context os_brick.privileged.default --privsep_sock_p

Oct 17 13:21:28 int3-compute-1 sudo[140806]:     nova : TTY=unknown ; PWD=/ ;
USER=root ; COMMAND=/openstack/venvs/nova-14.1.1/bin/nova-rootwrap
/etc/nova/rootwrap.conf touch -c /var/lib/nova/instances/_b
Oct 17 13:21:28 int3-compute-1 sudo[140806]: pam_unix(sudo:session): session
opened for user root by (uid=0)
Oct 17 13:22:29 int3-compute-1 sudo[140840]:     nova : TTY=unknown ; PWD=/ ;
USER=root ; COMMAND=/openstack/venvs/nova-14.1.1/bin/nova-rootwrap
/etc/nova/rootwrap.conf touch -c /var/lib/nova/instances/_b
Oct 17 13:22:29 int3-compute-1 sudo[140840]: pam_unix(sudo:session): session
opened for user root by (uid=0)
Oct 17 13:22:30 int3-compute-1 sudo[140840]: pam_unix(sudo:session): session
closed for user root
Oct 17 13:23:29 int3-compute-1 sudo[140887]:     nova : TTY=unknown ; PWD=/ ;
USER=root ; COMMAND=/openstack/venvs/nova-14.1.1/bin/nova-rootwrap
/etc/nova/rootwrap.conf touch -c /var/lib/nova/instances/_b
Oct 17 13:23:29 int3-compute-1 sudo[140887]: pam_unix(sudo:session): session
opened for user root by (uid=0)
Oct 17 13:23:30 int3-compute-1 sudo[140887]: pam_unix(sudo:session): session
closed for user root
Oct 17 13:24:29 int3-compute-1 sudo[140931]:     nova : TTY=unknown ; PWD=/ ;
USER=root ; COMMAND=/openstack/venvs/nova-14.1.1/bin/nova-rootwrap
/etc/nova/rootwrap.conf touch -c /var/lib/nova/instances/_b
Oct 17 13:24:29 int3-compute-1 sudo[140931]: pam_unix(sudo:session): session
opened for user root by (uid=0)
```

During the Open Platform for DBaaS on Power Systems deployment, the nova-service is properly configured and added to the cluster as a compute node. The configuration file `/etc/nova/nova.conf` contains all the configuration that enables it as a compute node. Figure 3-17 shows the **`nova service-list`** command, which shows the nova services running in the cluster, including the nova-compute services running in the compute nodes.

```
root@int3-controller-1-utility-container-1d6b2eea:~# nova service-list
+-----+------------------+-----------------------------------------------------+----------+---------+-------+----------------------------+-----------------+
| Id  | Binary           | Host                                                | Zone     | Status  | State | Updated_at                 | Disabled Reason |
+-----+------------------+-----------------------------------------------------+----------+---------+-------+----------------------------+-----------------+
| 5   | nova-consoleauth | int3-controller-2-nova-console-container-f23b1260   | internal | enabled | up    | 2017-10-17T18:13:06.000000 | -               |
| 8   | nova-consoleauth | int3-controller-1-nova-console-container-b67cbd8f   | internal | enabled | up    | 2017-10-17T18:13:05.000000 | -               |
| 11  | nova-consoleauth | int3-controller-3-nova-console-container-3b6c6d98   | internal | enabled | up    | 2017-10-17T18:13:02.000000 | -               |
| 14  | nova-cert        | int3-controller-3-nova-cert-container-fc720fa8      | internal | enabled | up    | 2017-10-17T18:13:07.000000 | -               |
| 17  | nova-cert        | int3-controller-1-nova-cert-container-0370f9c3      | internal | enabled | up    | 2017-10-17T18:13:08.000000 | -               |
| 20  | nova-cert        | int3-controller-2-nova-cert-container-56ac88a5      | internal | enabled | up    | 2017-10-17T18:13:08.000000 | -               |
| 23  | nova-conductor   | int3-controller-3-nova-conductor-container-4aef33a4 | internal | enabled | up    | 2017-10-17T18:13:08.000000 | -               |
| 26  | nova-conductor   | int3-controller-2-nova-conductor-container-6b08668a | internal | enabled | up    | 2017-10-17T18:13:07.000000 | -               |
| 29  | nova-conductor   | int3-controller-1-nova-conductor-container-85c43e90 | internal | enabled | up    | 2017-10-17T18:13:04.000000 | -               |
| 89  | nova-scheduler   | int3-controller-1-nova-scheduler-container-13839798 | internal | enabled | up    | 2017-10-17T18:13:02.000000 | -               |
| 92  | nova-scheduler   | int3-controller-2-nova-scheduler-container-1568f938 | internal | enabled | up    | 2017-10-17T18:13:06.000000 | -               |
| 95  | nova-scheduler   | int3-controller-3-nova-scheduler-container-d19a6e41 | internal | enabled | up    | 2017-10-17T18:13:03.000000 | -               |
| 98  | nova-compute     | int3-compute-2                                      | nova     | enabled | up    | 2017-10-17T18:13:01.000000 | -               |
| 101 | nova-compute     | int3-compute-1                                      | nova     | enabled | up    | 2017-10-17T18:13:04.000000 | -               |
+-----+------------------+-----------------------------------------------------+----------+---------+-------+----------------------------+-----------------+
```

Figure 3-17 The nova-compute service running in the compute nodes

Figure 3-18 shows the output of the **`nova host-list`** command, showing the compute nodes running the `compute` service in the nova availability zone.

```
root@int3-controller-1-utility-container-1d6b2eea:~# nova host-list
+-----------------------------------------------------+-------------+----------+
| host_name                                           | service     | zone     |
+-----------------------------------------------------+-------------+----------+
| int3-controller-2-nova-console-container-f23b1260   | consoleauth | internal |
| int3-controller-1-nova-console-container-b67cbd8f   | consoleauth | internal |
| int3-controller-3-nova-console-container-3b6c6d98   | consoleauth | internal |
| int3-controller-3-nova-cert-container-fc720fa8      | cert        | internal |
| int3-controller-1-nova-cert-container-0370f9c3      | cert        | internal |
| int3-controller-2-nova-cert-container-56ac88a5      | cert        | internal |
| int3-controller-3-nova-conductor-container-4aef33a4 | conductor   | internal |
| int3-controller-2-nova-conductor-container-6b08668a | conductor   | internal |
| int3-controller-1-nova-conductor-container-85c43e90 | conductor   | internal |
| int3-controller-1-nova-scheduler-container-13839798 | scheduler   | internal |
| int3-controller-2-nova-scheduler-container-1568f938 | scheduler   | internal |
| int3-controller-3-nova-scheduler-container-d19a6e41 | scheduler   | internal |
| int3-compute-2                                      | compute     | nova     |
| int3-compute-1                                      | compute     | nova     |
+-----------------------------------------------------+-------------+----------+
```

Figure 3-18 The compute nodes available in the nova zone

The **`nova hypervisor-list`** command shows the compute nodes that are available in the cluster, as shown in Figure 3-19.

```
root@int3-controller-1-utility-container-1d6b2eea:~# nova hypervisor-list
+----+--------------------------+-------+---------+
| ID | Hypervisor hostname      | State | Status  |
+----+--------------------------+-------+---------+
| 5  | int3-compute-2.local.lan | up    | enabled |
| 8  | int3-compute-1.local.lan | up    | enabled |
+----+--------------------------+-------+---------+
```

Figure 3-19 Nova hypervisors available in the cluster

All VMs that are deployed in the Open Platform for DBaaS on Power Systems solution run on top of Kernel-based Virtual Machine (KVM) or Quick Emulator (QEMU) in the compute nodes, as shown in Example 3-4.

Example 3-4   KVM running in the Open Platform for DBaaS on Power Systems cluster

```
root@int3-compute-1:~# virsh list
 Id    Name                           State
----------------------------------------------------
 2     instance-00000008              running
 13    instance-00000029              running
 14    instance-00000023              running
 16    instance-00000032              running
 24    instance-00000065              running
 27    instance-00000074              running
```

You can convert the KVM instance name to the OpenStack VM name by using the **`nova show`** command, as shown in Figure 3-20.

```
root@int3-controller-1-utility-container-1d6b2eea:~# nova show test-mariadb
+--------------------------------------+------------------------------------------------------------
| Property                             | Value
+--------------------------------------+------------------------------------------------------------
| OS-DCF:diskConfig                    | MANUAL
| OS-EXT-AZ:availability_zone          | nova
| OS-EXT-SRV-ATTR:host                 | int3-compute-1
| OS-EXT-SRV-ATTR:hostname             | test-mariadb
| OS-EXT-SRV-ATTR:hypervisor_hostname  | int3-compute-1.local.lan
| OS-EXT-SRV-ATTR:instance_name        | instance-00000023
| OS-EXT-SRV-ATTR:kernel_id            |
| OS-EXT-SRV-ATTR:launch_index         | 0
| OS-EXT-SRV-ATTR:ramdisk_id           |
| OS-EXT-SRV-ATTR:reservation_id       | r-mv1lt3r9
| OS-EXT-SRV-ATTR:root_device_name     | /dev/vda
| OS-EXT-SRV-ATTR:user_data            | -
| OS-EXT-STS:power_state               | 1
| OS-EXT-STS:task_state                | -
| OS-EXT-STS:vm_state                  | active
| OS-SRV-USG:launched_at               | 2017-10-12T13:42:39.000000
| OS-SRV-USG:terminated_at             | -
| accessIPv4                           |
| accessIPv6                           |
| config_drive                         | True
| created                              | 2017-10-11T14:09:58Z
| description                          | test-mariadb
| external network                     | 9.3.80.166
| flavor                               | medium (64c1b226-249e-4e79-adc9-9215f4288ee5)
| hostId                               | c2480eb46dc481af299cdf047960cbf0067b9c804bc84f012453af04
| host_status                          | UP
| id                                   | 42eb9836-4782-4b64-bf55-c205b59461ad
| image                                | mariadb-10.1-c-dib-2017-10-06 (7f890077-ab1d-4d84-bddd-5ffcfb5638a0)
| key_name                             | -
| locked                               | False
| metadata                             | {}
| name                                 | test-mariadb
| os-extended-volumes:volumes_attached | [{"id": "f048baae-4e1e-4556-968f-177b30ba2bb1", "delete_on_termination": true}, {"id": "6f61dabf-893b-4b2c-8a00-4d9e28e54da8", "delete_on_termination": false}] |
| progress                             | 0
| security_groups                      | SecGroup_bdb17397-dc8f-4dc8-8e9d-cbd124ee2d44
| status                               | ACTIVE
| tags                                 | []
| tenant_id                            | 6e5d1884c6ab45bfb00594b1a921bd61
| trove_infra_net network              | 172.29.236.107
| updated                              | 2017-10-12T13:44:19Z
| user_id                              | e52bc2cfeadb418d840798d6e4650bbe
+--------------------------------------+------------------------------------------------------------
```

Figure 3-20   Obtaining the virtual machine information

3.5.2 Controller nodes

The controller nodes in the Open Platform for DBaaS on Power Systems solution are deployed on IBM Power S822LC servers with the following configuration:

- Model and type: 8001-22C
- Two 10-core POWER8 2.92 GHz
- Eight 16 GB DDR4 Memory DIMMs
- One integrated SATA Controller
- One 1.9 TB SATA SSD Disk
- Two Intel 2-Port SFP+ 10 Gbps network adapters
- Two power supplies

For more information about the Power System S822LC server, see *IBM Power System S822LC Technical Overview and Introduction*, REDP-5283.

The controller nodes also run Ubuntu 16.04.3 LTS, as shown in Example 3-5.

Example 3-5 Ubuntu operating system running in the controller node

```
ubuntu@int3-controller-1:~$ cat /etc/*release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.3 LTS"
# Ansible managed:
/etc/ansible/roles/openstack_hosts/templates/openstack-release.j2 modified on
2017-03-01 03:41:06 by root on int3-controller-1

DISTRIB_ID="OSA"
DISTRIB_RELEASE="14.1.1"
DISTRIB_CODENAME="Newton"
DISTRIB_DESCRIPTION="OpenStack-Ansible"
NAME="Ubuntu"
VERSION="16.04.3 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.3 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial
```

All the OpenStack components and services in the controller nodes are deployed by using LXC containers. LXC (also known as Linux Containers) is a virtualization technology on the OS level, enabling the running of multiple isolated Linux systems on a control host by using a single Linux Kernel. For more information about LXC, see the LXC documentation at Ubuntu.

A different container is created for each of the OpenStack components that run in controller node. Also, other containers are created for support tools, such as Kibana, Elasticsearch, Logstash, and Nagios, as shown in Example 3-6. The usage of multiple containers enables the isolation of components, avoiding problems in a single component that can affect others. While logged in as root in the controller nodes, you can use the **lxc-ls** command to view the available containers.

Example 3-6 Containers running in the controller node

```
root@int3-controller-1:~# lxc-ls
int3-controller-1-elasticsearch           int3-controller-1-kibana
int3-controller-1-logstash
int3-controller-1-nagios
int3-controller-1_cinder_api_container-5dfc036b
int3-controller-1_cinder_scheduler_container-3c5d9754
int3-controller-1_cinder_volumes_container-f15adc2f
int3-controller-1_galera_container-b9c48dbc
int3-controller-1_glance_container-1d55dadc
```

```
int3-controller-1_heat_apis_container-62e84c50
int3-controller-1_heat_engine_container-f79d3397
int3-controller-1_horizon_container-8d751b8b
int3-controller-1_keystone_container-b7bd46d0
int3-controller-1_memcached_container-e06a95b2
int3-controller-1_neutron_agents_container-ac8f401c
int3-controller-1_neutron_server_container-26d23625
int3-controller-1_nova_api_metadata_container-2d6c1eb5
int3-controller-1_nova_api_os_compute_container-23d417d2
int3-controller-1_nova_cert_container-0370f9c3
int3-controller-1_nova_conductor_container-85c43e90
int3-controller-1_nova_console_container-b67cbd8f
int3-controller-1_nova_scheduler_container-13839798
int3-controller-1_rabbit_mq_container-b0984b2f
int3-controller-1_repo_container-0b08a8ab
int3-controller-1_rsyslog_container-49104485
int3-controller-1_swift_proxy_container-024651d1
int3-controller-1_trove_api_container-56496a67
int3-controller-1_trove_conductor_container-cc84f397
int3-controller-1_trove_taskmanager_container-fae8accb
int3-controller-1_utility_container-1d6b2eea
```

To connect to one of the containers and view its processes or perform other operations, use the **lxc-attach** command, as shown in Example 3-7.

Example 3-7 Performing operations in a container

```
root@int3-controller-1:~# lxc-attach -n int3-controller-1_nova_conductor_container-85c43e90
root@int3-controller-1-nova-conductor-container-85c43e90:~# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Oct05 ?        00:00:05 /sbin/init
root        43     1  0 Oct05 ?        00:00:08 /lib/systemd/systemd-journald
root        92     1  0 Oct05 ?        00:00:01 /usr/sbin/cron -f
root       145     1  0 Oct05 ?        00:00:00 /sbin/dhclient -1 -v -pf
/run/dhclient.eth0.pid -lf /var/lib/dhcp/dhclient.eth0.leases -I -df
/var/lib/dhcp/dhclient6.eth0.leases e
root       194     1  0 Oct05 ?        00:00:00 /usr/sbin/sshd -D
root       196     1  0 Oct05 pts/1    00:00:00 /sbin/agetty --noclear --keep-baud pts/1
115200 38400 9600 vt220
root       197     1  0 Oct05 pts/3    00:00:00 /sbin/agetty --noclear --keep-baud pts/3
115200 38400 9600 vt220
root       198     1  0 Oct05 lxc/console 00:00:00 /sbin/agetty --noclear --keep-baud console
115200 38400 9600 vt220
root       199     1  0 Oct05 pts/0    00:00:00 /sbin/agetty --noclear --keep-baud pts/0
115200 38400 9600 vt220
root       200     1  0 Oct05 pts/2    00:00:00 /sbin/agetty --noclear --keep-baud pts/2
115200 38400 9600 vt220
nova      5145     1  0 Oct05 ?        01:18:07 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova      5153  5145  0 Oct05 ?        00:19:46 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova      5154  5145  0 Oct05 ?        00:20:00 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova      5155  5145  0 Oct05 ?        00:19:49 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
```

```
nova       5156   5145  0 Oct05 ?        00:19:48 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5157   5145  0 Oct05 ?        00:19:43 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5158   5145  0 Oct05 ?        00:20:15 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5159   5145  0 Oct05 ?        00:20:06 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5160   5145  0 Oct05 ?        00:20:09 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5161   5145  0 Oct05 ?        00:19:55 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5162   5145  0 Oct05 ?        00:19:44 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5163   5145  0 Oct05 ?        00:19:53 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5164   5145  0 Oct05 ?        00:20:20 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5165   5145  0 Oct05 ?        00:19:39 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5166   5145  0 Oct05 ?        00:19:39 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5167   5145  0 Oct05 ?        00:20:19 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
nova       5168   5145  0 Oct05 ?        00:20:23 /openstack/venvs/nova-14.1.1/bin/python
/openstack/venvs/nova-14.1.1/bin/nova-conductor --log-file=/var/log/nova/nova-conductor.log
syslog     5398      1  0 Oct05 ?        00:00:09 /usr/sbin/rsyslogd -n
root       5758      1  0 Oct05 ?        00:20:42 /usr/share/filebeat/bin/filebeat -c
/etc/filebeat/filebeat.yml -path.home /usr/share/filebeat -path.config /etc/filebeat -path.data
root      44442      0  0 10:40 ?        00:00:00 /bin/bash
root      44452  44442  0 10:40 ?        00:00:00 ps -ef
root@int3-controller-1-nova-conductor-container-85c43e90:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
321: eth0@if322: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
qlen 1000
    link/ether 00:16:3e:98:54:3a brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.3.16/24 brd 10.0.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:fe98:543a/64 scope link
       valid_lft forever preferred_lft forever
323: eth1@if324: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UP group default
qlen 1000
    link/ether 00:16:3e:ec:6e:2a brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.29.236.249/22 brd 172.29.239.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:feec:6e2a/64 scope link
       valid_lft forever preferred_lft forever
root@int3-controller-1-nova-conductor-container-85c43e90:~# exit
```

```
exit
root@int3-controller-1:~#
```

Every controller node in the Open Platform for DBaaS on Power Systems cluster has a special container that is called utility-container, which contains the OpenStack client tools for the CLI OpenStack commands, as shown in Figure 3-21.

```
root@int3-controller-1:~# lxc-ls | grep utility
int3-controller-1_trove_conductor_container-cc84f397     int3-controller-1_trove_taskmanager_container-fae8accb    int3-controller-1_utility_container-1d6b2eea
root@int3-controller-1:~# lxc-attach -n int3-controller-1_utility_container-1d6b2eea
root@int3-controller-1-utility-container-1d6b2eea:~# ls -l /root/
total 291468
-rw------- 1 root root       731 Oct  5 15:27 openrc
-rw-r--r-- 1 root root 298451456 Oct  5 09:54 xenial-server-cloudimg-ppc64el-disk1.img
root@int3-controller-1-utility-container-1d6b2eea:~# source /root/openrc
root@int3-controller-1-utility-container-1d6b2eea:~# nova list
+--------------------------------------+---------------------+--------+------------+-------------+------------------------------------------------------+
| ID                                   | Name                | Status | Task State | Power State | Networks                                             |
+--------------------------------------+---------------------+--------+------------+-------------+------------------------------------------------------+
| efda00f8-b48c-4f8a-8706-ac68c970ab1c | dibvm               | ACTIVE | -          | Running     | external=9.3.80.162                                  |
| ad29e9c3-dcc7-41b5-bde7-ae6805855dfb | eric-mysql-restored | ACTIVE | -          | Running     | external=9.3.80.169; trove_infra_net=172.29.236.109 |
| 477f1fc2-bef4-4da2-8c65-14ee58af8a62 | luke-deployer       | ACTIVE | -          | Running     | external=9.3.80.167                                  |
| 1b5a6136-32a0-4a80-babe-150054789a00 | luke-postgres       | ACTIVE | -          | Running     | external=9.3.80.164; trove_infra_net=172.29.236.110 |
| 42eb9836-4782-4b64-bf55-c205b59461ad | test-mariadb        | ACTIVE | -          | Running     | external=9.3.80.166; trove_infra_net=172.29.236.107 |
| 163e2075-9467-41cd-ad0a-d29943e5852a | test-redis          | ACTIVE | -          | Running     | external=9.3.80.160; trove_infra_net=172.29.236.106 |
| bea6c6f9-9198-4f35-89ba-4c3cc30ce0d5 | trove-dev-manoj     | ACTIVE | -          | Running     | external=9.3.80.165                                  |
+--------------------------------------+---------------------+--------+------------+-------------+------------------------------------------------------+
root@int3-controller-1-utility-container-1d6b2eea:~# neutron subnet-list
+--------------------------------------+---------------------------------------------------+-------------------+------------------------------------------------------+
| id                                   | name                                              | cidr              | allocation_pools                                     |
+--------------------------------------+---------------------------------------------------+-------------------+------------------------------------------------------+
| 17fe1862-8c3b-4edf-97a7-ae2578835cf1 | trove_infra_subnet                                | 172.29.236.0/22   | {"start": "172.29.236.100", "end": "172.29.236.110"} |
| 38f03fda-3ba0-4689-8b39-43aef44b9bbe | tenant1-subnet1                                   | 192.168.20.0/24   | {"start": "192.168.20.2", "end": "192.168.20.254"}   |
| 3eb7cecc-a56c-440f-8df2-1e9ca12ba475 | external-subnet                                   | 9.3.80.0/24       | {"start": "9.3.80.157", "end": "9.3.80.169"}         |
| 45819159-6647-45a7-a20f-d325aa408e8d | HA subnet tenant 6e5d1884c6ab45bfb00594b1a921bd61 | 169.254.192.0/18  | {"start": "169.254.192.1", "end": "169.254.255.254"} |
+--------------------------------------+---------------------------------------------------+-------------------+------------------------------------------------------+
root@int3-controller-1-utility-container-1d6b2eea:~# openstack
WARNING: openstackclient.common.utils is deprecated and will be removed after Jun 2017. Please use osc_lib.utils
(openstack) compute service list
+-----+------------------+-----------------------------------------------------+----------+---------+-------+----------------------------+
|  ID | Binary           | Host                                                | Zone     | Status  | State | Updated At                 |
+-----+------------------+-----------------------------------------------------+----------+---------+-------+----------------------------+
|   5 | nova-consoleauth | int3-controller-2-nova-console-container-f23b1260   | internal | enabled | up    | 2017-10-18T16:14:02.000000 |
|   8 | nova-consoleauth | int3-controller-1-nova-console-container-b67cbd8f   | internal | enabled | up    | 2017-10-18T16:14:04.000000 |
|  11 | nova-consoleauth | int3-controller-3-nova-console-container-3b6c6d98   | internal | enabled | up    | 2017-10-18T16:14:10.000000 |
|  14 | nova-cert        | int3-controller-3-nova-cert-container-fc720fa8      | internal | enabled | up    | 2017-10-18T16:14:08.000000 |
|  17 | nova-cert        | int3-controller-1-nova-cert-container-0370f9c3      | internal | enabled | up    | 2017-10-18T16:14:06.000000 |
|  20 | nova-cert        | int3-controller-2-nova-cert-container-56ac88a5      | internal | enabled | up    | 2017-10-18T16:14:04.000000 |
|  23 | nova-conductor   | int3-controller-3-nova-conductor-container-4aef33a4 | internal | enabled | up    | 2017-10-18T16:14:08.000000 |
|  26 | nova-conductor   | int3-controller-2-nova-conductor-container-6b08668a | internal | enabled | up    | 2017-10-18T16:14:09.000000 |
|  29 | nova-conductor   | int3-controller-1-nova-conductor-container-85c43e90 | internal | enabled | up    | 2017-10-18T16:14:10.000000 |
|  89 | nova-scheduler   | int3-controller-1-nova-scheduler-container-13839798 | internal | enabled | up    | 2017-10-18T16:14:02.000000 |
|  92 | nova-scheduler   | int3-controller-2-nova-scheduler-container-1568f938 | internal | enabled | up    | 2017-10-18T16:14:02.000000 |
|  95 | nova-scheduler   | int3-controller-3-nova-scheduler-container-d19a6e41 | internal | enabled | up    | 2017-10-18T16:14:06.000000 |
|  98 | nova-compute     | int3-compute-2                                      | nova     | enabled | up    | 2017-10-18T16:14:05.000000 |
| 101 | nova-compute     | int3-compute-1                                      | nova     | enabled | up    | 2017-10-18T16:14:04.000000 |
+-----+------------------+-----------------------------------------------------+----------+---------+-------+----------------------------+
```

Figure 3-21 The utility container

HAproxy and keepalive in the Open Platform for DBaaS

The Open Platform for DBaaS on Power Systems solution uses HAproxy to balance the workload of its services through the available nodes in the cluster (compute, controller, block storage, and object storage). The HAproxy daemon runs in the controller nodes and balances the workload for the following services:

- Cinder
- Elasticsearch
- Galera
- Glance
- Heat
- Horizon
- Keystone
- Kibana
- Logstash
- Nagios
- Neutron

- Nova
- RabbitMQ
- Swift
- Trove

HAproxy is a no-charge open source software that provides high-availability and load-balancing features for HTTP and TCP connections. The Open Platform for DBaaS on Power Systems solution uses HAproxy as a load balancer for its components, balancing the workload of requests to the applications mentioned previously. The HAproxy instance is put in front of the connections to the remaining applications and distributes the workload between the existing instances in the cluster.

Figure 3-22 shows how the HAproxy load balancer distributes the workload between the controller nodes for one of the services, in this example, the Kibana service. All connections that are received from clients in the floating IP address (used to access the Horizon interface) in the port 8443 (the Kibana port) are distributed to one of the controller nodes that is running Kibana and listening on port 8443.

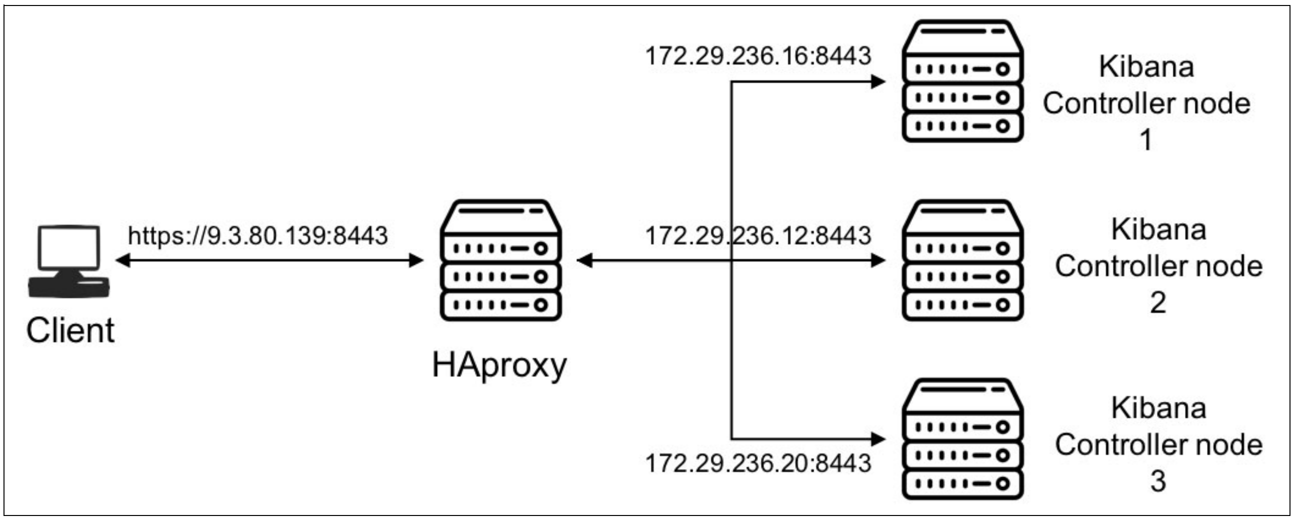


Figure 3-22 HAproxy Kibana load balance workflow

The main configuration file for HAproxy (located on all controller nodes) is `/etc/haproxy/haproxy.cfg`. Example 3-8 shows the section of the `haproxy.cfg` configuration file that handles the Kibana workload distribution between the existing Kibana instances in the Open Platform for DBaaS on Power Systems cluster. HAproxy listens on port 8443 and sends the connections to the back end `kibana-https-back` that contains the IP addresses of the existing LXC containers (in the controller nodes) where the Kibana service is running, so the workload is distributed between them.

Example 3-8 HAproxy configuration file distributing the Kibana workload

```
frontend kibana-https-front
bind *:8443
    mode tcp
    timeout client 60m
    option  tcplog
    default_backend kibana-https-back

backend kibana-https-back
    mode tcp
    option ssl-hello-chk
```

```
timeout server 60m
balance source
server int3-controller-1-kibana 172.29.236.12:8443 check port 8443 inter 10s fall 1 rise 1
server int3-controller-2-kibana 172.29.236.16:8443 check port 8443 inter 10s fall 1 rise 1
server int3-controller-3-kibana 172.29.236.20:8443 check port 8443 inter 10s fall 1 rise 1
```

This configuration helps workload distribution, enables high availability of the services, and is also highly scalable, given that new instances of each service can be added to the cluster and HAproxy can include them in the workload distribution.

If a single HAproxy instance is used to provide load balancing to the Open Platform for DBaaS on Power Systems components, then the HAproxy itself can become a single point of failure, given that all components are unavailable if it fails. To avoid such a situation, the Open Platform for DBaaS on Power Systems solution uses Keepalived to provide high availability for the HAproxy instance that is handling the connections to the cluster.

Keepalived is open source software that is used for load-balancing and high-availability purposes. Keepalived runs on a Linux Virtual Server (LVS) kernel module, where one LVS server is denominated as the *MASTER* (in this case, one of the controller nodes) while other LVS servers are denominated as *BACKUP* (the other controller nodes). The MASTER node must balance the workload through the real servers (in this case, the HAproxy servers, which also run in the controller nodes) and checks the integrity of the service (in the the Open Platform for DBaaS on Power Systems solution, the HAproxy service) on each real server (the controller nodes).

Keepalived uses the Virtual Router Redundancy Protocol (VRRP) to check the integrity between its instances, which the MASTER uses to send keepalive packets at regular intervals. If an issue happens to the MASTER and stops sending such packets, a new MASTER is elected among the BACKUP nodes, as shown in Figure 3-23.

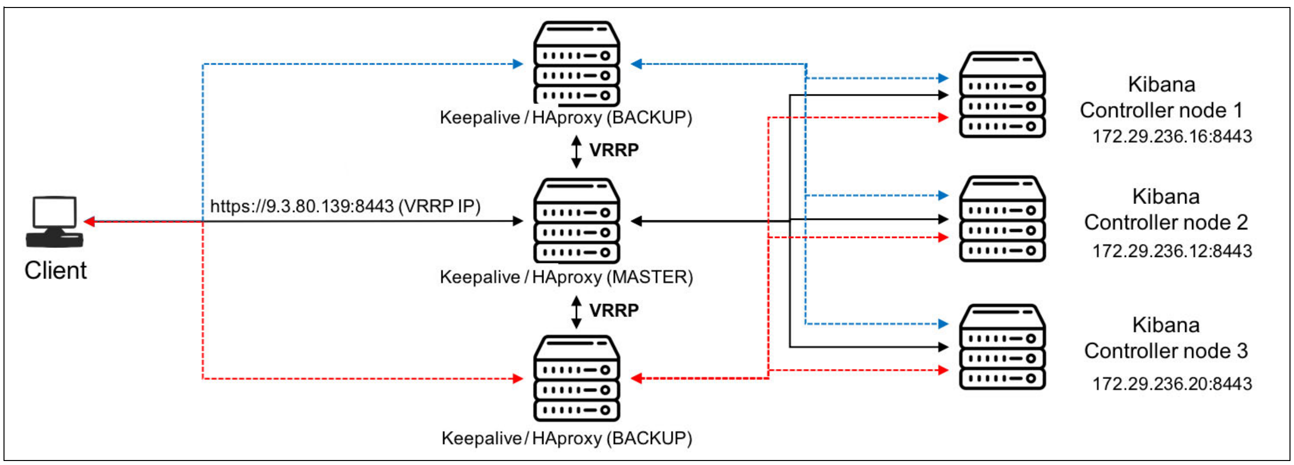


Figure 3-23 Keepalived managing connections to the HAproxy instances

Using such technology (Keepalived and HAproxy), the Open Platform for DBaaS on Power Systems solution helps ensure highly available, load-balanced, and scalable connections to all its components. The first controller node is selected as the MASTER Keepalived, as shown in Example 3-9.

Example 3-9 Keepalived configuration file in the MASTER node

```
root@int3-controller-1:~# cat /etc/keepalived/keepalived.conf
# Copyright 2015, Jean-Philippe Evrard <jean-philippe@evrard.me>
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

vrrp_sync_group haproxy {
  group {
    external
    internal
  }
  notify "/etc/keepalived/haproxy_notify.sh"
}

vrrp_script haproxy_check_script {
  script "killall -0 haproxy"
  interval "5"   # checking every "5" seconds (default: 5 seconds)
  fall "3"           # require "3" failures for KO (default: 3)
  rise "6"           # require "6" successes for OK (default: 6)
}
vrrp_script pingable_check_script {
  script "ping -c 1 193.0.14.129 1>&2"
  interval "10"   # checking every "10" seconds (default: 5 seconds)
  fall "2"           # require "2" failures for KO (default: 3)
  rise "4"           # require "4" successes for OK (default: 6)
}

vrrp_instance internal {
  interface br-mgmt
  state MASTER
  virtual_router_id 11
  priority 150
  authentication {
    auth_type PASS
    auth_pass 2ceeee79c02237c4e357c7efb19ce4290f9cf851160342b8f1b6
  }
  virtual_ipaddress {
    172.29.236.50 dev br-mgmt
  }
  track_script {
```

```
    haproxy_check_script
    pingable_check_script
  }

}
vrrp_instance external {
  interface osbond0
  state MASTER
  virtual_router_id 160
  priority 150
  authentication {
    auth_type PASS
    auth_pass 2ceeee79c02237c4e357c7efb19ce4290f9cf851160342b8f1b6
  }
  virtual_ipaddress {
    9.3.80.139 dev osbond0
  }
  track_script {
    haproxy_check_script
    pingable_check_script
  }

}
```

The remaining controller nodes are configured as BACKUP in the Keepalived configuration file, as shown in Example 3-10.

Example 3-10 Keepalived configuration file in the BACKUP node

```
root@int3-controller-2:~# cat /etc/keepalived/keepalived.conf .
# Copyright 2015, Jean-Philippe Evrard <jean-philippe@evrard.me>
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

vrrp_sync_group haproxy {
  group {
    external
    internal
  }
  notify "/etc/keepalived/haproxy_notify.sh"
}

vrrp_script haproxy_check_script {
  script "killall -0 haproxy"
  interval "5"   # checking every "5" seconds (default: 5 seconds)
  fall "3"           # require "3" failures for KO (default: 3)
```

```
  rise "6"           # require "6" successes for OK (default: 6)
}
vrrp_script pingable_check_script {
  script "ping -c 1 193.0.14.129 1>&2"
  interval "10"   # checking every "10" seconds (default: 5 seconds)
  fall "2"           # require "2" failures for KO (default: 3)
  rise "4"           # require "4" successes for OK (default: 6)
}

vrrp_instance internal {
  interface br-mgmt
  state BACKUP
  virtual_router_id 11
  priority 50
  authentication {
    auth_type PASS
    auth_pass 2ceeee79c02237c4e357c7efb19ce4290f9cf851160342b8f1b6
  }
  virtual_ipaddress {
    172.29.236.50 dev br-mgmt
  }
  track_script {
    haproxy_check_script
    pingable_check_script
  }

}
vrrp_instance external {
  interface osbond0
  state BACKUP
  virtual_router_id 160
  priority 50
  authentication {
    auth_type PASS
    auth_pass 2ceeee79c02237c4e357c7efb19ce4290f9cf851160342b8f1b6
  }
  virtual_ipaddress {
    9.3.80.139 dev osbond0
  }
  track_script {
    haproxy_check_script
    pingable_check_script
  }

}
```

3.5.3 Block storage nodes

Block storage nodes are Ceph OSD nodes servers with built-in storage providing block storage for the VMs that are hosted on the compute nodes.

Ceph is an open source, highly scalable software storage platform whose main focus is object-based storage. It provides block-based and file-based storage under a unified and scalable distributed storage system without a single point of failure.

Ceph is based on three main services:

- Ceph monitor (ceph-mon): Maintains maps of the cluster state, which are required for Ceph daemons coordination. These maps are:
  - Monitor map
  - Manager map
  - OSD map
  - Controlled Replication Under Scalable Hashing (CRUSH) map
- Ceph manager (ceph-mgr): Tracks the state of the Ceph cluster: storage utilization, performance metrics, system load, and so on.
- Ceph OSD (object storage daemon, which is known as ceph-osd): Responsible for data storing, data replication, data retrieval, data recovery, and rebalancing.

> **Note:** For redundancy and high availability purposes, at least three monitors, three OSDs, and two managers are normally required.

Ceph cluster

The Ceph storage cluster is based on the RADOS service, which consists of a collection of storage nodes, daemons, and the CRUSH algorithm.

Storage nodes

Storage nodes provide the operating environment, both hardware and software, for block storage. As mentioned in 3.3, “Infrastructure sizing” on page 61, the Power System 8001-12C servers are used as the storage nodes.

Power System 8001-12C servers provide great data throughput and performance for high-value, storage-centric workloads with up to 12 drives (SFFs/LFFs) built into the chassis.

Figure 3-24 shows the Ceph OSDs running on a storage node.

```
root@int3-storage-3:~# ps aux --sort=uid | grep -v root | grep -v grep | grep ceph
ceph       12838  0.4  0.0 1185344 257984 ?      Ssl  Oct05  82:49 /usr/bin/ceph-osd -f --cluster ceph --id 2 --setuser ceph --setgroup ceph
ceph       13442  0.4  0.0 1223872 309248 ?      Ssl  Oct05  90:12 /usr/bin/ceph-osd -f --cluster ceph --id 5 --setuser ceph --setgroup ceph
ceph       14064  0.4  0.0 1236608 319744 ?      Ssl  Oct05  91:22 /usr/bin/ceph-osd -f --cluster ceph --id 8 --setuser ceph --setgroup ceph
ceph       14682  0.3  0.0 1167424 258496 ?      Ssl  Oct05  78:24 /usr/bin/ceph-osd -f --cluster ceph --id 10 --setuser ceph --setgroup ceph
ceph       15369  0.4  0.0 1240640 337152 ?      Ssl  Oct05  90:12 /usr/bin/ceph-osd -f --cluster ceph --id 12 --setuser ceph --setgroup ceph
ceph       16181  0.4  0.0 1221440 302464 ?      Ssl  Oct05  87:44 /usr/bin/ceph-osd -f --cluster ceph --id 15 --setuser ceph --setgroup ceph
ceph       17044  0.5  0.0 1228096 297536 ?      Ssl  Oct05 102:30 /usr/bin/ceph-osd -f --cluster ceph --id 18 --setuser ceph --setgroup ceph
root@int3-storage-3:~#
```

Figure 3-24 OSDs running on a storage-3 node

RADOS daemons

Instead of using a centralized metadata server that manages data storage and retrieval operations, and the communications with external services and applications, Ceph RADOS uses lightweight daemons that handle these tasks on each storage node:

OSDs The ceph-osd daemons access and write data into a file system and provides access to the data over the cluster network.

Monitors The ceph-mon daemons communicate with the clients to manipulate the stored data inside the cluster. They are next to the OSDs and manage the data consistency in the cluster.

CRUSH algorithm

RADOS cluster uses the CRUSH algorithm to manages data placement and how data is striped, mapped, and replicated across the cluster. The cluster also manages how data is retrieved from the nodes the fastest way without bottlenecks.

CRUSH enables the storage cluster to scale, rebalance, and recover dynamically. It also helps the cluster operate efficiently by determining the best way to distribute workloads to clients and OSDs in the following manner:

- CRUSH uses a hierarchical cluster map, the CRUSH map, to collect information about storage capacity on nodes, about the topology and infrastructure, and to manage cluster redundancy.
- The CRUSH map contains a list of all available nodes in the cluster and their storage, and the hierarchical definition of the existing infrastructure: servers, racks, rows, and sites.
- CRUSH enables fault tolerance by logically nesting OSDs into the hierarchical definition components, such as racks or switches, so that you can isolate a zone of faulty hardware.

The ceph-mon daemons are in charged of the adjustment and propagation of the CRUSH map in case of infrastructure changes.

Ceph block storage

Ceph provides block-storage capability through RBD, which is a virtual disk that can be attached to a bare metal or VM Linux-based server.

When an application writes data to Ceph by using a block device, it stores block device images as objects and automatically stripes and replicates the data across the cluster. By striping images across the cluster, Ceph improves read access performance for large block device images.

RBD provides the following features:

- Thin-provisioned images
- Up to 16 EB images
- Resizable images
- Asynchronously mirrored images between two Ceph clusters
- Image copy or rename
- Image import and export
- Configurable striping
- In-memory caching
- Read-only snapshots
- Revert to snapshots
- Copy-on-write cloning
- Incremental backup
- Multisite asynchronous replication
- Back end for OpenStack

Ceph back end

Cinder is the OpenStack component that manages volumes on block storage back ends that provide the storage to the databases. In the Open Platform for DBaaS on Power Systems solution, Ceph is used as the block storage back end.

The interaction between storage requests, cinder, and the Ceph back end is achieved through three main cinder services:

cinder-api The API server that handles storage requests and forwards them to either cinder-scheduler or to cinder-volume.

cinder-scheduler This service receives the storage requests from the API server and determines where the volumes are allocated, and then forwards the request to cinder-volume.

cinder-volume This service processes cinder-api and cinder-scheduler volume requests, interacting with the storage back ends through their drivers.

Ceph presents its block devices as a pool of volumes to the cinder-volume, so when a volume request is received, a volume can be taken from the pool and attached to the compute instance.

Figure 3-25 shows the interaction between Cinder and Ceph RBD.

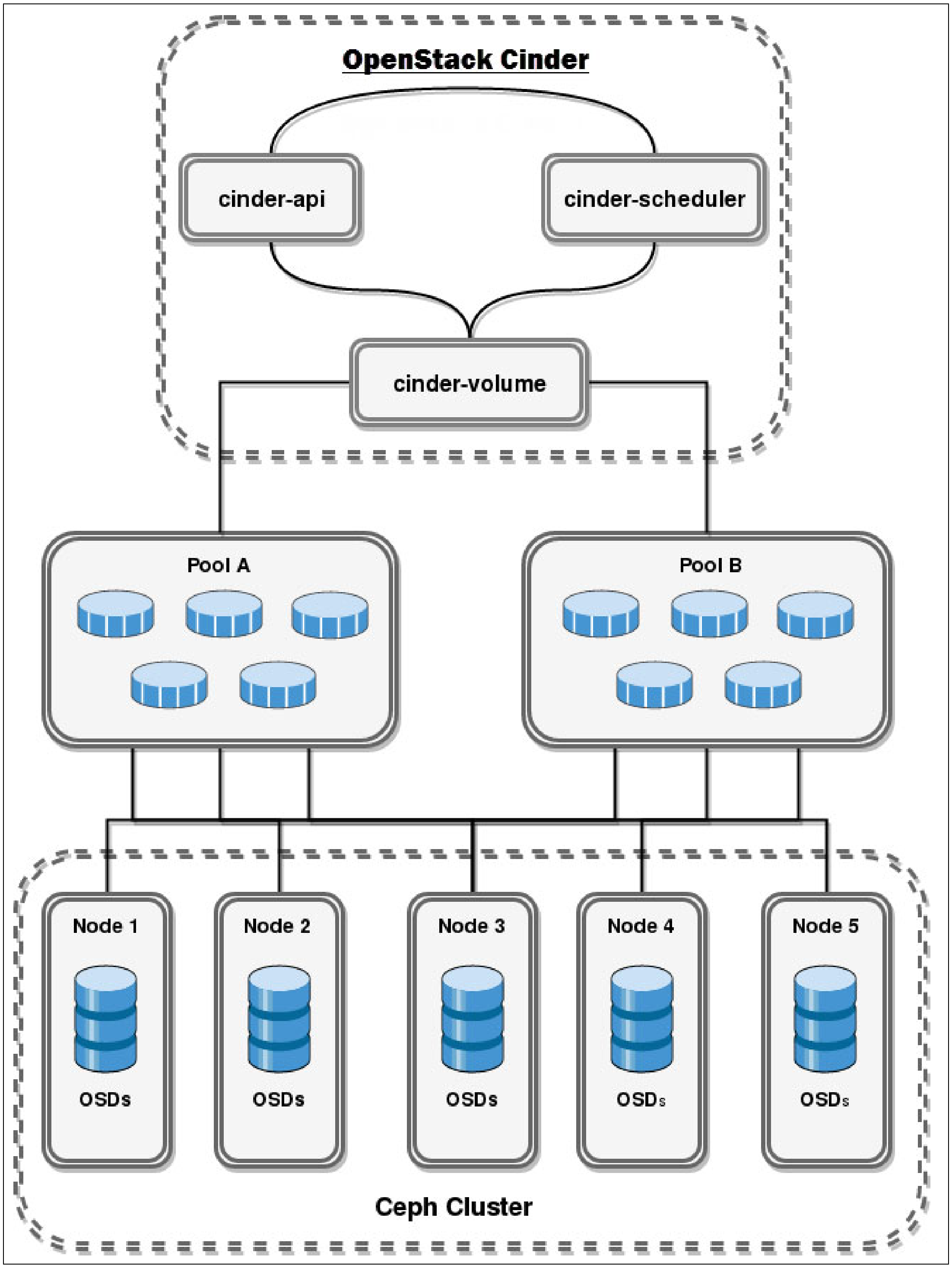


Figure 3-25 Ceph RBD

Figure 3-25 on page 87 also shows how Ceph RBD maps the RBDs that are assigned to a database instance. You can also run the verification check from the control node 2. The OSDs running on storage nodes can be verified as shown in Example 3-11.

Example 3-11 OSD running on storage nodes

```
root@int2-controller-2:~# ceph osd tree | more
ID WEIGHT   TYPE NAME               UP/DOWN REWEIGHT PRIMARY-AFFINITY
-1 65.02464 root default
-2 21.67488     host int2-cephosd-1
 0  1.44499         osd.0                up  1.00000          1.00000
 3  1.44499         osd.3                up  1.00000          1.00000
 8  1.44499         osd.8                up  1.00000          1.00000
10  1.44499         osd.10               up  1.00000          1.00000
14  1.44499         osd.14               up  1.00000          1.00000
16  1.44499         osd.16               up  1.00000          1.00000
19  1.44499         osd.19               up  1.00000          1.00000
23  1.44499         osd.23               up  1.00000          1.00000
26  1.44499         osd.26               up  1.00000          1.00000
29  1.44499         osd.29               up  1.00000          1.00000
31  1.44499         osd.31               up  1.00000          1.00000
35  1.44499         osd.35               up  1.00000          1.00000
38  1.44499         osd.38               up  1.00000          1.00000
41  1.44499         osd.41               up  1.00000          1.00000
43  1.44499         osd.43               up  1.00000          1.00000
-3 21.67488     host int2-cephosd-3
 1  1.44499         osd.1                up  1.00000          1.00000
 4  1.44499         osd.4                up  1.00000          1.00000
 6  1.44499         osd.6                up  1.00000          1.00000
 9  1.44499         osd.9                up  1.00000          1.00000
12  1.44499         osd.12               up  1.00000          1.00000
15  1.44499         osd.15               up  1.00000          1.00000
18  1.44499         osd.18               up  1.00000          1.00000
20  1.44499         osd.20               up  1.00000          1.00000
22  1.44499         osd.22               up  1.00000          1.00000
25  1.44499         osd.25               up  1.00000          1.00000
28  1.44499         osd.28               up  1.00000          1.00000
32  1.44499         osd.32               up  1.00000          1.00000
34  1.44499         osd.34               up  1.00000          1.00000
37  1.44499         osd.37               up  1.00000          1.00000
40  1.44499         osd.40               up  1.00000          1.00000
-4 21.67488     host int2-cephosd-2
 2  1.44499         osd.2                up  1.00000          1.00000
 5  1.44499         osd.5                up  1.00000          1.00000
 7  1.44499         osd.7                up  1.00000          1.00000
11  1.44499         osd.11               up  1.00000          1.00000
13  1.44499         osd.13               up  1.00000          1.00000
17  1.44499         osd.17               up  1.00000          1.00000
21  1.44499         osd.21               up  1.00000          1.00000
24  1.44499         osd.24               up  1.00000          1.00000
27  1.44499         osd.27               up  1.00000          1.00000
30  1.44499         osd.30               up  1.00000          1.00000
33  1.44499         osd.33               up  1.00000          1.00000
36  1.44499         osd.36               up  1.00000          1.00000
39  1.44499         osd.39               up  1.00000          1.00000
42  1.44499         osd.42               up  1.00000          1.00000
44  1.44499         osd.44               up  1.00000          1.00000
```

After verifying that the OSDs in the cluster nodes are running, verify the existing volumes in the cluster, as shown in Example 3-12.

Example 3-12 Current cluster volumes

```
root@int2-controller-2:~# ceph osd lspools
1 images,2 volumes,3 vms,
```

Example 3-12 shows that in the cluster there are three pools that are created: volume number one, which is named `images`, volume number two, which is named `volumes`, and volume number three, which is named `vms`. Now, block device images within the pool `volumes` can be listed, as shown in Example 3-13.

Example 3-13 Block device images

```
root@int2-controller-2:~# rbd ls volumes
volume-1cdb19e9-5e19-49bd-9ecd-a78ff2431efa
volume-265de7eb-fa83-462f-b770-c495484331bf
volume-2ff5f8b7-019e-4ff7-b1f8-71fce8f5835e
volume-44f48440-8103-45d8-a0ea-59d447d6b9bc
volume-4ec97781-7e4e-4ed6-9c84-19d6f1990e96
volume-79994568-4b13-4f68-a7e8-73d468f3cfd8
volume-7c6aaf59-393e-4ebc-9f21-0bf5d64b8083
volume-7f78052c-eddb-46a9-8ef8-dc93e2742dbb
volume-7fe29920-fa8a-4a9b-8d0d-0b3e588955a9
volume-81a3f7b9-23b6-4fc9-9c09-54cff0fe9507
volume-880c7142-6ddf-453f-9af9-128fcb739cc8
volume-9030f72b-d202-4550-9f58-8f02028920b5
volume-91b0df84-c3dd-4411-a7f2-95664af169c1
volume-a5f013c3-1618-47ea-b05e-3feaa64b78ca
volume-ac6b2923-b606-43c7-bb27-69603eb8df4b
volume-acaed8aa-8293-4ddd-8259-cefd9299a582
volume-b0b10093-1bc6-4ee8-88f1-8aeb528310e5
volume-c1d25369-6a7c-45a3-97f5-65cce484ac53
volume-c3429cac-fc0f-48e2-8610-1c350941c2b8
volume-e4fce2b2-0c85-454c-a234-c82750b211a2
volume-f1c79c88-100a-42a1-9d63-ac50487f45f9
volume-f1f0c355-0876-4b31-a3e9-47638974a847
volume-f8995313-df8a-4d11-a111-8622d21eb877
volume-fa9bdb39-1a25-48da-bd28-6105e681621d
volume-ff9fe7fe-d9bb-4522-9bb0-78c7da1c186f
```

From the list that is shown in Example 3-13 on page 89, you can retrieve the image `volume-9030f72b-d202-4550-9f58-8f02028920b5` to inspect important details, including its size (2048 MB (2 GB)) and the prefix, as shown in Example 3-14.

Example 3-14 Block device image details

```
root@int2-controller-2:~# rbd info
volumes/volume-9030f72b-d202-4550-9f58-8f02028920b5
rbd image 'volume-9030f72b-d202-4550-9f58-8f02028920b5':
   size 2048 MB in 512 objects
   order 22 (4096 kB objects)
   block_name_prefix: rbd_data.1f1994e7b4275
   format: 2
   features: layering, exclusive-lock, object-map, fast-diff, deep-flatten
   flags:
root@int2-controller-2:~#
```

With the image identifier and size, `volume-9030f72b-d202-4550-9f58-8f02028920b5` can be found in the Cinder block devices list. Figure 3-26 shows the block device identifier, size, and the attached compute instance.

```
root@int2-controller-2-utility-container-0e5ad74d:~# cinder list
+--------------------------------------+-----------+-----------------+------+-------------+----------+--------------------------------------+
| ID                                   | Status    | Name            | Size | Volume Type | Bootable | Attached to                          |
+--------------------------------------+-----------+-----------------+------+-------------+----------+--------------------------------------+
| 1cdb19e9-5e19-49bd-9ecd-a78ff2431efa | in-use    | -               | 1    | ceph        | false    | 4603375b-7f10-41f6-b04c-f370524675c1 |
| 44f48440-8103-45d8-a0ea-59d447d6b9bc | available | -               | 1    | ceph        | false    |                                      |
| 4ec97781-7e4e-4ed6-9c84-19d6f1990e96 | in-use    | -               | 1    | -           | false    | c73bee01-59e1-41b4-86f0-21c97c01a37c |
| 79994568-4b13-4f68-a7e8-73d468f3cfd8 | in-use    | -               | 1    | -           | false    | 51a7248c-aa5d-4ba5-864f-4d1129cdcc26 |
| 7c6aaf59-393e-4ebc-9f21-0bf5d64b8083 | in-use    | -               | 1    | -           | false    | 8042b093-e2fb-4c4c-a446-050ced004789 |
| 7f78052c-eddb-46a9-8ef8-dc93e2742dbb | available | -               | 1    | ceph        | false    |                                      |
| 7fe29920-fa8a-4a9b-8d0d-0b3e588955a9 | in-use    | -               | 1    | ceph        | false    | 4603375b-7f10-41f6-b04c-f370524675c1 |
| 81a3f7b9-23b6-4fc9-9c09-54cff0fe9507 | in-use    | -               | 2    | -           | false    | 0d78de71-bc30-4223-9e22-eeb7939e92b8 |
| 880c7142-6ddf-453f-9af9-128fcb739cc8 | in-use    | -               | 2    | -           | false    | dff488df-dbac-470e-979b-5ca0e90a2697 |
| 9030f72b-d202-4550-9f58-8f02028920b5 | in-use    | -               | 2    | -           | false    | 0d78de71-bc30-4223-9e22-eeb7939e92b8 |
| a5f013c3-1618-47ea-b05e-3feaa64b78ca | in-use    | -               | 1    | -           | false    | c73bee01-59e1-41b4-86f0-21c97c01a37c |
| ac6b2923-b606-43c7-bb27-69603eb8df4b | available | -               | 1    | ceph        | false    |                                      |
| acaed8aa-8293-4ddd-8259-cefd9299a582 | available | forcephperftest | 1    | ceph        | false    |                                      |
| b0b10093-1bc6-4ee8-88f1-8aeb528310e5 | in-use    | -               | 2    | -           | false    | dff488df-dbac-470e-979b-5ca0e90a2697 |
| c1d25369-6a7c-45a3-97f5-65cce484ac53 | available | -               | 1    | ceph        | false    |                                      |
| c3429cac-fc0f-48e2-8610-1c350941c2b8 | in-use    | -               | 1    | -           | false    | 51a7248c-aa5d-4ba5-864f-4d1129cdcc26 |
| e4fce2b2-0c85-454c-a234-c82750b211a2 | in-use    | -               | 1    | -           | false    | 8042b093-e2fb-4c4c-a446-050ced004789 |
| f1c79c88-100a-42a1-9d63-ac50487f45f9 | available | -               | 1    | ceph        | false    |                                      |
| f1f0c355-0876-4b31-a3e9-47638974a847 | in-use    |                 | 10   | -           | true     | dc82e1f8-cbea-43b0-a555-a6ba5d4c6440 |
| f8995313-df8a-4d11-a111-8622d21eb877 | available | -               | 1    | ceph        | false    |                                      |
| ff9fe7fe-d9bb-4522-9bb0-78c7da1c186f | in-use    |                 | 5    | -           | true     | ca68538d-4610-45ee-8c4c-352a043b96e0 |
+--------------------------------------+-----------+-----------------+------+-------------+----------+--------------------------------------+
root@int2-controller-2-utility-container-0e5ad74d:~#
```

Figure 3-26 Cinder block list

From Figure 3-26, the ID of the compute instance to which the image of the block device is attached can be obtained. Figure 3-27 shows, finally, which database is attached to the 2 GB block device.

```
root@int2-controller-2-utility-container-0e5ad74d:~# nova list
+--------------------------------------+---------------+--------+------------+-------------+---------------------------------------------------+
| ID                                   | Name          | Status | Task State | Power State | Networks                                          |
+--------------------------------------+---------------+--------+------------+-------------+---------------------------------------------------+
| 8042b093-e2fb-4c4c-a446-050ced004789 | MyInstance    | ACTIVE | -          | Running     | external=9.3.80.132; trove_infra_net=172.29.236.108 |
| 9e3ae09e-4676-41b8-a554-f9fd29a1a36e | demo-client   | ACTIVE | -          | Running     | external=9.3.80.136                               |
| 51a7248c-aa5d-4ba5-864f-4d1129cdcc26 | eric-backup   | ACTIVE | -          | Running     | external=9.3.80.131; trove_infra_net=172.29.236.116 |
| 0d78de71-bc30-4223-9e22-eeb7939e92b8 | fabio-mysql   | ACTIVE | -          | Running     | external=9.3.80.35; trove_infra_net=172.29.236.113 |
| ca68538d-4610-45ee-8c4c-352a043b96e0 | fabio-nova    | ACTIVE | -          | Running     | external=9.3.80.33                                |
| dff488df-dbac-470e-979b-5ca0e90a2697 | fabiodb       | ACTIVE | -          | Running     | external=9.3.80.40; trove_infra_net=172.29.236.109 |
| 9e1ec1e2-8c4e-49bc-aa53-b836f03e4c78 | kyle1         | ACTIVE | -          | Running     | external=9.3.80.137                               |
| dc82e1f8-cbea-43b0-a555-a6ba5d4c6440 | luke-deployer | ACTIVE | -          | Running     | external=9.3.80.129                               |
| 4603375b-7f10-41f6-b04c-f370524675c1 | luke-mariadb  | ACTIVE | -          | Running     | external=9.3.80.128; trove_infra_net=172.29.236.102 |
+--------------------------------------+---------------+--------+------------+-------------+---------------------------------------------------+
root@int2-controller-2-utility-container-0e5ad74d:~#
```

Figure 3-27 Database instances list

3.5.4 Object storage nodes

The object nodes together with their attached JBOD storage drawers provide the object storage services for backups based on OpenStack Swift, which is a highly scalable object storage system.

To store the database backups, the guest agent sends the requests to one or more Swift proxies, then the backups are stored as objects: the data itself and its metadata. These objects are allocated as binary files on the drives by using a path that contains its *partition*.

This partition is the collection of stored data that is formed by the *account database*, the *container database*, and *the object itself*. These three elements are the basis of the storage and retrieval of the data that is stored as objects:

Account database The SQLite database that stores a group of containers.

Container database The SQLite database within an account that logically stores and groups the objects.

Object This is where the data and its metadata are stored. Objects in different containers can have the same name.

Figure 3-28 shows the object partition scheme.

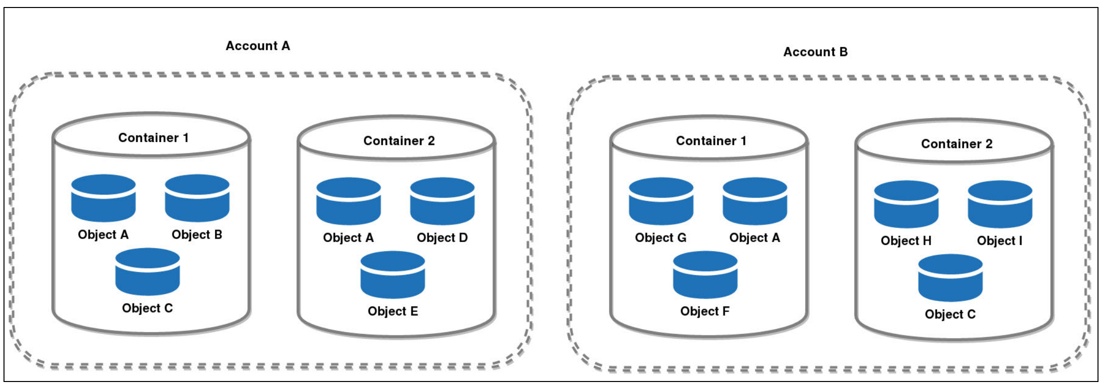


Figure 3-28 Swift partition scheme

To perform writing and reading operations on objects, the *object location* is used as shown in Example 3-15. This is the canonical name that is formed by the account database, the container database, and the object.

Example 3-15 The location for three objects

```
https://swiftobject-3/v1/accountA/container2/objectA
https://swiftobject-3/v1/accountA/container1/objectA
https://swiftobject-3/v1/accountB/container1/objectA
```

Swift offers great advantages for storing backups because the data is spread throughout the cluster nodes and written to multiple drives, placing three copies into the cluster: first by region, then by zone, and last by server and drive.

Thus, in case of a failure of a disk or the server, the backups are replicated to a new location in the cluster.

Here are the main advantages of Swift:

Scalability Swift provides linear scaling based on demand: the data to be stored and the number of requests to be served.

Reliability Swift ensures data availability by distributing data across the cluster, and data integrity by running an audit process that verifies data status.

Fault tolerance Swift can distribute the data in regions and zones to ensure that the replicas can be placed in the cluster in such a way that they can be fault tolerant.

High throughput Based on shared-nothing cluster, Swift uses all the available server capacity across the entire cluster, enabling multiple requests to be handled simultaneously.

Storage flexibility Swift offers a pluggable architecture that enables, through adapters and policies, different underlying storage systems, which can be adjusted according to particular needs.

Object cluster

A Swift object cluster consists of a group of nodes running a set of processes and services as a distributed storage system, which is organized in a scheme of regions and zones.

Nodes

Nodes are the physical servers that run Swift processes, as described in 3.3, “Infrastructure sizing” on page 61. Power System 8001-21C servers are the servers that are used as object nodes.

Each cluster node can run one or more of the following processes:

Proxy The process that communicates with external clients and handles the read and write requests, determining which node attends the corresponding request.

Account The process that provides the metadata for accounts and the containers within an account.

Container This manages the container metadata and the objects within each container.

Object This provides the Binary Large Object (BLOB) storage service to store and retrieve objects.

Figure 3-29 shows Swift services running on an object node.

```
root@swiftobject-2:~# ps aux --sort=uid | grep -v grep | grep swift | awk '{print $1, $2, $12}' | more
swift 4941 /openstack/venvs/swift-14.0.6/bin/swift-container-updater
swift 4942 /openstack/venvs/swift-14.0.6/bin/swift-object-replicator
swift 4944 /openstack/venvs/swift-14.0.6/bin/swift-account-replicator
swift 4947 /openstack/venvs/swift-14.0.6/bin/swift-object-server
swift 4949 /openstack/venvs/swift-14.0.6/bin/swift-container-auditor
swift 4954 /openstack/venvs/swift-14.0.6/bin/swift-container-reconciler
swift 4955 /openstack/venvs/swift-14.0.6/bin/swift-object-expirer
swift 4959 /openstack/venvs/swift-14.0.6/bin/swift-container-replicator
swift 4964 /openstack/venvs/swift-14.0.6/bin/swift-account-auditor
swift 4971 /openstack/venvs/swift-14.0.6/bin/swift-container-sync
swift 4972 /openstack/venvs/swift-14.0.6/bin/swift-object-updater
swift 4975 /openstack/venvs/swift-14.0.6/bin/swift-account-reaper
swift 4987 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 4989 /openstack/venvs/swift-14.0.6/bin/swift-object-auditor
swift 5000 /openstack/venvs/swift-14.0.6/bin/swift-container-server
swift 5001 /openstack/venvs/swift-14.0.6/bin/swift-object-reconstructor
swift 5236 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5237 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5238 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5239 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5240 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5241 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5242 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5243 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5244 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5245 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5246 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5247 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5248 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5249 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5250 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5251 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5252 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5253 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5254 /openstack/venvs/swift-14.0.6/bin/swift-account-server
swift 5255 /openstack/venvs/swift-14.0.6/bin/swift-account-server
```

Figure 3-29 Swift services on an object node

Regions

The regions define a set of nodes in the cluster that are physically separated. Typically, this physical separation occurs in terms of different geographic locations, for example, two data centers in different cities. Clusters must have at least one region (single-region; when the cluster has two or more regions, the cluster is multi-regional.

In multi-regional clusters, the read requests are handled based on the measurement of the latency to determine which of the regions is closer to attend the request; this is known as *read affinity*.

For the writing process, the latency level determines two ways of writing the data. For low-latency connections, data is written simultaneously to different locations regardless of the region.

For high-latency connections, each write request creates a number of local copies and then asynchronously sends them to other regions; this is known as *write affinity*.

> **Note:** Simultaneously writing to multiple locations is the default write process in multi-regions cluster.

Zones

The zones define availability domains that allow the isolation of faults within the same region. These domains must be defined by a set of physical hardware that in case of failure, for example hardware, does not affect hardware in other zones. The zones are chosen based on particular needs.

For example, a zone can be defined by a top-of-rack (ToR) switch that gives connectivity to a certain rack. If the ToR switch fails, only that rack can be disconnected. A second zone can be determined by a power supply unit that feeds a group of racks, if the power supply unit fails, just that particular group of racks is affected.

Figure 3-30 shows an object cluster organizational scheme.

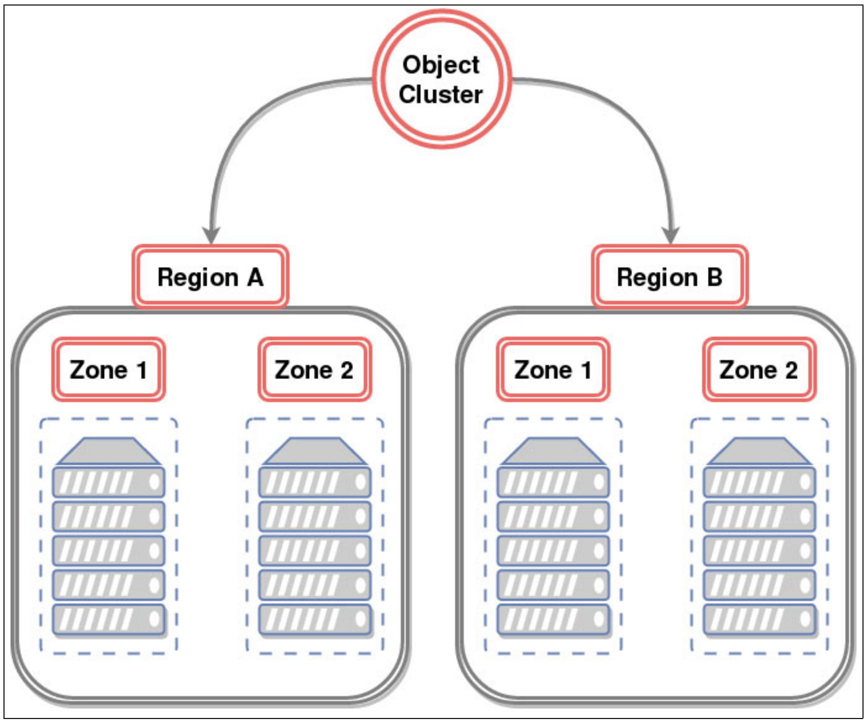


Figure 3-30 Object cluster organization

Storage policies

This is a way of defining segmentation levels within the cluster to provide differentiated services to meet specific storage needs.

Such segmentation can be as follows:

- At the hardware layer: For example, to define specific copies to be stored in SSDs only or policies that are applied to partitions
- At the server process layer: For example, to define that certain requests are served by a specific proxy server
- At the cluster layer: For example, to distribute certain replicas to a specific region or a specific group of regions

Figure 3-31 shows how storage policies can be applied to hardware resources, to the objects, to processes and services, and to the cluster environment, so that particular storage needs can be met.

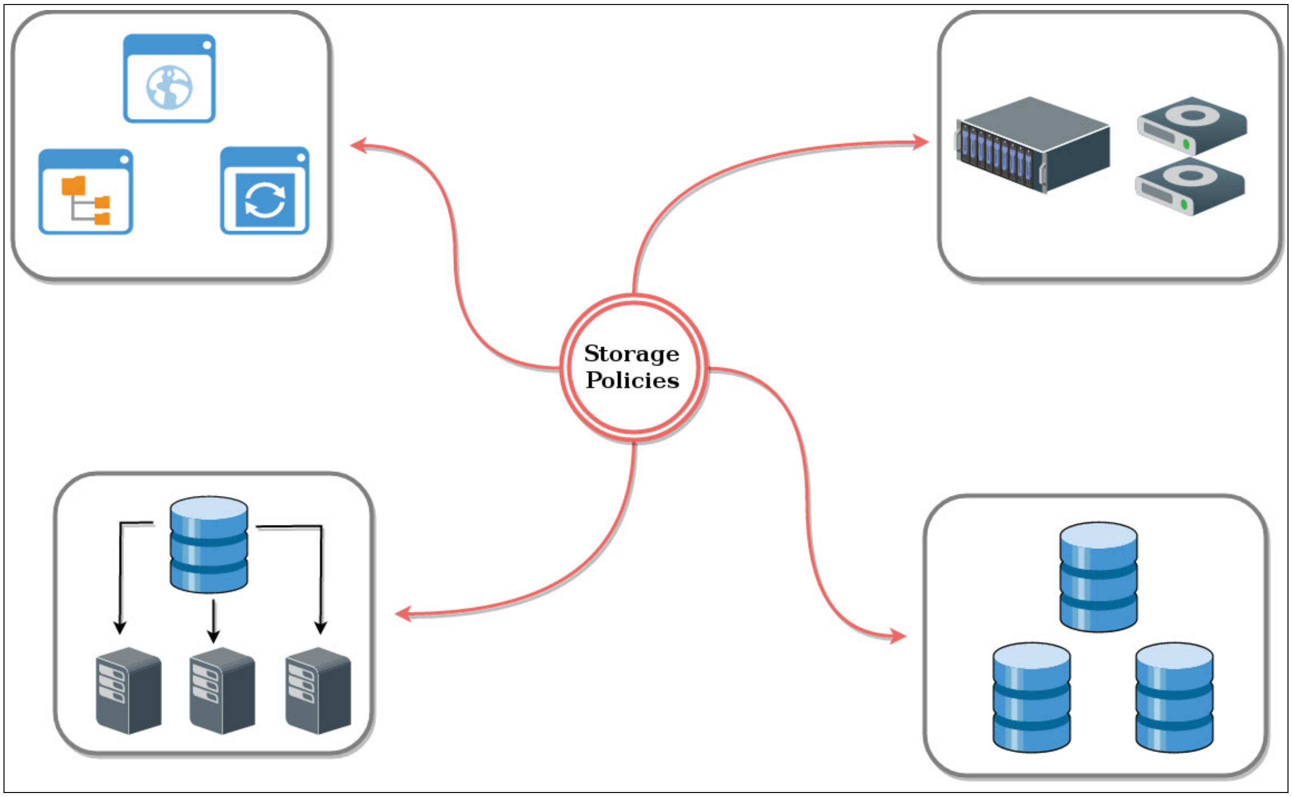


Figure 3-31 Storage policies

4

Usage

This chapter describes how to use and manage the Open Platform for Database as a Service (DBaaS) on IBM Power Systems solution. This chapter describes how to use the Dashboard interface to perform common operations for creating and managing databases in the cloud.

You can also use the Trove command-line API for any of these tasks.

After reading this chapter, you can perform the following tasks:

- Getting and building guest instances
- Launching, restarting, resizing, and renaming instances
- Resizing volumes
- Managing backups and recovery
- Creating and deleting users
- Managing user access

This chapter contains the following sections:

- Get and build images
- Deploying and maintaining instances
- Backup and recovery
- Security

4.1 Get and build images

This section provides a quick guide about building images for the Open Platform for DBaaS on Power Systems solution by using the dibimage-builder charms bundle. The image building process is described in more detail in Appendix A, “Servers provisioning and deployment” on page 201.

Here are the high-level steps to build a data store image:

1. Open the Juju GUI on your browser. For example:

   `https://<Juju server IP>:17070/gui`

2. Download the dbimages-builder charms.
3. Import the dbimage-builder charms as a local bundle. Add the bundle to your selected model.
4. Commit your changes and deploy.

> **Note:** The dbimage-builder charms bundle connects to the OpenStack Nova compute service and launches an instance in the cloud to build the target image.

After the bundle is deployed, the data store image becomes automatically available in the OpenStack Glance image service for launching new guest instances.

4.2 Deploying and maintaining instances

Deployment and maintenance tasks of database guest instances are performed through the Life-cycle Management actions frame under the Shortcuts menu, as shown in Figure 4-1 on page 99. You can manage instances, volumes, and databases by clicking the icons that are available in the frame. The capabilities include the following ones:

- Launch instance
- Restart instance
- Resize instance
- Rename instance
- Delete instance
- Resize volume
- Create database
- Delete database

These lifecycle management tasks are described in more detail in the following sections.

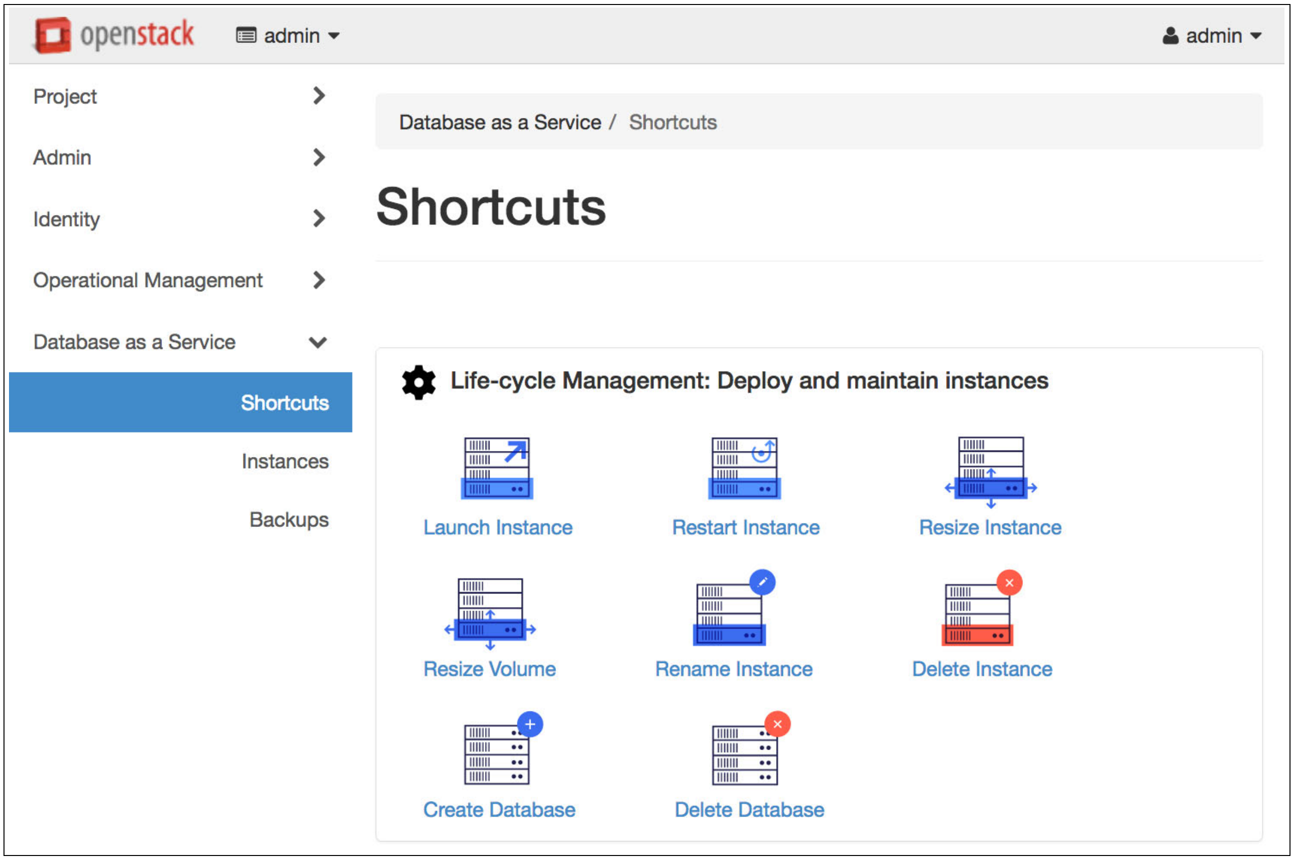


Figure 4-1 Life-cycle Management tasks

4.2.1 Launching an instance

To start using the databases that are available in the Open Platform for DBaaS on Power Systems solution, you must launch a guest instance with the correct data store image. When launching guest instances, the Open Platform for DBaaS on Power Systems solution creates a virtual machine (VM) in the target compute node with the database that you specify. The data store image has the database that was previously installed and configured for an operating system (OS). To read more about data store images, see 4.1, “Get and build images” on page 98.

Complete the following steps:

1. To launch a database guest instance in the cloud, go to the **Shortcuts** menu in the left pane and click the **Launch Instance** icon, as highlighted in Figure 4-2. Alternatively, instead of using the Shortcuts menu, you can launch instances by clicking **Launch Instance** in the **Instances** menu, as shown in Figure 4-3.

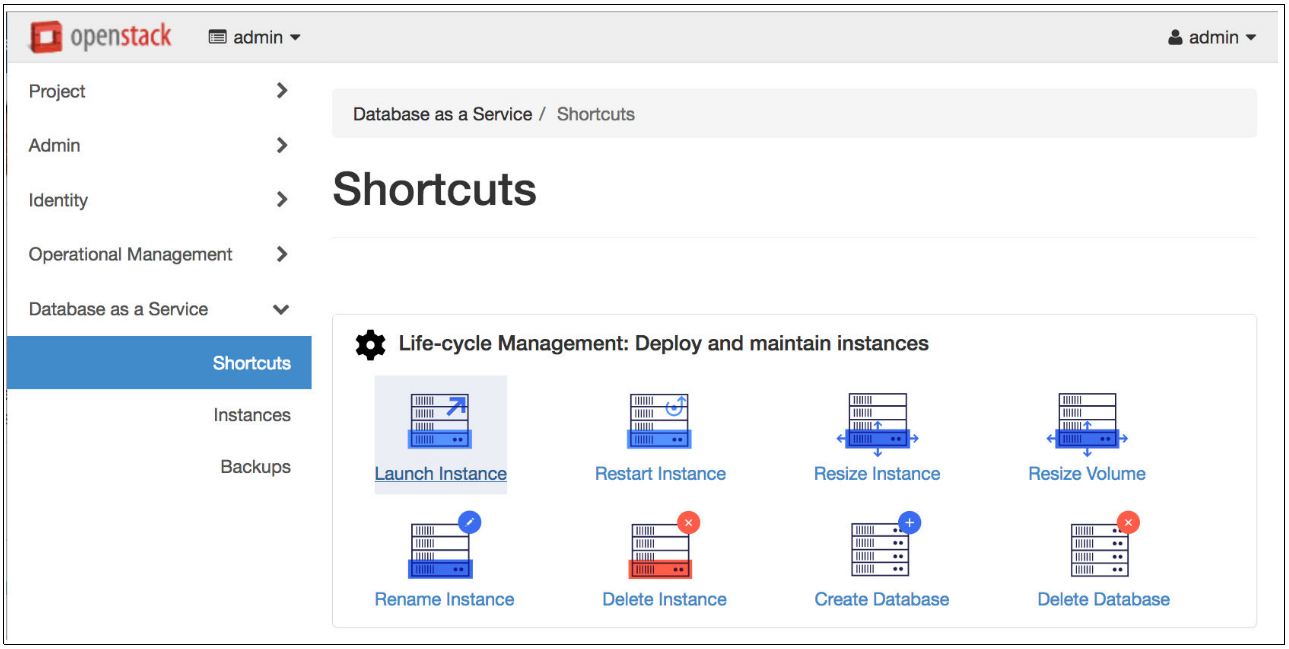


Figure 4-2 Launching an instance from the Shortcuts menu

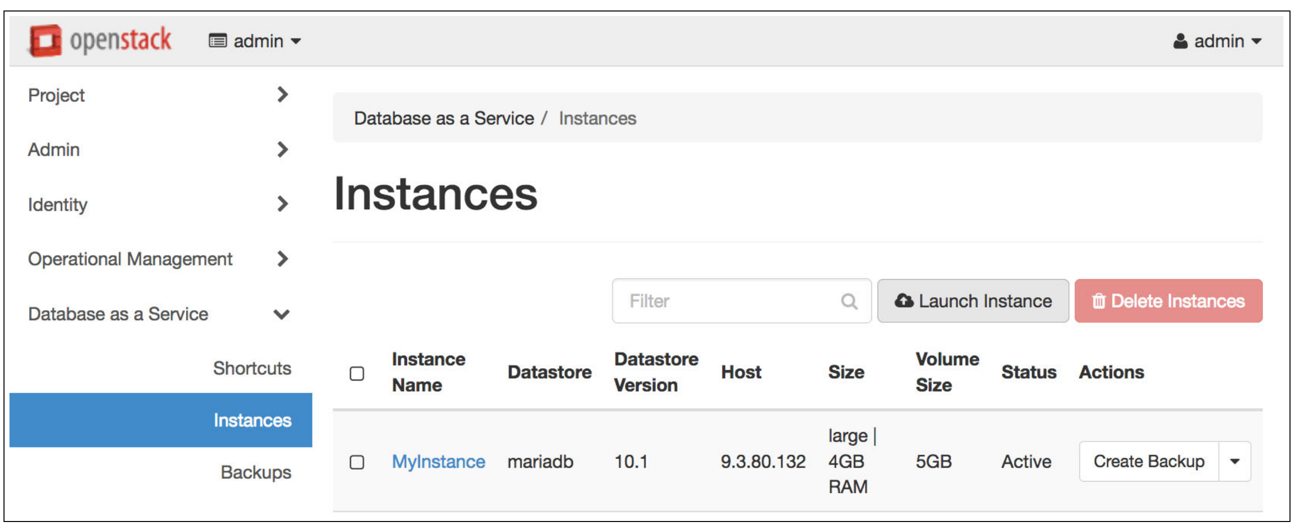


Figure 4-3 Displaying the launched instances

You must provide the requested input for launching the instance, as illustrated in Figure 4-4 on page 101. Under the Details tab, all supported data stores are pre-populated and listed in a drop-down menu. Additional data stores can also be built and uploaded to the OpenStack Glance image service. To read more about the image building process, see 4.1, “Get and build images” on page 98.

2. Type an instance name and volume size of your preference. Choose the data store and flavor size for the instance. In this example, flavors Medium and Large are available to use. To see the configuration settings for these flavors, go to the Flavors menu in the left pane and click the flavor that you want.

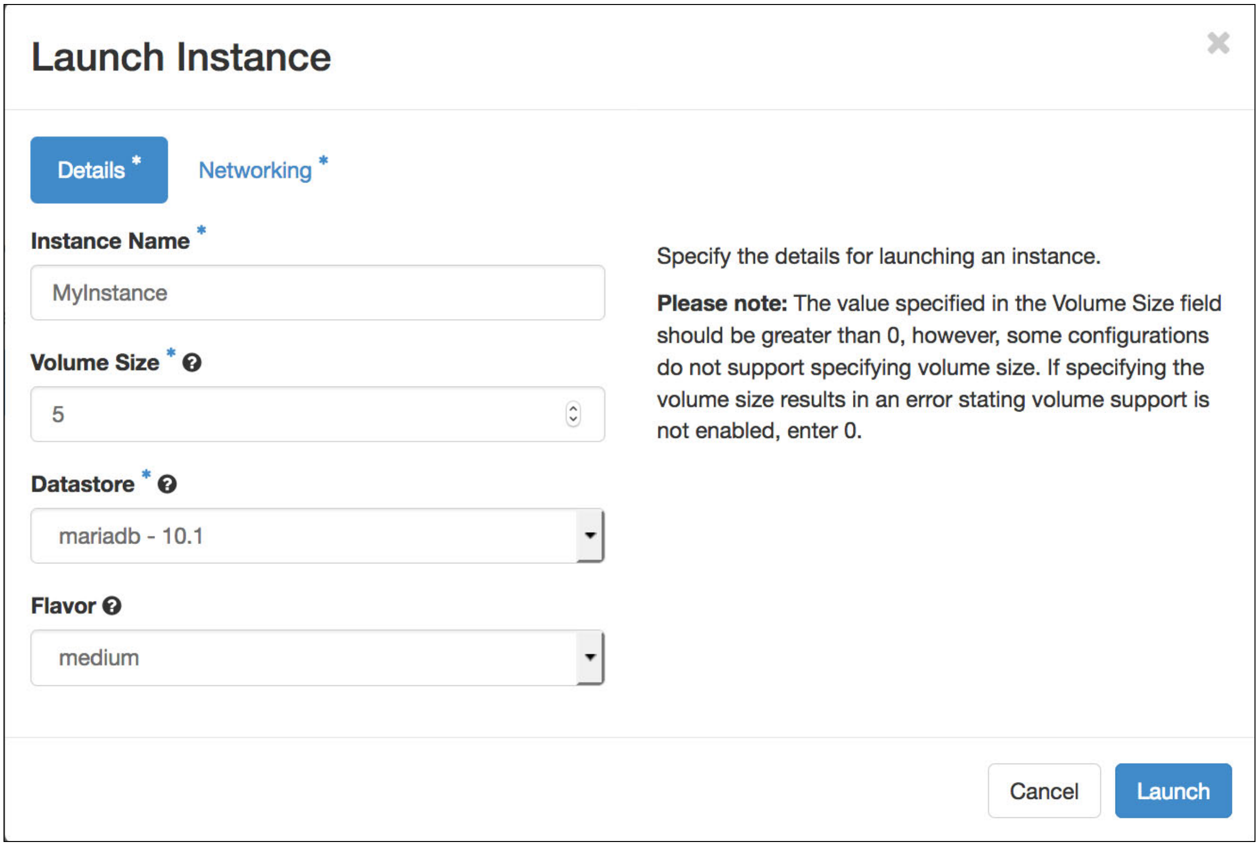


Figure 4-4 Launching an instance

In the Networking tab, you can specify the instance's connectivity, that is, how the instance interacts with the external network and how it can be accessed, which grants external access to the instance, as shown in Figure 4-5. You can also make your instance connect to a particular VLAN to isolate it and secure your data traffic in the network. For more information about networking considerations, see 3.4, "Networking" on page 68.

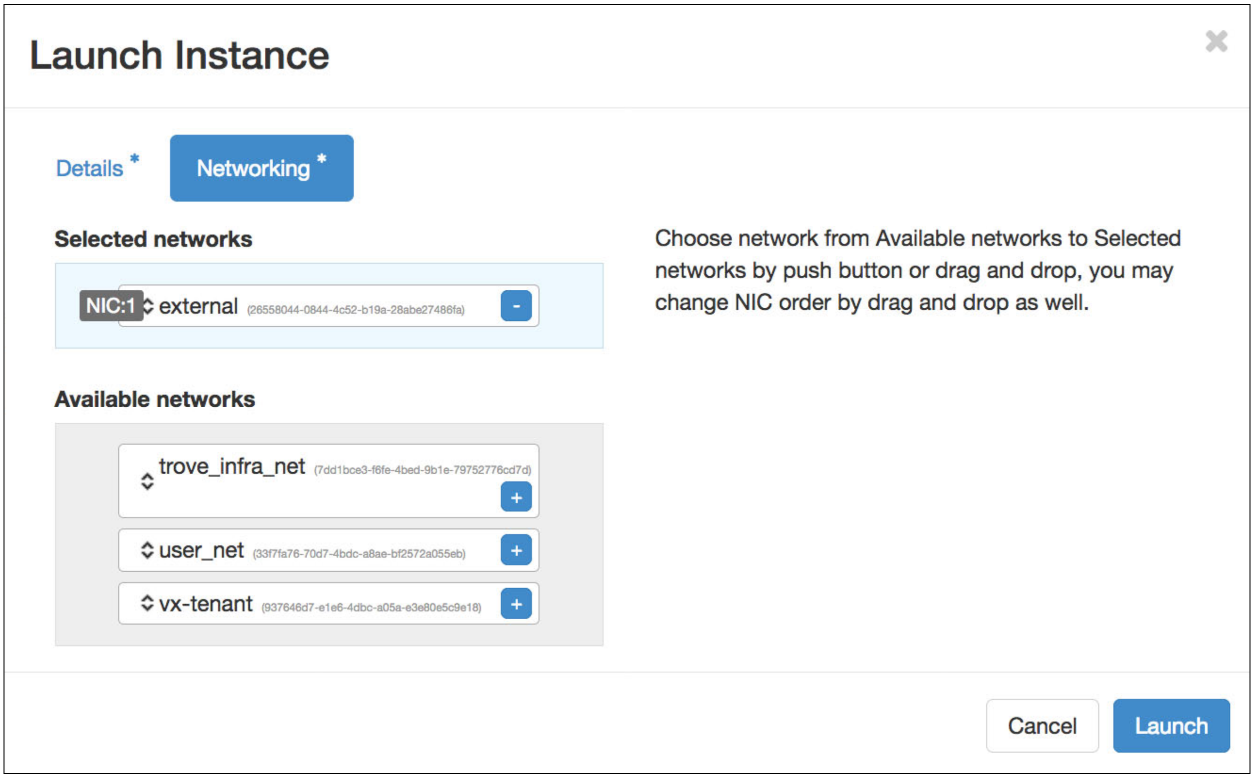


Figure 4-5 Networking settings for launching an instance

3. After you finish selecting instance options, click **Launch**. The instance launches and changes its status to Building. The instance launch takes a few seconds and becomes available for connecting and using. The instance status then changes from Building to Active. The launch progress can be viewed in the Instances menu, as shown in Figure 4-6 on page 103.

4.2.2 Checking the instance information

To check information from an instance, go to the Instances menu from the left pane. If you have many instances that are launched, you can filter your instance by typing its name in the search field. To see all the detailed information for your instance, click the link with the instance name, as illustrated in Figure 4-6.

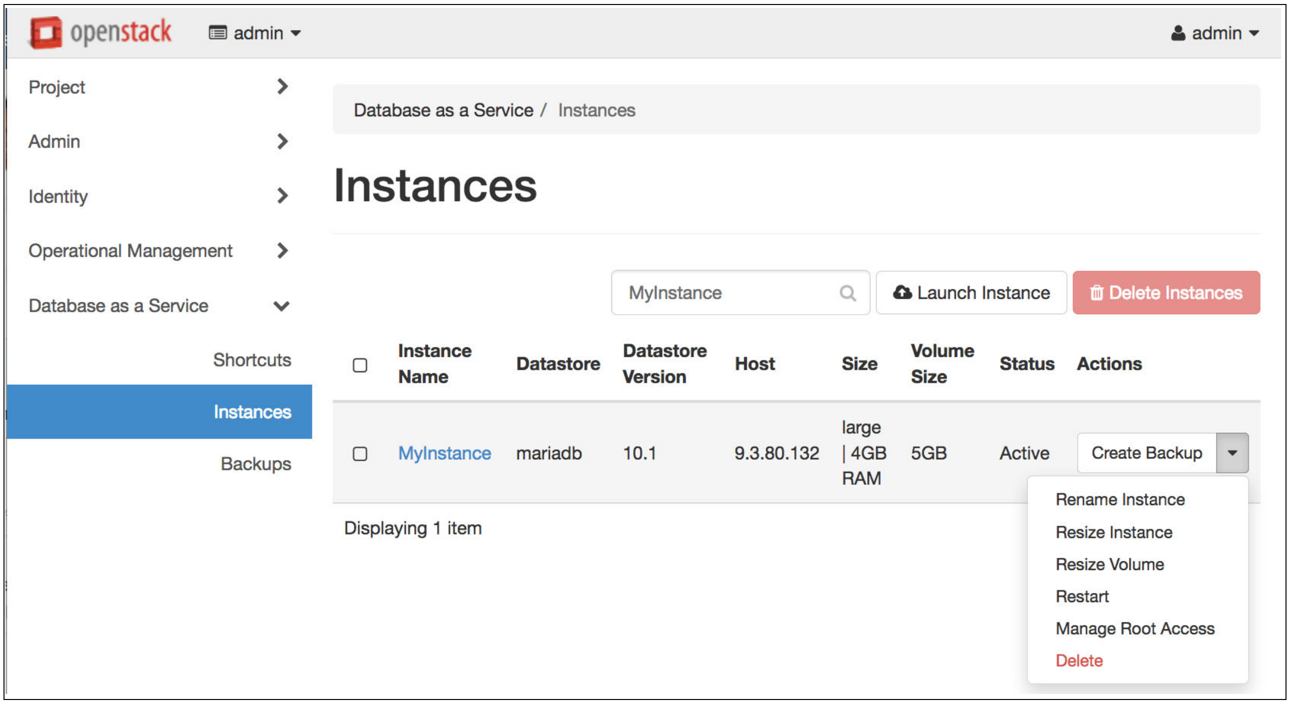


Figure 4-6 Instance launch progress view

Figure 4-7 shows an example of the detailed information that is retrieved from an instance, including instance specifications and connection information.

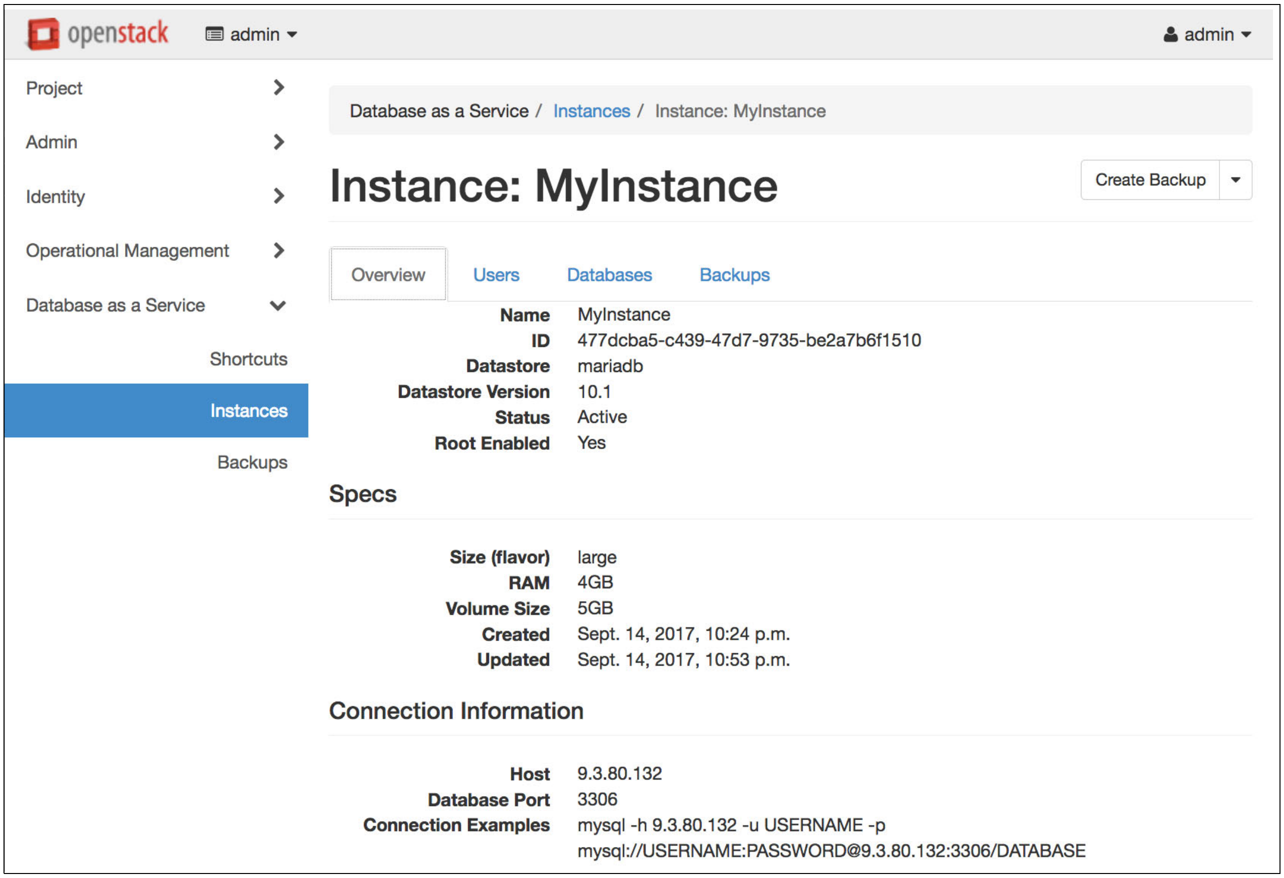


Figure 4-7 Checking the instance information

Connecting to the database

After the instance is running, you can get access to the database by using the connection information that is provided, such as the host and database port. To connect to a database, use the client utility for the database you are connecting to and specify the user and password.

Note: You must create a user and grant access to the user before connecting to the database. For more information, see 4.4, “Security” on page 117.

A connection example to a MySQL database as an admin user is shown in Example 4-1.

Example 4-1 Connecting to a database

```
$ mysql -h 9.3.80.132 -u admin -p
password:
mysql>
```

4.2.3 Maintaining an instance

The usual operation of the Open Platform for DBaaS on Power Systems solution does not require manual access to the instance through the SSH port. However, you can perform some maintenance or check something in the system.

By default, SSH and ICMP protocols are disabled in the security policies that are created for the launched database instances. To log in to the instance, you can add an SSH port to the existing security group or create an SSH policy to the instance, as shown in Figure 4-8. The SSH-ping policy in this example is a security group that is created to enable network traffic on TCP/IP protocol port 22 and ICMP protocol, enabling you to SSH and ping the instance.

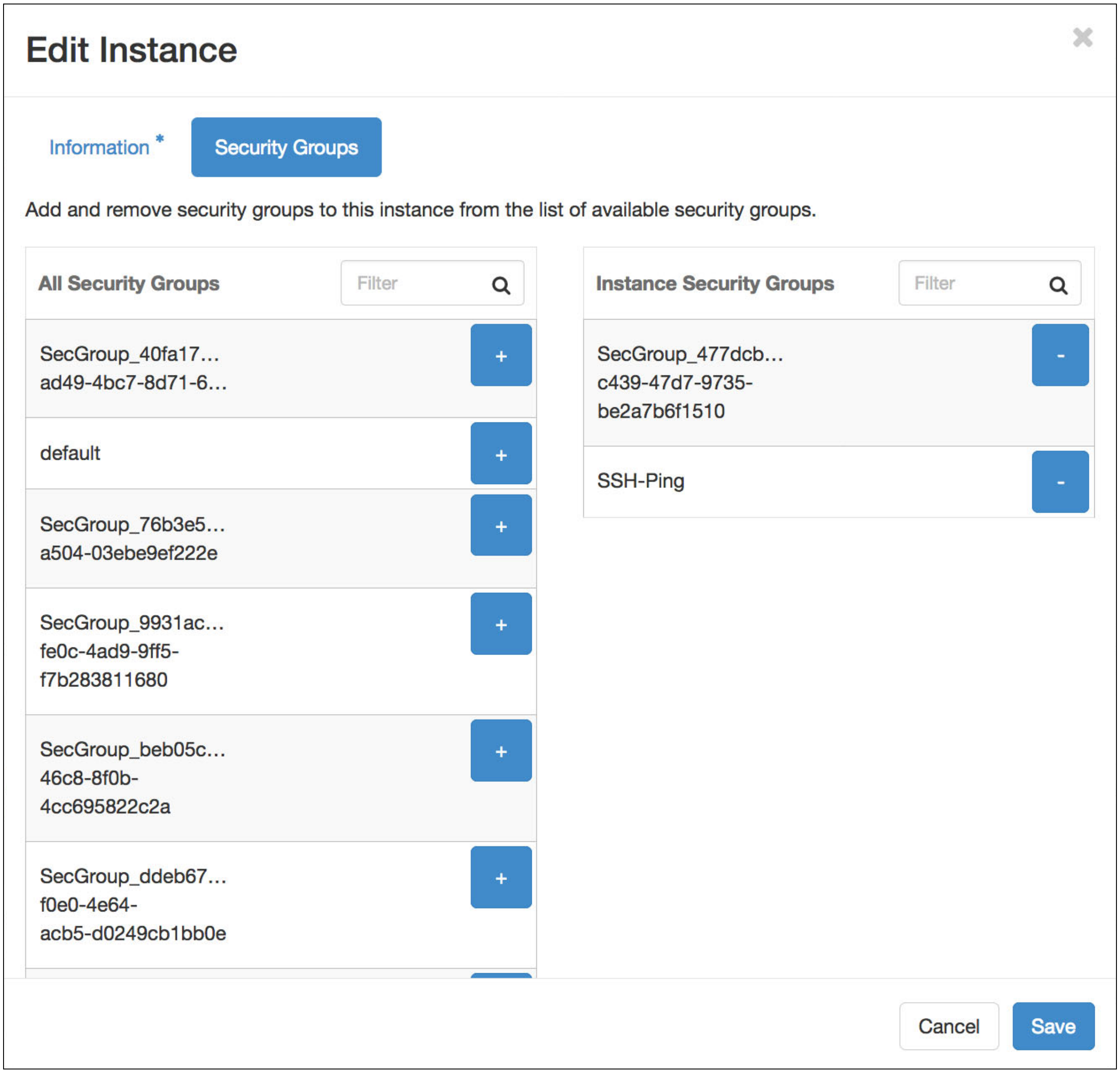


Figure 4-8 Adding a SSH-ping security group to an instance

OpenStack Trove, as opposed to the standard instance creating on Nova compute service, does not inject the keypair for logging in to the instance. The Open Platform for DBaaS on Power Systems solution injects the keypair into the instance in the image building stage. You can log in to the instance by using the public SSH key that is associated to the keypair that is used in the image building stage.

Note: Given the nature of Trove, the keypair is injected during the image building stage and the SSH key is associated to the data store image, not the instance.

For more information about keypair injection on data store images, see “Image building” on page 212.

4.2.4 Restarting an instance

To restart an instance, click the **Restart Instance** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Select the instance and click **Restart**, as illustrated in Figure 4-9. The instance status can be checked in the Instances menu under Database as Service pane (Figure 4-7 on page 104).

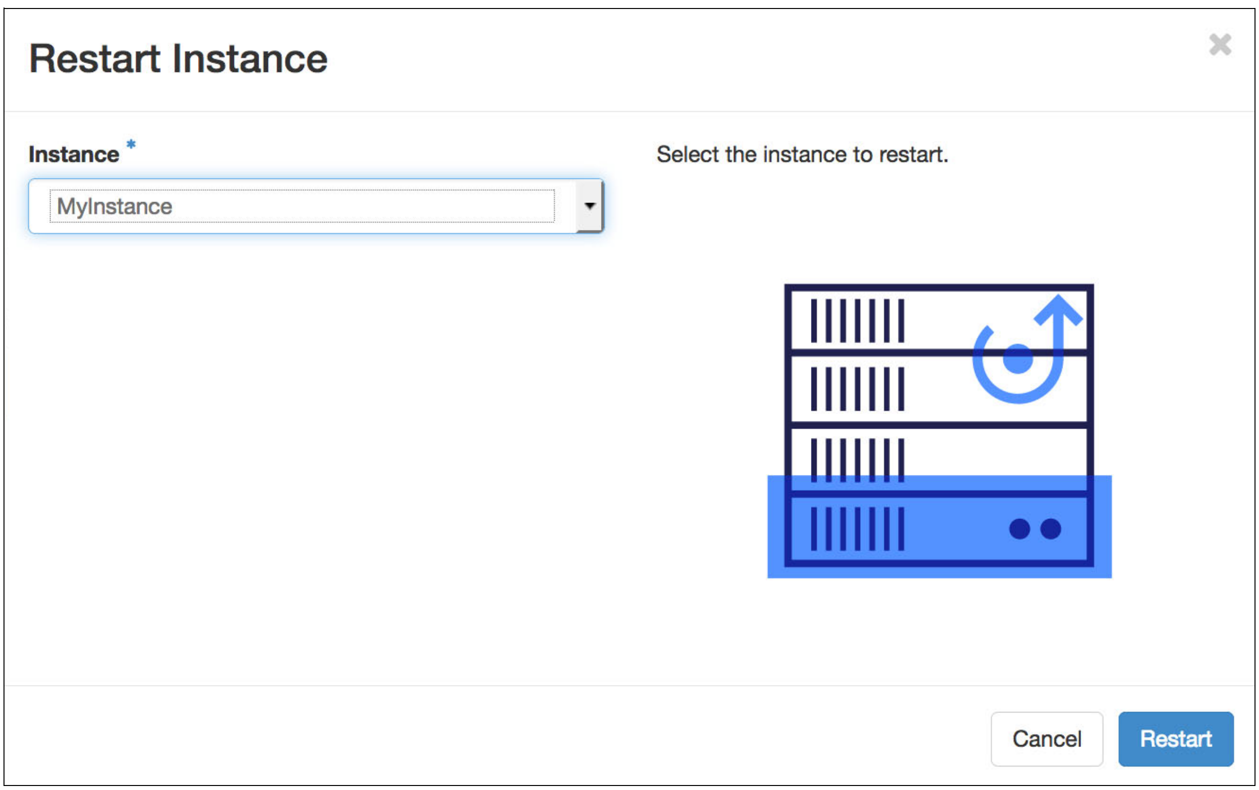


Figure 4-9 Restarting an instance

4.2.5 Resizing an instance

To resize an instance, click the **Resize Instance** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Select the instance and the new flavor. Then, click **Resize**, as illustrated by Figure 4-10.

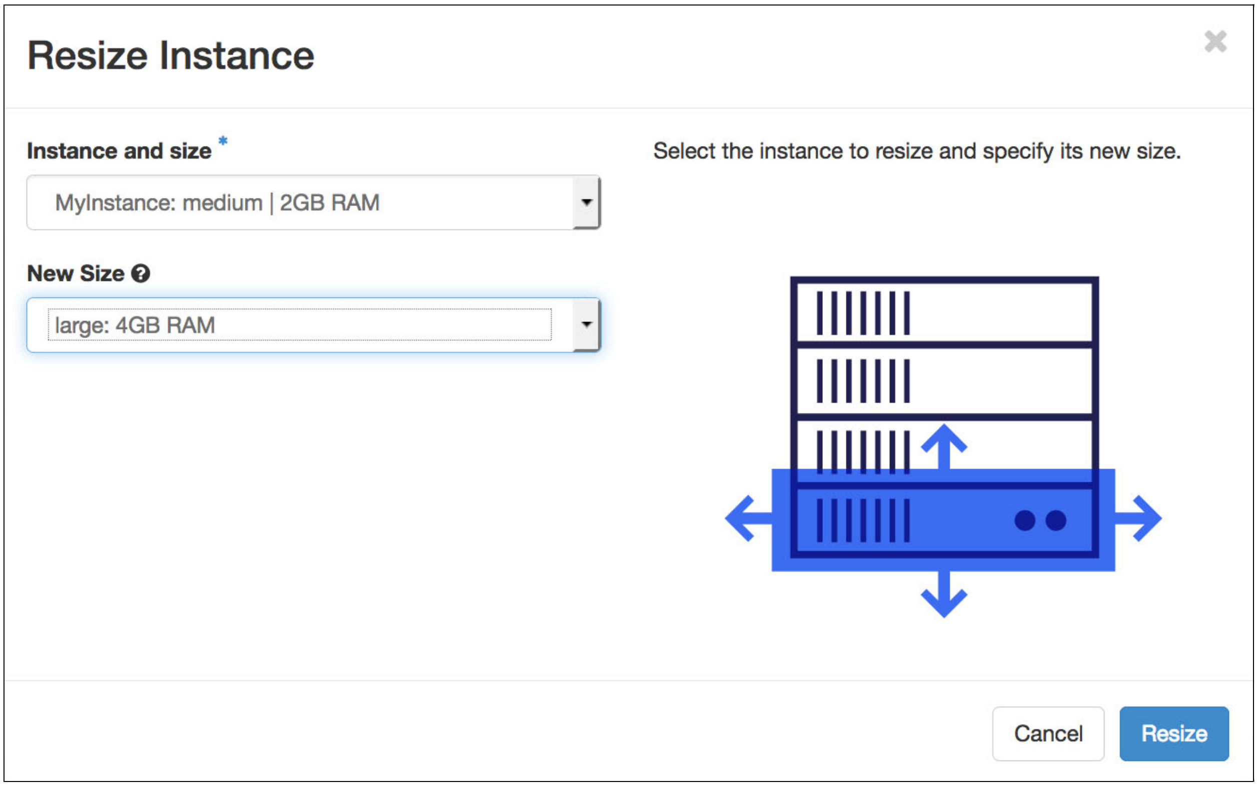


Figure 4-10 Resizing an instance

4.2.6 Deleting an instance

To delete an instance, click the **Delete Instance** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Select the instance and click **Delete**, as illustrated in Figure 4-11.

> **Note:** First, check that you have a backup of your data before deleting an instance because the instance data is destroyed.

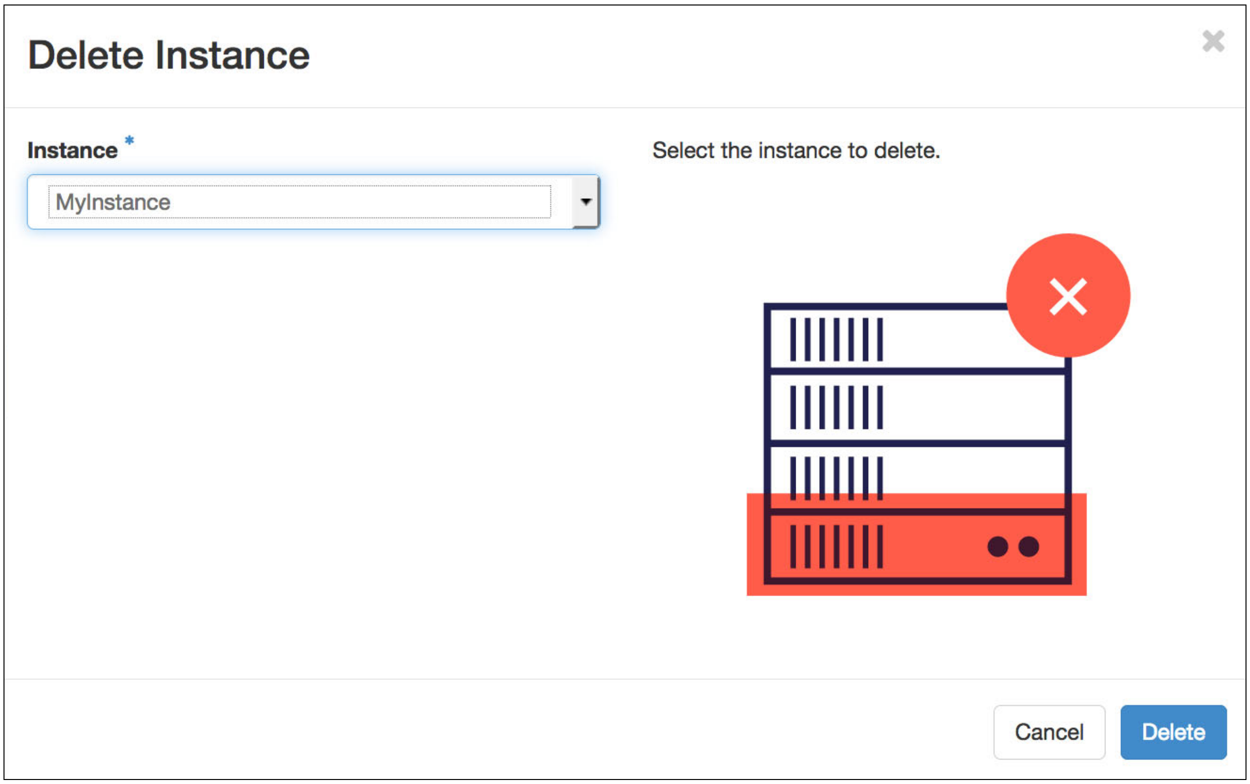


Figure 4-11 Deleting an instance

4.2.7 Resizing a volume

To resize a volume, click the **Resize Volume** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Select the instance and the new flavor. Then, click **Resize**, as illustrated in Figure 4-12.

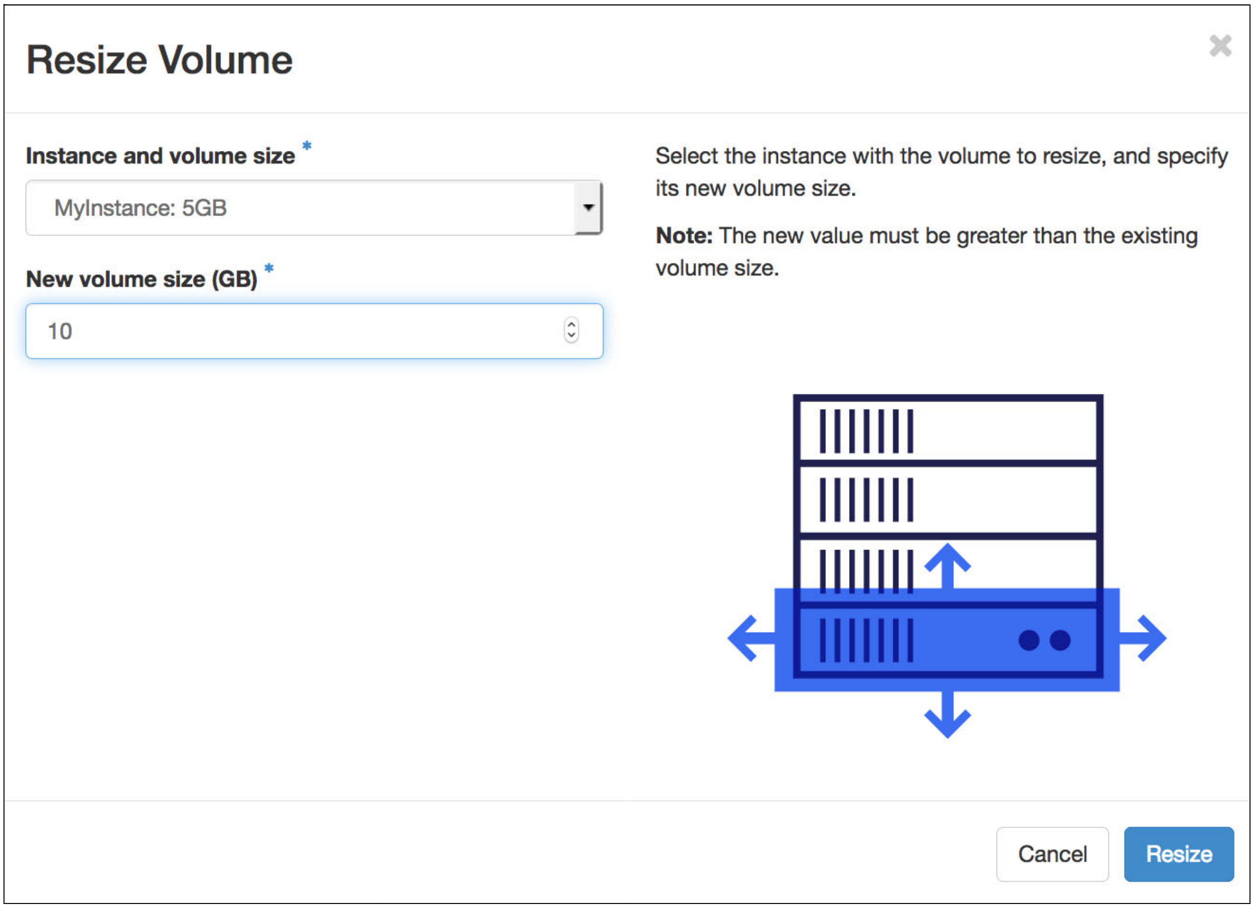


Figure 4-12 Resizing an instance

4.2.8 Renaming an instance

To rename an instance, click the **Rename Instance** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Select the instance that you want to rename. Then, type the new instance name in the field, as indicated in Figure 4-13. Click **Rename** to confirm.

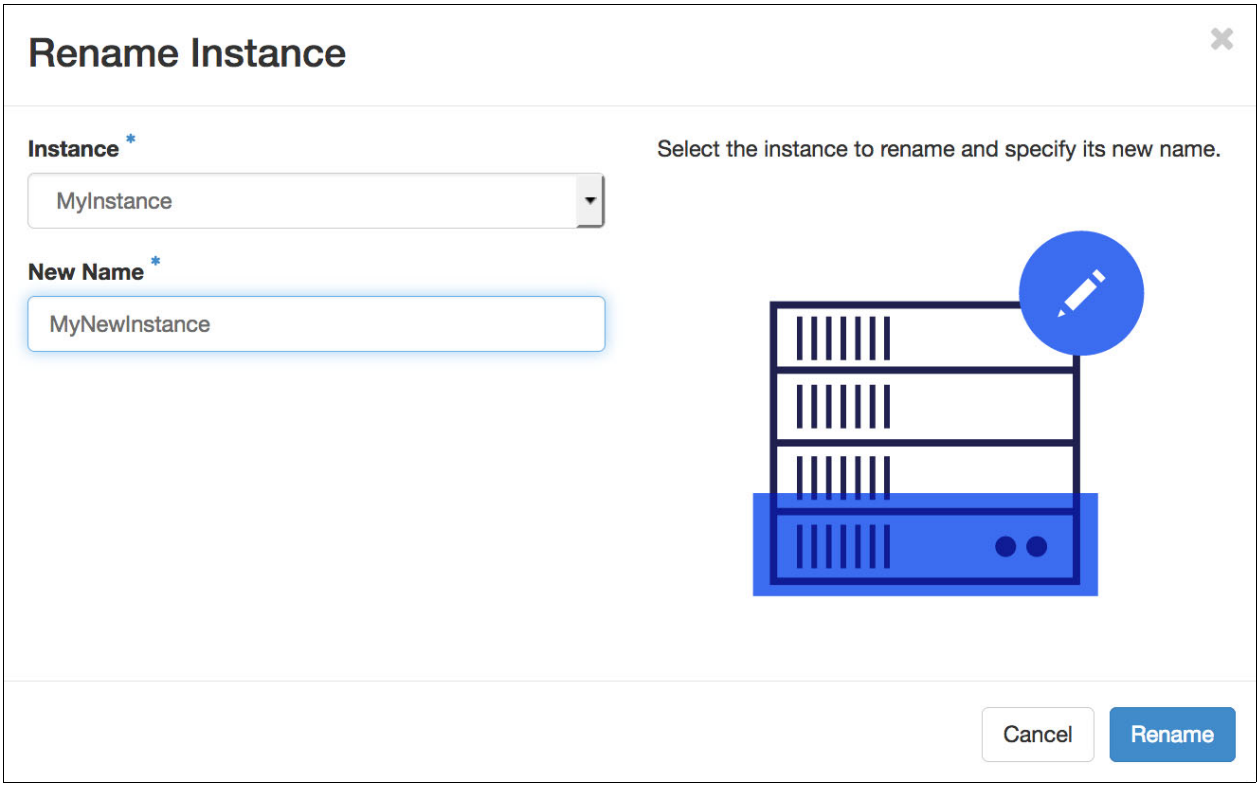


Figure 4-13 Renaming an instance

4.2.9 Creating a database

To create a database, click the **Create Database** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Type a name for the database to be created and select an instance from the list. Then, click **Create**, as illustrated in Figure 4-14.

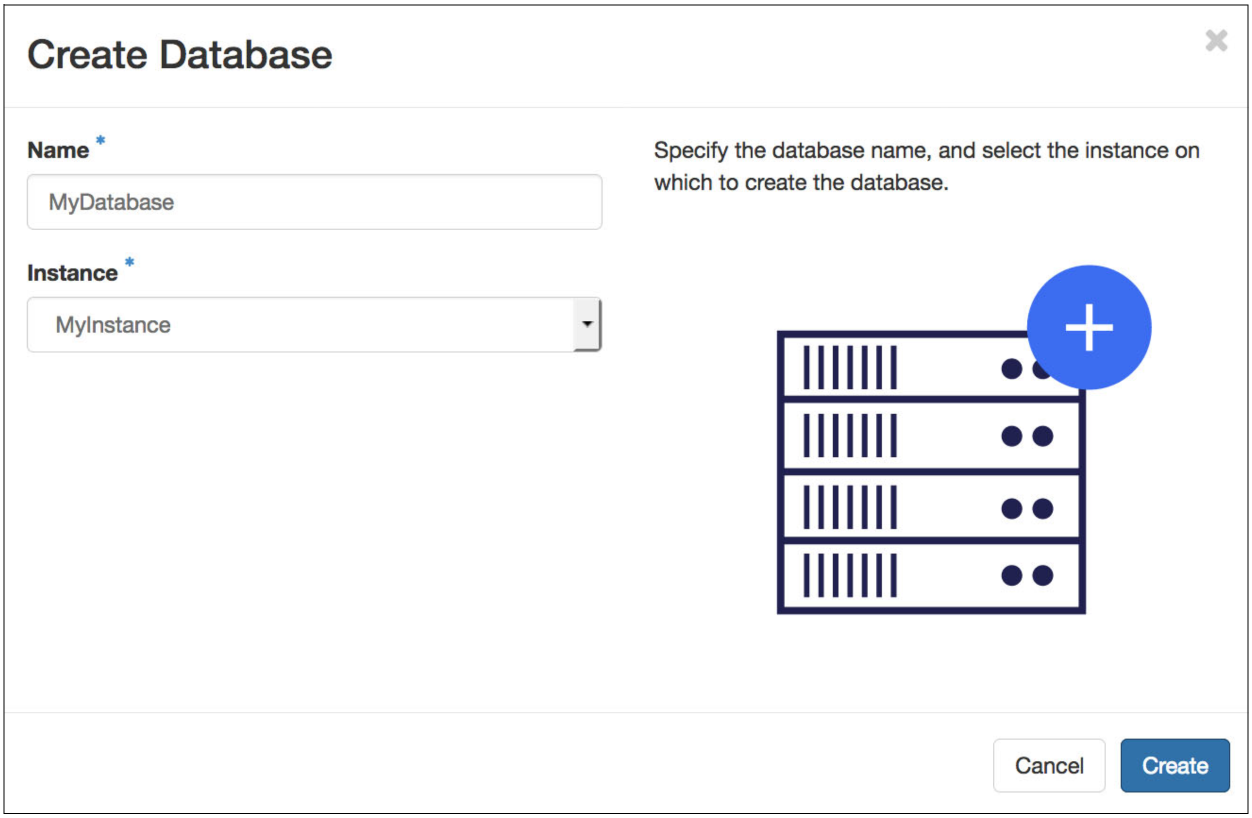


Figure 4-14 Creating a database

4.2.10 Deleting a database

To delete a database, click the **Delete Database** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Select the database to be deleted and then click **Delete**, as illustrated by Figure 4-15.

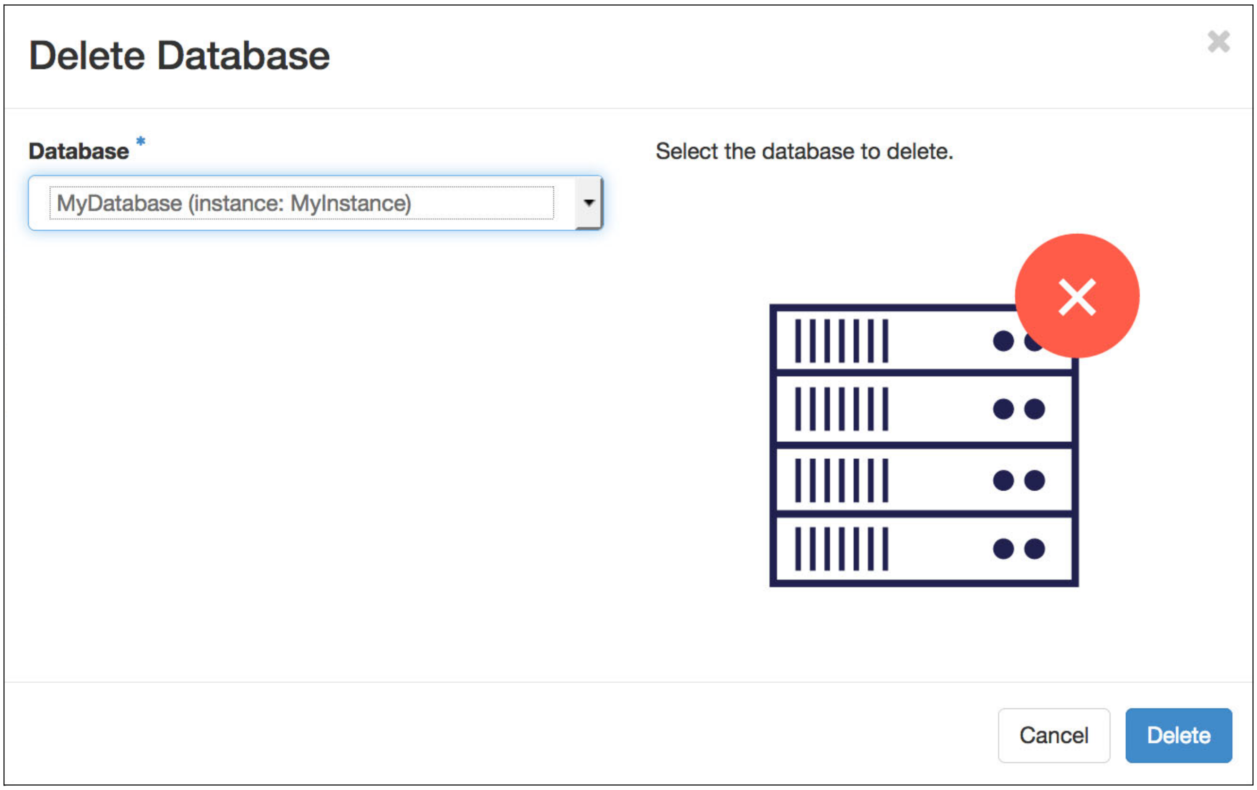


Figure 4-15 Deleting a database

4.3 Backup and recovery

The Open Platform for DBaaS on Power Systems solution creates backups through the OpenStack Swift object storage containers. Swift is ideal for backups because it stores data efficiently and safely. For more information about OpenStack Swift, see Welcome to Swift's documentation.

Backups can be managed from the Backups menu or from the Shortcuts menu. To list and manage backups, click **Backups** under the **Database as a Service** menu, as illustrated in Figure 4-16.

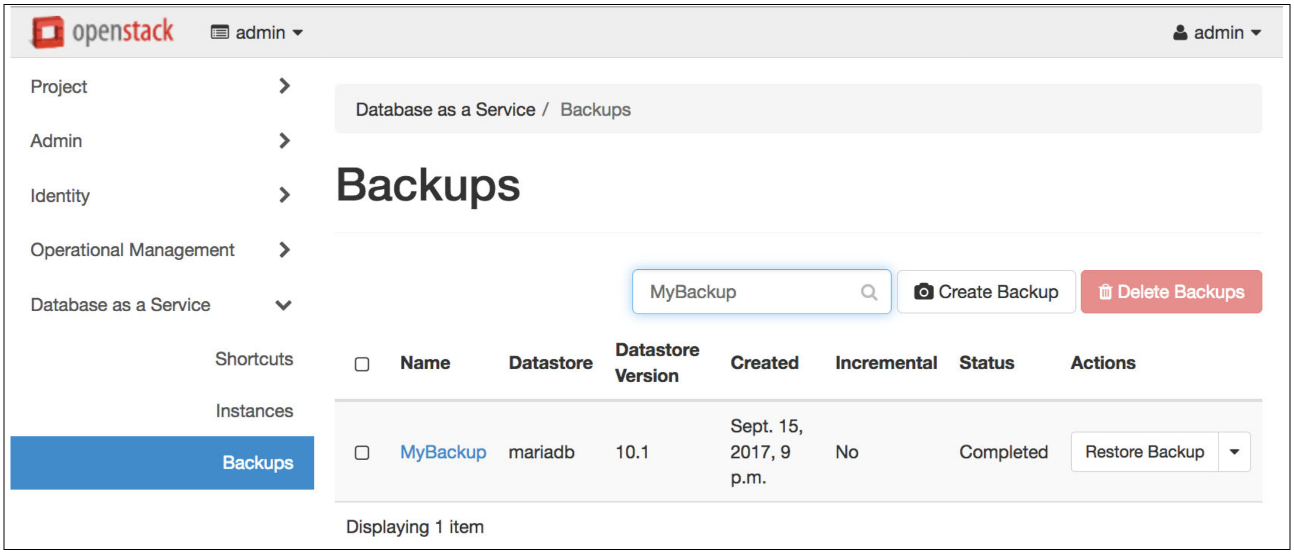


Figure 4-16 Managing backups

You can also show the backup details by clicking the backup name. The backup information for the MariaDB data store with an ID ending in f8bf is illustrated in Figure 4-17.

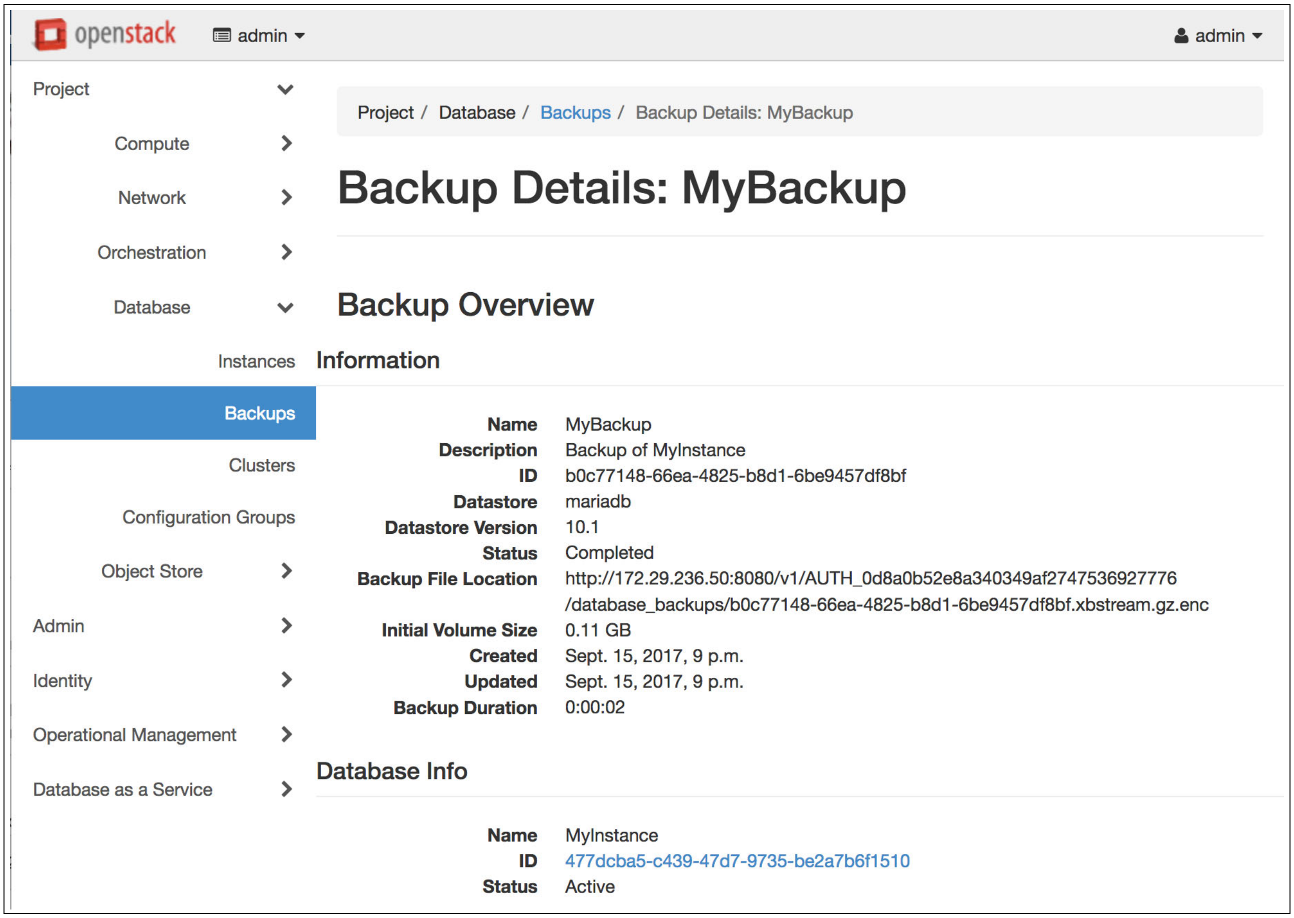


Figure 4-17 Displaying backup details

4.3.1 Creating a backup

To create a backup, click the **Create Backup** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Type a name for the backup and a description. Select the instance to be backed up, and then click **Create**, as illustrated in Figure 4-18 on page 115.

> **Note:** Select the parental backup from the list if you want to do an incremental backup or leave it cleared for a new backup.

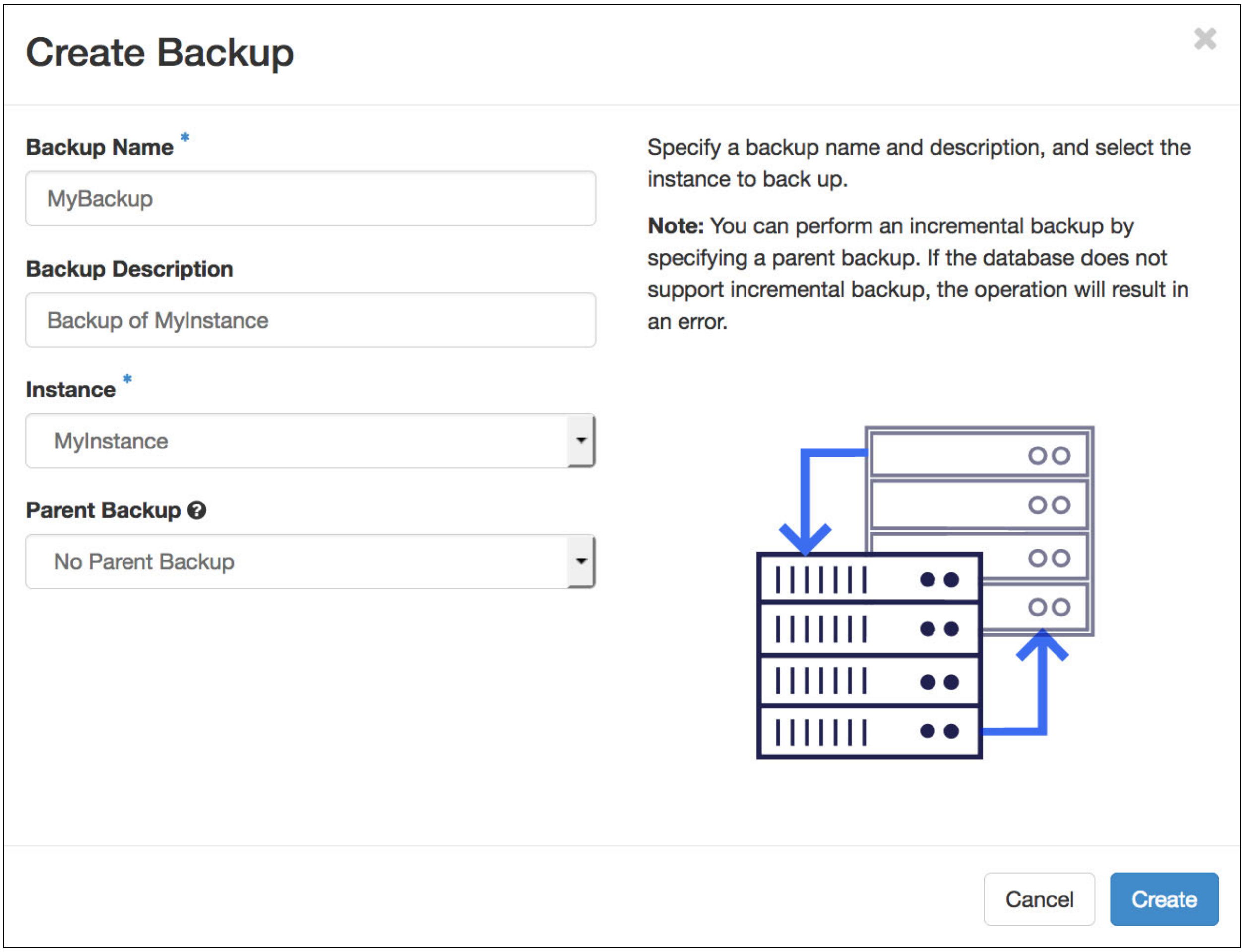


Figure 4-18 Creating a backup

4.3.2 Restoring from backup

Restoring an instance from backup is similar to creating one. The only difference is that you specify the backup that you want to restore and it launches the instance to recover the data.

To restore an instance from a backup, click the **Restore Backup** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Provide the instance information because you are creating an instance and go to the **Backup** tab. Select the backup to be restored and then click **Restore**, as illustrated by Figure 4-19.

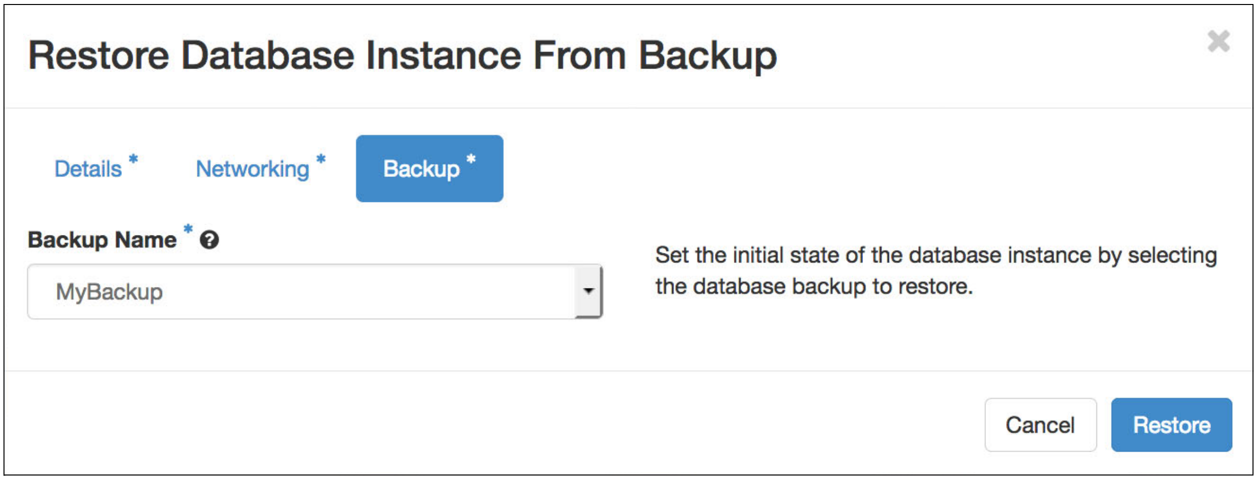


Figure 4-19 Restoring a database instance from backup

4.3.3 Deleting a backup

To delete a backup, click the **Delete Backup** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Select the backup to be deleted and then click **Delete**, as illustrated in Figure 4-20.

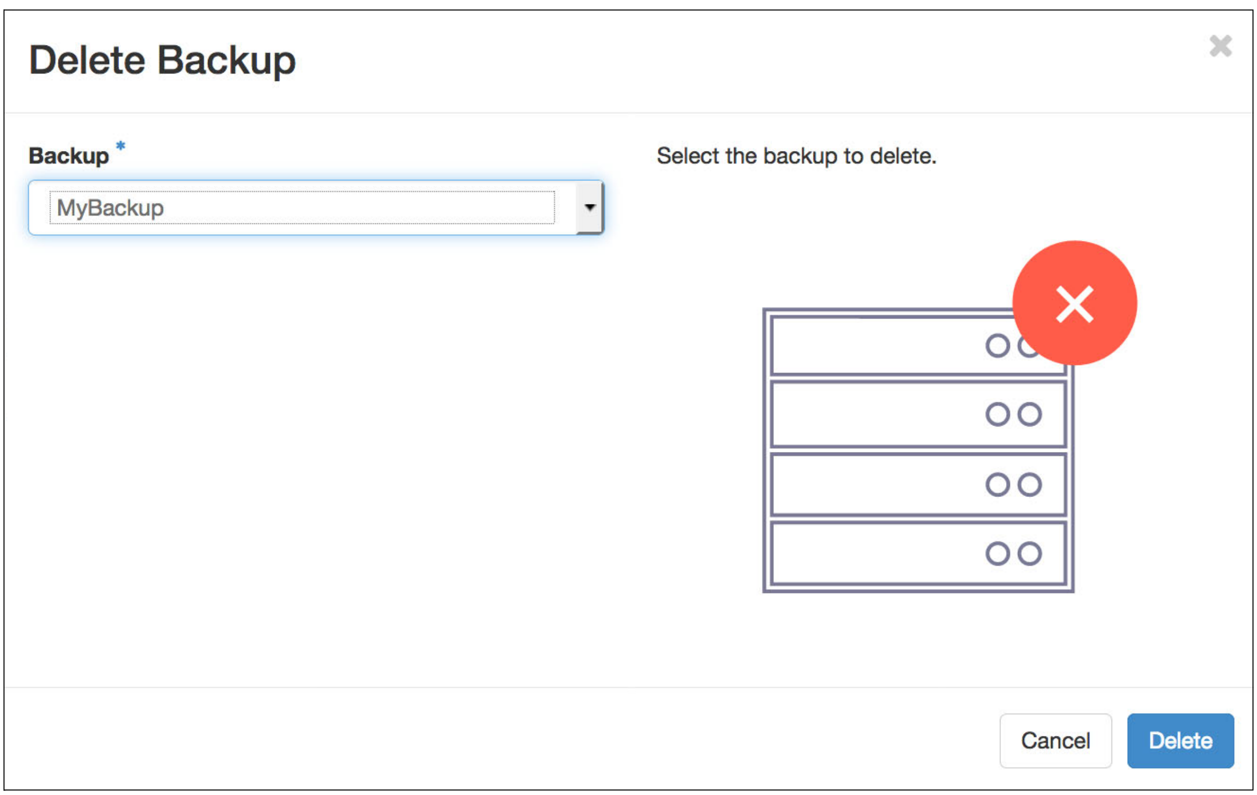


Figure 4-20 Deleting a backup

4.3.4 Backup containers

Backup objects are stored in a Swift container and can be accessed for viewing, editing, deleting, and downloading through the Containers menu, as shown in Figure 4-21.

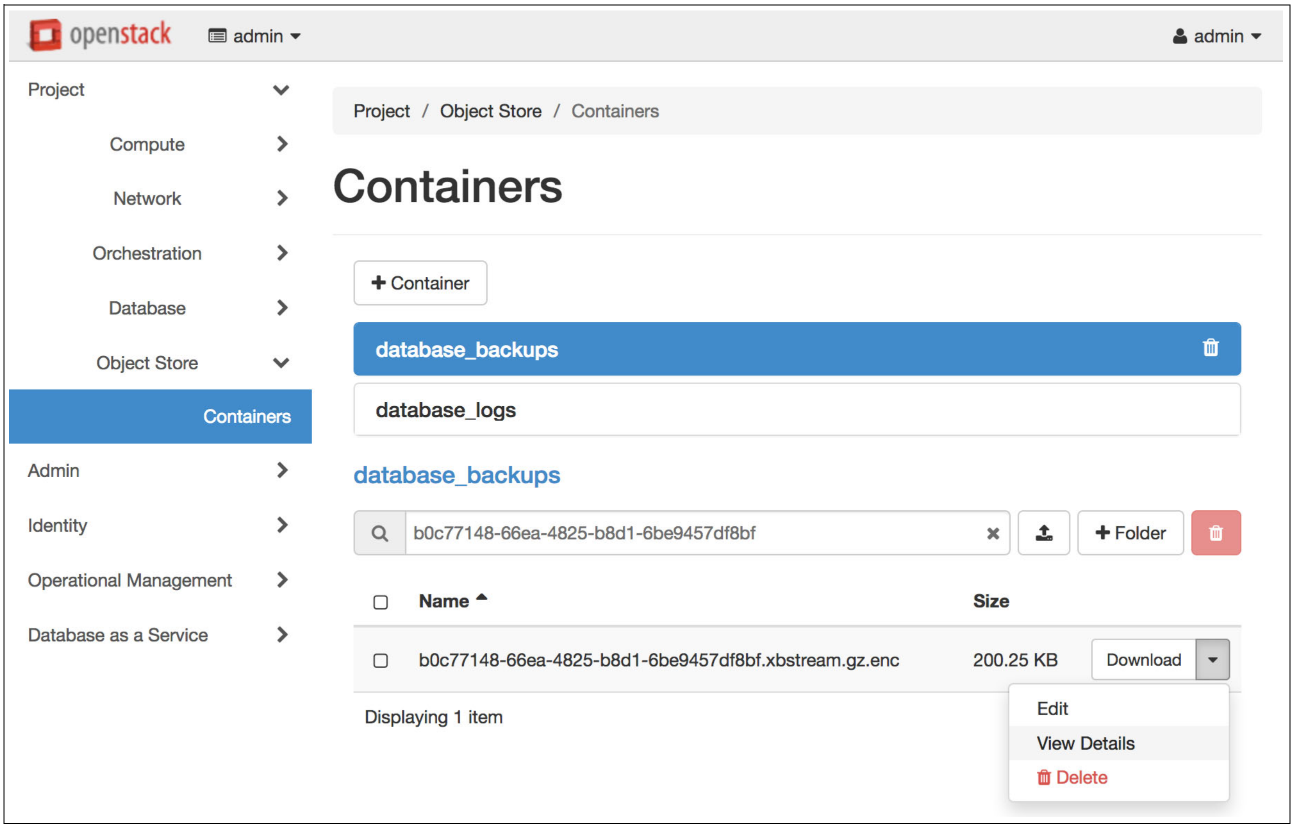


Figure 4-21 Backup containers

You can find the backup file location in Backup Details, as illustrated in Figure 4-17 on page 114.

4.4 Security

You can manage user access for each of your database instances to secure data. This section describes the security aspects of the Open Platform for DBaaS on Power Systems solution and how to manage users' access to databases.

4.4.1 Creating a user

To create a user, click the **Create User** icon in the **Security** group under the **Shortcuts** menu. Type a name and a password for the new user. Select an existing instance and check the databases that you want the user to have access to, as illustrated in Figure 4-22. Click **Create** to confirm.

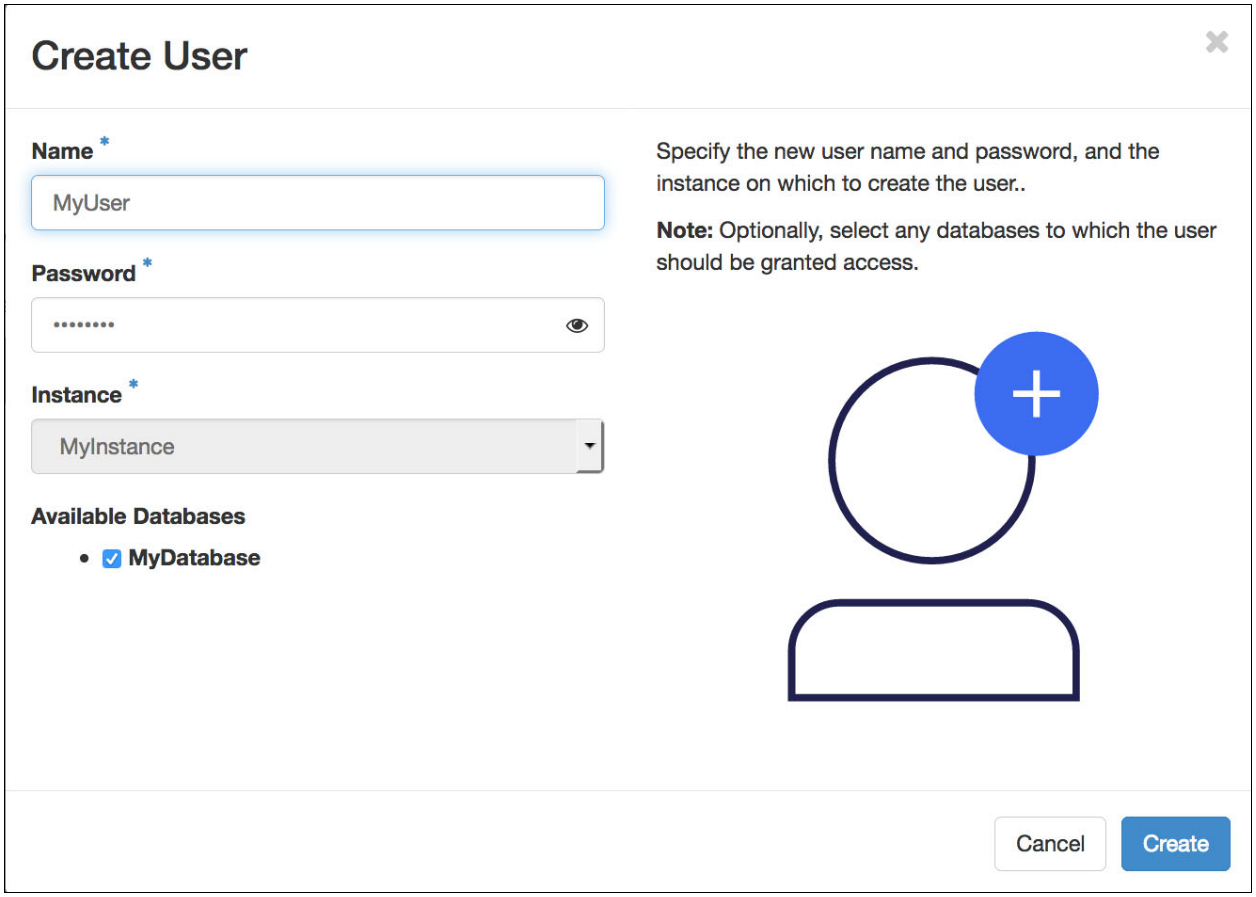


Figure 4-22 Creating a user

4.4.2 Deleting a user

To delete a user, click the **Delete User** icon in the **Life-cycle Management** group under the **Shortcuts** menu. Select the user to be deleted and then click **Delete**, as illustrated in Figure 4-23.

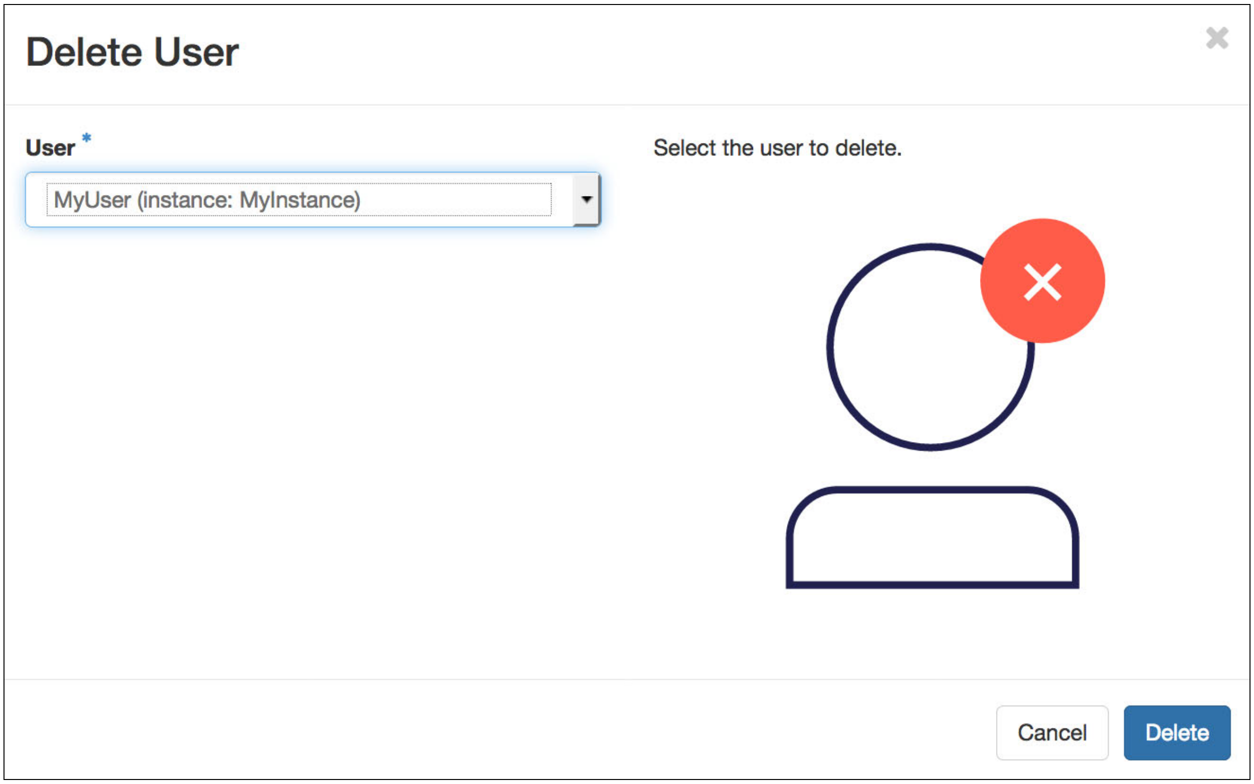


Figure 4-23 Deleting a user

4.4.3 Managing root access

To manage root access for an instance, click the **Manage Root** icon in the **Security** group under the **Shortcuts** menu. Select the instance and then click the **Manage Root Access**, as illustrated in Figure 4-24.

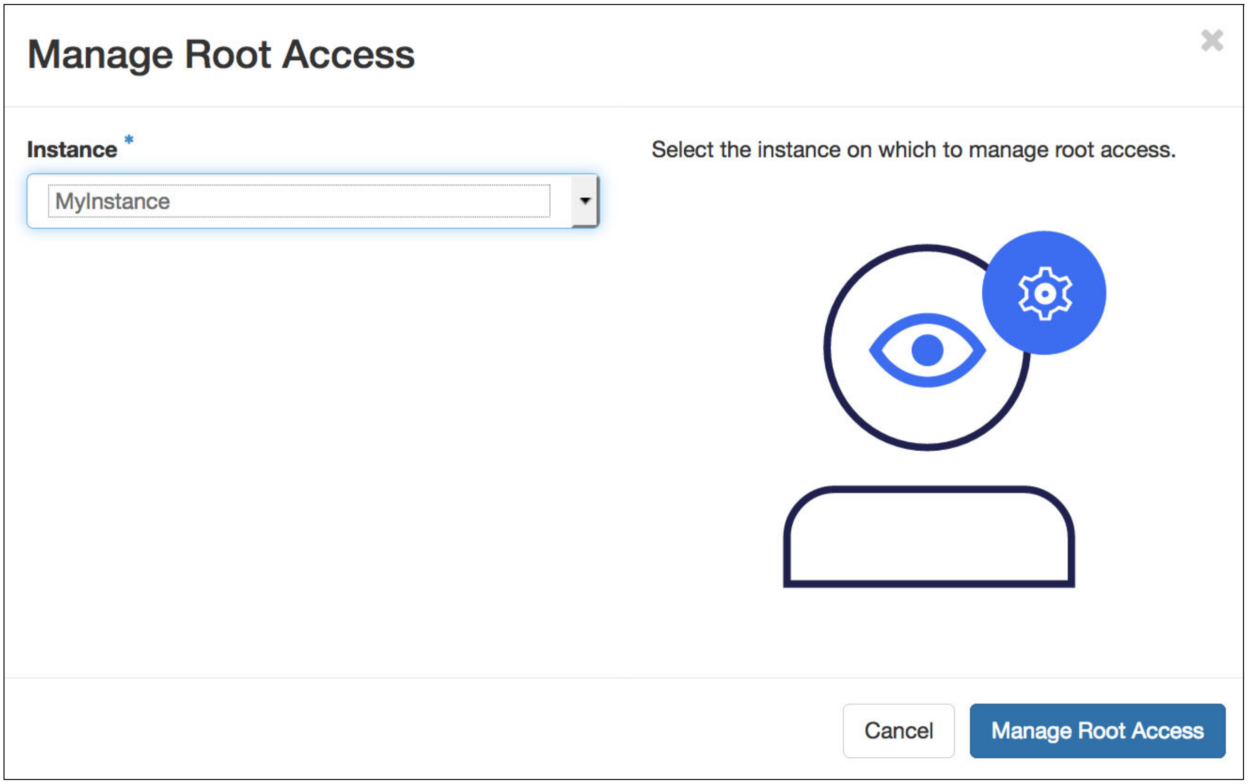


Figure 4-24 Managing root access

The root access for the selected instance is shown in Figure 4-25. This is the first time that the root access is being enabled, and therefore, there is no password that is associated to the instance. To enable root access and generate a password, click **Enable Root**.

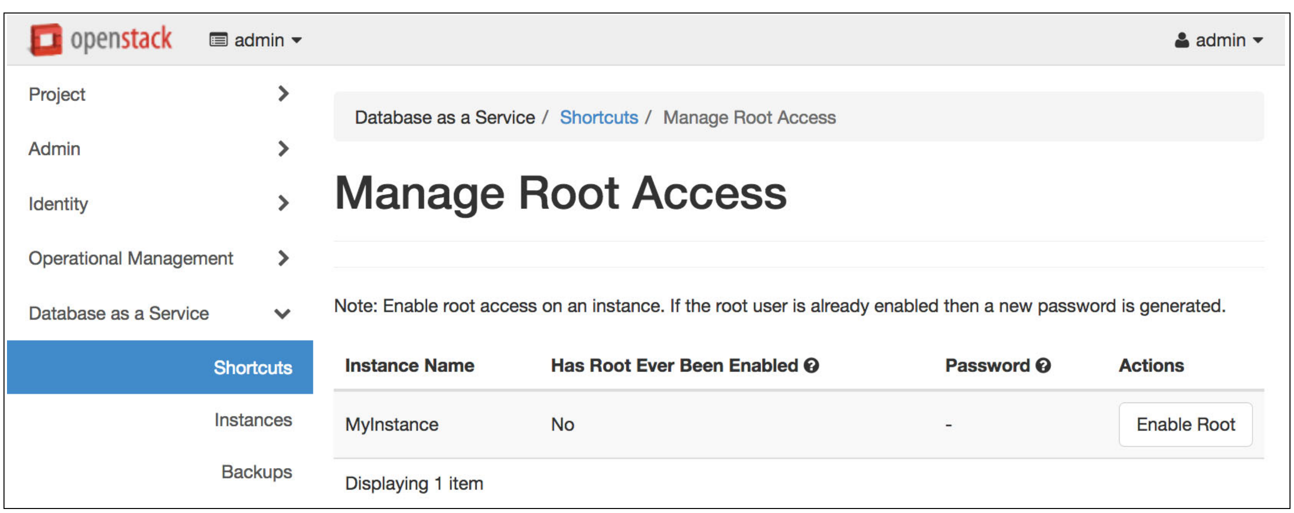


Figure 4-25 Enabling root access

Finally, the root access is enabled for the selected instance and a password is generated, as shown in Figure 4-26.

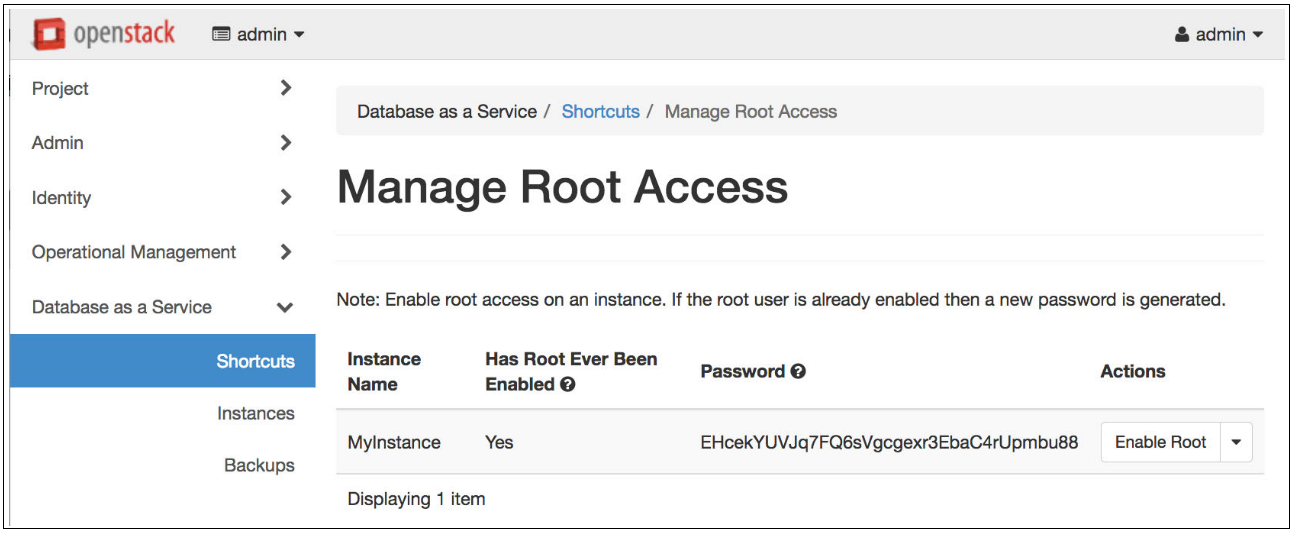


Figure 4-26 Root access enabled with a password generated

4.4.4 Managing user access

To manage user access, click the **Manage User Access** icon in the **Security** group under the **Shortcuts** menu. Select the instance to be managed by the user and then click **Manage User Access**, as illustrated by Figure 4-27.

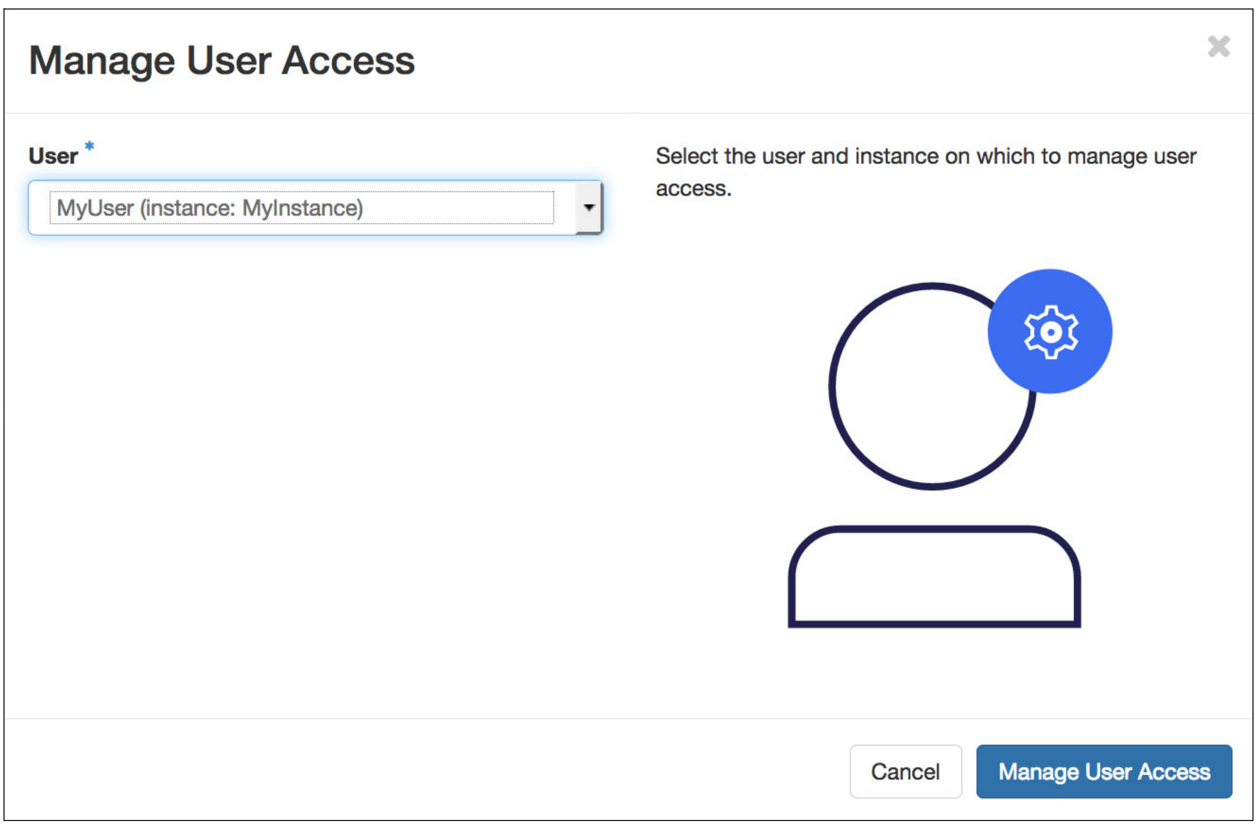


Figure 4-27 Managing user access

The user access for the selected instance is shown in Figure 4-28. This is the first time that the user access is being granted, and therefore, the instance is not accessible. To grant user access, click **Grant Access**.

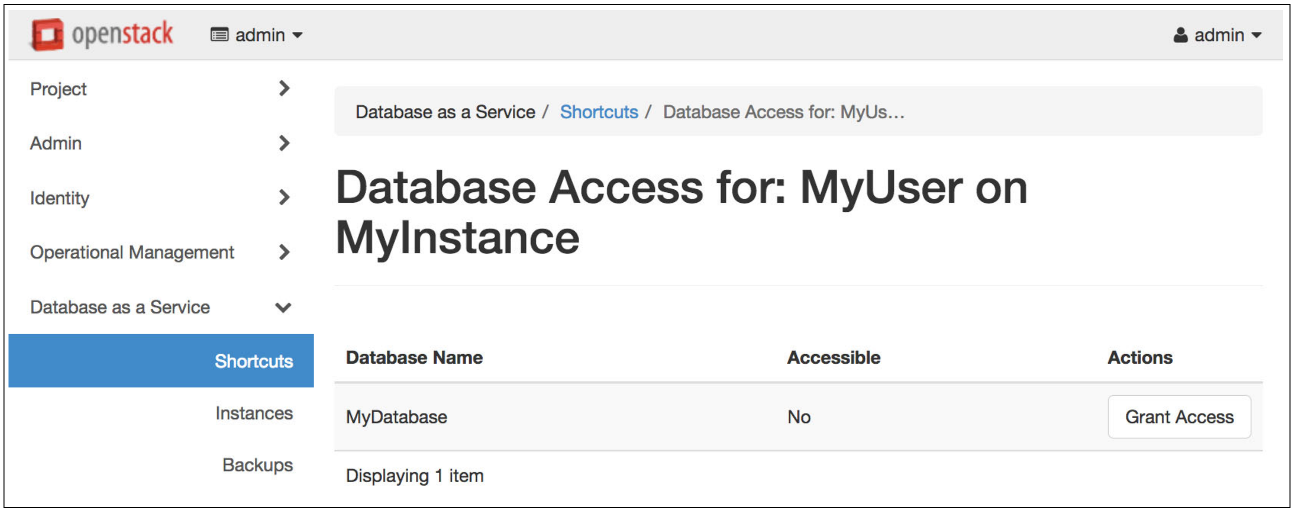


Figure 4-28 Granting user access to an instance

Finally, the user access is granted for the selected instance and the instance is accessible. To revoke access, click **Revoke Access**, as shown in Figure 4-29.

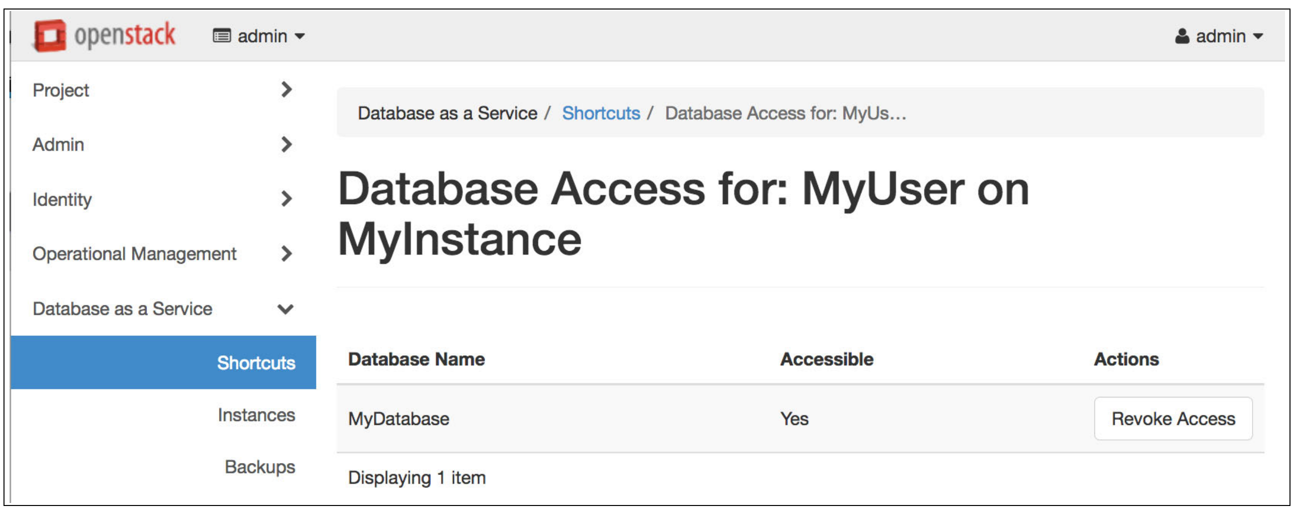


Figure 4-29 User access granted

5

Monitoring and troubleshooting

This chapter explains the monitoring tools that are available in the Open Platform for Database as a Service (DBaaS) on IBM Power Systems solution. This chapter shows how you can access these tools, and use them for monitoring and problem determination.

After reading this chapter, you will understand the following items:

- Which tools are available in the Open Platform for DBaaS on Power Systems solution.
- What is Nagios Core is used for.
- How to access the Nagios Core interface.
- What is ELK Stack and its components (Elasticsearch, Logstash, and Kibana).
- How to access the Kibana interface.
- Usage examples of Nagios Core and Kibana.

This chapter contains the following sections:

- Introduction to cluster monitoring and troubleshooting
- Accessing the operations management tools
- Nagios
- Elastic stack (Kibana)

5.1 Introduction to cluster monitoring and troubleshooting

The Open Platform for DBaaS on Power Systems solution provides operations management (OpsMgr) components for monitoring and problem determination. OpsMgr is an open source project, and all IBM contributions are streamed and freely available on GitHub at Automated deployment of Operational Management services on OpenPOWER.

The OpsMgr components are already deployed in the Open Platform for DBaaS on Power Systems solution, so all these features are available when the appliance is running in your environment.

The OpsMgr project implements a unified portal with Nagios Core and ELK Stack (Elasticsearch, Logstash, and Kibana), which are leading open source software that are widely used for monitoring, and data and log analysis. The OpsMgr project also provides a component that is used for inventory collection of the nodes in the Open Platform for DBaaS on Power Systems infrastructure. Together with the management interface, the nodes that are used in the solution also have other components that send data to the management interface (including metrics, statistics, and logs), keeping the Nagios Core and ELK Stack informed of the activity of the Open Platform for DBaaS on Power Systems cluster. This solution enables the operator and administrator to monitor the environment, use the tools for problem determination purposes, detect the root cause of possible issues, and help the DevOps professional to provide the strategic department with metrics and usage information of the environment, which supports decision making.

All the nodes in the Open Platform for DBaaS on Power Systems solution run a component that is called Nagios Remote Plugin Executor (NRPE), which monitors the node, its services for detection of hardware or software failures, and feeds the Nagios Core (which is running in the controller nodes) such information. The Nagios Core can alert the administrator of any issue.

The nodes also have the Filebeat and Metricbeat components, which are used by ELK Stack to send logs (Filebeat) and metrics (Metricbeat) to the Logstash component (which runs in the controller node). Logstash manipulates the logs and information, standardizing and indexing them, enabling Elasticsearch (a powerful search engine) to use such information. Kibana is a web interface that integrates with ElasticSeach and provides useful tools so that the administrator or DevOps professional can use this information for statistics, metrics, and problem determination.

Figure 5-1 shows the architecture of the OpsMgr components in the Open Platform for DBaaS on Power Systems solution.

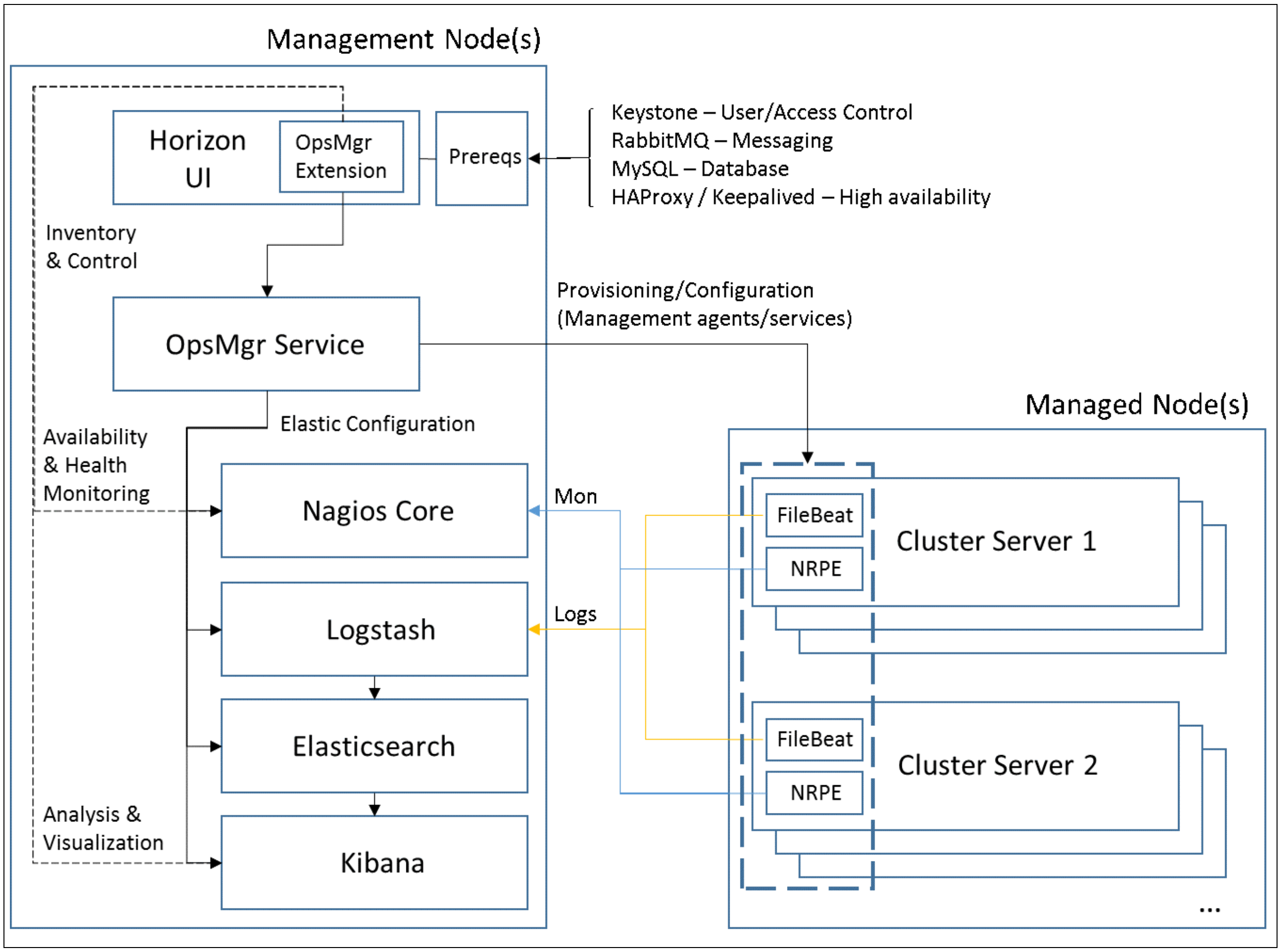


Figure 5-1 Architecture of the OpsMgr in the Open Platform for DBaaS on Power Systems solution

The OpsMgr tools deliver the following types of components:

- A GUI that is accessible through a browser.
- Server-side components that receive information from the agents and provide management functions.
- Client-side components that send events, alerts, logs, and metrics to the server-side components.

In the Open Platform for DBaaS on Power Systems solution, the OpsMgr components provide integration in the Horizon interface to access the web GUI interfaces for Nagios Core and Kibana. Also, OpsMgr handles the communication between server-side and client-side components as follows:

- Filebeat and Metricbeat send data to the ELK stack services.
- THe NRPE service monitors hardware and software services and sends data to Nagios Core.

The server-side components are deployed by using LXC containers, where the Nagios and ELKS stack services run, as shown in Example 5-1. Such containers run in the controller nodes.

Example 5-1 Containers running the OpsMgr components in the controller nodes

```
ubuntu@int3-controller-1:~$ sudo su -
root@int3-controller-1:~# lxc-ls
int3-controller-1-elasticsearch           int3-controller-1-kibana
int3-controller-1-logstash
int3-controller-1-nagios
int3-controller-1_cinder_api_container-59d7c0d3
int3-controller-1_cinder_scheduler_container-22cb2b47
int3-controller-1_cinder_volumes_container-84904059
int3-controller-1_galera_container-60dd2018
int3-controller-1_glance_container-20025c0f
int3-controller-1_heat_apis_container-47d7649d
int3-controller-1_heat_engine_container-f5b99ff8
int3-controller-1_horizon_container-58d67e89
int3-controller-1_keystone_container-5b56181d
int3-controller-1_memcached_container-6d4a2903
int3-controller-1_neutron_agents_container-fd7ffcc5
int3-controller-1_neutron_server_container-98cbee41
int3-controller-1_nova_api_metadata_container-031da642
int3-controller-1_nova_api_os_compute_container-2e185141
int3-controller-1_nova_cert_container-72785691
int3-controller-1_nova_conductor_container-4f8f5a5a
int3-controller-1_nova_console_container-ccc4201e
int3-controller-1_nova_scheduler_container-75db61d6
int3-controller-1_rabbit_mq_container-689cbf0d
int3-controller-1_repo_container-0faabfb0
int3-controller-1_rsyslog_container-3d257960
int3-controller-1_swift_proxy_container-aa58df5a
int3-controller-1_trove_api_container-41db10c1
int3-controller-1_trove_conductor_container-53fa0cc1
int3-controller-1_trove_taskmanager_container-f1b5f150
int3-controller-1_utility_container-772b622e
root@int3-controller-1:~# lxc-attach --name int3-controller-1-nagios

root@int3-controller-1-nagios:~# ps -ef | grep nagios | grep cfg
nagios    39747      1  0 Sep27 ?        00:00:00 /usr/sbin/nrpe -c
/etc/nagios/nrpe.cfg -d
nagios    68438      1  0 17:23 ?        00:00:06 /usr/local/nagios/bin/nagios
/usr/local/nagios/etc/nagios.cfg
```

5.2 Accessing the operations management tools

You can access the OpsMgr tools in the Horizon interface by using the *Operational Management* function. During the planning phase of your Open Platform for DBaaS on Power Systems environment, you selected the floating IP address that is used to access the Horizon interface in your cluster (see “Information that is required from the existing infrastructure” on page 47, especially Table 3-4 on page 49). This IP address is used in your browser to access the Horizon interface:

```
https://<floating_horizon_ip_address>
```

Log in by using the admin or other user name and password, as shown in Figure 5-2.

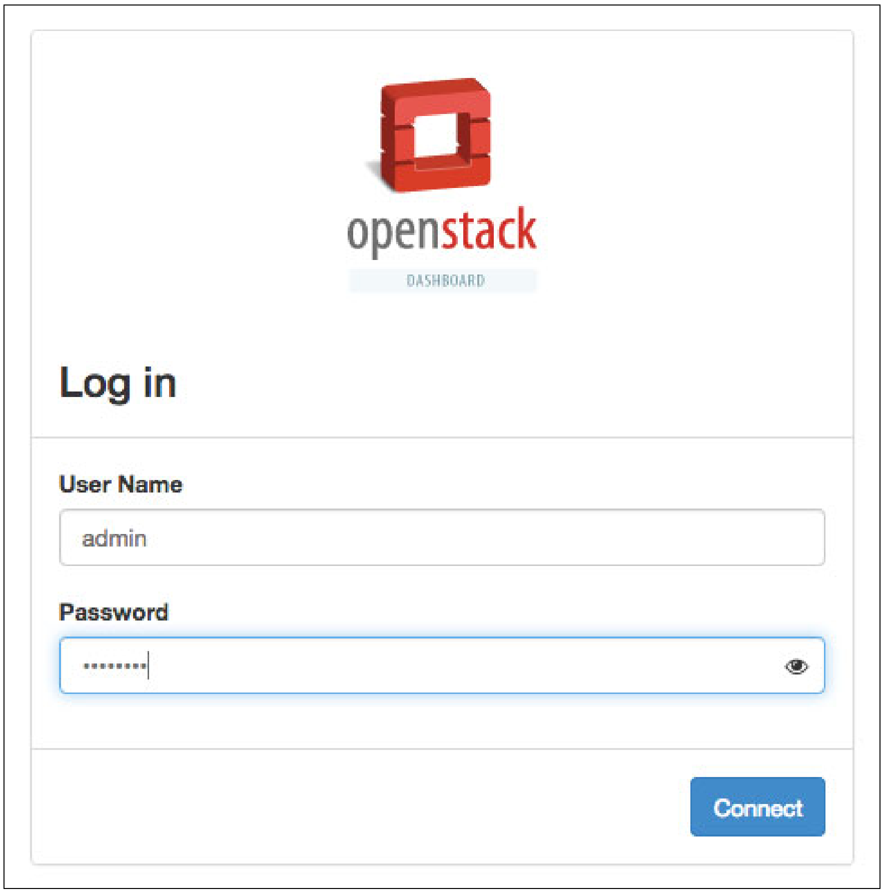


Figure 5-2 Horizon login interface

After logging in to the Horizon Dashboard, click **Operational Management** in the left pane and then click **Inventory**, as shown in Figure 5-3.

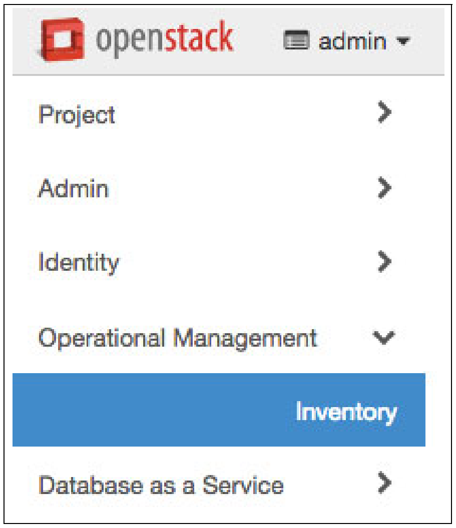
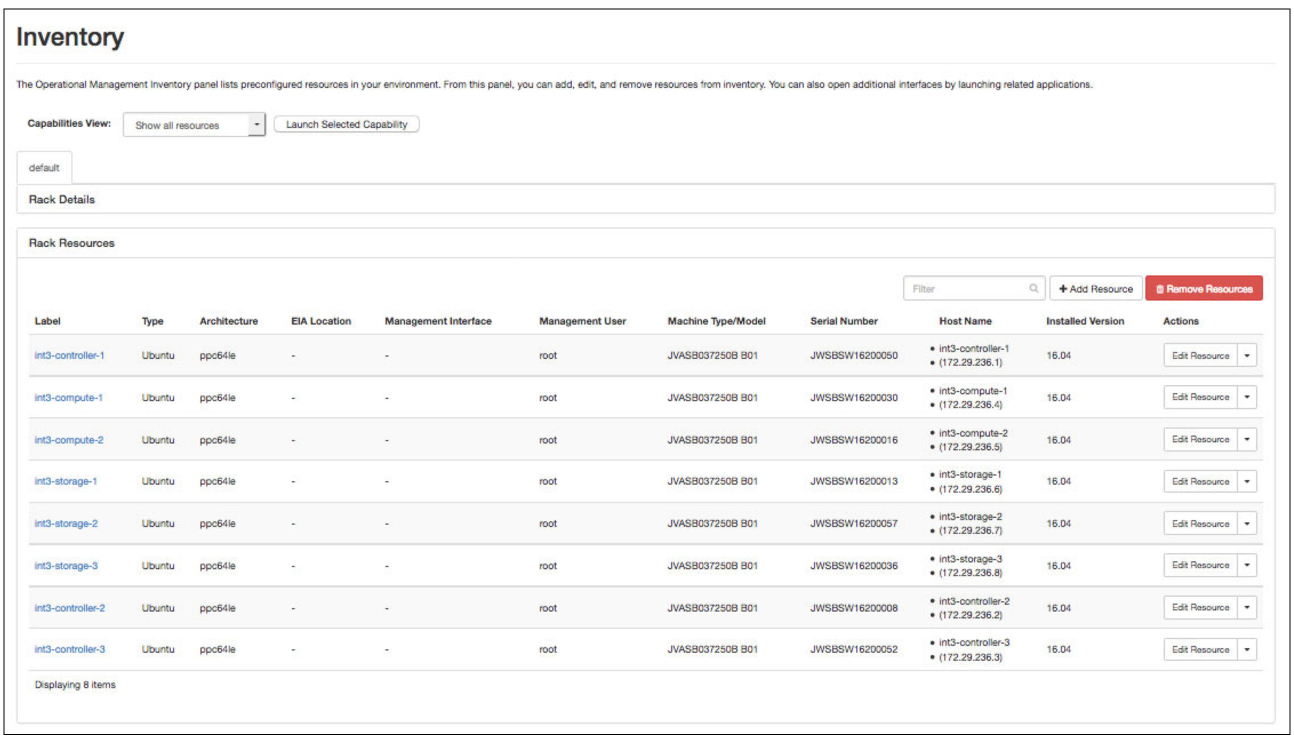


Figure 5-3 Operational management pane

The Inventory window lists all resources that are configured in your environment. The window shows the nodes in your cluster (controller, compute, and storage nodes), and their respective IP addresses, machine serial numbers, and the version of the operating system (OS) that is running, as shown in Figure 5-4.



Label	Type	Architecture	EIA Location	Management Interface	Management User	Machine Type/Model	Serial Number	Host Name	Installed Version	Actions
int3-controller-1	Ubuntu	ppc64le	-	-	root	JVASB037250B B01	JWSBSW16200050	• int3-controller-1 • (172.29.236.1)	16.04	Edit Resource
int3-compute-1	Ubuntu	ppc64le	-	-	root	JVASB037250B B01	JWSBSW16200030	• int3-compute-1 • (172.29.236.4)	16.04	Edit Resource
int3-compute-2	Ubuntu	ppc64le	-	-	root	JVASB037250B B01	JWSBSW16200016	• int3-compute-2 • (172.29.236.5)	16.04	Edit Resource
int3-storage-1	Ubuntu	ppc64le	-	-	root	JVASB037250B B01	JWSBSW16200013	• int3-storage-1 • (172.29.236.6)	16.04	Edit Resource
int3-storage-2	Ubuntu	ppc64le	-	-	root	JVASB037250B B01	JWSBSW16200057	• int3-storage-2 • (172.29.236.7)	16.04	Edit Resource
int3-storage-3	Ubuntu	ppc64le	-	-	root	JVASB037250B B01	JWSBSW16200036	• int3-storage-3 • (172.29.236.8)	16.04	Edit Resource
int3-controller-2	Ubuntu	ppc64le	-	-	root	JVASB037250B B01	JWSBSW16200008	• int3-controller-2 • (172.29.236.2)	16.04	Edit Resource
int3-controller-3	Ubuntu	ppc64le	-	-	root	JVASB037250B B01	JWSBSW16200052	• int3-controller-3 • (172.29.236.3)	16.04	Edit Resource

Displaying 8 items

Figure 5-4 Inventory window in the Operational Management interface

From the Inventory window, you can select, in the Capabilities View, the Logging (ELK Stack interface - Kibana) or Monitoring (Nagios Core) tool that you want to start, and then click **Launch Selected Capability**, as shown in Figure 5-5.

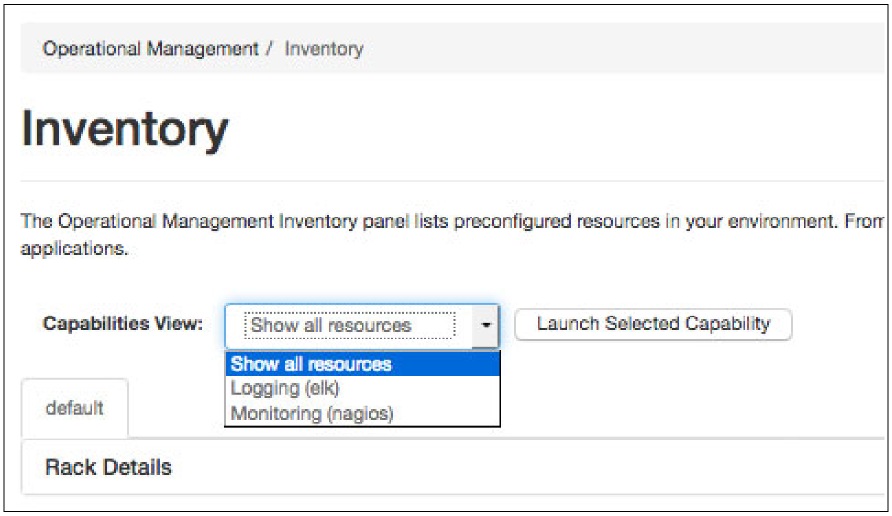


Figure 5-5 Capabilities View in the Inventory window

The selected Capability (Kibana or Nagios) starts, and you are prompted to provide the user name and password for authentication. The default user name and passwords are shown in Table 5-1.

Table 5-1 Default user name and password for Nagios Core and Kibana

Capability	User name	Password
ELK Stack (Kibana)	kibana	kibana
Nagios Core	nagios	nagios

Change the password after the first login. For more information about how to manage users and passwords for Nagios and Kibana, see the Nagios documentation and KIbana documentation.

The following sections provide more details about Nagios Core and Kibana usage.

5.3 Nagios

It is important to constantly monitor your environment to detect any issue before it affects production systems, giving you time to take appropriate actions to fix them, hence avoiding any impact on operations. There are several monitoring tools that are available to help you detect failures and send alerts to the system administrators. One of the most popular tools, widely adopted by the enterprises and community, is *Nagios Core*.

Nagios Core is an extensible monitoring system that can monitor servers, virtual machines (VMs), services, network switches, routers, and so on. Nagios Core detects IT infrastructure problems, helping you to take actions to correct them as quick as possible, therefore avoiding or minimizing impacts to your critical business servers.

Nagios Core is no-charge, open source, and aligned with the strategy for the Open Platform for DBaaS on Power Systems solution. Nagios Core alerts the system administrator when issues happen and then alerts again when the problem is solved.

Here are some of the items that are monitored by Nagios Core:

- Network services (SMTP, HTTP, ICMP, SNMP, FTP, and SSH)
- Host resources (CPU workload, disk usage, and system logs)
- Hardware issues (including temperature alarms)

> **Note:** Nagios Enterprise has a licensed version of Nagios that is called Nagios XI. The Open Platform for DBaaS on Power Systems solution uses the Nagios Core, which is the open source, no-charge version of the monitoring tool that is provided by Nagios.

Many situations can be monitored by using SSH commands or the NRPE. The NRPE is an agent that is provided by Nagios Core that performs monitoring of remote systems by using scripts that are hosted on the remote system being monitored. Nagios Core polls information from the remote system by using a plug-in that is called `check_nrpe`.

With the `check_nrpe` calls, Nagios Core runs its plug-ins on remote systems to monitor machine metrics (CPU, memory, disk usage, network throughput, and so on) and provide such information in a web GUI for the system administrator. Nagios Core also sends alerts through email, SMS, SNMP, or other methods with the use of Nagios plug-ins.

Figure 5-6 illustrates the process that is used by NRPE to run scripts in the hosts that are monitored by Nagios Core.

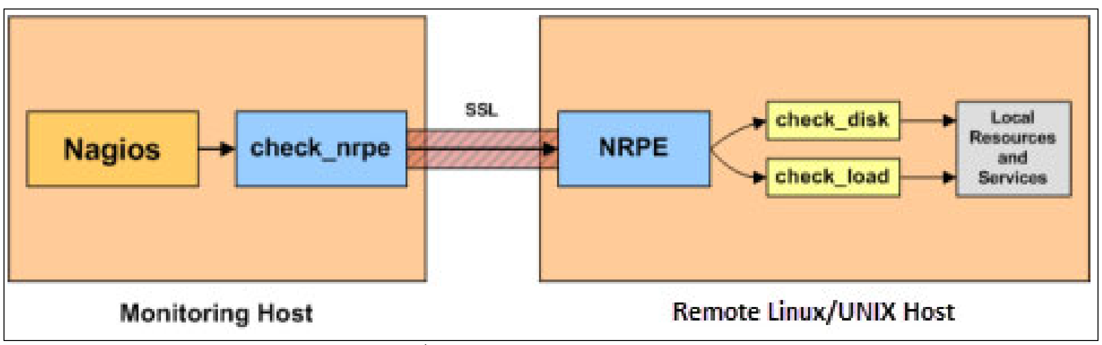


Figure 5-6 The NRPE agent functions[1]

> **Note:** In the Open Platform for DBaaS on Power Systems solution, the servers and services are monitored by using the NRPE agent.

[1] The source of the image is https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/monitoring-linux.html.

5.3.1 Nagios Core basic monitoring concepts

Nagios Core has some basic monitoring concepts. All the elements that are used by Nagios Core in the monitoring, notification process, and logic are named *objects*. Some of the object types that are used by Nagios Core are as follows:

- Commands: This type of object is used to tell Nagios Core which commands, scripts, or agents it needs to run to check the host and service status and provide notification actions. An example of a command object is `check_http` or `check_nrpe`.
- Hosts: The physical machines that are monitored by Nagios Core (including servers and network switches, for example). In the definition of the host object, you must provide information about how it can be reached (IP or MAC address), who can be contacted in a notification event, which checks need to be performed, and when to perform them.
- Host groups: The hosts can be grouped in host groups, for example, `Controller_Nodes`.
- Services: Services, functions, and resources to monitor the host or host group, such as SSH and OpenStack services (Heat, Cinder, Nova, RabbitMQ, and so on). In the services object, you also provide instructions for Nagios Core about how to monitor such services, when to perform the monitoring, and who to contact in case of issues.
- Service groups: You can group your services objects into a service group, such as `OpenStack_services`.
- Contact: Define the people that can be notified during the notification process. Each contact has one or more notification methods (such as email and mobile phone for SMS). The contact receives notification for hosts and services for which they are responsible.
- Contact groups: The contacts can be groups in the contact groups, such as `Database_administrators` or `OpenStack_administrators`.
- Timeperiods: This type of object is used to determine when hosts and services are monitored and when contacts are notified.

Throughout its monitoring checks, Nagios Core uses four categories to describe the state of the object being monitored: `OK`, `WARNING`, `CRITICAL`, and `UNKNOWN`. To avoid temporary or random problems, Nagios Core also uses the concept of `SOFT` and `HARD` states. The `SOFT` state is related to situations where a server or service is down temporarily. Additional checks are performed and then Nagios Core determines whether the server or services have recovered or if they are still down. If they are still down, they are moved to a `HARD` state because the problem is considered permanent. You can use Nagios Core to check for a history of events in your systems so you can see whether there was a `SOFT` or `HARD` event of a `WARNING` or `CRITICAL` problem.

5.3.2 Nagios Core deployment in Open Platform for DBaaS on Power Systems

In the Open Platform for DBaaS on Power Systems cluster, the Nagios Core server is installed in all controller nodes. Ubuntu Linux LXC containers are used for the deployment of Nagios Core, as shown in Example 5-2.

Example 5-2 Nagios Core container in one of the controller nodes

```
root@int3-controller-1:~# lxc-ls | grep nagios
int3-controller-1-nagios
```

All controller nodes have a Nagios Core container that is similar to the one that is shown in Example 5-2 on page 131. The haproxy daemon balances the workload between all available Nagios Core servers, as shown in Example 5-3, which contains a section of the haproxy configuration file balancing the workload between the three Nagios Core servers (in this Open Platform for DBaaS on Power Systems cluster, there are three controller nodes).

Example 5-3 Nagios core workload balanced by haproxy

```
root@int3-controller-1:~# ps -ef | grep /usr/sbin/haproxy
root      11725      1  0 06:37 ?        00:00:00
/usr/sbin/haproxy-systemd-wrapper -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
haproxy   11728  11725  0 06:37 ?        00:00:00 /usr/sbin/haproxy -f
/etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds
haproxy   11738  11728  0 06:37 ?        00:02:49 /usr/sbin/haproxy -f
/etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds
root      20277   4111  0 12:45 pts/27   00:00:00 grep --color=auto
/usr/sbin/haproxy

root@int3-controller-1:~# vi /etc/haproxy/haproxy.cfg
...
frontend nagios-http-front
bind *:8001
    mode http
    option httplog
    option forwardfor except 127.0.0.0/8
    option http-server-close
    default_backend nagios-http-back

backend nagios-http-back
    mode http
    option forwardfor
    option httpchk
    option httplog
    server int3-controller-1-nagios 172.29.236.9:80 check port 80 inter 10s fall 1
rise 1
    server int3-controller-2-nagios 172.29.236.13:80 check backup
    server int3-controller-3-nagios 172.29.236.17:80 check backup
...
```

Because there are multiple Nagios Core servers, any configuration that you modify in the Nagios Core configuration files must be performed on all Nagios Core instances, or you have different results in Nagios, depending on which Nagios Core instance is tending to your connection.

5.3.3 Nagios Core configuration files

The Nagios Core configuration files are in the containers where Nagios Core is installed in each of the controller nodes. To access these containers, run the **lxc-ls** and **lxc-attach** commands, as shown in Example 5-4.

Example 5-4 Connecting to one of the Nagios Core containers

```
root@int3-controller-1:~# lxc-ls | grep nagios
int3-controller-1-nagios
int3-controller-1_cinder_api_container-5dfc036b
int3-controller-1_cinder_scheduler_container-3c5d9754
```

```
root@int3-controller-1:~# lxc-attach -n int3-controller-1-nagios
root@int3-controller-1-nagios:~# ps -ef | grep nagios
nagios    39411      1  0 Oct05 ?        00:00:00 /usr/sbin/nrpe -c
/etc/nagios/nrpe.cfg -d
nagios    60186      1  0 Oct05 ?        00:04:08 /usr/local/nagios/bin/nagios
/usr/local/nagios/etc/nagios.cfg
...
```

Example 5-4 on page 132 shows that the main Nagios Core daemon is using the configuration file `/usr/local/nagios/etc/nagios.cfg`. This is the main configuration file for Nagios Core. In this file, you can tell Nagios Core where are the objects it needs to monitor, the notification methods, and the contacts.

The objects are defined in one or more configuration files and directories, which are specified in the `cfg_file` and `cfg_dir` entries in the `nagios.cfg` file. Example 5-5 shows a section of the `nagios.cfg` configuration files, where several objects are defined. Each of these entries point to a text file with the information that Nagios Core needs to work with such object.

Example 5-5 Object configuration files in nagios.cfg

```
...
# OBJECT CONFIGURATION FILE(S)
# These are the object configuration files in which you define hosts,
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you want (as shown below), or keep them all in a single config file.

# You can specify individual object config files as shown below:
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg
...
```

The object files are configured in text files, making it easier to understand and modify any necessary configuration. Example 5-6 shows the `contacts.cfg` configuration file, which you can modify and add the email of the group of system administrators that manage the Open Platform for DBaaS on Power Systems solution so that they are notified by email in case of any problem that is detected by Nagios Core.

Example 5-6 The contacts.cfg configuration file

```
root@int3-controller-1-nagios:~# cat /usr/local/nagios/etc/objects/contacts.cfg
###############################################################################
# CONTACTS.CFG - SAMPLE CONTACT/CONTACTGROUP DEFINITIONS
#
#
# NOTES: This config file provides you with some example contact and contact
#        group definitions that you can reference in host and service
#        definitions.
#
#        You don't need to keep these definitions in a separate file from your
#        other object definitions.  This has been done just to make things
```

```
#        easier to understand.
#
###############################################################################



###############################################################################
###############################################################################
#
# CONTACTS
#
###############################################################################
###############################################################################

# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the
'generic-contact'
# template which is defined elsewhere.

define contact{
        contact_name                    nagiosadmin; Short name of user
   use         generic-contact; Inherit default values from generic-contact
template (defined above)
        alias                           Nagios Admin; Full name of user

        email                           nagios@localhost; <<***** CHANGE THIS TO
YOUR EMAIL ADDRESS ******
        }



###############################################################################
###############################################################################
#
# CONTACT GROUPS
#
###############################################################################
###############################################################################

# We only have one contact in this simple configuration file, so there is
# no need to create more than one contact group.

define contactgroup{
        contactgroup_name       admins
        alias                   Nagios Administrators
        members                 nagiosadmin
        }
```

The `cfg_dir` directive in the nagios.cfg has a similar function as the `cfg_file`. With the `cfg_dir`, you can direct Nagios Core to process all configuration files ending with `.cfg` in a specific directory, as shown in the Example 5-7 on page 135.

Example 5-7 The cfg_dir directive

```
cfg_dir=/usr/local/nagios/opsmgr/nagios_config/commands
cfg_dir=/usr/local/nagios/opsmgr/nagios_config/hosts
```

In the `/usr/local/nagios/opsmgr/nagios_config/commands` directory, you can see some configuration files, which are used by Nagios Core to determine the commands it must run to monitor the Power Systems nodes and the Mellanox and the Lenovo network switches, as shown in Example 5-8.

Example 5-8 Nagios commands in cfg_dir

```
root@int3-controller-1-nagios:~# ls -l
/usr/local/nagios/opsmgr/nagios_config/commands
total 16
-rw-r--r-- 1 root root 276 Oct  5 21:55 common_commands.cfg
-rw-r--r-- 1 root root 969 Oct  5 21:55 LenovoRackSwitch.cfg
-rw-r--r-- 1 root root 817 Oct  5 21:55 MLNX-OS.cfg
-rw-r--r-- 1 root root 599 Oct  5 21:55 PowerNode.cfg
```

In the `/usr/local/nagios/opsmgr/nagios_config/hosts` directory, there are configuration files for all objects that are monitored by Nagios Core. A snippet of such a directory is shown in Example 5-9.

Example 5-9 Snippet of the files in /usr/local/nagios/opsmgr/nagios_config/hosts

```
root@int3-controller-1-nagios:~# ls -l
/usr/local/nagios/opsmgr/nagios_config/hosts/
total 476
-rw-r--r-- 1 nagios nagios  346 Oct  5 22:15 int3-compute-1.cfg
-rw-r--r-- 1 nagios nagios  223 Oct  5 22:32 int3-compute-1-osa_compute.cfg
-rw-r--r-- 1 nagios nagios  211 Oct  5 22:16 int3-compute-1-server.cfg
-rw-r--r-- 1 nagios nagios  346 Oct  5 22:16 int3-compute-2.cfg
-rw-r--r-- 1 nagios nagios  223 Oct  5 22:32 int3-compute-2-osa_compute.cfg
-rw-r--r-- 1 nagios nagios  211 Oct  5 22:16 int3-compute-2-server.cfg
-rw-r--r-- 1 nagios nagios  217 Oct  5 22:45 int3-controller-1-ceph_monitor.cfg
-rw-r--r-- 1 nagios nagios  355 Oct  5 22:15 int3-controller-1.cfg
-rw-r--r-- 1 nagios nagios  219 Oct  5 22:37 int3-controller-1-elasticsearch.cfg
-rw-r--r-- 1 nagios nagios  205 Oct  5 22:37 int3-controller-1-kibana.cfg
-rw-r--r-- 1 nagios nagios  209 Oct  5 22:37 int3-controller-1-logstash.cfg
-rw-r--r-- 1 nagios nagios  227 Oct  5 22:33 int3-controller-1-osa_cinder_api.cfg
...
```

By checking the configuration files, you can see the commands that are used by Nagios Core to monitor the servers and services, and the interval at which such checks are performed. Example 5-10 shows the checks that are performed for the Kibana daemon in node int3-controller-3 and the server check performed in int3-compute-1. In these examples, all the verifications are based on the NRPE agent daemon.

Example 5-10 Checks that are performed by Nagios Core during monitoring

```
root@int3-controller-1-nagios:~# cat
/usr/local/nagios/opsmgr/nagios_config/hosts/int3-controller-3-kibana.cfg
define service {
   use                  generic-service
   host_name            int3-controller-3
   service_description  Kibana
```

```
   check_command        check_nrpe!kibana ""
   check_interval       10
}

root@int3-controller-1-nagios:~# cat
/usr/local/nagios/opsmgr/nagios_config/hosts/int3-compute-1-server.cfg
define service {
   use                  generic-service
   host_name            int3-compute-1
   service_description  Standard Server
   check_command        check_nrpe!server ""
   check_interval       10
}
```

Important: Any modification of `nagios.cfg` or any other configuration file that is specified in the `cfg_file` or `cfg_dir` entries must be performed on all Nagios Core instances in all controller nodes to keep the configuration uniform on all Nagios Core instances.

In the Open Platform for DBaaS on Power Systems solution, all nodes that are monitored by Nagios Core are running the NRPE agent. In such nodes, you can see the daemon running and reading the configuration from the `/etc/nagios/nrpe.cfg` configuration file. Example 5-11 shows the `nrpe` daemon running in a compute node and a snippet of the `/etc/nagios/nrpe.cfg` file, which shows the commands that Nagios Core can tell `nrpe` to run for monitoring purposes.

Example 5-11 Nagios Core nrpe agent configuration

```
ubuntu@int3-compute-1:~$ ps -ef | grep nagios
ubuntu    21479  20938  0 14:38 pts/0    00:00:00 grep --color=auto nagios
nagios    75437      1  0 Oct05 ?        00:00:00 /usr/sbin/nrpe -c
/etc/nagios/nrpe.cfg -d

ubuntu@int3-compute-1:~$ cat /etc/nagios/nrpe.cfg
...
command[check_users]=/usr/lib/nagios/plugins/check_users -w 5 -c 10
command[check_load]=/usr/lib/nagios/plugins/check_load -w 15,10,5 -c 30,25,20
command[check_hda1]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% -p /dev/hda1
command[check_zombie_procs]=/usr/lib/nagios/plugins/check_procs -w 5 -c 10 -s Z
command[check_total_procs]=/usr/lib/nagios/plugins/check_procs -w 150 -c 200
...

ubuntu@int3-compute-1:~$ file /usr/lib/nagios/plugins/check_disk
/usr/lib/nagios/plugins/check_disk: ELF 64-bit LSB executable, 64-bit PowerPC or
cisco 7500, version 1 (SYSV), dynamically linked, interpreter /lib64/ld64.so.2,
for GNU/Linux 3.2.0, BuildID[sha1]=e59d3fb799dee1b5f2c1729414a5226a70d50e1d,
stripped
```

The Nagios Core configuration is performed during the Open Platform for DBaaS on Power Systems deployment for Nagios Core to monitor the Open Platform for DBaaS on Power Systems components. Later, you can perform additional customizations, such as using several Nagios Core plug-ins, which you can find at Nagios plug-ins and Nagios Exchange.

Note: Although you may download, install, and use the plug-ins, they are not supported by IBM.

5.3.4 Nagios Core usage examples

This section provides monitoring examples that are performed by Nagios Core and how you can view such information in the web GUI. For instructions about how to access the Nagios Core web interface, see 5.2, “Accessing the operations management tools” on page 127.

Tip: There are other actions that you can perform with Nagios Core. For more information, see the Nagios Core documentation.

After using the Operational Management window in the Horizon interface and accessing the Nagios Core web GUI, you see the Nagios Home window, as shown in Figure 5-7.

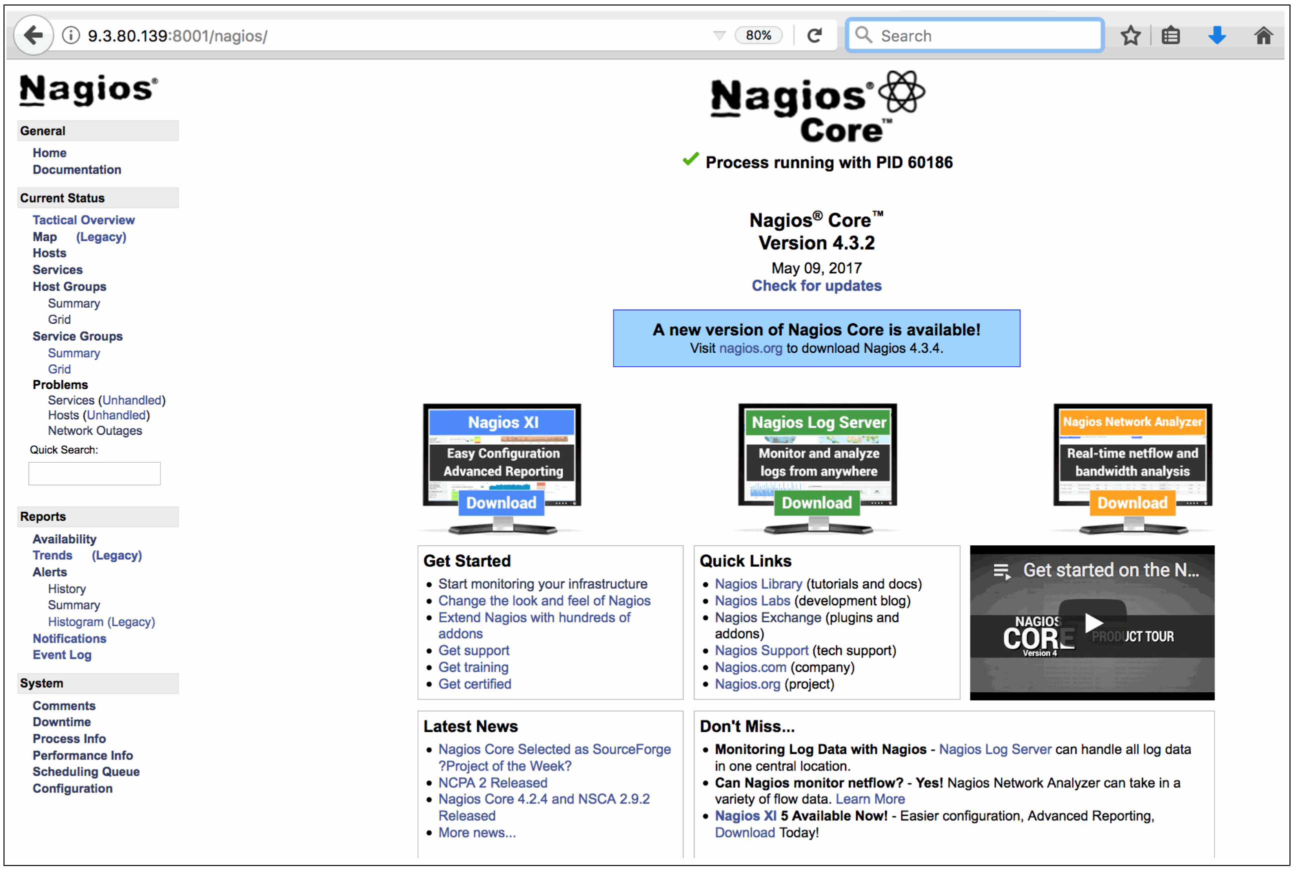


Figure 5-7 Nagios Home window

Figure 5-7 shows the four main menus, and each of the menus has its own submenus:

- General: This menu option takes you back to the Home window and the Nagios documentation website.
- Current Status: This menu shows the state of monitored servers and services (that can be grouped in Host Groups or Service Groups) and problems happening in your system (if any).
- Reports: This menu helps you create reports of events, availability of systems, and services and trace trends of instability, which are based on past events.
- System: This administration menu can be used to configure the commands that are used to perform checks on the hosts, add comments that other administrators can see in the Nagios interface, restart the Nagios daemon, and see the monitoring queue.

Viewing monitored hosts

In the Nagios main menu, under the **Current Status** menu, click **Hosts**. Nagios Core shows the status of all monitored hosts, as shown in Figure 5-8.

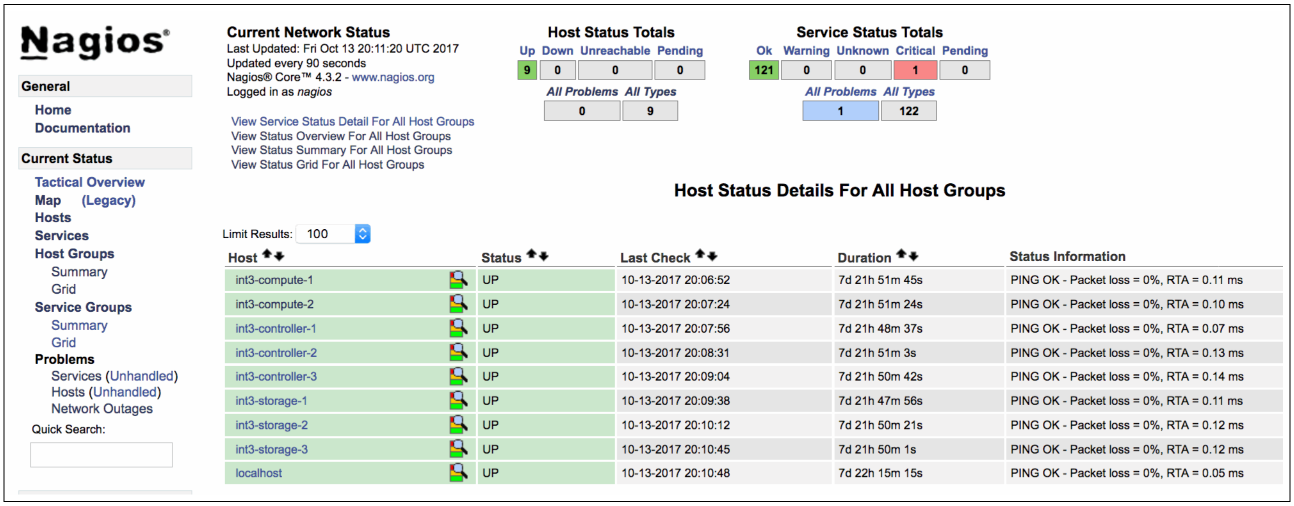


Figure 5-8 Status of the monitored hosts

Figure 5-8 shows a table containing all the monitored hosts, their status, information about when the last check was performed, how long this node has been monitored by Nagios, and the ping loss information.

You can click any of the nodes to obtain more information and perform Nagios actions. After clicking a node, several details are displayed, as shown in Figure 5-9.

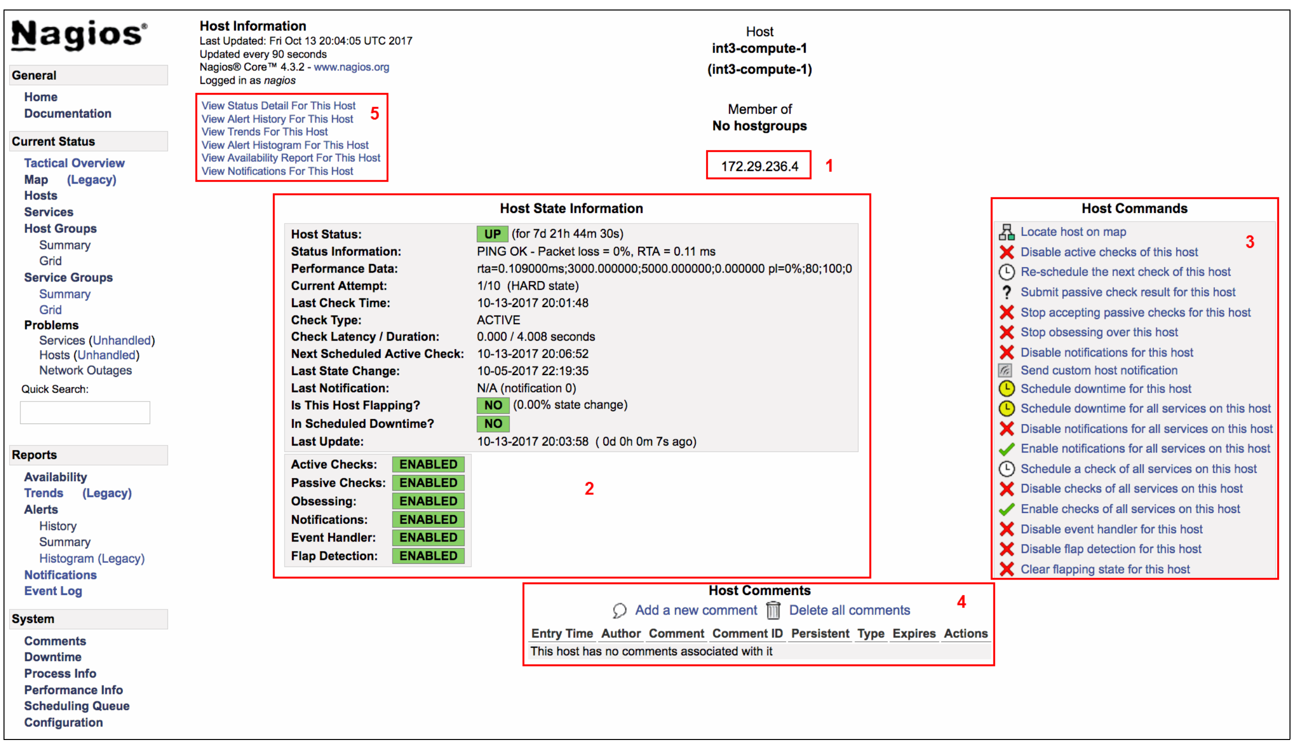


Figure 5-9 Host information interface in the Nagios Core

The information that is displayed in Figure 5-9 on page 138 is explained in the following list:

1. Displays the IP address of the monitored host that is selected.
2. Shows details of the Host State Information, status, last time the check was performed, performance date, last time a notification was issued for this node, and shows whether monitoring is enabled or disabled for this host.
3. The Host Commands area helps perform Nagios commands for the specified host, such as disabling or managing the schedule of checks and notifications for the host.
4. Helps the administrator to add comments for this host so that other users of the Nagios interface are aware of its status. For example, the system administrator can add a comment saying that this node is under maintenance, so operators do not try to perform recovery actions if they see alarms for this host.
5. Takes you to actions that are performed under the Reports area, for example, generating an availability report, and viewing the alert history and trends for the host.

Viewing monitored services

In the main menu, under the **Current Status** section, click **Services**. Figure 5-10 shows the services that are monitored by Nagios.

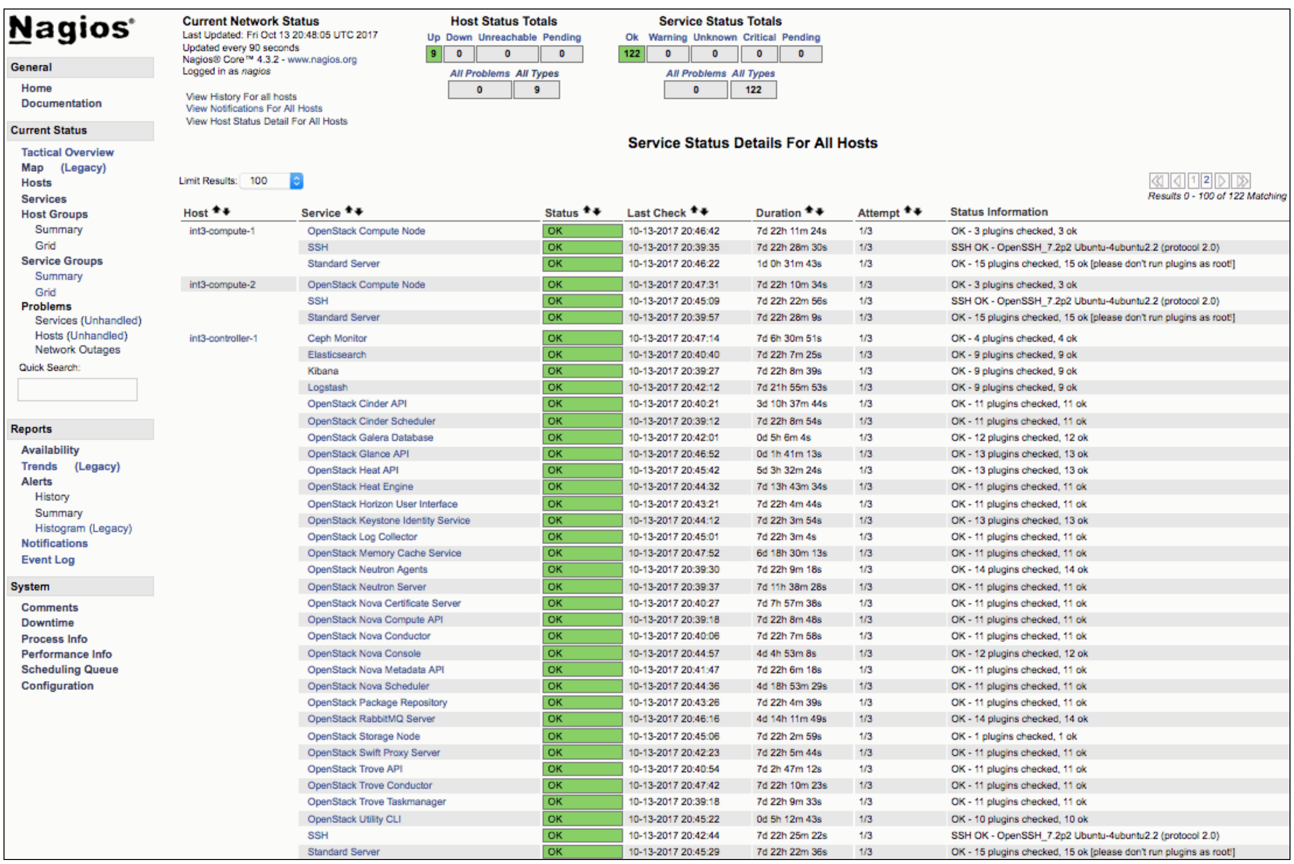


Figure 5-10 Services that are monitored by Nagios

The controller node has many services that are monitored by Nagios. The compute nodes also have components that are monitored, but because most of the components run in the controller nodes, they have most of the monitored services.

Note: The VMs where the open databases run are not monitored by Nagios at this time. The Open Platform for DBaaS on Power Systems infrastructure is monitored.

If you click any of the monitored services, Nagios displays details of the monitored item, status, status information of the checks that have been performed and service commands to control the schedule of the service checks, obtains reports of the service availability and notification history, and enables you to add comments. Figure 5-11 shows an example of the details of monitoring the Elasticsearch container and services on the controller node.

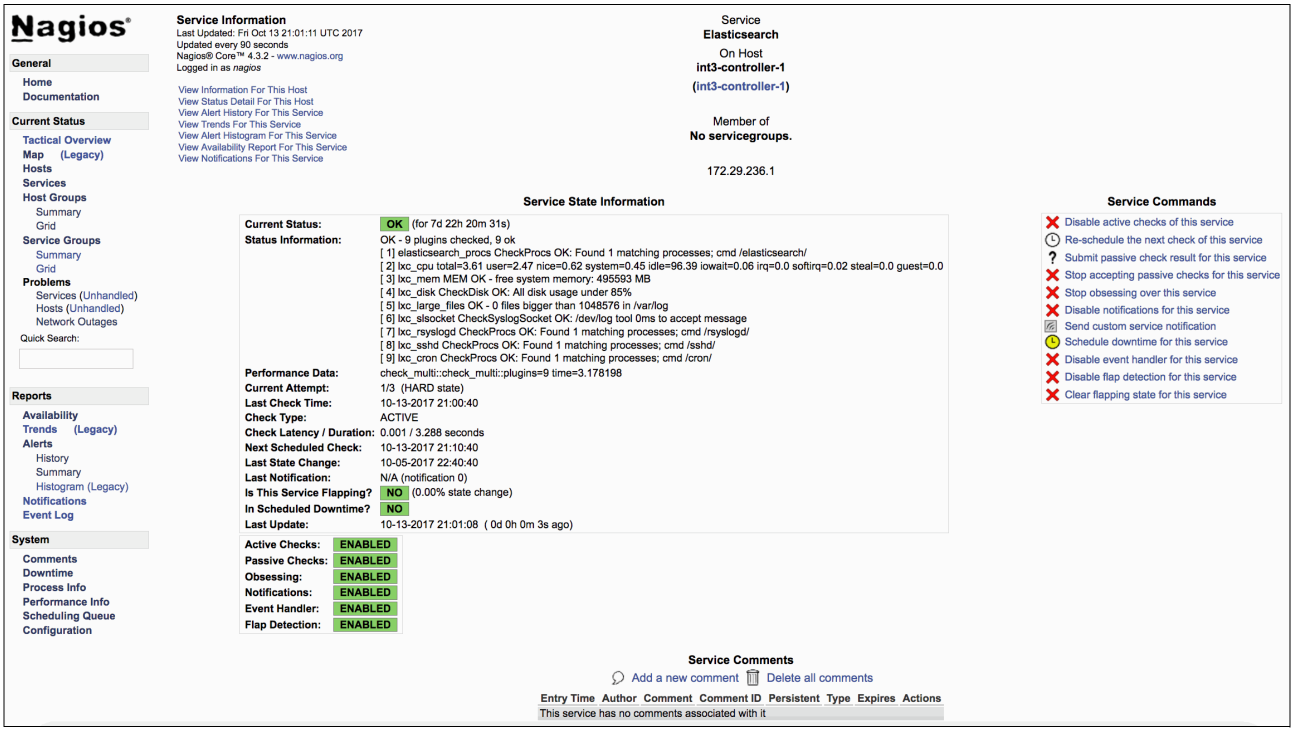


Figure 5-11 Elasticsearch service monitoring by Nagios Core

Viewing problems

Under the Current Status area of your Nagios, you can see the Problem section. To check any problem that is detected by Nagios, use the appropriate link in this section so that you can view the problems with Services, Hosts, or Networks that Nagios detects, as shown in Figure 5-12.

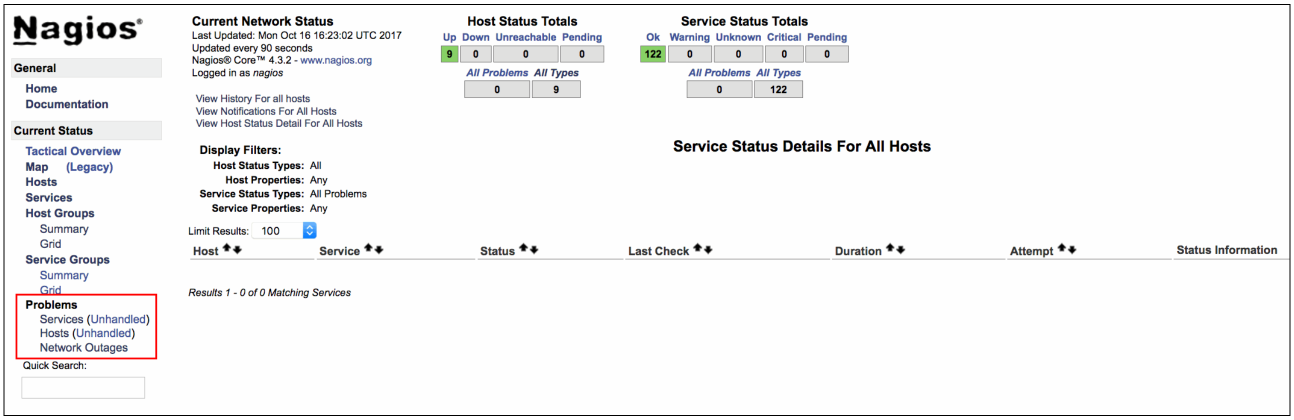


Figure 5-12 The Nagios Problem menu

Figure 5-13 shows an example where Nagios detects critical errors in services running in the controller nodes.

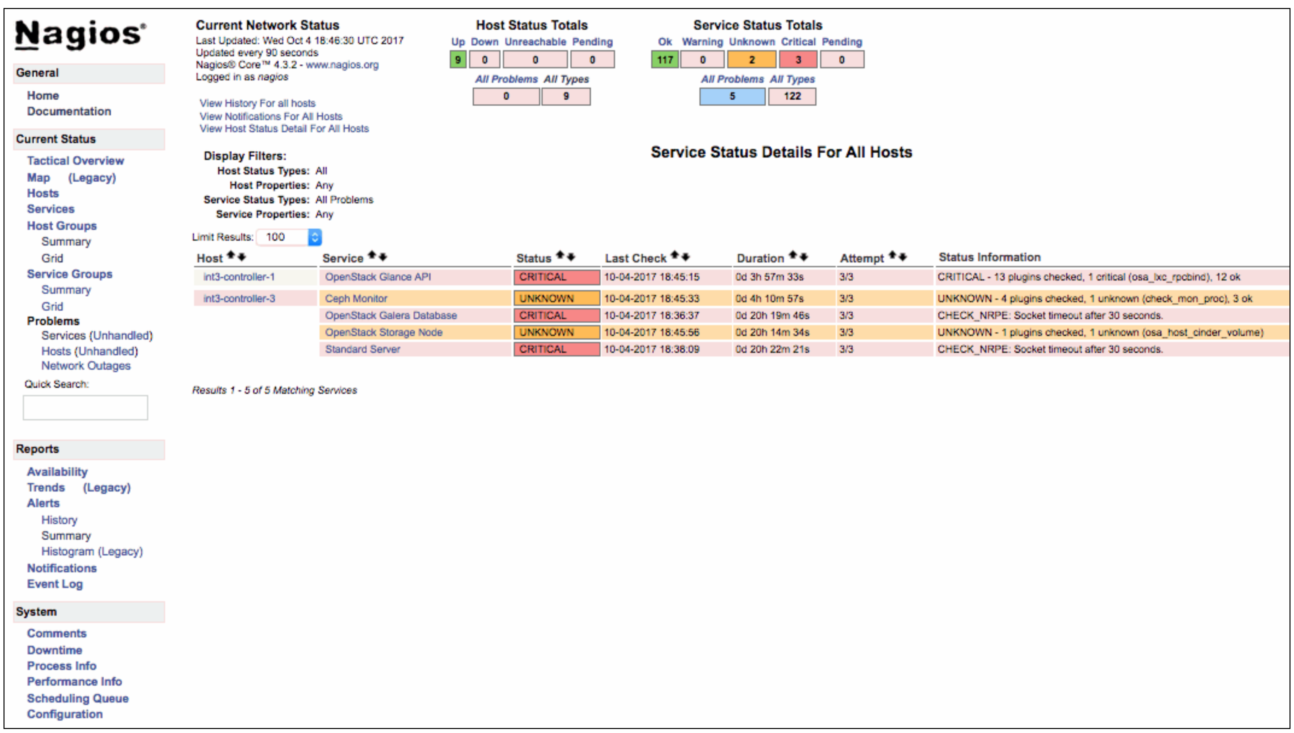


Figure 5-13 Service problems that are detected by Nagios

Obtaining reports

You can obtain availability reports or view the history of all alerts for a specific host, service, host group, or service group. To generate an availability report showing all the issues that are detected by Nagios in the controller Node 1 during the last month, complete the following steps:

1. Click **Availability** under the **Reports** section. Select the target component to obtain the report. Your selection can be **Hosts**, **Hostgroups**, **Services**, or **Servicegroups**. Click **Continue to step 2**, as shown in Figure 5-14.

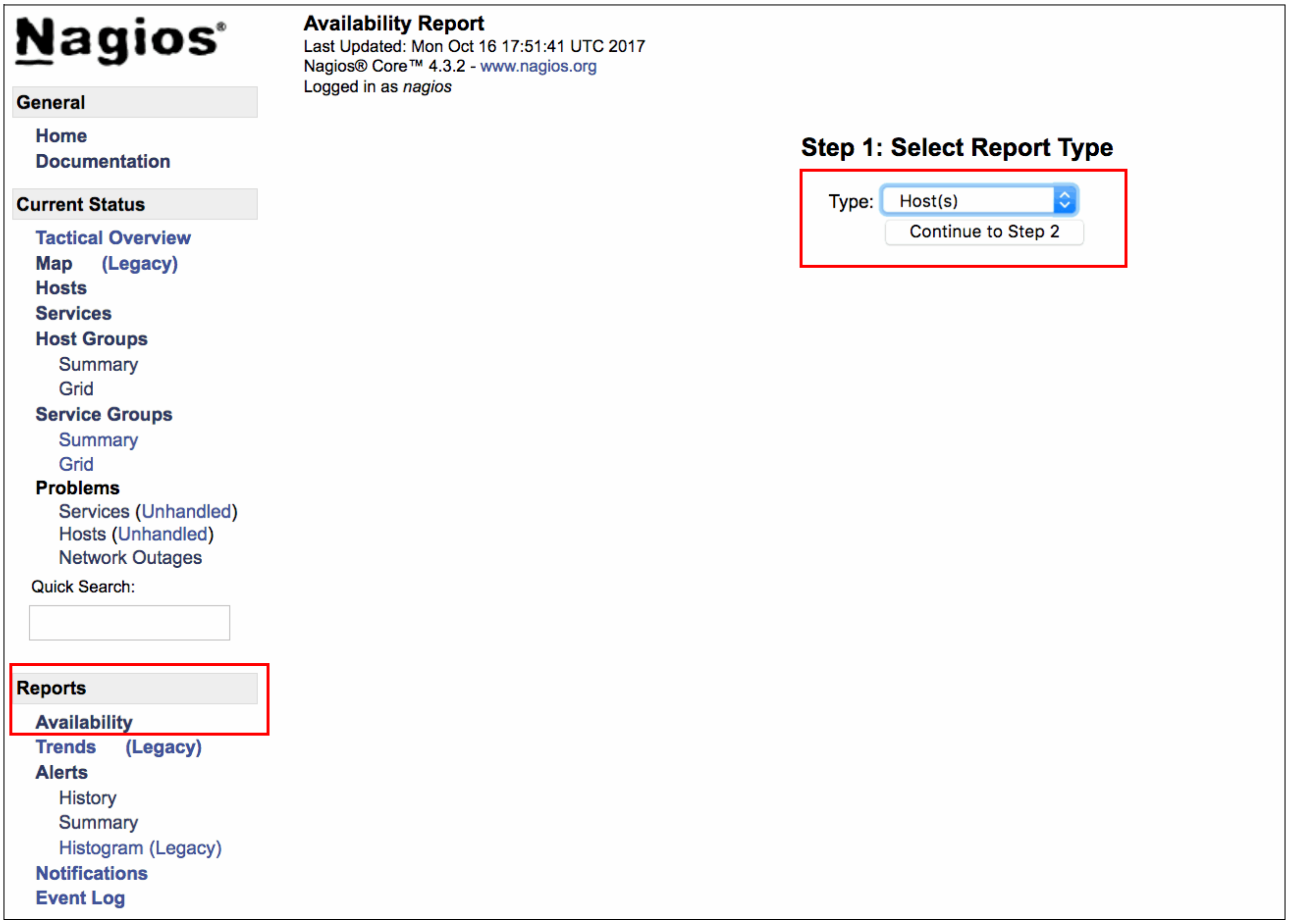


Figure 5-14 Obtaining an availability report

2. Select the host to generate the report, and click **Continue to step 3**, as shown in Figure 5-15 on page 143.

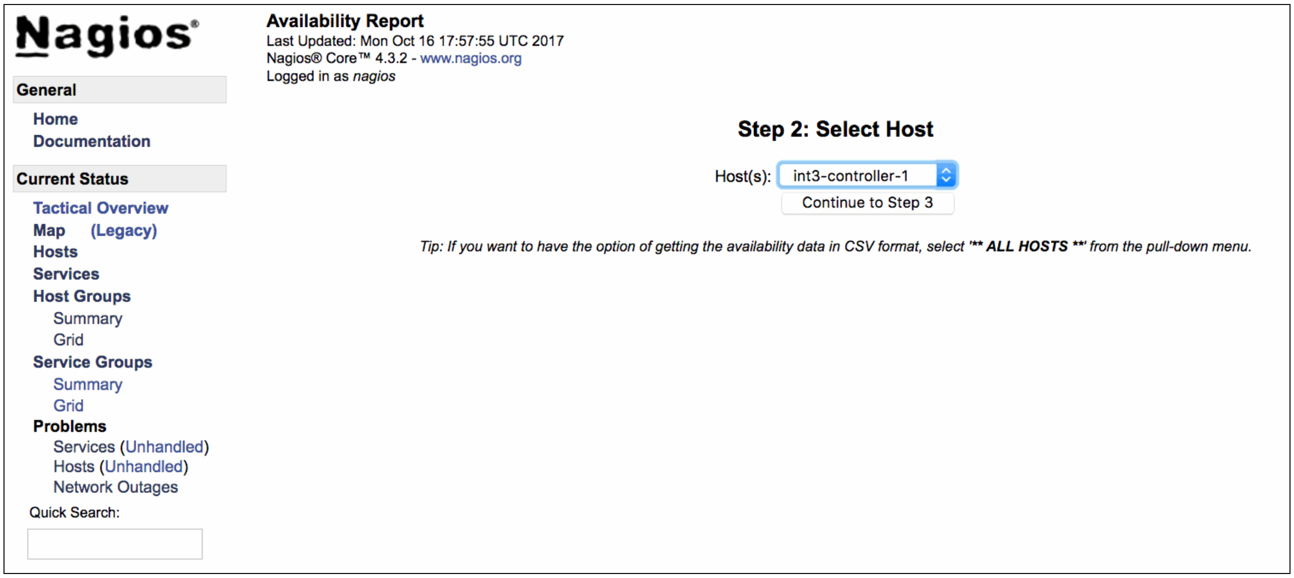


Figure 5-15 Choose the host to generate the report

> **Tip:** You can optionally select ** **ALL HOSTS** ** in this menu to generate reports from all hosts that are monitored by this Nagios instance.

3. Select then the length of the report. Figure 5-16 shows the creation of a report from the past month.

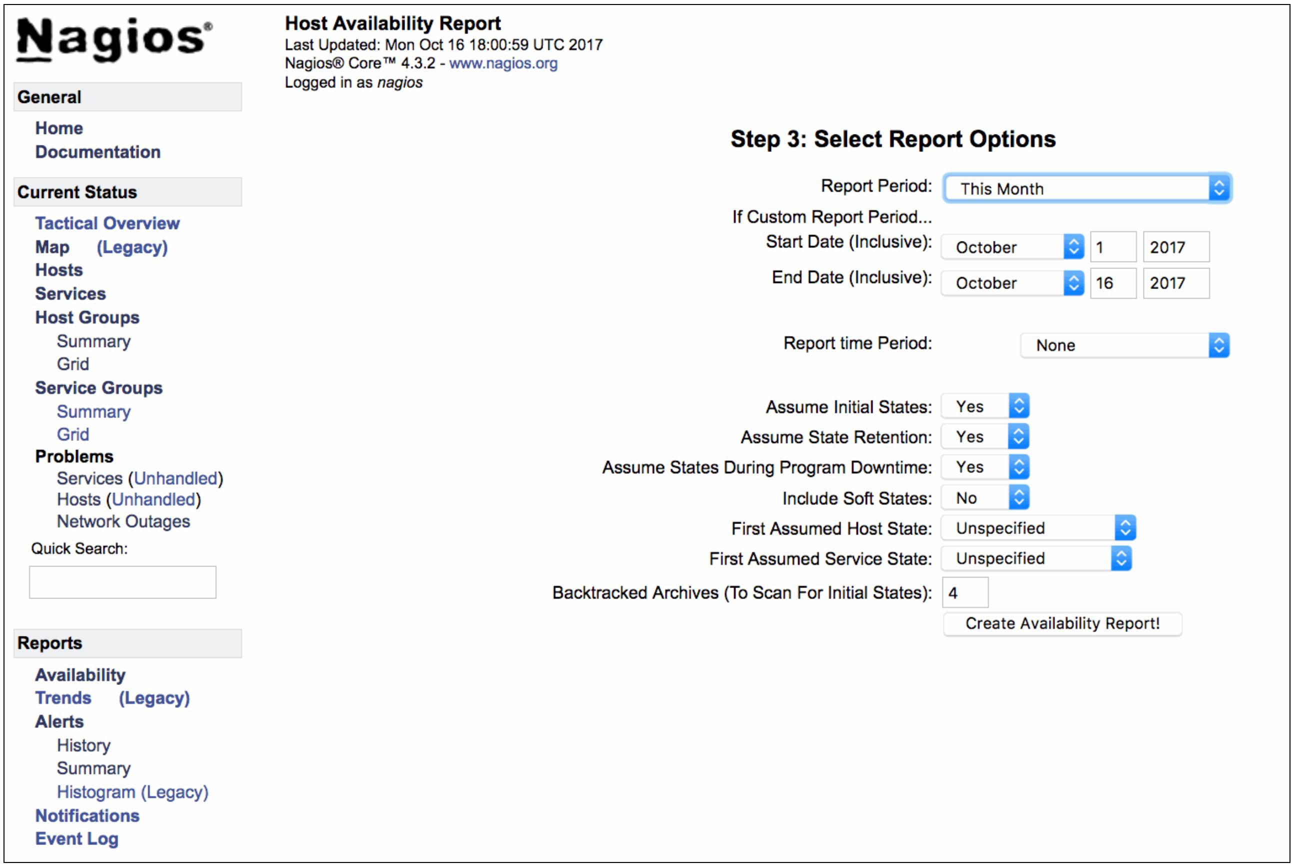


Figure 5-16 Selecting the length of the report

4. Click **Create Availability Report!**. Nagios generates a report from all the services in this host for the past month, as shown in Figure 5-17.

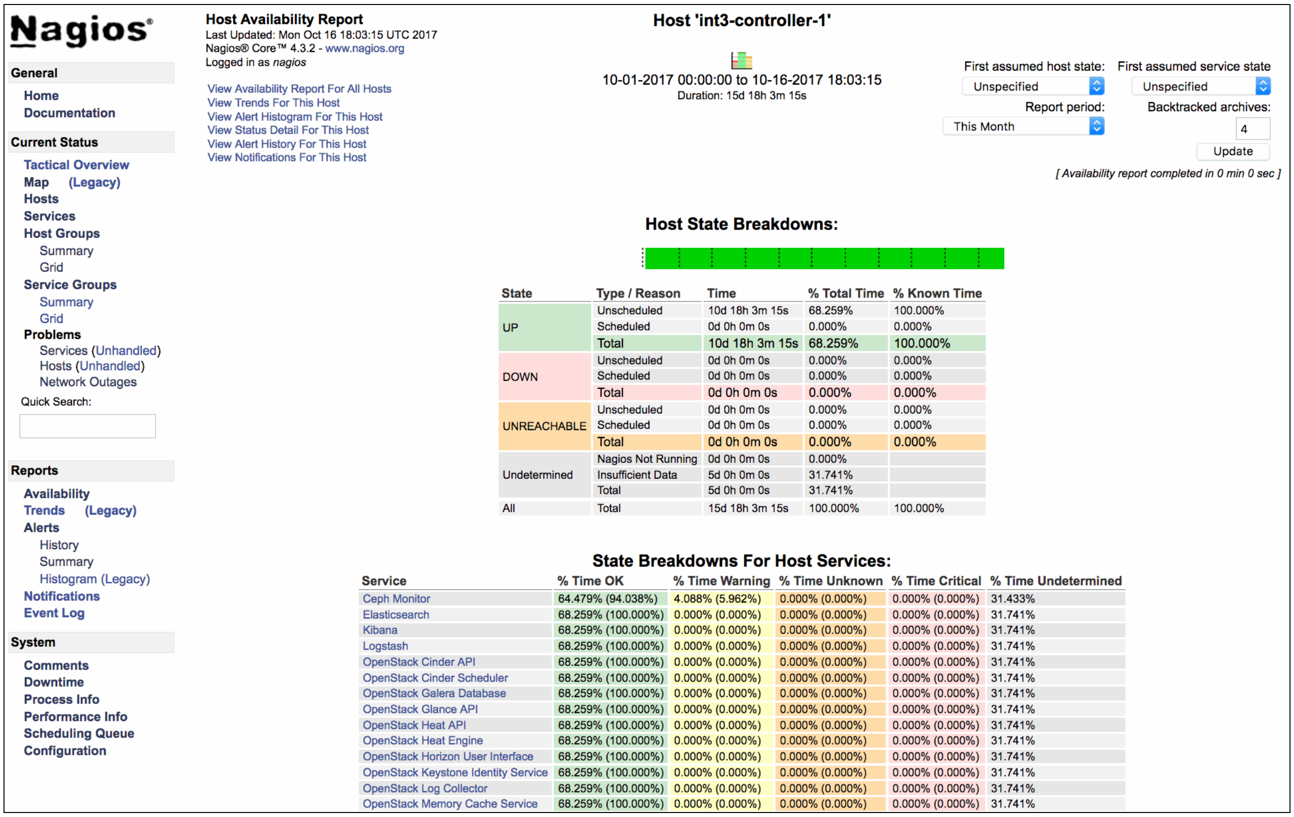


Figure 5-17 Availability report for past month for controller node 1

Accessing the system configuration

Under the Systems area, you can perform administrative tasks in Nagios and its monitored hosts and services. To schedule a host's downtime so that all operators can see an alert on Nagio and avoid unwanted page-outs during a maintenance window, click **Downtime**, click **Schedule service downtime** or **Schedule host downtime**, and complete the information of the affected Host or Service, and the time and end that the maintenance window starts and ends, as shown in Figure 5-18.

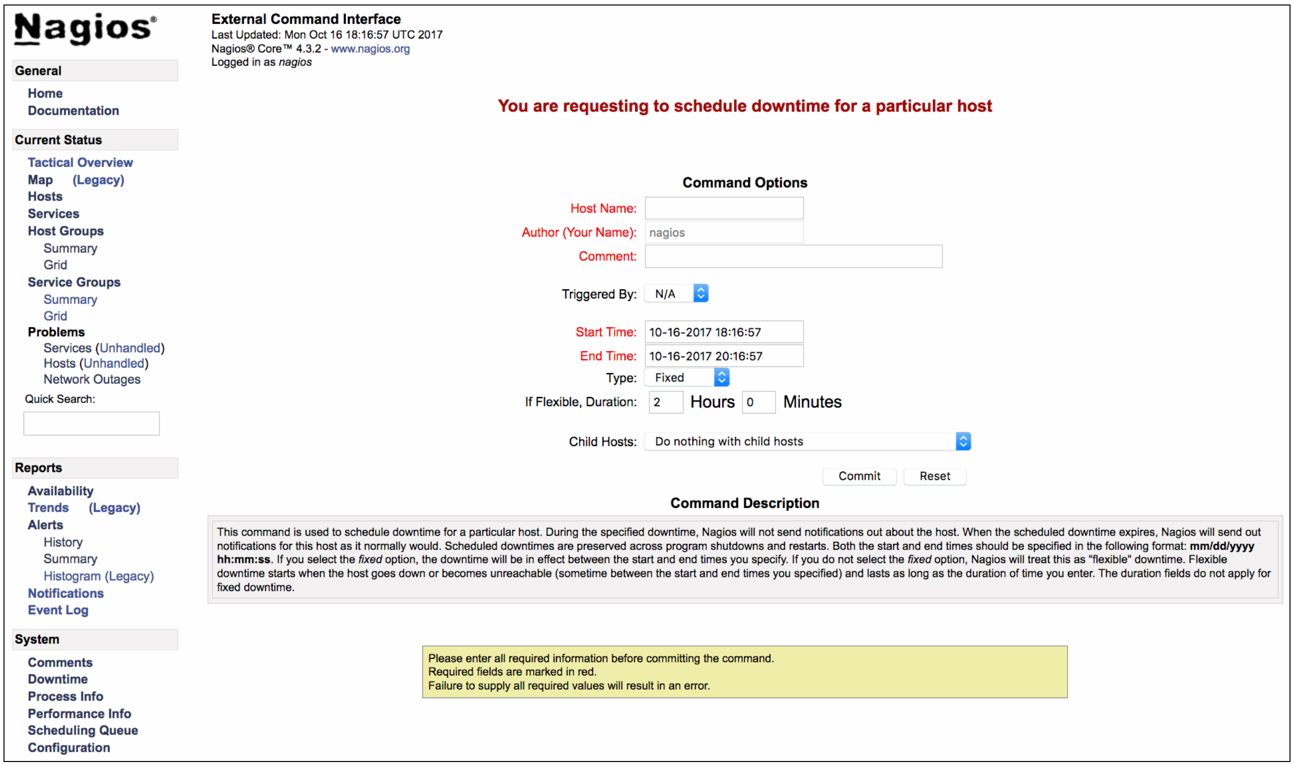


Figure 5-18 Scheduling the host downtime

You can also restart the Nagios, temporarily disable notifications, and stop checking services and hosts clicking **Process Info** under the **Systems** area, as shown in Figure 5-19.

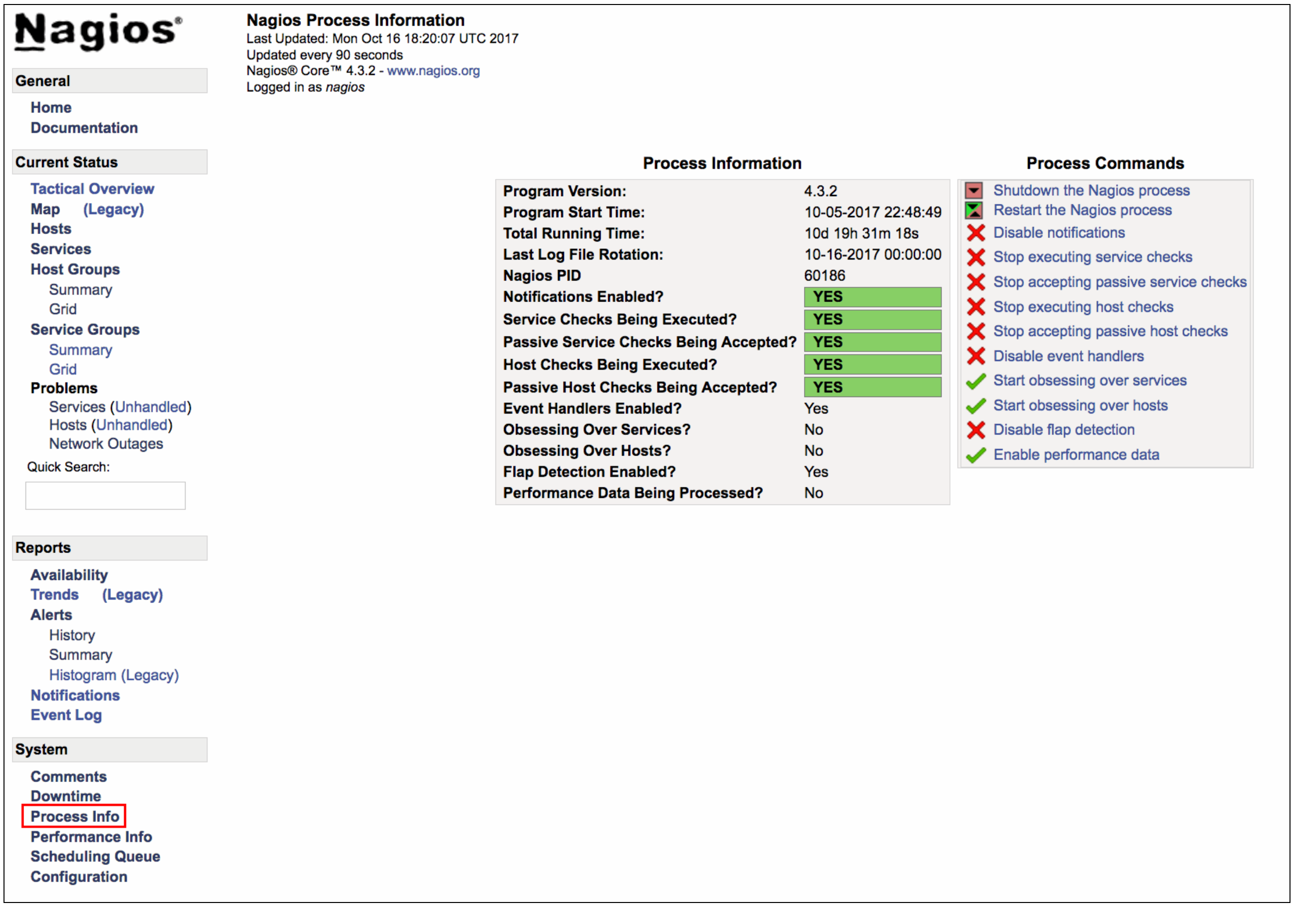
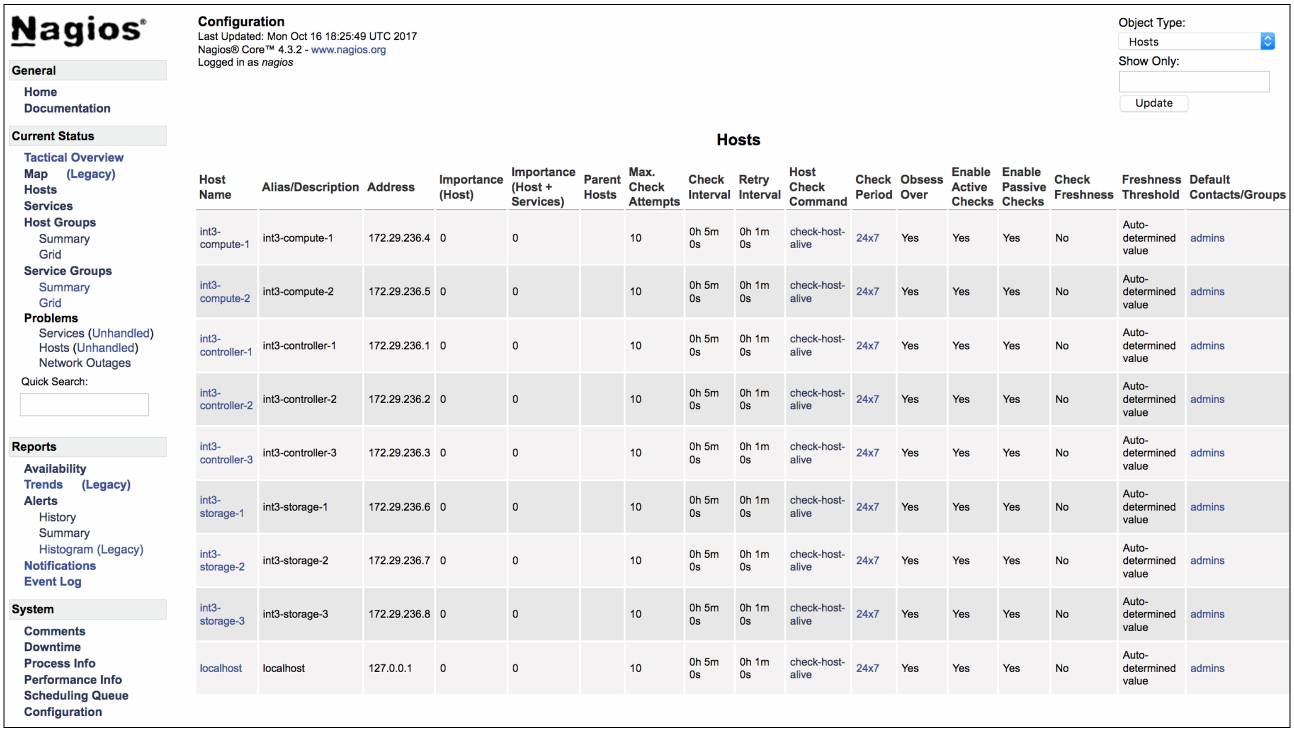


Figure 5-19 Administering Nagios process information

You can check the status, disable monitoring, or change the schedule of individual checks that are performed in Hosts or Services in the Scheduling Queue area, as shown in Figure 5-20.

Figure 5-20 Checking or changing the scheduling queue

In the Configuration section, you can view and modify the NRPE commands that are used to check the status of hosts and services and the period that they can be used for monitoring. You can also view and change the default contact for notification purposes in case of an event that is detected by Nagios, as shown in Figure 5-21.



Host Name	Alias/Description	Address	Importance (Host)	Importance (Host + Services)	Parent Hosts	Max. Check Attempts	Check Interval	Retry Interval	Host Check Command	Check Period	Obsess Over	Enable Active Checks	Enable Passive Checks	Check Freshness	Freshness Threshold	Default Contacts/Groups
int3-compute-1	int3-compute-1	172.29.236.4	0	0		10	0h 5m 0s	0h 1m 0s	check-host-alive	24x7	Yes	Yes	Yes	No	Auto-determined value	admins
int3-compute-2	int3-compute-2	172.29.236.5	0	0		10	0h 5m 0s	0h 1m 0s	check-host-alive	24x7	Yes	Yes	Yes	No	Auto-determined value	admins
int3-controller-1	int3-controller-1	172.29.236.1	0	0		10	0h 5m 0s	0h 1m 0s	check-host-alive	24x7	Yes	Yes	Yes	No	Auto-determined value	admins
int3-controller-2	int3-controller-2	172.29.236.2	0	0		10	0h 5m 0s	0h 1m 0s	check-host-alive	24x7	Yes	Yes	Yes	No	Auto-determined value	admins
int3-controller-3	int3-controller-3	172.29.236.3	0	0		10	0h 5m 0s	0h 1m 0s	check-host-alive	24x7	Yes	Yes	Yes	No	Auto-determined value	admins
int3-storage-1	int3-storage-1	172.29.236.6	0	0		10	0h 5m 0s	0h 1m 0s	check-host-alive	24x7	Yes	Yes	Yes	No	Auto-determined value	admins
int3-storage-2	int3-storage-2	172.29.236.7	0	0		10	0h 5m 0s	0h 1m 0s	check-host-alive	24x7	Yes	Yes	Yes	No	Auto-determined value	admins
int3-storage-3	int3-storage-3	172.29.236.8	0	0		10	0h 5m 0s	0h 1m 0s	check-host-alive	24x7	Yes	Yes	Yes	No	Auto-determined value	admins
localhost	localhost	127.0.0.1	0	0		10	0h 5m 0s	0h 1m 0s	check-host-alive	24x7	Yes	Yes	Yes	No	Auto-determined value	admins

Figure 5-21 Nagios configuration menu

This section demonstrated some of the Nagios functions that are immediately available after the deployment of the Open Platform for DBaaS on Power Systems solution. For more information about Nagios and instructions of usage, see the Nagios documentation.

5.4 Elastic stack (Kibana)

As explained in 5.1, “Introduction to cluster monitoring and troubleshooting” on page 124, the ELK Stack is formed by several components:

- Metricbeat: An agent that runs in the compute, Ceph, and Swift nodes and monitors and collects metrics (disk usage, CPU workload, available memory, and so on) and sends them to the Logstash.
- Filebeat: An agent that also runs in the compute, Ceph, and Swift nodes and collects logs from the OS, OpenStack components, and Ceph components, and sends them to the Logstash.
- Logstash: An application that runs in an LXC container in the controller nodes. This application receives the information that is provided by Metricbeat and Filebeat agents, centralizing and transforming such information. It standardizes and indexes such information in a way that makes it easier to handle multiple logs and statistics from different sources by using different formats. All the information that is prepared by Logstash is sent to Elasticsearch.

- Elasticsearch: An application running in an LXC container in the controller nodes. Elasticsearch is a search and analytics engine, which is accessible from RESTful web interfaces, that enables queries on the data that is stored (received by Logstash).
- Kibana: A web interface that consumes data that is stored in Elasticsearch. Kibana runs in a container in the controller nodes. Through the Kibana interface, you can have multiple Dashboards with useful information, such as logs and statistics from your cluster nodes, that you can use to make decisions and understand when events are happening in your environment.

Figure 5-22 shows the flow of the information from the nodes by using Beats in the Operation Management tools in ELK Stack.

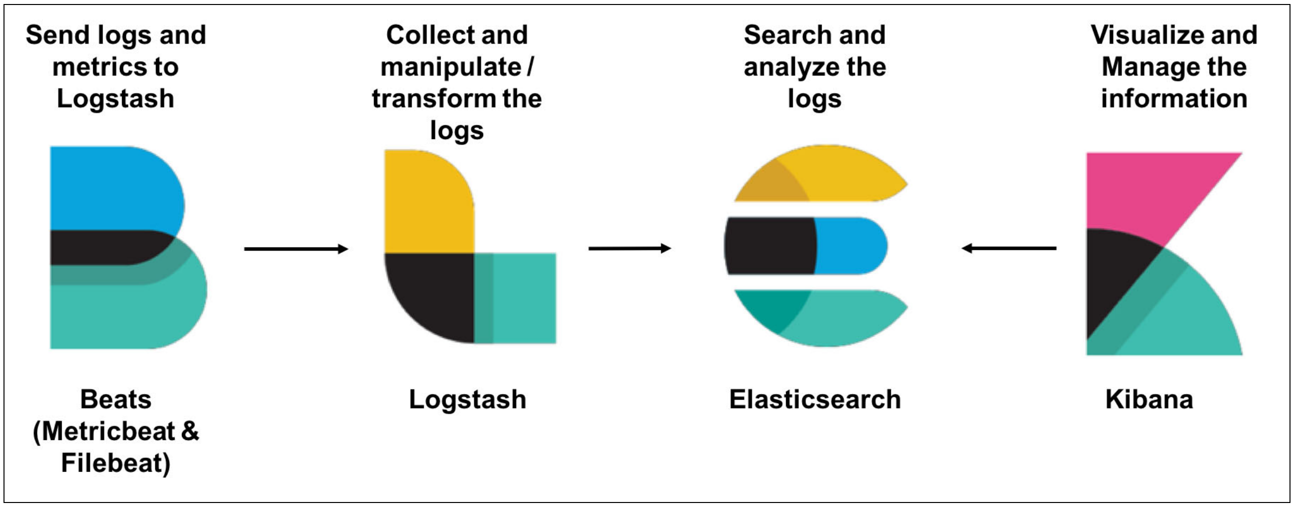


Figure 5-22 Information flow in the ELK Stack

Example 5-12 shows the Filebeat and Metricbeat components running in one of the compute nodes of the solution.

Example 5-12 Filebeat and Metricbeat running in a compute node

```
ubuntu@int3-compute-1:~$ ps -ef | grep beat
ubuntu     40982  40962  0 16:13 pts/2    00:00:00 grep --color=auto beat
root       97692      1  0 Sep28 ?        00:08:42
/usr/share/metricbeat/bin/metricbeat -c /etc/metricbeat/metricbeat.yml -path.home
/usr/share/metricbeat -path.config /etc/metricbeat -path.data /var/lib/metricbeat
-path.logs /var/log/metricbeat
root       98980      1  0 Sep28 ?        00:01:15 /usr/share/filebeat/bin/filebeat
-c /etc/filebeat/filebeat.yml -path.home /usr/share/filebeat -path.config
/etc/filebeat -path.data /var/lib/filebeat -path.logs /var/log/filebeat
```

In the `/etc/filebeat/filebeat.yml` configuration file, you can see that all information that is collected by the agent is sent to the Logstash, as shown in Example 5-13.

Example 5-13 Filebeat configuration file sending information to Logstash

```
output:
  logstash:
    hosts:
    - '172.29.236.50:5044'
shipper: null
logging:
```

```
  files:
    rotateeverybytes: 10485760  # = 10MB
    keepfiles: 7
filebeat.config_dir: /etc/filebeat/conf
```

You can see that the IP that is configured in `filebeat.yml` (`172.29.236.50` in Example 5-13 on page 150) is available in one of the controller nodes. The reason is that all components point to this IP, and the haproxy component running in the controller node coordinates the routing of such information to the appropriate IP address of the Logstash LXC container, as shown in `/etc/haproxy/haproxy.cfg` in Example 5-14.

Example 5-14 The haproxy configuration file

```
frontend logstash-beats-front
bind *:5044
    mode tcp
    timeout client 60m
    option  tcplog
    default_backend logstash-beats-back

backend logstash-beats-back
    mode tcp
    timeout server 60m
    balance leastconn
    server int3-controller-1-logstash 172.29.236.11:5044 check port 5044 inter 10s
fall 1 rise 1
    server int3-controller-2-logstash 172.29.236.15:5044 check port 5044 inter 10s
fall 1 rise 1
    server int3-controller-3-logstash 172.29.236.19:5044 check port 5044 inter 10s
fall 1 rise 1
```

Metricbeat sends the information directly to Elasticsearch, as shown in the section of `/etc/metricbeat/metricbeat.yml` configuration file in Example 5-15.

Example 5-15 Metricbeat configuration file

```
metricbeat.modules:
- module: system
  metricsets:
    - cpu
    - load
    - fsstat
    - memory
    - network
  enabled: true
  period: "1m"
  processes: ['.*']
- module: system
  metricsets:
    - filesystem
    - process
  enabled: true
  period: "5m"
  processes: ['.*']
output.elasticsearch:
```

```
    hosts:
    - '172.29.236.50:9200'
```

The haproxy also routes this information to the Elasticsearch LXC container, as shown in the section of `/etc/haproxy/haproxy.cfg` in Example 5-16.

Example 5-16 Elasticsearch routing in the haproxy configuration file

```
frontend elasticsearch-http-front
bind *:9200
    mode http
    option httplog
    option forwardfor except 127.0.0.0/8
    option http-server-close
    default_backend elasticsearch-http-back

backend elasticsearch-http-back
    mode http
    option forwardfor
    option httpchk
    option httplog
    balance source
    server int3-controller-1-elasticsearch 172.29.236.12:9200 check port 9200
inter 10s fall 1 rise 1
    server int3-controller-2-elasticsearch 172.29.236.16:9200 check port 9200
inter 10s fall 1 rise 1
    server int3-controller-3-elasticsearch 172.29.236.20:9200 check port 9200
inter 10s fall 1 rise 1
```

All this information is processed by Logstash and stored in Elasticsearch, which run in LXC containers in the controller nodes, as shown in Example 5-17.

Example 5-17 LXC Containers running with Kibana, Logstash, and Elasticsearch

```
root@int3-controller-1:~# lxc-ls
int3-controller-1-elasticsearch          int3-controller-1-kibana
int3-controller-1-logstash
...
```

The Kibana software also runs in an LXC container, and it can be the web interface that is used for consuming information that is stored in Elasticsearch. Through Kibana, you can create graphics, view logs, and statistics of your Open Platform for DBaaS on Power Systems infrastructure. Use the instructions that are provided in 5.2, “Accessing the operations management tools” on page 127 to open the Kibana interface.

5.4.1 Using the Kibana Dashboard

As soon as you log in to the Kibana, the Kibana Dashboard is displayed, as shown in Figure 5-23.

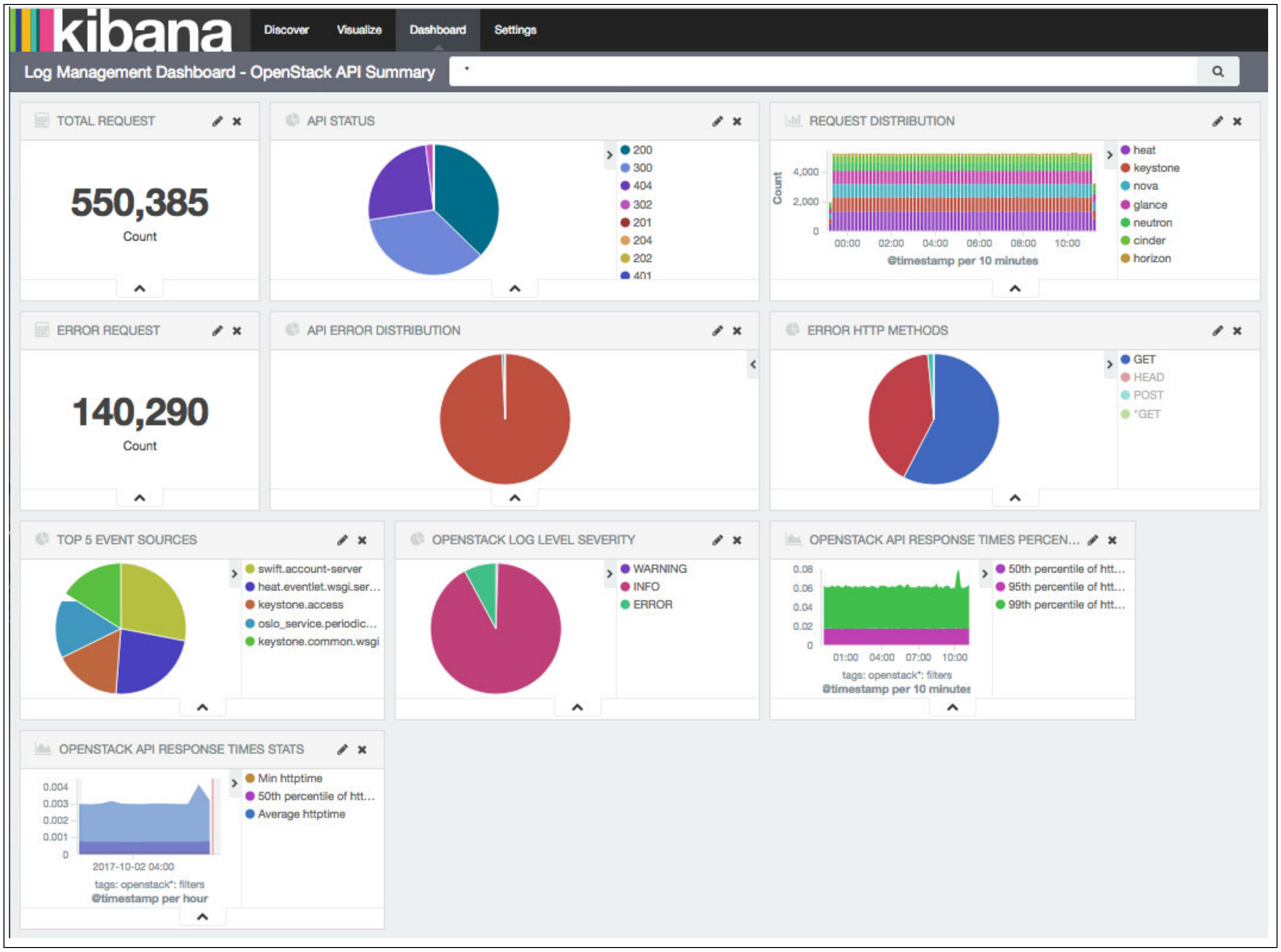


Figure 5-23 The Kibana Dashboard

In the Kibana Dashboard, you can add views of searches, metrics, and graphics to monitor the Open Platform for DBaaS on Power Systems environment. There is information that is already displayed, but you can customize the view according to your preferences and needs. After the initial deployment of the Open Platform for DBaaS on Power Systems solution, the following topics are initially displayed in the Kibana Dashboard:

- Total Requests
- Error Request
- API Status
- Request Distribution
- API Error Distribution
- Error HTTP Methods
- Top 5 Event Sources
- OpenStack Log Level Severity
- OpenStack API Response Times Percentage
- OpenStack API Response Times Stats

To understand the information that is displayed in each box, you can view the source of this data and how Kibana is projecting this information. Click the **Edit** icon (identified by a pencil) of the box you want to open, as shown in Figure 5-24.

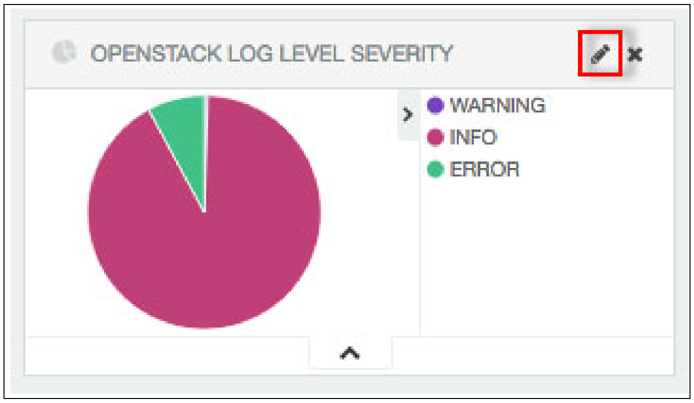


Figure 5-24 Editing the visualization

The upper level menu changes from Dashboard to Visualize, and several pieces of information are displayed, as shown in Figure 5-25, and explained in the following list.

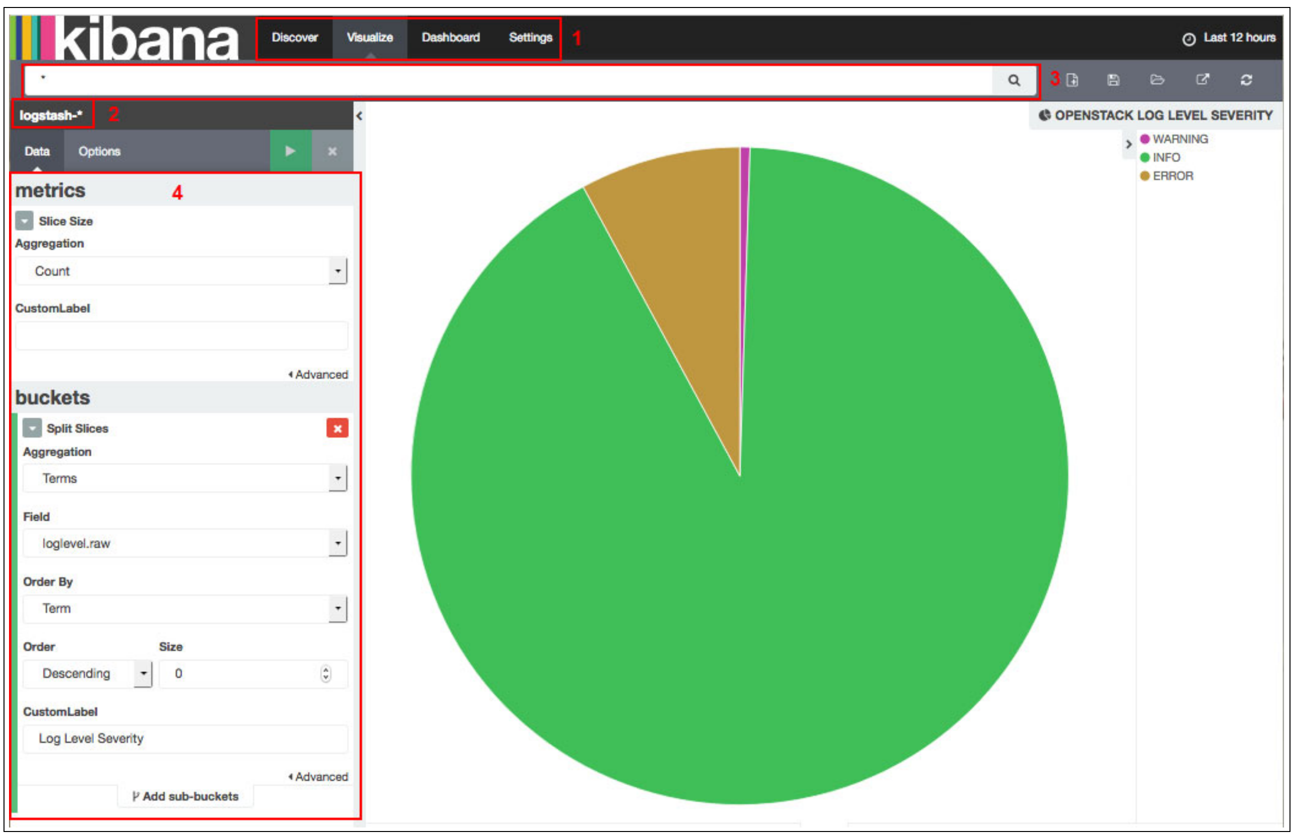


Figure 5-25 Editing the visualization

1. The upper menu shows which section of Kibana you are using. There are four tabs you can use:
   a. Discover: This is where you can perform searches on the logs and metrics that are provided by Filebeat and Metricbeat. Search can be used to later create a Visualization on the Visualize tab.
   b. Visualize: This is the tab that is shown in Figure 5-25 on page 154. Here, you can create graphics and visualizations that are based on the information that is provided by Filebeat and Metricbeat. The visualization can use information that is filtered by searches that are performed in the Discover tab.
   c. Dashboard: A Dashboard where you can add multiple visualizations or searches that were previously created. The information is dynamically displayed as the information is received in Logstash by Filebeat and Metricbeat from the components of the Open Platform for DBaaS on Power Systems solution. You can add multiple visualizations to the existing Dashboards or create your own Dashboard with the visualizations you want to display.
   d. Settings: Here you can control several settings of the Kibana interface.
2. The source of this information. The source can be Logstash (uses information that is provided by Filebeat) or Metricbeat. When you create a visualization, you are prompted to choose whether you want to use Logstash or Metricbeat. This visualization uses the Logstash information.
3. shows the search that is performed in Discover to obtain the information for which you are creating the visualization. In this case, all the messages that are saved in Logstash are used.

4. How the data is used to create the graphic. When you start creating the visualization, you must choose between multiple options, and in this case, a Pie Chart is chosen. In this example, each slice size is based on the Count of entries, and what determines the classification of each slice is a field of the logs called `loglevel.raw` (obtained through the search “*” that is shown in step 3 on page 155). This is possible because Logstash receives the logs through Filebeat and manipulates it, indexing and standardizing the available fields of the logs. Figure 5-26 shows an example of the loglevel field in a specific log entry.

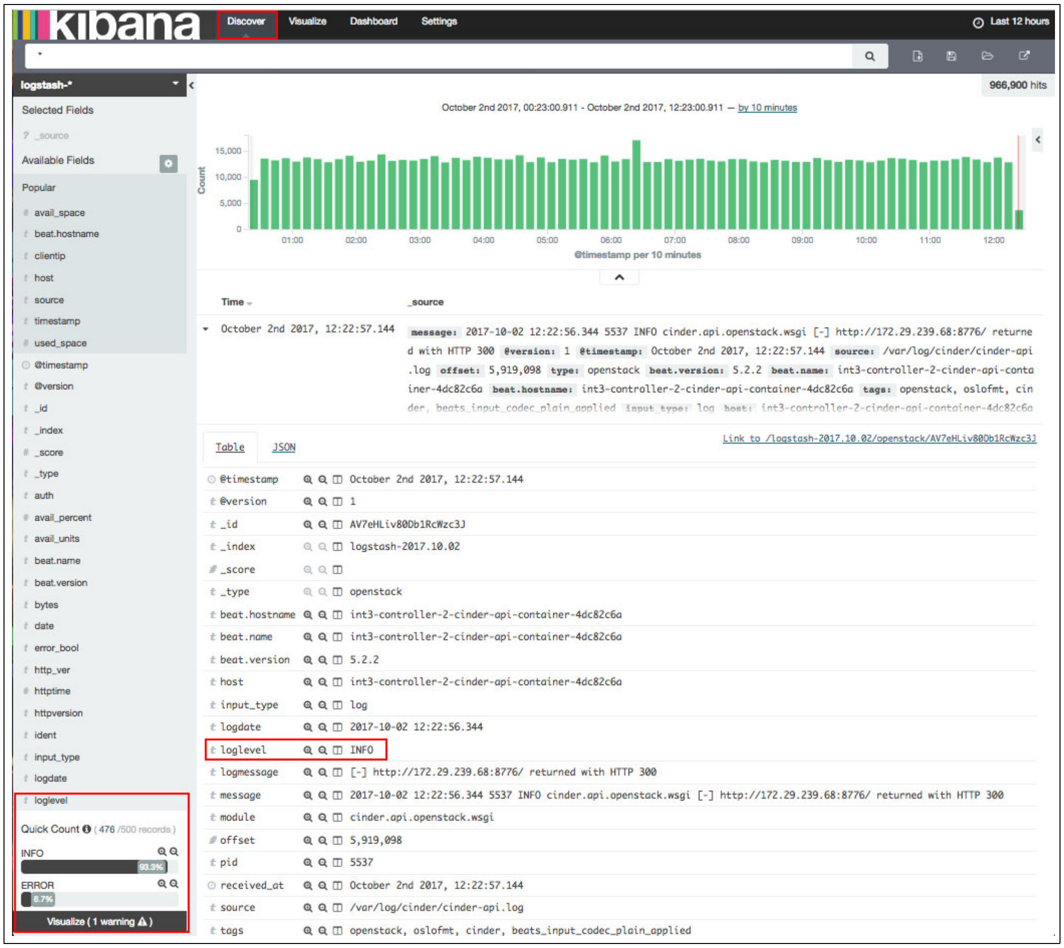


Figure 5-26 Loglevel field of a specific log entry

You can customize the Kibana Dashboard by resizing, adding, or removing the Visualizations so that you can focus on relevant information for your environment. To resize or remove a visualization, position the cursor at the lower right area of the visualization and drag it to resize, or use the “X” button at the upper right area to remove it from the Dashboard, as shown in Figure 5-27.

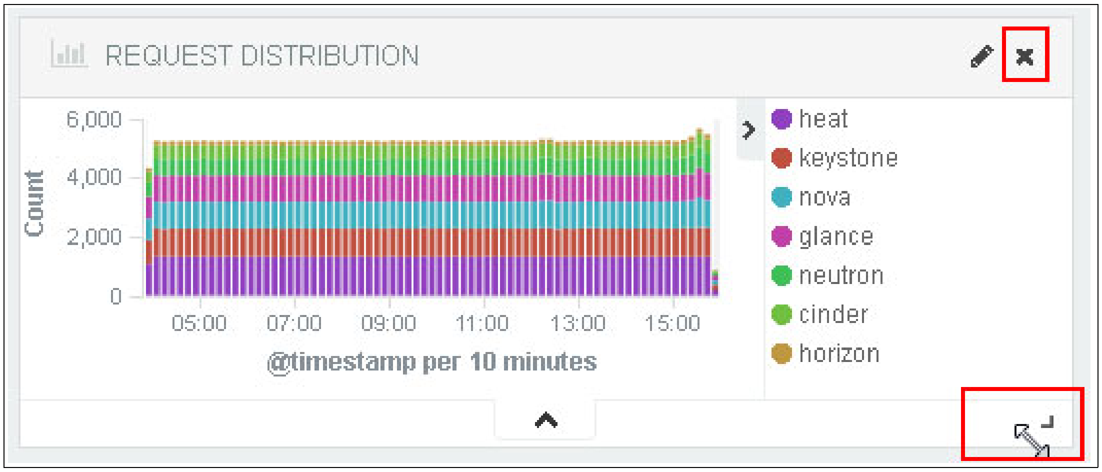


Figure 5-27 Resizing or removing the visualization from the Dashboard

To add a visualization to the existing Dashboard, use the **Add Visualization** button at the upper right area of the Dashboard, as shown in Figure 5-28.

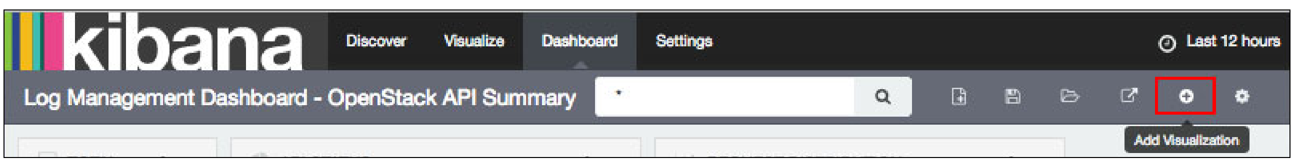


Figure 5-28 Adding a visualization to the Dashboard

You can create a Dashboard and add the Visualizations by using the **New Dashboard** button at the upper right, as shown in Figure 5-29.

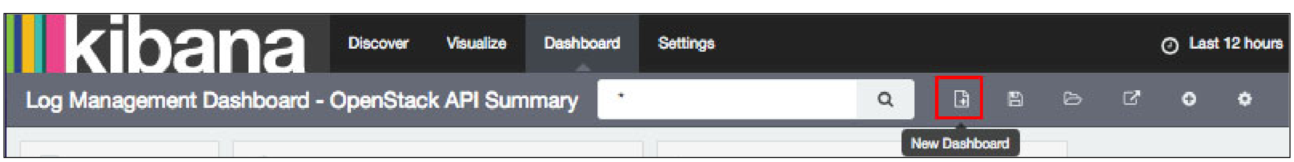


Figure 5-29 Creating a Dashboard

Note: You must be on the Dashboard tab to manipulate the Dashboard. When you are in a different tab, such as Discover or Visualize, the same buttons still exist in the upper right area, but they have different purposes.

5.4.2 Selecting other Dashboards that are available in the Open Platform for DBaaS on Power Systems solution

The Open Platform for DBaaS on Power Systems solution provides several Dashboards with different information previously prepared, including many visualizations to help you monitor your environment. You can use the **Load Saved Dashboards** button at the upper right corner while in the Dashboards tab to load the previously prepared Dashboards that are available in the Open Platform for DBaaS on Power Systems solution, as shown in Figure 5-30.

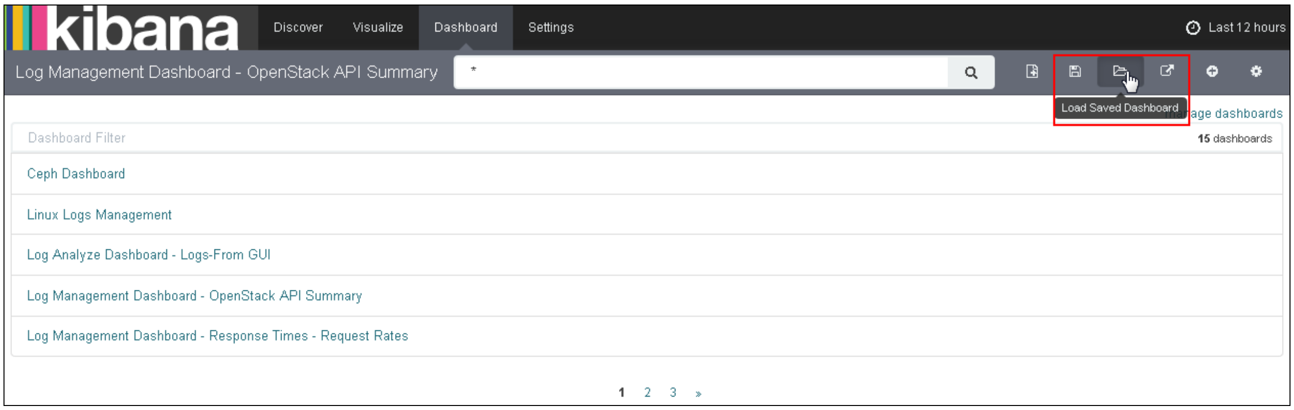


Figure 5-30 Load saved Dashboards

The default Dashboards in the Open Platform for DBaaS on Power Systems solution are:

- Ceph Dashboard
- Linux Logs Management
- Log Analyze Dashboard - Logs from GUI
- Log Management Dashboard - OpenStack API Summary
- Log Management Dashboard - Response Times - Request Rates
- Metricbeat file system per Host
- Metricbeat system overview
- Metricbeat CPU
- Metricbeat File system
- Metricbeat Memory
- Metricbeat Network
- Metricbeat Overview
- Metricbeat Processes
- SWIFT Dashboard
- Trove Dashboard

After clicking the wanted Dashboard, its information is immediately displayed. Figure 5-31 shows an example of the Ceph Dashboard, displaying several Ceph-related pieces of information from this Open Platform for DBaaS on Power Systems environment.

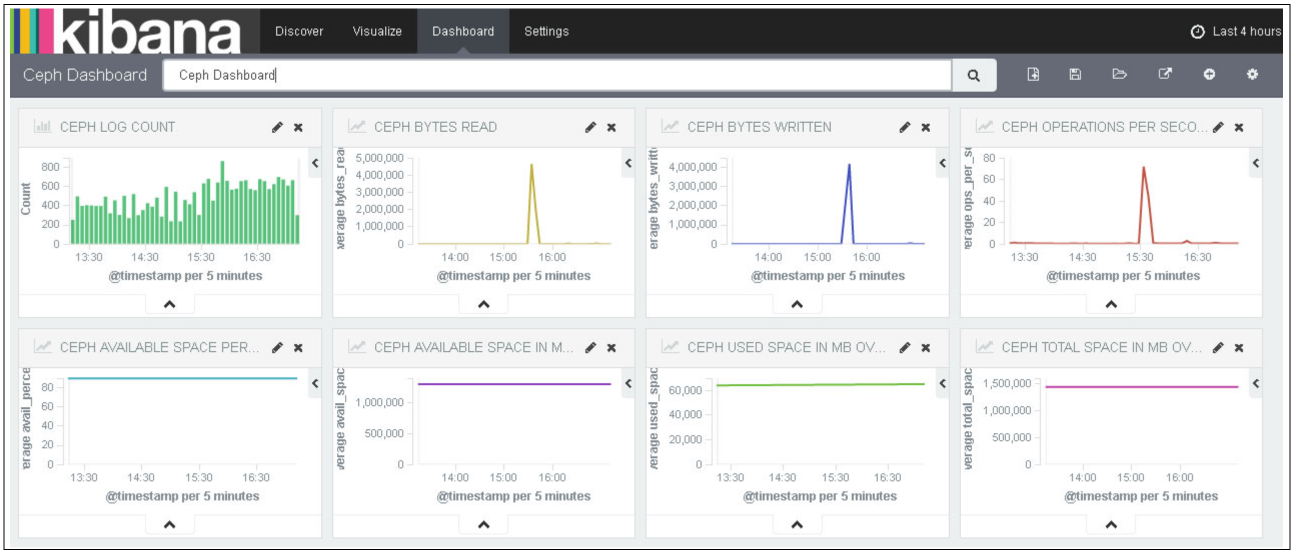


Figure 5-31 Ceph Dashboard

5.4.3 Performing searches with Kibana

You can use Kibana to search through the logs from the entire cluster. Later, you can use information from such logs to generate graphics or visualizations to monitor the Open Platform for DBaaS on Power Systems cluster. You can also use Kibana search for troubleshooting or problem determination purposes. For more information and a troubleshooting example, see 5.4.9, “Using Kibana for troubleshooting” on page 178.

To perform a search by using Kibana, click the **Discover** tab, as shown in Figure 5-32.

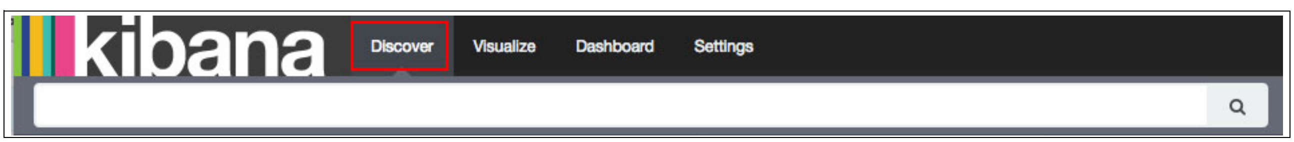


Figure 5-32 The Discover tab in Kibana

You can perform text searches. The search is performed by Elasticsearch, which uses features and indexes that are created by Logstash. In Figure 5-33, a search is performed for messages containing `fabio-nova`, which is the name of one VM that was previously created in the cluster.

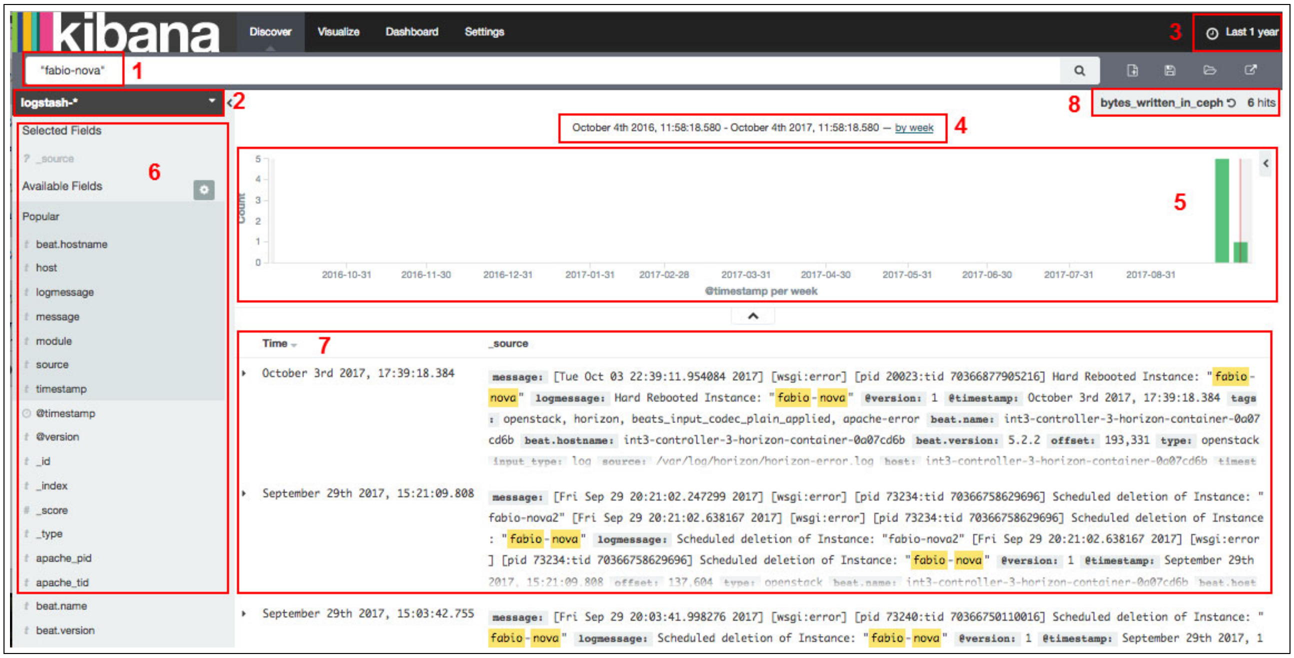


Figure 5-33   Performing a search by using Kibana

The numbers that are shown in Figure 5-33 are explained in the following list:

1. The `search` field to type the information for which you want to look.

> **Note:** You can use wildcards or quotation marks (“”) to look for specific or more precise information.

2. Here, you must select the source of the information, where the search is performed. If you click `Logstash-*`, you can switch to `Metricbeat-*`. As explained in 5.4, “Elastic stack (Kibana)” on page 149, Filebeat sends information to Logstash while Metricbeat sends it directly to Elasticsearch.
3. Here, you select the time frame in which to perform the search.

> **Important:** The Kibana search is restricted to messages that happened during that time frame.

4. The time frame that is considered for this search (based on the selection that is made in step 3.
5. This graphic shows the `count` of entries that resulted from this search (or the total number of results that resulted from this search), in a timeline, respecting the time frame that is selected in step 3.
6. The left pane shows the available fields in the log entries. Logstash receives the logs from multiple applications and standardizes them, creating fields to index and store the information that is received in such a log (for example, there is a `timestamp` field storing the data and time of the log entry, and a `message` field storing the message itself).

Note: You can use such fields to refine your search. By clicking in one of the fields, Kibana shows the available `results` for information in that field, and you can select one of them to refine your query. For example, if you click in the `loglevel` field, you can select `ERROR`, and only `ERROR` messages are displayed in the result of the search (`INFO` or `WARNING` messages are not shown).

7. This area shows the results of your search.

You can click in the log entry and expand it, showing all the available fields and information from that log, as shown in Figure 5-34.

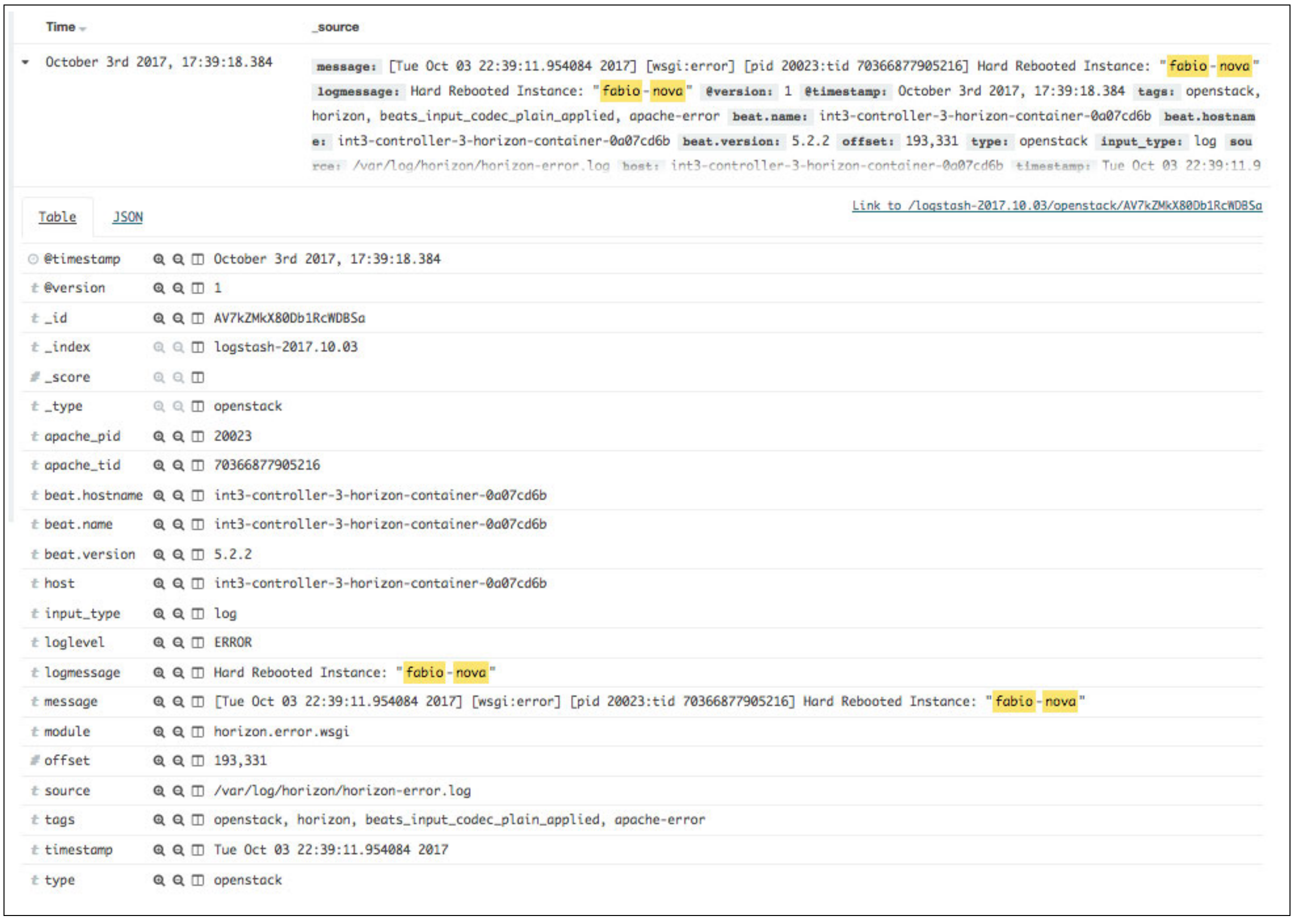


Figure 5-34 Details of the log entry

In Figure 5-34, you can see all the information for that specific log entry that resulted from your search. You can see that the field `message` contains the message itself, `@timestamp` shows the exact time and date when that message happened, `host` shows the server where this log came from, `loglevel` shows whether this message is an `ERROR`, `WARNING`, or `INFO`, and `source` shows the file name where this entry was logged. All of this information was passed from Filebeat to Logstash, transformed, and then stored in Elasticsearch. Kibana used Elasticsearch to obtain such information.

This section shows an example about how to search for logs that contain information about the number of bytes that are written to Ceph devices. This `search` is later used to create a graphic in 5.4.5, “Using Kibana to create a graph that is based on a search” on page 164.

One of the fields that is added by Logstash is the `tags` field. It contains tags that are related to that log entry, enabling you to use such tags in your queries. For example, the Ceph-related logs have `ceph` as a tag. There is also another field that is named `bytes_written`, which stores the total bytes that are written per second (this is part of the information in the Ceph log), as shown in Figure 5-35.

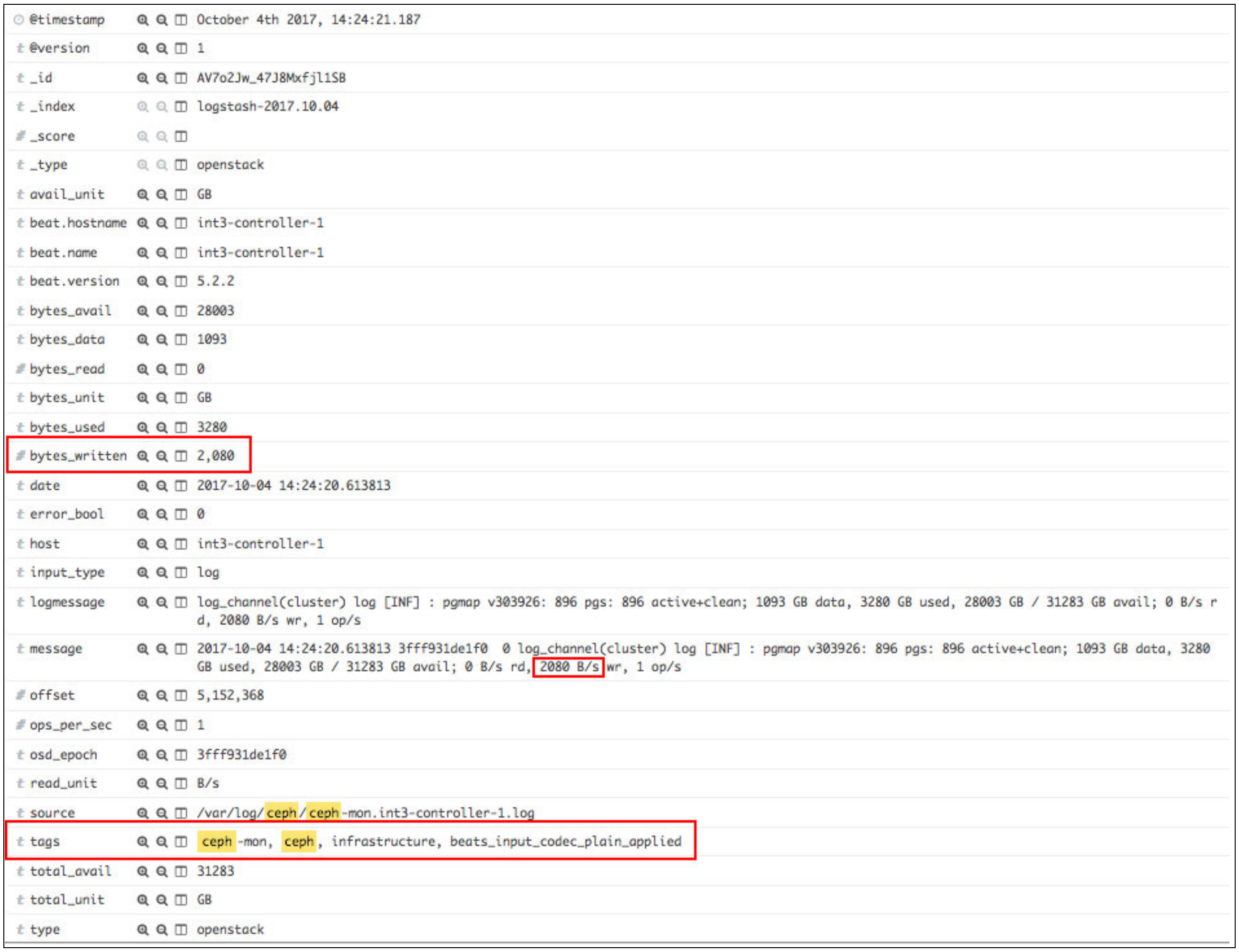


Figure 5-35 Ceph log in the Kibana interface showing the tags and number of bytes_written

Knowing such information, you can search for messages with the tag `ceph` and request only messages that have the `bytes_written` field by performing a search, as shown in Example 5-18.

Example 5-18 Searching for Ceph logs with the bytes_written field

```
tags:ceph AND _exists_:bytes_written
```

Figure 5-36 shows the search results for this query. It considers only logs from the past 12 hours, according to the selection that is made at the upper right corner of the window.

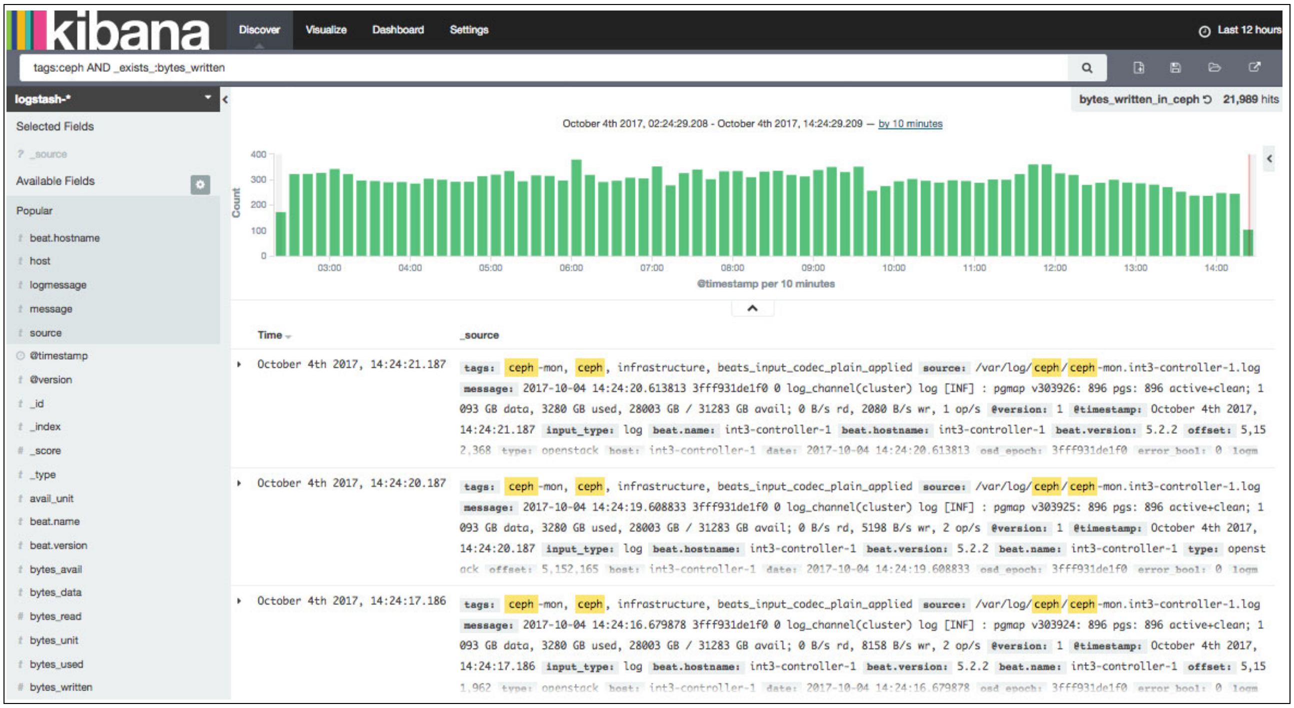


Figure 5-36 Search results from the Ceph logs with bytes_written

Click the **Save Search** button at the upper right corner of your window to save this query. Later, you can use this query to create a graphic, as explained in 5.4.5, “Using Kibana to create a graph that is based on a search” on page 164. Figure 5-37 shows the Save Search option.

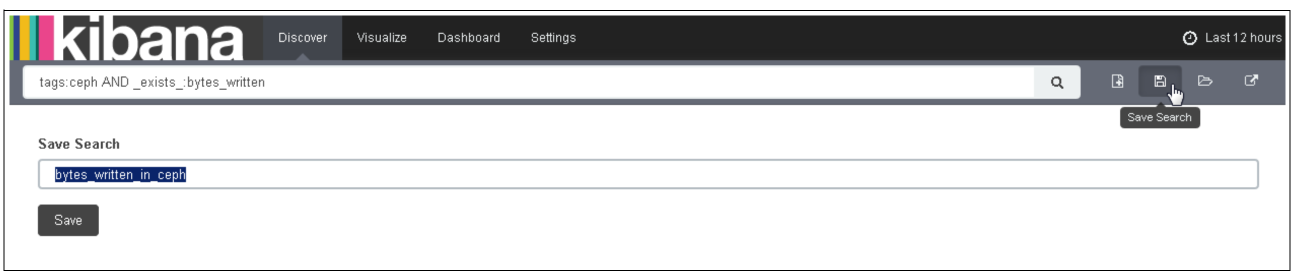


Figure 5-37 Save Search option

5.4.4 Viewing saved searches

After you save a search, you can open it at by clicking the **Discover** tab, clicking the **Open** button, and then selecting the correct search among the available ones, as shown in Figure 5-38.

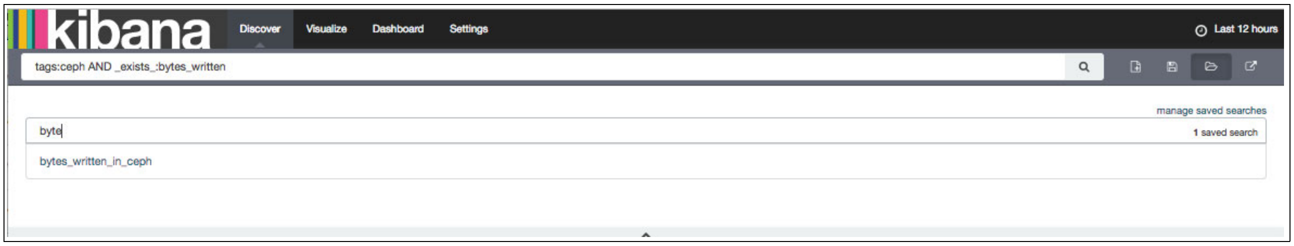


Figure 5-38 Opening saved searches

The search is automatically performed by Kibana and Elasticsearch based on the time frame that is selected at the upper right corner of your window.

5.4.5 Using Kibana to create a graph that is based on a search

You can use Kibana to create a visualization that is based on information that is obtained by a search. In this example, you can create a graph (line chart) to show the number of bytes that was written by Ceph in the past few hours by using the information that is provided by the search that was prepared and saved in 5.4.3, “Performing searches with Kibana” on page 159.

Complete the following steps:

1. Click **Visualize** and select the visualization method (**Line Chart**), as shown in Figure 5-39.

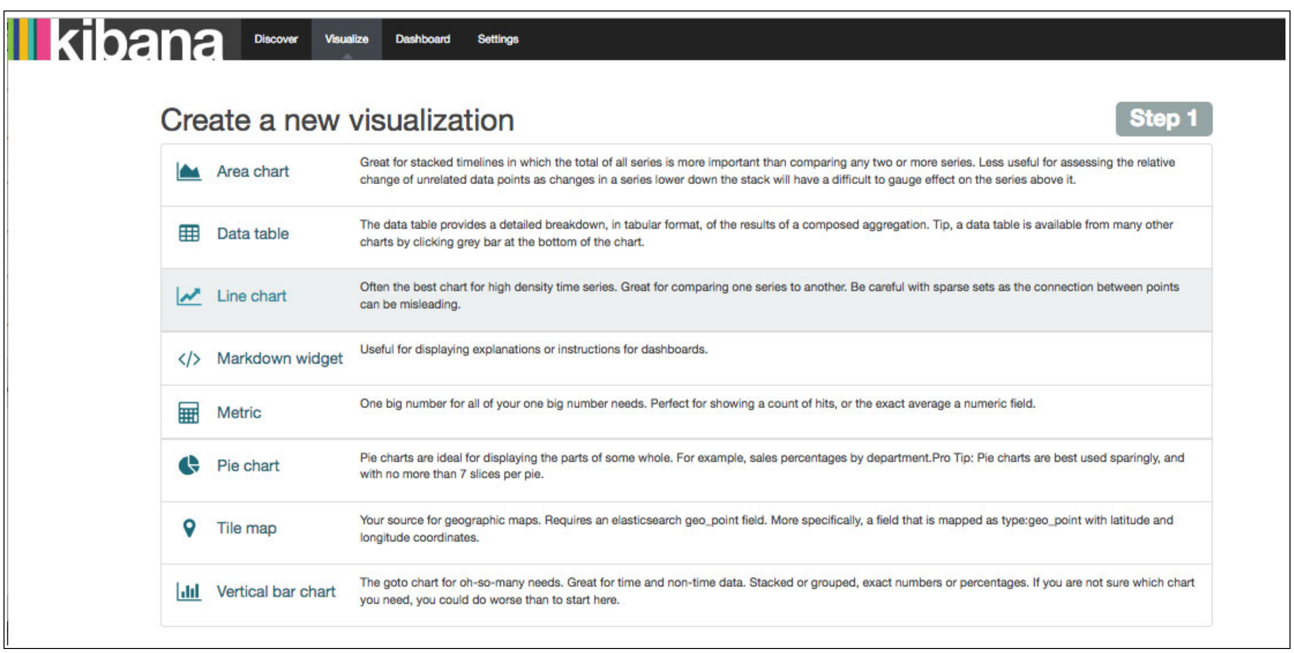


Figure 5-39 Creating a line chart visualization

2. You must select the source of information for this line chart. This case uses the previously saved search, so select it, as shown in Figure 5-40.

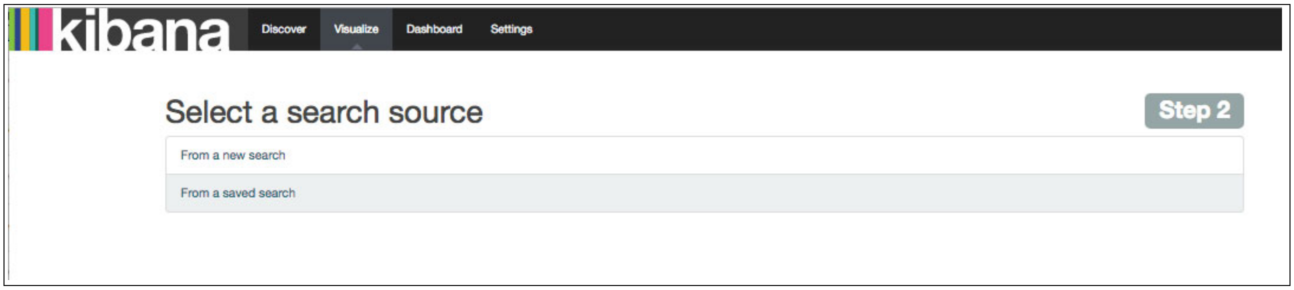


Figure 5-40 Using a saved search as the source of information

3. Then, select the saved search that you want to use as the source of information for this visualization. Figure 5-41 uses the `bytes_written_in_ceph` saved search.

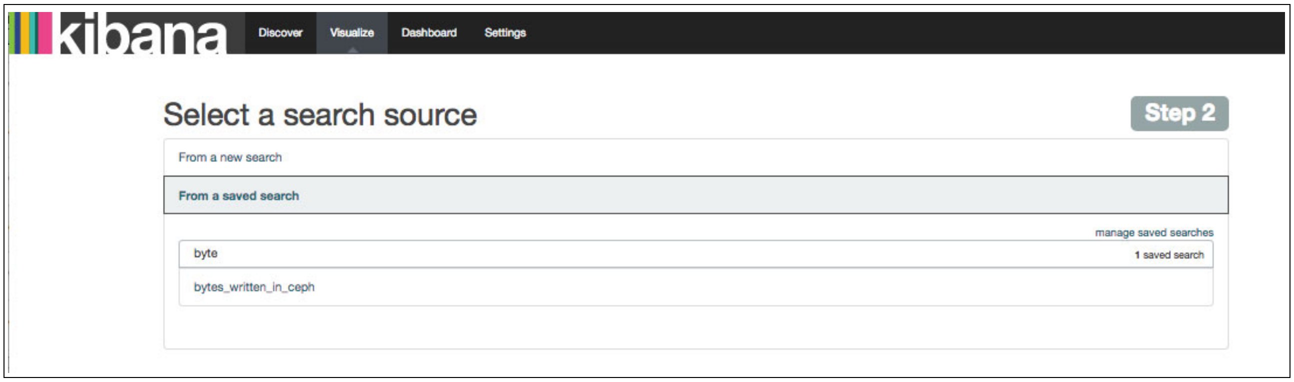


Figure 5-41 Selecting the saved search

4. Now, customize your visualization. Given that it is a Line chart, it has two axes (Y is the vertical axis and X is the horizontal axis). In this example, the Y-Axis shows the Average (Aggregation) of data in the Field bytes.written (obtained from the search that is performed in Kibana). The Y-Axis aggregates the data based on a Date Histogram that uses the `@timestamp` field of the logs to determine how the data is aggregated. After making these selections, click Play. The graph is shown (Figure 5-42).

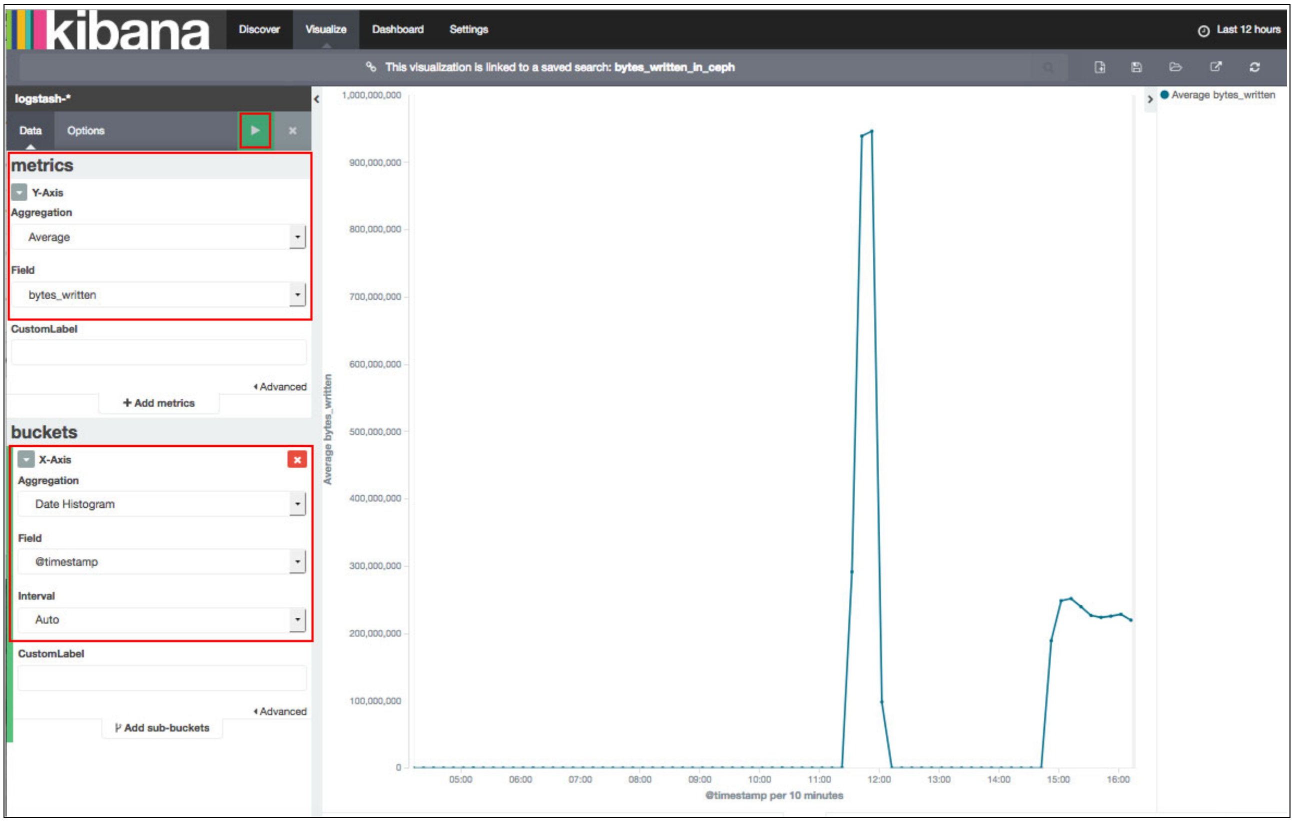


Figure 5-42 Customizing the visualization

The data is based on the time frame that is selected at the upper right corner of the window. You can modify this time frame so that the graph is also dynamically changed, as shown in Figure 5-43 on page 167.

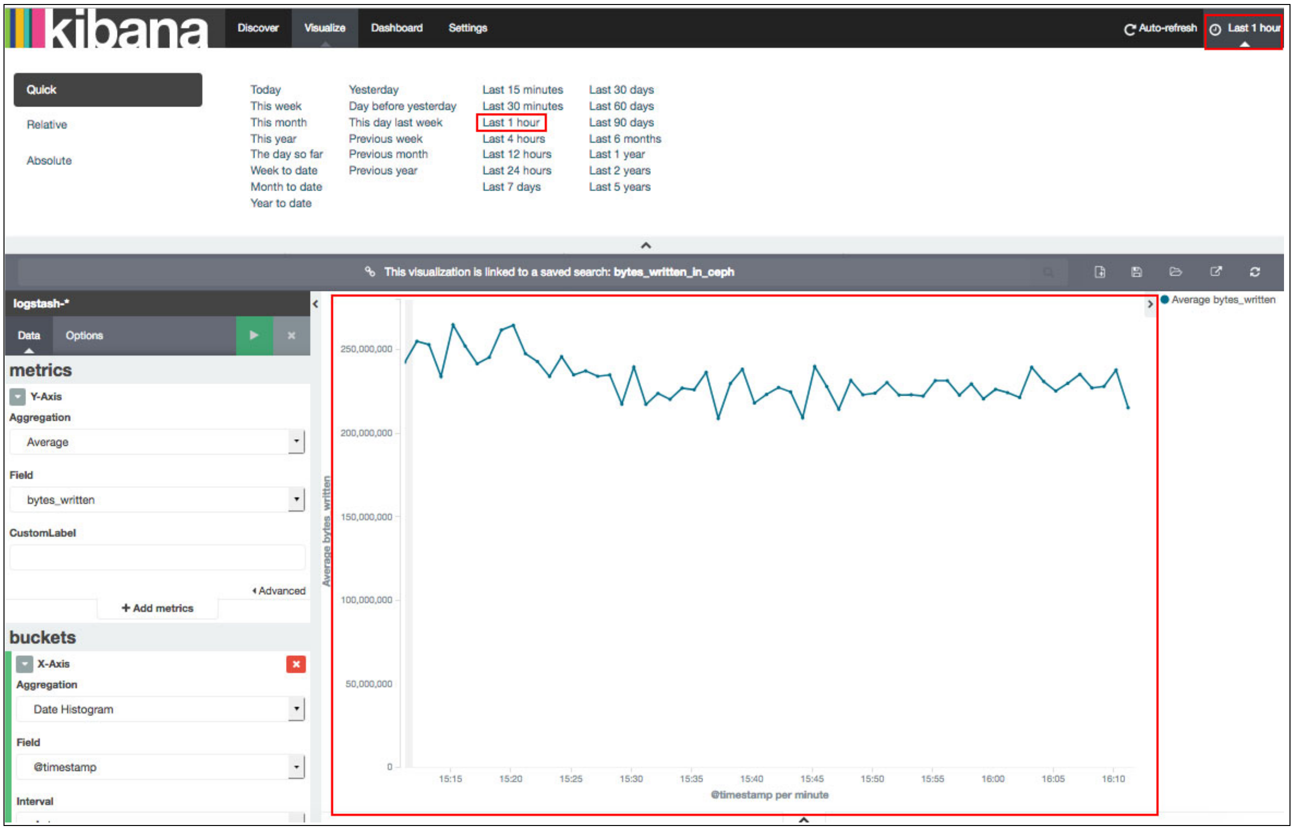


Figure 5-43 Changing the time frame

5. After you prepare your visualization, click **Save** to save this visualization, as shown in Figure 5-44.

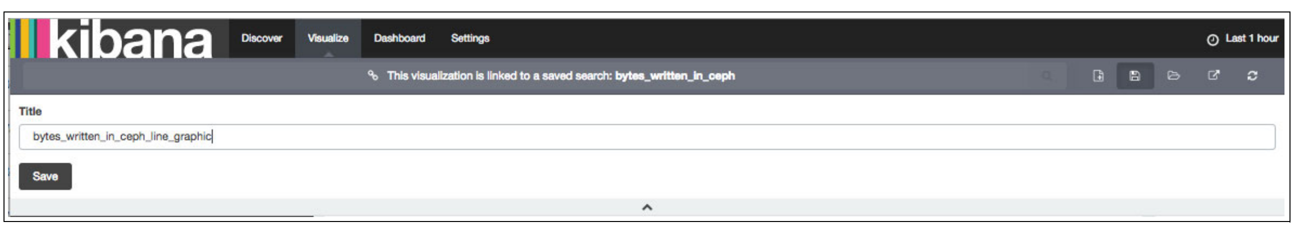


Figure 5-44 Saving the visualization

5.4.6 Viewing saved visualizations

When you have a saved visualization, you can reload it by opening the saved visualization. To complete this action, click the Visualize tab and click **Open**. Look for your visualization and click it, as shown in Figure 5-45.

Figure 5-45 Opening a saved visualization

5.4.7 Adding the graph visualization to the Dashboard

You can add a saved visualization to your Dashboard so that you can use this information to monitor your Open Platform for DBaaS on Power Systems cluster. To complete this action, open the Dashboard that you want to use, click +, which is Add Visualization button, and then search for the visualization that you want to add and click it, as shown in Figure 5-46.

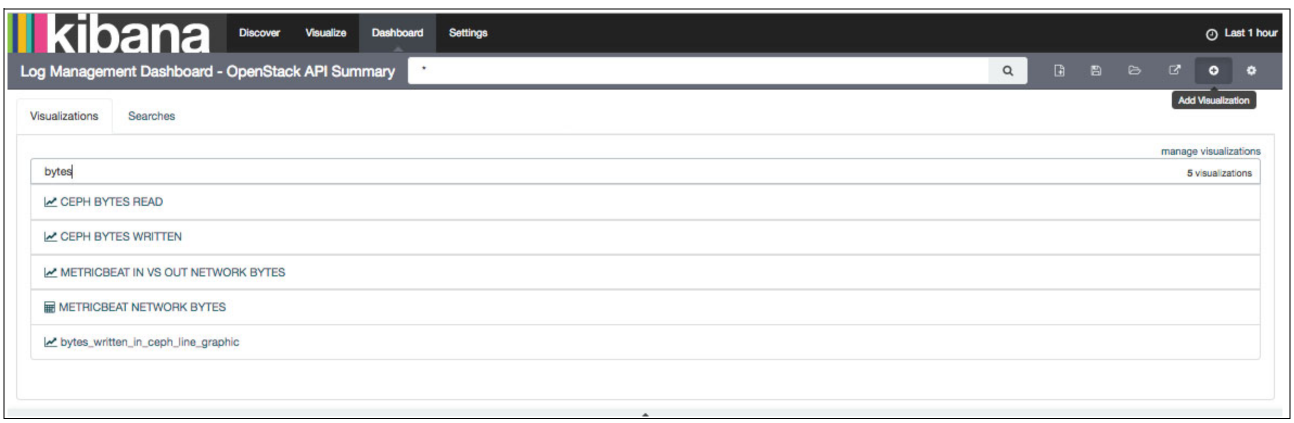


Figure 5-46 Adding a visualization to the Dashboard

The Visualization is added to the bottom of the Dashboard. You can move it to the position where you want this information to appear, and also resize it, as shown in Figure 5-47.

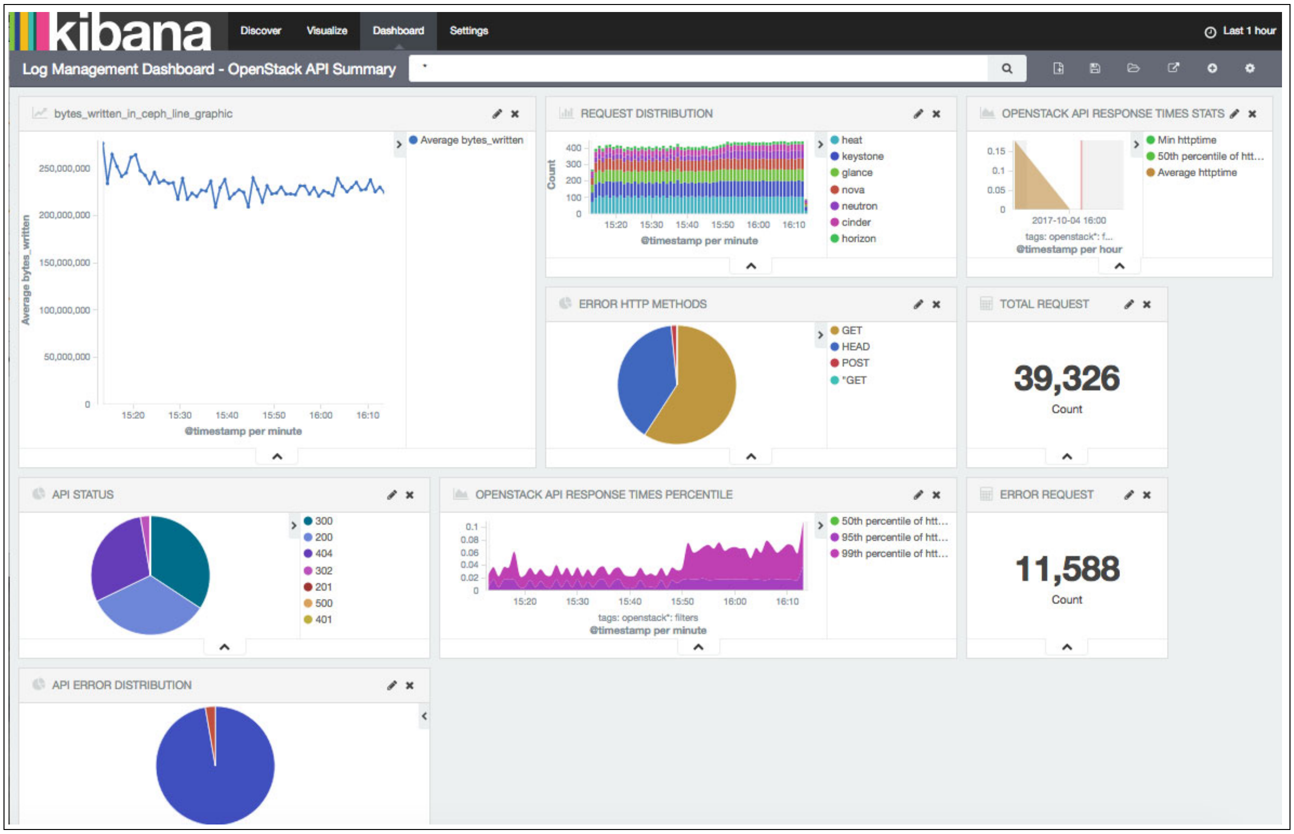


Figure 5-47 Positioning and resizing the visualization in the Dashboard

After completing this action, click **Save** at the upper right corner of the window to save the changes that are made to this Dashboard so that this visualization is added persistently to it, as shown in Figure 5-48.

Figure 5-48 Saving the changes to the Dashboard

5.4.8 Using Kibana to view metrics

Kibana also receives metrics information that is provided by Metricbeat, such as the workload of the systems CPU, memory usage, and file system utilization. You can use such information to generate graphics or tables to monitor your Open Platform for DBaaS on Power Systems cluster.

Note: Metricbeat can also run in the VMs and provide information such as connectivity to a database engine. At the time of writing, only the infrastructure is monitored by Metricbeat, which provides information only about the controller, compute, and storage (Ceph and Swift) nodes, but not the VM information, so no metrics about the database engines are available at this time.

This section shows an example of using information that is provided by Metricbeat to create a table showing the CPU workload in your nodes. Complete the following steps:

1. In the Discover tab, select `metricbeat-*` as the source of information and then perform a search for entries with the `metricset.module` as `system` and `metricset.name` as `cpu` or `load`, as shown in Example 5-19.

   Example 5-19 Searching for metrics from Metricbeat

   ```
   metricset.module: system AND (metricset.name: cpu OR  metricset.name: load)
   ```

   Figure 5-49 shows the results of this query.

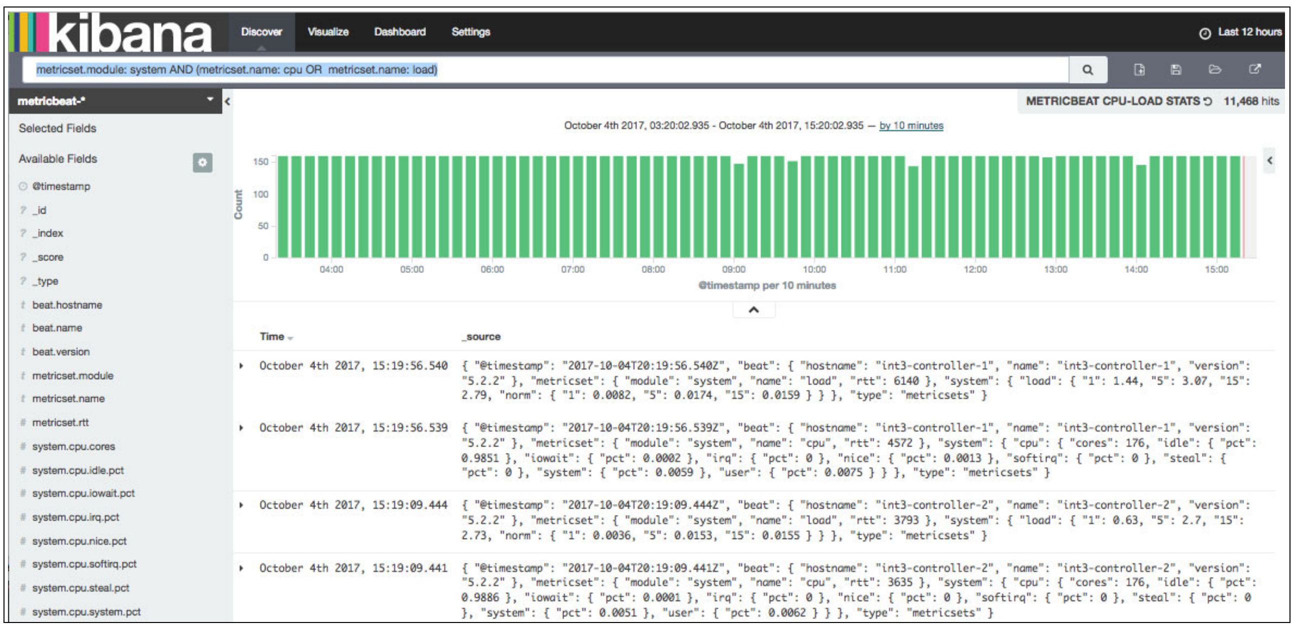


Figure 5-49 Searching for information from Metricbeat

Viewing the details of the log entries, you can see the type of information that is provided by Metricbeat to Elasticseach. Figure 5-50 on page 171 shows node `int3-controller-1` (one of the controller nodes) has 98.51% of idle CPU.

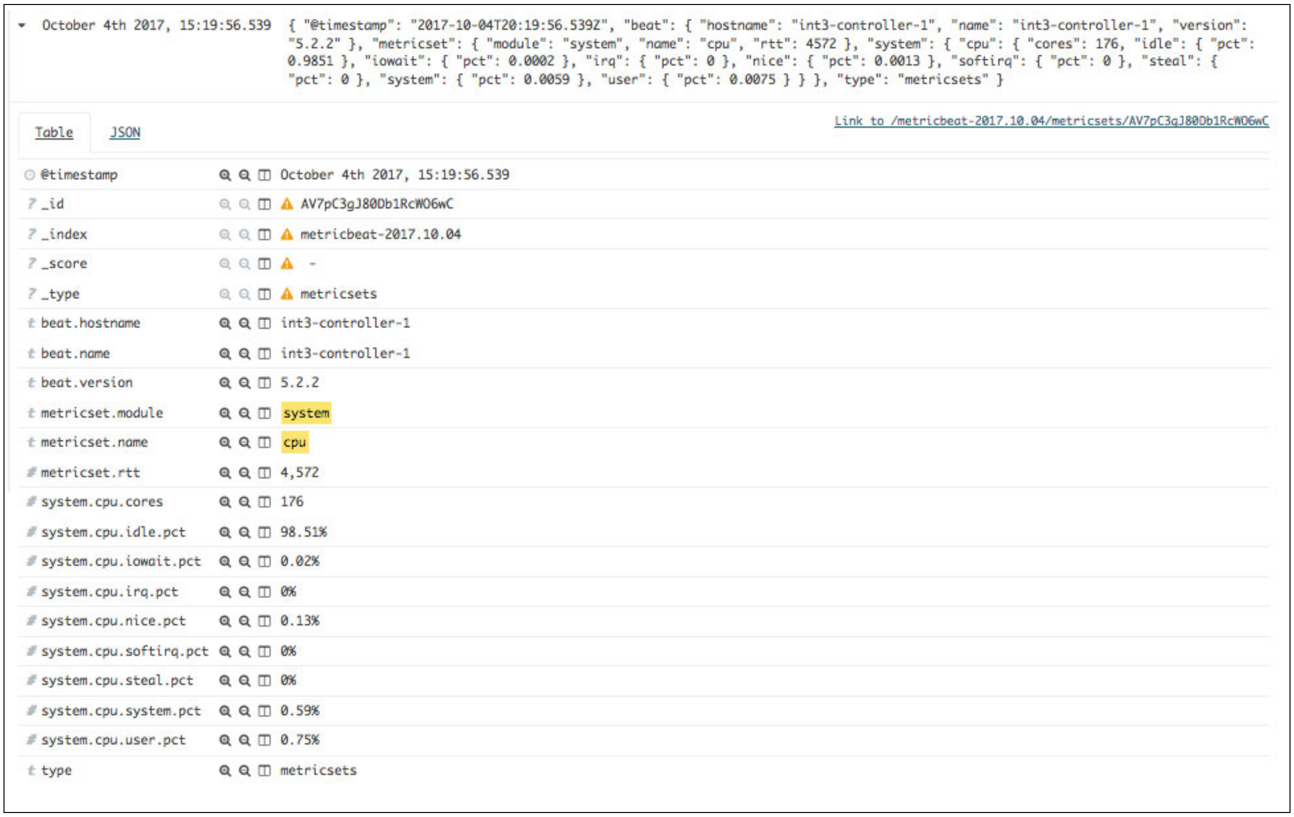


Figure 5-50 CPU usage information by Metricbeat

2. You can save this search by using the **Save Search** button on the upper right corner of your window, as shown in Figure 5-51.

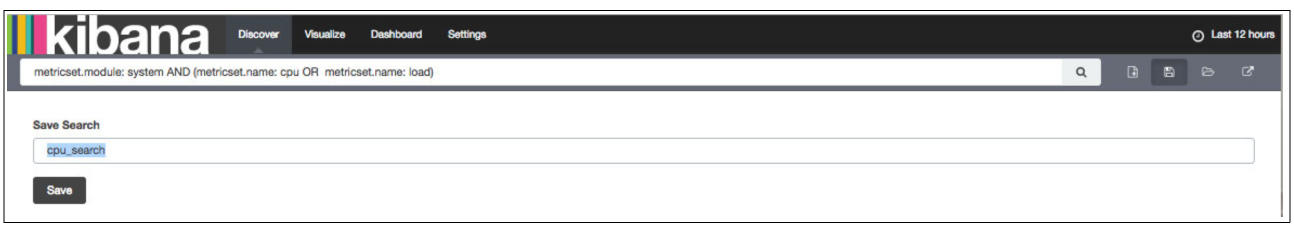


Figure 5-51 Saving the Metricbeat search

3. You can then create a visualization that is based on the information that is provided by this search. Click the **Visualize** tab and then, in the **Create new visualization** area, select the type of visualization that you want to create. In this example, a **Data table** visualization is selected, as shown in Figure 5-52.

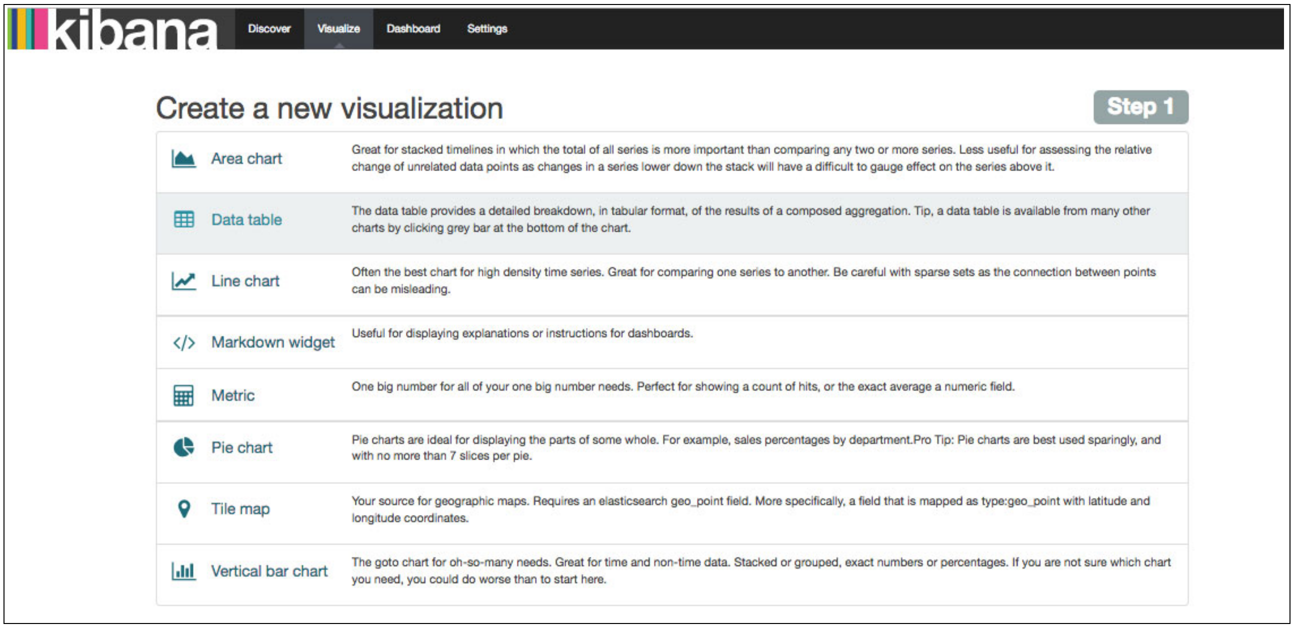


Figure 5-52 Creating a Data table visualization

You can create this visualization based on a new search (you must perform this search in the search area of the Kibana) or by using a previously saved search. Figure 5-53 shows a selection from a saved search to use a search that was performed earlier on this section.

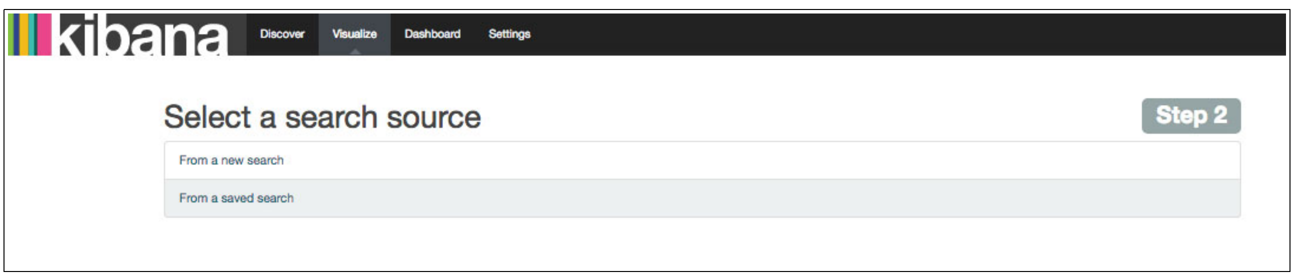


Figure 5-53 Using a saved search to create a visualization

4. After selecting **From a saved search**, select the saved search to use as the source for this visualization, as shown in Figure 5-54 on page 173.

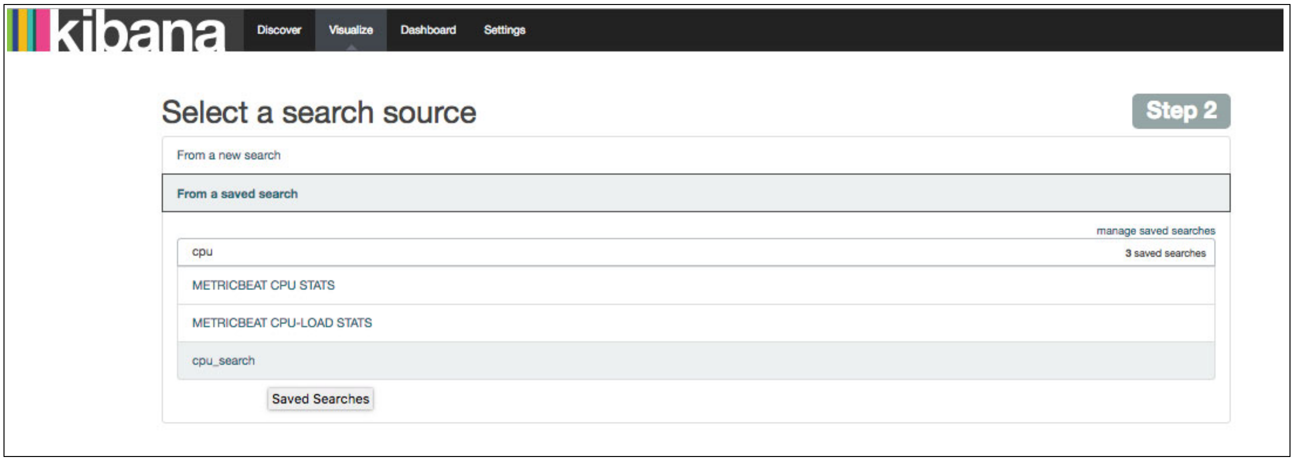


Figure 5-54 Selecting the saved search

5. In this section, customize your visualization, as shown in Figure 5-55, based on a saved search. In the left pane, you have the metrics and buckets fields. The metrics area determines the columns that this table will have. Add as many columns as you want by clicking **+Add metric** and include a new metric. This example uses the Average as the Aggregation method for the data, and then selects the field to aggregate in the table, for example, `system.cpu.idle.pct`. This column shows the average of idle CPU percentage according to the information that is provided by Metricbeat in the last 12 hours (per the selection at the upper right corner of the window).

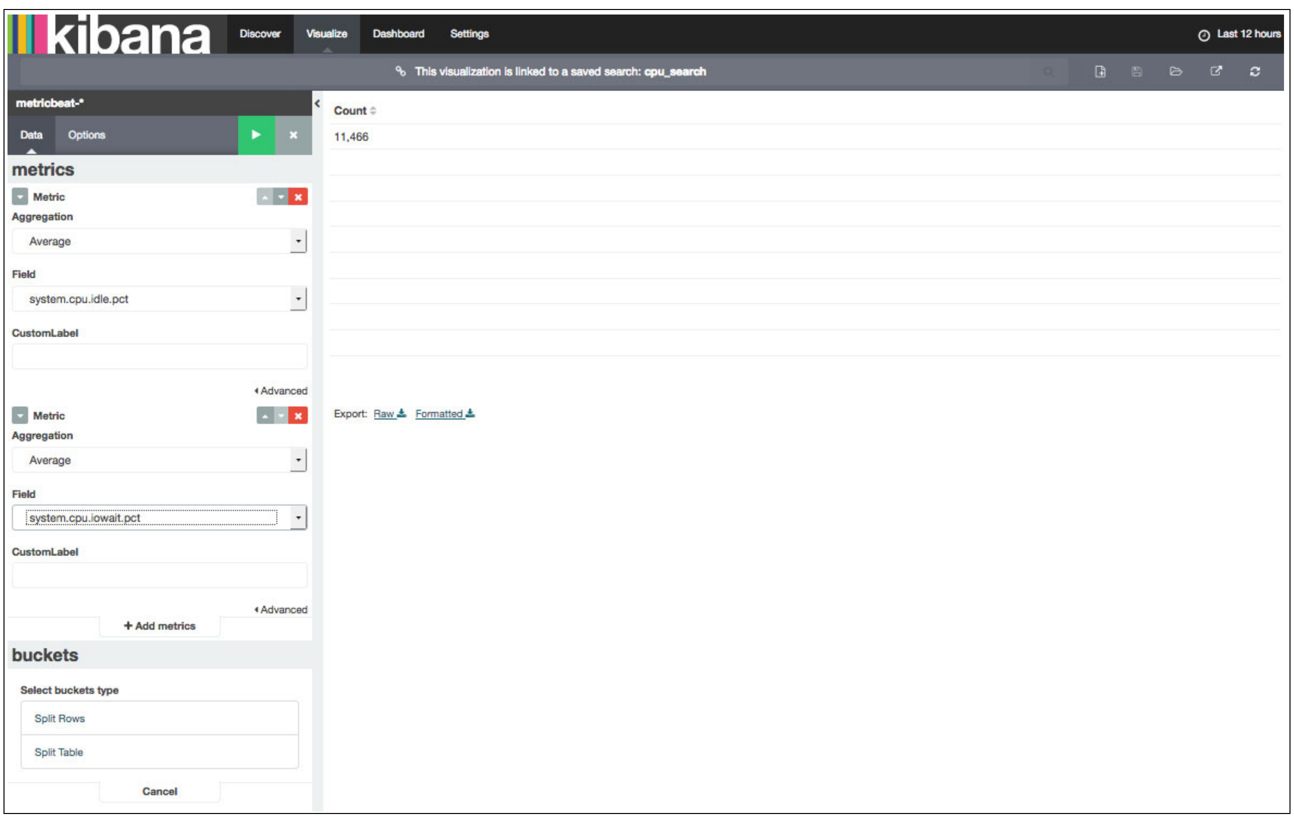


Figure 5-55 Adding metrics to the visualization Data table

While adding the metrics, you can customize the label of each metric, which is the information that is used in the header of the column in the Table, as shown in Figure 5-56.

Note: If you do not customize the label, the column of the table has the field name as the header (for example, `system.cpu.iowait.pct`).

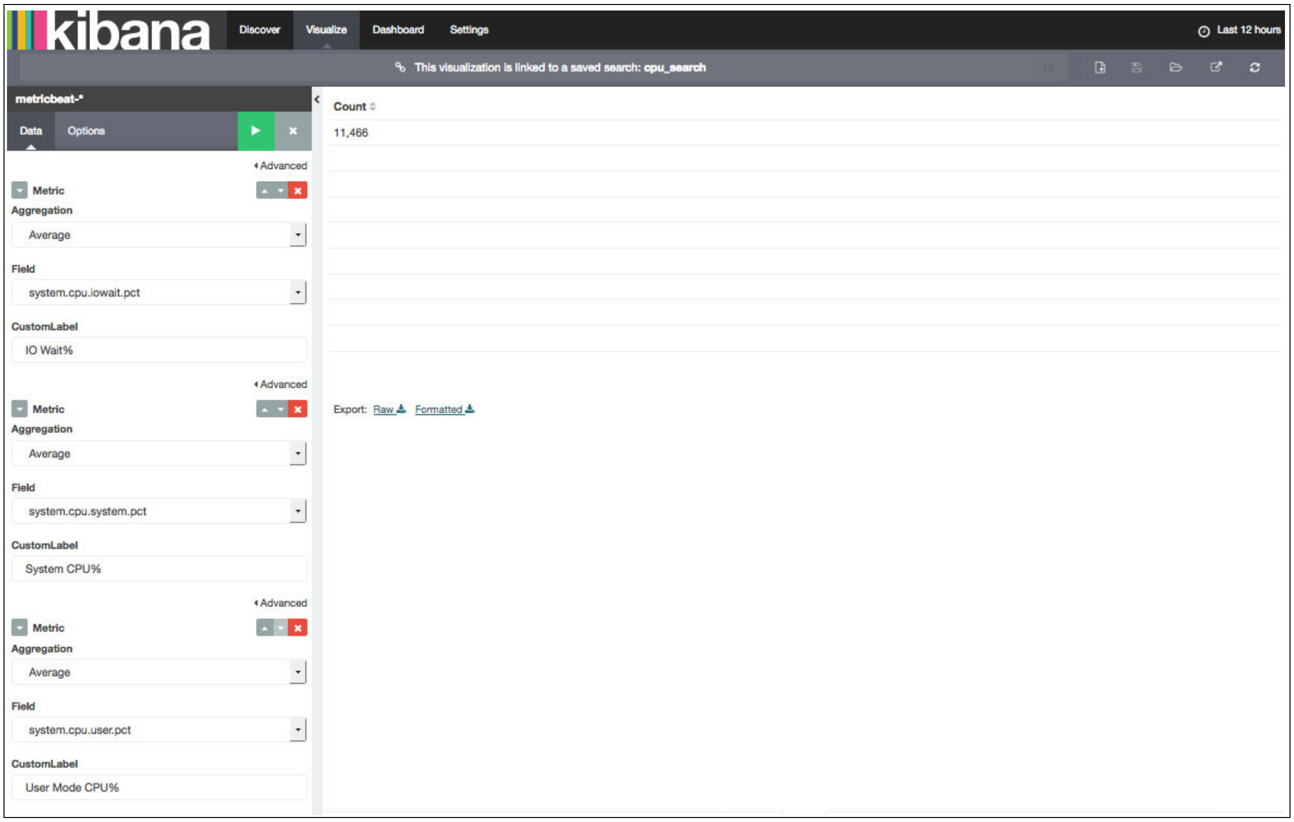


Figure 5-56 Customizing the label

6. After adding all the metrics, choose in the buckets area how this table will be divided by determining how the rows of the table will be split. Here, click **Split Rows** and then select how you want to split the rows of the table. Figure 5-57 on page 175 shows that the rows are divided by Terms and the term is used to determine how the data is aggregated. The rows are divided by the `beat.name`, which is name of the Metricbeat agent sending information (which also corresponds to the host where the beat is running). In this example, a custom label is added.

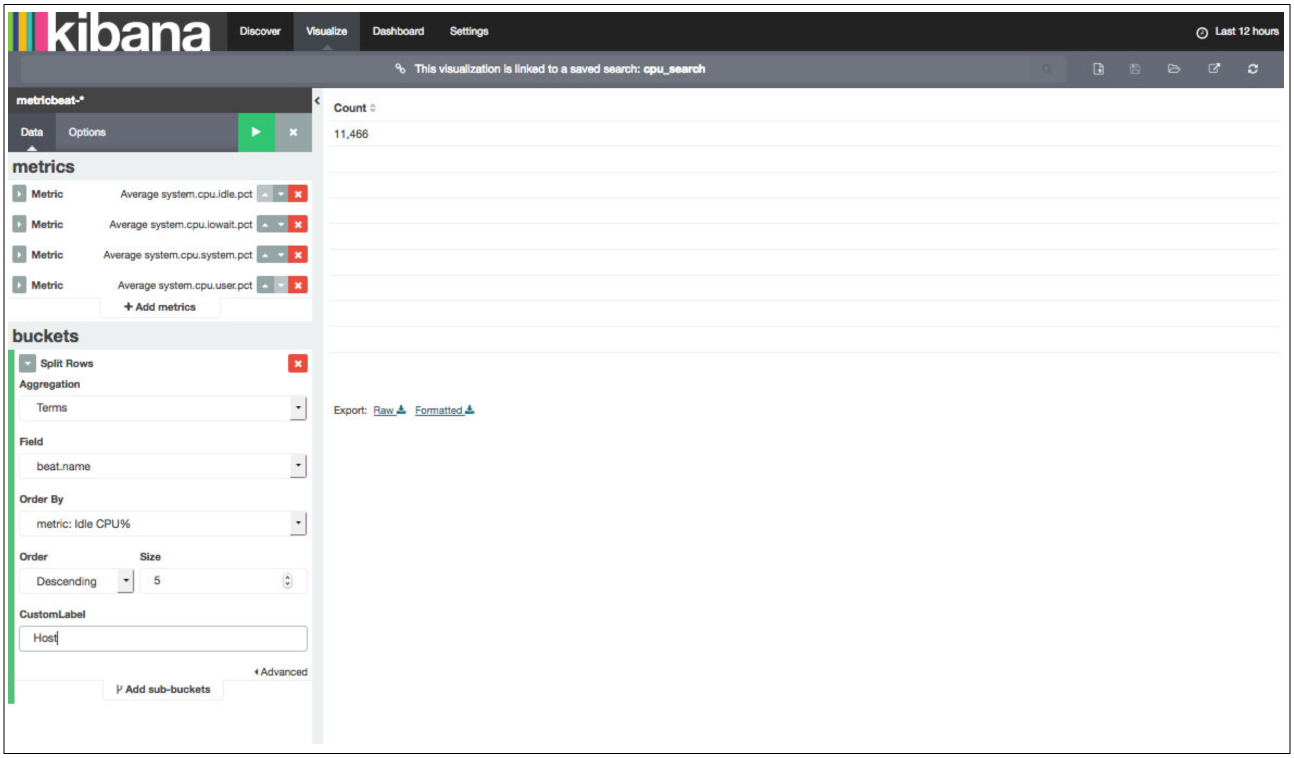


Figure 5-57 Splitting the rows of the table

7. After preparing all the details of this visualization, click the **Play** button at the top of the left menu and the visualization is then run by Kibana, creating the table based on the information from the previous search, and respecting the time frame at the upper right corner of your window, as shown in Figure 5-58.

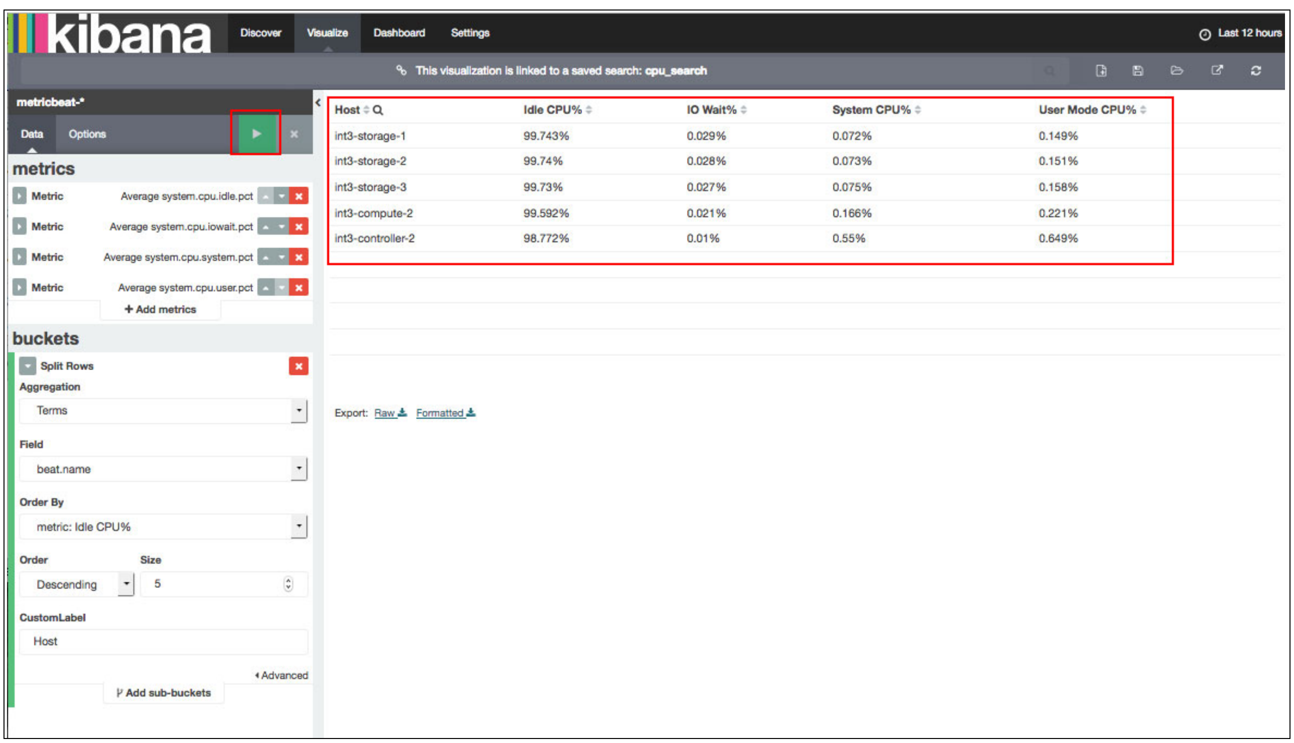


Figure 5-58 Playing the visualization to create the table

You can save your new visualization, as shown in Figure 5-59.

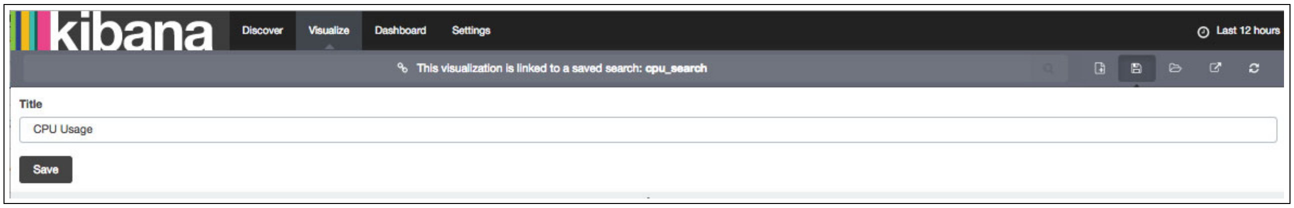


Figure 5-59 Saving the visualization

8. After saving the visualization, you can add it to your Dashboard so that you can track this information and monitor the CPU utilization of your cluster. Go to your Dashboard and click **Add Visualization**, and then select the visualization, as shown in Figure 5-60.

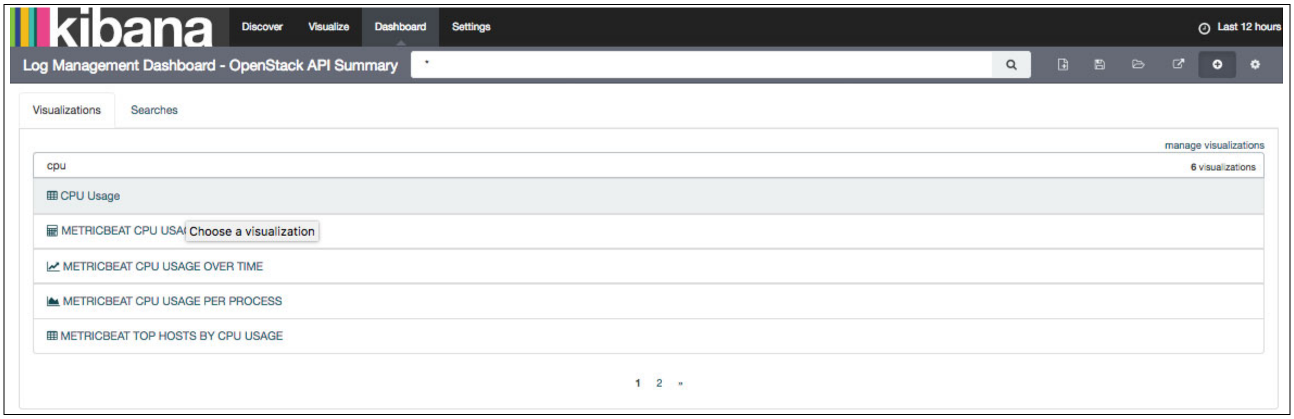


Figure 5-60 Adding a visualization to the Dashboard

9. Then, you can resize and place the visualization on your Dashboard, as shown in Figure 5-61 on page 177. The information that is displayed in this Dashboard is dynamic and shows the visualization data according to the time frame that is selected at the upper right corner of your window.

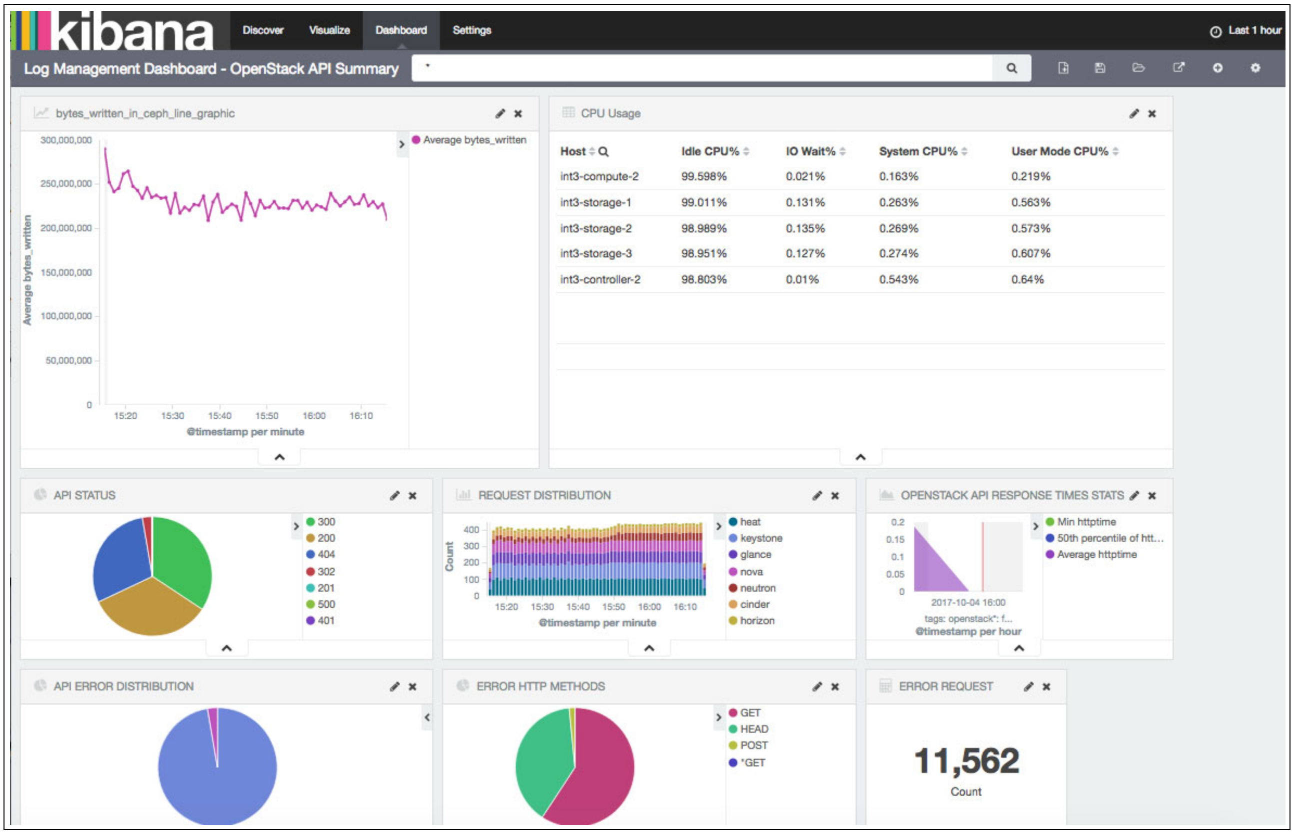


Figure 5-61 Visualization added to the Dashboard

10. Now that you have modified your Dashboard, save it so that this modification persists every time that you open this Dashboard, as shown in Figure 5-62.

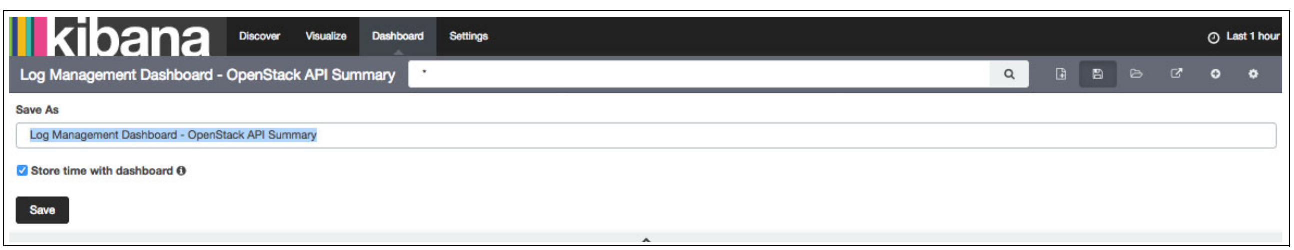


Figure 5-62 Saving the modifications in the Dashboard

If you prefer, you can create a Dashboard and add all the relevant visualizations that you like to have in the same window to monitor your environment.

> **Important:** The New, Open, and Save menus vary according to the tab where you are working. To open a saved visualization, you must be in the Visualizations tab, for example.

5.4.9 Using Kibana for troubleshooting

The Kibana interface is also useful for searching for errors and understanding the root cause of an issue. For example, Figure 5-63 shows an error that occurred while attempting to start a VM instance.

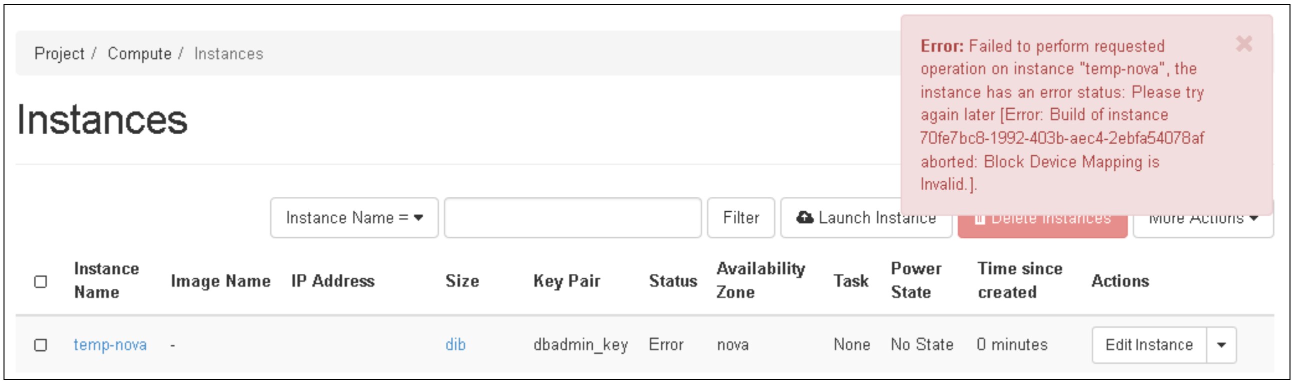


Figure 5-63 Error while starting a virtual machine instance

The error message that is displayed is shown in Example 5-20.

Example 5-20 Error message while starting a virtual machine

```
Error: Failed to perform requested operation on instance "temp-nova", the instance
has an error status: Please try again later [Error: Build of instance
70fe7bc8-1992-403b-aec4-2ebfa54078af aborted: Block Device Mapping is Invalid.].
```

As you can see from the message that is shown in Example 5-20, the VM ID that Kibana was trying to start is `70fe7bc8-1992-403b-aec4-2ebfa54078af`. You can view all messages that are related to this VM by using the Kibana interface (Filebeat constantly sends new information to Logstash, and the data is available immediately after the error). Click **Discover**, paste the Instance ID in the search area, and click **Search**. Kibana queries Elasticsearch and filters all messages that are related to this Instance ID, as shown in Figure 5-64.

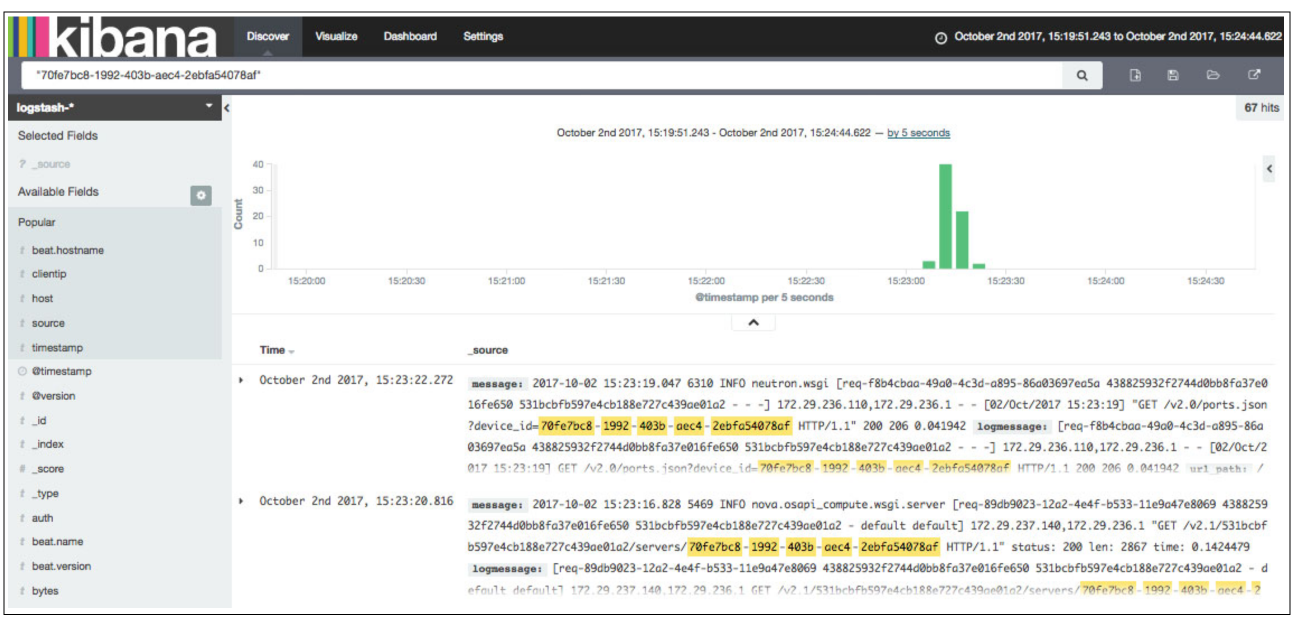


Figure 5-64 Searching for errors by using Kibana

Important: Remember to use double quotation marks “” during the search if you know exactly the message for which you are looking. Example 5-20 on page 178 shows that the Instance ID used “`70fe7bc8-1992-403b-aec4-2ebfa54078af`” for a precise query.

You can see that Kibana returned 81 hits. In this case, you are interested in error messages, so you can change the filter of the `loglevel` to `ERROR`. In the menu on the left, click **loglevel** and then click + for the ERROR messages, as shown in Figure 5-65.

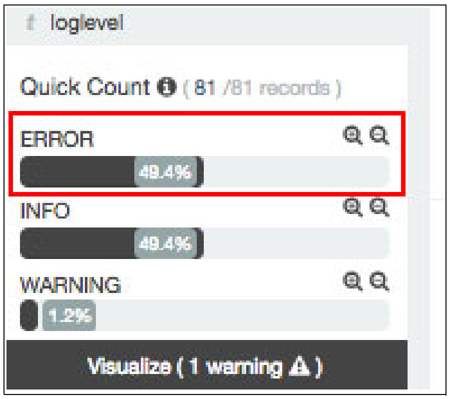


Figure 5-65 Changing the filter for loglevel to ERROR

You can see that the Kibana places a filter for ‘`loglevel:` “`ERROR`”’ under the search bar. The number of hits is reduced to 40, and only the error messages are displayed, as shown in Figure 5-66.

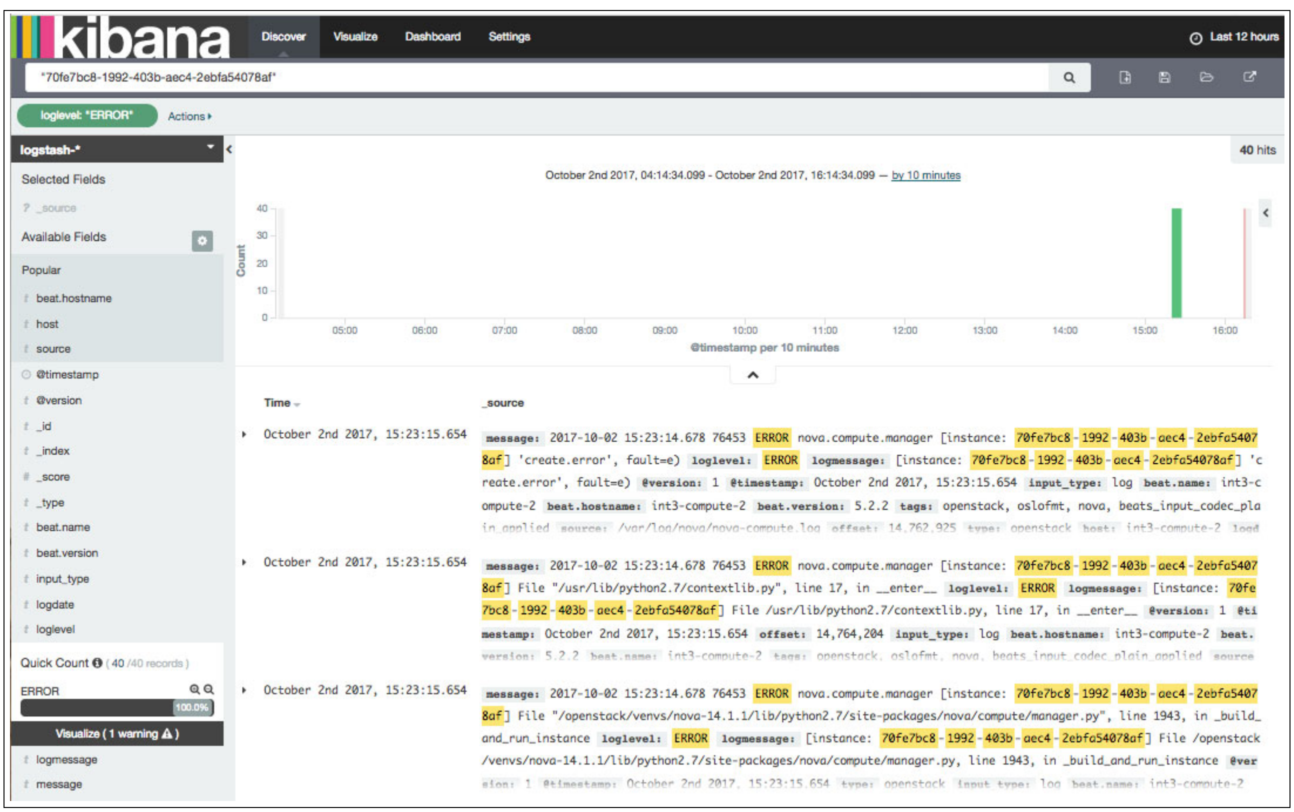


Figure 5-66 Log messages that are filtered to display ERROR messages

Note: At the top of the window, under the search bar, you can see the filters that are applied in this search. If you want to remove filters, move your cursor over the filter and a `Trash can` icon appears, which you can use to remove the filter.

You can expand each of the entries to display its details. Given that all data is standardized by Logstash, the core part of the message is displayed in the `message` field, as shown in Figure 5-67.

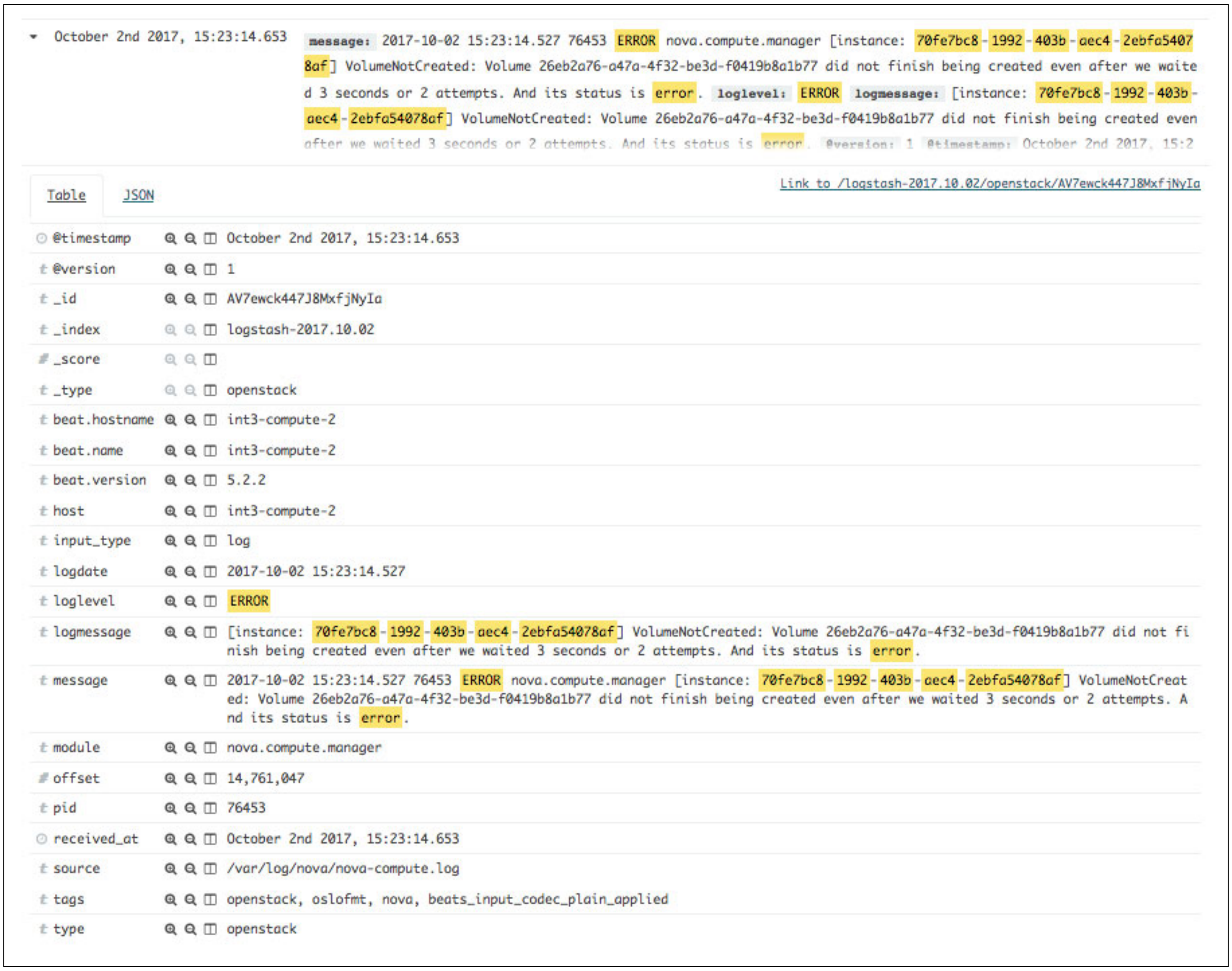


Figure 5-67 Displaying the message field in Kibana

From the message that is shown in Figure 5-67 on page 180, you can see that the instance failed to spawn due to a failure in creating the volume that this instance must use. The message shows the volume ID, which can also be used to search in Kibana for failure messages, as shown in Figure 5-68.

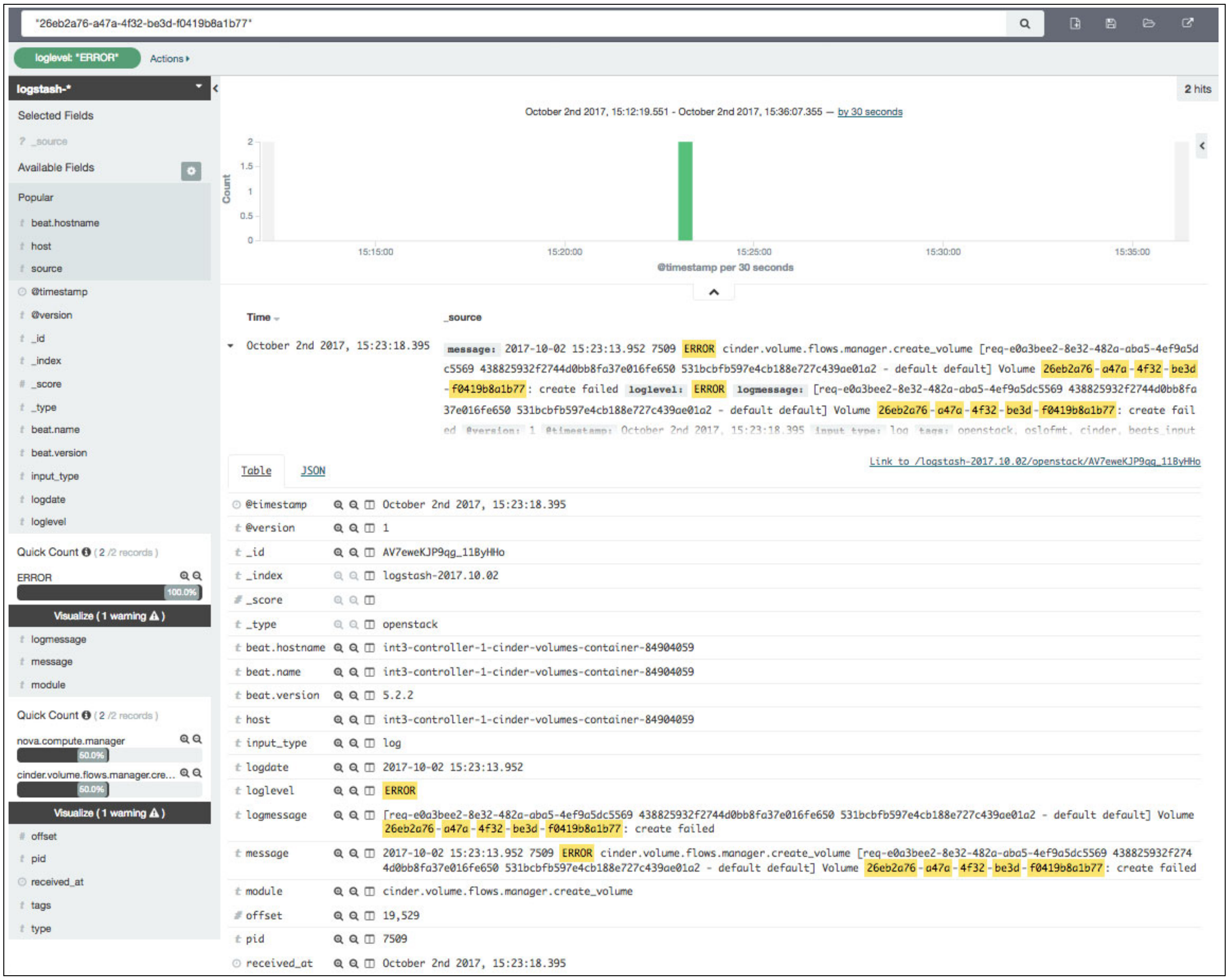


Figure 5-68 Searching by the volume ID

There were only two hits of errors with this volume ID. One of the hits is the message that is shown in Figure 5-67 on page 180 and the other is displayed in Figure 5-68. You know that the instance failed to spawn due to failure in Cinder to create the volume for the instance. In Figure 5-67 on page 180, you can see that the failure came from a module that is named `cinder.volume.flows.manager.create_volume`, so you can filter messages coming from this module to check whether it gives more details of such a failure.

To check the module, clear the search box and remove the filter that was applied (`loglevel:` **“**`ERROR`**”**). Then, using the left pane, look for the module field. You can see in Figure 5-69 that only five modules are shown. When you click the field name, only a quick count or the top five items are displayed.

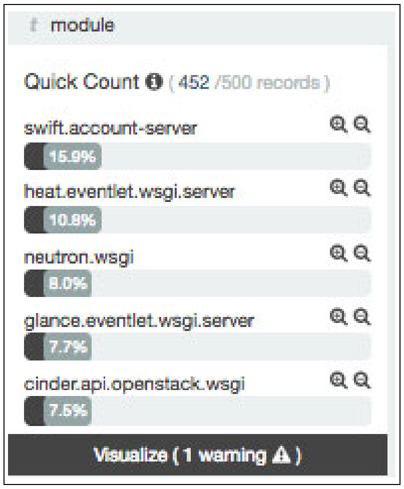


Figure 5-69 Filtering messages by module

The module to be filtered is not available here, but you know its name from the previous log message. You can use the search field with a special syntax to filter the messages based on a field, or if you do not know the syntax, you can select a different module and then use the same syntax to apply the correct filter. In this case, the syntax that you want to use is shown in Figure 5-70.

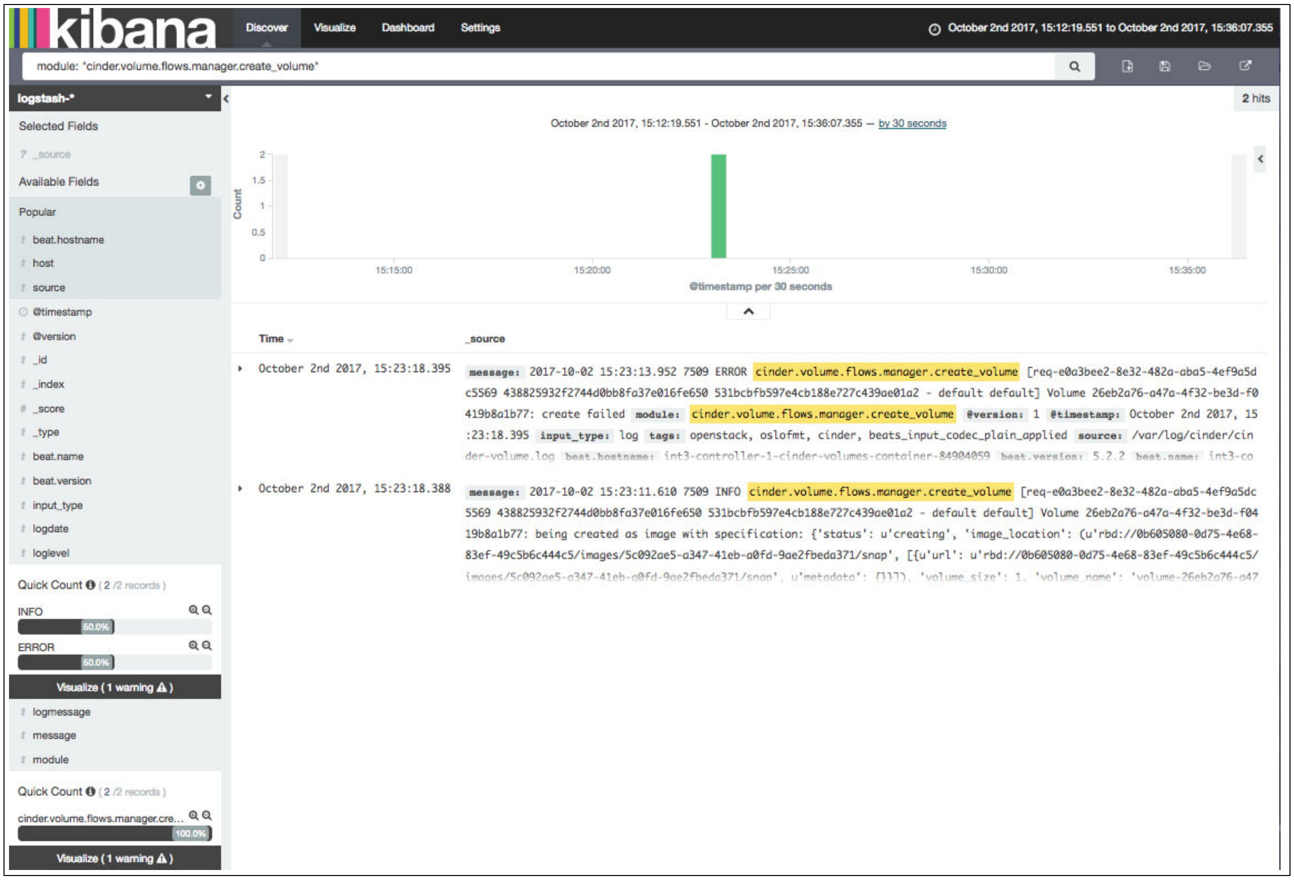


Figure 5-70 Filtering messages per module

Expanding the messages, you see that the volume that failed is a 1 GB volume, based on a Glance image with ID `5c092ae5-a347-41eb-a0fd-9ae2fbeda371`, as shown in Figure 5-71.

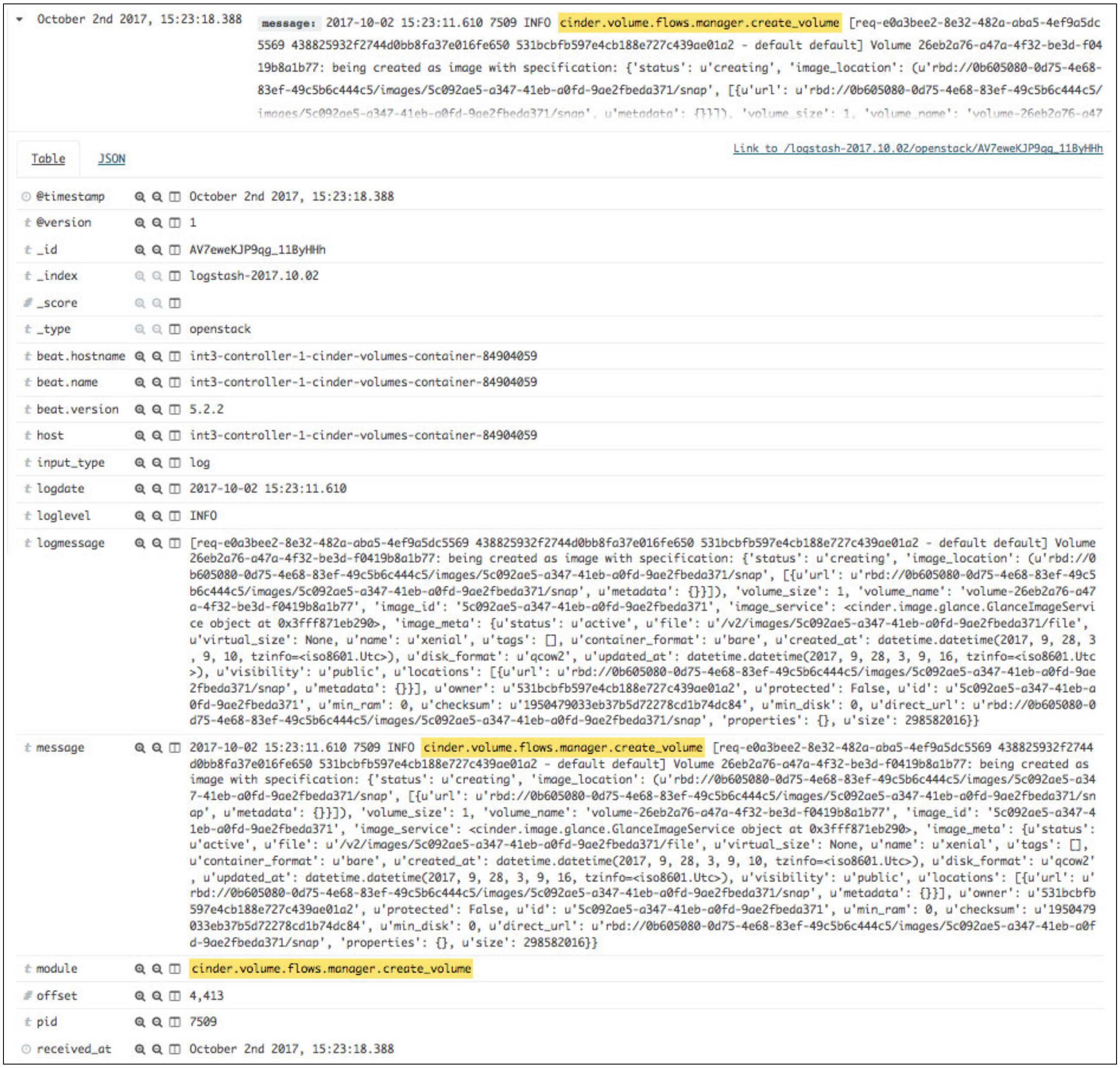


Figure 5-71 Obtaining the source of the volume (Glance image)

You can now use the search bar to look for failures that are related to this image ID, as shown in Figure 5-72.

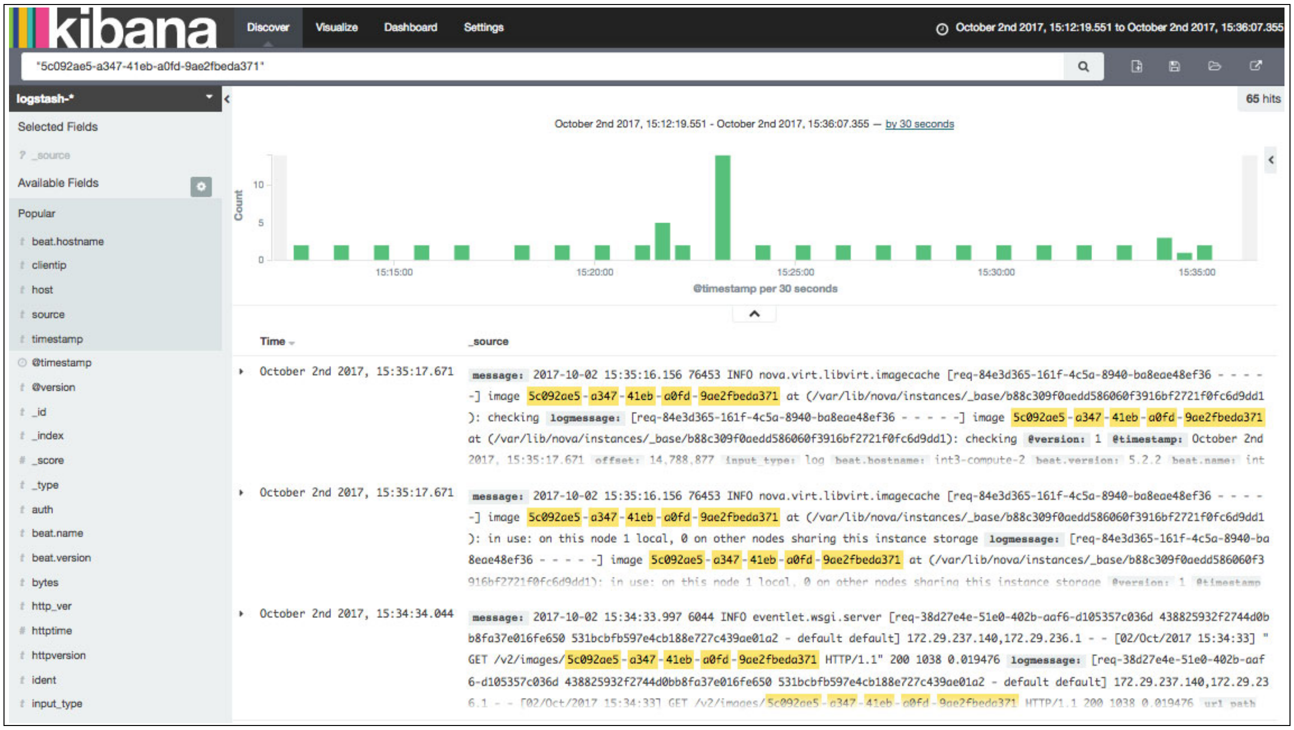


Figure 5-72 Messages that are filtered by image ID

As this image is constantly being used, there are messages that are not related to the instance that failed during deployment. Given that you know the approximate time stamp of the failure, you can hover your cursor over the graphic, click it, and drag over the time that you want to filter, as shown in Figure 5-73.

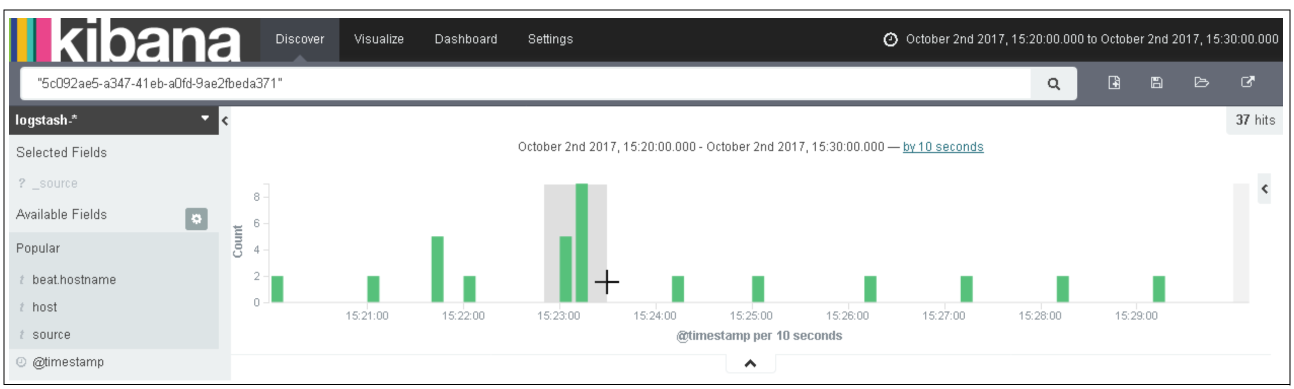


Figure 5-73 Filtering messages by time

You can also filter by the exact time by clicking the watch area at the upper right corner of the window and providing the exact date and time, as shown in Figure 5-74.

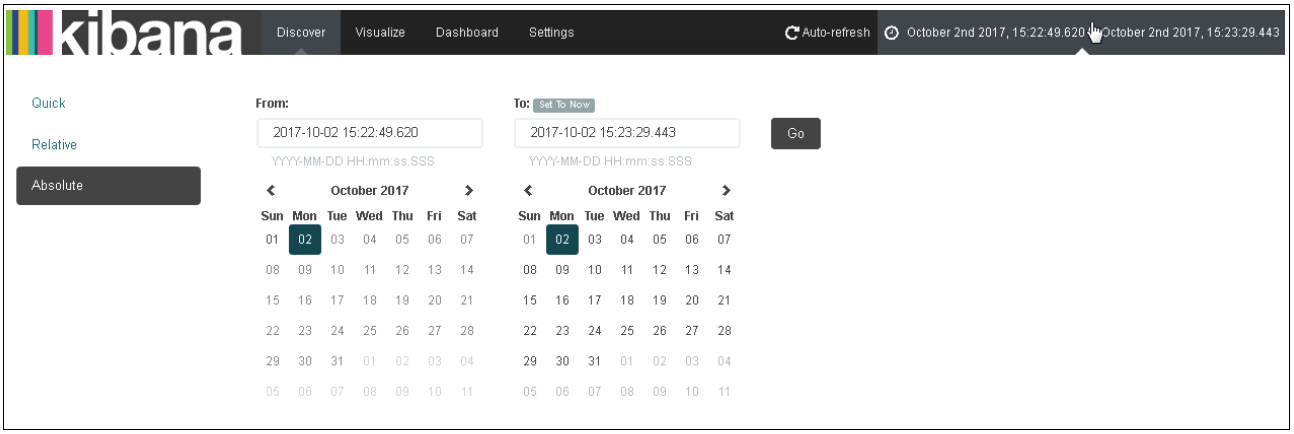


Figure 5-74 Filtering messages by the exact date and time

In our example, we use only the drag option, and did not filter by the exact date and time.

After you set the filter, you can display only the ERROR messages by filtering the `loglevel` to `ERROR`. In this case, the search obtained only two hits, as shown in Figure 5-75.

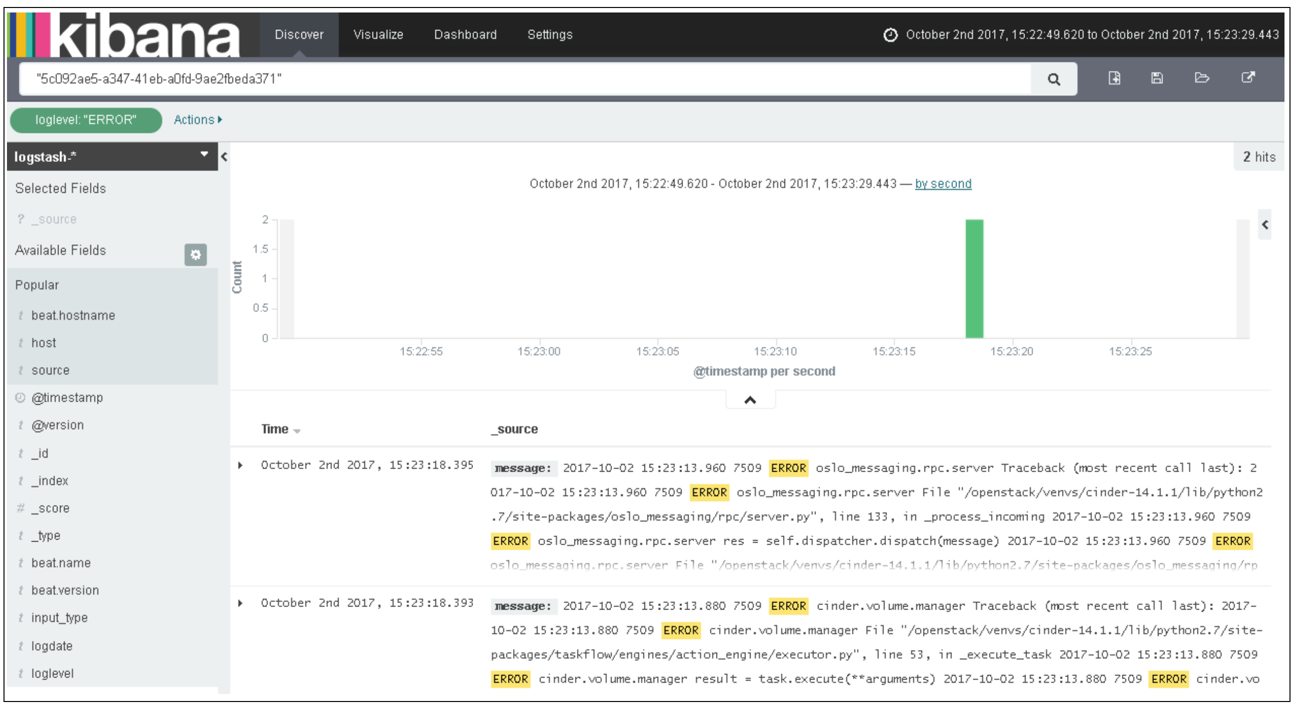


Figure 5-75 Messages that are filtered by time and loglevel

By expanding the messages, you can see the root cause of this issue, as shown in Figure 5-76.

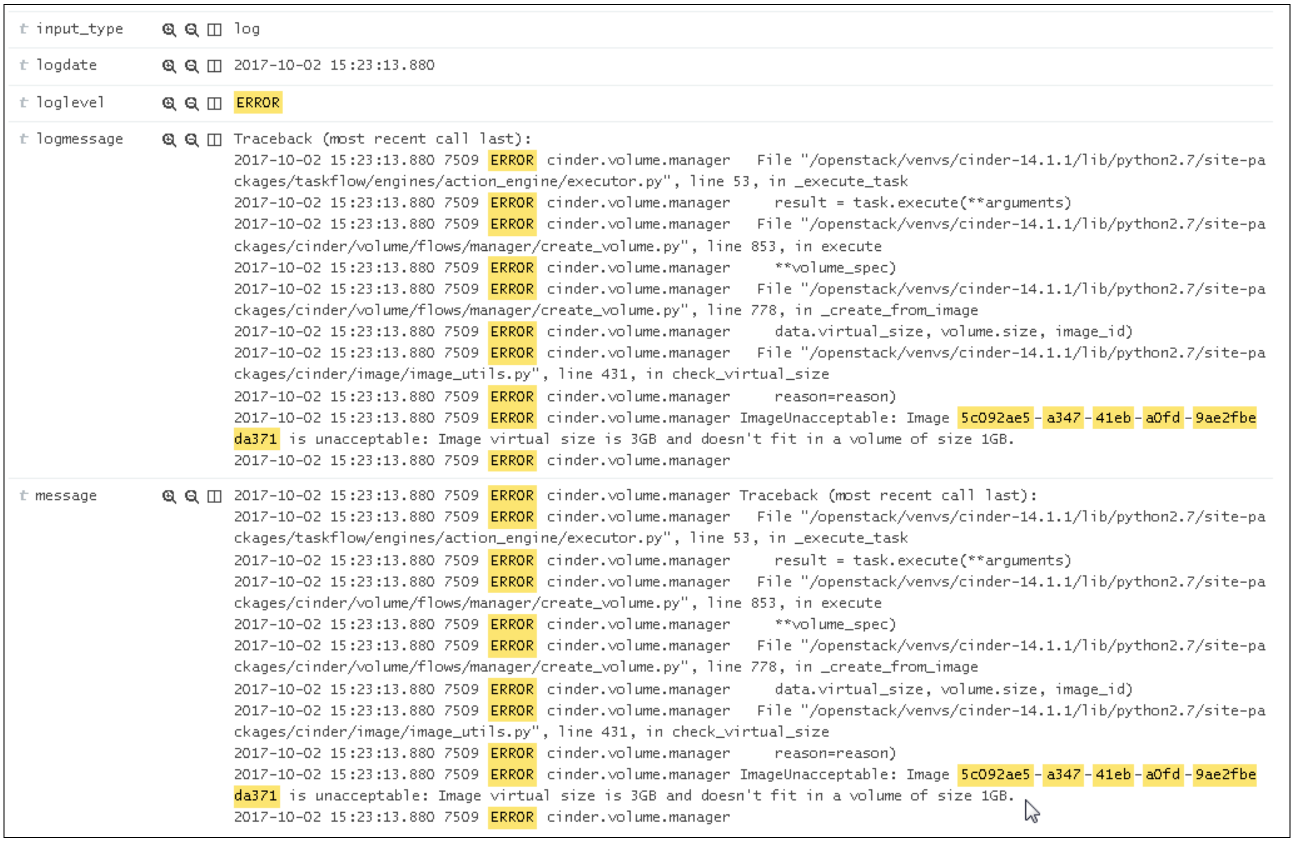


Figure 5-76 Root cause message of the failure

From the message in Figure 5-76, you can see that the root cause of the failure to spawn the VM is that the volume had only 1 GB, and the image requires 3 GB. With this information, you can select a bigger volume size when deploying your VM to avoid such errors.

6

Scaling

This chapter describes how to scale your cluster, and describes the considerations for expanding Open Platform for Database as a Service (DBaaS) on IBM Power Systems into a larger configuration size, commonly called *vertical scaling*.

This chapter also covers the aspects of horizontal scaling to expand units across existing hardware and improve load efficiency.

This chapter contains the following sections:

- Scaling up your cluster
- Horizontal scaling

6.1 Scaling up your cluster

Scaling up your Open Platform for DBaaS on Power Systems cluster results in more computing resources being available. You can scale up your cluster by adding more hardware, which increases the number of compute nodes, and storage disks. This section describes how to upgrade your Open Platform for DBaaS on Power Systems cluster from one configuration size to another one, and it also gives an overview of the scaling considerations for resizing your cluster.

Figure 3-11 on page 61 shows the current reference configuration sizes for the Open Platform for DBaaS on Power Systems solution. The starter cluster size shares servers for compute, controller, and Ceph. For this cluster configuration, scaling up to higher configurations with dedicated servers requires deployment changes in the existing nodes. These deployment changes include scaling back units that are deployed in a shared node. Scaling back units is described in“Scaling back” on page 199.

> **Note:** This section describes the software aspects of cluster scaling. To read about the hardware considerations for different configuration sizes, see Chapter 3, “Architecture” on page 43.

To perform an upgrade from an entry cluster size to a cloud scale size, you must add more compute nodes and storage disks to your existing cluster. These tasks are described in more detail in the following sections.

6.1.1 Adding a compute node

The compute node has the services that are responsible for starting instances in the cloud. If you want your cluster to increase its capacity to start instances with more computing power or larger flavors, complete these steps:

1. Configure the server to use static IP for IPMI on the baseboard management controller (BMC) interface. Set an IPMI user and password.
2. Power on the machine from the BMC or IPMI interface. Metal as a Service (MAAS) does the auto-discovery for interfaces that are connected to the same network. The server is automatically discovered by MAAS. For more information about how to manually add nodes to the cluster, see “Baremetal provisioning” on page 202. To power on the server by using IPMI, run the command that is shown in Example 6-1.

Example 6-1 Powering on the server through IPMI

```
$ ipmitool -I lanplus -H 192.168.20.107 -U ADMIN -P ADMIN power on
```

3. Check whether the node is auto-discovered by MAAS enlisting. Figure 6-1 shows the new node that was automatically discovered by MAAS enlisting.

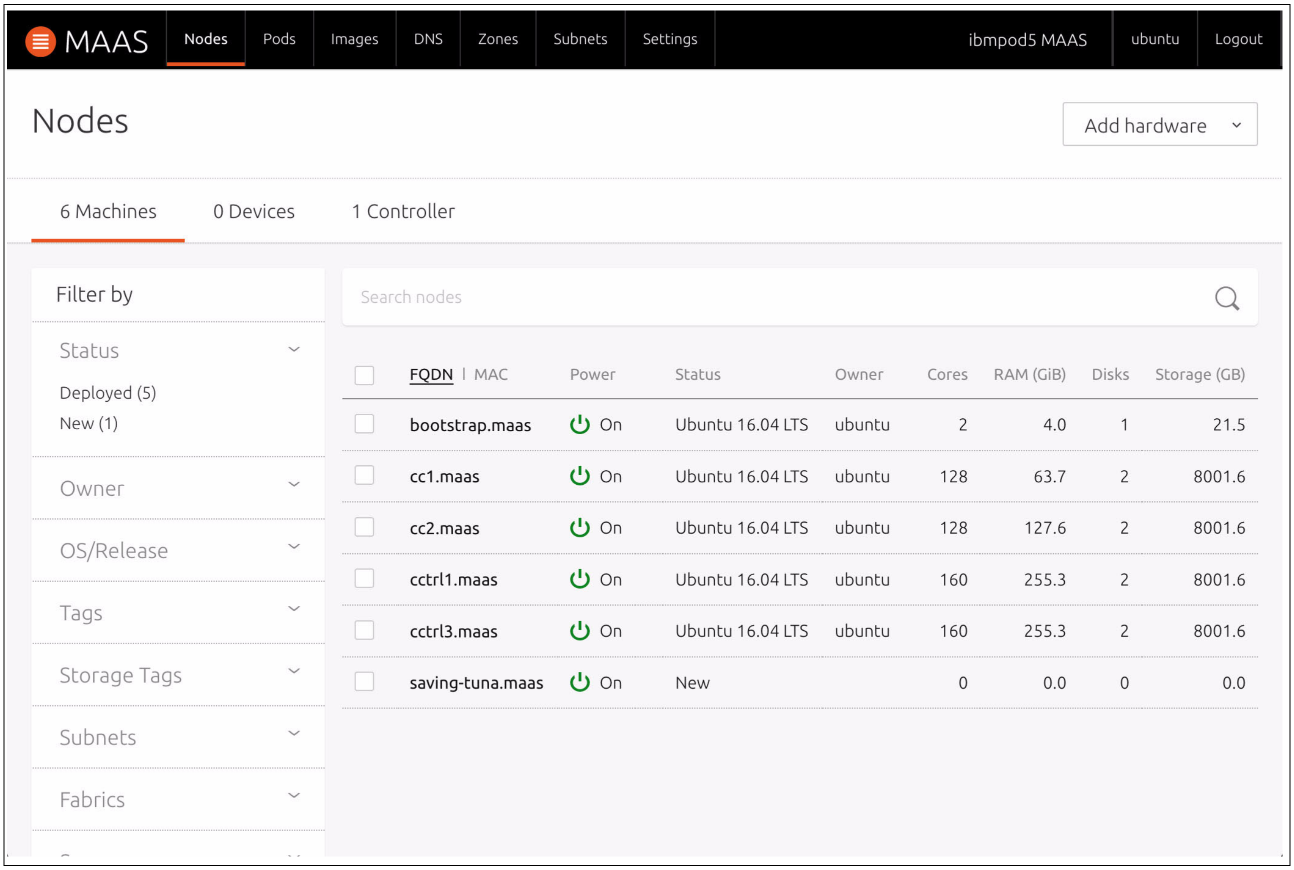


Figure 6-1 MAAS enlisting nodes

4. New discovered nodes receive an automatic name in MAAS. The node `saving-tuna.maas` in this example is the machine that MAAS detected. You can change the node's name by clicking the node and editing its name, as illustrated in Figure 6-2.

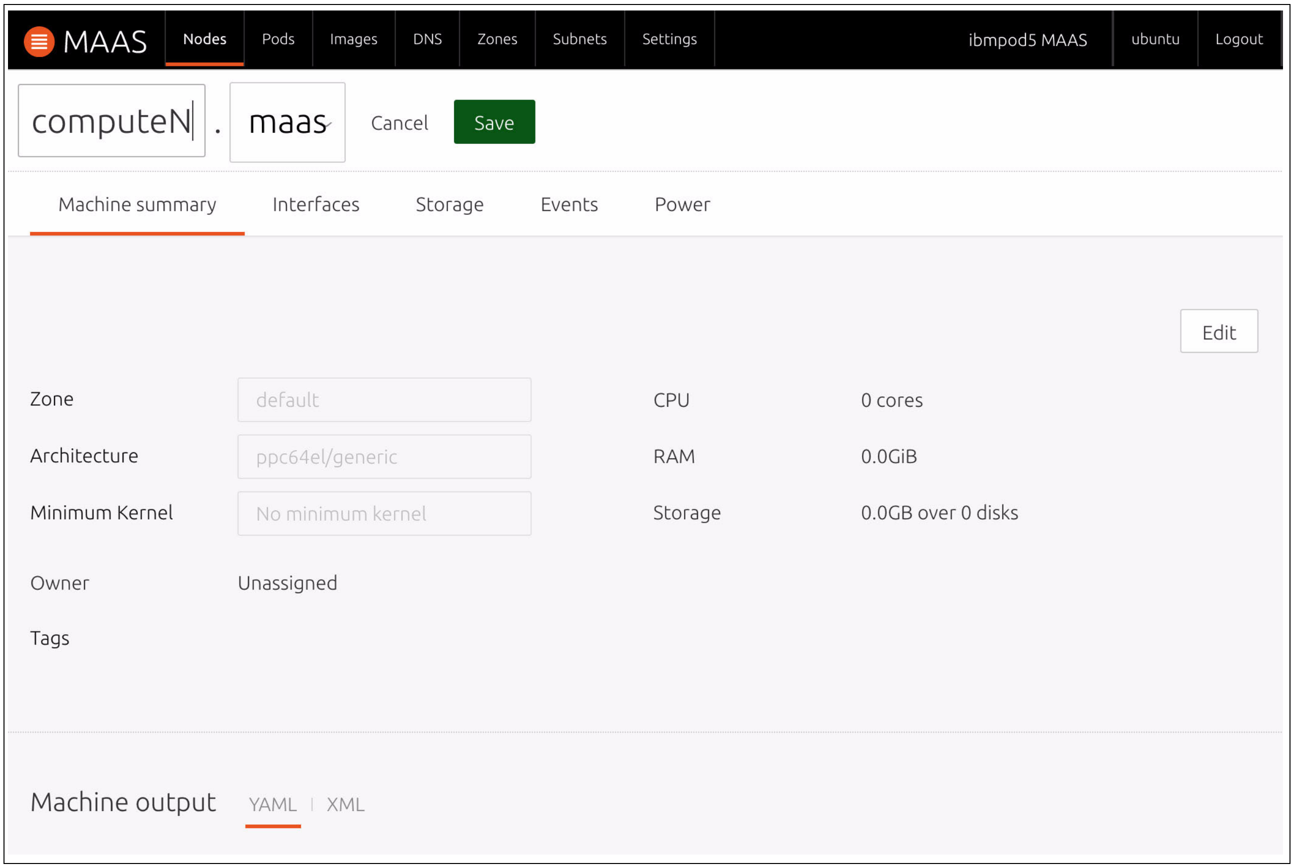


Figure 6-2 Renaming the discovered node

5. Commission the node by clicking **Commission** under the **Take action** menu, as shown in Figure 6-3. This action runs a series of scripts and checks to collect system information from the added node.

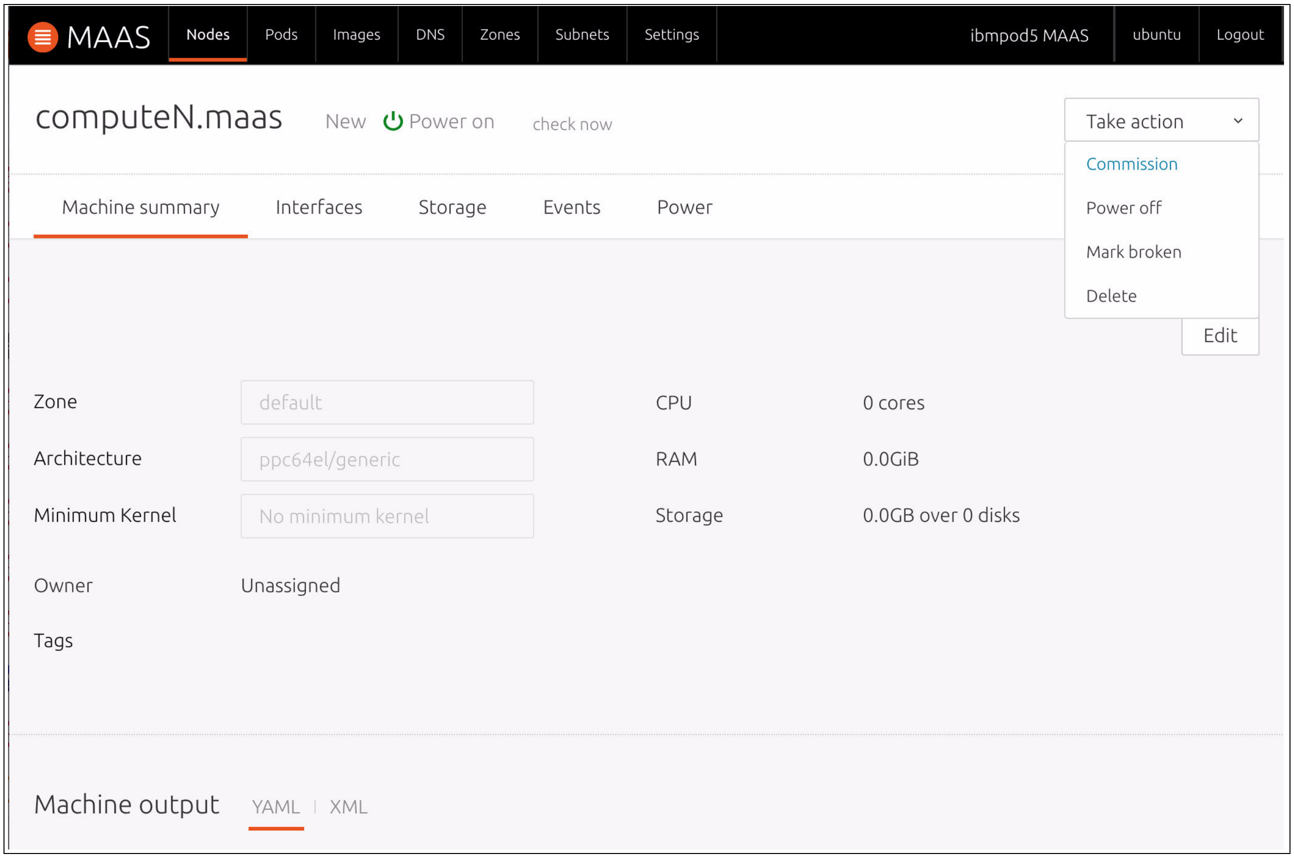
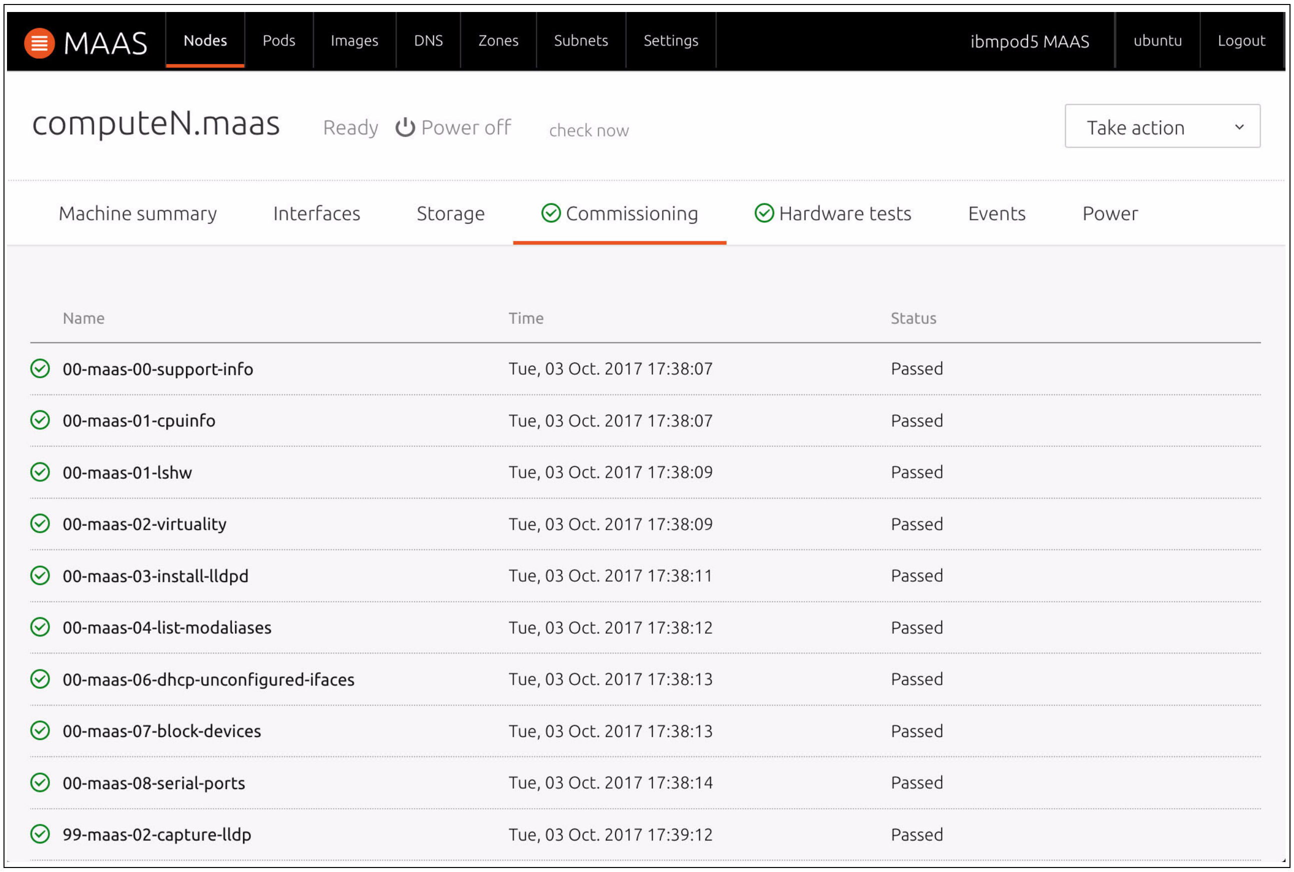


Figure 6-3 Commission node

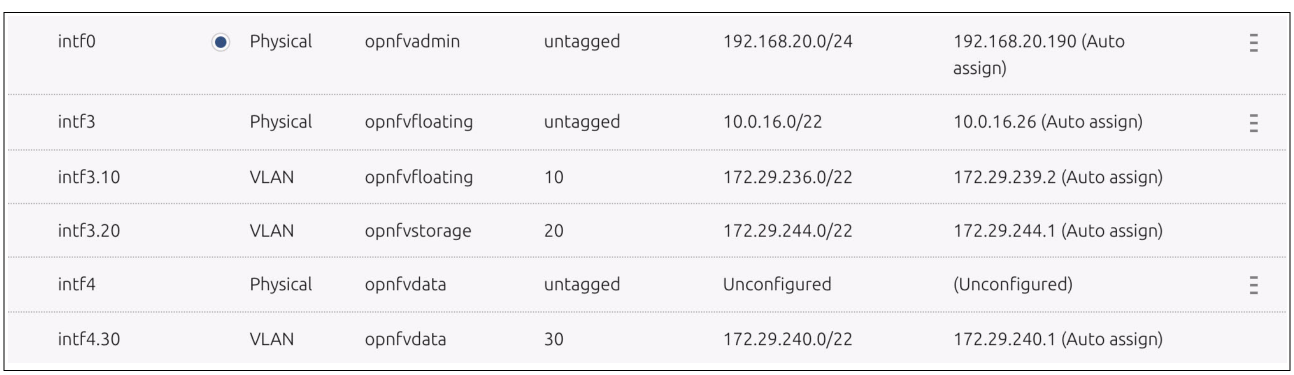
6. Commissioning scripts take time to run and collect all the system information from the discovered node. The commissioning status is illustrated in Figure 6-4.



Name	Time	Status
00-maas-00-support-info	Tue, 03 Oct. 2017 17:38:07	Passed
00-maas-01-cpuinfo	Tue, 03 Oct. 2017 17:38:07	Passed
00-maas-01-lshw	Tue, 03 Oct. 2017 17:38:09	Passed
00-maas-02-virtuality	Tue, 03 Oct. 2017 17:38:09	Passed
00-maas-03-install-lldpd	Tue, 03 Oct. 2017 17:38:11	Passed
00-maas-04-list-modaliases	Tue, 03 Oct. 2017 17:38:12	Passed
00-maas-06-dhcp-unconfigured-ifaces	Tue, 03 Oct. 2017 17:38:13	Passed
00-maas-07-block-devices	Tue, 03 Oct. 2017 17:38:13	Passed
00-maas-08-serial-ports	Tue, 03 Oct. 2017 17:38:14	Passed
99-maas-02-capture-lldp	Tue, 03 Oct. 2017 17:39:12	Passed

Figure 6-4 Commissioning status

7. After commissioning, you see the node's status change to Ready. You can match the machine constraints for deploying applications on your new node. Figure 6-5 illustrates the network interfaces that are required for adding units to your existing model. At the time of writing, network interfaces must be manually configured on MAAS. For detailed information, see 3.4, "Networking" on page 68.



intf0	Physical	opnfvadmin	untagged	192.168.20.0/24	192.168.20.190 (Auto assign)
intf3	Physical	opnfvfloating	untagged	10.0.16.0/22	10.0.16.26 (Auto assign)
intf3.10	VLAN	opnfvfloating	10	172.29.236.0/22	172.29.239.2 (Auto assign)
intf3.20	VLAN	opnfvstorage	20	172.29.244.0/22	172.29.244.1 (Auto assign)
intf4	Physical	opnfvdata	untagged	Unconfigured	(Unconfigured)
intf4.30	VLAN	opnfvdata	30	172.29.240.0/22	172.29.240.1 (Auto assign)

Figure 6-5 Server network interfaces

8. You can deploy compute services on a new node in the Juju GUI or CLI. To deploy a compute node through Juju's command line, run the command that is shown in Example 6-2 on page 195.

Example 6-2   Deploying a compute node

```
$ juju add-unit nova-compute
```

Note: You do not need to deploy all dependencies and services that run on a compute node. Juju charms automatically pull all the required units that are specified in the Open Platform for DBaaS on IBM Power Systems model.

Alternatively, you can add a nova-compute unit by using the Juju GUI. Under the Machines tab, add one nova-compute unit and click **Auto place**, as shown in Figure 6-6. Commit your changes to deploy the compute node.

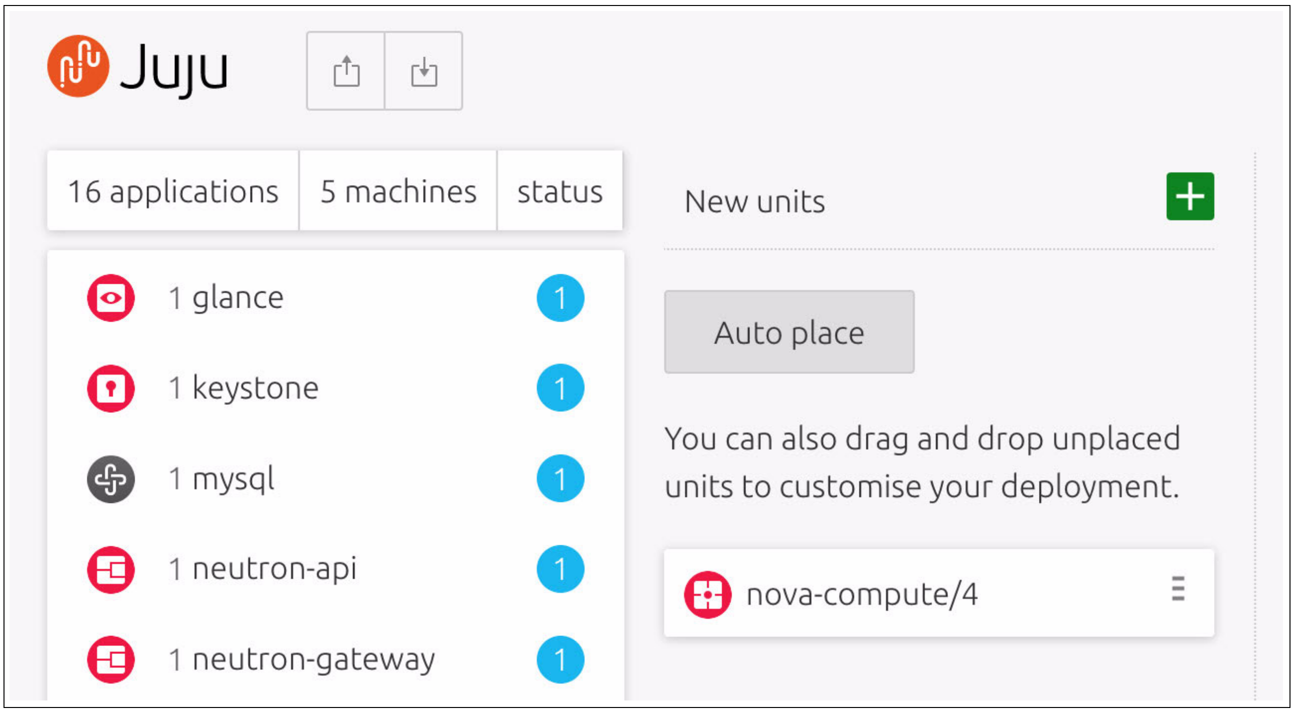


Figure 6-6   Adding a nova-compute unit on the Juju GUI

9. Check the deployment status by using the MAAS Nodes tab view, as illustrated in Figure 6-7.

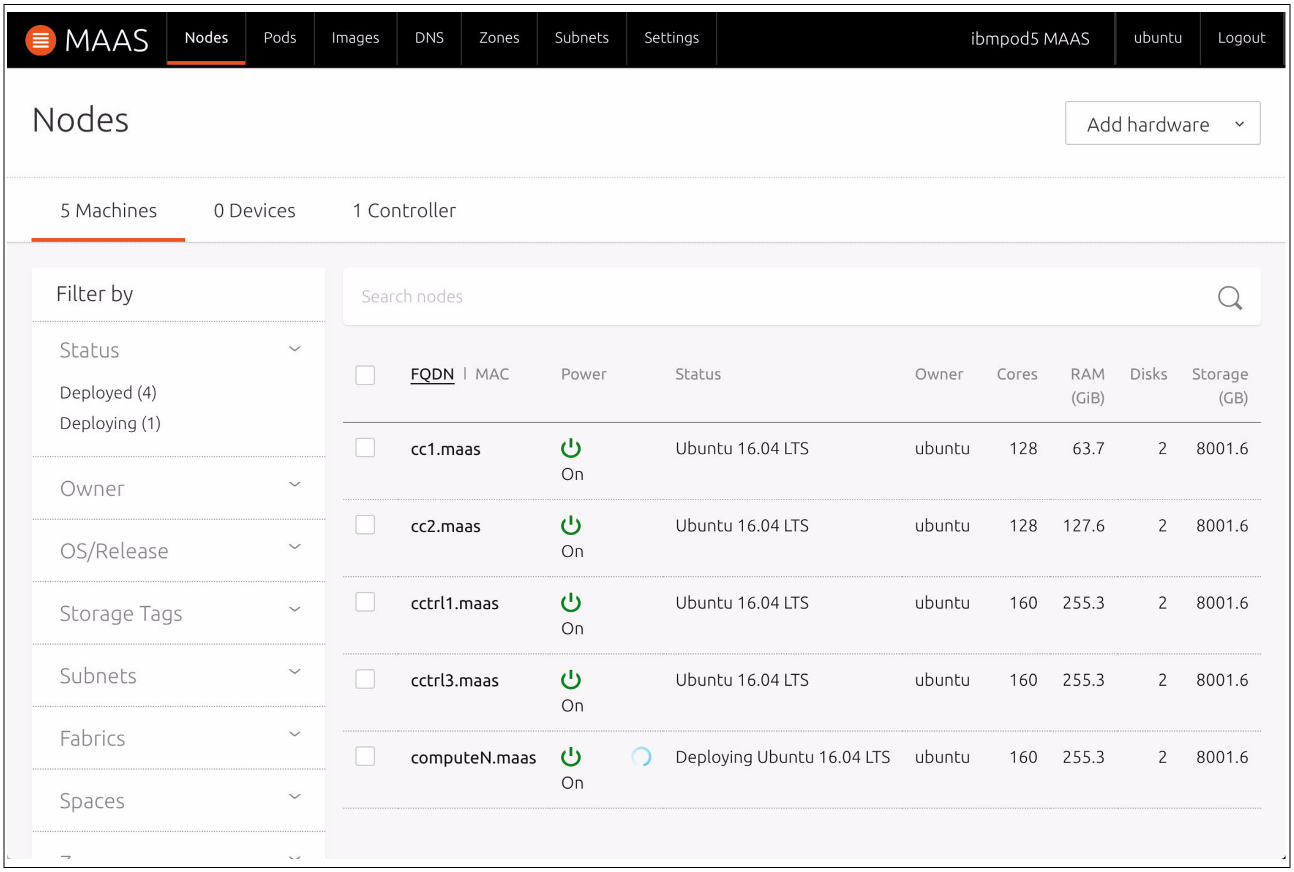


Figure 6-7 Checking the deployment status

10. You can check for deployment completion by using the MAAS Nodes tab. Figure 6-8 shows how the node status looks when deployment is complete.

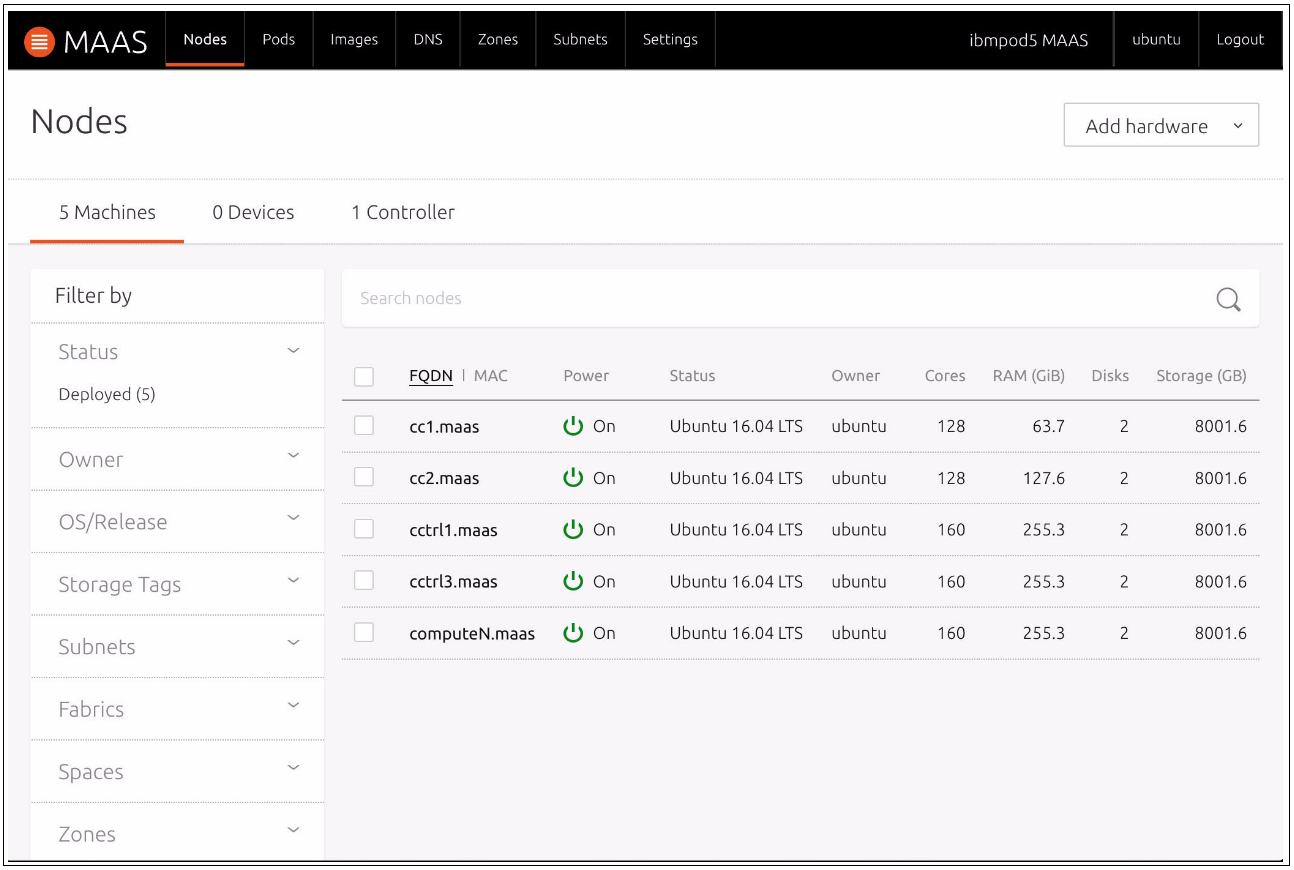


Figure 6-8 Checking the completion of the deployment

6.1.2 Adding a storage node

The storage node has the services that are responsible for storing and retrieving data for the database instances.

Adding a storage node is similar to adding a compute node from the deployment perspective. If you want your cluster to increase its capacity to store data, complete the steps from 6.1.1, "Adding a compute node" on page 190 until step 7 on page 194. Then, replace step 8 on page 194 with the command that is shown in Example 6-3.

Example 6-3 Adding a storage node

```
$ juju add-unit ceph-osd
```

Example 6-3 deploys storage services into the discovered node.

6.2 Horizontal scaling

You can expand services across the existing machines to make better usage of the hardware resources in the cluster. To deploy additional applications, add the unit to your model and commit your changes. Juju deploys the units according to the machine constraints. For more information about the constraints, see the Juju constraints documentation.

Figure 6-9 illustrates how to add a Ceph unit to the cluster.

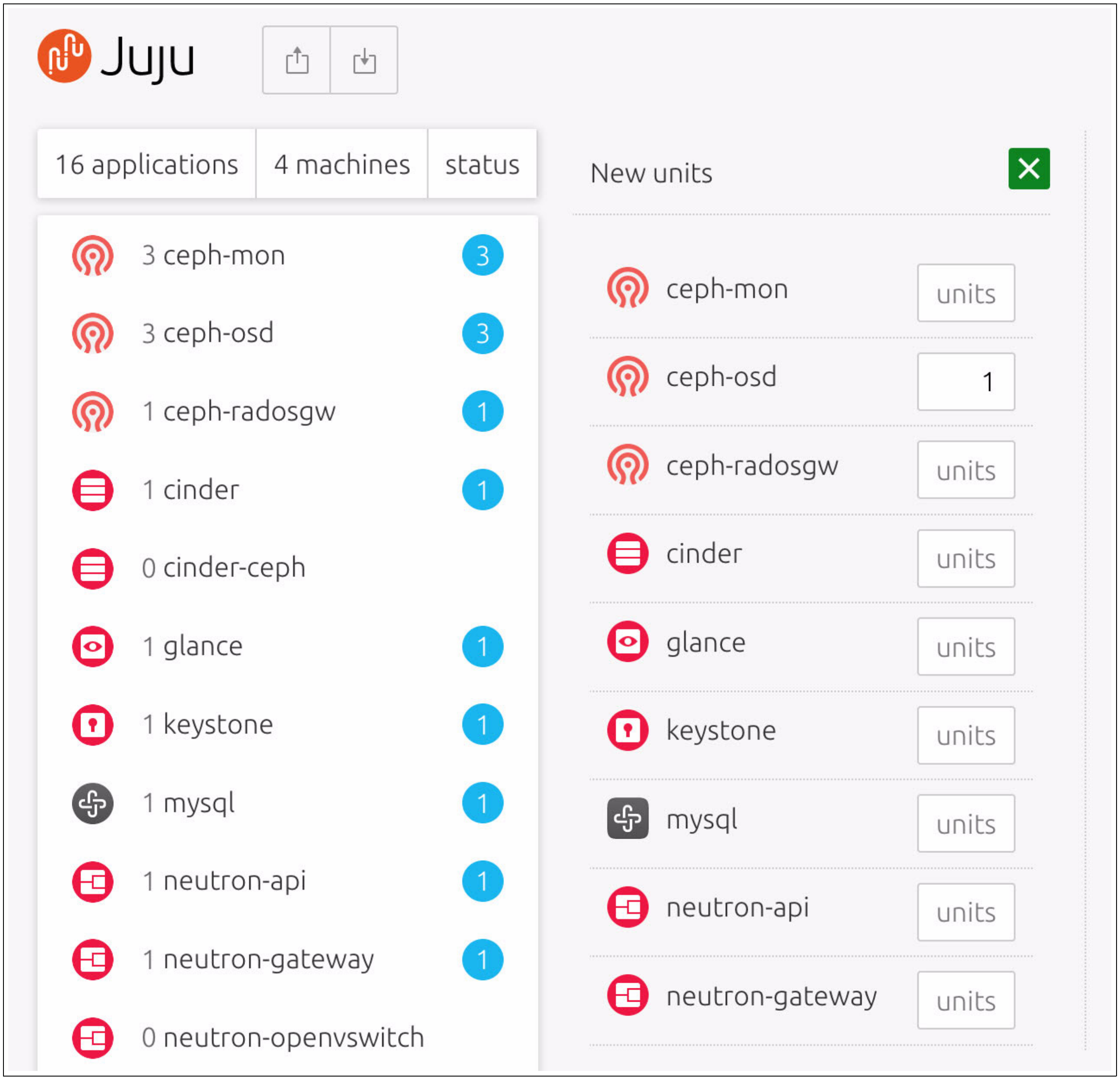


Figure 6-9 Scaling horizontally

Colocating

You can also colocate applications. In this mode, you specify the machine to which you want to deploy. Example 6-4 illustrates how to colocate an application on a machine.

Example 6-4 Colocating an application on a machine

```
$ juju add-unit ceph-osd --to 5
```

In Example 6-4, `ceph-osd` and all its relationships are deployed to node number 5. The equivalent action in the Juju GUI is to drag the unit you are adding to the machine you want to deploy.

Scaling back

You can shrink your deployment by removing applications from your cluster machines. To remove a unit, run the command that is shown in Example 6-5.

Example 6-5 Removing a unit

```
$ juju remove-unit nova-compute/4
```

Note: If you remove a unit that has no running units, controllers, or containers, it is destroyed automatically.

A

Servers provisioning and deployment

This appendix describes the deployment tools for Open Platform for Database as a Service (DBaaS) on IBM Power Systems and how its components are installed into a cluster. This appendix also describes how you can build additional data store images.

This appendix includes the following sections:

- Baremetal provisioning
- OpenStack deployment
- Alternative deployment
- Image building

Baremetal provisioning

The Open Platform for DBaaS on Power Systems solution uses open source tools for installing and configuring a cluster of Power Systems baremetal servers.

The preferred method automates the baremetal deployment through Metal as a Service (MAAS), an open source tool that supports OpenPower hardware. It performs the provisioning of physical servers by installing an operating system (OS) and configuring the network.

> **Note:** The alternative deployment method is to use the Cluster Genesis scripts. Cluster Genesis deployment is described in "Alternative deployment" on page 208.

MAAS requires a physical server for baremetal provisioning. The MAAS deployment server runs on Power Systems or x86 machines. You can install MAAS by following the documentation found at Ubuntu provisioning.

To do baremetal provisioning, complete the following steps:

1. The Open Platform for DBaaS on Power Systems solution deploys baremetal servers by using an Ubuntu Power Little-Endian (ppc64el) image. To enable the ppc64el image, go to the Images tab and select the **ppc64el** check box, as illustrated in Figure A-1. Click **Save selection** to start downloading the image and making it available for deployment.

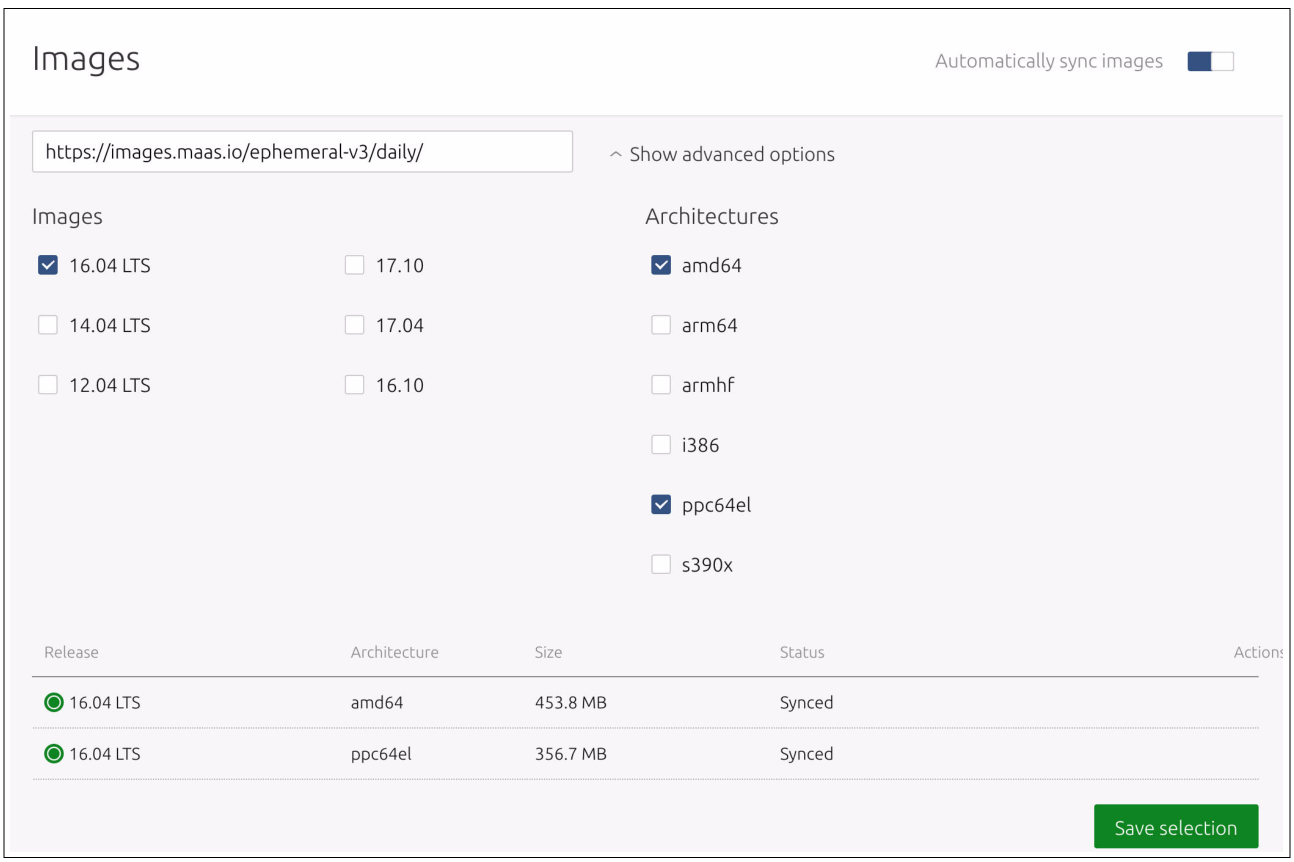
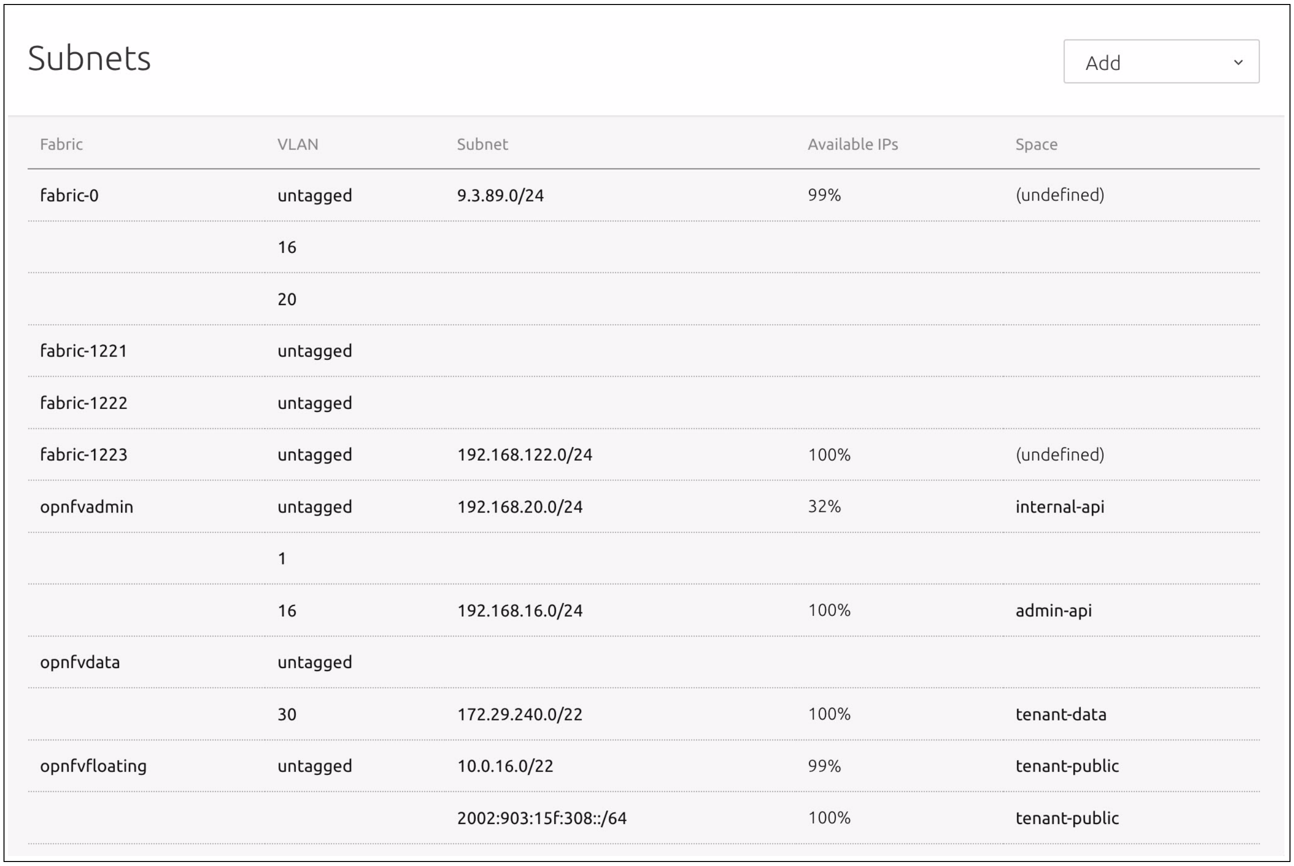


Figure A-1 Enabling the ppc64el image

2. To manage subnets and VLANs, click the **Subnets** menu, as shown in Figure A-2. For more information about how to configure subnets and VLANs for the Open Platform for DBaaS on Power Systems solution, see 3.4, “Networking” on page 68.



Subnets

Add

Fabric	VLAN	Subnet	Available IPs	Space
fabric-0	untagged	9.3.89.0/24	99%	(undefined)
	16			
	20			
fabric-1221	untagged			
fabric-1222	untagged			
fabric-1223	untagged	192.168.122.0/24	100%	(undefined)
opnfvadmin	untagged	192.168.20.0/24	32%	internal-api
	1			
	16	192.168.16.0/24	100%	admin-api
opnfvdata	untagged			
	30	172.29.240.0/22	100%	tenant-data
opnfvfloating	untagged	10.0.16.0/22	99%	tenant-public
		2002:903:15f:308::/64	100%	tenant-public

Figure A-2 Managing VLANs and subnets

You can view, add, remove, and edit servers by using the Nodes menu. Servers are listed with their hardware specifications as processors, memory, and disk, as shown in Figure A-3.

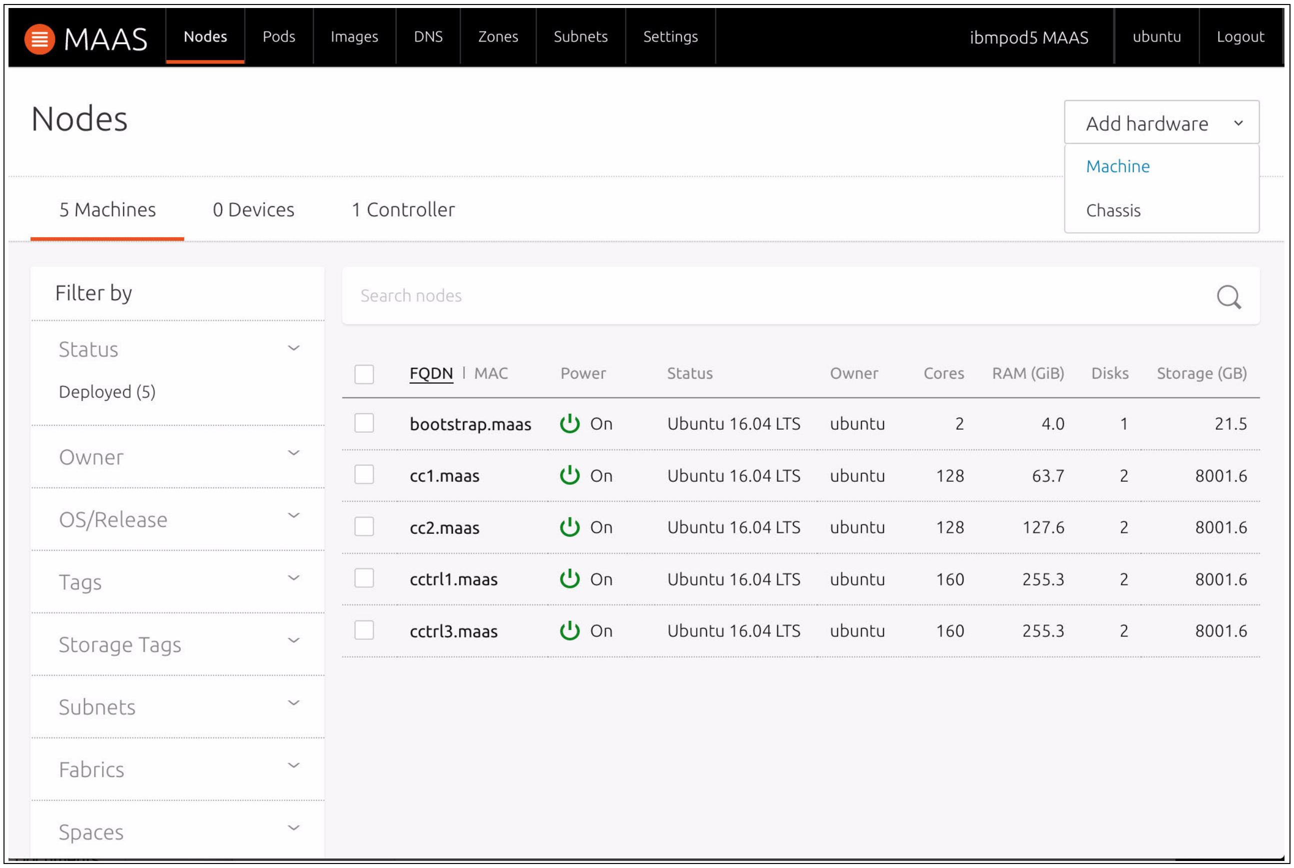


Figure A-3 Managing servers

3. To add a server, click **Add hardware** and select **Machine**. Type a name for the node, select **ppc64el/generic** for the **Architecture**, and type the node's MAC address. For **Power type**, choose **IPMI** (Intelligent Platform Management Interface (IPMI)), as shown in Figure A-4.

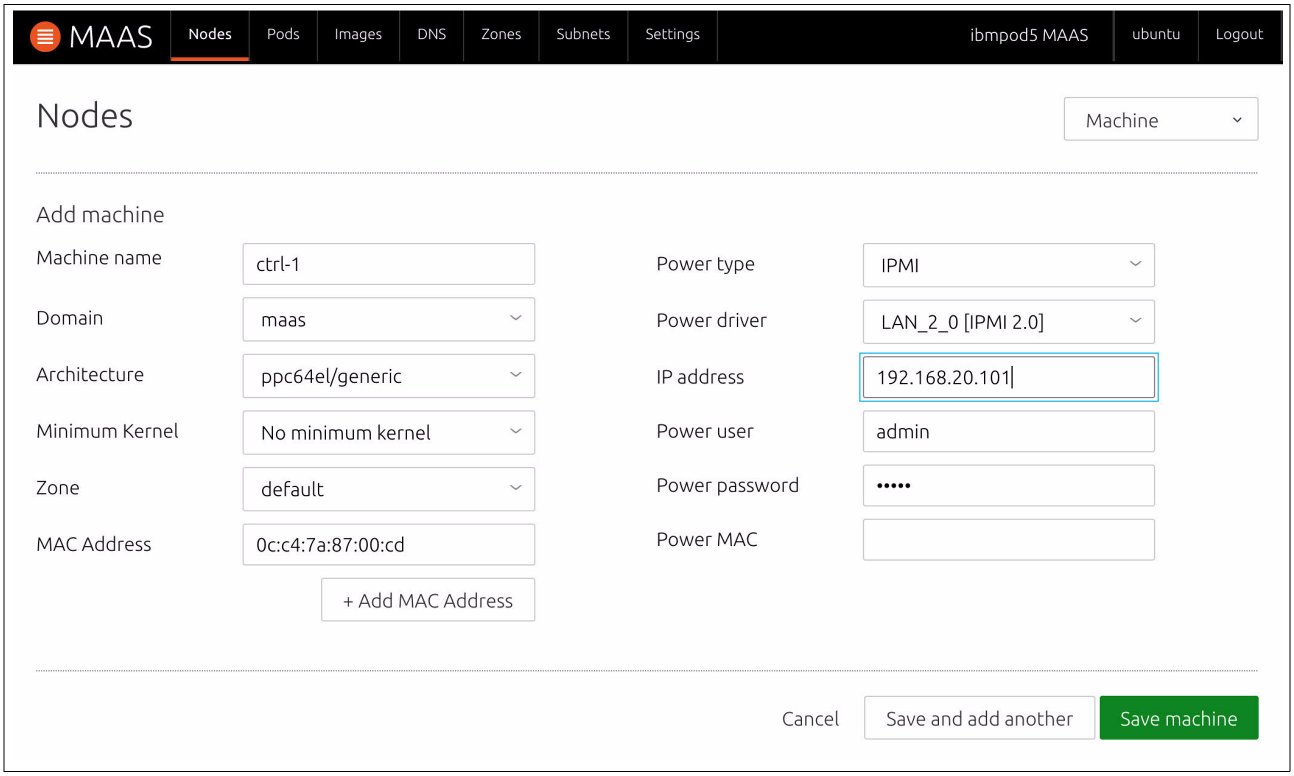


Figure A-4 Adding a server

Note: At the time of writing, Open Platform for DBaaS on Power Systems integration with MAAS supports only semi-automated baremetal deployment for servers and networking. IBM intends to improve the deployment automation features by using MAAS in upcoming releases of the Open Platform for DBaaS on Power Systems solution.

After you add a server, MAAS starts commissioning your node by connecting to IPMI and running scripts for hardware discovery. If the commissioning succeeds, the node status changes to Ready and you can deploy your server.

4. To deploy a server, select it from the nodes list and click **Deploy** under the **Take action** menu. The available actions are illustrated in Figure A-5.

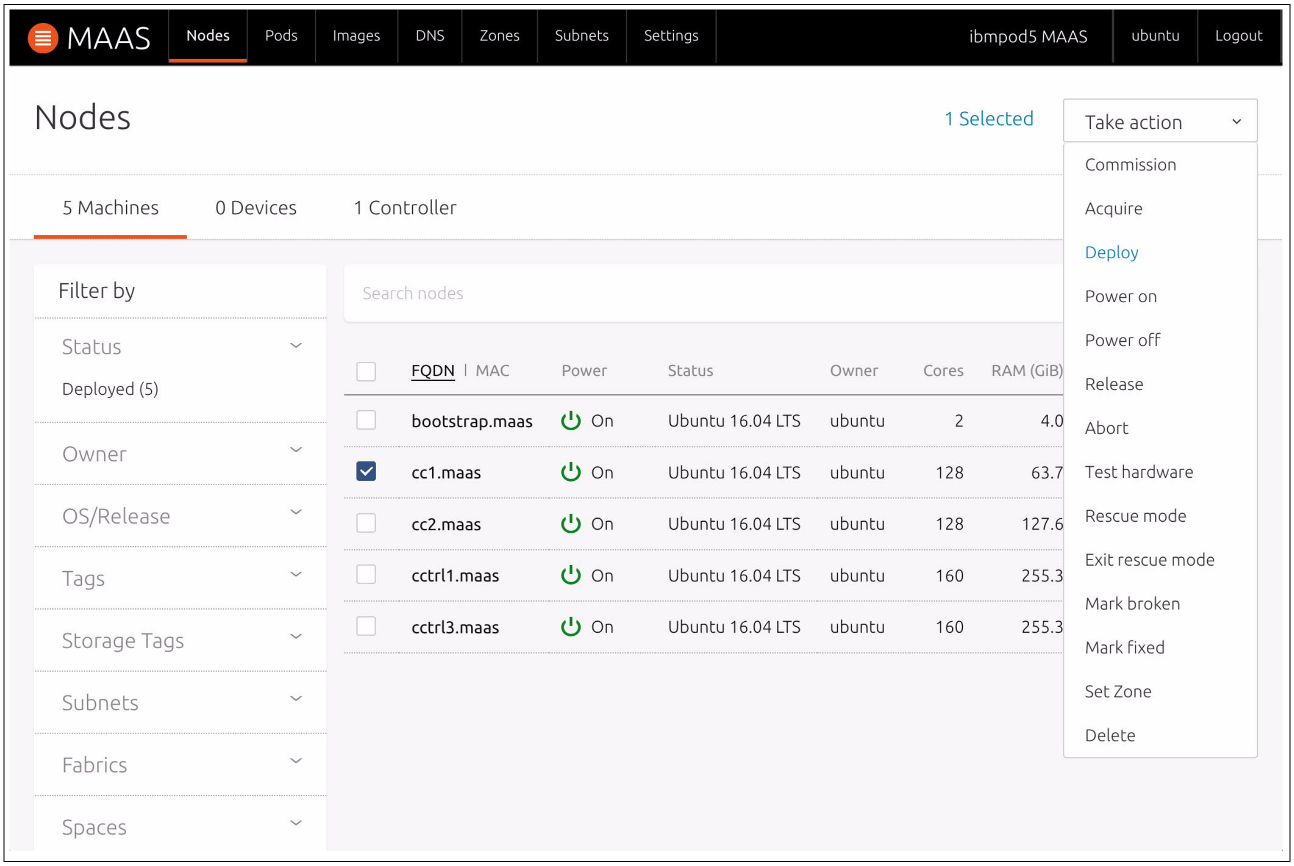


Figure A-5 Deploying a server

For more information, see the MAAS documentation.

OpenStack deployment

The Open Platform for DBaaS on Power Systems solution deploys a customized charms bundle to install and configure the cluster. It uses the OpenStack charms bundle as the base model, and integrates additional charms into a single scalable deployment.

The charms bundle includes the following units:

- ceph-mon
- ceph-osd
- ceph-radosgw
- cinder
- cinder-ceph
- glance
- keystone
- neutron-api
- neutron-gateway
- neutron-openvswitch
- nova-cloud-controller
- nova-compute

- ntp
- openstack-dashboard
- rabbitmq-server
- trove
- kibana
- nagios

You can use the Juju GUI or command-line interface (CLI) for deploying Open Platform for DBaaS on your cluster. The Open Platform for DBaaS on Power Systems charms bundle can be downloaded from GitHub.

To deploy the Open Platform for DBaaS on Power Systems solution, complete these steps:

1. Open the Juju GUI in your browser.
2. Get the Open Platform for DBaaS on Power Systems charms bundle from GitHub. Save the YAML file to your local disk.
3. Import the YAML file as a local bundle. Click **Import**, as shown in Figure A-6. You can also drag the YAML file in to the model to import the local bundle.

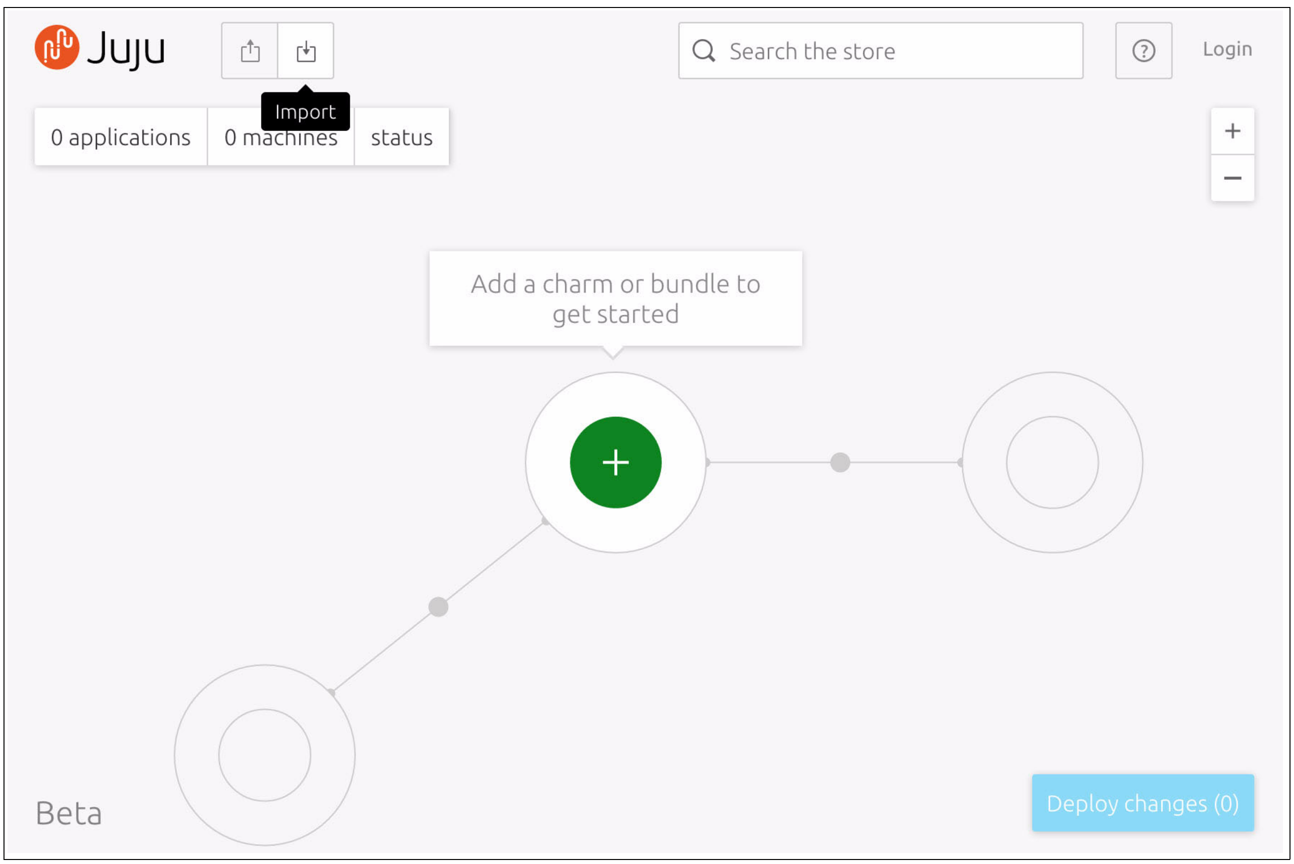


Figure A-6 Importing a local bundle

4. Figure A-7 illustrates a deployment model for OpenStack base bundle. You can review the model applications and machines in the left pane. Click **Deploy changes** to confirm.

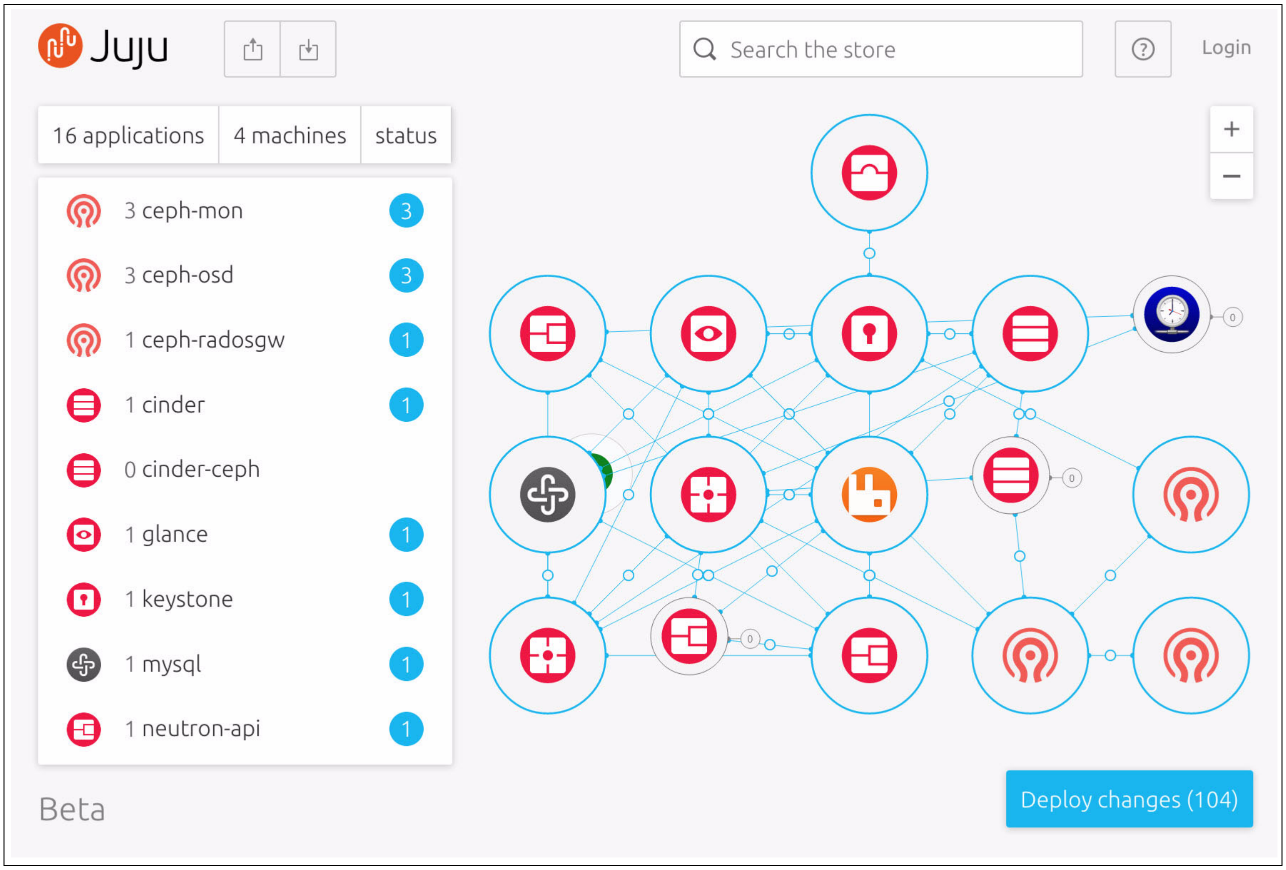


Figure A-7 Deploying a Juju charms bundle

> **Note:** You can also deploy Juju charms bundle by using the CLI. For more information about Juju charms and bundles, see the Juju Charms documentation.

Alternative deployment

Cluster Genesis is an open source deployment tool that automates and simplifies cluster configuration of OpenPower baremetal servers. It combines baremetal provisioning and OpenStack services configuration for a full cluster deployment of the Open Platform for DBaaS on Power Systems solution.

> **Note:** The Cluster Genesis open source tool is provided *as is* and it is *not* officially supported by the Open Platform for DBaaS on Power Systems solution.

Here are the steps that are involved in deploying a cluster:

1. Obtain the configuration file.
2. Tailor the configuration for your environment.
3. Validate the configuration file.

4. Provision the cluster.
5. Configure the operational management tools.

These steps are performed by the Cluster Genesis tool. This section contains an overview of the Cluster Genesis deployment. For more information, see the Cluster Genesis documentation.

Obtaining the configuration file

The deployment automation tool uses a YAML file to specify the target cluster configuration. The configuration file defines the settings and details that are required for the deployment, including the IP address locations of the managed switches and their attached servers.

You can get a copy of the configuration from GitHub.

Tailoring the configuration for your environment

The `config.yaml` file contains much configuration information. To enable a cluster that is tailored to your environment, edit the YAML file by replacing the configuration parameters with your data. Here are the editable parameters:

- Management network IP address:

  ```
  ipaddr-mgmt-network: 192.168.16.0/24
  ```

- Management switch IP address:

  ```
  ipaddr-mgmt-switch:
     rack1: 192.168.16.20
  ```

- Data switch IP addresses:

  ```
  ipaddr-data-switch:
     rack1:
        192.168.16.25
        192.168.16.30
  ```

- External floating IP address:

  ```
  external-floating-ipaddr: 10.0.16.50
  ```

- External network settings:

  ```
  networks:
     external1:
        description: organization site or external network
        addr: 10.0.16.0/22
        broadcast: 10.0.19.255
        gateway: 10.0.16.1
        dns-nameservers: 10.0.16.200
        dns-search: mycompany.domain.com
        method: static
        eth-port: osbond0
  ```

- Controller node settings:

  ```
  node-templates:
     controllers:
        hostname: controller1
        userid-ipmi: ADMIN
        password-impi: admin
        cobbler-profile: ubuntu-16.04.1-server-ppc64el
  ```

Validating the configuration file

To ensure that the format of the specified configuration file is valid, validate it by running the commands that are shown in Example A-1.

Example A-1 Validating the configuration file

```
$ git clone https://github.com/open-power-ref-design/dbaas
$ cd dbaas
$ TAG=$(git describe --tags $(git rev-list --tags --max-count=1))
$ cd ..
$ sudo apt-get install python-pip
$ sudo pip install pyyaml
$ git clone https://github.com/open-power-ref-design-toolkit/os-services
$ cd os-services
$ git checkout $TAG
$ ./scripts/validate_config.py --file ../dbaas/config.yml
$ cd ..
```

Provisioning the cluster

Before provisioning the cluster, ensure that all nodes have direct access to the internet. If the cluster is being configured in a private network without direct internet access, then the deployer node needs internet access so that it can act as a NAT host to route all the nodes in the cluster.

After the deployment cluster's configuration is finalized, the cluster can be provisioned by starting the automation tool. The tool runs the commands that are shown in Example A-2.

Example A-2 Provisioning the cluster

```
$ git clone https://github.com/open-power-ref-design-toolkit/cluster-genesis
$ cd cluster-genesis
$ TAG=$(git describe --tags $(git rev-list --tags --max-count=1))
$ git checkout $TAG
$ /scripts/install.sh
$ source scripts/setup-env
$ gen deploy
```

Configuring OpenStack services

In a typical deployment, the provisioning takes a couple hours to complete the installation of the OS on all nodes. Upon the completion of the installation, it is ready for bootstrapping the cluster for OpenStack services.

The bootstrap step is automatically triggered when cluster-genesis is completed. Various OpenStack parameters must then be configured. To prepare for this phase, collect the information that is shown in Table A-1.

Table A-1 Information parameters that are required

Parameter	Description	Example
Keystone password	The password that is used for the OpenStack authentication service.	mypassword
Virtual Router Redundancy Protocol (VRRP) ID	Virtual Router ID of 1 - 255 and unique across the network.	202

Parameter	Description	Example
USED IPs	The range of IP addresses in the private network that is already taken, and cannot be assigned to nodes in the cluster. This includes the addresses that are assigned by cluster-genesis and those reserved for use by Trove. In this example, 172.29.236.100 - 200 are reserved for Trove.	172.29.236.100..172.29.236.200 172.29.236.1..172.29.236.50 172.29.240.1..172.29.240.50 172.29.244.50..172.29.244.50

For more information about the various options to configure the OpenStack deployment, the openstack-ansible documentation.

The minimal configuration set is as follows:

1. Log in to the first controller node (ctrl-1) by running the following command:

```
$ ssh ctrl-1
```

2. Edit the keystone stanza and set the keystone password in the `/etc/openstack_deploy/user_secrets.yml` file by running the following command:

```
$ vi /etc/openstack_deploy/user_secrets.yml file
```

Here is the output of the command:

```
## Keystone options
keystone_container_mysql_password:
keystone_auth_admin_token: password
keystone_auth_admin_password:
keystone_service_password:
keystone_rabbitmq_password:
```

3. Set the external virtual router ID in the `/etc/openstack_deploy/user_variables.yml` file. The valid range for this parameter is 1 - 255 and it must be unique for each cluster.

```
haproxy_keepalived_external_virtual_router_id: 202
```

4. Edit the `/etc/openstack_deploy/openstack_user_config.yml` file to reserve IP addresses for the 172.X.X.1 - 50 networks. For the 172.29.236.X network, the range 100 - 200 is also reserved. This is done by using the `used_ips` field that is described in `/etc/openstack_deployopenstack_user_config.yml.example`. The following values can be placed just before the global_overrides field:

```
$ vi /etc/openstack_deploy/openstack_user_config.yml
```

Here is the output of the command:

```
used_ips:
   - “172.29.236.100,172.29.236.200”
   - “172.29.236.1,172.29.236.50”
   - “172.29.240.1,172.29.240.50”
   - “172.29.244.50,172.29.244.50”
```

5. Now, the cluster is ready to complete the final step of deployment. Run the following commands:

```
cd os-services
$ ./scripts/create-cluster.sh 2>&1 | tee -a /tmp/create-cluster.out
```

Note: Monitor `/tmp/create-cluster.out` for progress and indication of completion. You can verify whether the cluster is operational by following the instructions that are found at GitHub.

Configuring the operational management services

The Open Platform for DBaaS on Power Systems solution uses popular DevOps tools that provide additional function to monitor availability and health of your cluster. These tools are Nagios Core and Elasticstack. To read more about operational management tools, see OpsMgr for Cloud.

Image building

The Open Platform for DBaaS on Power Systems solution provides the images for all the supported databases with their respective versions, as listed in Table 2-2 on page 41. The available images that are uploaded to Glance can be displayed by clicking the **Images** option under **Project** or **Admin** from the left pane of the Dashboard.

To get a different image, you must build it by using one of the methods that are provided by the deployment tools. These building methods are described in more detail in the following sections.

The dbimage-builder charm

The dbimage-builder charm uses the dbimage-builder tool to create a bootable virtual disk image that is configured to provide a user-specified database.

Deployment requirements

The dbimage-builder charm is a subordinate charm of Trove, so the Trove charm must be installed before deployment. The charm also requires access to OpenStack for creating a virtual machine (VM). This VM is used for creating images and needs an Ubuntu image, Trove tenant network, and an external network.

During deployment, the charm attempts to use configuration defaults to create the VM. If this fails, the charm monitors the configuration updates and attempts to create the VM again.

Defaults

Here are the configuration defaults:

```
vm_image     = xenial-1604
external_net = external_net
trove_net    = trove_net
```

Where:

`vm_image`
: The Ubuntu image that is used to create the VM. Defaults to `xenial-1604`.

`trove_net`
: The OpenStack network that is used by Trove tenants. Defaults to `trove_net`.

`external_net`
: The OpenStack network that is used to connect externally. Defaults to `external_net`.

Deployment commands

Example A-3 shows the deployment commands.

Example A-3 Deployment commands

```
git clone https://github.com/open-power-ref-design-toolkit/dbimage-builder-charms
juju deploy dbimage-builder
juju add-relation dbimage-builder trove
```

Command for the dbimage-make

Here is the command for running dbimage-make:

```
juju run-action dbimage-builder/<unit_number> dbimage-make db=<db-name> >
[version=<version> ] [ c=True | e=True ] [ keyname=<keyname> ]
```

This command creates a bootable OS image that contains the named database and database version, and creates an OpenStack Trove data store from the image. The image is built by using the OpenStack diskimage-builder (DIB) project. Here are the guidelines regarding the command parameters:

- The **db** parameter must be either **mariadb**, **mongodb**, **mysql**, **postgresql**, or **redis**.
- The **c** parameter may be specified to select the community edition of a database if one is provided and supported by this tool. The **e** argument may be specified to select the enterprise edition of a database if one is provided and supported by this tool. When **c** and **e** are not specified, the selection defaults to a distribution-provided database if one is provided and supported by the tool.
- The **keyname** parameter names an SSH key pair that is registered with OpenStack. If this parameter is specified, then the public SSH key is obtained from OpenStack and is placed in a virtual disk image in `/home/ubuntu/.ssh/authorized_keys`. This is intended for DBA access.

The dbflavor commands

These commands are used after the data store is created by the dbimage-make command. They are used to create the data store flavors, which dictate the capacity of data store instances vCPUs, RAM, and storage.

The commands that are shown in Example A-4 show, change, and upload database flavors for Glance images that are created by the dbimage-make command.

Example A-4 The dbflavor commands

```
juju run-action dbimage-builder/<unit_number> dbflavor-show db=<db-name> >
   [predefined=True ]
juju run-action dbimage-builder/<unit_number> dbflavor-change db=<db-name> >
   flavor=flavor-name { [ vcpus=<val> ] | [ mem=val ] | [ vdisk=<val> ] }
juju run-action dbimage-builder/<unit_number> dbflavor-upload db=<db-name>
```

The dibimage-builder scripts

The alternative method for building images in the Open Platform for DBaaS on Power Systems solution is by using the dbimage-builder scripts.

> **Note:** This section contains a simplified guide for building database images through dibimage-builder scripts to be used in the Open Platform for DBaaS on Power Systems solution. For more information, see the Git repository's readme file.

To build an image, complete these steps:

1. Create the deployer VM with one vCPU, 4 GB of RAM, and 80 GB of storage.
2. Create the database builder VM with four vCPUs, 12 GB of RAM, and 100 GB of storage.
3. Ensure the SSH connectivity between the deployer, builder, and controller.
4. Clone the os-services repository and find the `dbimage-builder` directory by running the following commands:

   ```
   git clone https://github.com/open-power-ref-design-toolkit/os-services
   cd os-services/osa/dbaas/dbimage-builder
   ```

5. Edit `scripts/dbimagerc replacing` <*controller-ip*> and input the controller's address:

   ```
   export DBIMAGE_CONTROLLER_IP=<controller-ip>
   ```

6. Run the following command to create the image, upload it to OpenStack Glance, and associate it with a Trove data store:

   ```
   scripts/dbimage-make.sh -i <builder-vm-ip> -d <database-name> -k <ssh-keypair>
   ```

> **Note:** The SSH key pair name that is provided is a valid keypair that was registered in OpenStack, and it is used by the database administrator to access the instances.

Related publications

The publications that are listed in this section are considered suitable for a more detailed description of the topics that are covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Some publications that are referenced in this list might be available in softcopy only.

- *IBM Power System S821LC Technical Overview and Introduction*, REDP-5406
- *IBM Power System S822LC Technical Overview and Introduction*, REDP-5283

You can search for, view, download, or order these documents and other Redbooks, Redpapers, web docs, drafts, and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- Cluster Genesis documentation:

  http://cluster-genesis.redthedocs.io/en/latest
- Juju charms documentation:

  https://jujucharms.com/docs
- MAAS documentation:

  https://docs.ubuntu.com/maas
- Metal as a Service (MAAS) installation:

  https://www.ubuntu.com/download/server/provisioning
- Nagios documentation

  https://library.nagios.com/library/products/nagios-core/manuals
- The openstack-ansible documentation:

  https://docs.openstack.org/project-deploy-guide/openstack-ansible-newton

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

IBM Open Platform for DBaaS on IBM Power Systems

(0.2”spine)
0.17”<->0.473”
90<->249 pages

SG24-8408-00

ISBN 0738442798

Printed in U.S.A.