Redbooks

ibm.com/redbooks

Software Defined Data Center with Red Hat Cloud and Open Source IT Operations Management

Dino Quintero

Shubham Dhar

Luis Cruz Huertas

Doyoung Im

Afzal Khan

Donthy Venkatesh Krishna Chaitanya

Ramesh Kumar Kumar Singh

Manas Mohsin Kunnathodika

Guru Prasad

Shashi Ranjan

Vishal Vinayak Redij

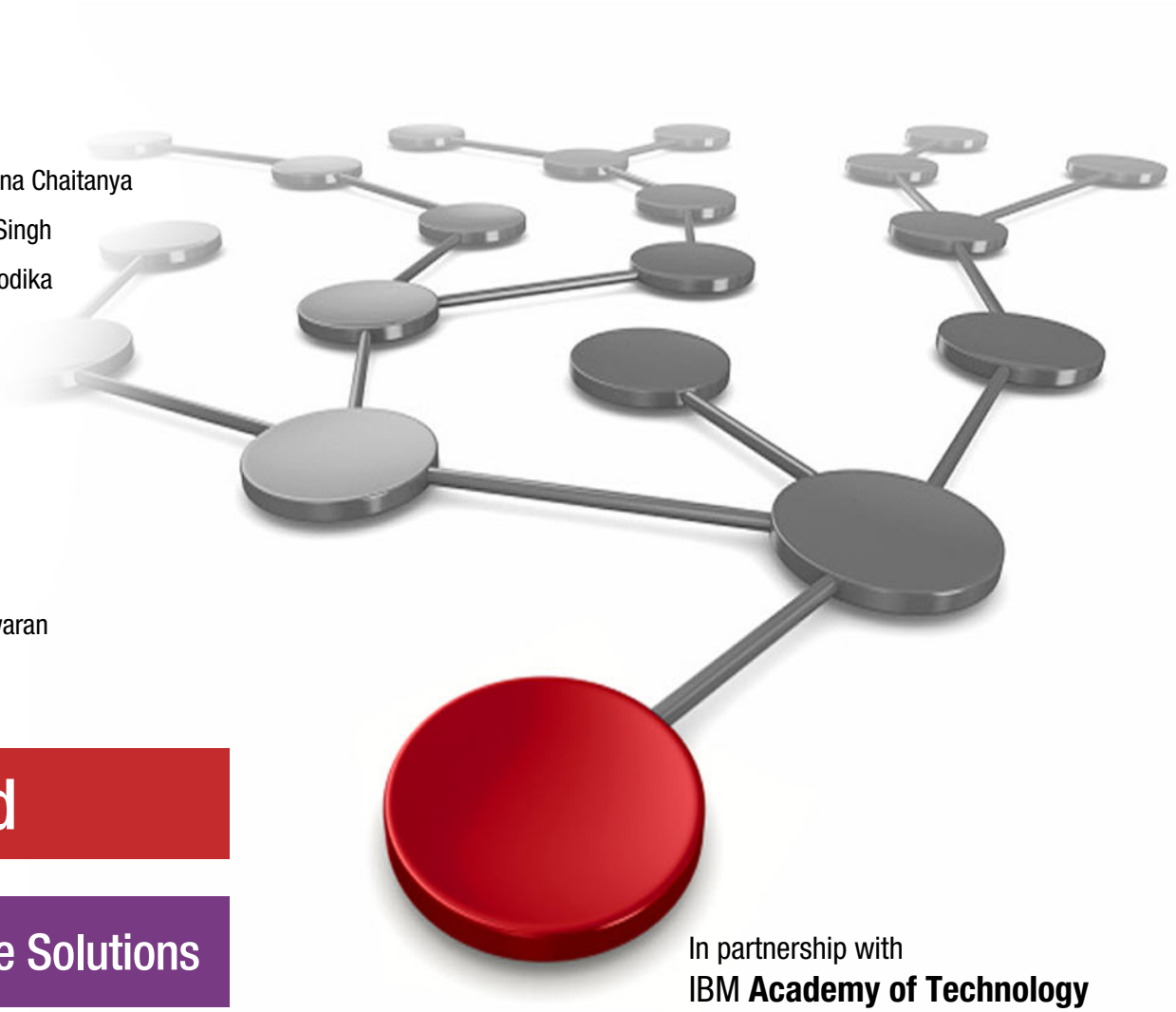
Baldeep Singh

Saurabh Srivastava

Sukrit Thareja

Sreekrishnan Venkiteswaran

Ajit Yadav



Cloud

Infrastructure Solutions

In partnership with
IBM **Academy of Technology**

IBM

Redbooks

IBM Redbooks

Software Defined Data Center with Red Hat Cloud and Open Source IT Operations Management

November 2020

SG24-8473-00

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

First Edition (November 2020)

This edition applies to Version:
Windows Server 2019
Red Hat Enterprise Linux V7.7
IPMI View V2.16.0 (GUI based application)
Freeipmi-1.5 (CLI based application)
CentOS V7
MySQL Cluster V7.6.13
Apache Tomcat V9.0.33
Elasticsearch V7.6.0
Logstash V7.6.0
Kibana V7.6.0
Metricbeat V7.6.0
Red Hat OpenStack Platform V13 or Higher
Red Hat OpenShift Container Platform V4.3
Red Hat Satellite V6.x
Red Hat Ceph Storage V3.3 or Higher
Red Hat CloudForms V5
Red Hat Ansible Tower V3.7.1
Red Hat Enterprise Linux CoreOS
Red Hat Quay V3

Contents

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

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

IBM®	Insight®	Satellite™
IBM Cloud®	Redbooks®	Smarter Cities®
IBM Services®	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

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

ITIL is a Registered Trade Mark of AXELOS Limited.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

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

Ansible, Ceph, CloudForms, OpenShift, Red Hat, are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

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

VMware, and the VMware logo are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

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

Preface

This IBM® Redbooks® publication delivers a Site Reliability Engineering (SRE) solution for cloud workloads that uses Red Hat OpenStack for Infrastructure as a Service (IaaS), Red Hat OpenShift for Platform as a Service (PaaS), and IT operations management that uses open source tools.

Today, customers are no longer living in a world of licensed software. Curiosity increased the demand for investigating the Open Source world for Community Open Source and Enterprise grade applications. IBM as one of the contributors to the Open Source community is interested in helping the software be maintained and supported. Having companies, such as IBM, support the evolution of Open Source software helps to keep the Open Source community striving for enterprise grade open source solutions.

Lately, companies are working on deciphering how to take advantage of Enterprise and Community Open Source to implement in their enterprises. The business case for open source software is no longer a mystery and no surprise that most of the new positions in IT enterprises are related to open source projects.

The ability of a large enterprise to manage this sort of implementations is to engage in a hypertrophied cooperation, where the ability to not only cooperate with teams and people outside your organization, but also to find new ways of working together and devise new ways to improve the software and its code.

A goal for this publication is to help the client's journey into the open source space and implement a private Cloud Container-based architecture with the ability to manage the entire IT Service Management processes from the open source framework.

This publication describes the architecture and implementation details of the solution. Although not every piece of this solution is documented here, this book does provide instructions for what was achieved incorporating open source technologies.

Moreover, with this publication, the team shares their collaboration experiences working in a team of technologists, open source developers, Red Hat, and the open source community.

This publication is for designers, developers, managers, and anyone who is considering starting a Cloud open source project, or users who started that journey. This book also can be a manual to guide the implementation of a technical viable architecture and help those enterprises participate in an open source project but have not done so before.

The reader must be familiar with principles in programming and basic software engineering concepts, such as source code, compilers, and patches.

Authors

This book was produced by a team of specialists from around the world working at IBM Redbooks, Poughkeepsie Center.

Dino Quintero is an IT Management Consultant and an IBM Level 3 Senior Certified IT Specialist with IBM Redbooks in Poughkeepsie, New York. He has 24 years of experience with IBM Power Systems technologies and solutions. Dino shares his technical computing passion and expertise by leading teams developing technical content in the areas of enterprise continuous availability, enterprise systems management, high-performance computing, cloud computing, artificial intelligence (including machine and deep learning), and cognitive solutions. He also is a Certified 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.

Shubham Dhar is a A TOGAF Certified Enterprise Architect experience in delivering multiple high stakes BFSI, telecom, and Power Energy sector's mission critical digital platforms. He specializes in strategically used open source software and methodologies to design large Cloud-Native and Hybrid cloud, big data and Data Lake, mission-critical DC platforms, Distributed Edge platform, and modern application platform. Shubham has more than 17 years of work experience, including Red Hat, Nokia, Dimension Data, Credit Suisse, and BMC Software, which enables customers to realize their strategic digital vision with Red Hat.

Luis Cruz Huertas is an Executive Solutions Enterprise Architect with over 15 years of experience in managing technical and business teams across the globe to deliver strategic projects for high-profile clients across various industries across the world. With extensive expertise in the use of Artificial Intelligence, big data, and Cloud Solutions, he helps to transform the nature of businesses irrespective of the industry. Beginning his career as an IT Manager, he now takes his strong leadership skills and technical knowledge about cloud tech, blockchain and IoT, and cloud to optimize processes for the Banking, Finance, and Telecoms industries. He specializes in systems integration, software architecture, and applications improvement. Luis is also an IBM Master Inventor and holds over four plateaus.

Doyoung Im is a Cloud Native Architect of Architecture Guild, cloud solutioning center east in Singapore, an IBM Level 3 Senior Certified IT Architect, and a member of IBM Academy of Technology. She has more than 20 years of experience with Container with Red Hat OpenShift, public cloud with IBM Cloud® and AWS, Java Platform, Enterprise Edition, IBM Smarter Cities®, Smart TV and solutions. She has been a technical evangelist and lead a first of its kind project with java, public cloud, and open source. She participated as an architect in IBM Cloud projects with Enterprise Virtual Desktop Interface in multi-countries and multi-cloud system monitoring with open source. She also is a Certified Open Group Distinguished IT Architect. Doyoung holds a Master of Business Management and a Bachelor of Science degree in Computer Science.

Afzal Khan is a Senior Solutions Architect - Cloud with Journey to Cloud Red Hat CoE - GTS in New Delhi, India. He has over 10 years of experience as an accomplished Solution Architect and is a Trusted Advisor to his clients. Afzal holds a degree in Masters of Technology degree in Systems Engineering and a Bachelor of Technology degree in Information Technology from BITS, Pilani, and diploma in Computer Science Engineering from AIT, New Delhi. Afzal areas of expertise include Solution Architecture, Designing, Evangelizing, Pre-Sales, Business Development, and Selling in the field of Cloud Solutions, Data Centers, Technology Consulting, Containers, DevOps, IoT, blockchain, and Application Delivery Solutions. He is an accomplished public speaker and represented in many forums, technology summits, and established business for previous employers in various vertical domains, such as IoT, Smart Cities, Federal Government, and Banking. Afzal is a certified cloud professional on various cloud technologies.

Donthy Venkatesh Krishna Chaitanya is a Chief Architect with Journey to Cloud Red Hat CoE - GTS Labs based out of Bangalore, India. He has over 17 years of experience working with Cloud solutions and being a Trusted Advisor to the clients across the globe. Krishna has extensive experience leading technological change within organizations. Krishna holds a degree in Bachelor of Technology degree in Electronics and communications Engineering from NITK, Surathkal. Krishna areas of expertise include Solution Architecture, Designing, developer Evangelist, Pre-Sales, Business Development in the field of Cloud Solutions, Data Centers, Technology Consulting, Containers, DevOps, CICD, SDN, NFV, and Application Delivery Solutions. He is an accomplished public speaker, participates in many technology summits, and authored many publications in various verticals domains, such as Datacenter automation and SDN.

Ramesh Kumar Kumar Singh is an Application Architect with IBM in India. He has 12 years of experience in application designing, development, and automation and four years of experience in the application of machine learning. He holds a Post Graduate Diploma in Data Analysis from IIIT, Bangalore along with Bachelors in Electronics and Telecommunication from Silicon Institute of Technology, Bhubaneswar. His areas of expertise include automation by using python and the application of machine and deep learning models. He has written extensively about the installation and configuration of Grafana and its dashboard to visualize the data that is collected.

Manas Mohsin Kunnathodika is an Enterprise IT Transformation Advisor and TOGAF certified pre-sales Automation Architect, which is part of IBM Global Technology, Innovation & Automation organization (TI&A). In his current role with SIH - East, he is responsible for automation and operations management solutions in more than 10 million TCV deals. Manas has more than 13 years of IT experience in software development, design, architecture, transition, consulting, pre-sales, and delivery. His areas of expertise are Artificial Intelligence for IT Operations (AIOPs) and Cloud Management Platforms (CMP). He is passionate about open source community driven technologies, Agile methodologies, and DevOps/SRE practices. His IT leadership experience at IBM includes Country Service line Manager for Singapore & Malaysia and ASEAN Leader for Hybrid Service Technologies.

Guru Prasad is a Data Engineer in Bangalore, India. He has 10 years of experience in IT operations and Business Analytics field. Guru holds a degree in Electronics and Communication from REVA Institute of Technology & Management, Bangalore and EPGDM-MBA in Business Analytics from IFIM B-School Bangalore. Guru's areas of expertise Business Analytics, Planning, and Strategic skills. Guru is also proficient with ETL tools, such as Logstash and NO SQL database Elasticsearch with hands-on experience with IT operational data, such as logs, metrics, events, and RDBMS tables and incorporating them into Elastic stack. Guru has written extensively on ETL tools, such as Logstash.

Shashi Ranjan is a Technical Specialist/DevOps Engineer with Journey to cloud Red Hat CoE - GTS Lab. Shashi has over 12 years of experience, and is passionate about cloud computing, agile, and various open source technologies. He has worked extensively with AWS and IBM SoftLayer® to design complex network and application architectures that span hundreds of hosts and specializes in Network security, infrastructure design, implementation, and deployed multiple products into production successfully.

Vishal Vinayak Redij is a Database Administrator having over six years of experience in managing MySQL and MS SQL databases. Based in Mumbai, India he has a Bachelor's Degree in Information Technology from University of Mumbai. His area of expertise includes implementation and maintaining MySQL and MS SQL Servers. An avid Pythonista, he has used his python skills to automate and optimize various tasks at his current project. He is a Data Science Enthusiast with keen interest in implementing it in day-to-day scenarios.

Baldeep Singh is a Systems Management Specialist and a Subject Matter Expert of UNIX, Linux Platforms in India. He has over 17 years of extensive experience in managing IT Enterprise systems with excellent Team management skills. Baldeep demonstrated his IT skills in successfully managing heterogeneous projects and working environments of Banking, Telecom, Manufacturing, and Automobile sectors. He is a Business graduate with a diploma in Computer Science and Engineering. He is professionally certified in Red Hat, HP-UX, Cisco, and Microsoft technologies. He also is ITILv3 Foundation certified and a Six Sigma Blackbelt Certified professional. He has written extensively about the installation, configuration, and troubleshooting procedures of Business critical servers, Enterprise Operating Systems, and Middleware.

Saurabh Srivastava is a specialist solution architect, working with Red Hat. With over 13 years of experience in the IT industry, he specializes in SDN, IaaS, Cloud, Hyper-converged Infrastructure, Software-defined data center, Red Hat OpenStack, Virtualization, Data Center Infrastructure, Networking, and Linux. Saurabh has extensive experience in the Telecom and Networking domain and has worked for Cisco, Ericsson, Nokia in 4G, 3G, and other telecom technologies. Saurabh holds an engineering degree in Computer science. He also has interests in embedded device drivers and Linux. An open source evangelist for life, he enthusiastically talks about Open source software in webinars and public events.

Sukrit Thareja is a TOGAF 9.2 certified technical solution architect, working with Red Hat. With over 12 years of experience in the IT industry, he specializes in PaaS, AppDev, and Middleware technologies. Sukrit is an expert in Red Hat OpenShift and Micro-services Architecture. He has also worked extensively in designing solutions around Application Integration, service-oriented architecture, API Management, Application Development (both web and mobile), Business Rules Management, and Business Process Management. Sukrit has been involved in architecting some of the major digital transformation programs, closely working with some of the biggest airlines, BFSI, government ministries, and telcos in India. Sukrit holds MBA degree from NMIMS University, Mumbai, and Engineering degree from ITM Gurgaon. He likes to evangelize his technical knowledge and has been a speaker in various public forums and technical conferences. Whenever possible, he also contributes to the community by publishing blogs and videos.

Sreekrishnan Venkiteswaran is an IBM Distinguished Engineer and CTO of the Cloud Center of Excellence, which is a worldwide team of client-facing cloud solution architects. Krishnan is an IBM Master Inventor and the General Chair of the IEEE International Conference on Cloud Computing for Emerging Markets (2017 - 2020). He is the author of a popular book on Linux internals that is cited as a reference inside the mainline Linux Kernel source tree. Krishnan has been with IBM for 24 years and holds a Master in Computer Science from the Indian Institute of Technology, Kanpur, India.

Ajit Yadav is a Tools and Automation specialist. He has 13 years of experience with automation (dynamic automation, robotic process automation, cognitive solution for auto-assignment, local automation, and using python/shell and Red Hat Ansible) and EMS tools (monitoring, fault, ticketing, analytics, and others). He is also certified in Kubernetes/containers, Blueprism, Automation Anywhere, CEH, ITIL, and IVCA. Ajit holds a Bachelor of Engineering degree, PGDAC in Computers from CDAC. He contributed extensively on EMS tools, components containerization, configuration integration, and automation.

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

Wade Wallace
IBM Redbooks, Austin Center

Prasad Mukhedkar
Red Hat, India

Miguel Gomez Gonzalez
IBM Guadalajara, Mexico

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 IBM Redbooks 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, IBM Redbooks
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

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

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

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

- Stay current on recent Redbooks publications with RSS Feeds:

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

1

Introduction

This chapter provides the overall introduction of the publication, including the reasoning behind its development, and a description of the Red Hat Cloud solution.

This chapter includes the following topics:

1.1 Introduction

We work in IBM Services® and Red Hat, and over the years we stumbled across many teams that are under the same crossroad situation in infrastructure management. The goal is to invest in system stability and make the infrastructure flexible to the businesses to accelerate development and growth.

Today, organizations run cloud services, on premises infrastructure, and use applications as services, but they cannot truly manage it or control it 100%. In this book, we provide guidance about one of many alternatives that you can use to set up a hybrid cloud that can run on your on-premises data center or in a public cloud that can provide bare metal services.

In this cloud, we manage all of the cloud elements and all ITSM processes by using community edition open source software, except the cloud components that are also open source but enterprise grade.

1.2 Why develop this publication

Server automation, containers, virtualization, software defined storage, software defined network, IT Service Management, and IT operations are many of the IT elements that organizations are adopting today. This huge set of technologies have companies in a race of an unmanageable system operation. It is in this area that IBM has a recommended approach across all enterprises to simplify the complexity of IT Systems and provide management across systems.

This publication is ideal for system administrators, IT architects, enterprise architects, and cloud subject matter experts. It demonstrates various open source tools, enterprise open sources that are based on Red Hat cloud, along with various techniques, patterns, reports, and integrations that can be used to set up a full infrastructure as code cloud management operation.

1.3 Red Hat Cloud on bare metal managed by open systems

As key contributors to the open source community, IBM and Red Hat are on a mission to provide enterprises with the ability to run IT systems on open source. At the same time, never forgetting the level of governance and security that is needed to run large-scale applications and data across technologies.

Red Hat and open source code are in a unique position to allow enterprises to achieve all of these goals. It can reduce the level of license dependencies in large-scale enterprise software and avoid losing management systems functions. By using this architecture, you can perform the following tasks:

- Establish infrastructure as code principles
- Set up a dynamic infrastructure platform
- Establish the correct definition of infrastructure tools
- Provide infrastructure services
- Create a generation of patterns and server provisioning
- Create templates for server management.
- Provide server updates and pattern changes to the servers
- Create patterns that define the infrastructure
- Move to a software engineering practice for Infrastructure

- Test Infrastructure changes
- Move to a change management pipeline infrastructure
- Run on new workflow infrastructure teams.
- Run with a continuity and dynamic infrastructure
- Set up your organization to run your extreme automation and infrastructure as code

The Red Hat and IBM approach is essential to manage cloud infrastructure of large-scale complexity. This is not exclusive for enterprises on cloud; it crosses the boundaries between public and private and establishes a Hybrid-Multi cloud environment and IT operations.

This publication demonstrates that the combination of Red Hat and open source tools deliver several characteristics of CAMS: culture, automation, measurement, and sharing to provide IT cloud and IT services.

1.3.1 Relevance of open source community platform

Open source software is software that makes the available source code for any developer in the world to inspect, change, transform, and enhance that software.

By having access to the source code, developers can change, optimize, transform, and adjust how software is used or works. Developers also can add features and repair elements that do not always work correctly.

By leveraging open source community edition software, developers also can adjust IT service operations to optimize the IT Service Management workflows and fit them to the enterprise culture and needs.

1.3.2 Selection of open source products

In this publication, there was a robust open source community edition evaluation of products, that evaluated features and ease of deployments, along with community adoption.

The following selection is not an IBM evaluation; rather, it is an evaluation that was done by team members who produced this publication. It is not intended to be used as a reference for global tools analytics, but rather a perspective from subject matter experts and what the book intention is trying to achieve as a result. The set of functional and non-functional requirements are a result of what the book encompasses.

The product selection is distributed across the following major components:

- Systems management and data collections
- CI/CD automation and data lake analytics
- Visualization and user interface, dashboards, and insights

To correctly map the tools' characteristics and address the need for each tool, we noticed that several open source tools met the criteria of many of the elements in the IT Service Management processes; however, we did not find a single tool that met the end-to-end criteria, such as a suite of tools that can achieve the entire ITSM processes as did the licensed enterprise ITSM tools.

To meet such a task, there was a selection based on key functions and its ability to seamlessly integrate with each other. The product mapping generated a result that maps to each ITSM element and meets the full end-to-end lifecycle of DevOps and ITSM processes. Figure 1-1 shows the tools grouped by three components and its function, along with the tool name that addresses that functions.

Systems Management & Data Collection	Authentication & Authorisation	Gluu
	Infrastructure Monitoring	Prometheus
	Log monitoring	Logstash
	Asset/CI Discovery	Open-AudIT
	VMs, Private cloud Management	CloudForms
	Patch Management	Satellite
	Stream processing & Integration	Kafka
CI/CD, Automation, Data Lake & Analytics	Time Series Data Lake	InfluxDB
	Event Management	SEC
	Data Analytics	Elasticsearch
	Data Processing	Python, Spark
	Automation	Ansible
	CI/CD Integration	Jenkins
	Version control code repository	Git
Visualization- User Interface, Dashboards, Insights	Monitoring & SLA Dashboard	Grafana
	Log Management Dashboard	Kibana
	Asset Management	Snipe-IT
	IT Service Management	iTop
	Agile Workflow Management	Jira
	VMs, Private cloud Management Portal	CloudForms Portal
	Patch Management Analysis Dashboard	Satellite Dashboard
	Unified Management Dashboard	Liferay

Figure 1-1 Tools components and functions

This tool maps to functions; however, it is also important to represent these elements in such a way that they map to specific ITSM processes. These processes (as shown in Figure 1-2) are the process that were addressed by this set of selected tools.

Figure 1-2 ITSM processes

Now that the open source product mapping exists, the set of processes in ITSM is grouped to meet the ITSM process that flows into a spiderweb diagram (see Figure 1-3) to assure that the criteria is met end-to-end in the service delivery cycle.

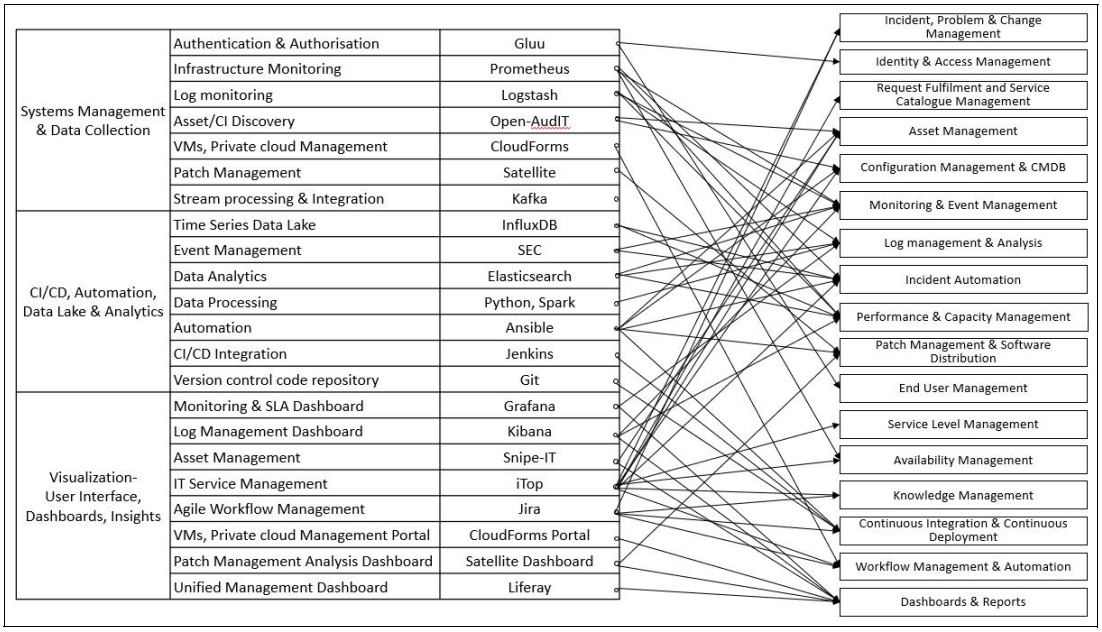


Figure 1-3 ITSM processes: Tools and functions

Multiple tools are integrated by using well-known and easily reproducible integration methods to meet technical requirements of each functional stream. Some of the integration methods are listed in Table 1-1 as a reference.

Table 1-1 Criteria set

Source	Destination	Integration methods	Remarks	Reference
Infrastructure	Gluu	N/A	N/A	N/A
Infrastructure	Prometheus	SNMP, Node Exporters	N/A	https://github.com/prometheus/prometheus/wiki/Default-port-allocations
Infrastructure	Logstash	file, syslog, redis, filebeat	N/A	https://www.elastic.co/guide/en/logstash/current/pipeline.html
Infrastructure	Open-AudIT	nmap	N/A	https://www.open-audit.org/
Infrastructure	Red Hat CloudForms	N/A	N/A	https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/8/html/quickstart_guide_for_cloudforms_with_red_hat_openstack_platform/introduction_to_red_hat_cloudforms
Infrastructure	Red Hat Satellite™	N/A	N/A	https://www.redhat.com/en/technologies/management/satellite

Source	Destination	Integration methods	Remarks	Reference
Infrastructure	Red Hat Ansible	N/A	N/A	https://go.redhat.com/delivery-with-ansible-20181012?sc_cid=701f2000000RlzzAAC&gclid=CjwKCAjwh472BRAGEiwAvHVfGnN_OFTJFfGOrvXkWOCHZ-OouvrrLSlQK6-5XZt9w8QNAldLfXRQNxoCREoQAvD_BwE&gclsrc=aw.ds
Red Hat Ansible	Infrastructure	N/A	N/A	https://www.ansible.com/
Prometheus	InfluxDB	Remote Read/Write APIs, Prometheus Node Exporter for InfluxDB, Kafka	For long-term storage of alerts	https://docs.influxdata.com/influxdb/v1.7/supported_protocols/prometheus/ https://github.com/prometheus/influxdb_exporter
Prometheus	Grafana	Node Exporter for Grafana	Few metrics to be directly displayed besides going into InfluxDB	https://github.com/frodenas/grafana_exporter https://grafana.com/docs/grafana/latest/features/datasources/prometheus/
Logstash	Elasticsearch, Kibana	Java API, RESTful HTTP/JSON API	N/A	N/A
Elasticsearch	Kibana	Java API, RESTful HTTP/JSON API	N/A	N/A
Elasticsearch	Grafana	Java API, RESTful HTTP/JSON API	N/A	https://grafana.com/docs/grafana/latest/features/datasources/elasticsearch/
Red Hat Ansible	InfluxDB	Built-in HTTP API, JSON over UDP	N/A	N/A
Red Hat Ansible	IT Operational Portal (iTop)	REST/JSON Services	N/A	https://www.itophub.io/wiki/page?id=2_5_0%3Aadvancedtopics%3Astart
Red Hat Ansible	Snipe-IT	JSON REST API	N/A	N/A
Red Hat Ansible	JIRA	Ansible plug-in in Jenkins or as per Red Hat Ansible documentation	N/A	https://plugins.jenkins.io/ansible/ https://www.redhat.com/en/blog/integrating-ansible-jenkins-cicd-process
Red Hat Ansible	Red Hat Satellite	Red Hat internal	N/A	https://www.redhat.com/en/blog/getting-started-ansible-satellite

Source	Destination	Integration methods	Remarks	Reference
Red Hat Satellite	Red Hat Ansible	Red Hat internal	N/A	https://www.redhat.com/en/blog/getting-started-ansible-satellite
Red Hat Satellite	InfluxDB	Kafka	N/A	N/A
Red Hat CloudForms	InfluxDB	Kafka	N/A	N/A
Red Hat CloudForms	iTop	Native CloudForms Technology,Ruby OR through Ansible automation OR REST/JSON API	CMDB Data needs to be regularly updated	https://cloudformsblog.redhat.com/2017/11/16/cmdb-integration-architecture-examples-for-cloudforms/ https://www.itophub.io/wiki/page?id=2_5_0%3Aadvancedtopics%3Astart
Red Hat CloudForms	Snipe-IT	Native CloudForms Technology,Ruby OR through Ansible automation	Asset Data needs to be regularly updated	https://cloudformsblog.redhat.com/2017/11/16/cmdb-integration-architecture-examples-for-cloudforms/
InfluxDB	Grafana	Built-in HTTP API, JSON over UDP	N/A	https://grafana.com/docs/grafana/latest/features/datasources/influxdb/
InfluxDB	iTop	Built-in HTTP API, JSON over UDP	For auto-ticket creation based on qualified incident	https://www.itophub.io/wiki/page?id=2_5_0%3Aadvancedtopics%3Astart
InfluxDB	SEC	Built-in HTTP API, JSON over UDP	For event correlation, enrichment, and others.	https://db-engines.com/en/system/Elasticsearch%3BInfluxDB
InfluxDB	Elasticsearch	Built-in HTTP API, JSON over UDP	N/A	N/A
InfluxDB	Red Hat Ansible	Built-in HTTP API, JSON over UDP	Automation Kick-off	N/A
OpenAudIT	Snipe-IT	JSON REST API	Reconfirm if JSON API is part of Open-AudIT Community Edition (https://www.open-audit.org/about.php)	https://community.opmantek.com/display/OA/The+Open-AudIT+API

Source	Destination	Integration methods	Remarks	Reference
OpenAudIT	iTop	REST/JSON Services	Reconfirm if JSON API is part of Open-AudIT Community Edition (https://www.open-audit.org/about.php)	https://community.opmantek.com/display/OA/The+Open-AudIT+API https://www.itophub.io/wiki/page?id=2_5_0%3Aadvancedtopics%3Astart
Jenkins	Red Hat Ansible	Ansible plug-in in Jenkins or per Red Hat Ansible documentation	N/A	https://plugins.jenkins.io/ansible/ https://www.redhat.com/en/blog/integrating-ansible-jenkins-cicd-process
Jenkins	JIRA	JIRA has Jenkins plug-in, some other options like apwide available too	N/A	https://plugins.jenkins.io/jira/ https://www.apwide.com/how-to-integrate-jira-with-jenkins-successfully/
Jenkins	GIT	Jenkins Git plug-in	Use the Jenkins Git plug-in to pull and perform build steps with source code from GitHub.	https://www.theserverside.com/tutorial/Attain-Jenkins-Git-integration-with-a-GitHub-pull-request
JIRA	Jenkins	JIRA has Jenkins plug-in, some other options like apwide available too	N/A	https://plugins.jenkins.io/jira/ https://www.apwide.com/how-to-integrate-jira-with-jenkins-successfully/
JIRA	Red Hat Ansible	Ansible plug-in in Jenkins or per Red Hat Ansible documentation	N/A	https://plugins.jenkins.io/ansible/ https://www.redhat.com/en/blog/integrating-ansible-jenkins-cicd-process
JIRA	Red Hat CloudForms	N/A	N/A	N/A
JIRA	iTop	REST/JSON Services	N/A	https://www.itophub.io/wiki/page?id=2_5_0%3Aadvancedtopics%3Astart
GIT	Satellite	N/A	N/A	N/A
GIT	Jenkins	Jenkins Git plug-in	Use the Jenkins Git plug-in to pull and perform build steps with source code from GitHub.	https://www.theserverside.com/tutorial/Attain-Jenkins-Git-integration-with-a-GitHub-pull-request

Source	Destination	Integration methods	Remarks	Reference
Liferay	Single sign-on	N/A	Customer's existing Single-sign On integration	https://blogs.perficient.com/2012/10/08/liferay-and-single-sign-on-sso-whats-here-and-whats-coming/
Liferay	Active Directory	LDAP support, Kerberos for authentication	N/A	https://portal.liferay.dev/docs/7-1/deploy/-/knowledge_base/d/authenticating-with-kerberos
Liferay	Portlets and other methods	N/A	Front-end customization	https://portal.liferay.dev/

The list of tools that are selected supply a large set of supported integrations. These integrations can easily be supported by any IT infrastructure support team that is part of the critical elements when selecting open source products along with its lifespan in the market.

In the selection of open source tools, we must eliminate several key beliefs. The focus is on lifelong open source projects that incubated as a project from a large university or enterprise company can be traced to the amount of code commits and code consumption. This generates a level of assurance that the product remains in the market for years.

We used a level set of scholar citations to relative search and research papers and studies to conduct the product and criteria selection. The following factors were considered:

- Ease of installation and use
- Are there issues or use problem reports raised. If so, is the owner taking care of them or was the project is abandoned?
- Does it list hardware requirements?
- Continues integration status.
- Does it have a license version?
- Does it have tests?
- Does it have a Dockerfile?
- Does it have badges?

Ultimately, the selection of a software solution comes down to the requirements of the project (time line, budget, and other critical items). The level of risk tolerance also is key because many items are at the edge of technological innovation (risk tolerance), and how capable are team members to master these solutions based on their skill levels (internal skill set). Then, the solutions are tested before fully committing.

1.4 About this publication

We are living in an unlimited provided computing environment with Cloud platforms. This publication was developed during by using IBM Cloud bare metal servers and networks to have a similar environment as a private cloud. We developed this publication with authors scattered across the world thanks to cloud capacity and capability with more than 60 servers at maximum.

2

Reference architecture

This chapter describes the reference architectures that were used for the solution implementation.

This chapter includes the following topics:

2.1 Functional architecture motivation

This reference architecture is created for those enterprises that seek a digital transformation and at the same time reduce the high cost in licensed software and avoid capital expenditure. The architecture that is described in this chapter contains a combination of Enterprise open source and community edition open source software.

The reference architecture is oriented towards private cloud implementations on-premises or run a unique possibility of a private cloud on top of a public cloud bare metal systems. One of the other unique architectural differentiations is the ability to run 100% IT service management (ITSM) and Information Technology Infrastructure Library (ITIL) processes with a combination of open source software community edition and sustain an enterprise grade performance.

As a result, this reference and demonstrated architecture serves the purpose for any enterprise that wants to move application workloads to containers and have an entire enterprise infrastructure management system running on containers. This configuration allows more flexibility and portability of the applications.

The high-level solution overview is shown in Figure 2-1. The IaaS layer is segmented into a managed zone and a managing zone.

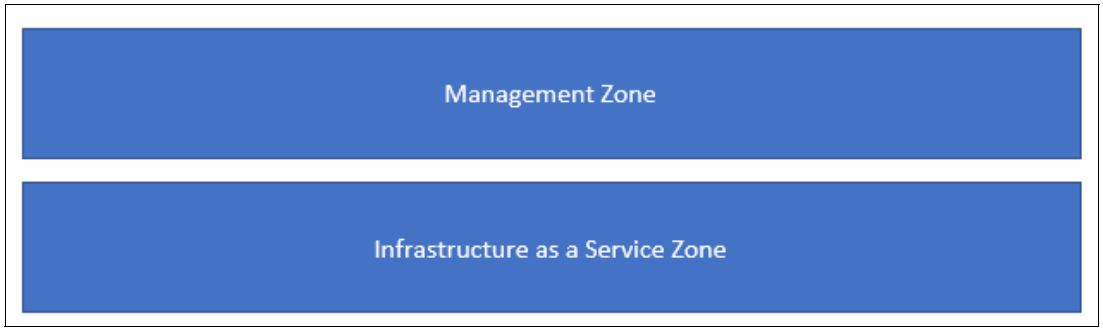


Figure 2-1 High-level solution overview

The defined private cloud is virtualized and cloud-enabled by using Red Hat OpenStack on IBM bare metal servers in IBM Cloud. A container platform that is based on Red Hat OpenShift is hosted on a portion of the Red Hat OpenStack. Servers and blade servers are under the management of Red Hat OpenStack.

Software-defined storage is realized through Ceph. The managing zone holds the Red Hat OpenStack control plane and the tools that are used to realize the design of the overall solution. We use Red Hat OpenStack Platform and Red Hat OpenShift Container Platform to implement Red Hat OpenStack and Containers, respectively.

The automation engine is based on Red Hat Ansible. A single provisioning pane is proposed around Liferay Portal, an open source portal aggregator. Individual dashboards and portlets are framed inside of Liferay, and single sign-on is implemented by using the open source tool, Gluu. The service management box shows ITSM tooling (iTop), backup (native MySQL backup), disaster recovery, capacity planning, and migration services. An end-to-end security is built that touches all facets of the solution. Individual parts of the solution are described in Chapter 3, “Building Red Hat hybrid cloud” on page 31 and Chapter 4, “Open source IT operations management” on page 221.

The managing zone holds the Red Hat OpenStack control plane on bare metal rack servers and has a virtualized portion (through Red Hat virtualization) to deploy the virtual machines (VMs) that are needed to operate the environment. IT operational analytics is designed around the ELK stack with various data sources in the solution that is ingested through Logstash and Kafka messaging hub. End-to-end monitoring is achieved through Prometheus and visualized through Grafana with enhanced analytics that are run in Elasticsearch with Kibana as a visualization pane.

Network virtualization is achieved by way of the IBM Cloud network, with the overlay integrated with the cloud orchestrator. Figure 2-2 shows the Infrastructure Service Zone.

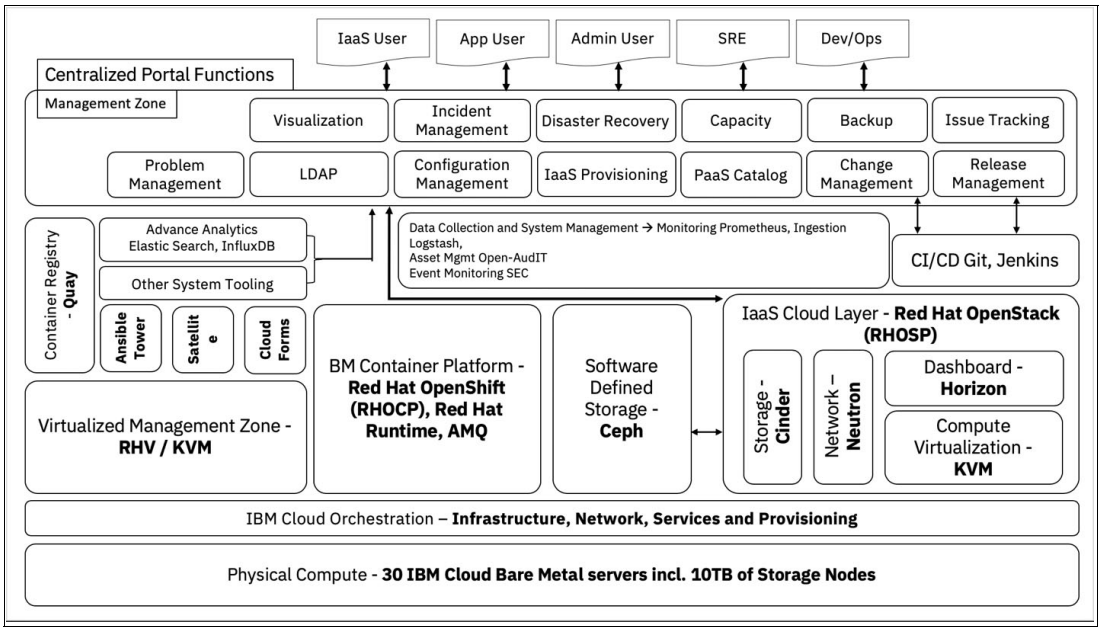


Figure 2-2 Overall functional architecture

The infrastructure management zone that is described in this publication runs on RHV/KVM, provisioning bare metal servers, for Red Hat OpenStack, and Red Hat OpenShift functions. Among the diversity of roles directors, controllers, and compute nodes are featured.

2.2 Overall solution architecture

The architecture uses only IBM Public Cloud services for bare metal server requests that are used as Infrastructure as a service resource (see Figure 2-3).

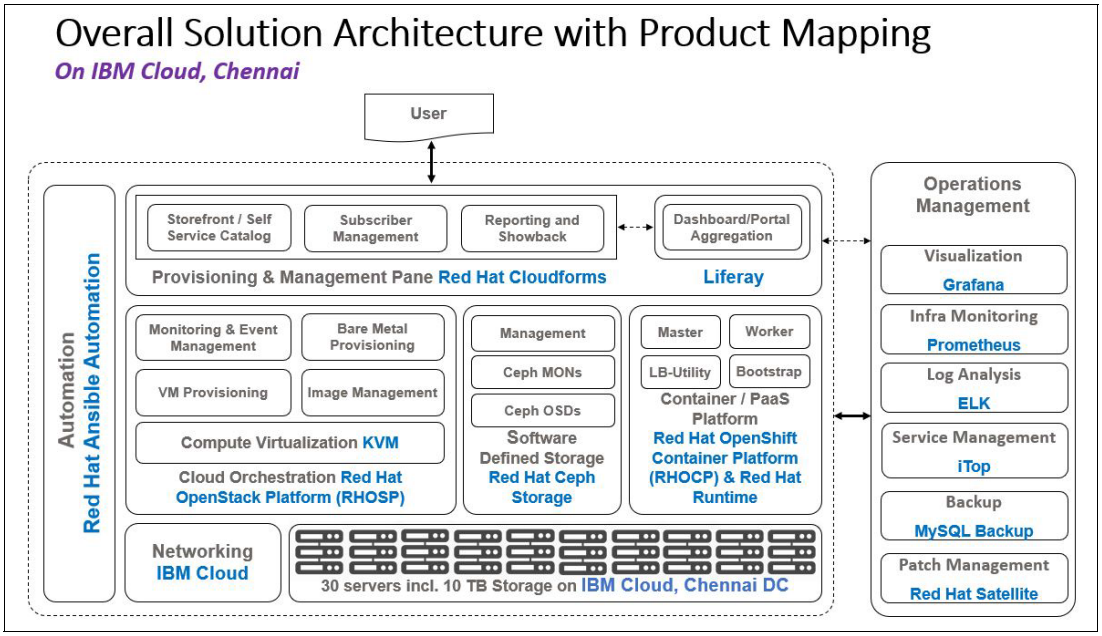


Figure 2-3 Overall solution architecture

In the reference architecture, our approach is to provide the following set of core functions as part of the architecture:

- Platform management
- Asset services
- Storage and data
- Runtime management
- Compute and run time
- Connectivity
- Security
- Cloud management

The architecture uses the benefit of public infrastructure provisioning, such as bare metal servers and Layer 2 network connectivity to overlay software-defined network and storage. The architecture that is shown in Figure 2-3 shows two runtime engines on top of bare metal (Red Hat OpenStack and Red Hat OpenShift a traditional and a container platform) with a common Red Hat at Ceph storage for common consumption.

In parallel to the cloud, a service management platform and systems management is available for monitoring visualization, service management, advanced analytics, backup, security, and provisioning. All tools are commonly integrated in a portal that self-contains and integrates all tools as a single management console.

This architecture covers the following cloud processes:

- Storefront
- Self-service catalog
- VM provisioning
- Storage provisioning

- Tenant network provisioning
- Multi-tenancy as projects
- Bare metal provisioning
- Container provisioning
- Application provisioning
- Show back reporting
- Subscriber management

2.3 Cloud reference architecture

In our architecture that is shown in Figure 2-4, all Red Hat Cloud solution elements are deployed. In the architecture, the creation and deployment of Red Hat OpenStack on top of bare metal servers is key to define and allow Red Hat OpenStack to control services, such as Nova, Cinder, Neutron, and Glance.

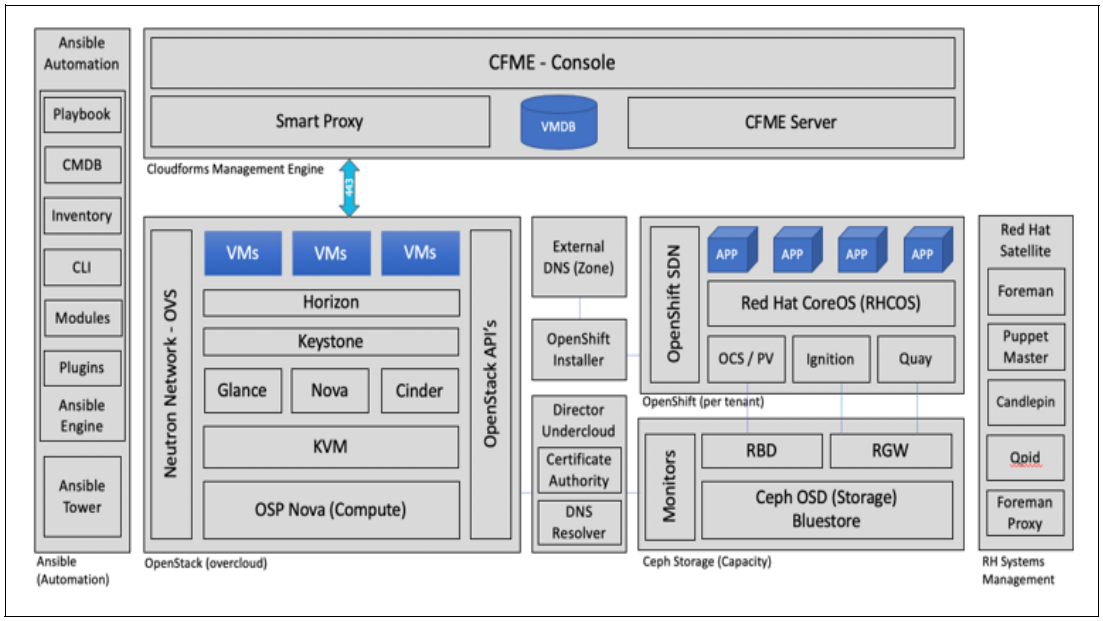


Figure 2-4 Cloud reference architecture

The Horizon user interface allows users to graphically manage the Red Hat OpenStack elements and obtain critical reporting. By using the Red Hat CloudForms as the management engine, auto provisioning of complex applications can make calls to Red Hat OpenStack to provision a fully automated VM with an Isolated tenancy design.

Ceph storage is a common pool that is shared across Red Hat OpenStack and Red Hat OpenShift, with the use of Quay as a container private registry. It also provides the capability to scan the container images and inspect them for the security of the containers when published.

As parallel deployments, Red Hat Ansible Tower and Red Hat Satellite are key for systems management and automation. With Red Hat Satellite, you can perform the complex job scheduling and provisioning of systems within built-in templates for provisioning objects and resources that are under management. It also helps in subscription management, vulnerability management, and patch management of the subscribed systems. Red Hat Ansible Tower is used as a specific solution for performing complex digital automations.

2.4 IaaS Cloud Reference Architecture - Red Hat OpenStack

The IBM Cloud bare metal is all Layer 2 interconnected and provides an effective solution for integration with the private cloud environment. Red Hat OpenStack Neutron is based on a pluggable architecture. Neutron plug-ins provide programmable interfacing that is necessary for Layer 2 and IP address management for IBM Cloud integration with the private cloud.

The IBM Cloud - Red Hat OpenStack can be deployed in two different configurations; however, in this architecture, we are running under cloud and overcloud deployment to manage the bare metals all the way to the provisioning layer. This deployment option implements a virtual machine manager (VMM) on APIC to provide the fabric administrator maximum visibility of the Red Hat OpenStack cloud.

As IBM Cloud programs, the physical fabric treats Red Hat OpenStack tenant traffic as part of physical domains. The IBM Cloud network allows the creation of networking constructs that can be passed to be driven from Red Hat OpenStack by using standard Neutron calls.

We recommend deploying the IBM Cloud network as default and adjusting the Red Hat OpenStack deploy scripts to IBM Cloud. This deployment provides advantages over non-controlled IBM Cloud configurations. To do so, it maps Neutron concepts, such as networks, subnets, and routers, to IBM Cloud concepts, such as EPGs, bridge domains, and contracts.

The AIM is installed on the Red Hat OpenStack controller nodes and is responsible for configuring IBM Cloud through a REST API call that is based on the defined Red Hat OpenStack policy model. The ML2 driver interacts with the AIM daemon, which communicates with the controller to configure policies.

The Red Hat OpenStack tenant administrator configures the Red Hat OpenStack networks through standard Neutron calls by using CLI, Heat, Horizon, or REST API calls. The ML2 driver translates the networks that were created into AIM policies. AIM stores the new configuration into its database and pushes network profiles into APIC through REST API calls.

The APIC creates related network profiles, for example, ACI tenants, bridge domains, EPGs, and contracts. The Red Hat OpenStack administrator creates instances and attaches those to the previously created networks. The APIC is notified that new instances were created. Then, the APIC pushes related policies to leaf switches.

Traditionally in Red Hat OpenStack, routing is performed only on servers that are hosting neutron services; with IBM Cloud integration, this is taken care of by routing VMs. Although each compute node has an agent, routing is performed in a distributed manner. Also, the agent performs local policy enforcement through Open Virtual Switch (OVS) rules locally on the same hypervisor where the instance lives (see Figure 2-5).

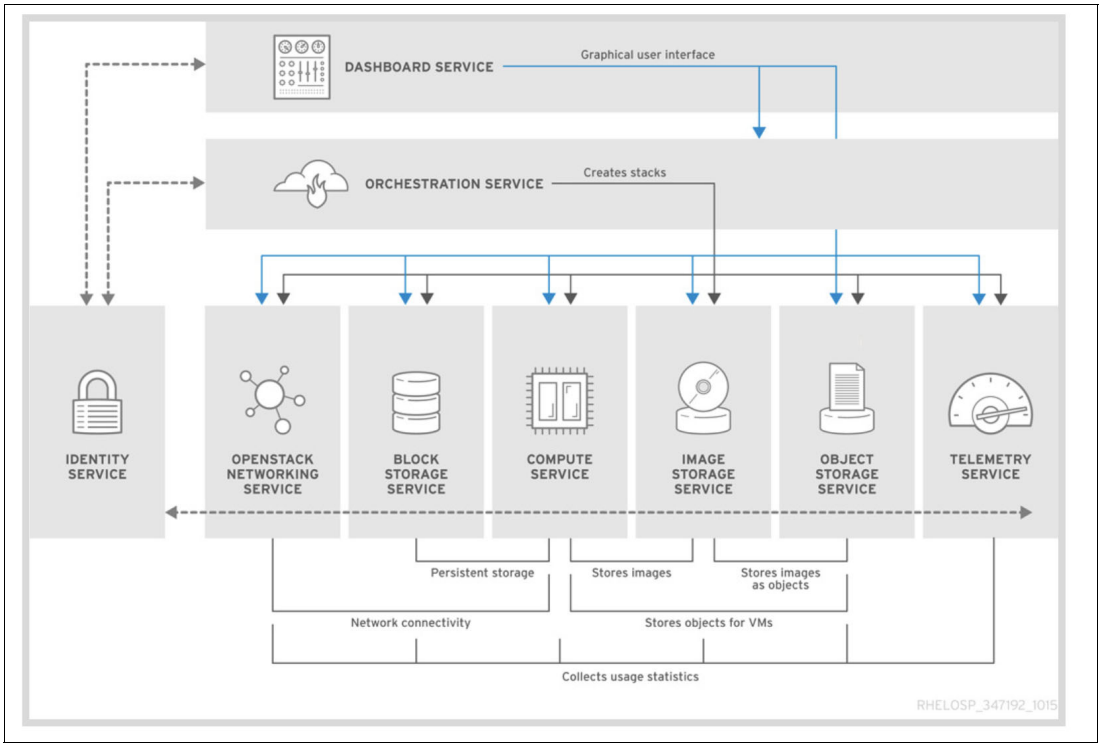


Figure 2-5 High-level Red Hat OpenStack system architecture

The managing zone services are segmented as non-virtualized part that holds the Red Hat OpenStack control plane, and virtualized portion (through Red Hat virtualization) to deploy the tools that are needed to operate the environment.

IT operational analytics is designed around the ELK stack with various data sources in the solution ingested through Logstash. End-to-end monitoring is achieved through Prometheus and visualized through Grafana. The provisioning pane also includes the AI and ML tools. Network virtualization is achieved by way of the Neutron Cloud stack, with the overlay integrated with the cloud orchestrator, which is Red Hat OpenStack.

The proposed solution is composed mostly of open source components. The solution contains two major categories of open source: Cloud and SMS. For these two categories, IBM selected specific tools. The following principles governed our open source support strategy for this solution:

- System software elements rely on a distribution-backed support model because of the need to deeply understand complex code and the sheer magnitude of test infrastructure that is required to qualify new releases and fixes.
- Wherever possible, community hub backed support models are used.
- For the remainder of the solution, we take advantage of custom-supported, community-backed open source projects.
- When none of the previous models are possible, non-open source software is used. We endeavored to keep this category to the bare minimum.

The software elements in the solution are listed in Table 2-1.

Table 2-1 Software elements in the proposed solution: Production setup

Solution requirement	Software element	Open source?	Enterprise or community
Operating system: Linux	Red Hat Enterprise Linux	Yes	Enterprise
Private Cloud Stack	Red Hat OpenStack	Yes	Enterprise
Container Platform (Kubernetes)	Red Hat OpenShift	Yes	Enterprise
SDS Platform	Red Hat Ceph	Yes	Enterprise
Automation	Red Hat Ansible Automation with Tower	Yes	Enterprise
IaaS provisioning pane	Red Hat CloudForms	Yes	Enterprise
Virtualization for management zone	Red Hat Virtualization	Yes	Enterprise
Patch management solution	Red Hat Satellite	Yes	Enterprise
Container Registry tool	Red Hat Quay with Clair Scanner	Yes	Enterprise
Operating system clustering (non-Cloud)	Red Hat HA	Yes	Enterprise
Monitoring solution	Prometheus	Yes	Community
Visualization pane for monitoring	Grafana	Yes	Community
Time series database for EMS/Analytics	InfluxDB	Yes	Community
ITSM tool	iTop	Yes	Community
CI/CD	JIRA	Yes	Community
Asset Management	Open-AudIT	Yes	Community
Single Sign-On tool	Gluu	Yes	Community
Event management tool	SEC	Yes	Community
PaaS Catalog - Load balancer	HAproxy (Red Hat OpenShift)	Yes	Enterprise
PaaS Catalog - Monitoring	Nginx (Red Hat OpenShift)	Yes	Enterprise
PaaS Catalog - Runtime	Springboot (Red Hat Runtime)	Yes	Enterprise
PaaS Catalogue- In-memory cache	Redis & Memcached (Red Hat OpenShift)	Yes	Enterprise
PaaS Catalog - Database	PostgreSQL (Red Hat OpenShift)	Yes	Enterprise

Solution requirement	Software element	Open source?	Enterprise or community
PaaS Catalog - Webserver	Tomcat (Red Hat Runtime)	Yes	Enterprise
Portal Aggregator	Liferay	Yes	Community
PaaS Catalog - CI/CD Toolchain	Git	Yes	Community
PaaS Catalog - Datastores, file systems, Analytics	Hadoop, HDFS, Hive, Pig, Solr, Flink, Pentaho, MapR	Yes	Community
Log analytics tool	ELK Stack	Yes	Community
PaaS Catalog - Database	MySQL	Yes	Community

2.5 Unified consumption pane across portals and dashboards

A unified dashboard with single sign-on access eases consumption of IT services and simplifies continued operations. Most dashboards and portals that are used by the services platform are “framed” inside a content management framework (the open source Liferay). Liferay is integrated to a directory server (Lightweight Directory Access Protocol [LDAP]) by way of the open source single sign-on tool (Gluu) to provide simplified authentication.

Portals of the cloud management platform (Red Hat CloudForms), the container platform (Red Hat OpenShift), the ITSM tool (iTop), log analytics visualization (Kibana), monitoring portal (Grafana), and DevOps tracking tool (Jira) are aggregated inside Liferay. Specific specialized dashboards (such as SDN controller and automation engine configuration) are outside of the ambit of portal aggregation for efficiency and user experience reasons.

2.6 Management zone

We designed a management zone to host all of the managing components of the solution. The isolated and segregated architecture of this managing environment is part of the overall architecture overview that is shown in Figure 2-3. The management zone sizing details are listed in Table 2-2.

Table 2-2 Management zone sizing

#	Component	Details	Hosting	(Servers)
1	Red Hat OpenStack Control Plane	Three controller nodes	Bare metal	12
		One director node per region (for two regions)	RHV	

#	Component	Details	Hosting	(Servers)
2	System Management Components	► Red Hat CloudForms ► Red Hat Ansible Tower ► Red Hat Satellite (Active - Active) ► Prometheus ► Kibana (on both sites, accessed through an LB) ► Grafana ► iTop ► SnipeIT ► Open-AudIT (Active - Passive) ► Liferay (Active - Active) ► SEC ► InfluxDB ► NetBackup	Red Hat OpenShift	12

2.7 Storage solution

The storage architecture supports storage transformation goals. The environment seeks to position a predominantly enterprise storage array-based deployment to primarily software-defined storage (Ceph SDS) based infrastructure. Figure 2-6 is the basis for the IBM solution around storage, backup, and migration, which are described in 3.7, “Red Hat Ceph Storage” on page 175.

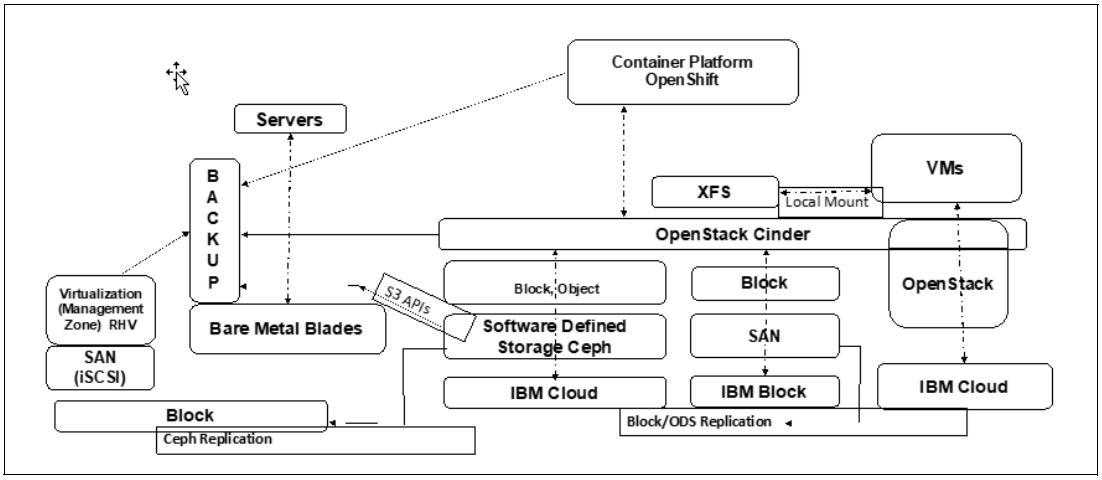


Figure 2-6 Storage architecture for Ceph

The SDS storage nodes are integrated with Red Hat OpenStack Cinder. The block storage (ODS) is also subjected to storage virtualization with Cinder. This also includes persistent volumes that are needed by containerized applications that are deployed over Red Hat OpenStack.

The SDS solution that was chosen is Red Hat's distribution of open source Ceph. The Ceph sizing is summarized in Table 2-3.

Table 2-3 Software-Defined Storage sizing

Servers	Number of disks per server	Total storage per server	Total raw storage	Total usable storage
4	4 * 960 GB SSD	3.2 TB	12.8 TB	10 TB

Synchronous redundancy of Ceph clusters is enabled, as are cross-site asynchronous replication. Both take advantage of a judicious combination of replication and erasure coding.

2.8 Network design

Software-defined networking is a principal tenet of the environment on the solution. Leaf-spine network architecture, along with the controller, provides the SDN platform for underlay and the overlay. The overlay integrates with the Red Hat OpenStack cloud orchestrator layer for automated provisioning of network.

The SDN solution provides built-in security through contracts for traffic and services integration with use of the IBM Cloud external firewall and routers for traffic flow and security. The network is integrated on Layer-2 with an existing network for communication and services architecture.

2.9 Services Management System

Through the Services Management System (SMS) design, we bring the rigor and structure of thinking and designing all facets of service management in an end-to-end fashion. For example, the entire estate, regardless of whether it is containerized, virtualized, or bare metal, feature the same monitoring strategy. Monitoring health can be viewed from a single dashboard. We also carefully investigated and chose open source tooling for all elements of service management, ranging from asset management (SnipeIT Open-AudIT) and ITSM tool (iTop) to monitoring (Prometheus) and event management (SEC).

The SMS solution encompasses a holistic design that subsumes operational log analytics, capacity planning, and anomaly detection.

2.10 Operational analytics

The IBM architecture infuses Artificial Intelligence (AI) and machine learning for the operation of IT. It collects data from various sources in the cloud data center, and then analyzes, correlates, and predicts system performance parameters. This process manifests itself in the form of increased visibility of the environment along with the ability to predict and isolate factors that lead to risk.

The analytics solution is anchored around the ELK stack, which detects unusual patterns and variations in log data to point at emerging problems. It can, for example, be used to debug slowness in a server hosting critical applications, if that manifests through logs. Similarly, it can trend usage metrics, perform correlation, and identify patterns that are suspected system anomalies.

2.11 Centralized management

Most dashboards and portals that are used by the platform are “framed” inside a content management framework (the open source Liferay portal). Liferay is integrated to a directory server (LDAP) by way of the open source single sign-on tool (Gluu) to provide simplified authentication.

Portals of the cloud management platform (Red Hat CloudForms), the container platform (Red Hat OpenShift), the ITSM tool (iTop), log analytics visualization (Kibana), monitoring portal (Grafana), and DevOps tracking tool (Jira) are aggregated inside Liferay. Specific specialized dashboards (such as SDN controller and automation engine configuration) are outside of the ambit of portal aggregation for efficiency and user experience reasons.

2.12 Container architecture

The container platform for reference is Red Hat OpenShift that builds an enterprise-grade support model around the open source container engine. The container platform is deployed on top of the Red Hat OpenStack private cloud to use on-demand and scale efficiencies. In this deployment, the Red Hat OpenShift container platform architecture is next to Red Hat OpenStack on bare metal servers (see Figure 2-7).

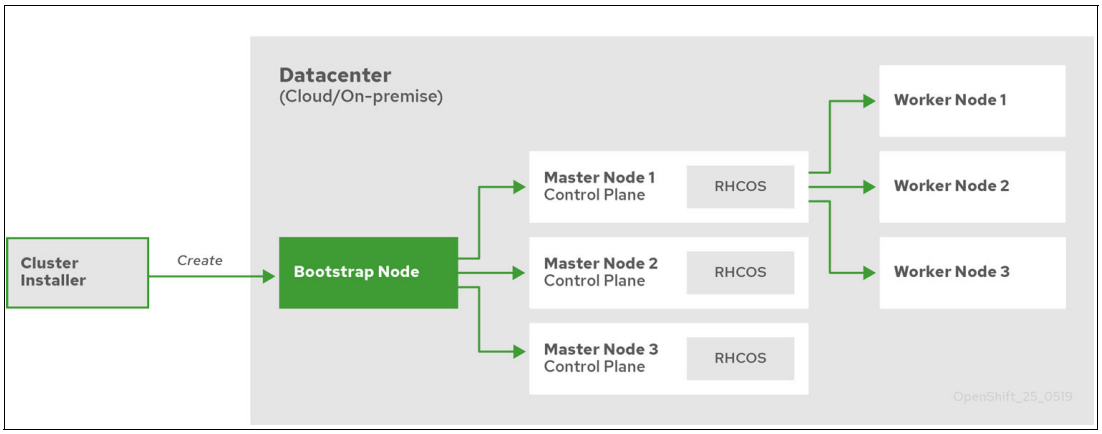


Figure 2-7 Container platform architecture

The container management solution components and the corresponding work breakdown structure are described by the following process:

1. Deploy the Red Hat OpenShift container engine on bare metal.
2. Create clusters (topology, as described in 3.3, “Red Hat OpenShift” on page 65) on Red Hat OpenShift.
3. Establish a network path from the clusters to management tooling.
4. Configure persistent storage for containerized applications.
5. Configure cluster managing components for monitoring, log collection, and capacity management.
6. Test integration of CI/CD environment with Red Hat OpenShift.
7. Configure user and role authorization.
8. Manage the operating system of the worker nodes and the master nodes.
9. Create and manage the Red Hat OpenShift dashboard (PaaS elements in the catalog are described in 3.3.7, “Administration of Red Hat OpenShift Container” on page 102).

The primary usage path for the Red Hat OpenShift platform is to integrate the DevOps chain with the PaaS catalog that is made available on the Red Hat OpenShift dashboard (as described in 3.3.7, “Administration of Red Hat OpenShift Container” on page 102).

Note: The private cloud and container stacks deployment model has been designed with Red Hat OpenShift regions: one region for production and the other region for the non-production environment. Also Red Hat CloudForms are deployed in active-passive mode as the front end for the private cloud.

2.13 Management platform architecture

The architecture tooling solution that is based on ITSM covers an end-to-end, integrated platform for IT infrastructure and its services, which is infused with intelligence that enables a higher quality of service, better visibility into the IT environment and improved cost and performance optimization. The platform uses open standards-based components that are reusable and integrate cognitive insights throughout the IT infrastructure and services delivery. It uses a data lake, a proprietary machine learning as cognitive insights engine, and runs on private cloud. The platform provides a unified architecture that ensures pervasive security, continuous governance and compliance, and an agile DevOps process that enables us to benefit and grow (see Figure 2-8).

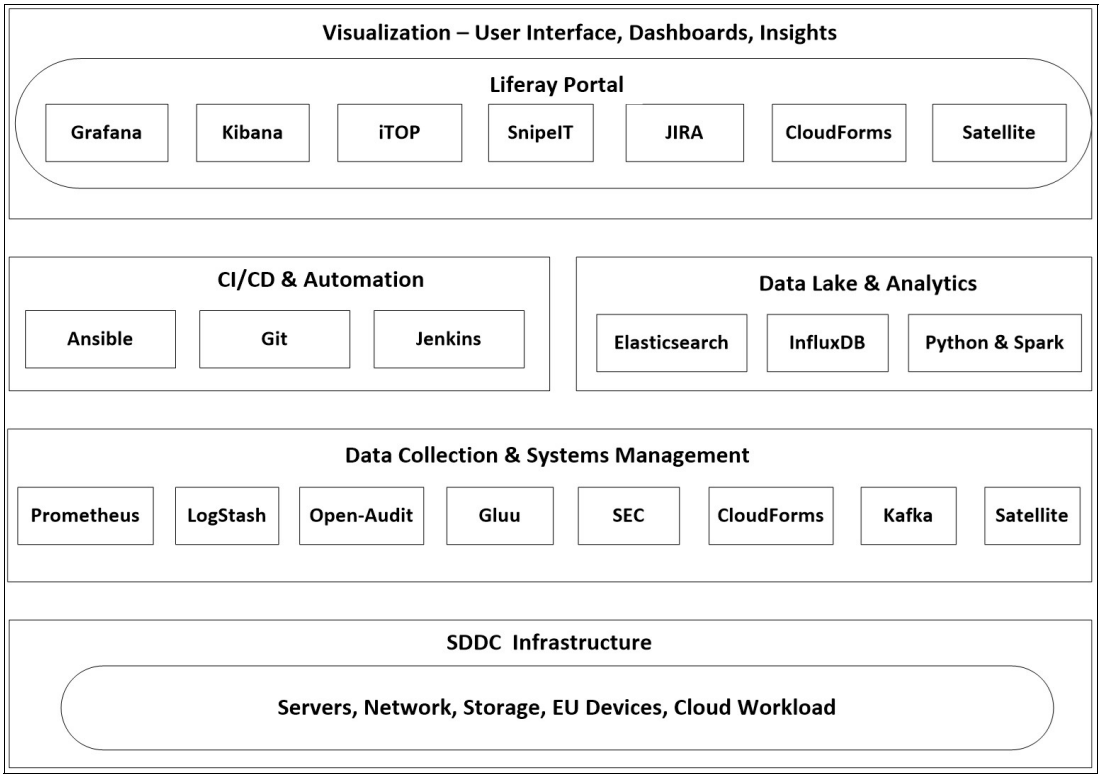


Figure 2-8 Management architecture: Visualization interface

2.13.1 Visualization: Dashboards and insights

Client Insight® dashboards provide architects and admins full transparency and control of managed services and systems, and advice on business and technology decisions. The following elements are the sources that comprise this service:

- Liferay Portal

  Liferay Portal is a no cost and open source enterprise portal that is written in Java and distributed. It allows users to set up features that are common to web pages and web content. An advantageous feature of Liferay is functional units called *portlets*. Liferay is sometimes described as a content management framework or a web application framework. Liferay's support for plug-ins extends into multiple programming languages, including support for PHP and Ruby portlets. Liferay acts as an integrator of all control panes across various tools that are deployed in the environment, which facilitates the ability to centralize all of this content into a single portal.

- Grafana

  Grafana provides you with a plethora of visualization options, which gives a consolidated view of the IT infrastructure. It visualizes based on time series data. With this tool, you can create, explore, and share dashboards with different teams with a diversity of visualizations. In this architecture, Grafana is used to visualize the data that is near real time for IT Analytics from Monitoring, events and SLAs.

- Kibana

  Kibana is an open source analytics and visualization platform that works with Elasticsearch. The architecture uses Kibana to search, view, and interact with data that is stored in Elasticsearch indices. Kibana runs advanced data analysis and visualizes data in various charts, tables, and maps.

- iTop

  iTop is an Open Source Web 2.0 service management application that helps IBM support the IT by providing modules, including CMDB, Service Request, Incident, Problem, Change, and Service Level Management. iTop supports JSON-based APIs integrations with other sources for events, although CMDB (among other integrations) is required.

- Jira

  A critical component of service delivery and continuous integration, continuous delivery is used to plan, track, and manage Site Reliability Engineering (SRE) and DevOps practices for agile methodology, software development projects, and platform remediation. In Jira, the service delivery teams can track the project, customize workflows, collaborate, and task release patch, software upgrades, and new application implementations.

- Red Hat CloudForms

  Red Hat CloudForms is a key data collection and systems management tool for private cloud. It brings automation and management capabilities that are not included in Red Hat OpenStack but are required for an enhanced automation experience.

- Red Hat Satellite

  Red Hat Satellite is also a key data collection and systems management tool. It permits systems management and data collection for all of the deployed hosts in the environment.

2.13.2 Continuous Integration Continuous Delivery: Automation

The following elements are the sources that comprise this service:

- Red Hat Ansible Tower

  Red Hat Ansible Tower is for provisioning, configuration management, incident automation, asset discovery, and application deployment tool. It operates as an IT automation engine that transforms the repetitive, inefficient tasks of software release cycles into predictable, scalable, and simple processes. Red Hat Ansible benefits and automates cloud provisioning, application deployment, configuration management, and service orchestration so that developers can spend more time on their work and help operations more easily support deployment pipelines.

- Git

  Git is an open source distributed version control system. It does not contain a GUI, but unlike GitHub, it can run locally. It handles minor to major projects with high speed and efficiency and was developed to coordinate the work among the developers. The version control allows users to track and work with service team members at the same workspace.

- Jenkins

  Jenkins is used for continuous deployment and continuous integration, self-contained, open source automation server that is used to automate various tasks that are related to building, testing, and delivering or deploying software.

2.13.3 Data lake and analytics

The following elements are the sources that comprise this service:

- Elasticsearch is a distributed, RESTful search and analytics engine that is part of ELK stack. It is used to store, search, and analyze significant volumes of IT Infrastructure data quickly and in near real time.
- InfluxDB is a time-series database (TSDB), which provides an SQL-like query language. In InfluxDB, a metric corresponds to a measurement in the database. In combination with Grafana as a visualization tool and the ITops service, InfluxDB supports various data sources, including InfluxDB databases, for the creation of a massive data lake that is used across all analytics tools in EMS.
- Python is used as a data science tool and a custom code connector that is used to create custom models and baselines from the historical data and near real-time data that is ingested into the data analytics platform. Sophisticated machine learning algorithms are created to detect anomalies by comparing the real-time data with historical data that is stored in Elasticsearch.
- Spark streaming enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Spark uses data from the Kafka layer.

2.13.4 Data collection and system management

The following elements are the sources that comprise this service:

- Prometheus: An open source time-series database and monitoring system, it is the monitoring solution for Kubernetes metrics. IBM uses its powerful query language to retrieve and evaluate monitoring metrics across the entire environment. Based on agentless monitoring, it can take advantage of the use of custom made monitoring scripts that are critical to generate a holistic monitoring solution infrastructure.
- Logstash: Part of ELK stack, Logstash is an open source data collection engine with real-time pipelining capabilities. Logstash can dynamically unify data from disparate sources and normalize the data into destinations of your choice. It also can cleanse and democratize all your data for diverse advanced downstream analytics and visualization use cases.
- Open-AudIT: This element is a discovery tool that scans network subnets and audit Windows and Linux computers, and SNMP scan network devices. Discovery runs entirely from the web interface.
- Gluu: The Gluu Server is a container distribution of free open source software (FOSS) for identity and access management (IAM) that is used for:
  - Single sign-on (SSO)
  - Mobile authentication
  - API access management
  - Two-factor authentication (2FA)
  - Customer identity and access management (CIAM)
  - Identity federation

- Snipe-IT is Open Source (FOSS) project is built on Laravel 5.5 to use for Asset Management. The discovered data from Open-Audit tool populates in Snipe-IT along with other data sources, such as Red Hat CloudForms, and then Asset Management functions, such as inventory management and software license management are performed by using the Snipe-IT tool.
- Simple Event Correlator (SEC) is an event correlation tool for advanced event processing. Event correlation is a procedure where a stream of events is processed to detect (and act on) specific event groups that occur within predefined time windows. Unlike many other event correlation products that are heavyweight solutions, SEC is a lightweight and platform-independent event correlator that runs as a single process.
- Red Hat CloudForms: Red Hat CloudForms is an infrastructure management platform that allows IT departments to control users' self-service abilities to provision, manage, and ensure compliance across VMs and private clouds from a single pane of glass.
- Kafka: Apache Kafka is an open source stream-processing software platform that is written in Scala and Java. The project aims to provide a unified, high-throughput, low-latency platform for real-time handling data feeds. It allows you to build sophisticated data pipelines. Kafka is acts as three pieces of technology rolled into one:
  - Message Queue: Reads and writes streams of data.
  - Message Broker: Processes messages and applies different logic to them to react to events in real time.
  - Data store: The decoupling of publishing and consuming messages means that Kafka stores those in-flight messages.
- Red Hat Satellite: Red Hat Satellite is an infrastructure management product that keeps Red Hat Enterprise Linux environments and other Red Hat infrastructure running efficiently with security, and is compliant with various standards. By automating most system maintenance tasks, Red Hat Satellite helps organizations increase efficiency, reduce operational costs, and enable IT to better respond to strategic business needs. Red Hat Satellite automates many tasks that are related to system management and easily integrates into existing workflow frameworks. The centralized console gives administrators a single location for accessing reports and for provisioning, configuring, and updating systems.

2.14 Integration architecture

Integration architecture explains how tools are solutions in the multi-layer architecture and how they integrated to deliver functional requirements with SMS solution (see Figure 2-9).

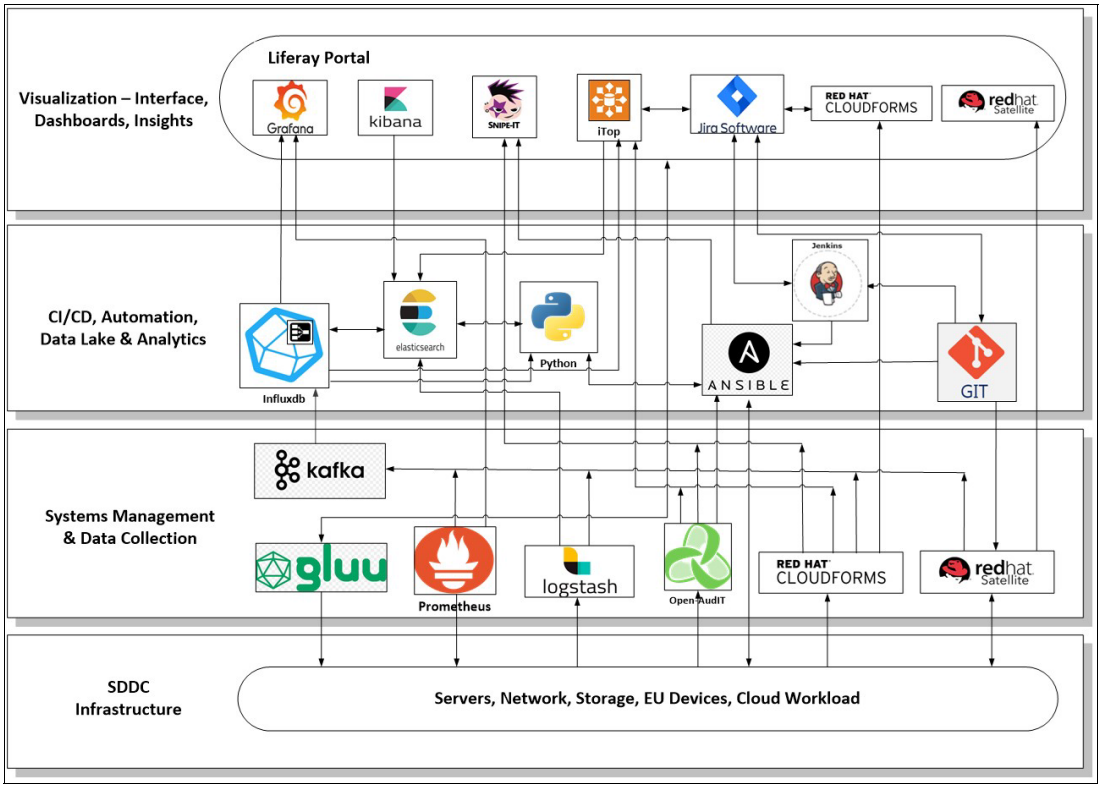


Figure 2-9 Integration architecture overview

Figure 2-9 shows the tools distribution across the four classified layers along with its integrated touch points. In this section, we provide a comprehensive explanation of these diagrams, along with its communication.

The environment consists of servers, a network, storage, user devices, and cloud workloads. All of these systems include data collection, such as logs, metrics for performance, alerts, and upgrades. The systems management and data collection layer acts as a data ingestion layer and management layer for any action, activity, or metric that occurs in the environment.

The Gluu, Kafka, Prometheus, Logstash, Open-Audit, Red Hat CloudForms, and Red Hat Satellite areas build the systems management and data collection layer. All of these tools together meet all of the defined requirements of data, process, and service execution to the control environment.

The communication of the systems management layer to the Data Lake and Analytics layer is through the Kafka messaging hub that captures the incoming data from Prometheus, Logstash, Open-Audit, Red Hat CloudForms, and Red Hat Satellite, and ingest the data to InfluxDB that is a core part of the Data Lake and Analytics layer.

In addition to this integration, three other integrations are essential. One integration comes from Logstash to Elasticsearch, the second integration comes from Red Hat CloudForms to Snipe-IT, and the third integrations comes from Open-Audit to Red Hat Ansible.

This integration allows the Data Lake and Analytics layer to obtain all of the required information for effective analysis.

After all of the data is in the Data Lake and Analytics layer, the following tools are introduced:

- InfluxDb
- Elasticsearch
- Simple Event Correlator
- Snipe-IT
- Red Hat Ansible
- Git
- Jenkins

The Data Lake and Analytics layer is the core engine logic and intelligence built into EMS.

The InfluxDB contains all data that is collected from the various IaaS data sources and monitoring tools. This database is now a central repository to apply data science over the stored data, with analytics tools, such as Python and R programming language. These tools use data from Influx to apply data analytics visualization and apply machine learning of specific data segments to trigger an autonomous script for self-healing and self-remediation.

The IaaS logs that are captured by Logstash become a consumption layer for Elasticsearch, which becomes a principal data log repository for logs and data aggregation that Elasticsearch also extracts from InfluxDB. This configuration helps visualize enhanced metrics in Kibana.

The data analytics layer is critical to EMS overall functions and is the core of the Portal and Visualization layer that is shown in Figure 2-9 on page 28. Grafana, Kibana, and iTop are top uses of information that come from the Data Lake and Analytics layer. In this layer, iTop plays a critical role in the introduction of ITSM processes, through change and release management with Site Reliability Engineering practices and moves this to ITSM practices and turns them into an agile delivery model that uses Jira as the tool.

After Jira is used, the delivery squads with the SRE trigger the changes and releases through Jenkins automation, which is part of the analytics layer. The automation in Jenkins uses Red Hat Ansible to push changes and releases into the IaaS or PaaS. The squads use the term Error Budgets to implement changes and releases into the environment.

Finally, the portal provides access to the cloud function, through that available functions in Red Hat CloudForms and Red Hat Satellite for accessing the service cloud catalog, and provisioning and de-provisioning of resources and cloud enhancements.

3

Building Red Hat hybrid cloud

This chapter discusses how to build the Red Hat cloud environment.

This chapter includes the following topics:

3.1 Setting a bare metal server on IBM Cloud for Red Hat OpenStack

Complete the following steps to provision and configure an IBM Bare metal server to prepare for the Red Hat OpenStack installation after a server request is submitted in the IBM Cloud:

1. Log in to IBM Cloud.

   Figure 3-1 shows the IBM cloud dashboard. The user can click items in the menu bar that is in the upper left corner of the page.

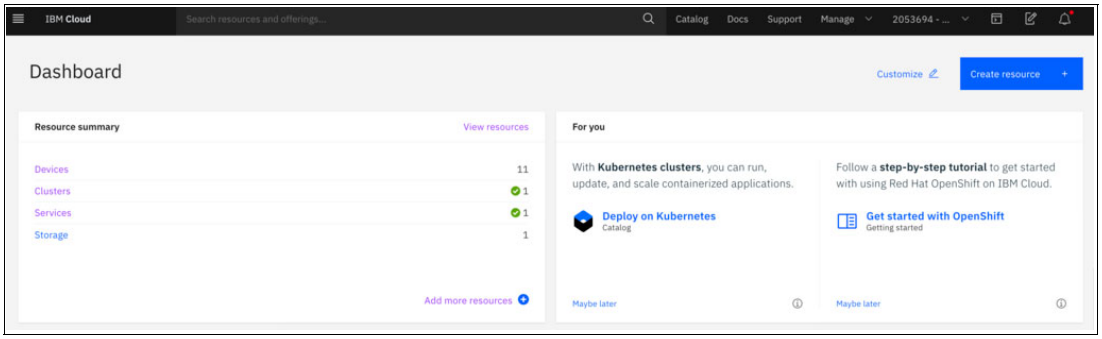


Figure 3-1 IBM Cloud Dashboard

2. Select **Classic Infrastructure**, as shown in Figure 3-2.

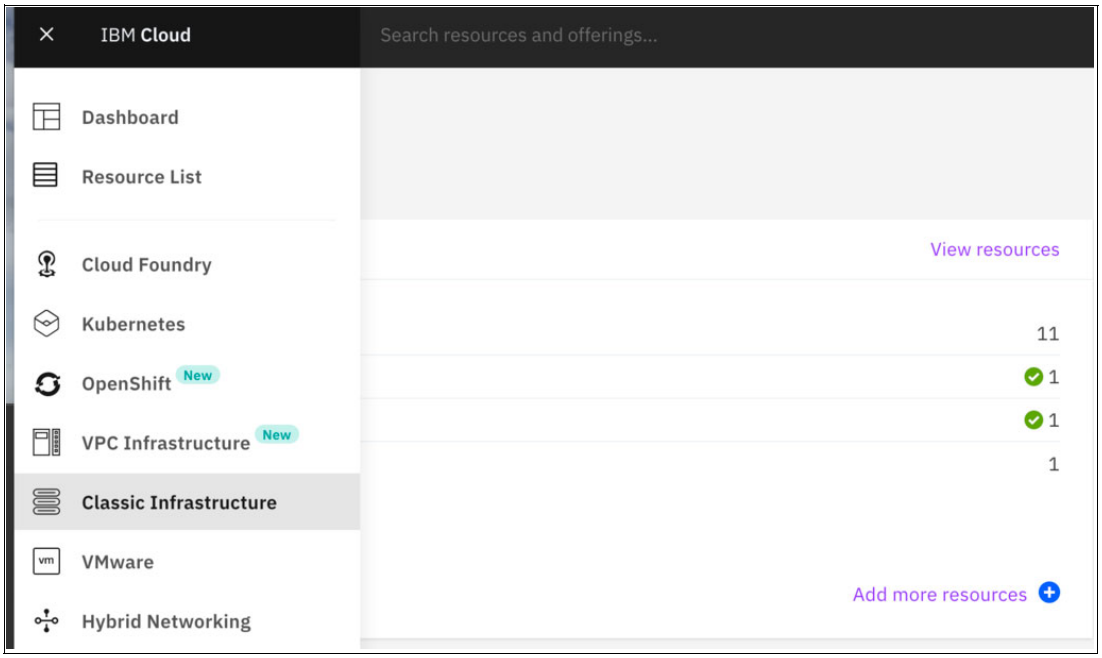


Figure 3-2 IBM Cloud: Classic Infrastructure window

3. A list of provisioned resources is shown. Click **Add more resources** → **Order Devices**, as shown in Figure 3-3.

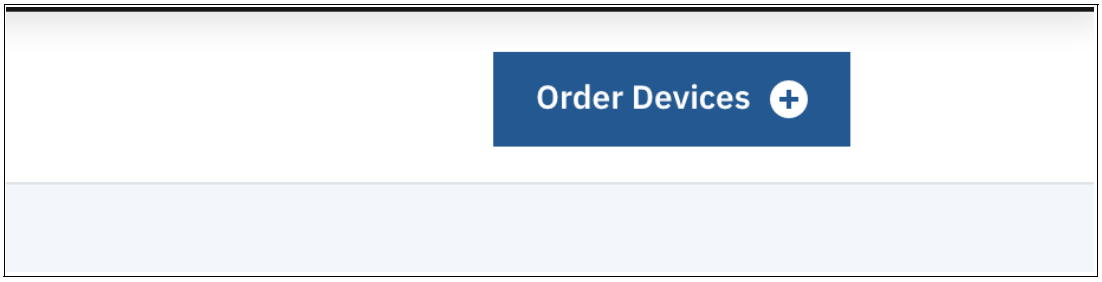


Figure 3-3 Order Devices menu window

4. After the Order Devices selection option appears, select **Bare Metal Server**, as shown in Figure 3-4.

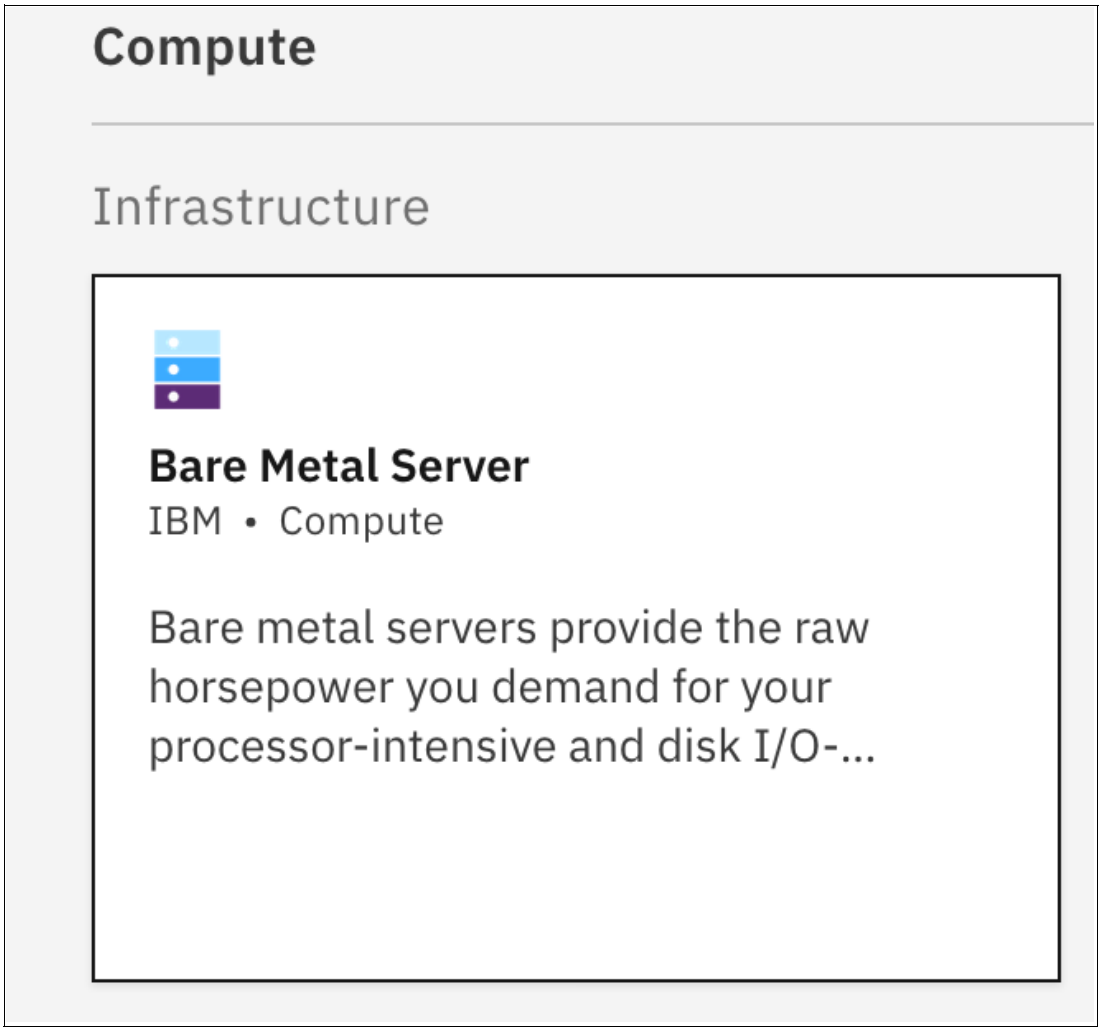


Figure 3-4 Compute Infrastructure: Bare metal server window

5. Select the first bare metal server for the Red Hat cloud installation, as shown in Figure 3-5.

> **Note:** All bare metal servers can be provisioned at once, if wanted. However, we recommend provisioning them in a staged approach because not all servers are required at the same time, and such an approach avoids charges for servers that are not used.

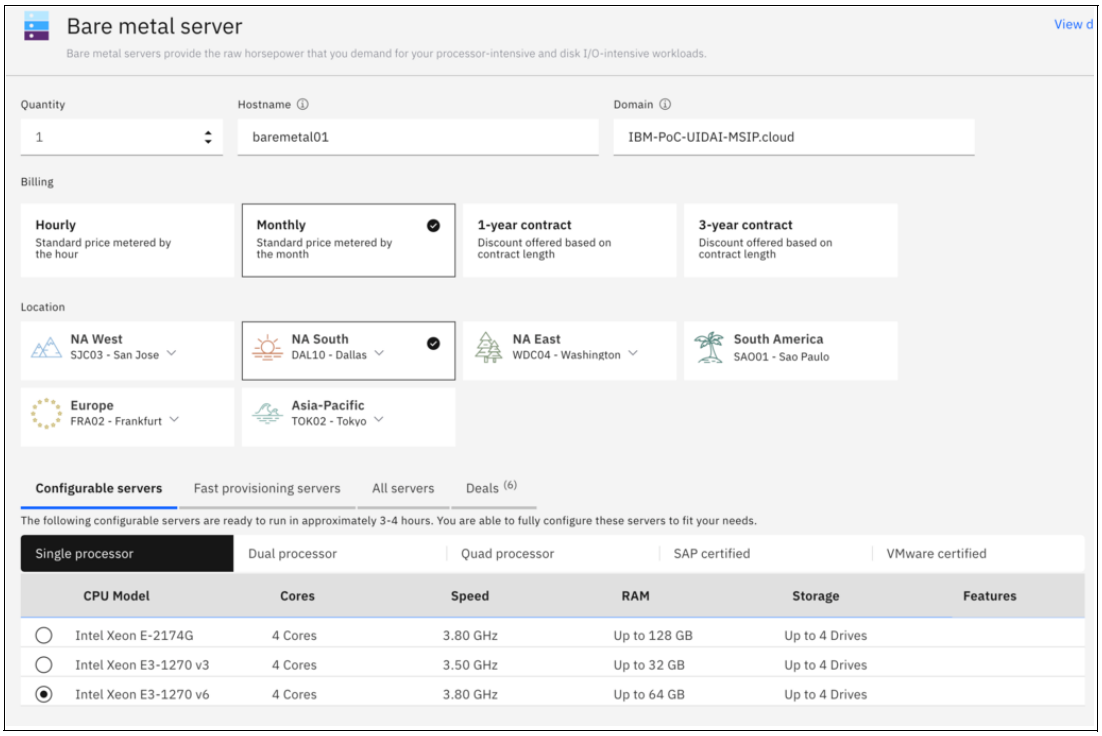


Figure 3-5 Bare metal server window

For more information about the configuration for the Red Hat OpenStack cluster, see 3.2, “Red Hat OpenStack installation” on page 40.

The server preparation process takes 1 - 3 hours to complete, depending on the server configuration that is requested. After the server is provisioned, several items must be changed in your browser configurations to ensure that the server is accessible.

Complete the following steps:

1. Access the Intelligent Platform Management Interface (IPMI), which is a set of computer interface specifications for an autonomous computer subsystem that provides management and monitoring capabilities independently of the host system’s CPU, firmware (BIOS or UEFI), and operating system.

   This step requires VPN access. VPN users require vpn-access authority. An IBM Cloud account administrator (Administrator) invites a user with VPN-only user.

   Browse by the using VPN in IBM Cloud with the following URL:

   `https://www.ibm.com/cloud/vpn-access`

2. Click the data center that is closer to your location, as shown in Figure 3-6.

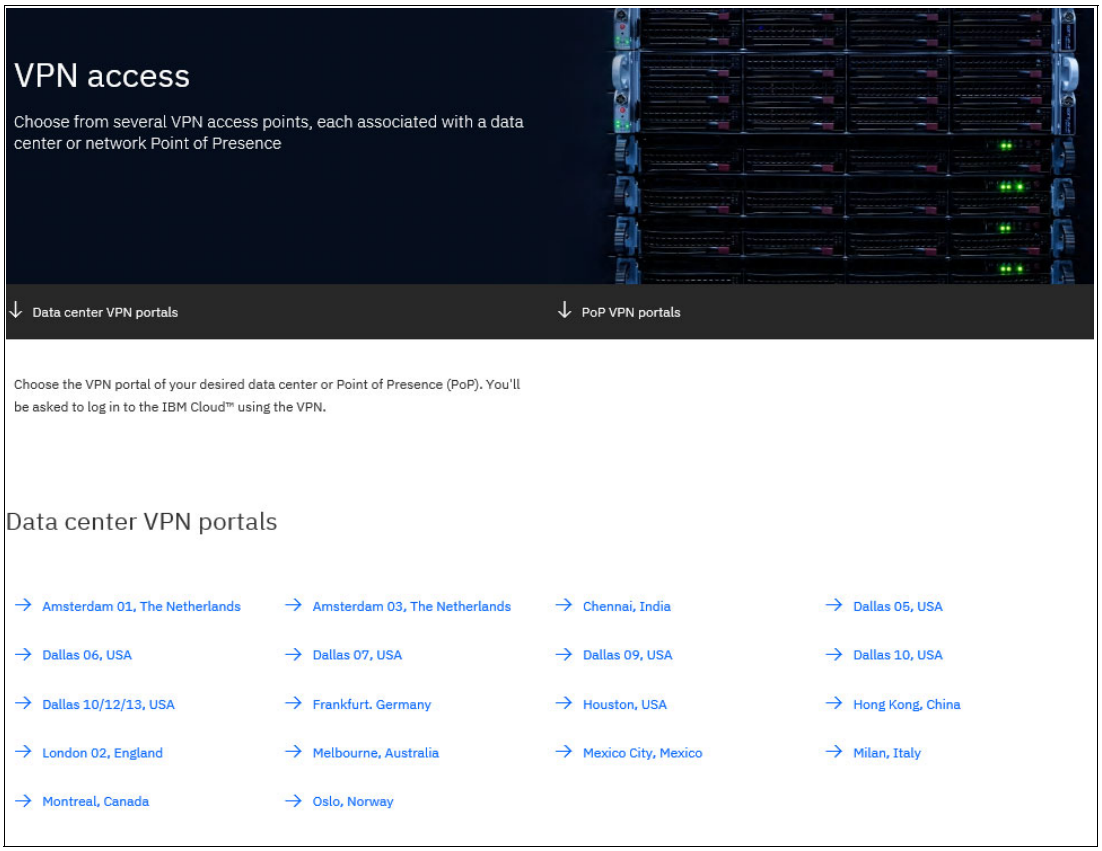


Figure 3-6 VPN access: Data Center VPN portals window

3. Click **Add VPN-only user**. Enter a VPN user name and password, as shown in Figure 3-7.

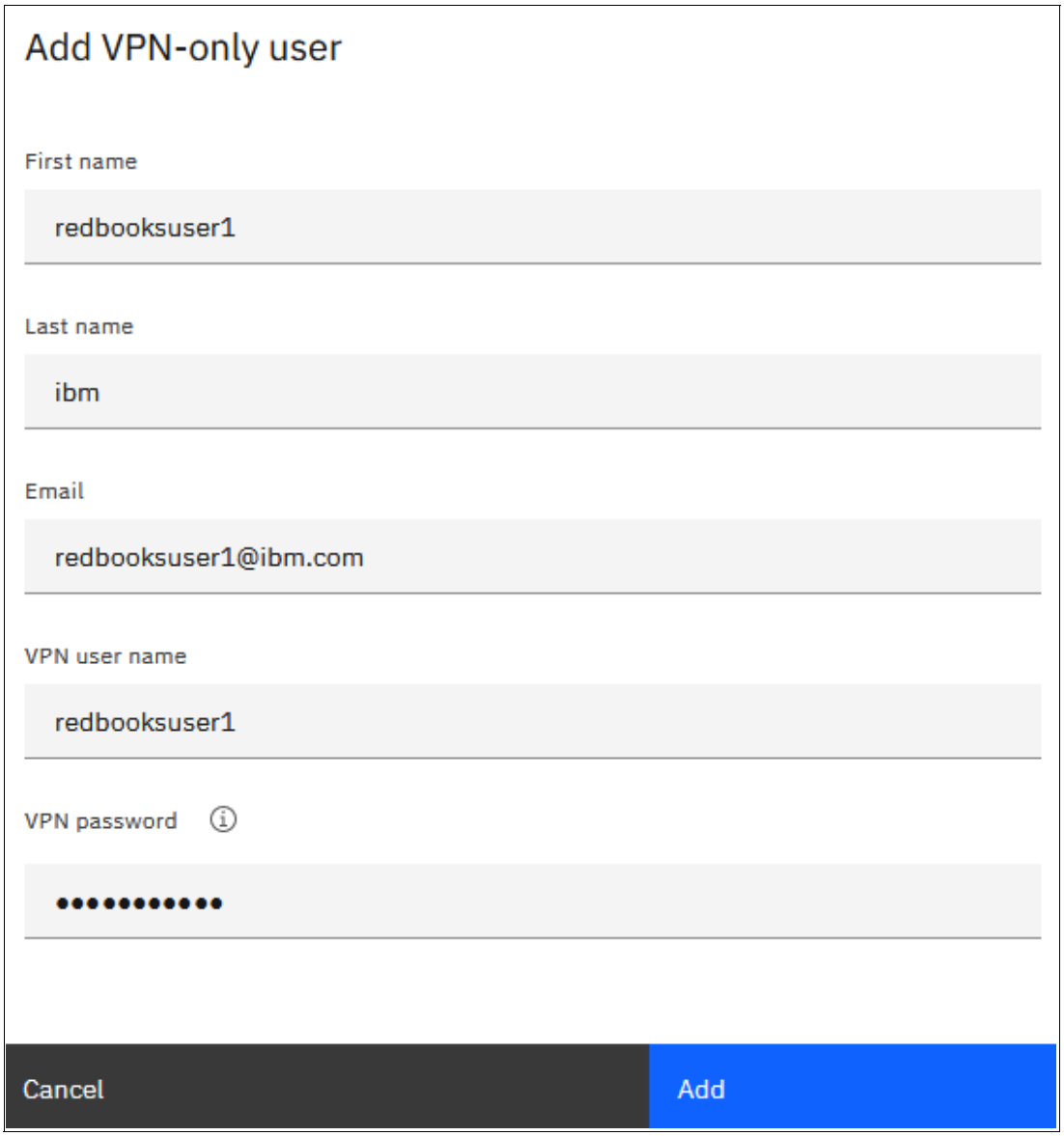


Figure 3-7 Add VPN-only user pane

4. Click **Manage** → **Access (IAM)** → **Users**, as shown in Figure 3-8.

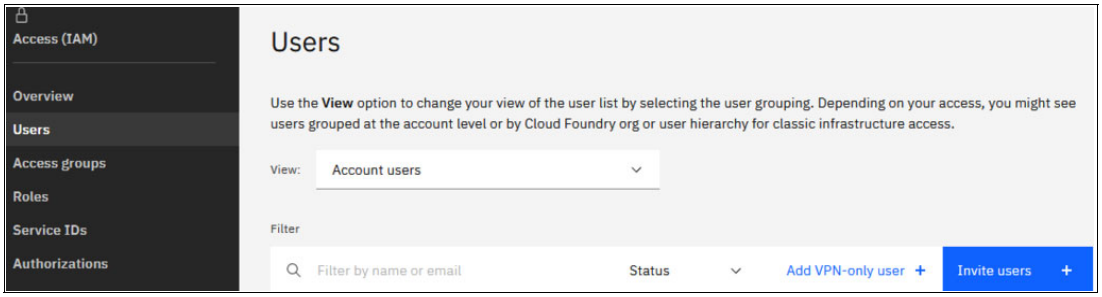


Figure 3-8 Access (IAM): Users window

The following prerequisite steps must be complete to access IPMI:

1. On a Windows machine, validate internet options. Go to the pop-up blocker settings, disable pop-up blocker settings, and confirm that no other pop-up blocker settings are enabled.
2. Configure the Private IPMI address of the server that is found in the **Server Devices** → **Private IP - Public IP** → **IP for IPMI**. Choose **IPMI IP** and include it in the allowed pop-up blocker websites for suitable access. Java also must be enabled and the browser must be set to the minimal level of security. After this process runs, restart your browser so that the configurations take effect.

   The IPMI IP is available after the configuration is run, as shown in Figure 3-9.

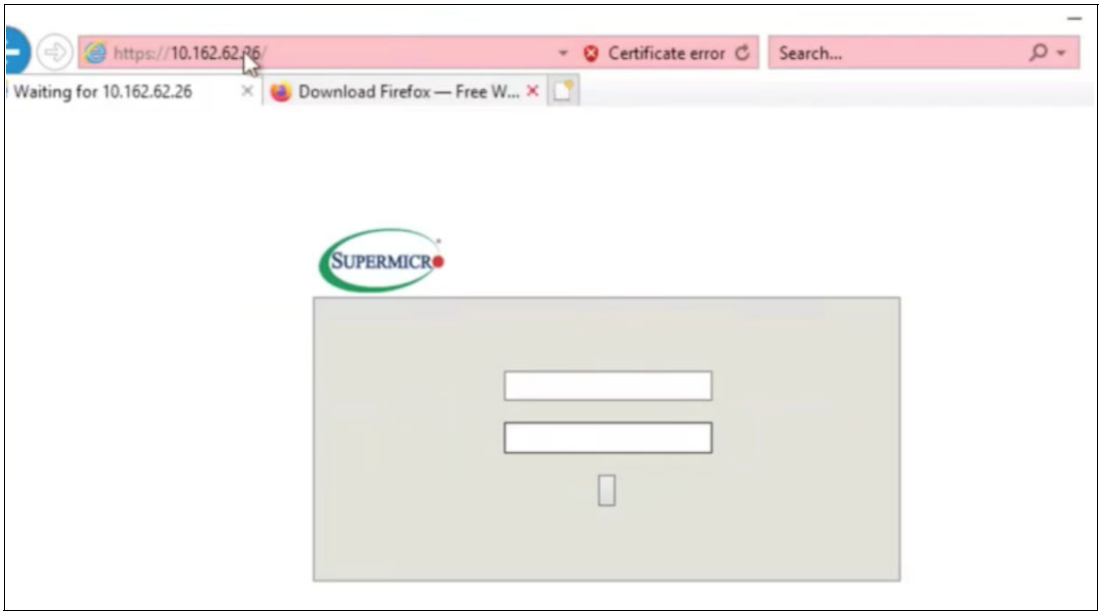


Figure 3-9 IPMI IP address information window

3. Access the bare metal server by using the credentials that are found in the IBM Cloud Dashboard (see Figure 3-1 on page 32). Under devices, select the bare metal server.

4. A set of configurations is required in the Jumpstation server. For this process, use Microsoft Server Manager, as shown in Figure 3-10. Select **Local Server**. In the Properties section, validate that no block of traffic for accessing local sites exists.

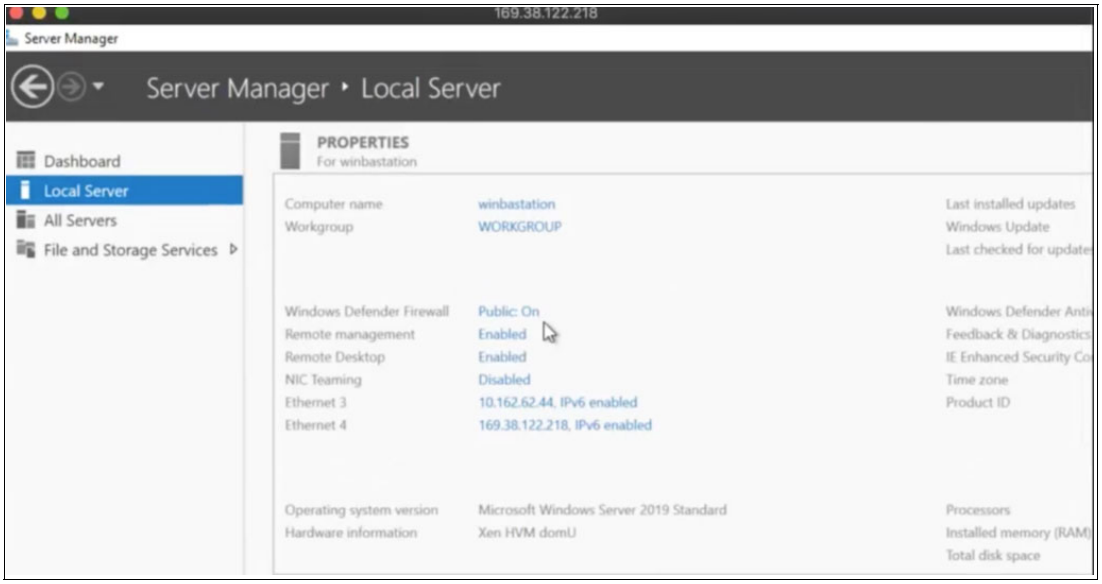


Figure 3-10 Server Manager: Local Server window

After the login is complete, the information that is shown in Figure 3-11 is displayed.

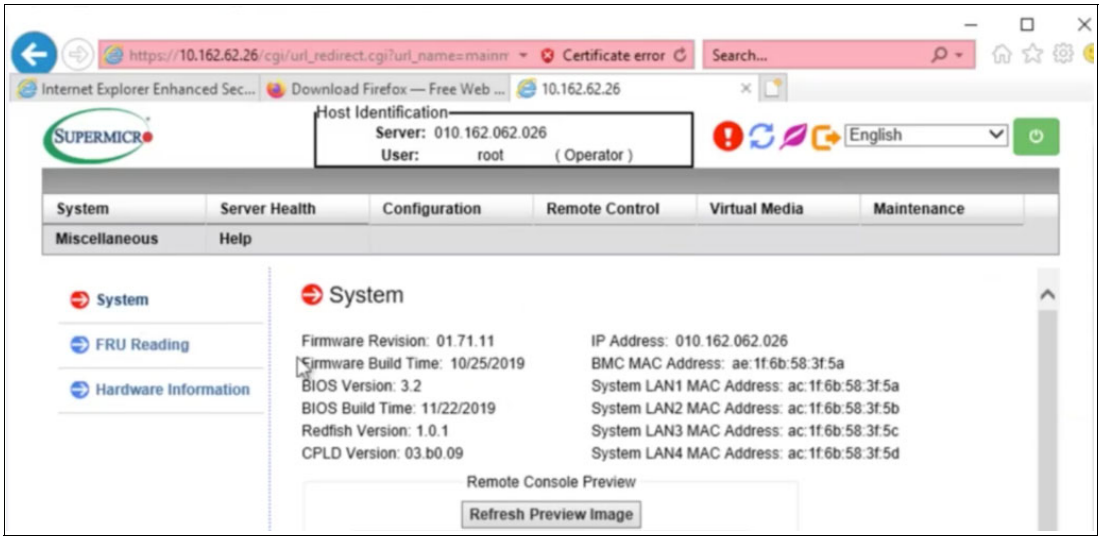


Figure 3-11 System Host Identification window

5. Although several methods can be used to access the server, we use the console redirection in the remote control section for this installation, because the java plug-in was enabled to run it. After the console redirection is started, the server console connect appears as shown in Figure 3-12.

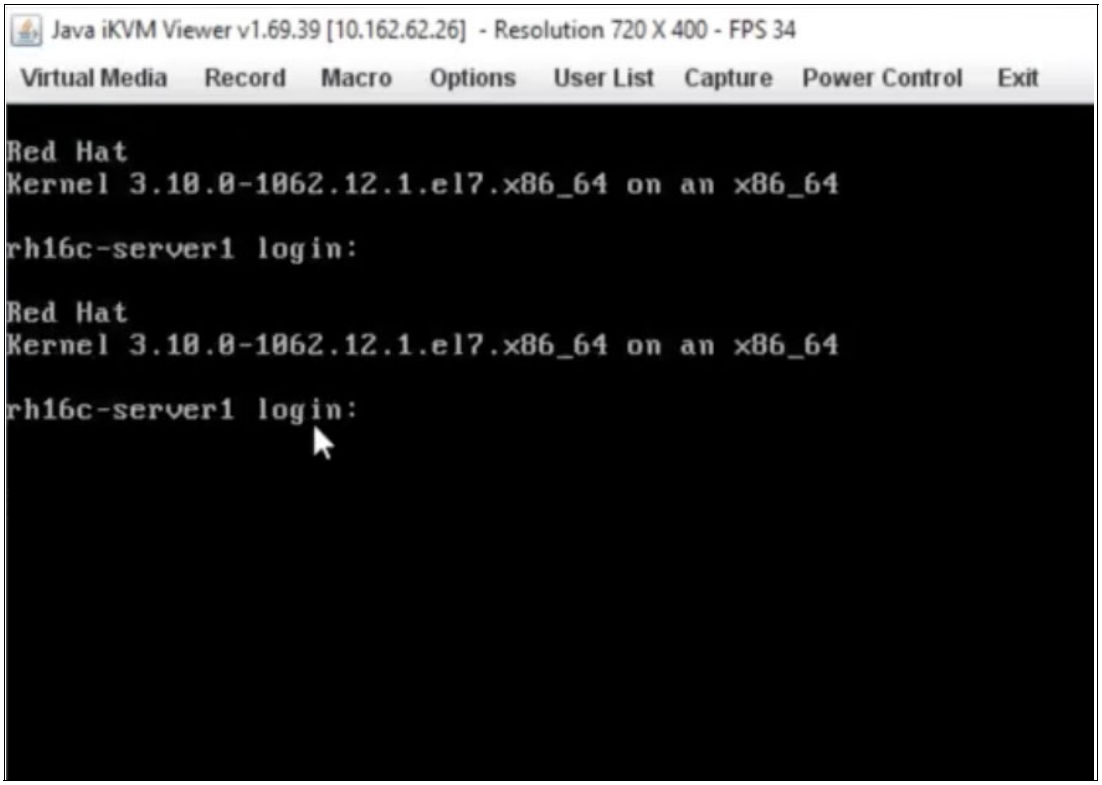


Figure 3-12 Console connection login

6. Test the access to the server by using the provided root account and password that was generated in the IBM Cloud console, as shown in Figure 3-13.

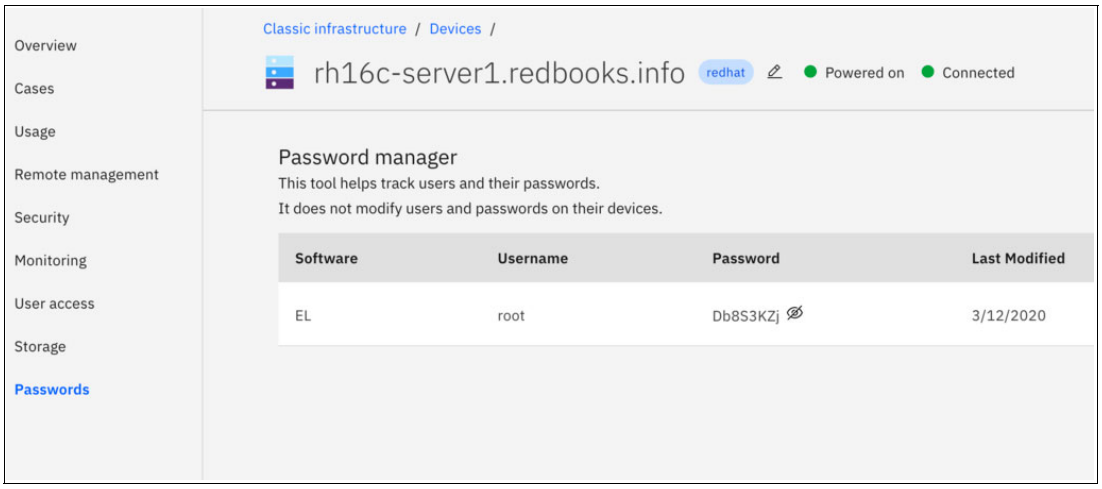


Figure 3-13 Classic Infrastructure: Devices: Password manager

7. Validate and display modifying the IP address by running the **sudo** command:

```
[root@rh16c-server1 ~]: # ip a
```

3.2 Red Hat OpenStack installation

This section describes the Red Hat OpenStack deployment.

3.2.1 Using Red Hat OpenStack Platform director to create a Red Hat OpenStack cloud

This process includes installing the director, planning your environment, and creating a Red Hat OpenStack environment by using the director.

Red Hat OpenStack Platform director is a toolset for installing and managing a complete Red Hat OpenStack environment. It is based on the Red Hat OpenStack project TripleO (OpenStack-On-OpenStack). This project takes advantage of Red Hat OpenStack components to install a fully operational Red Hat OpenStack environment.

Red Hat OpenStack components are included that provision and control bare metal systems to use as Red Hat OpenStack nodes. This director toolset provides a method for installing a complete Red Hat OpenStack Platform environment that is lean and robust. The Red Hat OpenStack Platform director uses two main concepts: an undercloud and an overcloud. The undercloud installs and configures the overcloud. These concepts are described next.

3.2.2 Undercloud

The undercloud is the main director node. It is a single-system Red Hat OpenStack installation that includes components for provisioning and managing the Red Hat OpenStack nodes that form your Red Hat OpenStack environment (the overcloud).

The components that form the undercloud provide the following functions:

- Environment planning: The undercloud provides planning functions for users to create and assign certain node roles. The undercloud includes a default set of nodes, such as Compute, Controller, and various storage roles, and the ability to use custom roles. In addition, you can select which Red Hat OpenStack Platform services to include on each node role, which provides a method to model new node types or isolate certain components on their own host.
- Bare Metal System Control: The undercloud uses out-of-band management interface (often the IPMI) of each node for power management control and a PXE-based service to discover hardware attributes and install Red Hat OpenStack to each node. This function provides a method to provision bare metal systems as Red Hat OpenStack nodes.
- Orchestration: The undercloud provides a set of YAML templates that acts as a set of plans for your environment. The undercloud imports these plans and follows their instructions to create the resulting Red Hat OpenStack environment. The plans also include hooks with which you can incorporate your own customizations as specific points in the environment creation process.
- Command Line Tools and a Web UI: The Red Hat OpenStack Platform director performs these undercloud functions through a terminal-based command line interface or a web-based user interface.
- Undercloud Components: The undercloud uses Red Hat OpenStack components as its base tool set, including the following components:
  - Red Hat OpenStack Identity (keystone): Provides authentication and authorization for the director's components.

- Red Hat OpenStack Bare Metal (ironic) and Red Hat OpenStack Compute (nova): Manages bare metal nodes.
- Red Hat OpenStack Networking (neutron) and Open vSwitch: Controls networking for bare metal nodes.
- Red Hat OpenStack Image Service (glance): Stores images that are written to bare metal machines.
- Red Hat OpenStack Orchestration (heat) and Puppet: Provides the orchestration and configuration of nodes after the director writes the overcloud image to disk.
- Red Hat OpenStack Telemetry (ceilometer): Performs monitoring and data collection. This also includes: Red Hat OpenStack Telemetry Metrics (gnocchi), which provides a time series database for metrics.
- Red Hat OpenStack Telemetry Alarming (aodh): Provides an alarm component for monitoring.
- Red Hat OpenStack Telemetry Event Storage (panko): Provides event storage for monitoring.
- Red Hat OpenStack Workflow Service (mistral): Provides a set of workflows for specific director-specific actions, such as importing and deploying plans.
- Red Hat OpenStack Object Storage (swift): Provides Object Storage for various Red Hat OpenStack Platform components, including image storage for Red Hat OpenStack Image Service:
  - Introspection data for Red Hat OpenStack Bare Metal
  - Deployment plans for Red Hat OpenStack Workflow Service

3.2.3 Overcloud

The overcloud is the result of the Red Hat OpenStack Platform environment that is created by using the undercloud. This overcloud includes different nodes roles that you define based on the Red Hat OpenStack Platform environment you want to create.

The undercloud includes a default set of overcloud node roles:

- Controller nodes that provide administration, networking, and high availability (HA) for the Red Hat OpenStack environment. An ideal Red Hat OpenStack environment recommends three of these nodes together in a HA cluster.

  A default Controller node contains the following components:

  - Red Hat OpenStack Dashboard (horizon)
  - Red Hat OpenStack Identity (keystone)
  - Red Hat OpenStack Compute (nova) API
  - Red Hat OpenStack Networking (neutron)
  - Red Hat OpenStack Image Service (glance)
  - Red Hat OpenStack Block Storage (cinder)
  - Red Hat OpenStack Object Storage (swift)
  - Red Hat OpenStack Orchestration (heat)
  - Red Hat OpenStack Telemetry (ceilometer)
  - Red Hat OpenStack Telemetry Metrics (gnocchi)
  - Red Hat OpenStack Telemetry Alarming (aodh)
  - Red Hat OpenStack Telemetry Event Storage (panko)
  - Red Hat OpenStack Clustering (sahara)
  - Red Hat OpenStack Shared File Systems (manila)
  - Red Hat OpenStack Bare Metal (ironic)
  - MariaDB

  - Open vSwitch
  - Pacemaker and Galera for HA services
- Compute nodes provide computing resources for the Red Hat OpenStack environment. You can add Compute nodes to scale out your environment over time. A default Compute node contains the following components:
  - Red Hat OpenStack Compute (nova)
  - KVM/QEMU
  - Red Hat OpenStack Telemetry (ceilometer) agent
  - Open vSwitch
- Storage nodes that provide storage for the Red Hat OpenStack environment. This includes nodes for:
  - Ceph Storage nodes: Used to form storage clusters. Each node contains a Ceph Object Storage Daemon (OSD). In addition, the director installs Ceph Monitor onto the Controller nodes in situations where it deploys Ceph Storage nodes.
  - Block storage (cinder): Used as external block storage for HA Controller nodes. This node contains the following components:
    - Red Hat OpenStack Block Storage (cinder) volume
    - Red Hat OpenStack Telemetry (ceilometer) agent
    - Open vSwitch
  - Object Storage (swift): These nodes provide an external storage layer for Red Hat OpenStack Swift. The Controller nodes access these nodes through the Swift proxy. This node contains the following components:
    - Red Hat OpenStack Object Storage (swift) storage
    - Red Hat OpenStack Telemetry (ceilometer) agent
    - Open vSwitch

3.2.4 Installing undercloud

This section describes installing the undercloud.

Creating a stack user

The director installation process requires a non-root user to run commands. Complete the following steps to create the user name `stack` and set a password:

1. Log in to your undercloud as the root user.
2. Create the stack user: `[root@director ~]# useradd stack`
3. Set a password for the user: `[root@director ~]# passwd stack`
4. Disable password requirements when using sudo:

   ```
   [root@director ~]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a
   /etc/sudoers.d/stack
   ```

5. Set permissions for user stack: `[root@red-director ~]# chmod 0440 /etc/sudoers.d/stack`
6. Switch to the new stack user: `[root@director ~]# su - stack`
7. New stack user command prompt: `[stack@red-director ~]$`
8. Continue the director installation as the stack user.

9. Create the templates and images directories.

   The director uses system images and Heat templates to create the overcloud environment. To keep these files organized, we recommend creating directories for images and templates:

   ```
   [stack@red-director ~]$ mkdir ~/images
   [stack@red-director ~]$ mkdir ~/templates
   ```

Setting up the undercloud hostname

Undercloud requires a cloud qualified domain name (FQDN) for its installation and configuration process. The DNS server that you use must resolve a fully qualified domain name. IBM Cloud DNS service was used for this deployment.

Complete the following steps:

1. Check the base hostname of the undercloud: `[stack@director ~]$ hostname`.
2. Check the full host name of the `[stack@director ~]$ hostname -f`.
3. If either of the previous commands do not report the correct fully qualified hostname or report an error, run the **`hostnamectl`** command to set a hostname:

   ```
   [stack@director ~]$ sudo hostnamectl set-hostname red-director.redbooks.info

   [stack@director ~]$ sudo hostnamectl set-hostname --transient
   red-director.redbooks.info
   ```

4. The director also requires an entry for the system's hostname and base name in `/etc/hosts`. The IP address in `/etc/hosts` must match the address that you plan to use for your undercloud public API. The system name is `red-director.redbooks.info` and the IP address is `169.38.122.217`:

   ```
   stack@red-director ~]$ cat /etc/hosts
   127.0.0.1 localhost.localdomain localhost
   169.38.122.217 red-director.redbooks.info red-director
   ```

   For more information about undercloud registration and updates, see Appendix A, "Red Hat subscription activation process" on page 401.

5. Register your system with the Content Delivery Network. Enter your Customer Portal user name and password when prompted:

   [stack@director ~]$ sudo subscription-manager register

6. Find the entitlement pool ID for Red Hat OpenStack Platform director (see Figure 3-14):

   sudo subscription-manager attach --pool=8a85f99c6ed1dabc016eef03caf62b

Figure 3-14 Red Hat OpenStack director node registration to RHSM

7. Locate the Pool ID value and attach the Red Hat OpenStack Platform 13 entitlement.
8. Install director packages.

   Install the command line tools for director installation and configuration:

   ```
   [stack@director ~]$ sudo yum install -y python-tripleoclient
   ```

9. Configure the director.

   The Director installation process requires settings specific to network determination configurations, as shown in Example 3-1. All settings are stored in a template called stack user's home directory as `undercloud.conf`.

Example 3-1 Director configuration parameters

```
[DEFAULT]

undercloud_hostname:

undercloud_hostname = red-director.redbooks.info

Local_Ip:

local_ip = 10.163.4.10/24

Undercloud_nameservers:

Undercloud_nameservers = 8.8.8.8,10.0.80.11,10.0.80.12

Undercloud_nameservers:

Undercloud_nameservers = 10.0.77.54,0.in.pool.ntp.org, 1.in.pool.ntp.org

overcloud_domain_name:

overcloud_domain_name = redbooks.info

subnets

subnets = ctlplane-subnet

local_subnet

local_subnet = ctlplane-subnet

local_interface

local_interface = bond0

local_mtu = 1500

inspection_interface = br-ctlplane

inspection_enable_uefi = true

undercloud_debug = true

enable_tempest = false

enable_telemetry = false

enable_ui = false

ipxe_enabled = true
```

```
clean_nodes = true

enabled_drivers = pxe_ipmitool,pxe_drac,pxe_ilo

enabled_hardware_types = ipmi,redfish,ilo,idrac

[ctlplane-subnet]

cidr = 10.163.4.0/24

dhcp_start = 10.163.4.15

dhcp_end = 10.163.4.25

inspection_iprange = 10.163.4.35,10.163.4.50

gateway = 10.163.4.1

masquerade = true
```

Undercloud installation

Run the following command to install the director on the undercloud:

```
[stack@red-director ~]$ sudo openstack undercloud install
```

The director installs more packages and configures its services to suit the settings in the `undercloud.conf` file.

Two files are created when complete:

- `undercloud-passwords.conf`: A list of all passwords for the director's services.
- `stackrc`: A set of initialization variables to help you access the director's command line tools.

The undercloud installation is complete. Consider the following points:

- The file that contains this installation's passwords is stored at:

  `/home/stack/undercloud-passwords.conf`
- A stackrc file is available at: `/home/stack/stackrc`.

As an intermediate step, it is a good practice to validate the Red Hat OpenStack Platform services that are automatically started by running the following command:

```
[stack@red-director ~]$ sudo systemctl list-units openstack-*
```

Running and obtaining images for the overcloud nodes

The director requires several disk images for provisioning overcloud nodes:

- An introspection kernel and ramdisk, which as used for bare metal system introspection over PXE boot.
- A deployment kernel and ramdisk, which is used for system provisioning and deployment.
- An overcloud kernel, ramdisk, and full image, which are a base overcloud system that is written to the node's hard disk.

Complete the following steps:

1. Source the stackrc file to enable the director's command line tools:

   ```
   [stack@director ~]$ source ~/stackrc
   ```

2. Install the `rhosp-director-images` and `rhosp-director-images-ipa` packages:

   ```
   (undercloud) [stack@director ~]$ sudo yum install rhosp-director-images
   rhosp-director-images-ipa
   ```

3. Extract the images archives to the images directory on the stack user's home (/home/stack/images):

   ```
   (undercloud) [stack@red-director ~]$ mkdir ~/images
   (undercloud) [stack@red-director ~]$ cd ~/images
   (undercloud) [stack@red-director images]$ for i in
   /usr/share/rhosp-director-images/overcloud-full-latest-13.0.tar
   /usr/share/rhosp-director-images/ironic-python-agent-latest-13.0.tar; do tar
   -xvf $i; done
   ```

4. Import these images into the director:

   ```
   (undercloud) [stack@red-director images]$ openstack overcloud image upload
   --image-path /home/stack/images/
   ```

   The following images are uploaded into the director:

   - bm-deploy-kernelv
   - bm-deploy-ramdisk
   - overcloud-full
   - overcloud-full-initrd
   - overcloud-full-vmlinuz

5. To validate that the images uploaded successfully, run the following command (see Figure 3-15):

   ```
   (undercloud) [stack@red-director images]$ openstack image list
   ```

```
(undercloud) [stack@red-director ~]$ openstack image list
+--------------------------------------+------------------------+--------+
| ID                                   | Name                   | Status |
+--------------------------------------+------------------------+--------+
| d94a3e03-8c3c-41a2-8937-b1adb7bdb00b | bm-deploy-kernel       | active |
| 7b7b62ff-f838-41fe-aab3-54723bf634d8 | bm-deploy-ramdisk      | active |
| 656f6458-2192-4223-a394-f716c155d41b | overcloud-full         | active |
| eb526b88-8d3c-4b3d-bcdf-31c50e793c7b | overcloud-full-initrd  | active |
| 0d66d2e3-2fdf-4d03-9fb5-71428d3a2bda | overcloud-full-vmlinuz | active |
+--------------------------------------+------------------------+--------+
```

Figure 3-15 Validate images

The list does not show the introspection PXE images (see Figure 3-15 on page 46). The director copies these files to `/httpboot` (see Figure 3-16):

```
(undercloud) [stack@red-director images]$ /httpboot
```

```
(undercloud) [stack@red-director ~]$ ls -l /httpboot/
total 486120
drwxr-xr-x. 2 ironic          ironic                63 Apr 23 09:44 2e4fd87f-7b1c-4c34-9e6a-d9d52a9a6ac3
drwxr-xr-x. 2 ironic          ironic                63 Apr 21 09:41 67dbbf29-cde2-44f3-b54d-bd06189631bc
-rwxr-xr-x. 1 root            root             6762800 Apr 10 02:10 agent.kernel
-rw-r--r--. 1 root            root           491008549 Apr 10 02:10 agent.ramdisk
-rw-r--r--. 1 ironic          ironic               758 Apr 10 01:00 boot.ipxe
-rw-r--r--. 1 ironic-inspector ironic-inspector     467 Apr 10 00:48 inspector.ipxe
drwxr-xr-x. 2 ironic          ironic                31 Apr 24 12:49 pxelinux.cfg
```

Figure 3-16 /httpboot contents

6. Configure and set up the name server and control plane (see Figure 3-17):

   `(undercloud) [stack@red-director ~]$ openstack subnet set --dns-nameserver 10.0.80.11 --dns-nameserver 10.0.80.12 ctlplane-subnet`

```
(undercloud) [stack@red-director ~]$ openstack subnet show ctlplane-subnet
+-------------------+-----------------------------------------------------------+
| Field             | Value                                                     |
+-------------------+-----------------------------------------------------------+
| allocation_pools  | 10.163.4.15-10.163.4.25                                   |
| cidr              | 10.163.4.0/24                                             |
| created_at        | 2020-04-10T06:06:10Z                                      |
| description       |                                                           |
| dns_nameservers   | 10.0.80.11, 10.0.80.12                                    |
| enable_dhcp       | True                                                      |
| gateway_ip        | 10.163.4.1                                                |
| host_routes       | destination='169.254.169.254/32', gateway='10.163.4.10'   |
| id                | 171e9d59-b3fa-45d7-858c-9eba52e4a092                      |
| ip_version        | 4                                                         |
| ipv6_address_mode | None                                                      |
| ipv6_ra_mode      | None                                                      |
| name              | ctlplane-subnet                                           |
| network_id        | ce975e72-8517-496e-855b-e1a32c1ae539                      |
| prefix_length     | None                                                      |
| project_id        | 44c48d0fec884a3fafe36ba3aa8f8a34                          |
| revision_number   | 1                                                         |
| segment_id        | None                                                      |
| service_types     |                                                           |
| subnetpool_id     | None                                                      |
| tags              |                                                           |
| updated_at        | 2020-04-10T07:14:22Z                                      |
+-------------------+-----------------------------------------------------------+
```

Figure 3-17 Control plane configuration

For a standard overcloud without network isolation, the name server is defined by using the undercloud's control plane subnet.

Complete the following steps to define name servers for the environment:

1. Source the stackrc file to enable the director's command line tools:

   `[stack@red-director ~]$ source ~/stackrc`

2. Set the name servers for the `ctlplane-subnet` subnet.
3. Configure the container registry.
4. Find the address of the local undercloud registry. The address uses the following format:

   `10.163.4.10:8787`

5. Create a template to upload the images to the local registry, and the environment file to refer to those images:

```
(undercloud) $ openstack overcloud container image prepare \
  --namespace=registry.access.redhat.com/rhosp13 \
  --push-destination=10.163.4.10:8787 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml \
  --output-images-file /home/stack/local_registry_images.yaml
```

   – `local_registry_images.yaml` includes container image information from the remote source. Use this file to pull the images from the Red Hat Container Registry (`registry.access.redhat.com`) to the undercloud.

   – `overcloud_images.yaml` contains the eventual image locations on the undercloud. You include this file with your deployment.

6. Pull the container images from registry.access.redhat.com to the undercloud:

```
(undercloud) $ sudo openstack overcloud container image upload \
  --config-file  /home/stack/local_registry_images.yaml \
  --verbose
```

The images are now stored on the undercloud's docker-distribution registry. To view the list of images on the undercloud's docker-distribution registry, run the following command (see Figure 3-18 on page 49):

```
(undercloud) $  curl http://10.163.4.10:8787/v2/_catalog | jq .repositories[]
```

```
(undercloud) [stack@red-director ~]$ curl http://10.163.4.10:8787/v2/_catalog | jq .repositories[]
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1502  100  1502    0     0   169k      0 --:--:-- --:--:-- --:--:--  183k
"rhosp13/openstack-aodh-api"
"rhosp13/openstack-aodh-evaluator"
"rhosp13/openstack-aodh-listener"
"rhosp13/openstack-aodh-notifier"
"rhosp13/openstack-ceilometer-central"
"rhosp13/openstack-ceilometer-compute"
"rhosp13/openstack-ceilometer-notification"
"rhosp13/openstack-cinder-api"
"rhosp13/openstack-cinder-scheduler"
"rhosp13/openstack-cinder-volume"
"rhosp13/openstack-cron"
"rhosp13/openstack-glance-api"
"rhosp13/openstack-gnocchi-api"
"rhosp13/openstack-gnocchi-metricd"
"rhosp13/openstack-gnocchi-statsd"
"rhosp13/openstack-haproxy"
"rhosp13/openstack-heat-api"
"rhosp13/openstack-heat-api-cfn"
"rhosp13/openstack-heat-engine"
"rhosp13/openstack-horizon"
"rhosp13/openstack-iscsid"
"rhosp13/openstack-keystone"
"rhosp13/openstack-mariadb"
"rhosp13/openstack-memcached"
"rhosp13/openstack-neutron-dhcp-agent"
"rhosp13/openstack-neutron-l3-agent"
"rhosp13/openstack-neutron-metadata-agent"
"rhosp13/openstack-neutron-openvswitch-agent"
"rhosp13/openstack-neutron-server"
"rhosp13/openstack-nova-api"
"rhosp13/openstack-nova-compute"
"rhosp13/openstack-nova-conductor"
"rhosp13/openstack-nova-consoleauth"
"rhosp13/openstack-nova-libvirt"
"rhosp13/openstack-nova-novncproxy"
"rhosp13/openstack-nova-placement-api"
"rhosp13/openstack-nova-scheduler"
"rhosp13/openstack-panko-api"
"rhosp13/openstack-rabbitmq"
"rhosp13/openstack-redis"
"rhosp13/openstack-swift-account"
```

Figure 3-18 Images in the docker-distribution registry

An `overcloud_images.yaml` environment file is now available that includes a list of your container image sources. Include this file with all future deployment operations.

3.2.5 Overcloud configuration with CLI tools

This process includes the following steps:

1. Create a node definition template and register blank nodes in the director.
2. Inspect hardware of all nodes.
3. Tag nodes into roles.
4. Define more node properties.

Node registration

The director requires a node definition template, which you create manually. This file (`instackenv.json`) uses the JSON format file, and contains the hardware and power management details for your nodes. For example, a template for registering two nodes can look like the examples that are shown in Figure 3-19 and Figure 3-20 on page 51.

```
(undercloud) [stack@red-director ~]$ cat instackenv.json
{
    "nodes": [

        {
            "mac":[
                "ac:1f:6b:c7:1c:4c"
            ],
            "name":"red-controller1.redbooks.info",
            "cpu":"2",
            "memory":"131097",
            "disk":"2000",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"root",
            "pm_password":"m6LDSq2ALE",
            "pm_addr":"10.162.62.54"
        }
    ]

}
```

Figure 3-19 Sample instackenv.json

```
(undercloud) [stack@red-director ~]$ cat instacknew.json
{
    "nodes": [
        {
            "mac":[
                "ac:1f:6b:c7:3a:5c"
            ],
            "name":"red-compute2.redbooks.com",
            "cpu":"2",
            "memory":"128378",
            "disk":"1000",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"root",
            "pm_password":"nYt8W46ktu",
            "pm_addr":"10.162.62.29"
        },
        {
            "mac":[
                "0c:c4:7a:39:85:0c"
            ],
            "name":"red-compute3.redbooks.info",
            "cpu":"2",
            "memory":"64149",
            "disk":"1000",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"root",
            "pm_password":"h25NBPgpPf",
            "pm_addr":"10.162.62.9"
        },
        {
            "mac":[
                "ac:1f:6b:c7:1a:48"
            ],
            "name":"red-compute5.redbooks.info",
            "cpu":"2",
            "memory":"63874",
            "disk":"1000",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"root",
            "pm_password":"r6m5yQQEEL",
            "pm_addr":"10.162.62.30"
        },
```

Figure 3-20 Sample instackenv.json continues

The template includes the following information:

- `Name`: The logical name for the node.
- `pm_type`: The power management driver to use. This example uses the IPMI driver (ipmi), which is the preferred driver for power management.
- `Memory` (optional): The amount of memory in MB.
- `cpu` (optional): The number of CPUs on the node.
- `pm_user` and `pm_password`: The IPMI user name and password. These attributes are optional for IPMI and Redfish, and are mandatory for iLO and iDRAC.
- `pm_addr`: The IP address of the IPMI device.

After creating the template, run the following commands to verify the formatting and syntax:

```
$ source ~/stackrc
(undercloud) $ openstack overcloud node import --validate-only ~/instackenv.json
```

Save the file to the stack user's home directory (`/home/stack/instackenv.json`); then, run the following command to import the template to the director:

```
(undercloud) $ openstack overcloud node import ~/instackenv.json
```

The template is imported and each node is registered from the template into the director.

After the node registration and configuration completes, view a list of these nodes in the CLI by running the following command:

```
(undercloud) $ openstack baremetal node list
```

3.2.6 Inspecting the hardware of nodes

A key process is to run a director and an introspection process on each node. This process causes each node to boot an introspection agent over PXE. This agent collects hardware data from the node and sends it back to the director. The director then stores this introspection data in the Red Hat OpenStack Object Storage (swift) service that is running on the director. The director uses hardware information for various purposes, such as profile tagging, benchmarking, and manual root disk assignment.

Run the following command to inspect the hardware attributes of each node:

```
(undercloud) $ openstack overcloud node introspect --all-manageable --provide
```

Consider the following points:

- The `--all-manageable` option introspects only nodes in a managed state. In this example, it is all of them.
- The `--provide` option resets all nodes to an available state after introspection.

Monitor the progress of the introspection by running the following command in a separate window:

```
(undercloud) $ sudo journalctl -l -u openstack-ironic-inspector -u
openstack-ironic-inspector-dnsmasq -u openstack-ironic-conductor -f
```

After the introspection completes, all nodes change to an available state.

To view introspection information about the node, run the following command:

```
(undercloud) $ openstack baremetal introspection data save <UUID> | jq .
```

Replace `<UUID>` with the UUID of the node that you want to retrieve introspection information for.

Running individual node introspection

To perform a single introspection on an available node, set the node to management mode and perform the introspection:

```
(undercloud) $ openstack baremetal node manage [NODE UUID]
(undercloud) $ openstack overcloud node introspect [NODE UUID] --provide
```

After the introspection completes, the nodes changes to an available state.

Performing node introspection after initial introspection

After an initial introspection, all nodes enter an available state because of the `--provide` option. To perform introspection on all nodes after the initial introspection, set all nodes to a manageable state and run the bulk introspection command:

```
(undercloud) $ for node in $(openstack baremetal node list --fields uuid -f value)
; do openstack baremetal node manage $node ; done
(undercloud) $ openstack overcloud node introspect --all-manageable --provide
```

After the introspection completes, all nodes change to an available state.

Performing network introspection for interface information

Network introspection retrieves link layer discovery protocol (LLDP) data from network switches. The following commands show a subset of LLDP information for all interfaces on a node, or full information for a particular node and interface. This can be useful for troubleshooting. The director enables LLDP data collection by default.

To get a list of interfaces on a node (see Figure 3-21), run the following command:

```
(undercloud) $ openstack baremetal introspection interface list [NODE UUID]
```

```
(undercloud) [stack@red-director ~]$ openstack baremetal introspection interface list 96c3c42e-16d1-49f6-bf2a-e05d56bd3f00
+-----------+-------------------+----------------------+-------------------+----------------+
| Interface | MAC Address       | Switch Port VLAN IDs | Switch Chassis ID | Switch Port ID |
+-----------+-------------------+----------------------+-------------------+----------------+
| eno1      | ac:1f:6b:58:ec:e4 | []                   | None              | None           |
| eno2      | ac:1f:6b:58:ec:e5 | []                   | None              | None           |
| eno3      | ac:1f:6b:58:ec:e6 | []                   | None              | None           |
| eno4      | ac:1f:6b:58:ec:e7 | []                   | None              | None           |
```

Figure 3-21 List node interfaces

To see interface data and switch port information (see Figure 3-22), run the following command:

```
(undercloud) $ openstack baremetal introspection interface show [NODE UUID]
[INTERFACE]
```

```
(undercloud) [stack@red-director ~]$ openstack baremetal introspection interface show 96c3c42e-16d1-49f6-bf2a-e05d56bd3f00 eno1
+--------------------------------------+--------------------------------------+
| Field                                | Value                                |
+--------------------------------------+--------------------------------------+
| interface                            | eno1                                 |
| mac                                  | ac:1f:6b:58:ec:e4                    |
| node_ident                           | 96c3c42e-16d1-49f6-bf2a-e05d56bd3f00 |
| switch_capabilities_enabled          | None                                 |
| switch_capabilities_support          | None                                 |
| switch_chassis_id                    | None                                 |
| switch_port_autonegotiation_enabled  | None                                 |
| switch_port_autonegotiation_support  | None                                 |
| switch_port_description              | None                                 |
| switch_port_id                       | None                                 |
| switch_port_link_aggregation_enabled | None                                 |
| switch_port_link_aggregation_id      | None                                 |
| switch_port_link_aggregation_support | None                                 |
| switch_port_management_vlan_id       | None                                 |
| switch_port_mau_type                 | None                                 |
| switch_port_mtu                      | None                                 |
| switch_port_physical_capabilities    | None                                 |
| switch_port_protocol_vlan_enabled    | None                                 |
| switch_port_protocol_vlan_ids        | None                                 |
| switch_port_protocol_vlan_support    | None                                 |
| switch_port_untagged_vlan_id         | None                                 |
| switch_port_vlan_ids                 | []                                   |
| switch_port_vlans                    | None                                 |
| switch_protocol_identities           | None                                 |
| switch_system_name                   | None                                 |
+--------------------------------------+--------------------------------------+
```

Figure 3-22 Interface and switch information

Hardware introspection details

The Bare Metal service hardware inspection extras (`inspection_extras`) is enabled by default to retrieve hardware details. You can use these hardware details to configure your overcloud.

For this implementation, the `numa_topology` collector is part of these hardware inspection extras and includes the following information for each NUMA node:

- RAM (in kilobytes).
- Physical CPU cores and their sibling threads.
- NICs associated with the NUMA node.

Run the **`openstack baremetal introspection data save _UUID_ | jq .numa_topology`** command to retrieve this information (see Example 3-2), with the UUID of the bare metal node:

```
(undercloud) [stack@red-director ~]$ openstack baremetal introspection data save
6fa72a36-0a0c-4249-8ac7-891603a6fa87 | jq .numa_topology
```

Example 3-2 NUMA topology information

```
{
  "cpus": [
    {
      "cpu": 1,
      "thread_siblings": [
        1,
        17
      ],
      "numa_node": 0
    },
    {
      "cpu": 2,
      "thread_siblings": [
        10,
        26
      ],
      "numa_node": 1
    },
    {
      "cpu": 0,
      "thread_siblings": [
        0,
        16
      ],
      "numa_node": 0
    },
    {
      "cpu": 5,
      "thread_siblings": [
        13,
        29
      ],
      "numa_node": 1
    },
    {
      "cpu": 7,
      "thread_siblings": [
```

```
      15,
      31
    ],
    "numa_node": 1
  },
  {
    "cpu": 7,
    "thread_siblings": [
      7,
      23
    ],
    "numa_node": 0
  },
  {
    "cpu": 1,
    "thread_siblings": [
      9,
      25
    ],
    "numa_node": 1
  },
  {
    "cpu": 6,
    "thread_siblings": [
      6,
      22
    ],
    "numa_node": 0
  },
  {
    "cpu": 3,
    "thread_siblings": [
      11,
      27
  ],
    "numa_node": 1
  },
  {
    "cpu": 5,
    "thread_siblings": [
      5,
      21
    ],
    "numa_node": 0
  },
  {
    "cpu": 4,
    "thread_siblings": [
      12,
      28
    ],
    "numa_node": 1
  },
  {
    "cpu": 4,
```

```
      "thread_siblings": [
        4,
        20
      ],
      "numa_node": 0
    },
    {
      "cpu": 0,
      "thread_siblings": [
        8,
        24
      ],
      "numa_node": 1
   },
    {
      "cpu": 6,
      "thread_siblings": [
        14,
        30
      ],
      "numa_node": 1
    },
    {
      "cpu": 3,
      "thread_siblings": [
        3,
        19
      ],
      "numa_node": 0
    },
    {
      "cpu": 2,
      "thread_siblings": [
        2,
        18
      ],
      "numa_node": 0
    }
  ],
  "ram": [
    {
      "size_kb": 32585276,
      "numa_node": 0
    },
    {
      "size_kb": 32993960,
      "numa_node": 1
   }
  ],
  "nics": [
    {
      "name": "eno1",
      "numa_node": 0
    },
    {
```

```
      "name": "eno3",
      "numa_node": 0
    },
    {
      "name": "eno2",
      "numa_node": 0
    },
    {
      "name": "eno4",
      "numa_node": 0
    }
  ]
}
```

3.2.7 Tagging nodes into profiles

After registering and inspecting the hardware of each node, you tag them into specific profiles. These profile tags match your nodes to flavors, and in turn the flavors are assigned to a deployment role. Table 3-1 lists the relationship across roles, flavors, profiles, and nodes for controller nodes.

Table 3-1 Relationship types and descriptions

Type	Description
Role	The Controller role defines how to configure controller nodes.
Flavor	The control flavor defines the hardware profile for nodes to use as controllers. You assign this flavor to the Controller role so the director can decide which nodes to use.
Profile	The control profile is a tag you apply to the control flavor. This defines the nodes that belong to the flavor.
Node	You also apply the control profile tag to individual nodes, which groups them to the control flavor and, as a result, the director configures them by using the Controller role.

Default profile flavors compute, control, swift-storage, ceph-storage, and block-storage are created during undercloud installation and are usable without modification in most environments.

To tag a node into a specific profile, add a profile option to the properties and capabilities parameter for each node. For example, to tag your nodes to use controller and compute profiles respectively, run the following commands (see Example 3-3 on page 58):

```
(undercloud) $ openstack baremetal node set --property
capabilities='profile:compute,boot_option:local'
96c3c42e-16d1-49f6-bf2a-e05d56bd3f00

(undercloud) $ openstack baremetal node set --property
capabilities='profile:control,boot_option:local'
6fa72a36-0a0c-4249-8ac7-891603a6fa87

(undercloud) [stack@red-director ~]$ openstack overcloud profiles list
```

Example 3-3 Tagging nodes with profiles

```
+--------------------------------------+------------------------------+----------------+---------------
-+------------------+
| Node UUID                            | Node Name                    | Provision State | Current Profile
| Possible Profiles |
+--------------------------------------+------------------------------+----------------+---------------
-+------------------+
| 6fa72a36-0a0c-4249-8ac7-891603a6fa87 | red-compute1.redbooks.info   | active          | control
|                  |
| 96c3c42e-16d1-49f6-bf2a-e05d56bd3f00 | blue-controller2.redbooks.info | active          | compute
|                  |
| e564423f-354b-4c8e-af4f-ae328a91b87f | red-compute7.redbooks.info   | active          | control
|                  |
| 8a945039-c44f-4e10-b759-485bc5499a40 | red-compute2.redbooks.com    | active          | compute
|                  |
| 589df7ce-1b8b-4dc4-8d68-5e8c67e9b04b | red-compute5.redbooks.info   | active          | compute
|                  |
| a457f0d6-abe3-4ae5-8b19-8669c86bcb45 | red-compute6.redbooks.info   | active          | control
|                  |
| 256d9a74-7267-43c9-b4a6-fc9f918fc751 | red-controller1.redbooks.info | active          | compute
|                  |
+--------------------------------------+------------------------------+----------------+---------
--------+------------------+
```

The addition of the `profile:compute` and `profile:control` options tag the two nodes into each respective profile.

These commands also set the `boot_option:local` parameter, which defines how each node boots. Depending on your hardware, you might also need to add the `boot_mode` parameter to UEFI so that nodes boot by using UEFI instead of the default BIOS mode.

3.2.8 Creating the overcloud with the CLI tools

The final stage in creating your Red Hat OpenStack environment is to run the **`openstack overcloud deploy`** command to create it. Before running this command, familiarize yourself with key options and how to include custom environment files.

Setting overcloud parameters

Table 3-2 lists the other parameters when the **`openstack overcloud deploy`** command is used.

Table 3-2 Deployment parameters

Parameter	Description
--templates [TEMPLATES]	The directory that contains the Heat templates to deploy. If blank, the command uses the default template location at: /usr/share/openstack-tripleo-heat-templates/.
--stack STACK	The name of the stack to create or update.
-t [TIMEOUT], --timeout [TIMEOUT]	Deployment timeout in minutes.
--libvirt-type [LIBVIRT_TYPE]	Virtualization type to use for hypervisors.

Parameter	Description
--ntp-server [NTP_SERVER]	Network Time Protocol (NTP) server to use to synchronize time. You can also specify multiple NTP servers in a comma-separated list, for example: `--ntp-server 0.centos.pool.org,1.centos.pool.org` For a HA cluster deployment, your controllers must consistently refer to the same time source. A typical environment might have a designated NTP time source with established practices.
--no-proxy [NO_PROXY]	Defines custom values for the environment variable no_proxy, which excludes specific hostnames from proxy communication.
--overcloud-ssh-user OVERCLOUD_SSH_USER	Defines the SSH user to access the overcloud nodes. Normally, SSH access occurs through the heat-admin user.
-e [EXTRA HEAT TEMPLATE], --extra-template [EXTRA HEAT TEMPLATE]	Extra environment files to pass to the overcloud deployment. Can be specified more than once. The order of environment files passed to the **`openstack overcloud deploy`** command is important. For example, parameters from each sequential environment file override the same parameters from earlier environment files.
--environment-directory	The directory that contains environment files to include in deployment. The command processes these environment files in numerical, then alphabetical order.
--validation-errors-nonfatal	The overcloud creation process performs a set of pre-deployment checks. This option exits if any non-fatal errors occur from the pre-deployment checks. It is advisable to use this option as any errors can cause your deployment to fail.
--validation-warnings-fatal	The overcloud creation process performs a set of pre-deployment checks. This option exits if any non-critical warnings occur from the pre-deployment checks.
--dry-run	Performs validation check on the overcloud but does not create the overcloud.
--skip-postconfig	Skip the overcloud post-deployment configuration.
--force-postconfig	Force the overcloud post-deployment configuration.
--skip-deploy-identifier	Skip generation of a unique identifier for the DeployIdentifier parameter. The software configuration deployment steps trigger only if the configuration is changed. Use this option with caution and only if you are confident you do not need to run the software configuration, such as scaling out certain roles.

Parameter	Description
--answers-file ANSWERS_FILE	Path to a YAML file with arguments and parameters.
--rhel-reg	Register overcloud nodes to the Customer Portal or Red Hat Satellite V6.
--reg-method	Registration method to use for the overcloud nodes. Red Hat Satellite V6 or Red Hat Satellite V5, portal for Customer Portal.
--reg-org [REG_ORG]	Organization to use for registration.
--reg-force	Register the system even if it is already registered
--reg-sat-url [REG_SAT_URL]	The base URL of the Red Hat Satellite server to register overcloud nodes. Use the Red Hat Satellite's HTTP URL and not the HTTPS URL for this parameter. For example, use `http://satellite.example.com` and *not* https://satellite.example.com. The overcloud creation process uses this URL to determine whether the server is a Red Hat Satellite 5 or Red Hat Satellite V6 server. If a Red Hat Satellite V6 server is used, the overcloud obtains the katello-ca-consumer-latest.noarch.rpm file, registers with subscription-manager, and installs katello-agent. If a Red Hat Satellite V5 server is used, the overcloud obtains the RHN-ORG-TRUSTED-SSL-CERT file and registers with rhnreg_ks.
--reg-activation-key [REG_ACTIVATION_KEY]	Activation key to use for registration.

Some command line parameters are outdated or deprecated in favor of using Heat template parameters, which you include in the `parameter_defaults` section on an environment file. Table 3-3 maps deprecated parameters to their Heat Template equivalents.

Table 3-3 Mapping deprecated CLI parameters to heat template parameters

Parameter	Description	Heat template parameter
--control-scale	The number of Controller nodes to scale out.	ControllerCount
--compute-scale	The number of Compute nodes to scale out.	ComputeCount
--ceph-storage-scale	The number of Ceph Storage nodes to scale out.	CephStorageCount
--block-storage-scale	The number of Cinder nodes to scale out.	BlockStorageCount
--swift-storage-scale	The number of Swift nodes to scale out.	ObjectStorageCount

Parameter	Description	Heat template parameter
--control-flavor	The flavor to use for Controller nodes.	OvercloudControllerFlavor
--compute-flavor	The flavor to use for Compute nodes.	OvercloudComputeFlavor
--ceph-storage-flavor	The flavor to use for Ceph Storage nodes.	OvercloudCephStorageFlavor
--block-storage-flavor	The flavor to use for Cinder nodes.	OvercloudBlockStorageFlavor
--swift-storage-flavor	The flavor to use for Swift storage nodes.	OvercloudSwiftStorageFlavor
--neutron-flat-networks	Defines the flat networks to configure in neutron plug-ins. Defaults to “datacenter” to permit external network creation.	NeutronFlatNetworks
--neutron-physical-bridge	An Open vSwitch bridge to create on each hypervisor. This defaults to *br-ex*. Typically, this must not need to be changed.	HypervisorNeutronPhysicalBridge
--neutron-bridge-mappings	The logical to physical bridge mappings to use. Defaults to mapping the external bridge on hosts (br-ex) to a physical name (datacenter). You use this for the default floating network.	NeutronBridgeMappings
--neutron-public-interface	Defines the interface to bridge onto br-ex for network nodes.	NeutronPublicInterface
--neutron-network-type	The tenant network type for Neutron.	NeutronNetworkType
--neutron-tunnel-types	The tunnel types for the Neutron tenant network. To specify multiple values, use a comma separated string.	NeutronTunnelTypes
--neutron-tunnel-id-ranges	Ranges of GRE tunnel IDs to make available for tenant network allocation.	NeutronTunnelIdRanges
--neutron-vni-ranges	Ranges of VXLAN VNI IDs to make available for tenant network allocation.	NeutronVniRanges
--neutron-network-vlan-ranges	The Neutron ML2 and Open vSwitch VLAN mapping range to support. Defaults to permitting any VLAN on the datacenter physical network.	NeutronNetworkVLANRanges

Parameter	Description	Heat template parameter
--neutron-mechanism-drivers	The mechanism drivers for the neutron tenant network. Defaults to “openvswitch”. To specify multiple values, use a comma-separated string.	NeutronMechanismDrivers
--neutron-disable-tunneling	Disables tunneling in case you aim to use a VLAN segmented network or flat network with Neutron.	No parameter mapping.
--validation-errors-fatal	The overcloud creation process performs a set of pre-deployment checks. This option exits if any fatal errors occur from the pre-deployment checks. It is advisable to use this option as any errors can cause your deployment to fail.	No parameter mapping.

3.2.9 Environment files in overcloud creation

The **-e** option includes an environment file to customize your overcloud. You can include as many environment files as necessary. However, the order of the environment files is important because the parameters and resources defined in subsequent environment files take precedence. The following example shows a list of the environment file order:

- The number of nodes per each role and their flavors. It is vital to include this information for overcloud creation.
- The location of the container images for containerized Red Hat OpenStack services.
- Any network isolation files, starting with the initialization file (`environments/network-isolation.yaml`) from the heat template collection; then, your custom NIC configuration file, and finally, any other network configurations.
- Any external load balancing environment files if you use an external load balancer.
- Any storage environment files, such as Ceph Storage, NFS, and iSCSI.
- Any environment files for Red Hat Content Delivery Network or Red Hat Satellite registration.
- Any other custom environment files.

Any environment files that are added to the overcloud by using the **-e** option become part of your overcloud’s stack definition. The following command is an example of how to start the overcloud creation with custom environment files included:

```
(undercloud) $ openstack overcloud deploy --verbose --templates \
  -e /home/stack/templates/node-info.yaml \
  -e /home/stack/templates/service_net.yaml \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/ips-from-pool-all.yaml \
  -e /home/stack/templates/network-environment.yaml \
  -r /home/stack/templates/roles_data.yaml \
  --ntp-server 10.0.77.54 &
```

This command contains the following **--templates** option creates the overcloud by using the Heat template collection in `/usr/share/openstack-tripleo-heat-templates` as a foundation:

```
-e /home/stack/templates/node-info.yaml
```

An environment file is added to define how many nodes and which flavors to use for each role. For example, the following `parameter_defaults` are included:

```
OvercloudControllerFlavor: control
OvercloudComputeFlavor: compute
ControllerCount: 3
ComputeCount: 4
```

Where:

```
-e /home/stack/templates/overcloud_images.yaml
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml
```

To add an environment file to initialize network isolation in the overcloud deployment:

```
-e /home/stack/templates/network-environment.yaml
```

To add an environment file to customize network isolation:

```
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml
```

To add an environment file to enable Ceph Storage services:

```
-e /home/stack/templates/ceph-custom-config.yaml
```

To add an environment file to customize our Ceph Storage configuration:

```
-e /home/stack/inject-trust-anchor-hiera.yaml
```

To add an environment file to install a custom certificate in the undercloud:

```
--ntp-server pool.ntp.org
```

Use an NTP server for time synchronization. This is required for keeping the Controller node cluster in synchronization:

```
-r /home/stack/templates/roles_data.yaml
```

If you aim to later modify the overcloud configuration, you must complete the following steps:

1. Modify parameters in the custom environment files and Heat templates.
2. Run the Red Hat OpenStack overcloud deploy command again with the same environment files.

Note: Do not edit the overcloud configuration directly because such manual configuration is overridden by the director's configuration when the overcloud stack is updated by using the director.

Including an environment file directory

You can add an entire directory that contains environment files by using the `--environment-directory` option. The deployment command processes the environment files in this directory in numerical, then alphabetical order. If this method is used, it is recommended to use file names with a numerical prefix to order how they are processed:

```
(undercloud) $ ls -l ~/templates
compute.yaml
controller.yaml
```

```
ips-from-pool-all.yaml
network-environment.yaml
node-info.yaml
overcloud_images.yaml
roles_data.yaml
service_net.yaml
```

Run the following deployment command to include the directory:

```
(undercloud) $ openstack overcloud deploy --verbose --templates \
  -e /home/stack/templates/node-info.yaml \
  -e /home/stack/templates/service_net.yaml \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/ips-from-pool-all.yaml \
  -e /home/stack/templates/network-environment.yaml \
  -r /home/stack/templates/roles_data.yaml \
  --ntp-server 10.0.77.54 &
```

3.2.10 Monitoring the overcloud creation

The overcloud creation process begins and the director provisions your nodes. This process takes some time to complete. To view the status of the overcloud creation, open a separate terminal as the stack user and run:

```
(undercloud) $ source ~/stackrc
(undercloud) $ openstack stack list --nested
```

The **`openstack stack list --nested`** command shows the current stage of the overcloud creation.

3.2.11 Viewing the overcloud deployment output

After a successful overcloud deployment, the shell returns the following information that you can use to access your overcloud:

```
Stack overcloud CREATE_COMPLETEStarted Mistral Workflow
tripleo.deployment.v1.get_horizon_url. Execution ID:
1fc01bd4-327b-4b8c-8509-72e2c3be0ac1
Overcloud Endpoint: http://10.163.4.20:5000/
Overcloud Horizon Dashboard URL: http://10.163.4.20:80/dashboard
Overcloud rc file: /home/stack/overcloudrc
Overcloud Deployed
```

3.2.12 Accessing the overcloud

The director generates a script to configure and help authenticate interactions with your overcloud from the director host. The director saves this file, `overcloudrc`, in your stack user's home director. Run the following command to use this file:

```
(undercloud) $ source ~/overcloudrc
```

The necessary environment variables are loaded to interact with your overcloud from the director host's CLI. The command prompt changes to indicate this:

```
(overcloud) $
```

To return to interacting with the director's host, run the following command:

```
(overcloud) $ source ~/stackrc
(undercloud) $
```

Each node in the overcloud also contains a user called heat-admin. The stack user has SSH access to this user on each node. To access a node over SSH, find the IP address of the wanted node (see Example 3-4):

```
(undercloud) $ openstack server list
```

Example 3-4 Server list

```
+--------------------------------------+------------------------+--------+---------------------
+---------------+---------+
| ID                                   | Name                   | Status | Networks
| Image          | Flavor  |
+--------------------------------------+------------------------+--------+---------------------
+---------------+---------+
| 1f3535a6-92bd-42dc-9a5f-c2d9c1fa38e2 | overcloud-compute-0    | ACTIVE | ctlplane=10.163.4.53
| overcloud-full | compute |
| 27acbc8c-5e13-4430-8259-8fc654478e7f | overcloud-compute-3    | ACTIVE | ctlplane=10.163.4.56
| overcloud-full | compute |
| 0e4c9b04-4927-4c0f-abcb-ab4b4f4cf021 | overcloud-controller-0 | ACTIVE | ctlplane=10.163.4.50
| overcloud-full | control |
| 62902558-840c-43d5-93a5-d9756f33a00f | overcloud-compute-1    | ACTIVE | ctlplane=10.163.4.54
| overcloud-full | compute |
| 9a49380c-3843-4a20-9d02-143d99ca9104 | overcloud-controller-2 | ACTIVE | ctlplane=10.163.4.52
| overcloud-full | control |
| fbcc8ac4-67cf-4f82-8342-bad27b4df520 | overcloud-controller-1 | ACTIVE | ctlplane=10.163.4.51
| overcloud-full | control |
| d87f6f9b-b8b8-4346-878b-eba2acbb21d3 | overcloud-compute-2    | ACTIVE | ctlplane=10.163.4.55
| overcloud-full | compute |
+--------------------------------------+------------------------+--------+---------------------
+---------------+---------+
```

Then, connect to the node by using the heat-admin user and the node's IP address:

```
(undercloud) [stack@red-director ~]$ ssh heat-admin@10.163.4.50
Last login: Fri May  8 07:31:50 2020 from 10.163.4.10
[heat-admin@overcloud-controller-0 ~]$
```

3.3 Red Hat OpenShift

Red Hat OpenShift Container Platform is a container solution that offers developers and IT organizations a cloud application platform for deploying modern applications on secure, scalable resources with minimal configuration and management overhead. Red Hat OpenShift provides the suitable integrated platform with all components for hosting microservices that are based applications on containers and operating them with ease.

Built on Red Hat Enterprise Linux, a container run time, and Google Kubernetes an orchestrator, Red Hat OpenShift provides a secure and scalable multi-tenant platform for today's enterprise-class applications and libraries. Red Hat OpenShift Container Platform brings a scalable container solution to customer data centers and public cloud providers, which enables organizations to implement an individualized microservices solution that meets security, privacy, compliance, and governance requirements.

3.3.1 High-level deployment scenario

Our basic installation architecture recommendation is shown in Figure 3-23.

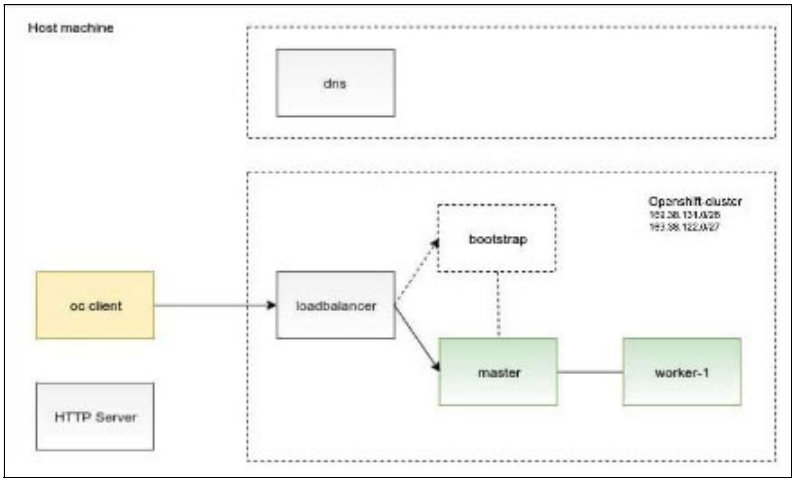


Figure 3-23 Red Hat OpenShift high level architecture for deployment

The Red Hat OpenShift solution provides standard, included components for Internal API authentication, clustered key-value data store, scheduling, replication, routing, services, and capability to attach persistent storage, as shown in Figure 3-24 on page 67.

The architecture provides three masters for Red Hat OpenShift system management capabilities, three infrastructure nodes for Red Hat OpenShift routers and container image registries, and eight Red Hat OpenShift application nodes that host the application pods.

The three masters in architecture provide HA for master components. HA for application requests is accounted for through redundant HAProxy routers.

Red Hat OpenShift can be deployed on bare metal servers and virtual machines (VMs). For private cloud PaaS services, we recommend deploying Red Hat OpenShift on Red Hat OpenStack.

Red Hat OpenShift add the following features to the container platform:

- Source code management, builds, and deployments for developers.
- Managing and promoting images at scale as they flow through your system.
- Application management at scale.
- Team and user tracking for organizing a large developer organization.
- Networking infrastructure that supports the cluster.
- Service Mesh for Traffic management, Observability, and Policy enforcement.
- Source to Image for containerization of application.

Figure 3-24 shows the high level deployment architecture that is deployed on IBM Cloud bare metal machines and orchestrated IBM Cloud network.

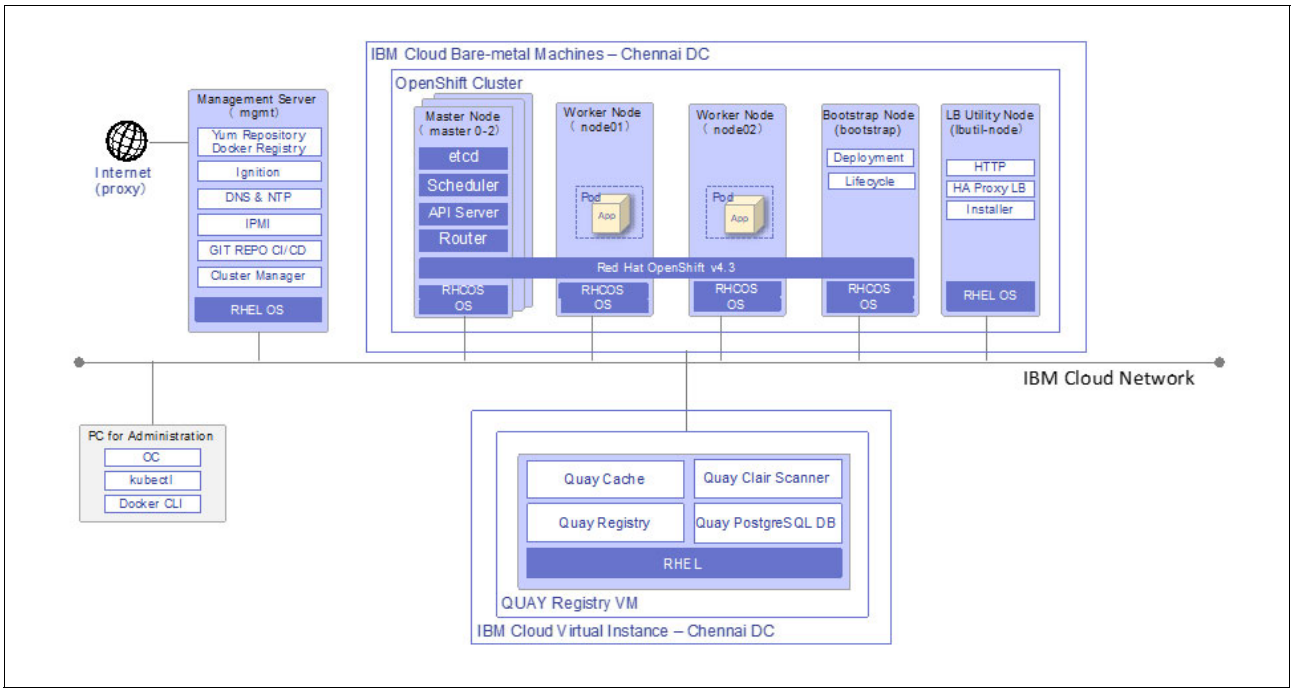


Figure 3-24 Proof of concept deployment architecture of Red Hat OpenShift on IBM Cloud bare metal

3.3.2 System prerequisites components

Table 3-4 lists the components that are required to deploy Red Hat OpenShift Container Platform, as shown in Figure 3-24 and discussed in 3.3.1, “High-level deployment scenario” on page 66.

Table 3-4 Components of the deployment in the cluster

Components	Instance number	Operating system	Purpose
DNS	Existing	IBM Cloud Platform DNS	Using IBM Cloud DNS capability for name resolution.
Load Balancer, HTTP server	1	Red Hat Enterprise Linux V7	HAProxy load balancer. Facilitates bootstrapping, and balances the load between the master nodes and between the ingress router pods.
Bootstrap	1	Red Hat Enterprise Linux CoreOS	The bootstrap machine. Used once to initialize the Red Hat OpenShift cluster.
Master	3	Red Hat Enterprise Linux CoreOS	Red Hat OpenShift Master Node.
Worker	2	Red Hat Enterprise Linux CoreOS	Red Hat OpenShift Worker and Application Node.

Because IBM Cloud bare metal Servers and IBM Cloud network play a crucial role in deployment, we must ensure provisioning of bare metal servers before starting deployment and test the DNS resolutions for each machine. We also must order servers for Bootstrap, Master, and workers without an operating system, wherein we can upload Red Hat Enterprise Linux CoreOS .iso images on the server and perform a boot server with the uploaded Red Hat Enterprise Linux CoreOS .iso image.

You need a valid Red Hat OpenShift subscription to download the Red Hat Enterprise Linux CoreOS from the Red Hat OpenShift Cluster Manager portal.

3.3.3 System hardware requirements

Table 3-5 lists the server requirements for bare metal Red Hat OpenShift deployment.

Table 3-5 Hardware requirements of machines

Server	Role	Hardware			IP address		Virtual/ physical	Operating system
		CPU cores	RAM (GB)	Storage (GB) - SSD	Public N/W, Private N/W, DNS, NTP	IPMI with admin access		
oc client	Bastion	4	16	50	Yes	N/A	Virtual	Red Hat Enterprise Linux V7.6 or later
bootstrap-0	Red Hat OpenShift - Bootstrap	4	32	120	Yes	Yes	Physical	No OS (Red Hat Enterprise Linux CoreOS will be pushed by IGN file)
lb-utility-0	HA proxy LB, HTTP Server, Internal LB/ External LB	4	32	120	Yes	N/A	Virtual	Red Hat Enterprise Linux V7.6 or later
master-0	Red Hat OpenShift - Master	4	32	250	Yes	Yes	Physical	No OS (Red Hat Enterprise Linux CoreOS will be pushed by IGN file)
master-1	Red Hat OpenShift - Master	4	32	250	Yes	Yes	Physical	No OS (Red Hat Enterprise Linux CoreOS will be pushed by IGN file)

master-2	Red Hat OpenShift - Master	4	32	250	Yes	Yes	Physical	No OS (Red Hat Enterprise Linux CoreOS will be pushed by IGN file)
appnode-0	Red Hat OpenShift - Worker	4	32	120	Yes	Yes	Physical	No OS (Red Hat Enterprise Linux CoreOS will be pushed by IGN file)
appnode-1	Red Hat OpenShift - Worker	4	32	120	Yes	Yes	Physical	No OS (Red Hat Enterprise Linux CoreOS will be pushed by IGN file)

3.3.4 Planning and prerequisites

This section shows the planning and prerequisites for the implementation.

Network planning sheet

To deploy Red Hat OpenShift Container Platform on IBM Cloud bare metal, you must have the network details of the same segment, as listed in Table 3-6:

- IPMI Access to provisioned machines with admin privileges.
- Assigned public network (same IP and subnet range preferred).
- Assigned DNS
- NTP server IP and name
- Gateway details

Note: For security reasons, we masked the IP addresses. It is recommended to use same range of IPs and subnet.

Table 3-6 Cluster network planning

Device name	IPMI/26	Public IP	Gateway	Netmask	DNS IP
occlient.redbooks.info	N/A	169.xxx.xxx.xxx	169.xxx.xxx.xxx	255.xxx.xxx.xxx	67.xxx.xxx.xxx
ocp4-lbutility-0.ocp4.redbooks.info	10.xxx.xxx.xxx	169.xxx.xxx.xxx	169.xxx.xxx.xxx	255.xxx.xxx.xxx	67.xxx.xxx.xxx
ocp4-appnode-2.ocp4.redbooks.info	10.xxx.xxx.xxx	169.xxx.xxx.xxx	169.xxx.xxx.xxx	255.xxx.xxx.xxx	67.xxx.xxx.xxx
ocp4-appnode-3.ocp4.redbooks.info	10.xxx.xxx.xxx	169.xxx.xxx.xxx	169.xxx.xxx.xxx	255.xxx.xxx.xxx	67.xxx.xxx.xxx

Device name	IPMI/26	Public IP	Gateway	Netmask	DNS IP
ocp4-master-0.redbooks.info	10.xxx.xxx.xxx	169.xxx.xxx.xxx	169.xxx.xxx.xxx	255.xxx.xxx.xxx	67.xxx.xxx.xxx
ocp4-master-2.redbooks.info	10.xxx.xxx.xxx	169.xxx.xxx.xxx	169.xxx.xxx.xxx	255.xxx.xxx.xxx	67.xxx.xxx.xxx
ocp4-master-2.redbooks.info	10.xxx.xxx.xxx	169.xxx.xxx.xxx	169.xxx.xxx.xxx	255.xxx.xxx.xxx	67.xxx.xxx.xxx
ocp4-master-1.redbooks.info	10.xxx.xxx.xxx	169.xxx.xxx.xxx	169.xxx.xxx.xxx	255.xxx.xxx.xxx	67.xxx.xxx.xxx
ocp4-bootstrap-0.ocp4.redbooks.info	10.xxx.xxx.xxx	169.xxx.xxx.xxx	169.xxx.xxx.xxx	255.xxx.xxx.xxx	67.xxx.xxx.xxx
ocp4-appnode-1.ocp4.redbooks.info	10.xxx.xxx.xxx	169.xxx.xxx.xxx	169.xxx.xxx.xxx	255.xxx.xxx.xxx	67.xxx.xxx.xxx

Configuring the DNS entries

Table 3-7 lists the DNS records that are required for a Red Hat OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, `<cluster_name>` is the cluster name and `<base_domain>` is the cluster base domain that you specify in the `install-config.yaml` file. A complete DNS record uses the following form:

`<component>.<cluster_name>.<base_domain>`

Table 3-7 DNS records for the cluster

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>.	This DNS A/AAAA or CNAME record must point to the load balancer for the control plane machines. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>.	This DNS A/AAAA or CNAME record must point to the load balancer for the control plane machines. This record must be resolvable from all the nodes within the cluster. The API server must resolve the worker nodes by the host names that are recorded in Kubernetes. If it cannot resolve the node names, proxied API calls can fail, and you cannot retrieve logs from Pods.
Routes	*.apps.<cluster_name>.<base_domain>.	A wildcard DNS A/AAAA or CNAME record that points to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

etcd	etcd-<index>.<cluster_name>.<base_domain>.	Red Hat OpenShift Container Platform requires DNS A/AAAA records for each etcd instance to point to the control plane machines that host the instances. The etcd instances are differentiated by <index> values, which start with 0 and end with n-1, where n is the number of control plane machines in the cluster. The DNS record must resolve to an unicast IPv4 address for the control plane machine, and the records must be resolvable from all the nodes in the cluster.

	_etcd-server-ssl._tcp.<cluster_name>.<base_domain>.	For each control plane machine, Red Hat OpenShift Container Platform also requires a SRV DNS record for etcd server on that machine with priority 0, weight 10 and port 2380. A cluster that uses three control plane machines requires the following records:
		`# _service._proto.name. TTL class SRV priority weight port target.`
		`_etcd-server-ssl._tcp.<cluster_name>.<base_domain>. 86400 IN SRV 0 10 2380 etcd-0.<cluster_name>.<base_domain>`
		`_etcd-server-ssl._tcp.<cluster_name>.<base_domain>. 86400 IN SRV 0 10 2380 etcd-1.<cluster_name>.<base_domain>`
		`_etcd-server-ssl._tcp.<cluster_name>.<base_domain>. 86400 IN SRV 0 10 2380 etcd-2.<cluster_name>.<base_domain>`

Figure 3-25, Figure 3-26 on page 72, and Figure 3-27 on page 72 show snippets of changes that were implemented in the IBM Cloud by editing the DNS lookup records.

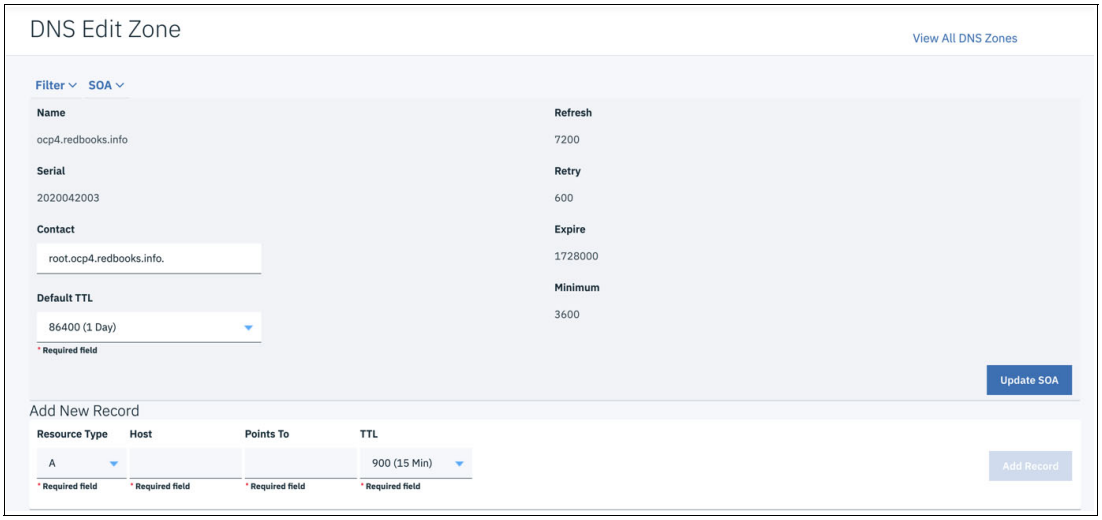


Figure 3-25 DNS edit zone IBM Cloud

cluster4	SRV	900 (15 Min)	**Service:** _etcd-server-ssl **Protocol:** _tcp **Priority:** 0 **Weight:** 10 **P**	etcd-0.cluster4
cluster4	SRV	900 (15 Min)	**Service:** _etcd-server-ssl **Protocol:** _tcp **Priority:** 0 **Weight:** 10 **P**	etcd-1.cluster4
cluster4	SRV	900 (15 Min)	**Service:** _etcd-server-ssl **Protocol:** _tcp **Priority:** 0 **Weight:** 10 **P**	etcd-2.cluster4
	SRV	900 (15 Min)	**Service:** _cluster4 **Protocol:** _tcp **Priority:** 0 **Weight:** 10 **Port:** 23	etcd-1.cluster4
	SRV	900 (15 Min)	**Service:** _cluster4 **Protocol:** _tcp **Priority:** 0 **Weight:** 10 **Port:** 23	etcd-2.cluster4
	SRV	900 (15 Min)	**Service:** _cluster4 **Protocol:** _tcp **Priority:** 0 **Weight:** 10 **Port:** 23	etcd-0.cluster4
www	A	86400 (1 Day)		10.162.59.60
webmail	A	86400 (1 Day)		10.162.59.60
quay	A	900 (15 Min)		169.38.131.72
ocp4-master-2	A	900 (15 Min)		169.38.131.93
ocp4-master-1	A	900 (15 Min)		169.38.131.73
ocp4-master-0	A	900 (15 Min)		169.38.131.82
ocp4-lbutility-0	A	900 (15 Min)		169.38.131.75
ocp4-bootstrap-0	A	900 (15 Min)		169.38.122.221
ocp4-appnode-2	A	900 (15 Min)		169.38.131.83
ocp4-appnode-1	A	900 (15 Min)		169.38.131.71
ocp4-appnode-0	A	900 (15 Min)		169.38.131.78
occlient	A	900 (15 Min)		169.38.131.69
mail	A	86400 (1 Day)		10.162.59.60
ftp	A	86400 (1 Day)		10.162.59.60
etcd-2.cluster4	A	900 (15 Min)		169.38.131.93
etcd-1.cluster4	A	900 (15 Min)		169.38.131.73
etcd-0.cluster4	A	900 (15 Min)		169.38.131.82
ems-test	A	900 (15 Min)		169.38.131.88

Figure 3-26 DNS edit zone IBM Cloud continues

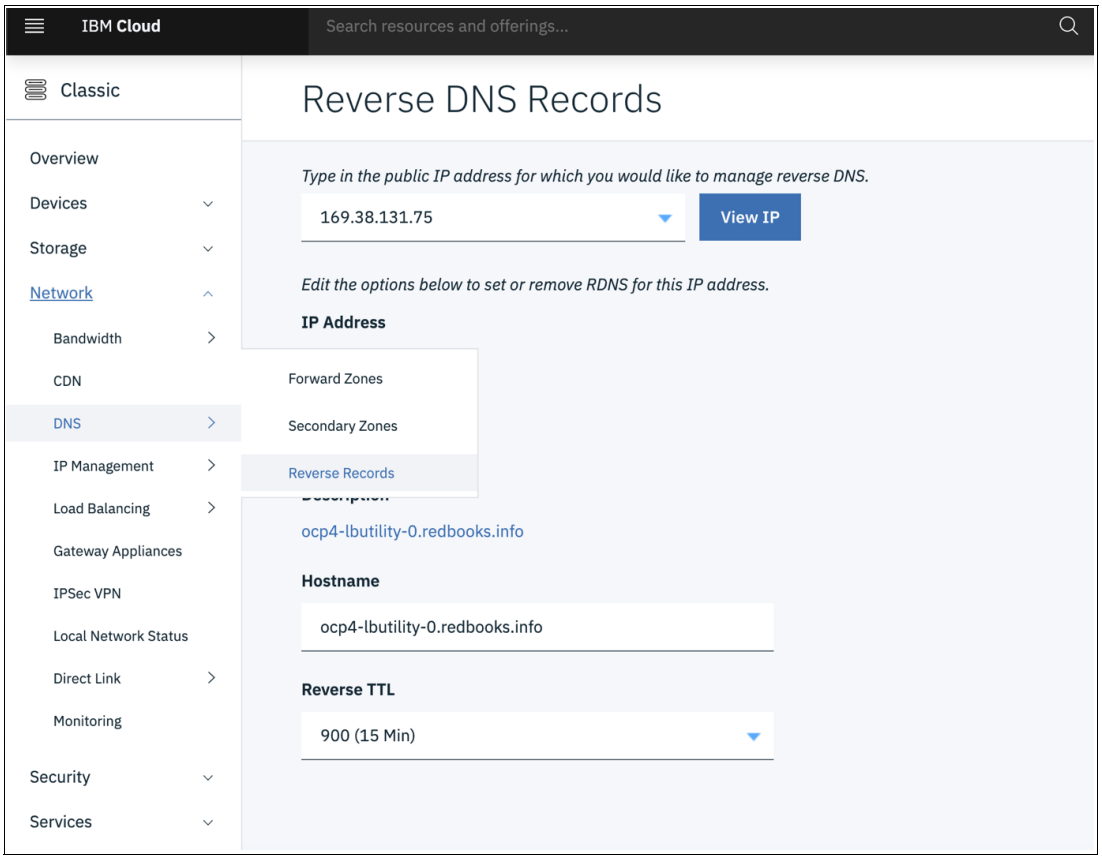


Figure 3-27 Updating and changing reverse records

Configuring the load balancers

This activity is to create the load balancer for Red Hat OpenShift deployment (Loadbalancer A and Loadbalancer B can be on the same hardware or software dispatcher). For more information about a detailed procedure of the deployment, see “Configuring the load balancers” on page 73.

This configuration adds the following load balancer entries to HA-Proxy:

```
$ yum install haproxy -y
```

3.3.5 Installing Red Hat OpenShift

The following high-level steps are used for installing and configuring Red Hat OpenShift Cluster on IBM Cloud bare metal machines:

1. The bootstrap machine boots and starts hosting the remote resources that are required for the master machines to boot.
2. The master machines fetch the remote resources from the bootstrap machine and finishes booting.
3. The master machines use the bootstrap node to form an `etcd` cluster.
4. The bootstrap node starts a temporary Kubernetes control plane by using the newly created `etcd` cluster.
5. The temporary control plane schedules the production control plane to the master machines.
6. The temporary control plane shuts down and yields to the production control plane.
7. The bootstrap node injects Red Hat OpenShift-specific components into the newly formed control plane.
8. The installer tears down the bootstrap node.
9. Worker nodes are added to the cluster.
10. The control plane installs other services in the form of Operators.
11. The Applications are deployed per the requirement.

The procedures that are used o deploy Red Hat OpenShift Container Platform are described in the following sections.

Performing pre-deployment validations

This section describes the pre-deployment checks that must be performed before the installation of the Red Hat OpenShift Container Platform is started. Then, the SSH keys are created for authorized communication between Red Hat OpenShift cluster.

Pre-deployment DNS validations

Perform a pre-deployment validation of the DNS lookups. For security reasons, the IPs are masked in Example 3-5.

Example 3-5 DNS lookups

```
[sthareja@sthareja ocp_ign_files]$ dig ocp4-lbutility-0.redbooks.info +short
<<< DNS Resolution Record IP>>>
[sthareja@sthareja ocp_ign_files]$ dig -x <IP> +short
ocp4-lbutility-0.redbooks.info.
[sthareja@sthareja ocp_ign_files]$ dig ocp4-appnode-1.ocp4.redbooks.info +short
<<< DNS Resolution Record IP>>>
```

```
<<< DNS Resolution Record IP>>>
[sthareja@sthareja ocp_ign_files]$ dig -x <IP> +short
ocp4-appnode-1.ocp4.redbooks.info.
[sthareja@sthareja ocp_ign_files]$ dig ocp4-appnode-0.ocp4.redbooks.info +short
<<< DNS Resolution Record IP>>>
[sthareja@sthareja ocp_ign_files]$ dig -x <IP> +short
ocp4-appnode-0.ocp4.redbooks.info.
[sthareja@sthareja ocp_ign_files]$ dig ocp4-appnode-2.ocp4.redbooks.info +short
<<< DNS Resolution Record IP>>>
<<< DNS Resolution Record IP>>>
[sthareja@sthareja ocp_ign_files]$ dig -x <IP> +short
ocp4-appnode-2.ocp4.redbooks.info.
[sthareja@sthareja ocp_ign_files]$ dig ocp4-master-0.ocp4.redbooks.info +short
<<< DNS Resolution Record IP>>>
<<< DNS Resolution Record IP>>>
[sthareja@sthareja ocp_ign_files]$ dig -x <IP> +short
ocp4-master-0.ocp4.redbooks.info.
[sthareja@sthareja ocp_ign_files]$ dig ocp4-master-1.ocp4.redbooks.info +short
<<< DNS Resolution Record IP>>>
<<< DNS Resolution Record IP>>>
[sthareja@sthareja ocp_ign_files]$ dig -x <IP> +short
48.83.26a9.ip4.static.sl-reverse.com.
[sthareja@sthareja ocp_ign_files]$ dig -x <IP> +short
ocp4-master-1.ocp4.redbooks.info.
[sthareja@sthareja ocp_ign_files]$ dig ocp4-master-2.ocp4.redbooks.info +short
<<< DNS Resolution Record IP>>>
<<< DNS Resolution Record IP>>>
[sthareja@sthareja ocp_ign_files]$ dig -x <IP> +short
ocp4-master-2.ocp4.redbooks.info.
[sthareja@sthareja ocp_ign_files]$ dig api.cluster4 +short
[sthareja@sthareja ocp_ign_files]$ dig *.apps.cluster4 +short
[sthareja@sthareja ocp_ign_files]$ dig api-int.cluster4 +short
[sthareja@sthareja ocp_ign_files]$ dig etcd-0.cluster4 +short
;; Truncated, retrying in TCP mode.
[sthareja@sthareja ocp_ign_files]$ dig etcd-1.cluster4 +short
[sthareja@sthareja ocp_ign_files]$ dig etcd-0.cluster4 +short
[sthareja@sthareja ocp_ign_files]$ dig etcd-2.cluster4 +short
[sthareja@sthareja ocp_ign_files]$ dig
_etcd-server-ssl._tcp.cluster4.ocp4.redbooks.info +short
```

Downloading Red Hat Enterprise Linux CoreOS and Pull Secret

Complete the following steps:

1. Browse the Red Hat website to download the Red Hat OpenShift installer by selecting the suitable infrastructure provider, as shown in Figure 3-28.

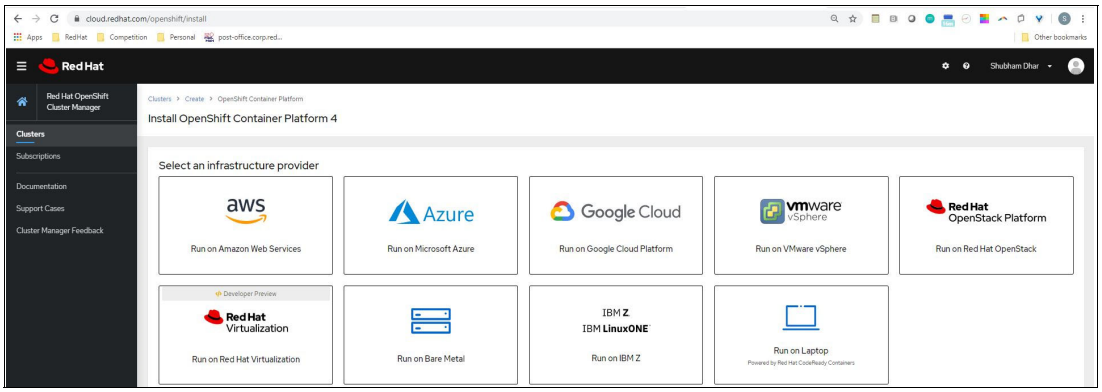


Figure 3-28 Red Hat OpenShift Cluster Manager on cloud.redhat.com

2. After the installer operating system is selected and downloaded, download or copy the Pull Secret by clicking **Download**, as shown in Figure 3-29.

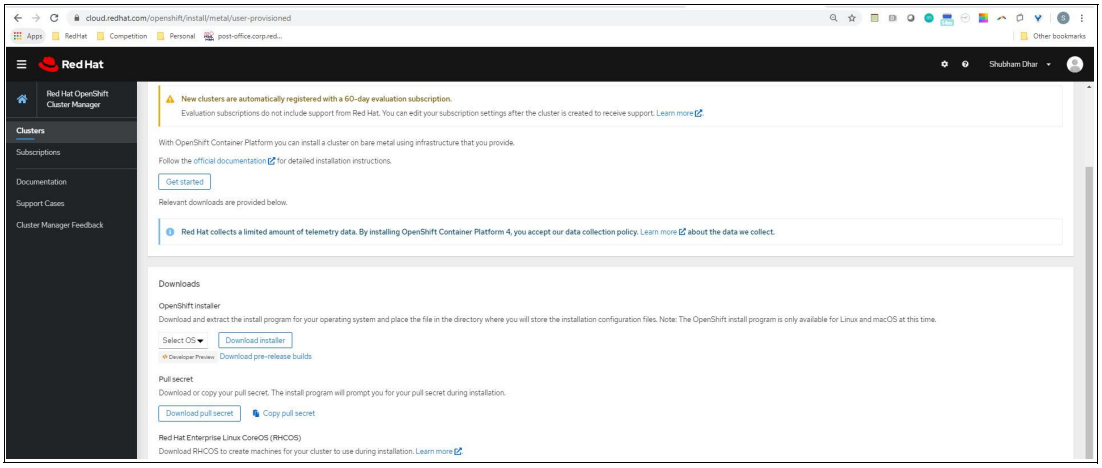


Figure 3-29 Downloading pull secret from cloud.redhat.com for deployment

3. You are redirected to the Red Hat OpenShift mirror file server for relevant image downloads, as shown in Figure 3-30. Click the relevant installer file, such as `.img` or `.iso`.

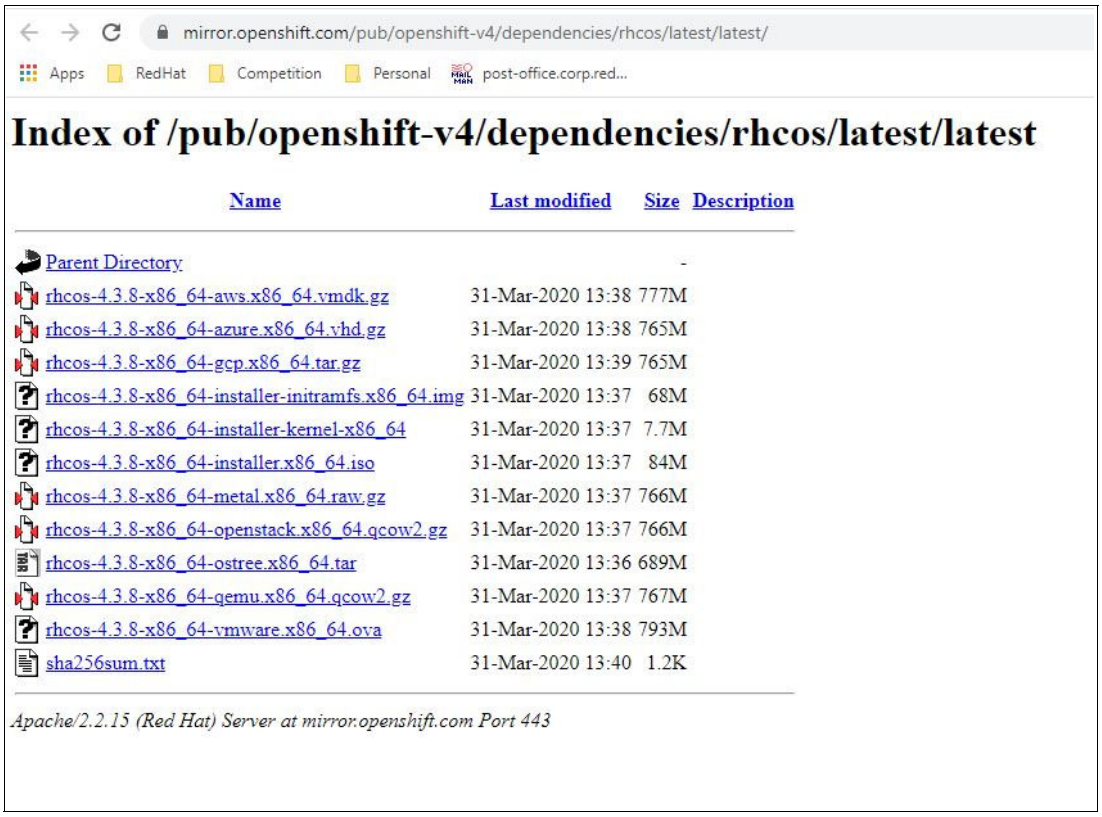


Figure 3-30 Red Hat OpenShift mirror for Red Hat Enterprise Linux CoreOS Images

4. Check the IP address configuration and make a directory on the bastion machine:

```
$ Ip a
$ mkdir ~/ocp4-downloads
```

Installing and configuring HAproxy

Install and set up HAproxy on the LB Utility node. Run the following commands to install HAproxy after SSH to LB utility server from Bastion host:

Note: As shown in Example 3-6, copy and paste the content of the `haproxy.cfg` file to the LB Utility server's `haproxy.cfg` file.

```
[COMMANDS]
bastion$ ssh root@lbutility          //SSH to LB Utility server as root user
$ yum  install haproxy -y            //Installing haproxy packages
$ vi /etc/haproxy/haproxy.cfg
```

Change the `haproxy.cfg` file as shown in Example 3-6.

Example 3-6 haproxy.cfg configuration file

```
[SAMPLE haproxy.cfg Configuration Changes]
#---------------------------------------------------------------------
# Example configuration for a possible web application.  See the
# full configuration options online.
```

```
#
#   http://haproxy.1wt.eu/download/1.4/doc/configuration.txt
#
#---------------------------------------------------------------------

#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global
    # to have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events.  This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    # 2) configure local2 events to go to the /var/log/haproxy.log
    #   file. A line like the following can be added to
    #   /etc/sysconfig/syslog
    #
    #    local2.*                       /var/log/haproxy.log
    #
    log         127.0.0.1 local2

    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

    # turn on stats unix socket
    stats socket /var/lib/haproxy/stats

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
```

```
#---------------------------------------------------------------------
# stats
#---------------------------------------------------------------------
listen  stats   169.38.131.75:1936
        mode            http
        log             global

        maxconn 10

        clitimeout      100s
        srvtimeout      100s
        contimeout      100s
        timeout queue   100s

        stats enable
        stats hide-version
        stats refresh 30s
        stats show-node
        stats uri  /haproxy?stats


#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
frontend  main *:5000
    acl url_static       path_beg       -i /static /images /javascript
/stylesheets
    acl url_static       path_end       -i .jpg .gif .png .css .js

    use_backend static          if url_static
    default_backend             app

#---------------------------------------------------------------------
# static backend for serving up images, stylesheets and such
#---------------------------------------------------------------------
backend static
    balance     roundrobin
    server      static 127.0.0.1:4331 check

#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend app
    balance     roundrobin
    server  app1 127.0.0.1:5001 check
    server  app2 127.0.0.1:5002 check
    server  app3 127.0.0.1:5003 check
    server  app4 127.0.0.1:5004 check

frontend openshift-api-server
 bind *:6443
 default_backend openshift-api-server
 mode tcp
 option tcplog
```

```
backend openshift-api-server
    balance     source
    mode tcp
#    server ocp4-bootstrap-0.ocp4.redbooks.info 169.38.122.221:6443 check
    server ocp4-master-1.ocp4.redbooks.info 169.38.131.82:6443 check
    server ocp4-master-2.ocp4.redbooks.info 169.38.131.73:6443 check
    server ocp4-master-3.ocp4.redbooks.info 169.38.131.93:6443 check
#    server api-int.cluster4.ocp4.redbooks.info 169.38.131.75:6443 check


frontend machine-config-server
 bind *:22623
 default_backend machine-config-server
 mode tcp
 option tcplog


backend machine-config-server
 balance source
 mode tcp
# server ocp4-bootstrap-0.ocp4.redbooks.info 169.38.122.221:22623 check
 server ocp4-master-1.ocp4.redbooks.info 169.38.131.82:22623 check
 server ocp4-master2.ocp4.redbooks.info 169.38.131.73:22623 check
 server ocp4-master3.ocp4.redbooks.info 169.38.131.93:22623 check
# server api-int.cluster4.ocp4.redbooks.info 169.38.131.75:22623 check
```

After saving the changes to the `haproxy.cfg` file, start the haproxy service and enable it by running the **`systemctl`** command:

```
$ systemctl start haproxy              //Start the haproxy service on RHEL OS
$ systemctl enable haproxy             //Enable haproxy service on RHEL OS
```

Installing and configuring HTTP server

Install and set up the Apache HTTP web server on the LB Utility node by running the commands that are shown in Example 3-7.

Example 3-7 Install and configure the HTTP server

```
[COMMANDS]
bastion$ ssh root@lbutility          //SSH to LB Utility server as root user

$ yum install httpd -y               //Installing HTTP Packages
$ vi /etc/httpd/conf/httpd.conf      //Editing port configuration for http server
                                     //Change port to 8080 from 80

$ systemctl start httpd              //Start HTTP service on the server
$ systemctl enable httpd             //Enable HTTP service on the server
$ systemctl status httpd             //Check Status of the HTTP service
```

Generating SSH keys

Generate the SSH keys for the server for accessing the cluster as shown in Example 3-8. Save the private *and* public keys for access of the cluster in future.

Example 3-8 Generate the SSH keys

```
[COMMANDS]
$ ssh-keygen -t rsa -b 4096 -N '' \
-f ~/.ssh/id_rsa
$ eval "$(ssh-agent -s)"
$ ssh-add ~/.ssh/id_rsa
$ SSH_PUB_KEY=`cat ~/.ssh/id_rsa`

[SAMPLE OUTPUTS]
[root@os4-lbutility-0 ~]# ssh-keygen -t rsa -b 4096 -N '' \
> -f ~/.ssh/id_rsa

Generating public/private rsa key pair.
Created directory '/root/.ssh'.
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:/Kw9hVlu4zgkYLLR/Ol+ZbLvXE4flzWOEUNJBa4ECXY
root@os4-lbutility-0.redbooks.info
The key's randomart image is:
+---[RSA 4096]----+
|       o.E. .++. |
|      . ... o.   |
|     .     . +   |
|    o +.  . o o  |
|     = oS  * . ..|
|    .   oo= * * +|
|         *o* B ++|
|         oB +. ++|
|        . .+ o+ o|
+----[SHA256]-----+


[root@os4-lbutility-0 ~]# eval "$(ssh-agent -s)"
Agent pid 15892


[root@os4-lbutility-0 ~]# ssh-add ~/.ssh/id_rsa
Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)


[root@os4-lbutility-0 ~]# SSH_PUB_KEY=`cat ~/.ssh/id_rsa`
[root@os4-lbutility-0 ~]#

[SAMPLE SSH Key Content]
```

```
'ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCkd5FNPfPCD8OmVpYi/mQIFPNoaMgNZkAkD9fthCCAww7ykuPGGL
BdqTmhhmF38PrfDcxG+wZYkE+l1RJ/ZbmYa2BYIOH6ucH3Nqj54K9U9kk8JRfbgYht7+V+pAwBmppmVxSQ
kJVaLdTdJY1QUn82UEi+WERzkOFLDfK52eAEO8NNH78YS6OYyGxqJrajPlOcdoMkc+VtuOB8m9VLLMcdun
Dn1QRe1/S5N37yCOIJI1iAnyehIk/JzfKia/efyvVPPyUPfbyrXdltNJwkh74nWKqIW2TuATtX+ZEn9rwD
tWLJP9d4+fIq1mASeGOGgW7TBlcBU7Pjtq8qOBzaO2sPYaDSrvuiM+ua1dOzUP2WXcLYJwNbLsoogbArhB
XO5umhRnaQV4jwPhxKZqVSCmvGWPC/3P8khMW/YL89SkBqhFKoMiN1rove91gabDtgEckkgSJalRLg6/Kb
LLbJ8A4XcXVxLLIHNzty+TbTeFttL5F69vD9xmmgkRgTiUI8ErVTlSGxYW5Oz8gUa1367NUW+q1F99q8/w
F469n4Owwnu5D48IYoX9Kf3f6xwZaSQHCe4YIbx3kFOCQlWpE8rH/O+6CnqjxWuiKI7iGKcKnrjktfq6A/O
vCYLbnZFLW4xkbPX1o/SpYY/dcF4fWvUSM5B8GxtG6qYNPpRimae3du93Q=='
```

Download the pull secret as `.txt` file from this website (see Figure 3-29 on page 75) and remove the email from the pull secret, as highlighted in Example 3-9.

Example 3-9 Remove the email

```
'{"auths":{"cloud.openshift.com":{"auth":"b3BlbnNoaWZOLXJlbGVhc2UtZGV2K2Fma2hhbjE5
MXNqZWlpbGlldmhqdWVyZGM5bmczbTdjMWh1OkhVRUk5TVkzNDZOMOZCN1I1OTlXWTI3UzRBUElTRkNES1
Q1TUpVTjU1UOg2UFdIWExXNVRZRVBDSkpaTkhRQlk=","email":"afkhan19@in.ibm.com"},"quay.i
o":{"auth":"b3BlbnNoaWZOLXJlbGVhc2UtZGV2K2Fma2hhbjE5MXNqZWlpbGlldmhqdWVyZGM5bmczbT
djMWh1OkhVRUk5TVkzNDZOMOZCN1I1OTlXWTI3UzRBUElTRkNES1Q1TUpVTjU1UOg2UFdIWExXNVRZRVBD
SkpaTkhRQlk=","email":"afkhan19@in.ibm.com"},"registry.connect.redhat.com":{"auth"
:"NTI5MDEONzh8dWhjLTFTSmVJSUxJRVZoanVlUmRjOW5nMOO3QzFoVTpleUpoYkdjaU9pSlNVelV4TWlK
OS5leUp6ZFdJaU9pSTNOMkZoWW1abVltSTJZelEwTVdZNE9ERTVOV1JtTm1GbE1XUXhNVFkOWmlKOS5jTD
BxWFZkZO9RMWZkMGdJaFJfbDZDXy1OQUlBZzhSeVNzU1BPeW5vSzdicDdPZ1R6aV9TUllIM2o4Z1VKYTBG
dDlSZGpheHRtcHpzZzBQM2JZUHpjaF9MMVFERzhFbFpOSE1KaUlQNkpqZC1pWFZONUY2TnpKWkZybkI1N3
ZwQ3AwTy1Ic2NOX1RCU3lFR2tWZTJiZmppVDJtQWtBMk1EVHlweEFLTzR5bWVlRDRNSVhIaUppcko1TkNu
V3hSNzlOOHIxam9fYmdoSTNRall2QzFBby1NMml4ZjJmVjR4eTVFdnQwbFB3c2RPc29tMTVDbXE5VUR5b1
JDZG1UTHlZdVVhLVdscTlrc19jRzJ1MDZQOHVtWXY1Y2E5NEYyQVVTU1UyOWlYbWswdi1sakJZTFd5X3Rv
RFpScFBDWmFRbFp6X3EyYmM2c2VFMVdPdzdfcnJFYmlTTTZvdDRjcXhNc215VmdBNOx3Qk1hS1F6cnFoMO
xKZ1BBdU9IRDRPNDZTejlBV2JBbjd2ZVNKX2O4Nm5Kd3BBVGlBN3pEbExFbjZXXzV4Q2lLMElrSnlEczYt
Y3VkcGtELVZCVWRjUVFPVOxKbTR1ZOZBR2NVZSO1blZsRHZCdjJSc1JXRUVKMnktUk5GVnRrRF9YRGo4Ul
ctdWlMS2ZINFNRYOEOZW9FTVgwZ3Q5TWdrSEhiTGp4WVRXUUhfcOJGMO9aMU14cO8wbXNZR3ExbFJ6bWpm
Unl3SzE5eW9FSXFOdklKWmZYWDl6VmdWTzB2dnJ2LVBLTThFTXI2anpFZjJOTjBCemFzNXgyZHFja2t2X3
A4RkFReUYzTk44amxLalhxTOZsUlRhTONCQ3FOaWxSekI1aGZRcUNtakMzSjgyUFJtUjJZWmRBN2FpcnVs
QkMOMA==","email":"afkhan19@in.ibm.com"},"registry.redhat.io":{"auth":"NTI5MDEONzh
8dWhjLTFTSmVJSUxJRVZoanVlUmRjOW5nMOO3QzFoVTpleUpoYkdjaU9pSlNVelV4TWlKOS5leUp6ZFdJa
U9pSTNOMkZoWW1abVltSTJZelEwTVdZNE9ERTVOV1JtTm1GbE1XUXhNVFkOWmlKOS5jTDBxWFZkZO9RMWZ
kMGdJaFJfbDZDXy1OQUlBZzhSeVNzU1BPeW5vSzdicDdPZ1R6aV9TUllIM2o4Z1VKYTBGdDlSZGpheHRtc
HpzZzBQM2JZUHpjaF9MMVFERzhFbFpOSE1KaUlQNkpqZC1pWFZONUY2TnpKWkZybkI1N3ZwQ3AwTy1Ic2N
OX1RCU3lFR2tWZTJiZmppVDJtQWtBMk1EVHlweEFLTzR5bWVlRDRNSVhIaUppcko1TkNuV3hSNzlOOHIxa
m9fYmdoSTNRall2QzFBby1NMml4ZjJmVjR4eTVFdnQwbFB3c2RPc29tMTVDbXE5VUR5b1JDZG1UTHlZdVV
hLVdscTlrc19jRzJ1MDZQOHVtWXY1Y2E5NEYyQVVTU1UyOWlYbWswdi1sakJZTFd5X3RvRFpScFBDWmFRb
Fp6X3EyYmM2c2VFMVdPdzdfcnJFYmlTTTZvdDRjcXhNc215VmdBNOx3Qk1hS1F6cnFoMOxKZ1BBdU9IRDR
PNDZTejlBV2JBbjd2ZVNKX2O4Nm5Kd3BBVGlBN3pEbExFbjZXXzV4Q2lLMElrSnlEczYtY3VkcGtELVZCV
WRjUVFPVOxKbTR1ZOZBR2NVZSO1blZsRHZCdjJSc1JXRUVKMnktUk5GVnRrRF9YRGo4UlctdWlMS2ZINFN
RYOEOZW9FTVgwZ3Q5TWdrSEhiTGp4WVRXUUhfcOJGMO9aMU14cO8wbXNZR3ExbFJ6bWpmUnl3SzE5eW9FS
XFOdklKWmZYWDl6VmdWTzB2dnJ2LVBLTThFTXI2anpFZjJOTjBCemFzNXgyZHFja2t2X3A4RkFReUYzTk4
4amxLalhxTOZsUlRhTONCQ3FOaWxSekI1aGZRcUNtakMzSjgyUFJtUjJZWmRBN2FpcnVsQkMOMA=="}}}'
```

Creating install-config.yaml for Red Hat OpenShift ignition files

Complete the following steps to create the base `install-config.yaml` for Red Hat OpenShift ignition file generation:

1. Log in to the HTTP LB Utility machine.
2. Create a `source-install-config.yaml` file; update the networking configuration as shown in **bold text** in Example 3-10.
3. Change the following:
   - `baseDomain`: This is the domain of your environment.
   - `metadata.name`: This is your cluster ID.

     Note: This change effectively makes all fully qualified domain names (FQDNs) ocp4.example.com.

   - `pullSecret`: This pull secret can be obtained by going to this website and completing the following steps:
     i. Login by using your Red Hat account.
     ii. Click **Bare Metal**.
     iii. Select **Download Pull Secret** or **Copy Pull Secret** as described in “Downloading Red Hat Enterprise Linux CoreOS and Pull Secret” on page 74.
   - `sshKey`: This is your public SSH key (for example, `id_rsa.pub`).

Example 3-10 Update the source-install-config.yaml file

```
$ cat >source-install-config.yaml
apiVersion: v1
baseDomain: redhatlab.example.net
### INCLUDE THE RED SECTION ONLY IF USING A LOCAL REGISTRY
Compute:
- hyperthreading: Disabled
  name: worker
  platform: {}
  replicas: 0
controlPlane:
  hyperthreading: Disabled
  name: master
  platform: {}
  replicas: 1
Metadata:
  creationTimestamp: null
  name: ocp43
Networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineCIDR: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
Platform:
  none: {}
```

```
pullSecret: '<CONTENTS OF .pullsecret.json if using internet pull or
$HOME/ocp_installer/merged_pullsecret.json if using a local registry(created in
the previous section>'
sshKey: '<PASTE contents of ~/ocp_installer/config/id_ocpinstaller_ed25519.pub >'
```

Note: Replace the subscriber email ID from the Pull Secret downloaded from this website before deploying (review the omitted or strike through lines that are shown in Example 3-11).

Example 3-11 Sample configuration install file

```
apiVersion: v1
baseDomain: ocp4.redbooks.info
compute:
- hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
metadata:
  name: cluster4
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
pullSecret:
'{"auths":{"cloud.openshift.com":{"auth":"b3BlbnNoaWZOLXJlbGVhc2UtZGV2K2Fma2hhbjE5
MXNqZWlpbGlldmhqdWVyZGM5bmczbTdjMWh1OkhVRUk5TVkzNDZOMOZCN1I1OTlXWTI3UzRBUElTRkNES1
Q1TUpVTjU1UOg2UFdIWExXNVRZRVBDSkpaTkhRQlk=","email":"afkhan19@in.ibm.com"},"quay.i
o":{"auth":"b3BlbnNoaWZOLXJlbGVhc2UtZGV2K2Fma2hhbjE5MXNqZWlpbGlldmhqdWVyZGM5bmczbT
djMWh1OkhVRUk5TVkzNDZOMOZCN1I1OTlXWTI3UzRBUElTRkNES1Q1TUpVTjU1UOg2UFdIWExXNVRZRVBD
SkpaTkhRQlk=","email":"afkhan19@in.ibm.com"},"registry.connect.redhat.com":{"auth"
:"NTI5MDEONzh8dWhjLTFTSmVJSUxJRVZoanVlUmRjOW5nMOO3QzFoVTpleUpoYkdjaU9pSlNVelV4TWlK
OS5leUp6ZFdJaU9pSTNOMkZoWW1abVltSTJZelEwTVdZNE9ERTVOV1JtTm1GbE1XUXhNVFkOWmlKOS5jTD
BxWFZkZO9RMWZkMGdJaFJfbDZDXy1OQUlBZzhSeVNzU1BPeW5vSzdicDdPZ1R6aV9TUllIM2o4Z1VKYTBG
dDlSZGpheHRtcHpzZzBQM2JZUHpjaF9MMVFERzhFbFpOSE1KaUlQNkpqZC1pWFZONUY2TnpKWkZybkI1N3
ZwQ3AwTy1Ic2NOX1RCU3lFR2tWZTJiZmppVDJtQWtBMk1EVHlweEFLTzR5bWVlRDRNSVhIaUppcko1TkNu
V3hSNzlOOHIxam9fYmdoSTNRall2QzFBby1NMml4ZjJmVjR4eTVFdnQwbFB3c2RPc29tMTVDbXE5VUR5b1
JDZG1UTHlZdVVhLVdscTlrc19jRzJ1MDZQOHVtWXY1Y2E5NEYyQVVTU1UyOWlYbWswdi1sakJZTFd5X3Rv
RFpScFBDWmFRbFp6X3EyYmM2c2VFMVdPdzdfcnJFYmlTTTZvdDRjcXhNc215VmdBNOx3Qk1hS1F6cnFoMO
xKZ1BBdU9IRDRPNDZTejlBV2JBbjd2ZVNKX2O4Nm5Kd3BBVGlBN3pEbExfbjZXXzV4Q2lLMElrSnlEczYt
Y3VkcGtELVZCVWRjUVFPVOxKbTR1ZOZBR2NVZSO1blZsRHZCdjJSc1JXRUVKMnktUk5GVnRrRF9YRGo4Ul
ctdWlMS2ZINFNRYOEOZW9FTVgwZ3Q5TWdrSEhiTGp4WVRXUUhfcOJGMO9aMU14cO8wbXNZR3ExbFJ6bWpm
Unl3SzE5eW9FSXFOdklKWmZYWDl6VmdWTzB2dnJ2LVBLTThFTXI2anpFZjJOTjBCemFzNXgyZHFja2t2X3
A4RkFReUYzTk44amxLalhxTOZsUlRhTONCQ3FOaWxSekI1aGZRcUNtakMzSjgyUFJtUjJZWmRBN2FpcnVs
QkMOMA==","email":"afkhan19@in.ibm.com"},"registry.redhat.io":{"auth":"NTI5MDEONzh
8dWhjLTFTSmVJSUxJRVZoanVlUmRjOW5nMOO3QzFoVTpleUpoYkdjaU9pSlNVelV4TWlKOS5leUp6ZFdJa
U9pSTNOMkZoWW1abVltSTJZelEwTVdZNE9ERTVOV1JtTm1GbE1XUXhNVFkOWmlKOS5jTDBxWFZkZO9RMWZ
```

```
kMGdJaFJfbDZDXy1OQUlBZzhSeVNzU1BPeW5vSzdicDdPZ1R6aV9TUllIM2o4Z1VKYTBGdDlSZGpheHRtc
HpzZzBQM2JZUHpjaF9MMVFERzhFbFpOSE1KaUlQNkpqZC1pWFZONUY2TnpKWkZybkI1N3ZwQ3AwTy1Ic2N
OX1RCU3lFR2tWZTJiZmppVDJtQWtBMk1EVHlweEFLTzR5bWVlRDRNSVhIaUppcko1TkNuV3hSNzlOOHIxa
m9fYmdoSTNRall2QzFBby1NMml4ZjJmVjR4eTVFdnQwbFB3c2RPc29tMTVDbXE5VUR5b1JDZG1UTHlZdVV
hLVdscTlrc19jRzJ1MDZQOHVtWXY1Y2E5NEYyQVVTU1UyOWlYbWswdi1sakJZTFd5X3RvRFpScFBDWmFRb
Fp6X3EyYmM2c2VFMVdPdzdfcnJFYmlTTTZvdDRjcXhNc215VmdBNOx3Qk1hS1F6cnFoMOxKZ1BBdU9IRDR
PNDZTejlBV2JBbjd2ZVNKX2O4Nm5Kd3BBVGlBN3pEbExFbjZXXzV4Q2lLMElrSnlEczYtY3VkcGtELVZCV
WRjUVFPVOxKbTR1ZOZBR2NVZSO1blZsRHZCdjJSc1JXRUVKMnktUk5GVnRrRF9YRGo4UlctdWlMS2ZINFN
RYOEOZW9FTVgwZ3Q5TWdrSEhiTGp4WVRXUUhfcOJGMO9aMU14cO8wbXNZR3ExbFJ6bWpmUnl3SzE5eW9FS
XFOdklKWmZYWDl6VmdWTzB2dnJ2LVBLTThFTXI2anpFZjJOTjBCemFzNXgyZHFja2t2X3A4RkFReUYzTk4
4amxLalhxTOZsUlRhTONCQ3FOaWxSekI1aGZRcUNtakMzSjgyUFJtUjJZWmRBN2FpcnVsQkMOMA=="}}}'
sshKey: 'ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCkd5FNPfPCD8OmVpYi/mQIFPNoaMgNZkAkD9fthCCAww7ykuPGGL
BdqTmhhmF38PrfDcxG+wZYkE+l1RJ/ZbmYa2BYIOH6ucH3Nqj54K9U9kk8JRfbgYht7+V+pAwBmppmVxSQ
kJVaLdTdJY1QUn82UEi+WERzkOFLDfK52eAEO8NNH78YS6OYyGxqJrajPlOcdoMkc+VtuOB8m9VLLMcdun
Dn1QRe1/S5N37yCOIJI1iAnyehIk/JzfKia/efyvVPPyUPfbyrXdltNJwkh74nWKqIW2TuATtX+ZEn9rwD
tWLJP9d4+fIq1mASeGOGgW7TBlcBU7Pjtq8qOBzaO2sPYaDSrvuiM+ua1dOzUP2WXcLYJwNbLsoogbArhB
XO5umhRnaQV4jwPhxKZqVSCmvGWPC/3P8khMW/YL89SkBqhFKoMiN1rove91gabDtgEckkgSJalRLg6/Kb
LLbJ8A4XcXVxLLIHNzty+TbTeFttL5F69vD9xmmgkRgTiUI8ErVTlSGxYW5Oz8gUa1367NUW+q1F99q8/w
F469n4Owwnu5D48IYoX9Kf3f6xwZaSQHCe4YIbx3kFOCQlWpE8rH/O+6CnqjxWuiKI7iGKcKnrjktfq6A/O
vCYLbnZFLW4xkbPX1o/SpYY/dcF4fWvUSM5B8GxtG6qYNPpRimae3du93Q=='
```

4. After the configuration file is updated with the required information, create a hard link of the `source-install-config.yaml` file to `install-config.yaml`:

```
$ ln {source-,}install-config.yaml      //Hard Link Creation
$ cp *.ign /root/ocp4-backup            //Creating backup of install-config.yaml
```

5. Generate the ignition file by using `openshift-install` utility:

```
$ ./openshift-install create ignition-configs
```

The files that are shown in Example 3-12 are left in your ~/`ocp4` working directory.

Example 3-12 Files in your ocp4 working directory

```
tree .
.

auth

     kubeadmin-password

     kubeconfig

bootstrap.ign

master.ign

metadata.json

worker.ign
```

When the ignition files are created, copy the ignition files to the web server on the LB Utility server. Add run permissions to the copied ignition files as shown in the following example:

Note: The certificate lasts only 24 hours after the `.ign` files are generated. Complete the installation process within 24 hours of generating the ignition files.

```
$ cp *.ign /var/www/html
$ chmod 755 *.ign
```

Installing Red Hat Enterprise Linux CoreOS on Bootstrap and Master

Complete the following steps:

1. Proceed with the CoreOS installation on the provisioned Bootstrap node on IBM Cloud Bare Metal server first.
2. Before you start the boot of the server, confirm that you uploaded the CoreOS image through the IPMI console. Set the Boot sequence to CD ROM/CD Drive.
3. Install the ISO image and then start the boot process.
4. When you are prompted with the installation, press TAB (see Figure 3-31), and enter the kernel parameters, as shown in Example 3-13 on page 86.

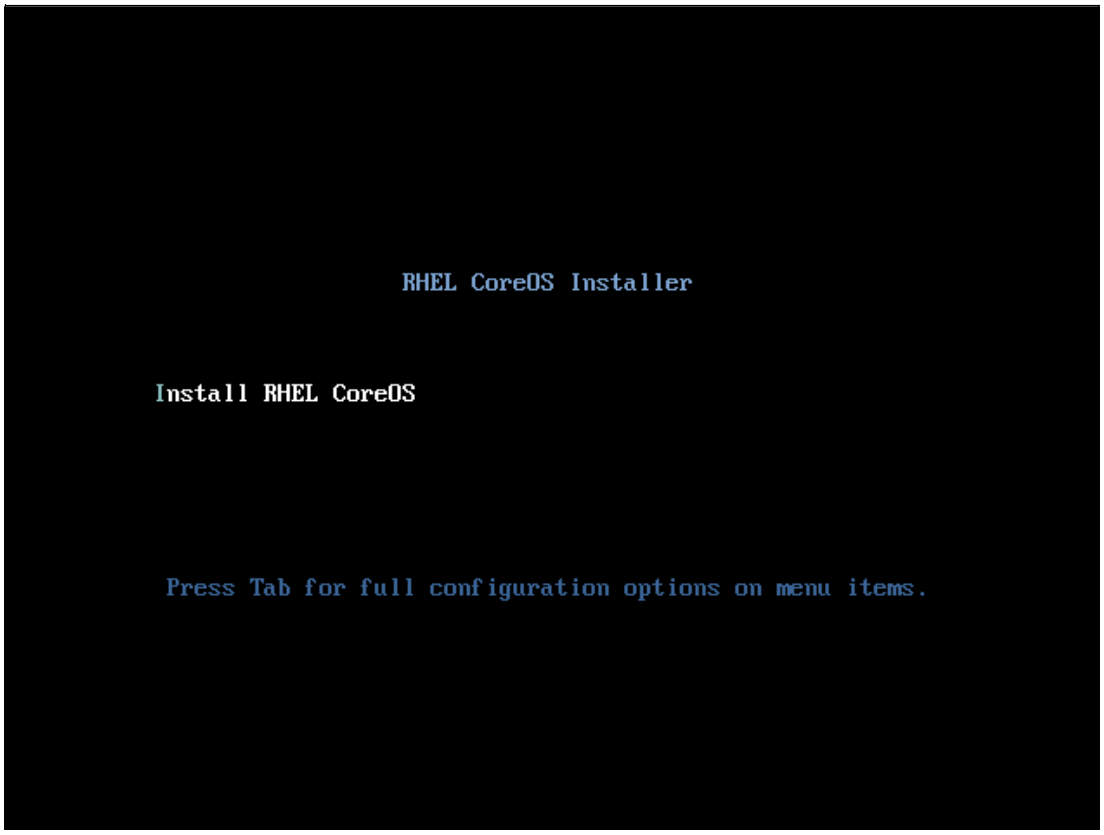


Figure 3-31 IPMI console during Red Hat Enterprise Linux CoreOS installation on bare metal nodes

The following CoreOS options are available:

– `coreos.inst.install_dev`: The block device on which Red Hat Enterprise Linux CoreOS installs.

– `coreos.inst.image_url`: The URL of the UEFI or BIOS image that you uploaded to the web server.

– `coreos.inst.ignition_url`: The URL of the Ignition configuration file for this machine type.

Example 3-13 CoreOS parameters

```
[BOOTSTRAP NODE KERNEL PARAMETER]
coreos.inst=yes
coreos.inst.install_dev=sda
coreos.inst.image_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/rhcos.raw.gz
coreos.inst.ignition_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/bootstrap
.ign
ip=169.38.122.221::169.38.122.209:255.255.255.240:ocp4-bootstrap-0.ocp4.redbooks.i
nfo:eno2:none
nameserver=67.228.254.4 nameserver=8.8.8.8
```

Figure 3-32 shows a sample view only of the kernel parameters, including the IP.

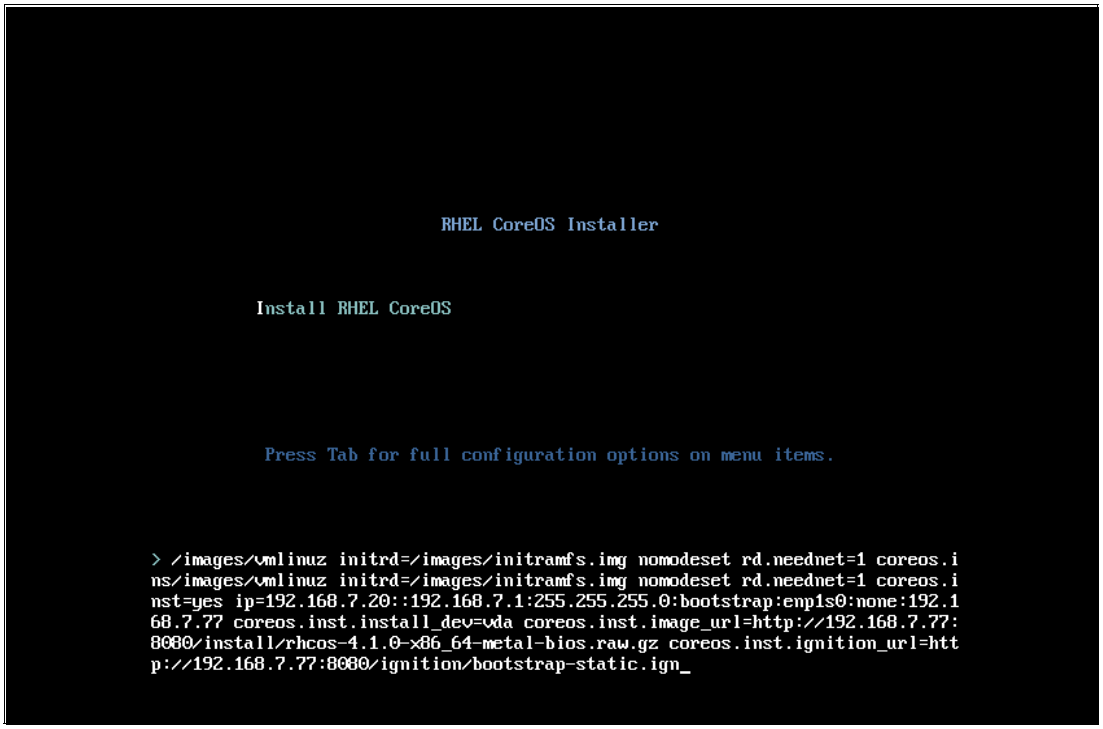


Figure 3-32 Passing Kernel Parameters to trigger installation of Red Hat OpenShift

Figure 3-33 shows the window of when the boot is done after the kernel parameters are added.

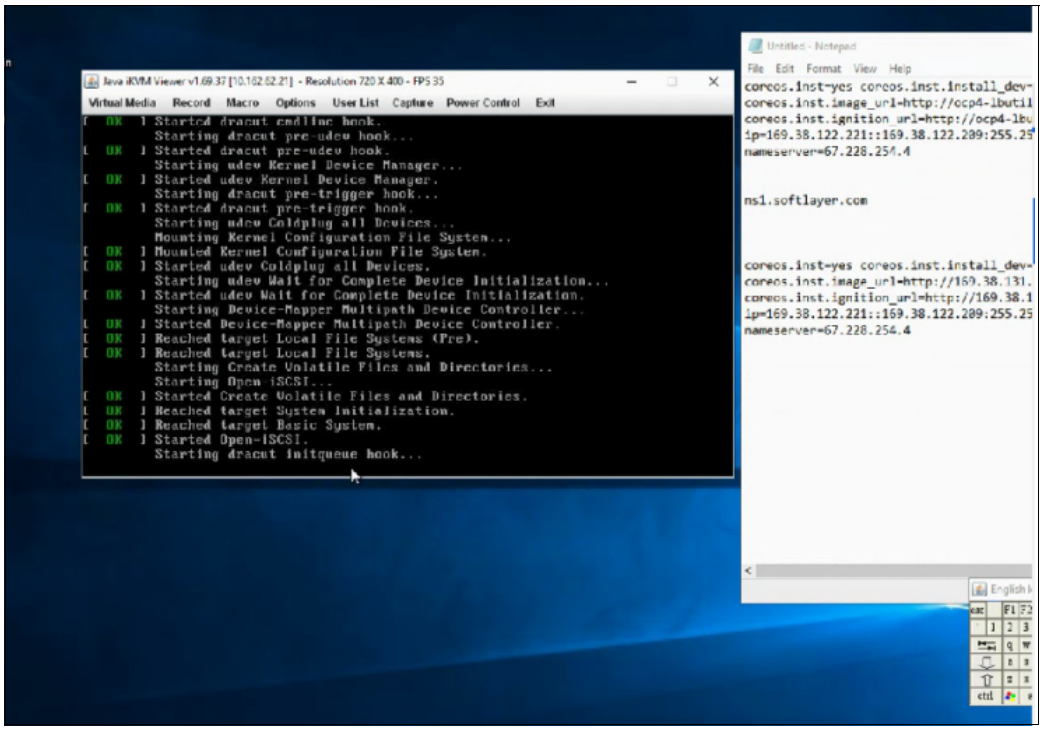


Figure 3-33 Installation of Red Hat Enterprise Linux CoreOS on bare metal servers

After the bootstrap node is deployed, it restarts three times. Therefore, after three successful restarts and when the node is up and running, verify the bootstrap node installation is complete:

```
$ ./openshift-install --dir=. wait-for bootstrap-complete --log-level=debug
```

After the bootstrap process is finished, you see the message that is shown in Example 3-14.

Example 3-14 Message after bootstrap finishes

```
DEBUG OpenShift Installer <Version>

DEBUG Built from commit <Built ID>

INFO Waiting up to 30m0s for the Kubernetes API at
https://api.cluster4.ocp4.redbooks.info:6443...

INFO API <Version> up

INFO Waiting up to 30m0s for bootstrapping to complete...

DEBUG Bootstrap status: complete

INFO It is now safe to remove the bootstrap resources
```

Note: Complete these steps for all node master installations from the installer machine and wait for the bootstrap completion message.

Perform all these steps to install all three master nodes following the bootstrap node procedure. Then, enter the kernel parameters by pressing TAB for each server (see Example 3-15).

Example 3-15 Install all three master nodes

```
######## MASTER 0 - 10.162.62.6###########
coreos.inst=yes
coreos.inst.install_dev=sda
coreos.inst.image_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/rhcos.raw.gz
coreos.inst.ignition_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/master.ig
n
ip=169.38.131.82::169.38.131.65:255.255.255.224:ocp4-master-0.ocp4.redbooks.info:e
np1s0f1:none
nameserver=67.228.254.4 nameserver=8.8.8.8

######## MASTER 1 - 10.162.62.58##########
coreos.inst=yes
coreos.inst.install_dev=sda
coreos.inst.image_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/rhcos.raw.gz
coreos.inst.ignition_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/master.ig
n
ip=169.38.131.73::169.38.131.65:255.255.255.224:ocp4-master-1.ocp4.redbooks.info:e
np1s0f1:none
nameserver=67.228.254.4 nameserver=8.8.8.8

######## MASTER 2 - 10.162.62.4##########
coreos.inst=yes
coreos.inst.install_dev=sda
coreos.inst.image_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/rhcos.raw.gz
coreos.inst.ignition_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/master.ig
n
ip=169.38.131.93::169.38.131.65:255.255.255.224:ocp4-master-2.ocp4.redbooks.info:e
np1s0f1:none
nameserver=67.228.254.4 nameserver=8.8.8.8
```

Installing Red Hat OpenShift client tools, jq, and worker (app) nodes

This section describes how to install the Red Hat OpenShift CLI Client Tools (oc Client) on your workstation.

Install the Red Hat OpenShift CLI Client Tools (oc Client) on your workstation, as shown in Example 3-16.

Example 3-16 Install Red Hat OpenShift CLI Client Tools

```
$ mkdir ~/oc-client

$ wget
https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-
linux.tar.gz?extIdCarryOver=true&sc_cid=701f2000001Css5AAC

$ tar xvf openshift-client-linux.tar.gz

$ PATH=$PATH:~/oc-client
$ oc version
```

From the workstation, complete the following steps:

1. Log in to the cluster by using the oc login. Set the KUBECONFIG parameter by exporting it as follows:

   ```
   $ export KUBECONFIG=/root/ocp4-downlods/ocp8apr/auth/kubeconfig
   ```

2. Check the Logged in user details:

   ```
   $ oc whoami
   system:admin
   ```

3. Install the jq tool on the LB Utility node, as shown in Example 3-17.

Example 3-17 Install jq tool

```
//Refer https://snapcraft.io/install/jq/rhel
$ sudo rpm -ivh
https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
$ sudo subscription-manager repos --enable "rhel-*-optional-rpms" --enable
"rhel-*-extras-rpms"
$ sudo yum update
$ sudo yum install snapd
$ sudo systemctl enable --now snapd.socket
$ sudo ln -s /var/lib/snapd/snap /snap
$ logout
$ sudo snap install jq
$ jq -version
```

4. Check the status of all the installed master nodes as ready:

   ```
   $ oc get nodes
   NAME                                STATUS   ROLES    AGE     VERSION
   ocp4-master-0.ocp4.redbooks.info    Ready    master,worker   23m   v1.16.2
   ocp4-master-1.ocp4.redbooks.info    Ready    master,worker   23m   v1.16.2
   ocp4-master-2.ocp4.redbooks.info    Ready    master,worker   23m   v1.16.2
   ```

5. When the masters are deployed successfully, deploy the Application/Worker Nodes of cluster. Follow the same procedure that is used when installing master and bootstrap nodes as described in “Installing Red Hat Enterprise Linux CoreOS on Bootstrap and Master” on page 85.

6. Plug in the ISO image and then start boot process. When you are prompted with the installation, press TAB, as shown in Figure 3-34.

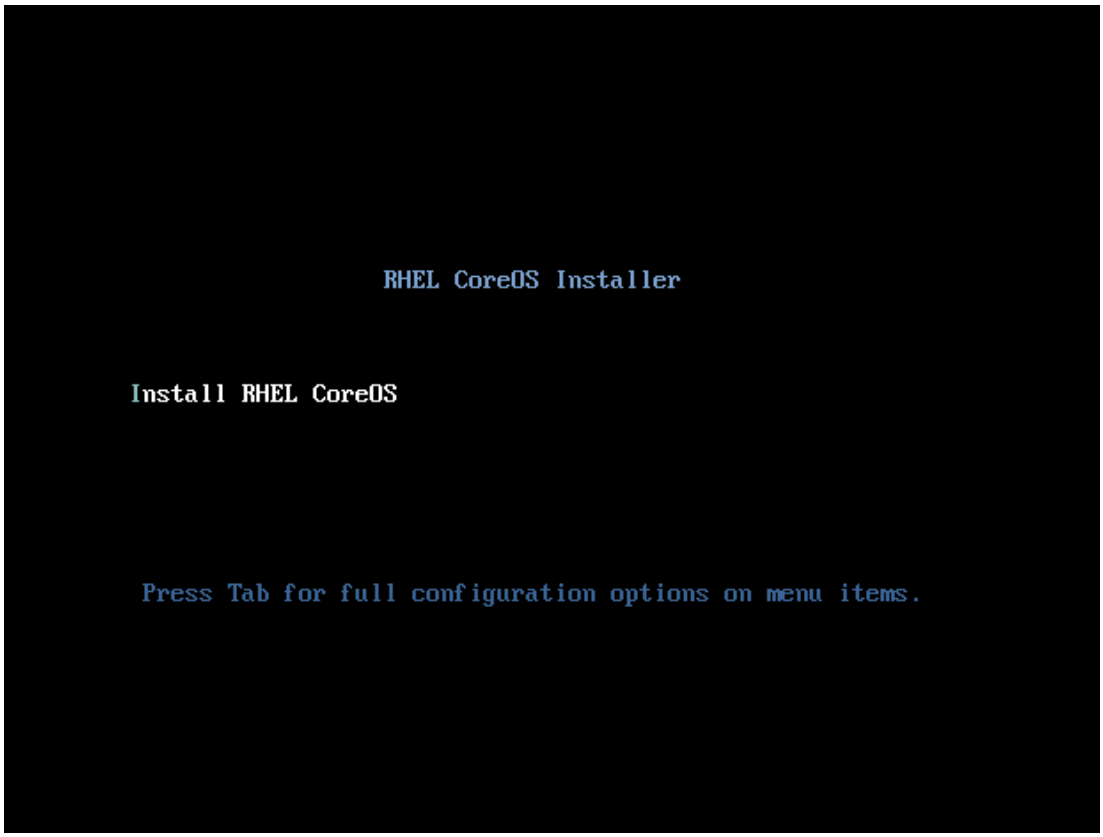


Figure 3-34 IPMI console during Red Hat Enterprise Linux CoreOS Installation on bare metal nodes

7. Enter the kernel parameters, as shown in Example 3-18.

Example 3-18 Application/Worker Node kernel parameters

```
######### APP NODE 0 #########
coreos.inst.install_dev=sda
coreos.inst.image_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/rhcos.raw.gz
coreos.inst.ignition_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/worker.ig
n
ip=169.38.131:78::169.38.131.65:255.255.255.224:ocp4-appnode-0.ocp4.redbooks.info:
eno2:none
nameserver=67.228.254.4 nameserver=8.8.8.8

######### APP NODE 1 #########
coreos.inst.install_dev=sda
coreos.inst.image_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/rhcos.raw.gz
coreos.inst.ignition_url=http://ocp4-lbutility-0.ocp4.redbooks.info:8080/worker.ig
n
ip=169.38.131:71::169.38.131.65:255.255.255.224:ocp4-appnode-1.ocp4.redbooks.info:
eno2:none
nameserver=67.228.254.4 nameserver=8.8.8.8
```

8. Check for any pending certificates by running the **oc** command. This command performs a check for any pending certificate request:

```
$ oc get csr
```

9. Approve all of the certificates on the installed cluster:

```
$ oc get csr -ojson | jq -r '.items[] | select(.status == {} ) |
.metadata.name' | xargs oc adm certificate approve
```

Note: This process might need to be completed twice.

3.3.6 Configuring Red Hat OpenShift container platform

This section describes how to configure Red Hat OpenShift Container Platform.

Post-deployment check-up

Check all of the installed nodes in the cluster, as shown in Example 3-19.

Example 3-19 List all installed nodes

```
$ oc get nodes
NAME                                STATUS   ROLES    AGE     VERSION
ocp4-appnode-0.ocp4.redbooks.info   Ready    worker   11m   v1.16.2
ocp4-appnode-1.ocp4.redbooks.info   Ready    worker   11m   v1.16.2
ocp4-master-0.ocp4.redbooks.info    Ready    master,worker   52m   v1.16.2
ocp4-master-1.ocp4.redbooks.info    Ready    master,worker   52m   v1.16.2
ocp4-master-2.ocp4.redbooks.info    Ready    master,worker   52m   v1.16.2
```

Check the Installed Cluster Operator, as shown in Example 3-20.

Example 3-20 Check the Installed Cluster Operator

```
$ oc get co
NAME                                 VERSION   AVAILABLE   PROGRESSING
DEGRADED   SINCE
authentication                      Unknown     Unknown        False
False      3d18h
cloud-credential                     4.3.9     True        False
False      4d1h
cluster-autoscaler                   4.3.9     True        False
False      4d
console                                        False        True
False      3d18h
dns                                  4.3.9     True        False
False      19m
image-registry                       4.3.9     True        False
False      32m
ingress                              4.3.9     True        False
False      3d18h
insights                             4.3.9     True        False
False      4d1h
kube-apiserver                       4.3.9     True        False
False      4d1h
kube-controller-manager              4.3.9     True        False
False      4d1h
```

```
kube-scheduler                             4.3.9     True        False
False      4d1h
machine-api                                4.3.9     True        False
False      4d1h
machine-config                             4.3.9     True        False
False      3h15m
marketplace                                4.3.9     True        False
False      45m
monitoring                                 4.3.9     True        False
False      3h12m
network                                    4.3.9     True        False
False      4d1h
node-tuning                                4.3.9     True        False
False      3h15m
openshift-apiserver                        4.3.9     True        False
False      2d16h
openshift-controller-manager               4.3.9     True        False
False      3d18h
openshift-samples                          4.3.9     True        False
False      4d
operator-lifecycle-manager                 4.3.9     True        False
False      4d1h
operator-lifecycle-manager-catalog         4.3.9     True        False
False      4d1h
operator-lifecycle-manager-packageserver   4.3.9     True        False
False      2d16h
service-ca                                 4.3.9     True        False
False      4d1h
service-catalog-apiserver                  4.3.9     True        False
False      4d1h
service-catalog-controller-manager         4.3.9     True        False
False      4d1h
storage                                    4.3.9     True        False
False      4d
```

Set the scheduler to false, as shown in Example 3-21.

Example 3-21 Set scheduler to false

```
$ oc edit scheduler
...
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  creationTimestamp: null
  name: cluster
spec:
  mastersSchedulable: false                //Changed this to false
  policy:
    name: ""
status: {}
```

Updating Ingress services for Worker Nodes

You must update the Ingress routers that are running on the Master nodes to worker nodes, so that applications that are deployed on worker nodes can take advantage of the same Ingress routers. After this process is done, modify the `haproxy.cfg` file with ingress and ingress-http configurations.

Delete the Red Hat OpenShift Ingress pods, as shown in Example 3-22.

Example 3-22 Identify Red Hat OpenShift Ingress pods

```
$ oc project openshift-ingress
Now using project "openshift-ingress" on server
"https://api.cluster4.ocp4.redbooks.info:6443"

$ oc get pods -o wide
NAME                               READY   STATUS              RESTARTS   AGE
IP              NODE                                NOMINATED NODE   READINESS
GATES
router-default-845d998b4c-279wv   0/1     MatchNodeSelector   0          6h27m
<none>          ocp4-master-2.ocp4.redbooks.info   <none>           <none>
router-default-845d998b4c-d4znc   1/1     Running             0          42m
169.38.131.82   ocp4-master-0.ocp4.redbooks.info   <none>           <none>
router-default-845d998b4c-njp54   1/1     Running             0          6h28m
169.38.131.73   ocp4-master-1.ocp4.redbooks.info   <none>           <none>
```

You are required to delete the identified pods, as shown in Example 3-23.

Example 3-23 Deleting the Pods

```
$ oc delete pod router-default-845d998b4c-279wv
pod "router-default-845d998b4c-279wv" deleted

$ oc delete pod router-default-845d998b4c-d4znc
pod "router-default-845d998b4c-d4znc" deleted

$ oc delete pod router-default-845d998b4c-njp54
pod "router-default-845d998b4c-njp54" deleted
```

Verify whether the Ingress pods are running on worker nodes after performing a delete operation, as shown in Example 3-24.

Example 3-24 Check Pods running

```
$ oc get pods -o wide
NAME                             READY   STATUS    RESTARTS   AGE   IP
NODE                                NOMINATED NODE   READINESS GATES

router-default-845d998b4c-bj2np   1/1     Running   0          92s   169.38.131.71
ocp4-appnode-1.ocp4.redbooks.info   <none>           <none>
router-default-845d998b4c-nb652   0/1     Running   0          20s   169.38.131.78
ocp4-appnode-0.ocp4.redbooks.info   <none>           <none>
```

Modify the HA Proxy configuration (`haproxy.cfg`) file with ingress and ingress-http configuration inputs, as shown in Example 3-25.

Example 3-25 Modify haproxy.cfg

```
#---------------------------------------------------------------------
# Example configuration for a possible web application.  See the
# full configuration options online.
#
#   http://haproxy.1wt.eu/download/1.4/doc/configuration.txt
#
#---------------------------------------------------------------------

#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global
    # to have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events.  This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    # 2) configure local2 events to go to the /var/log/haproxy.log
    #   file. A line like the following can be added to
    #   /etc/sysconfig/syslog
    #
    #    local2.*                       /var/log/haproxy.log
    #
    log         127.0.0.1 local2

    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

    # turn on stats unix socket
    stats socket /var/lib/haproxy/stats

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
```

```
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

#---------------------------------------------------------------------
# stats
#---------------------------------------------------------------------
listen  stats   169.38.131.75:1936
        mode            http
        log             global

        maxconn 10

        clitimeout      100s
        srvtimeout      100s
        contimeout      100s
        timeout queue   100s

        stats enable
        stats hide-version
        stats refresh 30s
        stats show-node
        stats uri  /haproxy?stats


#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
frontend  main *:5000
    acl url_static       path_beg       -i /static /images /javascript
/stylesheets
    acl url_static       path_end       -i .jpg .gif .png .css .js

    use_backend static          if url_static
    default_backend             app

#---------------------------------------------------------------------
# static backend for serving up images, stylesheets and such
#---------------------------------------------------------------------
backend static
    balance     roundrobin
    server      static 127.0.0.1:4331 check

#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend app
    balance     roundrobin
    server  app1 127.0.0.1:5001 check
    server  app2 127.0.0.1:5002 check
    server  app3 127.0.0.1:5003 check
```

```
    server  app4 127.0.0.1:5004 check

frontend openshift-api-server
 bind *:6443
 default_backend openshift-api-server
 mode tcp
 option tcplog


backend openshift-api-server
    balance     source
    mode tcp
#    server ocp4-bootstrap-0.ocp4.redbooks.info 169.38.122.221:6443 check
    server ocp4-master-0.ocp4.redbooks.info 169.38.131.82:6443 check
    server ocp4-master-1.ocp4.redbooks.info 169.38.131.73:6443 check
    server ocp4-master-2.ocp4.redbooks.info 169.38.131.93:6443 check
#    server api-int.cluster4.ocp4.redbooks.info 169.38.131.75:6443 check


frontend machine-config-server
 bind *:22623
 default_backend machine-config-server
 mode tcp
 option tcplog


backend machine-config-server
 balance source
 mode tcp
# server ocp4-bootstrap-0.ocp4.redbooks.info 169.38.122.221:22623 check
 server ocp4-master-0.ocp4.redbooks.info 169.38.131.82:22623 check
 server ocp4-master1.ocp4.redbooks.info 169.38.131.73:22623 check
 server ocp4-master2.ocp4.redbooks.info 169.38.131.93:22623 check


frontend ingress
 bind *:443
 default_backend ingress
 mode tcp
 option tcplog


backend ingress
    balance     source
    mode tcp
    server ocp4-appnode-0.ocp4.redbooks.info 169.38.131.78:443 check
    server ocp4-appnode-1.ocp4.redbooks.info 169.38.131.71:443 check


frontend ingress-http
 bind *:80
 default_backend ingress-http
 mode tcp
 option tcplog
```

```
backend ingress-http
 balance source
 mode tcp
# server ocp4-bootstrap-0.ocp4.redbooks.info 169.38.122.221:22623 check
    server ocp4-appnode-0.ocp4.redbooks.info 169.38.131.78:80 check
    server ocp4-appnode-1.ocp4.redbooks.info 169.38.131.71:80 check
# server api-int.cluster4.ocp4.redbooks.info 169.38.131.75:22623 check
```

Check the nodes by running the **oc get** command. The roles are assigned correctly to master and worker nodes, as shown in Example 3-26.

Example 3-26 Check the node assignments

```
$ oc get nodes
NAME                                STATUS   ROLES    AGE     VERSION
ocp4-appnode-0.ocp4.redbooks.info   Ready    worker   5d14h   v1.16.2
ocp4-appnode-1.ocp4.redbooks.info   Ready    worker   5d14h   v1.16.2
ocp4-master-0.ocp4.redbooks.info    Ready    master   5d21h   v1.16.2
ocp4-master-1.ocp4.redbooks.info    Ready    master   5d21h   v1.16.2
ocp4-master-2.ocp4.redbooks.info    Ready    master   5d21h   v1.16.2
```

Finishing configuration of Red Hat OpenShift V4.3

Verify that all of the operators are installed and running in the deployed Red Hat OpenShift V4.3 cluster, as shown in Example 3-27.

Example 3-27 Check running operators

```
$ oc get co
NAME                                       VERSION   AVAILABLE   PROGRESSING
DEGRADED   SINCE
authentication                             4.3.9     True        False
False      5d15h
cloud-credential                           4.3.9     True        False
False      5d22h
cluster-autoscaler                         4.3.9     True        False
False      5d22h
console                                    4.3.9     True        False
False      5d16h
dns                                        4.3.9     True        False
False      45h
image-registry                             4.3.9     True        False
False      46h
ingress                                    4.3.9     True        False
False      5d16h
insights                                   4.3.9     True        False
False      5d22h
kube-apiserver                             4.3.9     True        False
False      5d22h
kube-controller-manager                    4.3.9     True        False
False      5d22h
kube-scheduler                             4.3.9     True        False
False      5d22h
machine-api                                4.3.9     True        False
False      5d22h
```

```
machine-config                             4.3.9     True        False
False      2d
marketplace                                4.3.9     True        False
False      46h
monitoring                                 4.3.9     True        False
False      8h
network                                    4.3.9     True        False
False      5d22h
node-tuning                                4.3.9     True        False
False      2d
openshift-apiserver                        4.3.9     True        False
False      8h
openshift-controller-manager               4.3.9     True        False
False      5d16h
openshift-samples                          4.3.9     True        False
False      45h
operator-lifecycle-manager                 4.3.9     True        False
False      5d22h
operator-lifecycle-manager-catalog         4.3.9     True        False
False      5d22h
operator-lifecycle-manager-packageserver   4.3.9     True        False
False      4d14h
service-ca                                 4.3.9     True        False
False      5d22h
service-catalog-apiserver                  4.3.9     True        False
False      5d22h
service-catalog-controller-manager         4.3.9     True        False
False      5d22h
storage                                    4.3.9     True        False
False      5d22h
```

Also, verify that all of the operators are downloaded, as shown in Example 3-28. (The download process might take hours to complete.)

Example 3-28 Verify downloaded operators

```
$ oc get packagemanifests -n openshift-marketplace
NAME                                        CATALOG               AGE
crunchy-postgres-operator                   Certified Operators   47h
csi-operator                                Community Operators   47h
cockroachdb-certified                       Certified Operators   47h
sematext                                    Certified Operators   47h
strimzi-kafka-operator                      Community Operators   47h
jaeger                                      Community Operators   47h
3scale-operator                             Red Hat Operators     5d14h
tigera-operator                             Certified Operators   47h
perceptilabs-operator-package               Certified Operators   47h
kubefed                                     Community Operators   47h
joget-dx-operator                           Certified Operators   47h
ocs-operator                                Red Hat Operators     5d14h
citrix-adc-istio-ingress-gateway-operator   Certified Operators   47h
spark-gcp                                   Community Operators   47h
argocd-operator                             Community Operators   47h
citrix-cpx-istio-sidecar-injector-operator  Certified Operators   47h
event-streams-topic                         Community Operators   47h
radanalytics-spark                          Community Operators   47h
```

```
planetscale                                 Community Operators   47h
nuodb-ce-certified                          Certified Operators   47h
open-enterprise-spinnaker                   Certified Operators   47h
ibm-platform-api-operator-app               Certified Operators   47h
openunison-ocp-certified                    Certified Operators   47h
here-service-operator-certified             Certified Operators   47h
namespace-configuration-operator            Community Operators   47h
syndesis                                    Community Operators   47h
enmasse                                     Community Operators   47h
metering-ocp                                Red Hat Operators     5d14h
appsody-operator-certified                  Certified Operators   47h
resource-locker-operator                    Community Operators   47h
nxrm-operator-certified                     Certified Operators   47h
csi-driver-operator                         Certified Operators   47h
cam-operator                                Red Hat Operators     5d14h
planetscale-certified                       Certified Operators   47h
ibm-block-csi-operator                      Certified Operators   47h
multicluster-operators-subscription         Community Operators   47h
lib-bucket-provisioner                      Community Operators   47h
openshiftartifactoryha-operator             Certified Operators   47h
iot-simulator                               Community Operators   47h
cert-utils-operator                         Community Operators   47h
must-gather-operator                        Community Operators   47h
amq7-cert-manager                           Red Hat Operators     5d14h
elasticsearch-operator                      Red Hat Operators     5d14h
zabbix-operator-certified                   Certified Operators   47h
couchdb-operator-certified                  Certified Operators   47h
newrelic-infrastructure                     Certified Operators   47h
traefikee-operator                          Community Operators   47h
amq-online                                  Red Hat Operators     5d14h
seldon-operator                             Community Operators   47h
openshiftansibleservicebroker               Red Hat Operators     5d14h
portworx-certified                          Certified Operators   47h
crossplane                                  Community Operators   47h
postgresql-operator-dev4devs-com            Community Operators   47h
oneagent-certified                          Certified Operators   47h
f5-bigip-ctlr-operator                      Certified Operators   47h
ibmcloud-operator                           Community Operators   47h
argocd-operator-helm                        Community Operators   47h
keepalived-operator                         Community Operators   47h
uma-operator                                Certified Operators   47h
percona-xtradb-cluster-operator-certified   Certified Operators   47h
ocean-operator                              Certified Operators   47h
kube-arangodb                               Certified Operators   47h
traefikee-redhat-certified                  Certified Operators   47h
triggermesh                                 Community Operators   47h
dv-operator                                 Red Hat Operators     5d14h
appranix-cps                                Certified Operators   47h
atlasmap-operator                           Community Operators   47h
grafana-operator                            Community Operators   47h
maistraoperator                             Community Operators   47h
datagrid                                    Red Hat Operators     5d14h
ptp-operator                                Red Hat Operators     5d14h
jaeger-product                              Red Hat Operators     5d14h
kubevirt-hyperconverged                     Red Hat Operators     5d14h
```

```
descheduler                                 Community Operators   47h
ibm-spectrum-scale-csi-operator             Community Operators   47h
t8c                                         Community Operators   47h
quay                                        Community Operators   47h
microsegmentation-operator                  Community Operators   47h
nexus-operator-hub                          Community Operators   47h
rapidbiz-operator-certified                 Certified Operators   47h
esindex-operator                            Community Operators   47h
halkyon                                     Community Operators   47h
twistlock                                   Community Operators   47h
submariner                                  Community Operators   47h
keda                                        Community Operators   47h
skydive-operator                            Community Operators   47h
amq-streams                                 Red Hat Operators     5d14h
nginx-ingress-operator                      Certified Operators   47h
knative-camel-operator                      Community Operators   47h
horreum-operator                            Community Operators   47h
sriov-network-operator                      Red Hat Operators     5d14h
ubix-operator                               Certified Operators   47h
wavefront-operator                          Certified Operators   47h
etcd                                        Community Operators   47h
3scale-community-operator                   Community Operators   47h
fuse-apicurito                              Red Hat Operators     5d14h
robin-operator                              Certified Operators   47h
memql-certified                             Certified Operators   47h
hawtio-operator                             Community Operators   47h
kubeturbo                                   Community Operators   47h
nsm-operator-registry                       Community Operators   47h
kubestone                                   Community Operators   47h
ibm-spectrum-scale-csi                      Certified Operators   47h
transadv-operator                           Certified Operators   47h
storageos-1Otb                              Certified Operators   47h
federation                                  Community Operators   47h
servicemeshoperator                         Red Hat Operators     5d14h
local-storage-operator                      Red Hat Operators     5d14h
cortex-operator                             Certified Operators   47h
orca                                        Certified Operators   47h
percona-server-mongodb-operator-certified   Certified Operators   47h
t8c-certified                               Certified Operators   47h
synopsys-certified                          Certified Operators   47h
hyperfoil-bundle                            Community Operators   47h
openshift-pipelines-operator                Community Operators   47h
amq7-interconnect-operator                  Red Hat Operators     5d14h
apicast-operator                            Red Hat Operators     5d14h
datadog-operator-certified                  Certified Operators   47h
myvirtualdirectory                          Community Operators   47h
aqua-operator-certified                     Certified Operators   47h
ibm-auditlogging-operator-app               Certified Operators   47h
ivory-server-app                            Certified Operators   47h
aqua                                        Community Operators   47h
redis-enterprise-operator-cert              Certified Operators   47h
awss3-operator-registry                     Community Operators   47h
ibm-monitoring-grafana-operator-app         Certified Operators   47h
federatorai                                 Community Operators   47h
api-operator                                Community Operators   47h
```

```
prometheus                                    Community Operators   47h
kong                                          Certified Operators   47h
couchbase-enterprise-certified                Certified Operators   47h
cortex-certifai-operator                      Certified Operators   47h
knative-eventing-operator                     Community Operators   47h
ember-csi-operator                            Community Operators   47h
hazelcast-jet-enterprise-operator             Certified Operators   47h
storageos-1tb                                 Certified Operators   47h
akka-cluster-operator                         Community Operators   47h
composable-operator                           Community Operators   47h
hazelcast-enterprise-certified                Certified Operators   47h
microcks                                      Community Operators   47h
openshifttemplateservicebroker                Red Hat Operators     5d14h
sysdig-certified                              Certified Operators   47h
dotscience-operator                           Certified Operators   47h
ibm-metering-operator-app                     Certified Operators   47h
kubemq-operator                               Certified Operators   47h
konveyor-operator                             Community Operators   47h
businessautomation-operator                   Red Hat Operators     5d14h
kubeturbo-certified                           Certified Operators   47h
joget-openshift-operator                      Certified Operators   47h
tidb-operator-certified                       Certified Operators   47h
runtime-component-operator-certified          Certified Operators   47h
redis-operator                                Community Operators   47h
opsmx-spinnaker-operator                      Community Operators   47h
eclipse-che                                   Community Operators   47h
appdynamics-operator                          Certified Operators   47h
openebs                                       Community Operators   47h
portshift-operator                            Certified Operators   47h
teiid                                         Community Operators   47h
neuvector-certified-operator                  Certified Operators   47h
ibm-helm-repo-operator-app                    Certified Operators   47h
akka-cluster-operator-certified               Certified Operators   47h
special-resource-operator                     Community Operators   47h
cockroachdb                                   Community Operators   47h
kiali                                         Community Operators   47h
neuvector-community-operator                  Community Operators   47h
cluster-logging                               Red Hat Operators     5d14h
kubeturbo-marketplace-certified               Certified Operators   47h
aqua-certified                                Certified Operators   47h
ripsaw                                        Community Operators   47h
ibm-mongodb-operator-app                      Certified Operators   47h
postgresql                                    Community Operators   47h
jenkins-operator                              Community Operators   47h
codeready-workspaces                          Red Hat Operators     5d14h
federatorai-certified                         Certified Operators   47h
cic-operator                                  Certified Operators   47h
node-problem-detector                         Community Operators   47h
opendatahub-operator                          Community Operators   47h
amq-broker                                    Red Hat Operators     5d14h
driverlessai-deployment-operator-certified    Certified Operators   47h
open-liberty-certified                        Certified Operators   47h
service-binding-operator                      Community Operators   47h
instana-agent                                 Certified Operators   47h
eddi-operator-certified                       Certified Operators   47h
```

```
metering                                  Community Operators   47h
infinispan                                Community Operators   47h
insightedge-enterprise-operator2          Certified Operators   47h
egressip-ipam-operator                    Community Operators   47h
kiali-ossm                                Red Hat Operators     5d14h
serverless-operator                       Red Hat Operators     5d14h
k8s-triliovault                           Certified Operators   47h
presto-operator                           Certified Operators   47h
lightbend-console-operator                Community Operators   47h
apicurito                                 Community Operators   47h
apicast-community-operator                Community Operators   47h
fuse-online                               Red Hat Operators     5d14h
nfd                                       Red Hat Operators     5d14h
ibm-helm-api-operator-app                 Certified Operators   47h
ibm-management-ingress-operator-app       Certified Operators   47h
cost-mgmt-operator                        Community Operators   47h
snyk-operator                             Community Operators   47h
eap                                       Red Hat Operators     5d14h
keycloak-operator                         Community Operators   47h
camel-k                                   Community Operators   47h
container-security-operator               Community Operators   47h
kogito-operator                           Community Operators   47h
cpx-cic-operator                          Certified Operators   47h
twistlock-certified                       Certified Operators   47h
storageos                                 Certified Operators   47h
spinnaker-operator                        Community Operators   47h
codeready-toolchain-operator              Community Operators   47h
knative-kafka-operator                    Community Operators   47h
mongodb-enterprise                        Certified Operators   47h
anchore-engine                            Certified Operators   47h
seldon-operator-certified                 Certified Operators   47h
traefikee-certified                       Certified Operators   47h
```

The installation of Red Hat OpenShift Container Platform V4.3 deployment on IBM Cloud bare metal machines is complete.

At this web page, log in as the admin user to validate the credentials and in preparation for administration tasks.

The next section provides a few administration recommendations for Red Hat OpenShift Container.

3.3.7 Administration of Red Hat OpenShift Container

This section examines the administration of a deployed Red Hat OpenShift cluster. This section describes the Administrator Console, Developer Console and Cluster Monitoring Console, including the Alert Manager, Prometheus, and Grafana dashboards.

Red Hat OpenShift Administrator Console walkthrough

Complete the following steps:

1. Open a Web browser to this web page.

2. Log in by using the admin user in the prompt, as shown in Figure 3-35.

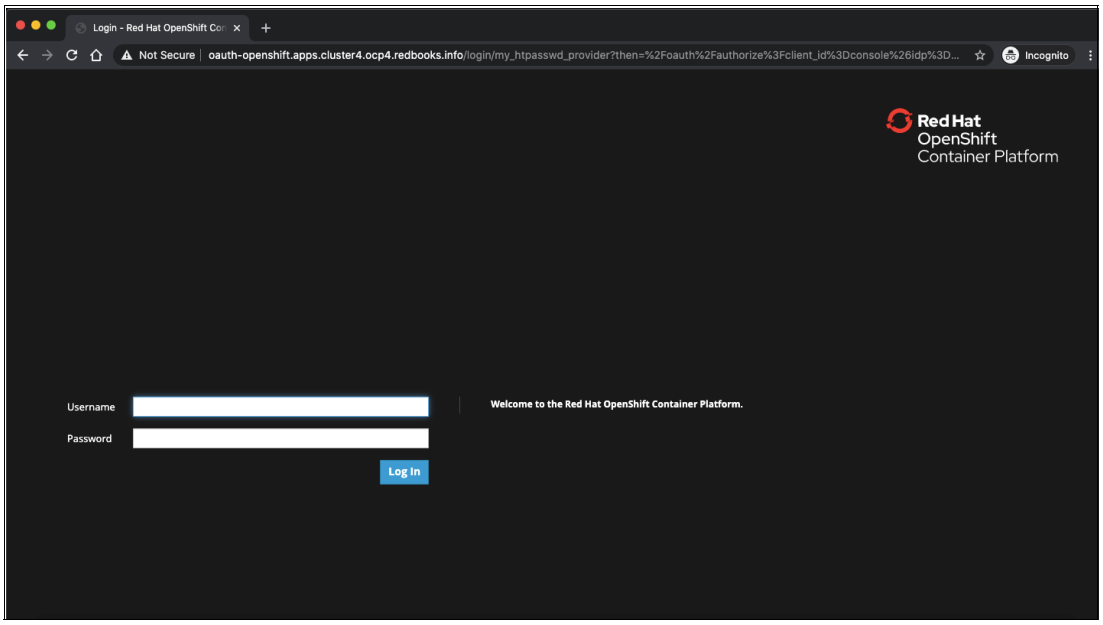


Figure 3-35 Red Hat OpenShift console login

3. The dashboard page of the Red Hat OpenShift console opens. Select the wanted console from the upper left side of the window as Administrator or Developer. You see the Administrator console of the Red Hat OpenShift Container Platform, as shown in Figure 3-36.

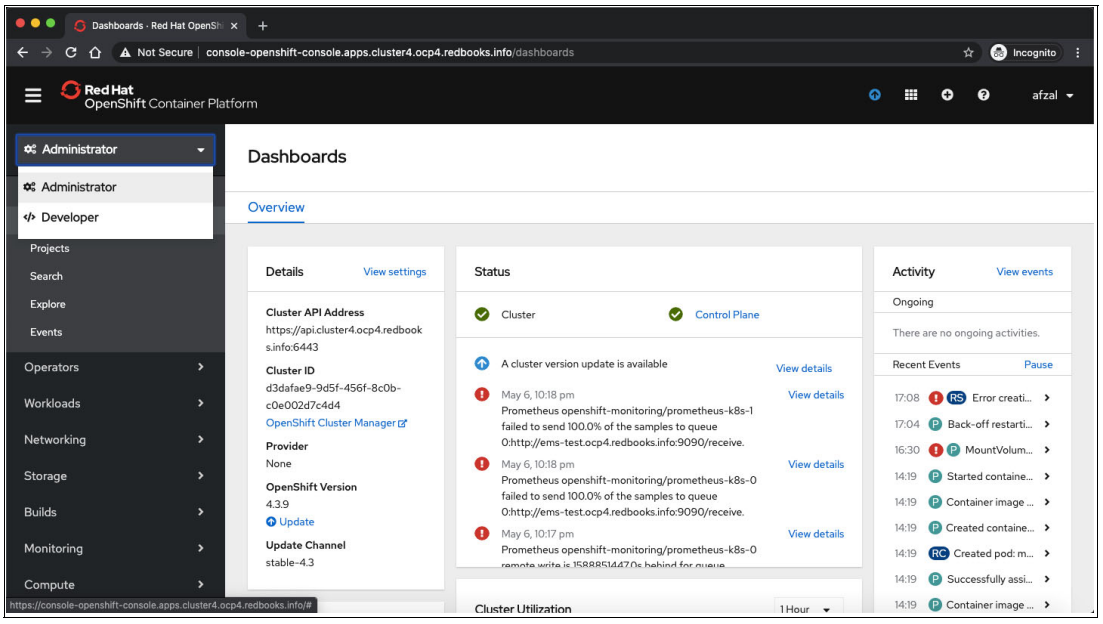


Figure 3-36 Red Hat OpenShift Dashboard Profile Selection

Figure 3-37 shows the cluster details, such as Cluster API Address, Cluster ID, Version, and Release. The status of the cluster is shown in the center of the window; the right side of the page shows all the activities and events that are performed on the cluster.

Browsing through the cluster, the Control pane on the left helps you manage the following components:

- Operators framework
- Workloads object of the cluster
- Networking of Red Hat OpenShift cluster and related objects
- Storage creation and assignment
- CI/CD, pipeline under builds
- Monitoring of Red Hat OpenShift
- Compute resources details
- User Management
- Administration like quotas, settings of cluster

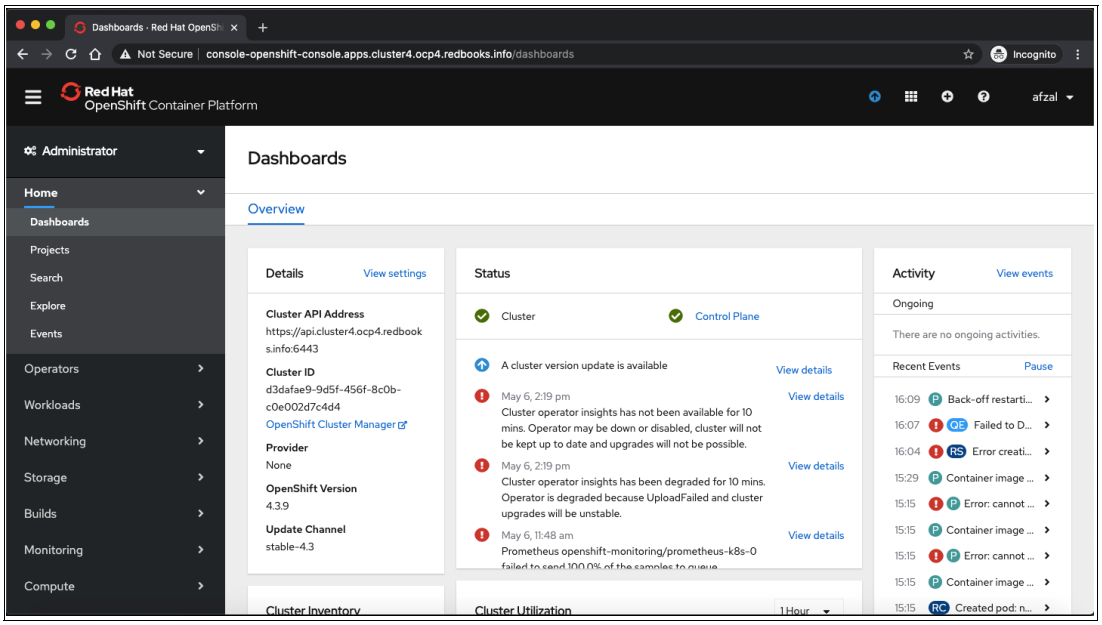


Figure 3-37 Red Hat OpenShift Dashboard

As shown in Figure 3-38, the bottom of the dashboard provides high-level view of Cluster Inventory and its objects. Cluster utilization information also is available.

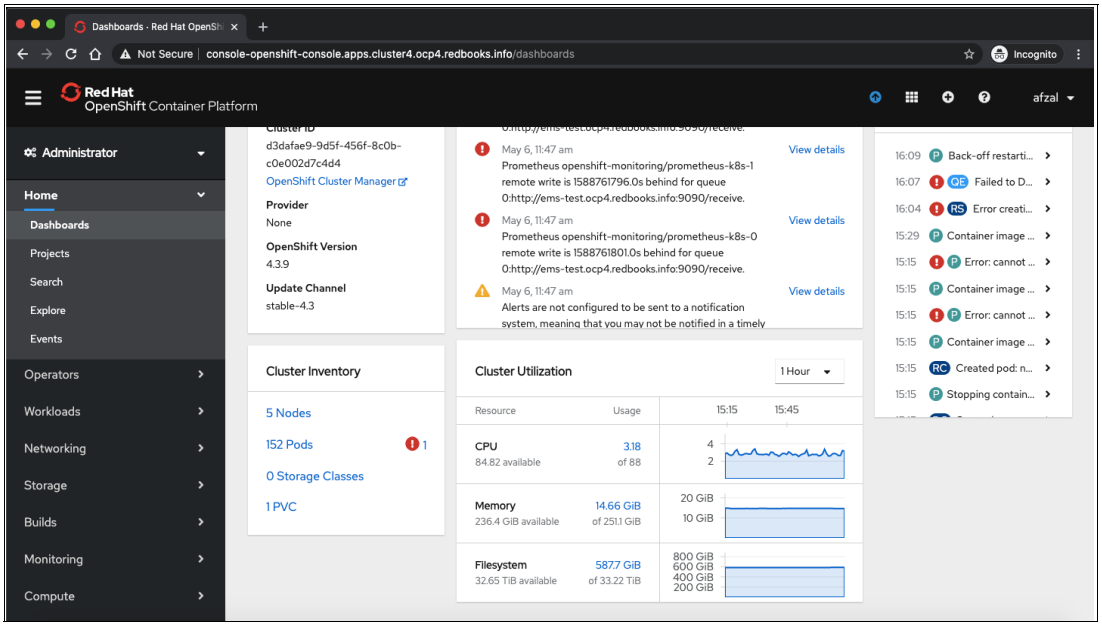


Figure 3-38 Red Hat OpenShift Dashboard continues

4. Click **Projects** (see Figure 3-39) under Home in the left control pane to list all of the available projects of Red Hat OpenShift. You can also create a project by clicking **Create Project** and entering a project name.

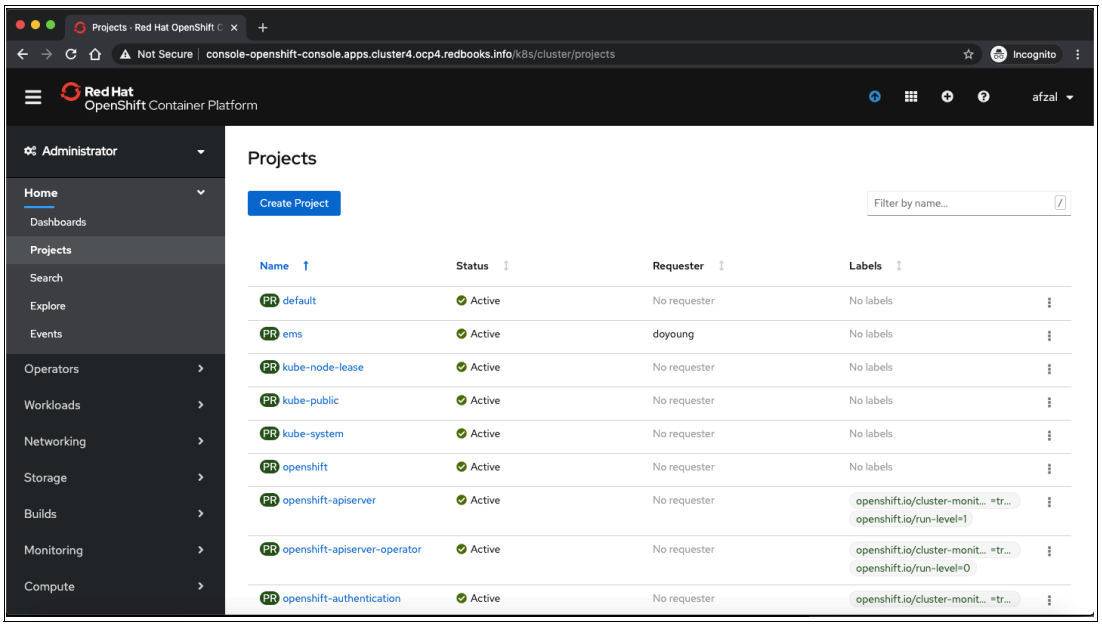


Figure 3-39 Projects view for the Red Hat OpenShift cluster

5. Click **Operators** in the left control pane, as shown in Figure 3-40. Then, click **OperatorHub** to install the required operator within the project.

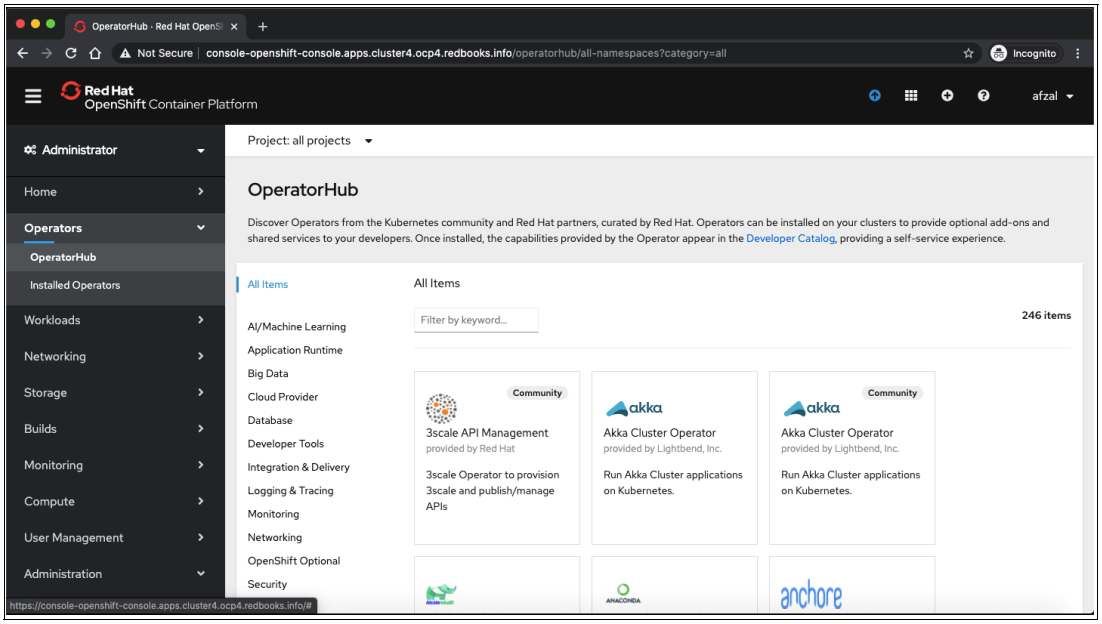


Figure 3-40 OperatorHub within the Red Hat OpenShift cluster

6. Click **Operators** on the left control pane and then, click **Installed Operators** to view all of the installed operators for the project and namespace, as shown in Figure 3-41.

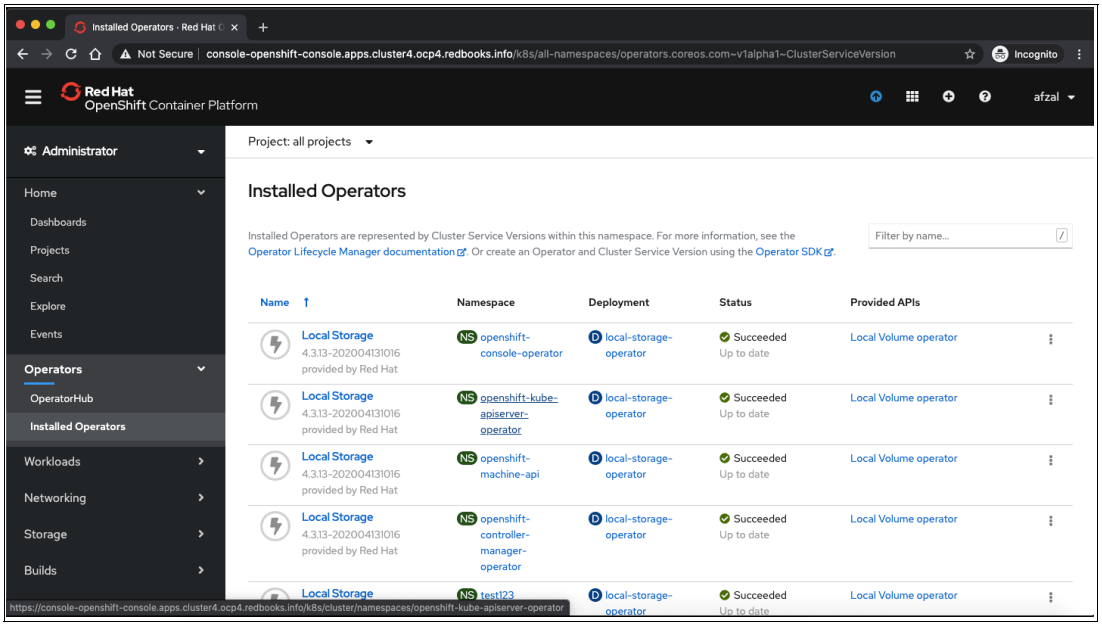


Figure 3-41 Installed Operators in the Red Hat OpenShift cluster

7. You see a list of the installed Operators. Review the deployments in the Red Hat OpenShift project. Click **Workloads** → **Deployments** to view the deployments that were used in specific project name spaces, as shown in Figure 3-42.

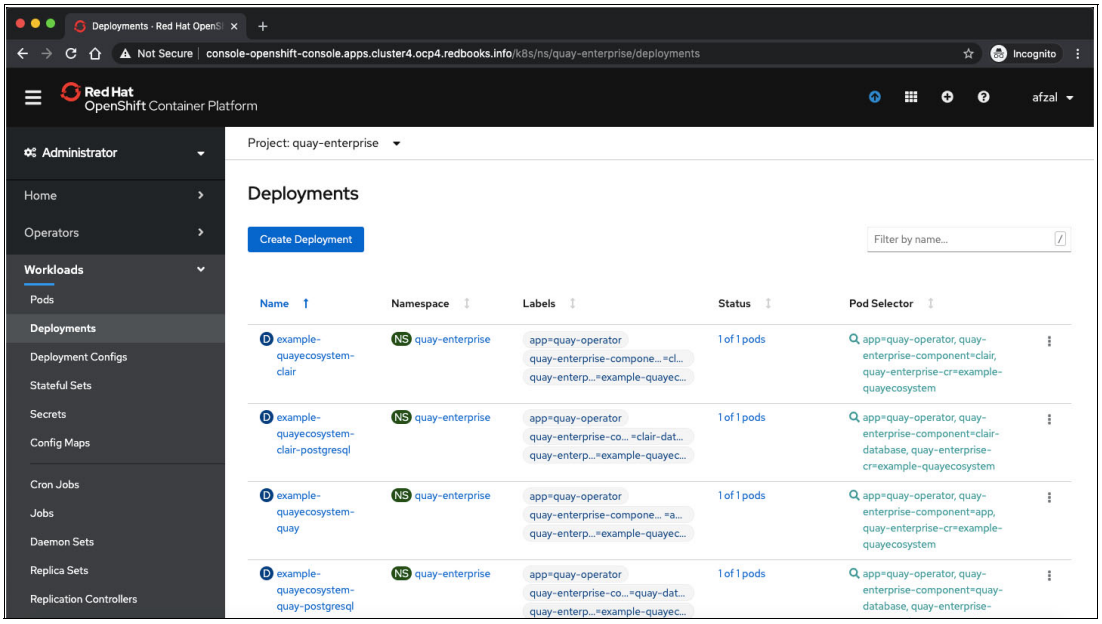


Figure 3-42 Deployments on Red Hat OpenShift

8. To see the deployment configuration map of the installed deployment (as shown in Figure 3-43), select **Workloads** on the left control pane and then, select **Config Maps** → **Config map for details** → **YAML** tab.

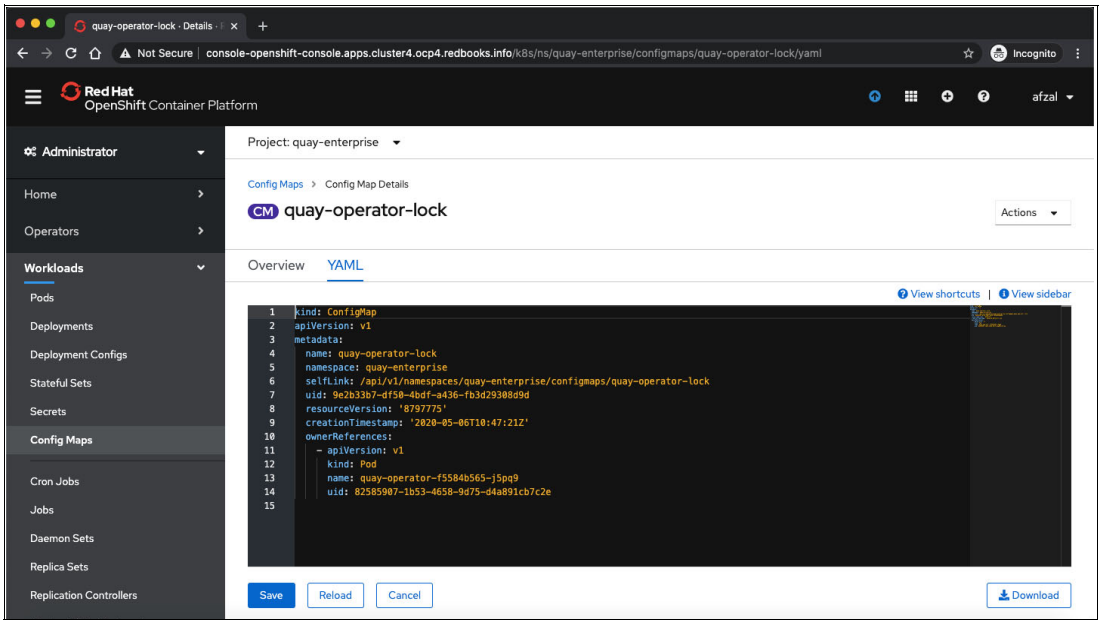


Figure 3-43 Configuration map of deployment on Red Hat OpenShift cluster

9. To view the routes of the deployments as shown in Figure 3-44, select **Networking** from the left control pane and then, click **Routes** to list all of the routes that are associated with a project. You can create a route by clicking **Create Route** for a deployment.

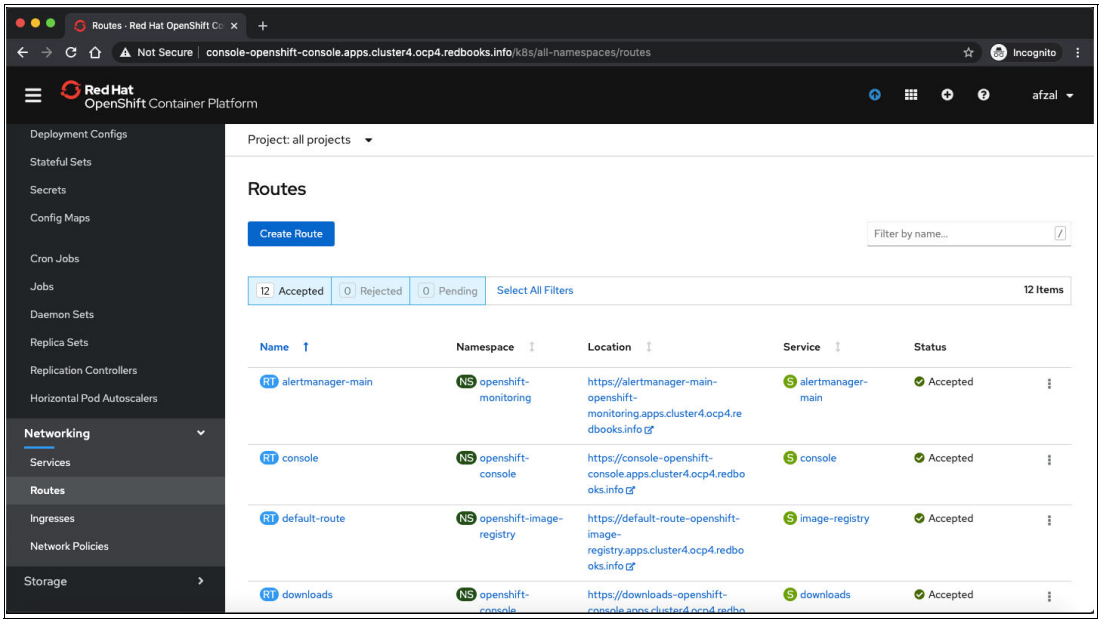


Figure 3-44 Routes of Red Hat OpenShift Deployment

10. Select **Services** in the Networking tab on the left side control pane to list all of the deployed services on the Red Hat OpenShift cluster, as shown in Figure 3-45. You can create a service by selecting **Create Service**. Then, assign a label by selecting the deployment and then, click **Create**.

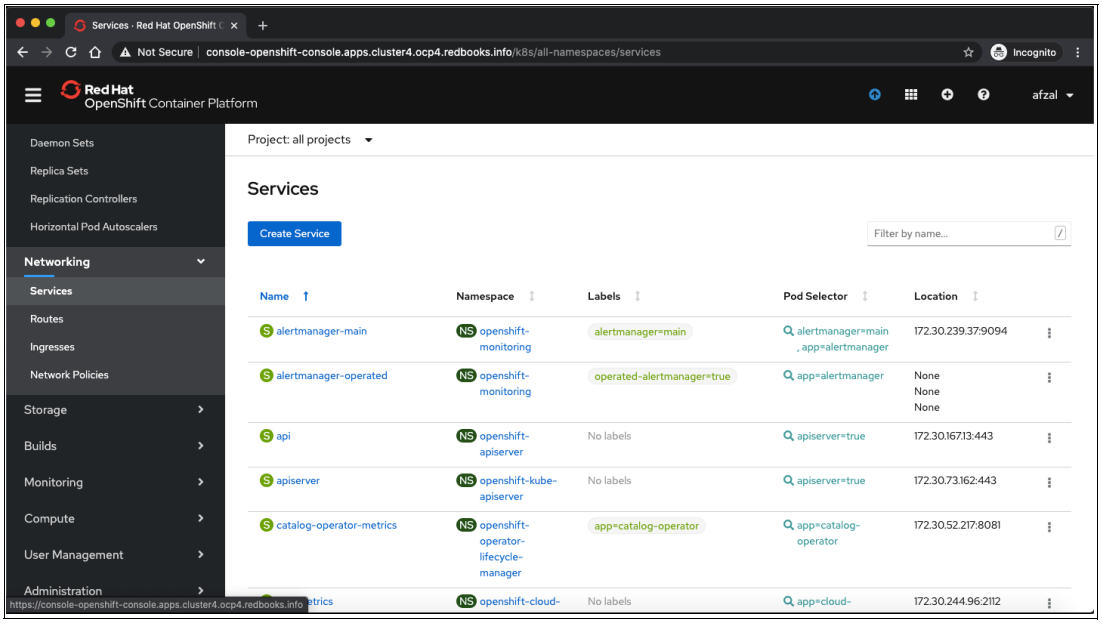


Figure 3-45 Services create and expose Red Hat OpenShift deployment

11. Configure the Red Hat OpenShift cluster storage by selecting the **Storage** tab and then click **Persistent Volumes**, as shown in Figure 3-46. You can also create a Persistent Volume claim for the project.

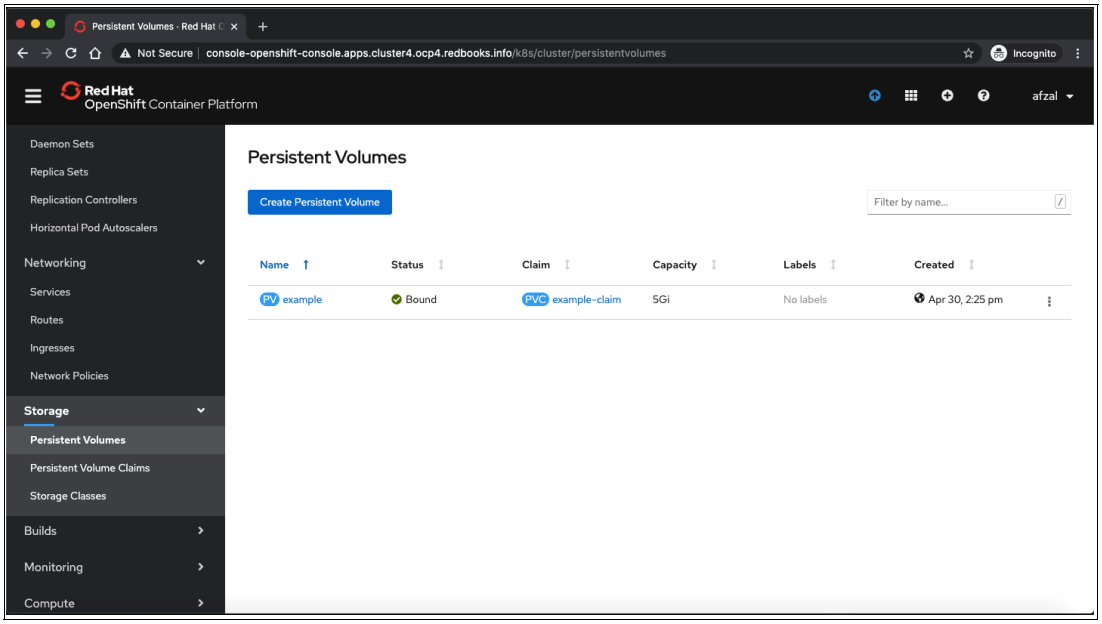


Figure 3-46 Storage PV creation and list of Red Hat OpenShift deployment

12. To administer cluster nodes, select the **Compute** option in the left control pane and then, click **Nodes**. You see a list of all the nodes that are associated with the deployed Red Hat OpenShift cluster, as shown in Figure 3-47.

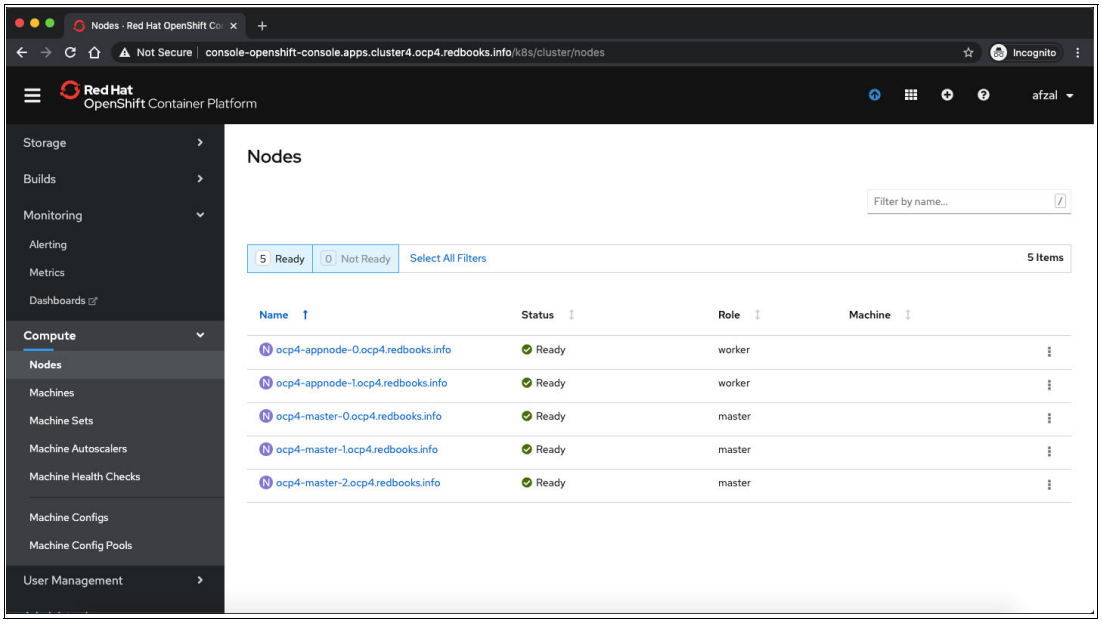


Figure 3-47 Nodes list associated to Red Hat OpenShift cluster

13. Click any node, in this case, we clicked **App Node 0**. The Node Overview page opens, as shown in Figure 3-48, which includes all of the health monitors of the node and information about deployments that are associated to that specific node.

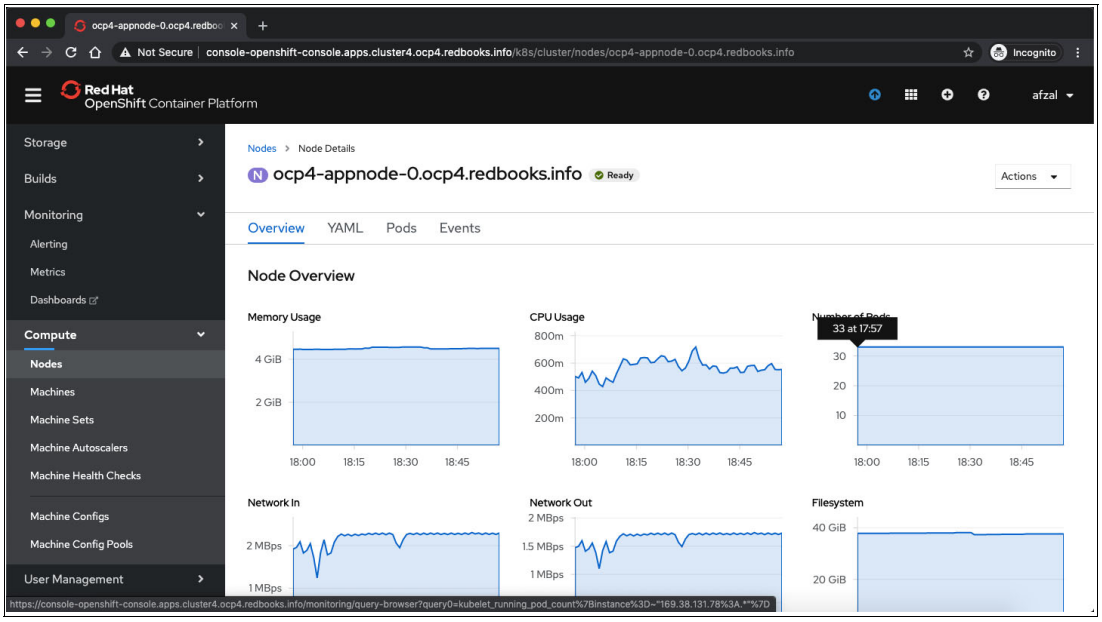


Figure 3-48 App Node details

14. To list the associated Service Account or to create one, select **User Management** in the left control pane and select **Service Accounts**. All of the associated service accounts with a project are listed, as shown in Figure 3-49. You can create a Service Account by clicking **Create Service Account**.

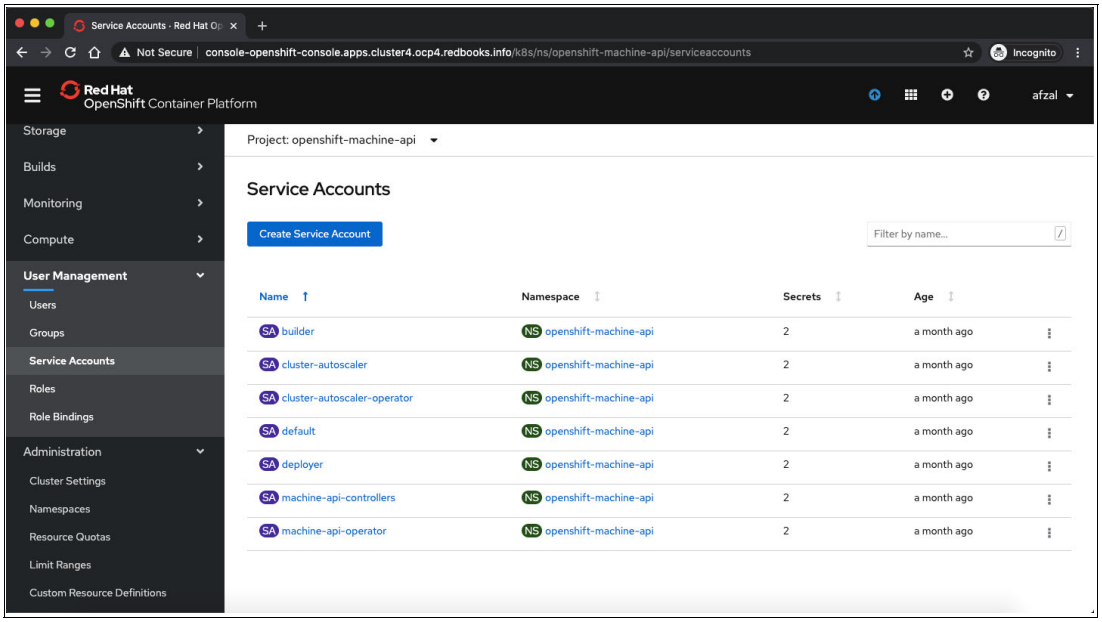


Figure 3-49 List of Service Accounts

You also can review settings that are associated with Name spaces, Cluster Settings, Resource Quotas, Limit Ranges, and CRDs from the Administration option in left control pane.

Red Hat OpenShift Monitoring Console overview

This section reviews the monitoring tools within Red Hat OpenShift. As a complete monitoring solution, Red Hat OpenShift includes the following tools:

- Grafana for Visualization
- Prometheus for Log Management
- Alert Manager for Alert Management

Grafana

Complete the following steps:

1. From the left control pane, select **Monitoring**; then, click **Grafana**. The Grafana visualization dashboard of Red Hat OpenShift opens. Log in to Grafana with same credentials that you use for Red Hat OpenShift Container Platform.
2. Select the suitable dashboard or you can create a custom dashboard per your requirements.

   Figure 3-50 shows a default dashboard that is available in Grafana. This dashboard uses Kubernetes, Compute Resources, and Cluster dashboard to visualize the resource use of the deployed Red Hat OpenShift cluster.

   These dashboards provides the following information (see Figure 3-51 on page 112 and Figure 3-52 on page 112):

   - Resources Headlines
   - CPU Utilization
   - Memory Utilization
   - Network Utilization
   - Bandwidth Utilization
   - Rate of packets on network

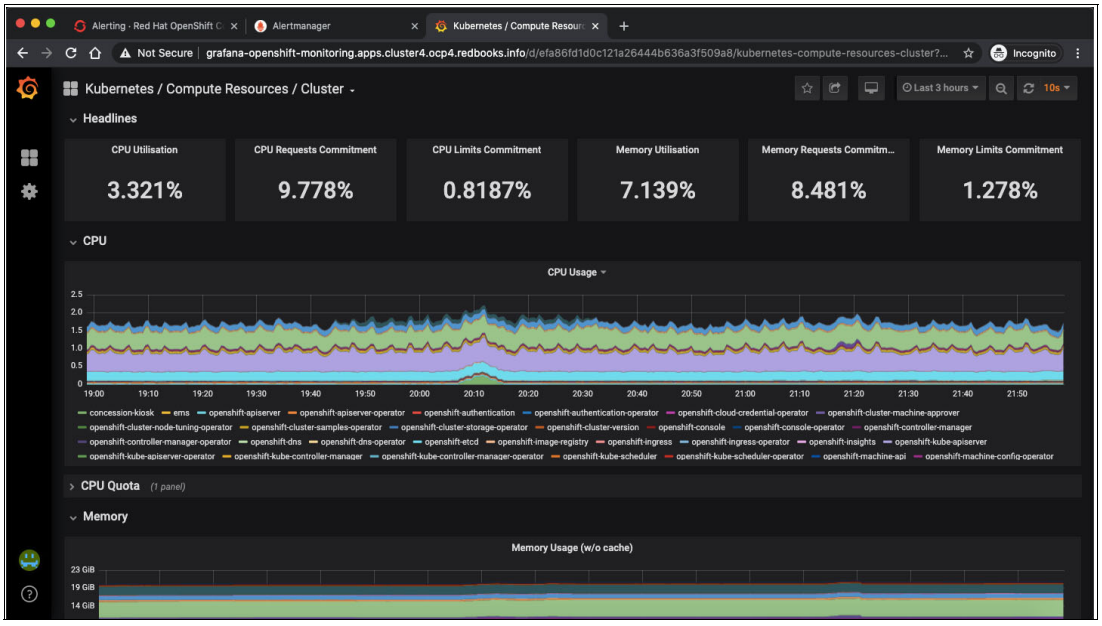


Figure 3-50 Preconfigured Grafana dashboard: CPU and memory

Figure 3-51 Pre-configured Grafana dashboard: Network

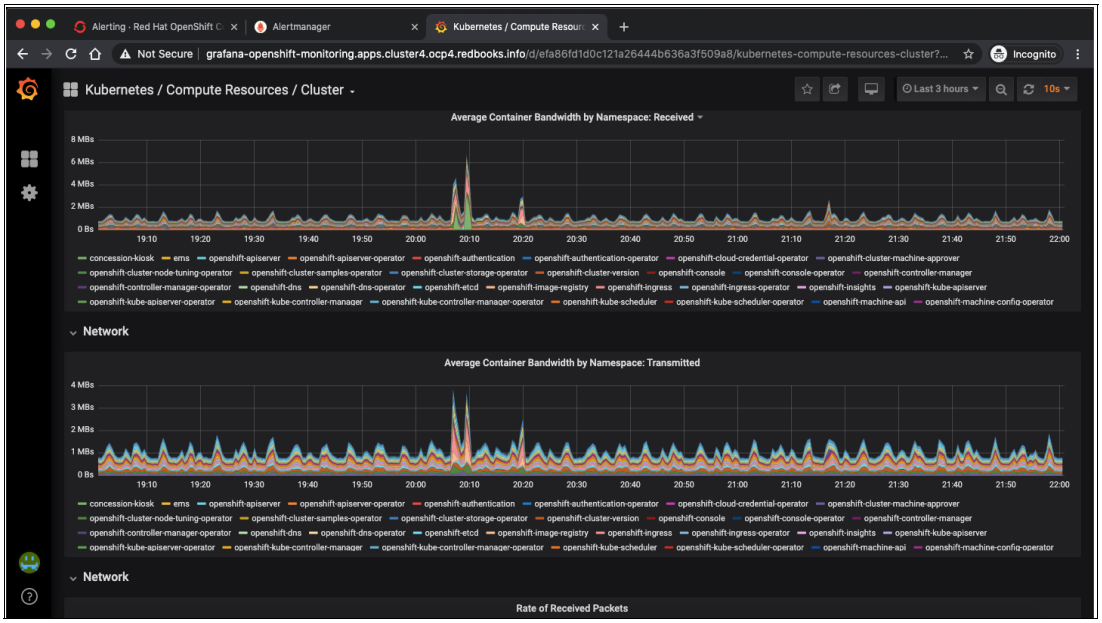


Figure 3-52 Pre-configured Grafana dashboard: Container bandwidth

Prometheus and alert manager

Complete the following steps:

1. To open the Prometheus console, select the **Prometheus** option from the **Monitoring** tab in left control pane. Trigger a sample predefined query to see graph of the alerts, as shown in Figure 3-53.

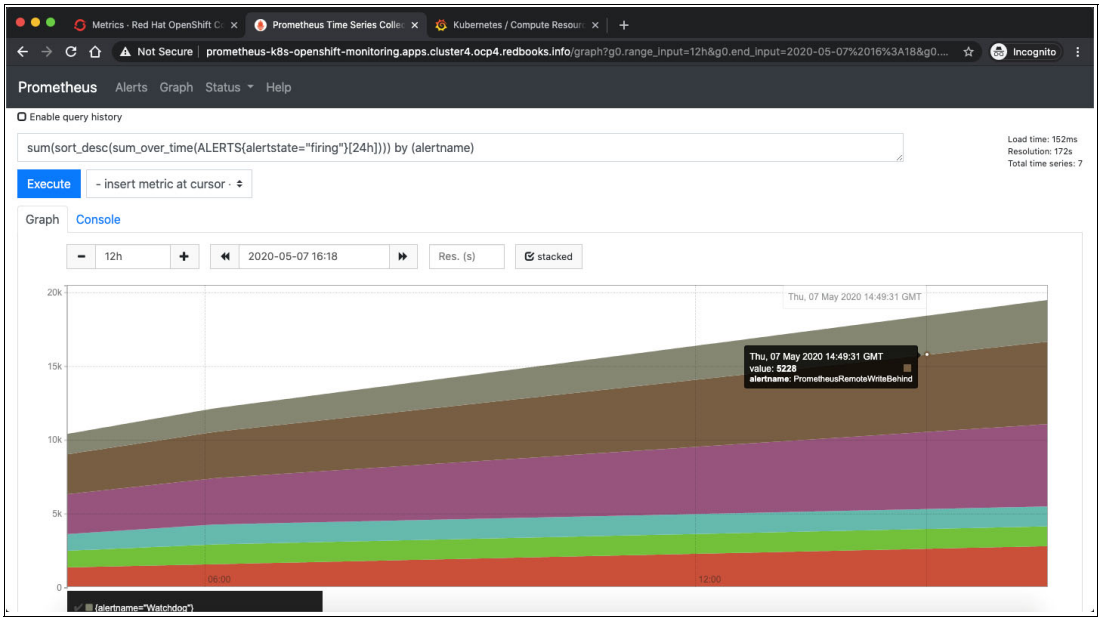


Figure 3-53 Pre-configured Prometheus dashboard

2. Select the **Alerts** option that is available in tabs and look for all of the alerts that were captured by Prometheus that are received from the alert manager, as shown in Figure 3-54.

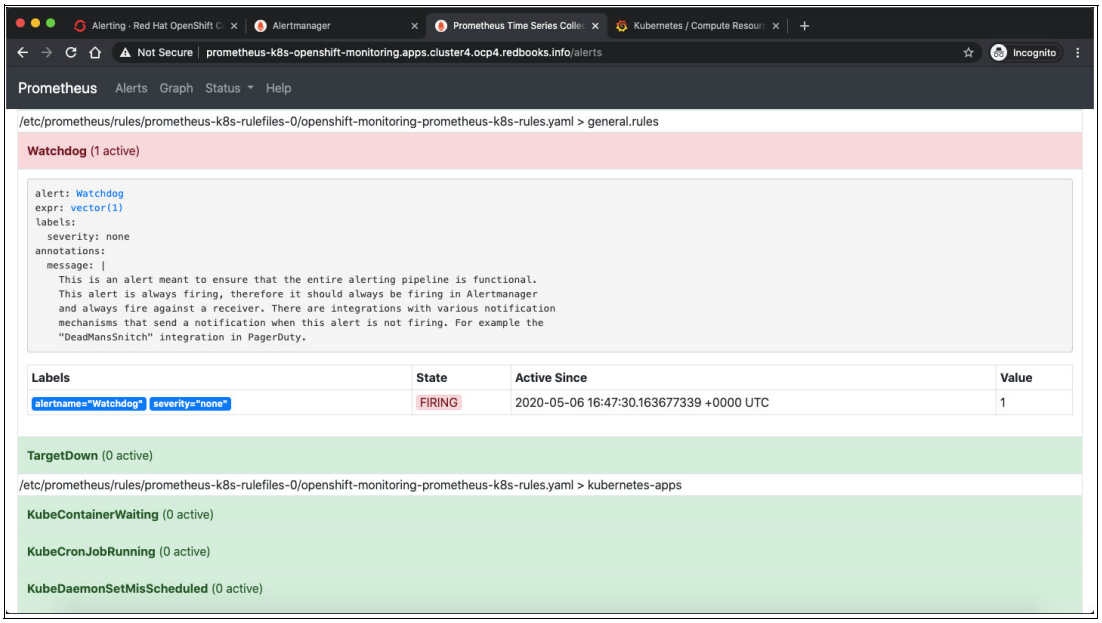


Figure 3-54 Prometheus alert page

3. From the Status tab drop-down list, select **Targets** to view all of the target systems, as shown in Figure 3-55.

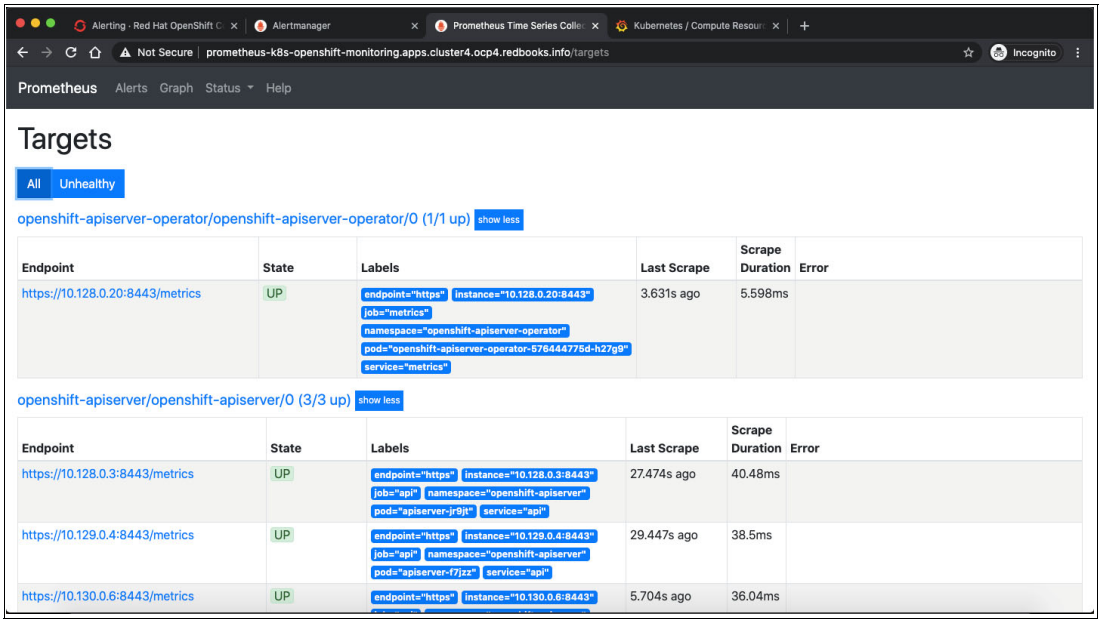


Figure 3-55 Targets view in Prometheus

4. From the **Status** tab, select the **Service Discovery** option to review the services that are available for discovery or being discovered, as shown in Figure 3-56.

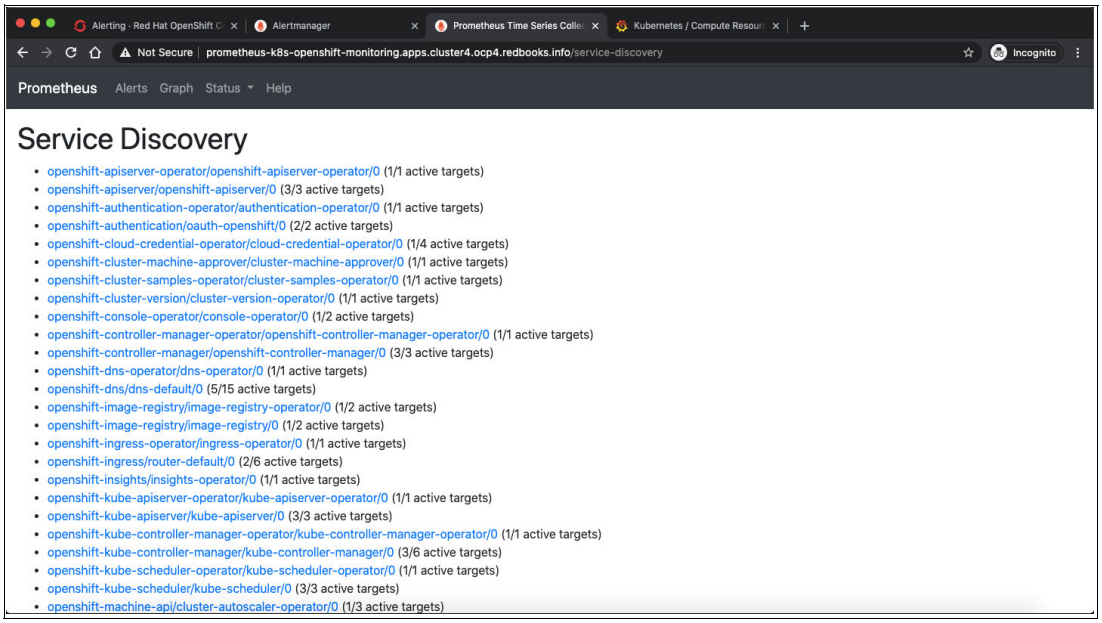


Figure 3-56 Service Discovery on Prometheus

Figure 3-57 shows Alertmanager of the deployed Red Hat OpenShift. You can browse to Alertmanager similar to Prometheus and review the captured alert of the system.

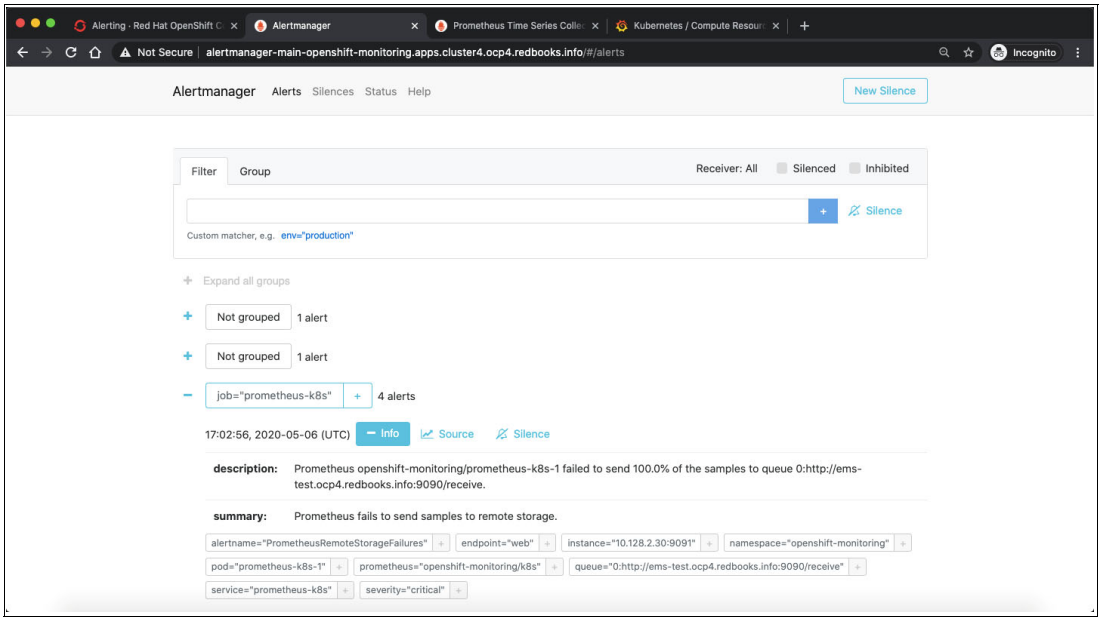


Figure 3-57 Alertmanager window

Red Hat OpenShift Developer Console overview

This section describes the Developer Console of Red Hat OpenShift Container Platform. This console helps developers to build and deploy applications.

Select the **Developer** option from upper left of the control pane, as shown in Figure 3-58. The Developer console opens.

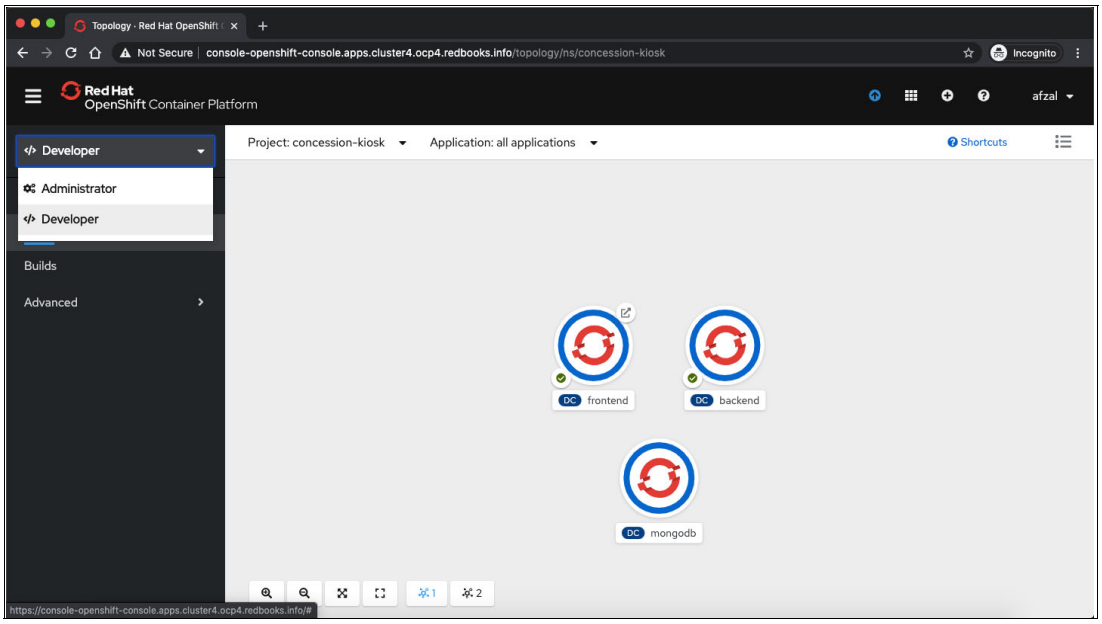


Figure 3-58 Browsing to the Developer console

After you are in the Developer console, you can see several options are available, such as Topology view of Deployed Project, Builds, and some advanced options, including Project Access, Metrics (TP).

Click **Topology** in the left control pane. You see the deployment topology of the project, where you can click the object to view details about the objects, as shown in Figure 3-59.

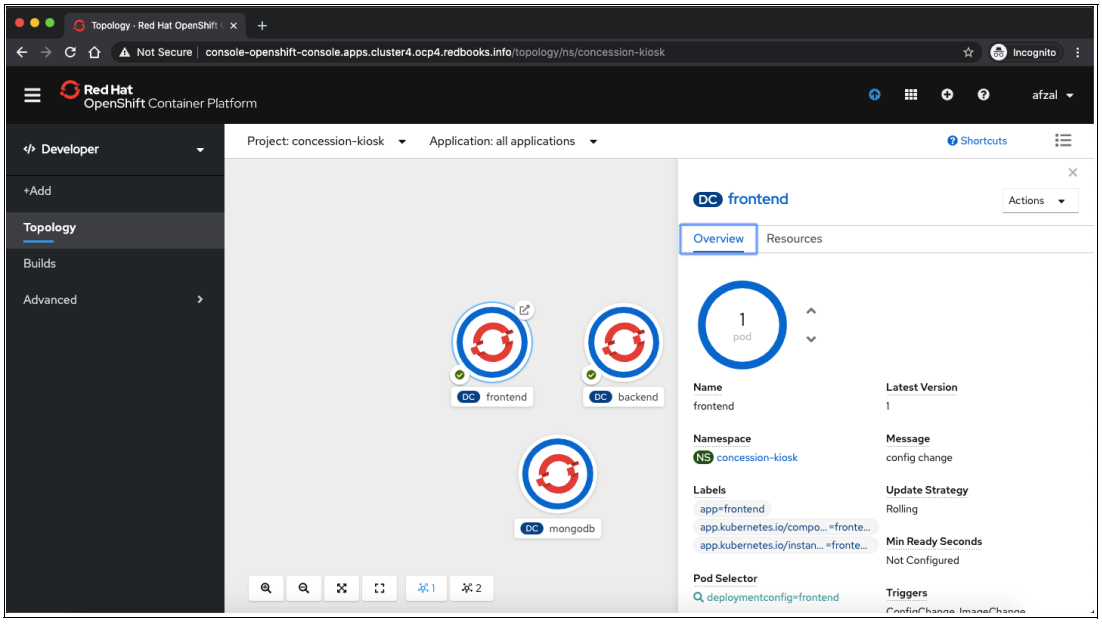


Figure 3-59 Project Topology view in Developer Console

Click the route of the deployed application (see Figure 3-60) to access it.

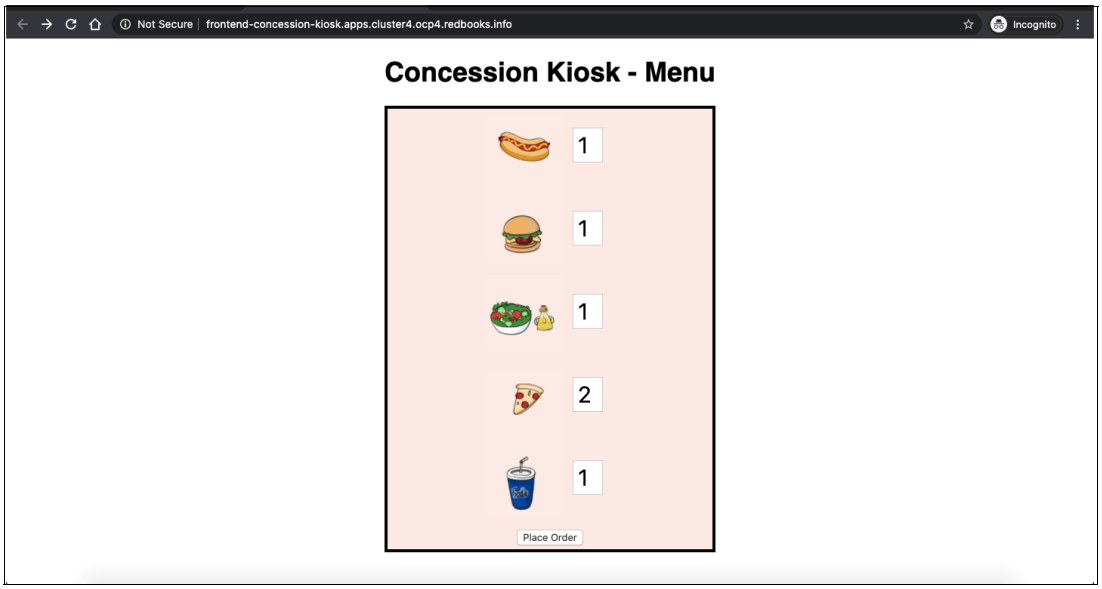


Figure 3-60 Deployed application in Red Hat OpenShift

Figure 3-61 shows the accessed deployed application.

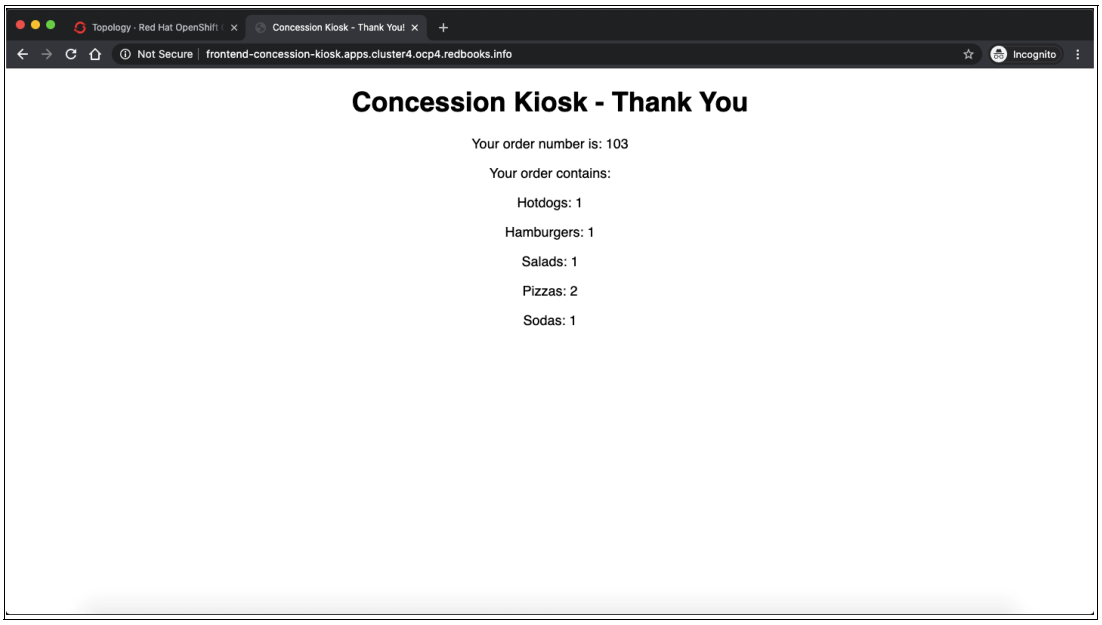


Figure 3-61 Deployed application in Red Hat OpenShift continues

Click **Builds** to see the Build configuration of the deployed project, as shown in Figure 3-62.

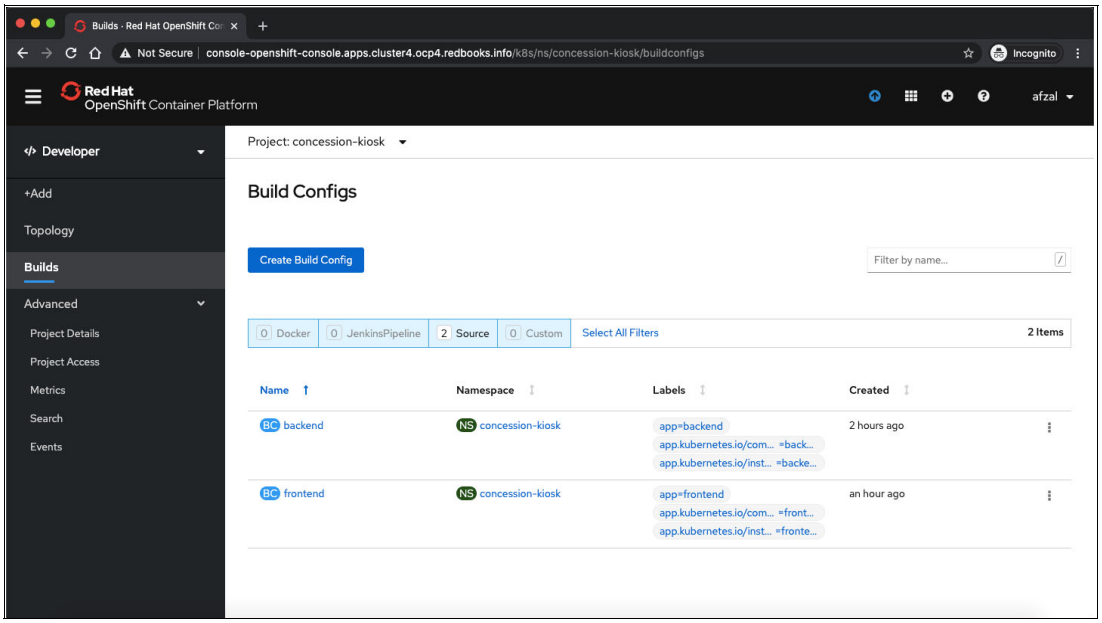


Figure 3-62 Build configuration of deployed project

Click **Advanced Option** → **Project Details** to view all the details about the project, including Inventory, Utilization, and Alerts, as shown in Figure 3-63.

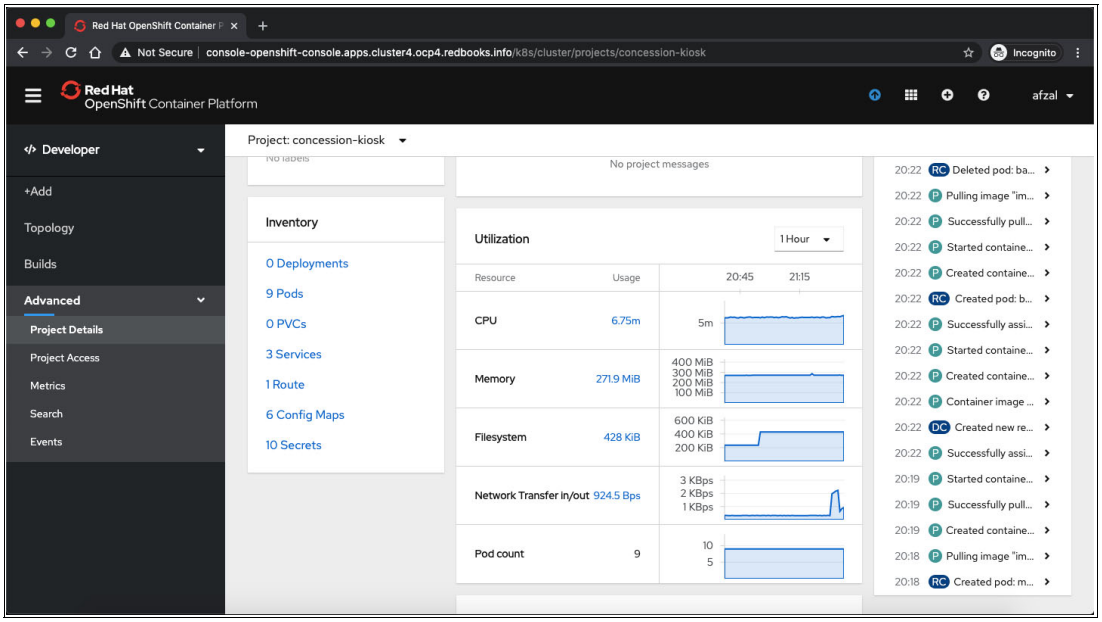


Figure 3-63 Project details page

3.4 Red Hat Quay

This section describes Red Hat Quay, which is a secure, private container registry that builds, analyzes, and distributes container images. It provides a high level of automation and customization. Red Hat Quay is also available as a hosted service called Quay.io.

3.4.1 Introduction

Red Hat Quay is an enterprise-quality container registry. Enterprises uses Quay to build and store containers, then deploy them to the servers across your enterprise. Quay is one essential component of as private registry for PaaS, where you store all your container images by taking advantage of docker hub.

Red Hat Quay includes the following features:

- High availability
- Geo-replication
- Repository mirroring
- Docker v2, schema 2 (multi-arch) support
- Continuous integration
- Security scanning with Clair
- Custom log rotation
- Zero downtime garbage collection
- 24/7 support

Red Hat Quay provides support for:

- Multiple authentication and access methods
- Multiple storage backends
- Custom certificates for Quay, Clair, and storage backends

- Application registries
- Different container image types

3.4.2 System architecture

This section describes the different components of Red Hat Quay and the architectural aspects of building a private registry within-build vulnerability scanner. This private registry Quay is made up of the following core components:

- Database: Used by Red Hat Quay as its primary metadata storage (not for image storage).
- Redis (key, value store): Stores live builder logs and the Red Hat Quay tutorial.
- Quay (container registry): Runs the quay container as a service, consisting of several components in the Pod.
- Clair: Scans container images for vulnerabilities and suggests fixes.

Figure 3-64 shows the different components of Quay and their relationship with connected PaaS components on the IBM Cloud Platform. For testing and demonstration purposes, we are deploying Red Hat Quay Private registry in a non-HA environment.

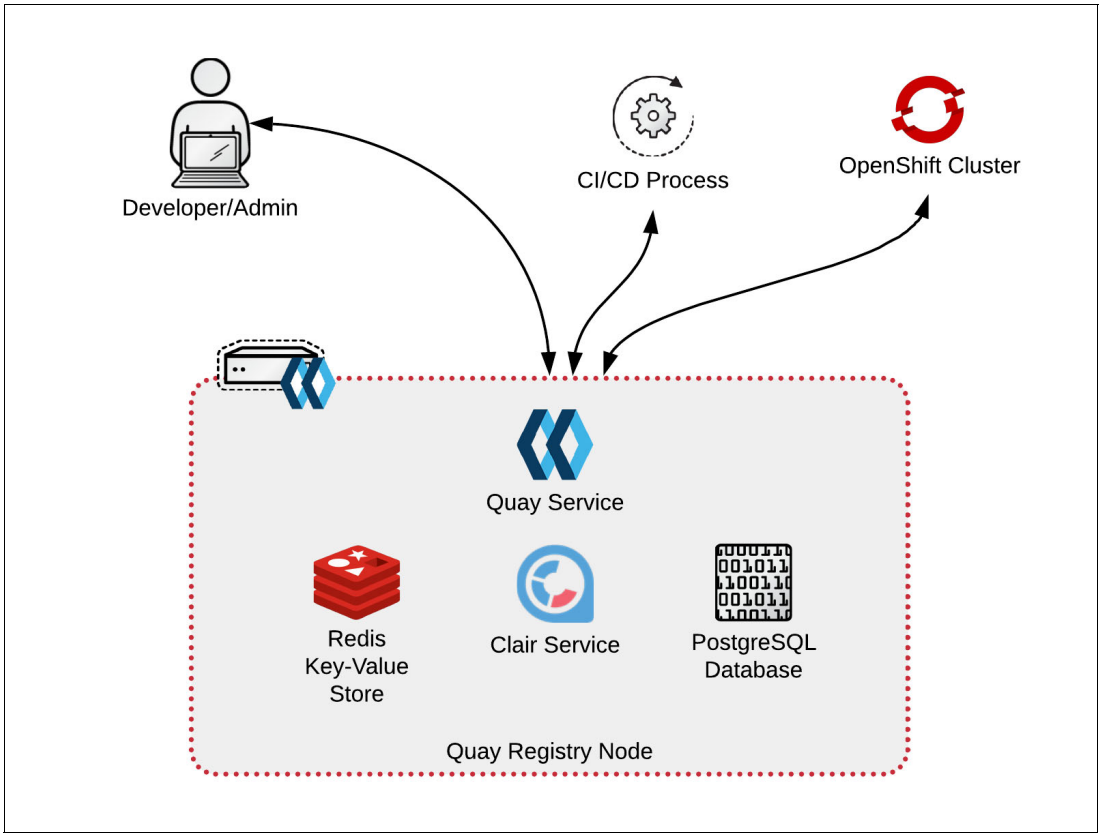


Figure 3-64 Red Hat Quay architecture

For supported deployments, you must use one of the following types of storage:

- Public cloud storage: In public cloud environments, you must use the cloud provider's Object Storage, such as Amazon S3 (for AWS) or Google Cloud Storage (for Google Cloud).

- Private cloud storage: In private clouds, a S3 or Swift compliant Object Store is needed, such as Ceph RADOS, or Red Hat OpenStack Swift.

3.4.3 Prerequisites

For this deployment of Red Hat Quay Registry installation, we recommended the use of an all in one machine (VM). Because all of the components of Red Hat Quay (including Registry, Scanner, DB, and Cache) are on the same machine, we require a high configuration machine. The following prerequisites must be met:

- Red Hat Enterprise Linux V7 or higher VM instance in IBM Cloud.
- Valid Red Hat Subscription: Obtain a valid Red Hat Enterprise Linux server subscription and a Red Hat Quay Subscription.
- CPUs: 8 vCPUs
- RAM: 24 GB or more
- Disk space: Total 50 GB of Storage with breakup as 20 GB OS, 20 GB Docker storage and 10 GB Quay local storage.

3.4.4 Installing Red Hat Quay

In this section, we describe installing Red Hat Quay private registry. For testing and demonstration purposes, we deploy Quay in a non-HA environment. If you want to deploy an HA environment or on Red Hat OpenShift as a project, see the documentation that is available at this web page.

The installation process includes the following overall steps:

1. Provision the Red Hat Enterprise Linux server on IBM Cloud.

   For more information about provisioning an IBM Cloud Virtual Servers, see the documentation that is available at this web page.

   For more information about provisioning bare metal machines, see 3.1, “Setting a bare metal server on IBM Cloud for Red Hat OpenStack” on page 32.

2. Activate the subscription of Red Hat Enterprise Linux Server.

   After the server is up and running, register it with a valid Red Hat subscription with RHSM. For more information, see Appendix A, “Red Hat subscription activation process” on page 401.

   Note: By default, when you provision IBM Cloud Virtual Instance with Red Hat Enterprise Linux, it includes a valid attached subscription. You can then skip this process.

   Registering the machine with Red Hat Network (RHN) and enabling the required repositories is shown in Example 3-29.

Example 3-29 Registering your machine

```
# subscription-manager register

# subscription-manager refresh

# subscription-manager list -available
```

```
# subscription-manager attach --pool=<pool_id>

# subscription-manager repos --disable="*"

# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms"

# yum update -y
```

3. Install Docker dependencies:

   a. Download a Docker engine dependency and install it on the machine. After the dependencies are downloaded, enable and start the Docker service:

   ```
   # yum install docker
   # systemctl enable docker
   # systemctl start docker
   # systemctl is-active docker
   Active
   ```

   b. Integrate your installed Docker engine with the public `quay.io` registry to download the required images of Quay for deployment. Log in to your Docker machine with the public registry `quay.io` by running the following command:

   ```
   # docker login -u="redhat+quay"
   -p="O81WSHRSJR14UAZBK54GQHJSOP1V4CLWAJV1X2C4SD7KO59CQ9N3RE12612XU1HR"
   quay.io
   ```

   c. Create a redis cache storage directory and change the directory ownership:

   # mkdir -p /var/lib/redis

   # chmod 777 /var/lib/redis

   d. Run a redis cache Docker container so that it can integrate with this created cache storage:

   ```
   # docker run -d --restart=always -p 6379:6379 \
       --privileged=true \
       -v /var/lib/redis:/var/lib/redis/data:Z \
       registry.access.redhat.com/rhscl/redis-32-rhel7
   ```

   e. You must ensure connectivity to UDP port 6379. Check the Telnet to this port after installing the Telnet utility on the server, as shown in Example 3-30.

   Example 3-30 Install telnet

   ```
   # yum install telnet -y
   [root@quay ~]# telnet 169.38.131.72 6379
   Trying 169.38.131.72...
   Connected to 169.38.131.72.
   Escape character is '^]'.
   MONITOR
   +OK
   QUIT
   +OK
   Connection closed by foreign host.
   [root@quay ~]#
   ```

3.4.5 Installing PostgreSQL database

In this section, we describe installing the PostgreSQL database for Clair scanner.

Create a DB container by running the following command:

```
[root@quay ~]# docker run -d --name clair-postgres -p 5432:5432 -e
POSTGRES_PASSWORD=mysecretpassword postgres
```

Example 3-31 lists all the running containers on the Docker engine to check the ports that are associated with each container.

Example 3-31 List running containers and check their ports

```
[root@quay ~]# docker ps
CONTAINER ID        IMAGE                             COMMAND                  CREATED
STATUS              PORTS                                                    NAMES
aec7e61df769        quay.io/redhat/quay:v3.2.2        "/quay-registry/qu..."   3 days ago
Up 3 days           7443/tcp, 0.0.0.0:80->8080/tcp, 0.0.0.0:443->8443/tcp    inspiring_minsky
951110e2273e        quay.io/redhat/clair-jwt:v3.2.1   "/clair/clair-entr..."   2 weeks ago
Up 3 days           0.0.0.0:6060-6061->6060-6061/tcp                         elastic_heyrovsky
1086dfc11bdd        0c11f8cceeab                      "docker-entrypoint..."   30 seconds ago
Up 30 seconds           0.0.0.0:5432->5432/tcp                                  clair-postgres
b29a2c507dfd        a00c397427d3                      "container-entrypo..."   2 weeks ago
Up 2 weeks          0.0.0.0:6379->6379/tcp                                   kickass_lalande
```

3.4.6 Creating a Quay database

Complete the following steps:

1. Create the databases for Quay on the deployed DB container, as shown in Example 3-32. After the databases are created, exit the DB shell.

Example 3-32 Create databases for Quay

```
[root@quay ~]# docker exec -it 1086dfc11bdd bash
root@1086dfc11bdd:/# psql -U postgres
psql (12.2 (Debian 12.2-2.pgdg100+1))
Type "help" for help.

postgres=# create database quay;
CREATE DATABASE
postgres-# \l
                                 List of databases
   Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
-----------+----------+----------+------------+------------+-----------------------
-
 postgres  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 quay      | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres
+
           |          |          |            |            | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres
+
           |          |          |            |            | postgres=CTc/postgres
(5 rows)
```

```
postgres=# \q
root@1086dfc11bdd:/# exit
exit
[root@quay ~]#
```

2. Run the Quay Docker container on port 8443 and set the configuration password:

```
# docker run --privileged=true -p 8443:8443 -d quay.io/redhat/quay:v3.2.2
config my-secret-password
```

3.4.7 Integrating Quay with Clair Vulnerability Scanner

Complete the following steps to integrate Quay with Clair Vulnerability Scanner:

1. Verify PostgreSQL is running, as shown in Example 3-33.

Example 3-33 Check PostgreSQL

```
[root@quay ~]# docker ps
CONTAINER ID        IMAGE                               COMMAND
CREATED           STATUS            PORTS
NAMES
aec7e61df769        quay.io/redhat/quay:v3.2.2        "/quay-registry/qu..."   3
days ago          Up 3 days           7443/tcp, 0.0.0.0:80->8080/tcp,
0.0.0.0:443->8443/tcp   inspiring_minsky
951110e2273e        quay.io/redhat/clair-jwt:v3.2.1   "/clair/clair-entr..."   2
weeks ago       Up 3 days         0.0.0.0:6060-6061->6060-6061/tcp
elastic_heyrovsky
1086dfc11bdd        0c11f8cceeab                      "docker-entrypoint..."   30
seconds ago       Up 30 seconds       0.0.0.0:5432->5432/tcp
clair-postgres
b29a2c507dfd        a00c397427d3                      "container-entrypo..."   2
weeks ago       Up 2 weeks        0.0.0.0:6379->6379/tcp
kickass_lalande
```

2. Create Clair database in PostgreSQL, as shown in Example 3-34.

Example 3-34 Create Clair database

```
[root@quay ~]# docker exec -it 1086dfc11bdd bash
root@1086dfc11bdd:/# psql -U postgres
psql (12.2 (Debian 12.2-2.pgdg100+1))
Type "help" for help.

postgres=# create database clair;
CREATE DATABASE
postgres-# \l
                                 List of databases
   Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
-----------+----------+----------+------------+------------+-----------------------
-
 clair     | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 quay      | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres
+
```

```
           |          |          |            |            | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres
+
           |          |          |            |            | postgres=CTc/postgres
(5 rows)


postgres=# \q
root@1086dfc11bdd:/# exit
exit
[root@quay ~]#
```

3. Pull Clair image:

   ```
   # docker pull quay.io/redhat/clair-jwt:v3.2.1
   ```

4. Create the Clair configuration directory:

   ```
   mkdir clair-config
   ```

5. Copy the `security-scanner.pem` certificate in the Quay set up to the `clair-config` directory.

6. Create the Clair configuration file in `clair-config` directory, as shown in Example 3-35.

Example 3-35 Sample Clair configuration

```
clair:
  database:
    type: pgsql
    options:
      # A PostgreSQL Connection string pointing to the Clair Postgres database.
      # Documentation on the format can be found at:
http://www.postgresql.org/docs/9.4/static/libpq-connect.html
      source:
postgresql://postgres:mysecretpassword@quay.ocp4.redbooks.info:5432/clair?sslmode=
disable
      cachesize: 16384
  api:
    # The port at which Clair will report its health status. For example, if Clair
is running at
    # https://clair.mycompany.com, the health will be reported at
    # http://clair.mycompany.com:6061/health.
    healthport: 6061

    port: 6062
    timeout: 900s

    # paginationkey can be any random set of characters. *Must be the same across
all Clair instances*.
    paginationkey: "XxoPtCUzrUv4JV5dS+yQ+MdW7yLEJnRMwigVY/bpgtQ="

  updater:
    # interval defines how often Clair will check for updates from its upstream
vulnerability databases.
    interval: 6h
    notifier:
      attempts: 3
      renotifyinterval: 1h
```

```
      http:
        # QUAY_ENDPOINT defines the endpoint at which Quay is running.
        # For example: https://myregistry.mycompany.com
        endpoint: http://quay.ocp4.redbooks.info/secscan/notify
        proxy: http://localhost:6063

jwtproxy:
  signer_proxy:
    enabled: true
    listen_addr: :6063
    ca_key_file: /certificates/mitm.key # Generated internally, do not change.
    ca_crt_file: /certificates/mitm.crt # Generated internally, do not change.
    signer:
      issuer: security_scanner
      expiration_time: 5m
      max_skew: 1m
      nonce_length: 32
      private_key:
        type: preshared
        options:
          # The ID of the service key generated for Clair. The ID is returned when
setting up
          # the key in [Quay Setup](security-scanning.md)
          key_id: 6f9d76928ad38e9e1ddbd77795adfd6bf8bd52a03092e700c8369824a2c118f6
          private_key_path: /clair/config/security_scanner.pem

  verifier_proxies:
  - enabled: true
    # The port at which Clair will listen.
    listen_addr: :6060

    # If Clair is to be served via TLS, uncomment these lines. See the "Running
Clair under TLS"
    # section below for more information.
    # key_file: /config/clair.key
    # crt_file: /config/clair.crt

    verifier:
      # CLAIR_ENDPOINT is the endpoint at which this Clair will be accessible.
Note that the port
      # specified here must match the listen_addr port a few lines above this.
      # Example: https://myclair.mycompany.com:6060
      audience: http://quay.ocp4.redbooks.info:6060

      upstream: http://localhost:6062
      key_server:
        type: keyregistry
        options:
          # QUAY_ENDPOINT defines the endpoint at which Quay is running.
          # Example: https://myregistry.mycompany.com
          registry: http://quay.ocp4.redbooks.info/keys/
```

7. Run Clair:

```
# docker run --restart=always -p 6060:6060 -p 6061:6061 \
    -v /clair/config:/clair/config \
    quay.io/redhat/clair-jwt:v3.2.1
```

8. Check the output, as shown in Figure 3-65.

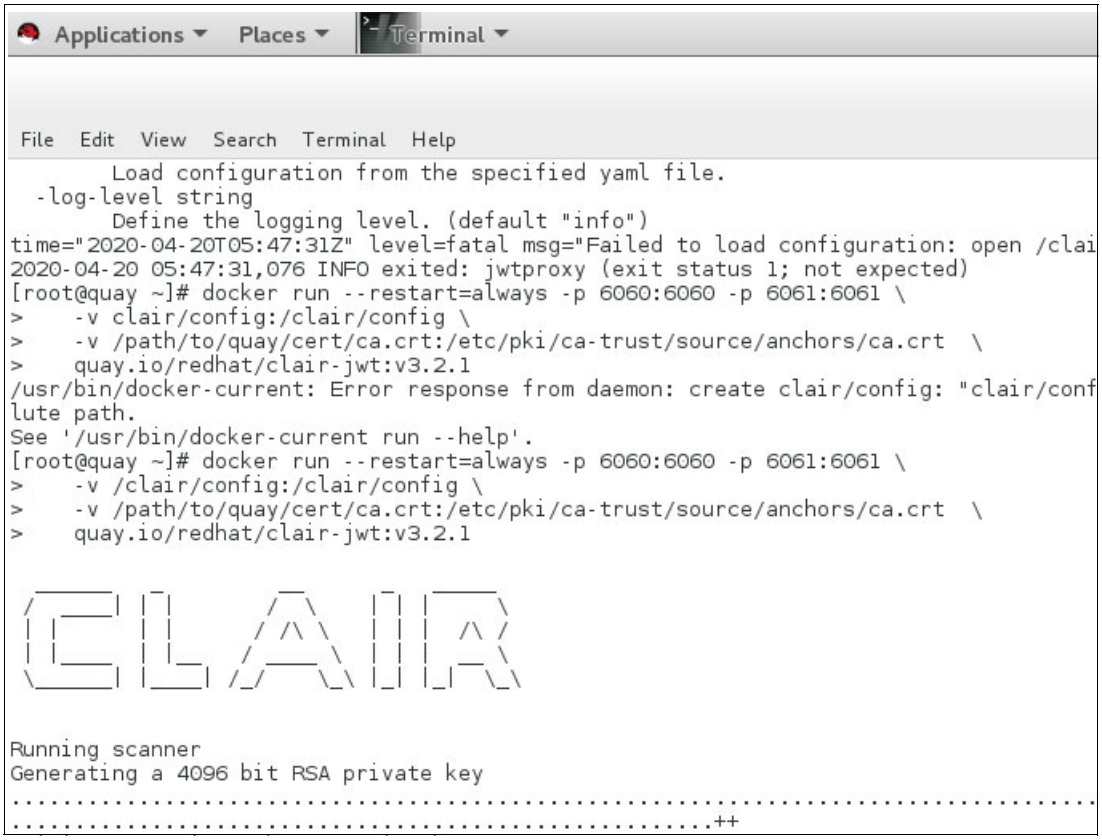


Figure 3-65 Clair output

3.4.8 Configuring Red Hat Quay

Complete the following steps to configure Red Hat Quay:

1. Open a browser and then, open quay IP or DNS.
2. After authentication prompt appears, enter the user name as `quayconfig` and password as entered in previous step (in this case, password is `my-secret-password`).

3. Click **Log In**, as shown in Figure 3-66.

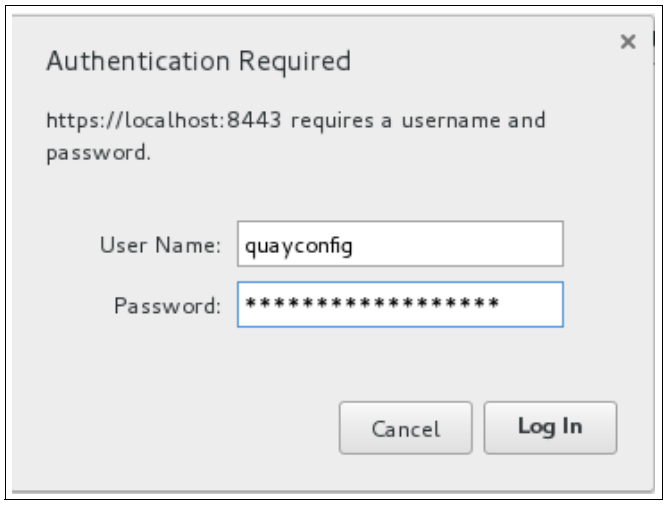


Figure 3-66 Quay authentication window

4. Select **Start new registry setup**, as shown in Figure 3-67. A New Registry set up process starts.

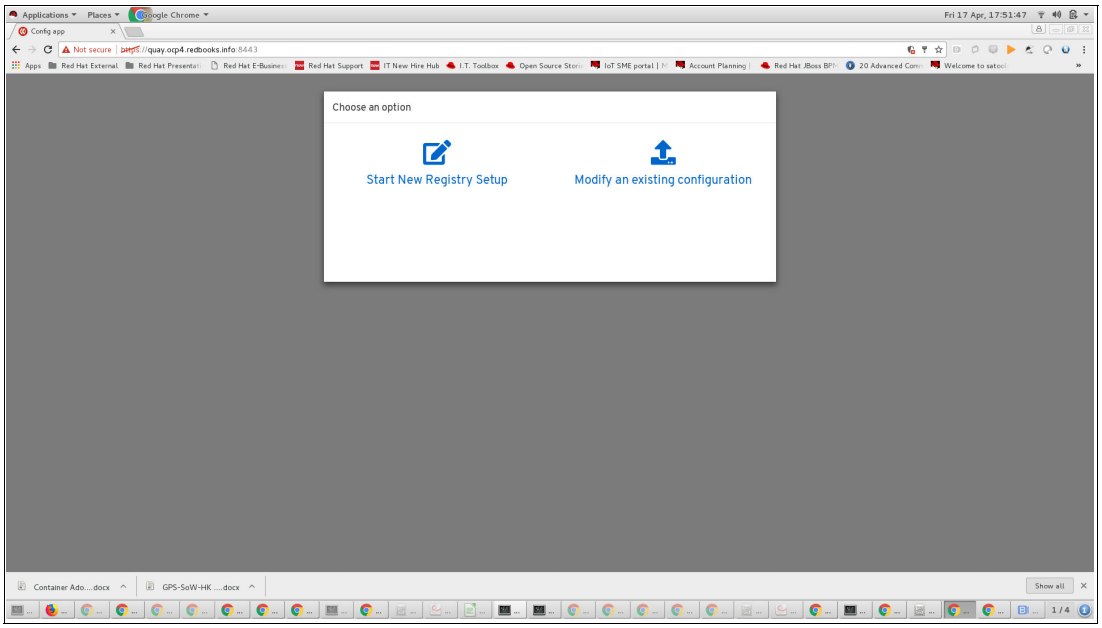


Figure 3-67 Starting a new registry configuration

5. Enter the database details (as entered in 3.4.7, “Integrating Quay with Clair Vulnerability Scanner” on page 123) and then click **Validate Database Settings**, as shown in Figure 3-68.

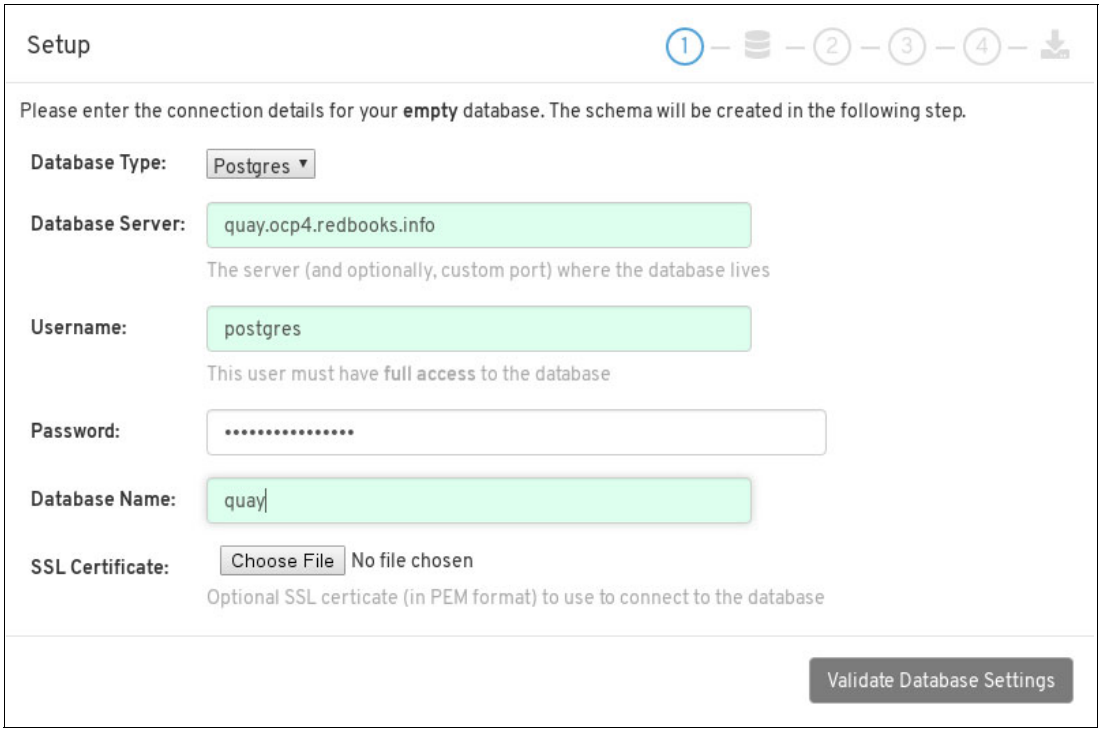


Figure 3-68 Setup database window

The set up process takes some time to create database schema. Wait for it to finish, as shown in Figure 3-69.

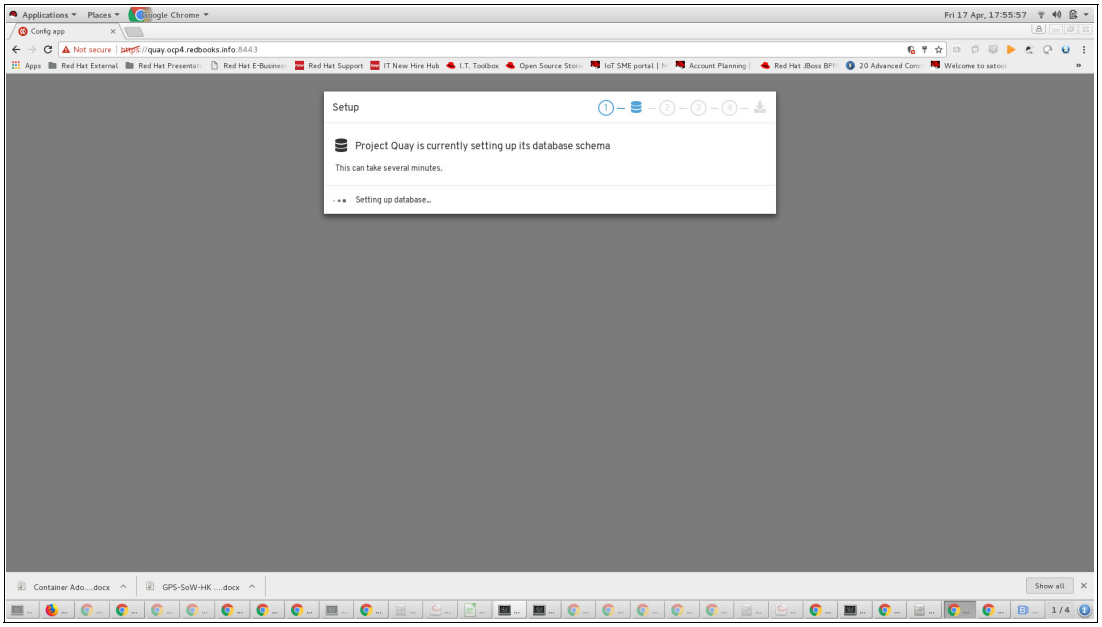


Figure 3-69 Database schema setup in progress

6. When prompted to enter user information, enter the superuser details and then, click **Create Super User**, as shown in Figure 3-70.

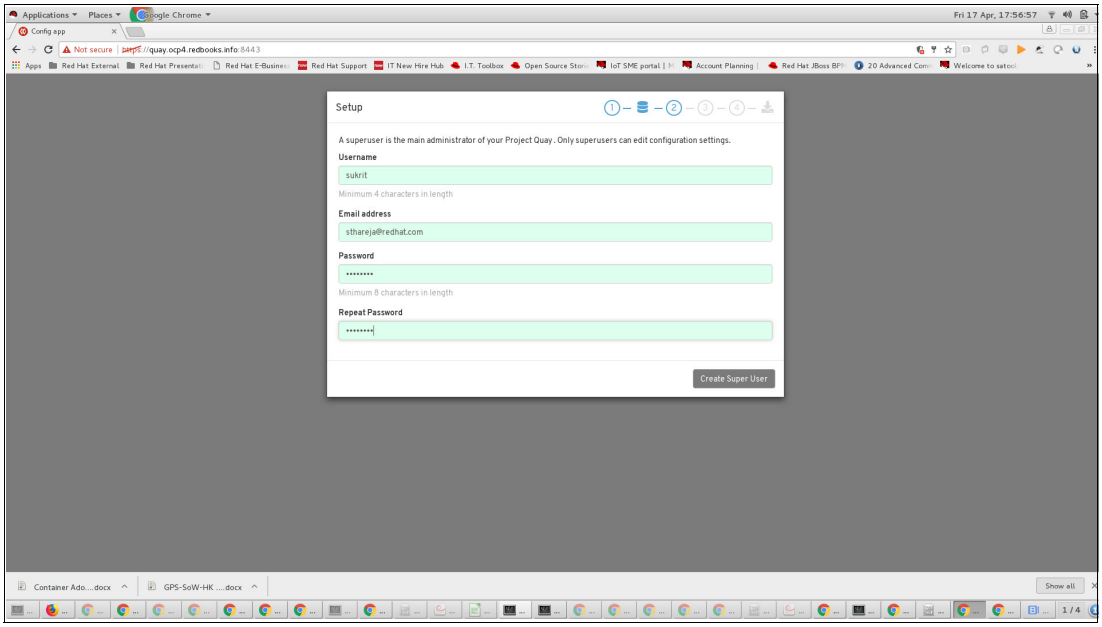


Figure 3-70 Setup continues: Enter user details

7. Configure the Redis cache by entering the Redis details, as shown in Figure 3-71.

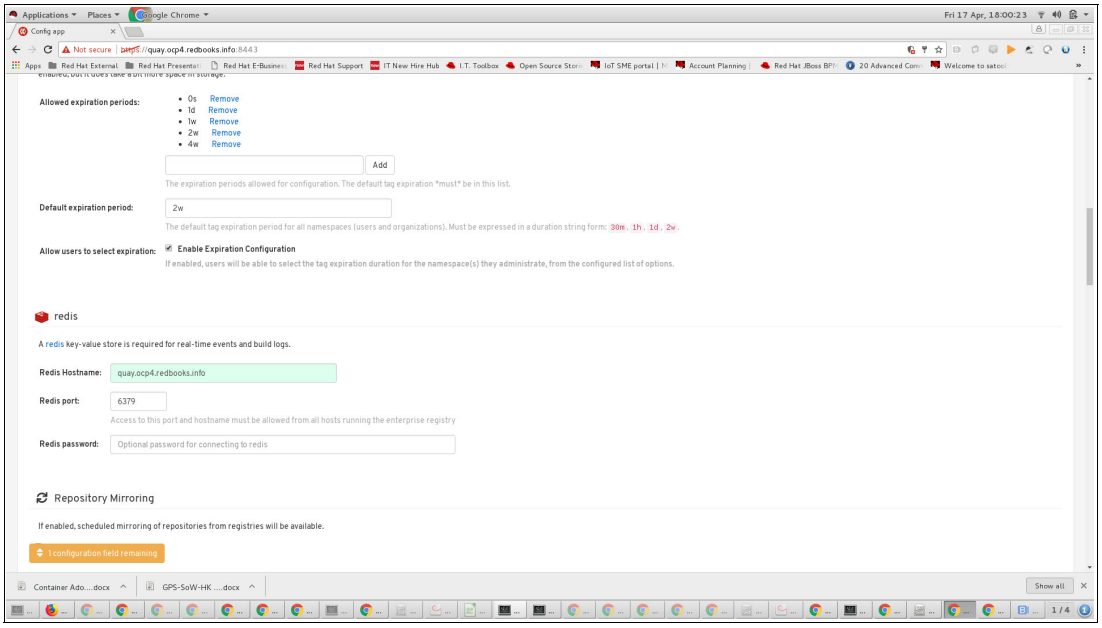


Figure 3-71 Enter Redis details

8. Enter the server hostname. Do *not* select Save Configuration Changes yet. Ignore The Checking your settings pop-up window, as shown in Figure 3-72.

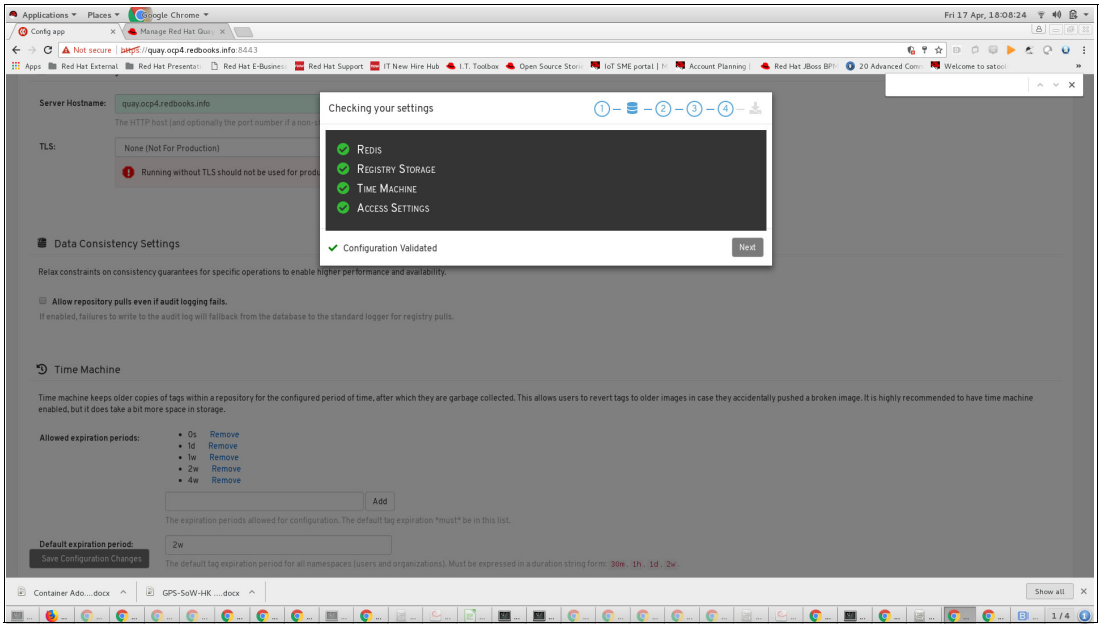


Figure 3-72 Checking the Redis settings window

9. Select **Enable Security Scanning** and click **Create Key** to open a pop-up window to generate the authentication key, as shown in Figure 3-73.

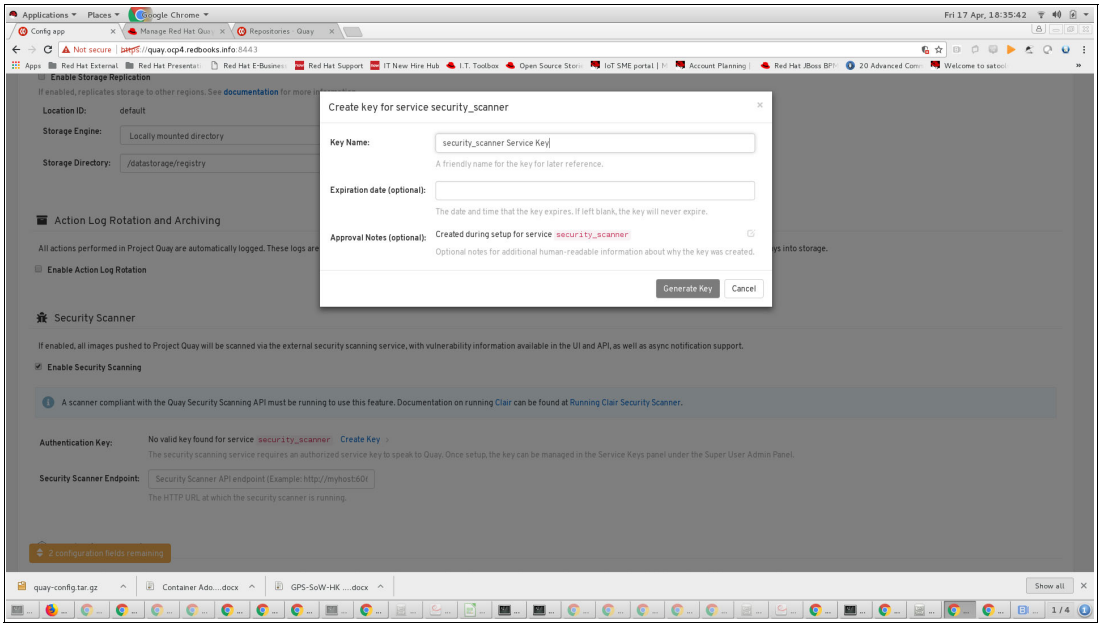


Figure 3-73 Create key for service security_scanner window

10. In the pop-up window (see Figure 3-74) click **Generate Key**.

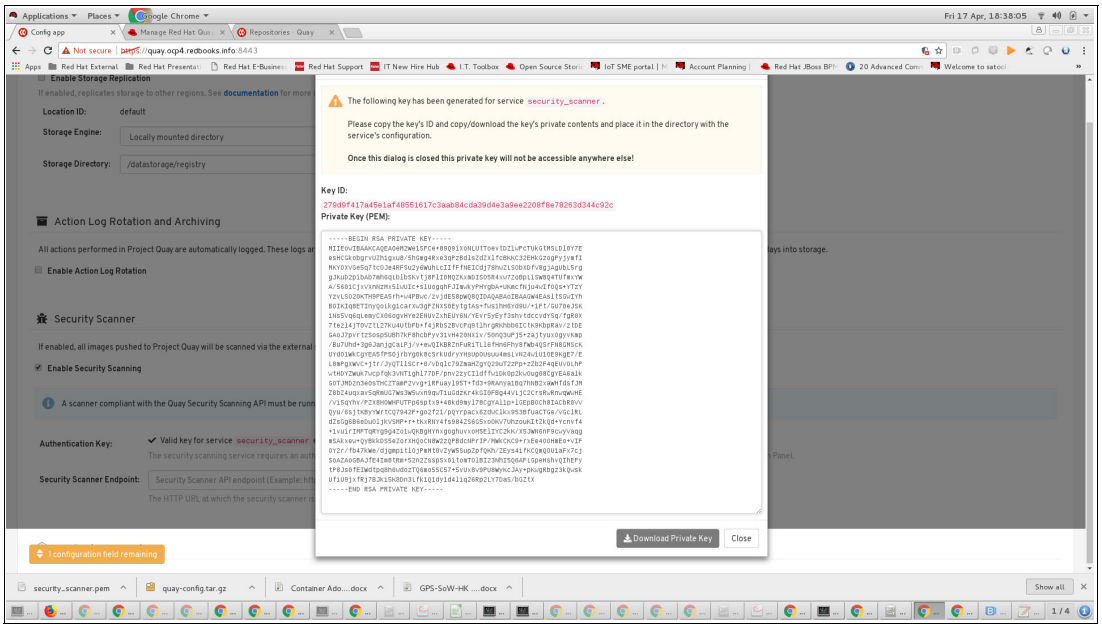


Figure 3-74 Generate the key

11. Make a note of the key ID and copy that key ID in your local file. Then, download the key to generate a `security-scanner.pem` file, as shown in Figure 3-75.

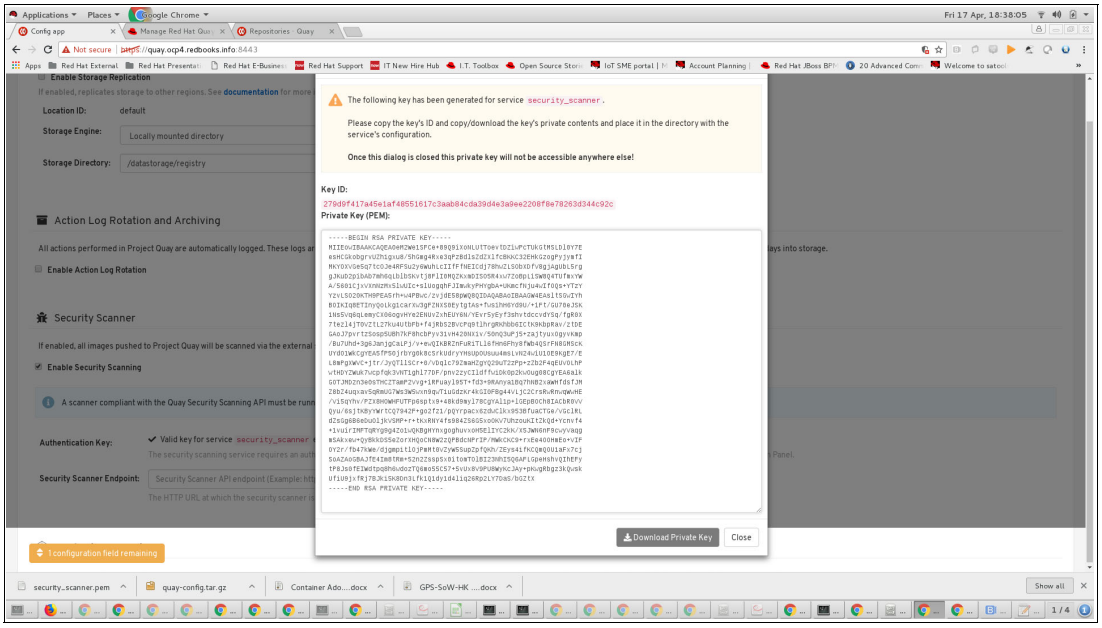


Figure 3-75 Generate a security-scanner.pem

12. Close the pop-up dialogue and configure the persistent storage.

13. Select **Enable Storage Replication**, as shown in Figure 3-76.

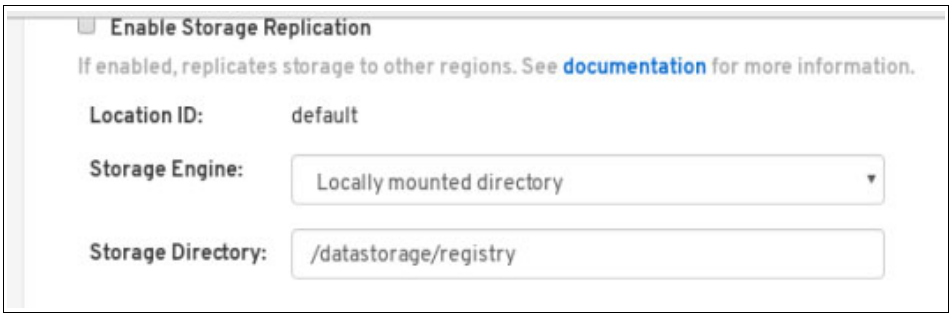


Figure 3-76 Storage Replication configuration window

14. Enter following details (see Figure 3-77):

- Storage Engine: `RadosGWStorage`
- Rados Server Hostname: `s3.che01.cloud-object-storage.appdomain.cloud`
- Access Key: `KSM5M4GtHrsGrM3bntdh`
- Secret Key: `T1MLlUDuRXOa4j3oYVF1ODpSagWn1j1XYDpABkVD`
- Bucket name for Object Storage: `quay-object-store.ocp4.redbooks.info`
- Storage Directory: /

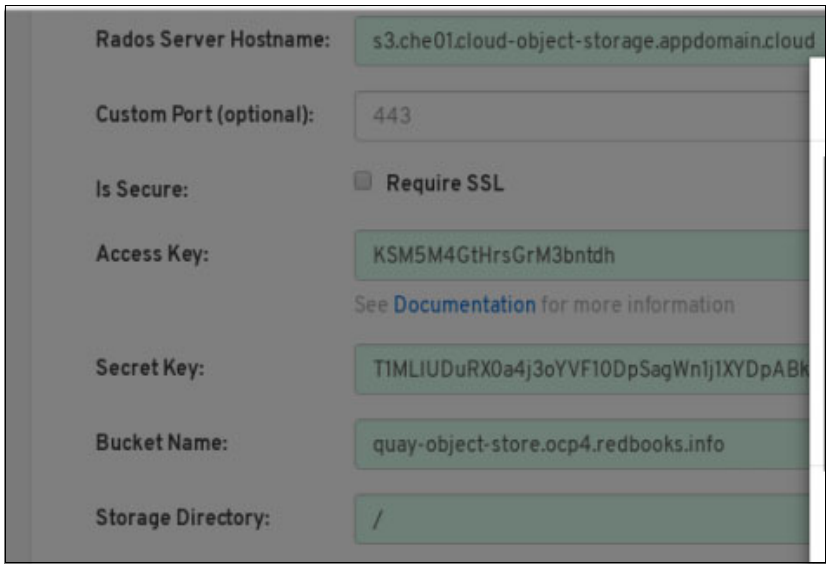


Figure 3-77 Enter more details

15. Click **Save Configuration Changes**, as shown in Figure 3-78.

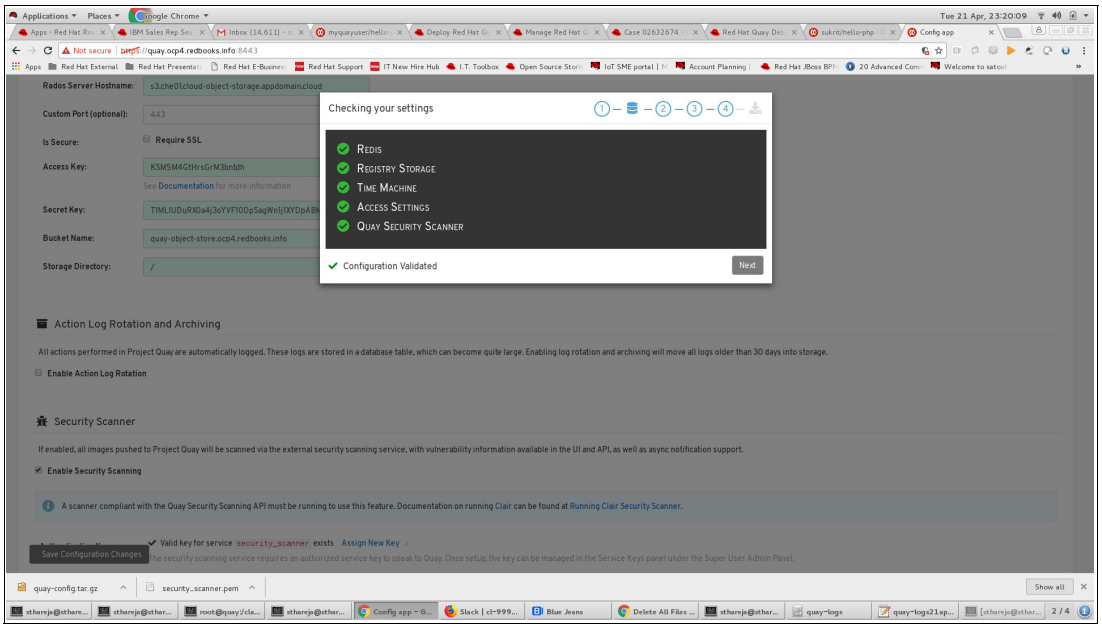


Figure 3-78 Save configuration

16. Download the configuration file `Quay-config.tar.gz`, as shown in Figure 3-79.

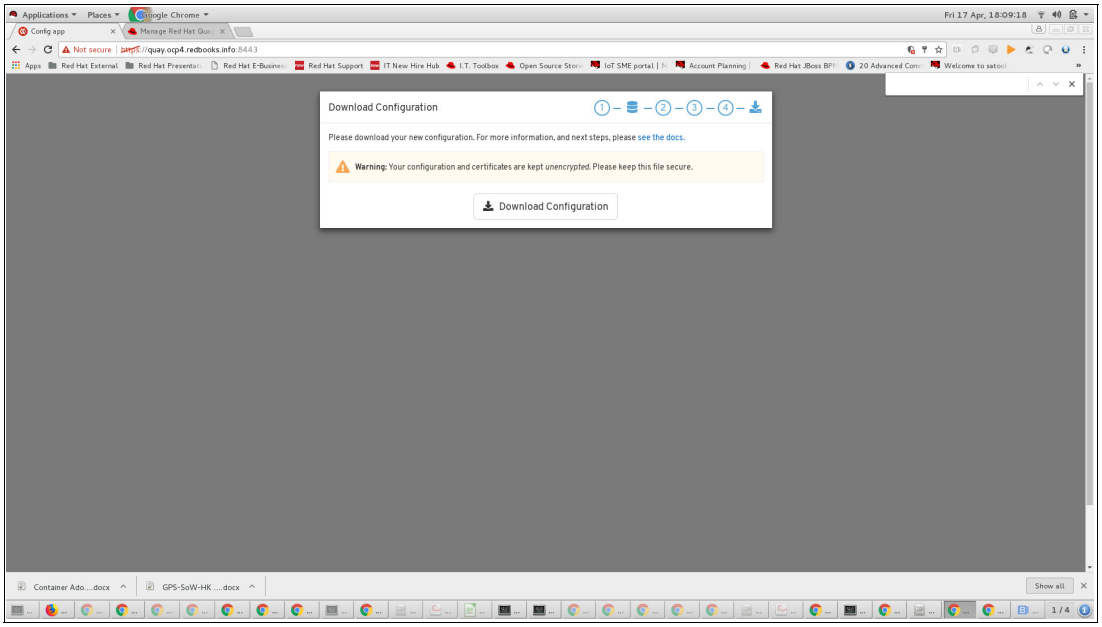


Figure 3-79 Download configuration file

17. Create a configuration mount point by creating a directory:

```
mkdir -p /mnt/quay/config
```

18. Copy the configuration file you downloaded:

```
cp quay-config.tar.gz /mnt/quay/config/
```

19. Extract the configuration file downloaded, as shown in Example 3-36.

Example 3-36 Extracting the configuration file

```
AUTHENTICATION_TYPE: Database
BITTORRENT_FILENAME_PEPPER: 88171130-fb08-4d0a-8b39-6423eba81e94
BUILDLOGS_REDIS:
  host: quay.ocp4.redbooks.info
  port: 6379
DATABASE_SECRET_KEY:
'700398737862150299589931803413791428577989975779958408043432685195811624588 6'
DB_URI: postgresql://postgres:mysecretpassword@169.38.131.72:5432/quay
DEFAULT_TAG_EXPIRATION: 2w
DISTRIBUTED_STORAGE_CONFIG:
  default:
  - RadosGWStorage
  - access_key: KSM5M4GtHrsGrM3bntdh
    bucket_name: quay-object-store.ocp4.redbooks.info
    hostname: s3.che01.cloud-object-storage.appdomain.cloud
    is_secure: false
    secret_key: T1MLlUDuRXOa4j3oYVF1ODpSagWn1j1XYDpABkVD
    storage_path: /
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
- default
ENTERPRISE_LOGO_URL: /static/img/quay-horizontal-color.svg
FEATURE_ACI_CONVERSION: false
FEATURE_ANONYMOUS_ACCESS: true
FEATURE_APP_REGISTRY: false
FEATURE_APP_SPECIFIC_TOKENS: true
FEATURE_BUILD_SUPPORT: false
FEATURE_CHANGE_TAG_EXPIRATION: true
FEATURE_DIRECT_LOGIN: true
FEATURE_MAILING: false
FEATURE_PARTIAL_USER_AUTOCOMPLETE: true
FEATURE_REPO_MIRROR: false
FEATURE_REQUIRE_TEAM_INVITE: true
FEATURE_RESTRICTED_V1_PUSH: true
FEATURE_SECURITY_NOTIFICATIONS: true
FEATURE_SECURITY_SCANNER: true
FEATURE_USERNAME_CONFIRMATION: true
FEATURE_USER_CREATION: true
FEATURE_USER_LOG_ACCESS: true
GITHUB_LOGIN_CONFIG: {}
GITHUB_TRIGGER_CONFIG: {}
GITLAB_TRIGGER_KIND: {}
GPG2_PRIVATE_KEY_FILENAME: signing-private.gpg
GPG2_PUBLIC_KEY_FILENAME: signing-public.gpg
LOG_ARCHIVE_LOCATION: default
MAIL_DEFAULT_SENDER: support@quay.io
MAIL_PORT: 587
MAIL_USE_TLS: true
PREFERRED_URL_SCHEME: http
REGISTRY_TITLE: Red Hat Quay
REGISTRY_TITLE_SHORT: Red Hat Quay
REPO_MIRROR_SERVER_HOSTNAME: null
```

```
REPO_MIRROR_TLS_VERIFY: true
SECRET_KEY:
'76206637356380924655390075248573242842105151823736990161050111190878470986822'
SECURITY_SCANNER_ENDPOINT: http://quay.ocp4.redbooks.info:6060
SECURITY_SCANNER_ISSUER_NAME: security_scanner
SERVER_HOSTNAME: quay.ocp4.redbooks.info
SETUP_COMPLETE: true
SIGNING_ENGINE: gpg2
SUPER_USERS:
- superuser
TAG_EXPIRATION_OPTIONS:
- 0s
- 1d
- 1w
- 2w
- 4w
TEAM_RESYNC_STALE_TIME: 60m
TESTING: false
USERFILES_LOCATION: default
USERFILES_PATH: userfiles/
USER_EVENTS_REDIS:
  host: quay.ocp4.redbooks.info
  port: 6379
USE_CDN: false
V3_UPGRADE_MODE: complete
```

20. The Quay is deployed successfully. Rerun the quay container with the activated configuration:

```
docker run --restart=always -p 443:8443 -p 80:8080 \
   --sysctl net.core.somaxconn=4096 \
   --privileged=true \
   -v /mnt/quay/config:/conf/stack:Z \
   -v /mnt/quay/storage:/datastorage:Z \
   -d quay.io/redhat/quay:v3.2.2
```

21. Now that the Quay is configured successfully, login to Quay by using the superuser that you created.

3.4.9 Administering Red Hat Quay

Complete the following steps:

1. Browse to the Quay URL and log in to Quay by using the superuser you created, as shown in Figure 3-80.

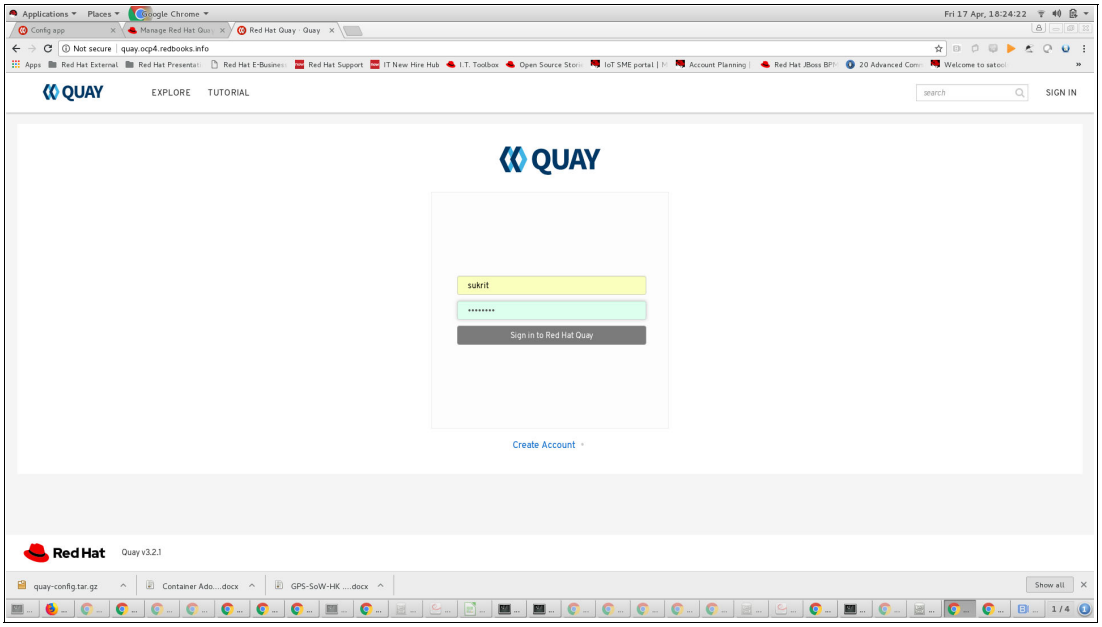


Figure 3-80 Log in to Quay

2. After you log in successfully, you see all of the created repositories, as shown in Figure 3-81. You can create a repository by clicking **Create New Repository**. An organization and repository are created for you.

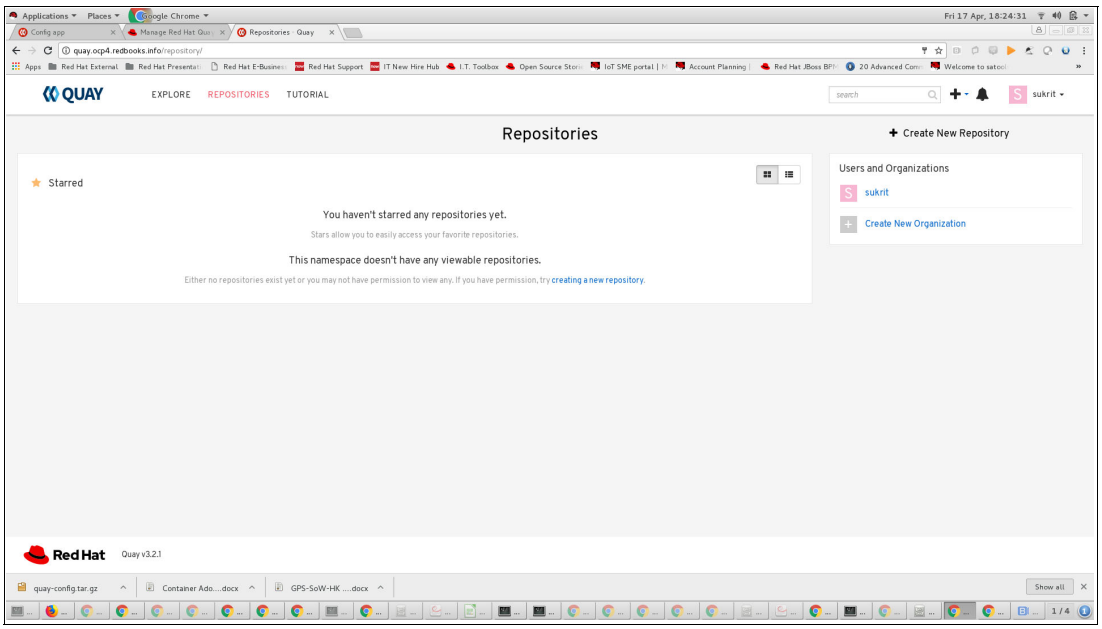


Figure 3-81 Quay Repositories window

3. After the repository is created successfully, log in to your Quay registry from the container platform and push the container image to your repository for vulnerability scanning, as shown in Figure 3-82.

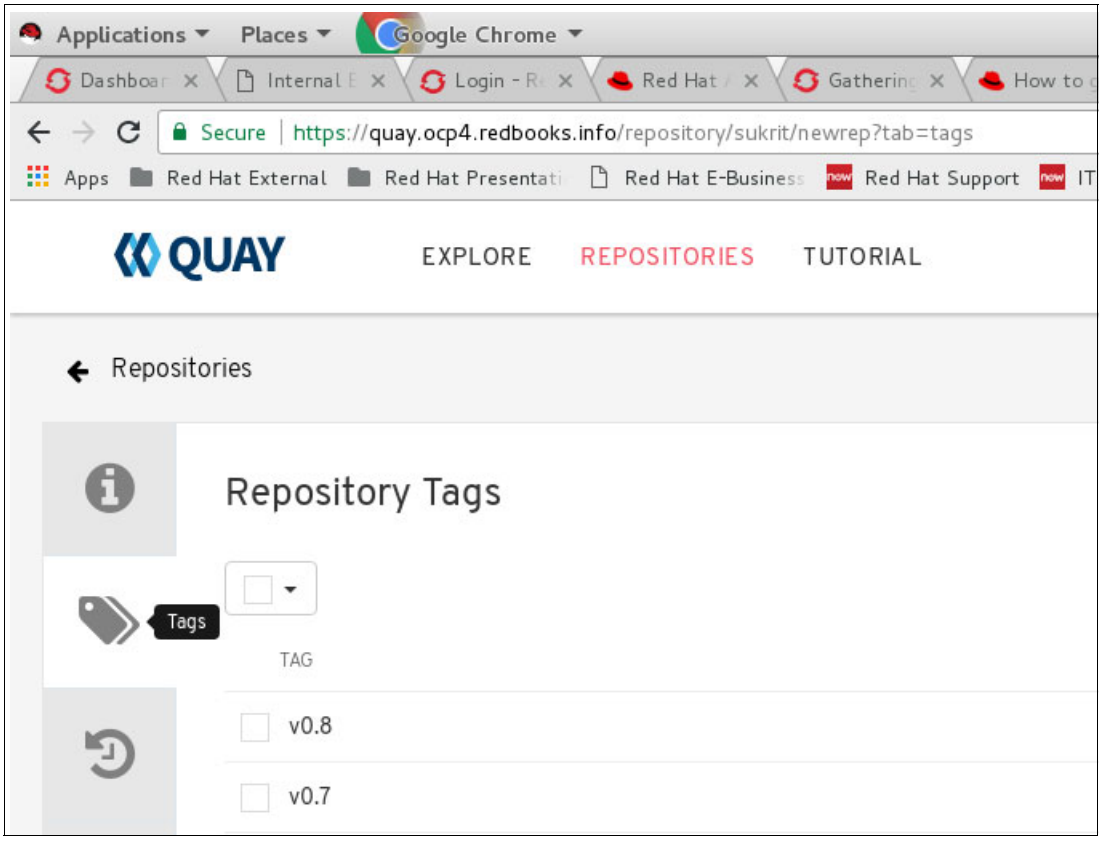


Figure 3-82 Quay Repositories: Repository Tags window

4. When the container image is pushed to your registry, Clair Scanner automatically scans the image for vulnerabilities. The associated CVE ID shows its severity and procedure to fix the vulnerability, as shown in Figure 3-83.

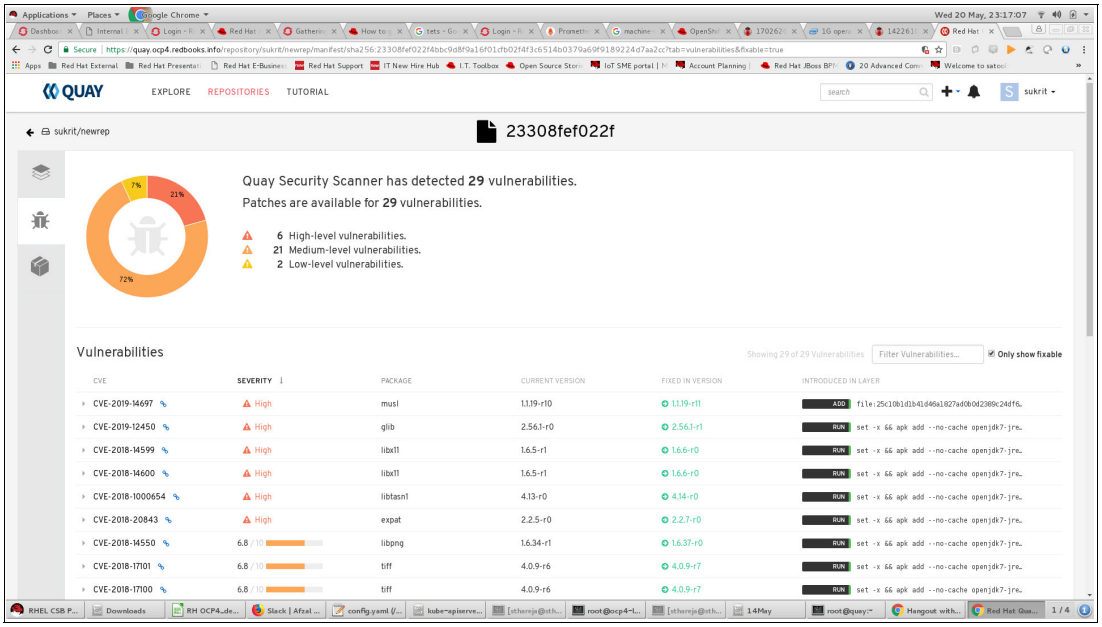


Figure 3-83 Quay Repositories: Security Scanner

3.5 Red Hat Ansible Automation Platform

This section describes Red Hat Ansible Automation Platform. This section also shows how Red Hat Ansible Tower is deployed as an independent deployer for a multi-hybrid cloud scenario.

This section also describes installing Red Hat Ansible Tower, its configuration, and provides a demonstration automation with the help of the deployed Red Hat Ansible Tower. Red Hat Ansible is widely used automation language, and is considered as a language of DevOps.

3.5.1 Red Hat Ansible Tower overview

Red Hat Ansible is a simple, open source automation language with which an organization can describe and automate IT applications and infrastructure in easy to read and write Playbooks. Red Hat Ansible is minimal in nature. It has no agent or similar software installation prerequisites; instead, it uses commonly built-in remote communications software, such as SSH for Linux/UNIX hosts, WinRM for Windows hosts, and specialized network connection plug-ins for network devices to run tasks on local and remote systems.

Red Hat Ansible Playbooks are written in YAML format and have minimal syntax. Unlike similar automation applications, it is not a programming language or script. It is more akin to a model of a configuration or process that you want to automate. Red Hat Ansible Tower complements Red Hat Ansible by providing a web-based UI with visual automation, management, and monitoring capabilities. It provides a dashboard to help manage deployments, create workflows, and monitor resources.

In addition, Tower adds Enterprise features, such as role-based access control (RBAC), Lightweight Directory Access Protocol (LDAP), and Active Directory integration, Push-Button Deployments, Job Scheduling, Centralized Logging, Graphical Inventory Management, and a RESTful API for further integration into the wider enterprise workflow.

3.5.2 Red Hat Ansible Automation architecture

Red Hat Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.

Designed for multitier deployments, Red Hat Ansible models your IT infrastructure by describing how all of your systems inter-relate, rather than just managing one system at a time.

Because it uses no agents and no other custom security infrastructure, it is easy to deploy and (most importantly), it uses a simple language (YAML, in the form of Red Hat Ansible Playbooks) with which you can describe your automation jobs in a way that approaches plain English.

Red Hat Ansible features the following components:

- Modules

  Red Hat Ansible works by connecting to your nodes and pushing out scripts that are called *Ansible modules* to them. Most modules accept parameters that describe the wanted state of the system. Red Hat Ansible then runs these modules (over SSH by default), and removes them when finished. Your library of modules can be on any machine, and servers, daemons, or databases are *not* required.

- Plug-ins

  Plug-ins augment Red Hat Ansible's core functions. Although modules run on the target system in separate processes (usually that means on a remote system), plug-ins run on the control node within the `/usr/bin/ansible` process. Plug-ins offer options and extensions for the core features of Red Hat Ansible: transforming data, logging output, connecting to inventory, and so on.

- Inventory

  By default, Red Hat Ansible represents the machines it manages in a file (INI, YAML, and others) that groups all of your managed machines per your own choosing.

  To add new machines, no other SSL signing server is involved; therefore, there is never any issue deciding why a specific machine was not linked because of obscure NTP or DNS issues.

  A plain text inventory file resembles the following example:

```
[webservers]
www1.example.com
www2.example.com

[dbservers]
db0.example.com
db1.example.com
```

- Playbooks

  Playbooks can finely orchestrate multiple slices of your infrastructure topology, with detailed control over how many machines to tackle at a time, as shown in Example 3-37.

Example 3-37 Playbook sample

```
---
- hosts: webservers
serial: 5 # update 5 machines at a time
roles:
- common
- webapp

- hosts: content_servers
roles:
- common
- content
```

- Managed devices

  End devices that Red Hat Ansible manages and performs automation can be any machine, including network devices, servers, and any machine that supports SSH connectivity.

- CMDB

  A centralized database stores all of the metadata of the Red Hat Ansible system, as shown in Figure 3-84.

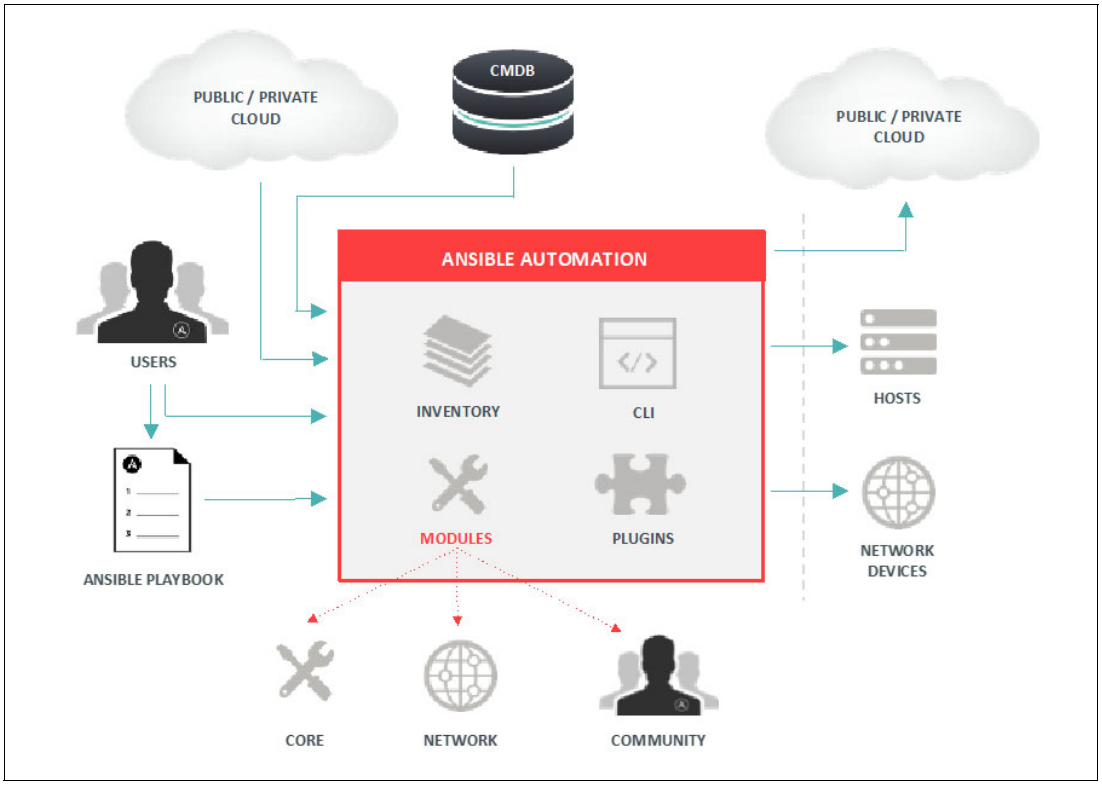


Figure 3-84 Red Hat Ansible centralized DB

3.5.3 Red Hat Ansible Tower prerequisites

Because we intend to perform a stand-alone tower deployment, we created a VM instance on the same bare metal node on IBM Cloud where Red Hat CloudForms is deployed. The following requirements for the VM must be met:

- Software:
  - Red Hat Enterprise Linux 7.4 or later 64-bit, recommended is the latest version.
  - PostgreSQL version 9.6 (at minimum) to run Red Hat Ansible Tower 3.2 and later.
  - Red Hat Ansible version 2.2 (at minimum) to run Ansible Tower versions 3.2 and later.
- Hardware (see Table 3-8):
  - DNS and NTP
  - SSH connectivity to managed devices
  - Internet and Proxy connection for downloading dependencies
  - 64-bit support (kernel and runtime)

Table 3-8 Red Hat Ansible prerequisites

VM count	1
CPU cores	8 vCPU
RAM memory	16 GB
Network interfaces	1G X1
Storage HDD	► 50 GB for / ► 20 GB for /var

3.5.4 Installation of Red Hat Ansible Tower

This deployment installs Red Hat Ansible Tower as single machine as an integrated installation. This machine is a single machine installation of Tower. The web front end, REST API backend, and database are all on a single machine.

This installation is the standard installation of Tower. It also installs PostgreSQL from your operating system vendor repository and configures the Tower service to use that as its database.

The latest version of Red Hat Ansible Tower is available at this web page.

A list of package dependencies from Red Hat Enterprise Linux repositories can be found in the `bundle/base_packages.txt` file that is inside the setup bundle. These dependencies must be fulfilled in a Staples Solution's environment. A third-party vendor might require more modules to be installed. These dependencies can be fulfilled by running the **`pip`** command.

The Tower installation includes the following settings:

- The `pg_sslmode` controls the SSL functions of the PostgreSQL client; for example, how the Tower server connects to the database. It defaults to `prefer`, which means if the database server offers SSL, the client uses it. You can also set it to `verify-full` to enforce SSL with full verification of certificate trust.
- The `web_server_ssl_cert` and `web_server_ssl_key` allows the user to provide a certificate and key to be installed in the web server for the Tower UI and API. These certificates must be provided or both must be absent. If they are absent, a self-signed (untrusted) certificate is generated at installation time.

- The `postgres_use_ssl` (true/false) controls whether the PostgreSQL server is configured to require SSL. This configuration affects only an internal or embedded database (for example, when the Tower install script is deploying the database server). It has no effect on an external database.
- The `postgres_ssl_cert` and `postgres_ssl_key` must be supplied when `postgres_use_ssl` is true. These certificates must have a CN (or wildcard, subject alternative name, and so on) that matches the hostname that the Tower nodes use to connect to the database server.

Server provisioning

This section describes how to create Red Hat Ansible Tower VM and install Red Hat Enterprise Linux operating system:

1. Provision a VM with the specified configuration that is described in 3.5.3, “Red Hat Ansible Tower prerequisites” on page 141.
2. Attach the downloaded Red Hat Enterprise Linux V7 or higher ISO image to the virtual CD/DVD-ROM of the VM.
3. Set the boot order to CD/DVD-ROM as priority.
4. After the ISO image is attached, boot the VM by using the attached Red Hat Enterprise Linux V7 or higher.
5. In the console, press Enter to continue installing the operating system.
6. Set the user name and password for the root and user.
7. After the operating system installation is complete and restart to machine is successful, SSH to the machine by using the root credentials.
8. Activate the subscription on the installed machine.

Activating subscriptions and repositories

Follow the procedure that is described in Appendix A, “Red Hat subscription activation process” on page 401 to subscribe the Red Hat Enterprise Linux host to RHN and enable required software repositories. However, the following high-level steps that must be performed are presented for reference:

```
# subscription-manager register
# subscription-manager list --available
# subscription-manager attach --pool=<pool id>
# subscription-manager repos --disable=*
# subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-7-server-extras-rpms
```

Downloading Tower package

After the server is subscribed successfully, complete the following steps:

1. Download the Red Hat Ansible Tower packages from the mirror site:

   ```
   # cd /opt ;  curl -k -O
   https://releases.ansible.com/awx/setup/ansible-tower-setup-latest.tar.gz
   ```

2. Extract the downloaded file to extract the content:

   ```
   # gunzip ansible-tower-setup-latest.tar.gz
   ```

3. Browse to the directory containing installers:

   ```
   # cd ansible-tower-setup-3.1.2
   ```

Updating the inventory file and running setup

Update the inventory file and run the installer, as shown in Example 3-38.

Example 3-38 Update inventory and install

```
# vi inventory

[tower]
 localhost ansible_connection=local

[database]

[all:vars]
 admin_password='redhat01'

pg_host=''
 pg_port=''

pg_database='awx'
 pg_username='awx'
 pg_password='redhat01'

rabbitmq_port=5672
 rabbitmq_vhost=tower
 rabbitmq_username=tower
 rabbitmq_password='redhat01'
 rabbitmq_cookie=cookiemonster

# Needs to be true for fqdns and ip addresses
 rabbitmq_use_long_name=false

# ./setup.sh
```

3.5.5 Configuring Red Hat Ansible Tower

This section describes how to configure Red Hat Ansible Tower. You add the license file to Red Hat Ansible Tower to finishing its setup.

Licensing Red Hat Ansible Tower

Red Hat Ansible Tower Provides a RESTful API, CLI, and UI.

Complete the following steps to connect to the UI:

1. Open the browser by using https and point to your Red Hat Ansible Tower IP or hostname:

   `https://<Ansible Tower IP or Hostname>`

2. Log in by using the user that you configured in the inventory file (in this case, `admin/redhat01`), as shown in Figure 3-85.

Figure 3-85 Red Hat Ansible Tower login pane

3. After you are logged in, configure the license. Tower license are in a file; therefore, browse to the file and accept the terms, as shown in Figure 3-86.

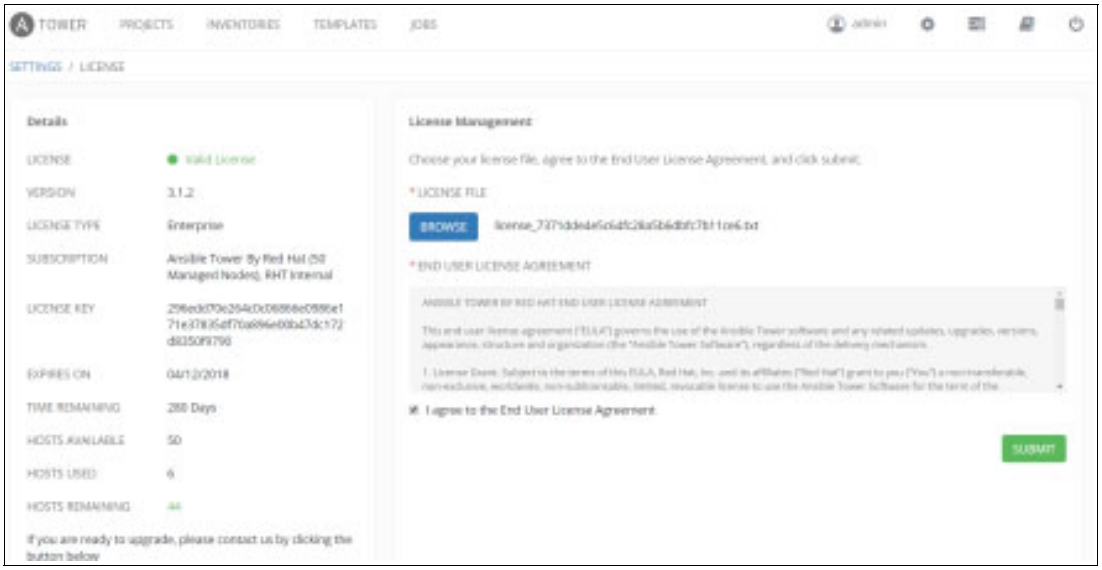


Figure 3-86 Red Hat Ansible Tower: License Management pane

3.5.6 Administering Red Hat Ansible Tower

This section describes an automation job procedure and the Red Hat Ansible Tower UI.

Red Hat Ansible Tower UI overview

After you apply a valid license to Red Hat Ansible Tower, you are redirected to the Red Hat Ansible Tower Dashboard that shows the Jobs Status, as shown in Figure 3-87.

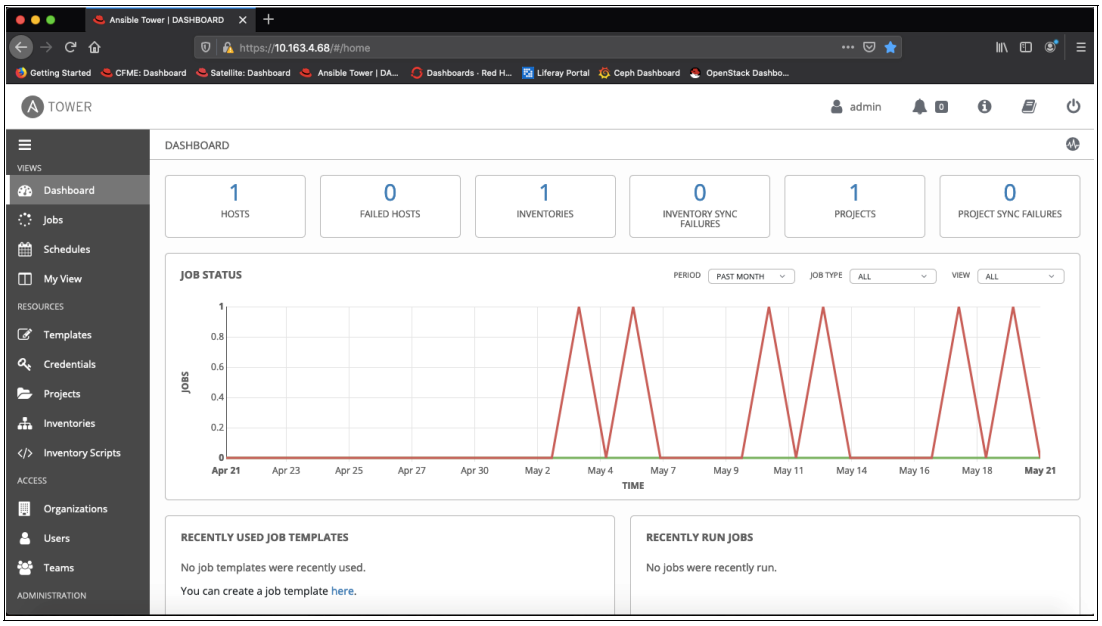


Figure 3-87 Red Hat Ansible Tower Dashboard

You can change the settings of Red Hat Ansible Tower from the Settings tab in the left control pane of the Red Hat Ansible Tower browser page, as shown in Figure 3-88.

Figure 3-88 Red Hat Ansible Tower Settings pane

To view and change system settings, such as Logging and Activity Stream, select **System Options** from the Settings page, as shown in Figure 3-89.

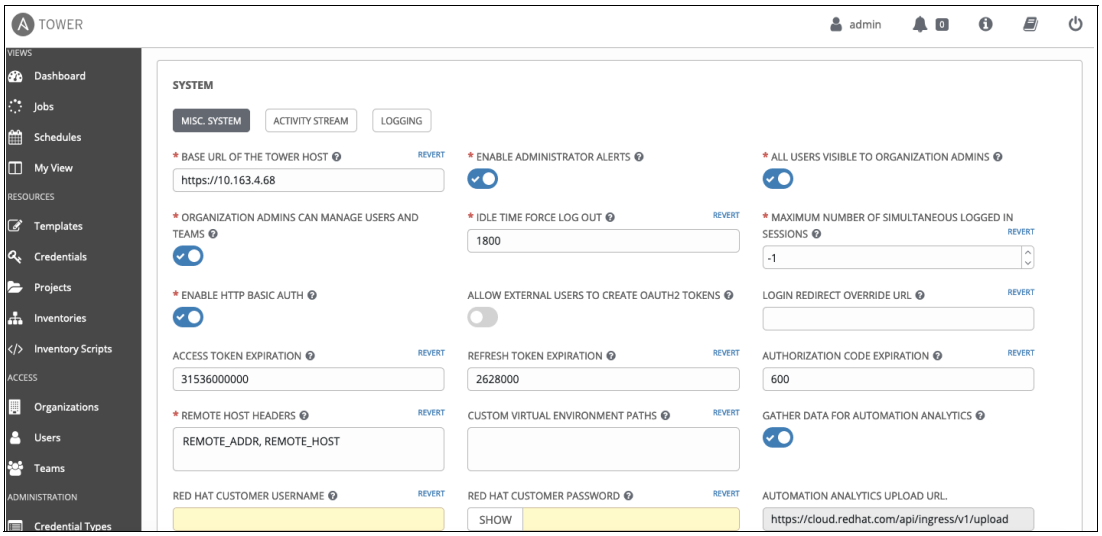


Figure 3-89 Red Hat Ansible Tower System pane

Running an automation job

By using Red Hat Ansible Tower, you can create jobs templates that contain plays and automation tasks.

Click **Templates** in the left control pane, as shown in Figure 3-90. You can run the job directly by clicking the Rocket symbol, or copy the template to create a custom one, or can delete the template.

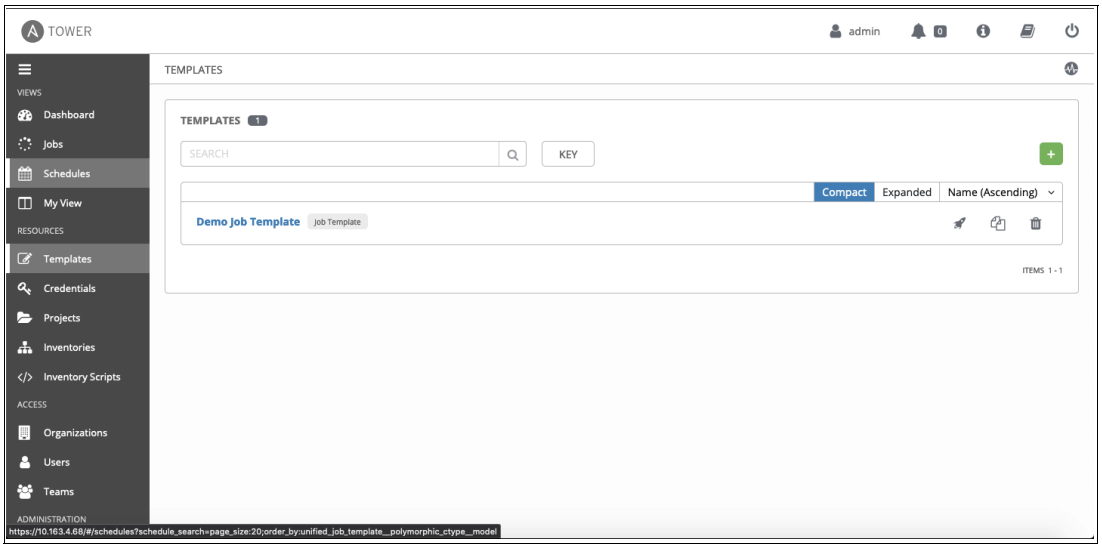


Figure 3-90 Red Hat Ansible Tower Templates pane

To create a Job template, click **+** . You must enter the details, such as Name, Inventory Details, Credentials, Playbook, and Repository details, as shown in Figure 3-91.

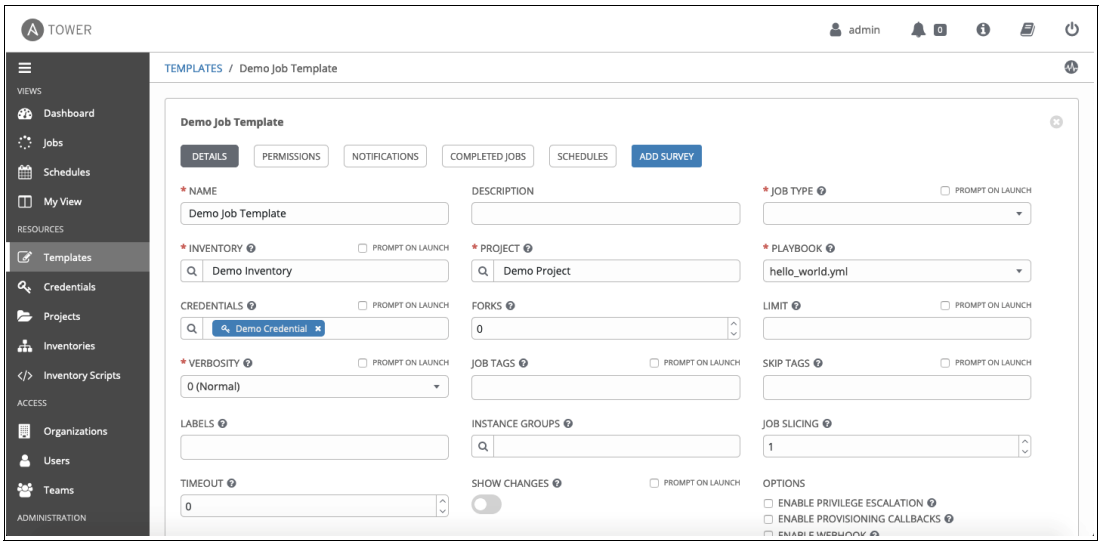


Figure 3-91 Red Hat Ansible Tower Templates: Demonstration job template

Before creating the Job template, check that an inventory of hosts was created where this template runs by selecting the **Inventories** tab from left control pane, as shown in Figure 3-92. After the Inventory is created, it is synced with the provider.

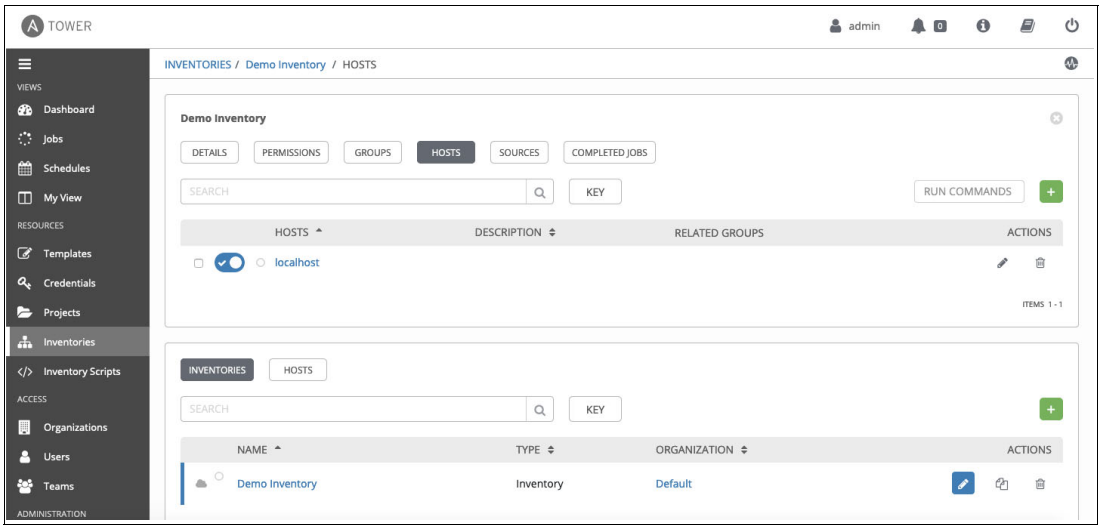


Figure 3-92 Red Hat Ansible Tower Inventory pane

You also must create the credential for running the Job template, as shown in Figure 3-93. You can create a credential or sync with the organization credentials, such as RBAC or AD system.

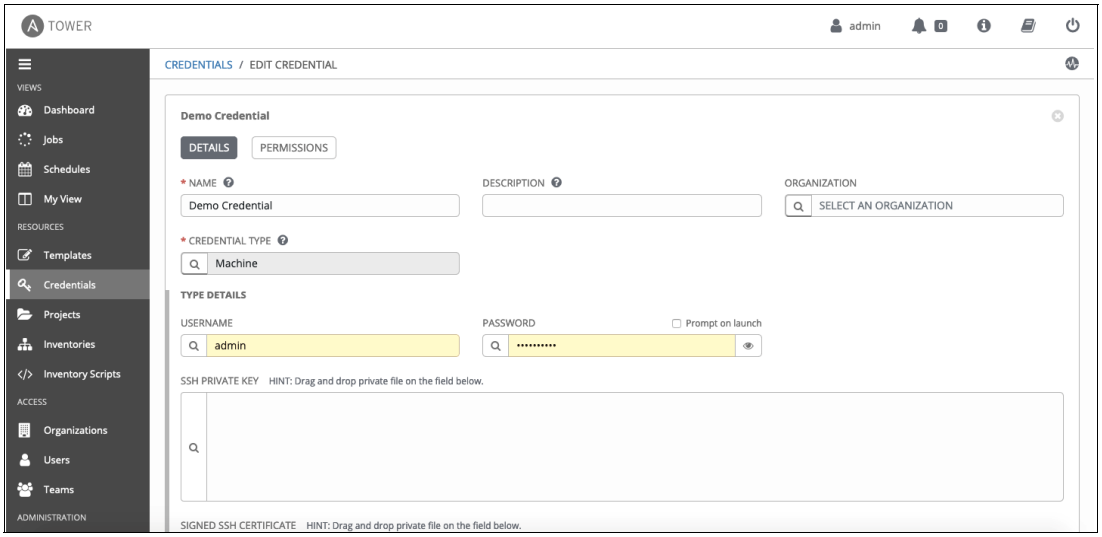


Figure 3-93 Red Hat Ansible Tower Credentials pane

Trigger the job by clicking the Rocket icon. You see the status from the output in the right half of the running jobs window, as shown in Figure 3-87 on page 145.

3.6 Red Hat Satellite

This section describes the Red Hat Systems Management Tool that is known as Red Hat Satellite. We see customers that manage their internal IT environment with the least amount of changes. When they surround those standardized configurations with operational and change management, they can enjoy the best return on that investment.

Red Hat Satellite is an infrastructure management product that keeps Red Hat Enterprise Linux environments and other Red Hat infrastructure running efficiently with security and is compliant with various standards.

3.6.1 Introduction

Red Hat Satellite is a system management solution that enables you to deploy, configure, and maintain your systems across physical, virtual, and cloud environments. Red Hat Satellite provides provisioning, remote management, and monitoring of multiple Red Hat Enterprise Linux deployments with a single, centralized tool.

Red Hat Satellite Server synchronizes the content from Red Hat Customer Portal and other sources, and provides functions, including fine-grained lifecycle management, user and group role-based access control, integrated subscription management, and advanced GUI, CLI, or API access.

Red Hat Satellite Capsule Server mirrors content from Red Hat Satellite Server to facilitate content federation across various geographical locations. Host systems can pull content and configuration from the Capsule Server in their location and not from the central Red Hat Satellite server. The Capsule Server also provides localized services, such as Puppet Master, DHCP, DNS, or TFTP. Capsule Servers assist you in scaling your Red Hat Satellite environment as the number of your managed systems increases.

The following steps are key for many organizations:

- Create a corporate-standard server configuration (or several configurations) for common workloads. Establish a robust plan for lifecycle management, day-to-day operational management, and patches and security updates. Establish a way to deliver those patches and updates quickly and efficiently.
- Inventory servers and determine what systems meet that corporate standard and what systems are close and can be brought into compliance with the corporate standard.
- Establish a road map to consolidate outlier systems to the corporate-standard system.
- Continue to optimize and improve the standard configuration and the surrounding management infrastructure.

3.6.2 System architecture

This section provides an overview of the system design and high-level system architecture of Red Hat Satellite V6 (see Figure 3-94).

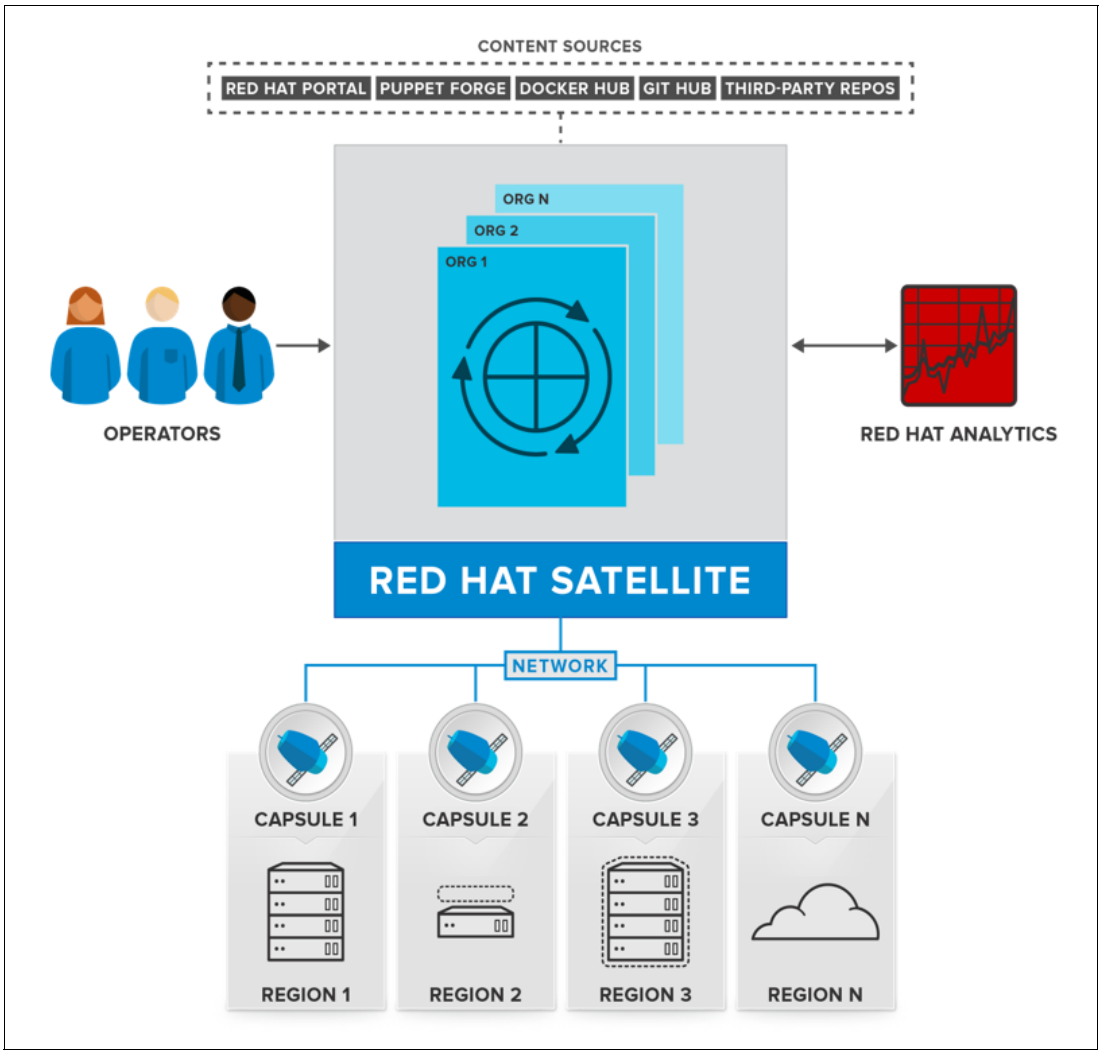


Figure 3-94 Red Hat Satellite system architecture

As shown in the system architecture that is shown in Figure 3-94, there are four stages through which content flows in this architecture:

- External content sources

  The Red Hat Satellite Server can use diverse types of content from various sources. The required connection is the one with Red Hat Customer Portal, which is the primary source of software packages, errata, Puppet modules, and container images.

- Red Hat Satellite server

  The Red Hat Satellite server enables you to plan and manage the content lifecycle and the configuration of Capsule Servers and hosts through GUI, CLI, or API.

- Capsule servers

  Capsule Servers mirror content from the Red Hat Satellite server to establish content sources in various geographical locations. This feature enables host systems to pull content and configuration from the Capsule Servers in their location and not from the central Red Hat Satellite server.

- Managed hosts

  Hosts are the recipients of content from Capsule Servers. The Red Hat Satellite server can include directly managed hosts.

3.6.3 System components

This section describes the component of Red Hat Satellite V6, which consists of several open source projects that are integrated, verified, delivered, and supported as Red Hat Satellite V6.

Red Hat Satellite V6 consists of the following open source projects:

- Foreman

  Foreman is an open source application that is used for the provisioning and lifecycle management of physical and virtual systems. Foreman automatically configures these systems by using various methods, including kickstart and Puppet modules. Foreman also provides historical data for reporting, auditing, and troubleshooting.

- Katello

  Katello is a Foreman plug-in for subscription and repository management. It provides a means to subscribe to Red Hat repositories and download content. You can create and manage different versions of this content and apply them to specific systems within user-defined stages of the application lifecycle.

- Candlepin

  Candlepin is a service within Katello that handles subscription management.

- Pulp

  Pulp is a service within Katello that handles repository and content management. Pulp ensures efficient storage space by not duplicating RPM packages, even when requested by content views in different organizations.

- Hammer

  Hammer is a CLI tool that provides command line and shell equivalents of most Web UI functions.

- REST API

  Red Hat Satellite V6 includes a RESTful API service that allows system administrators and developers to write custom scripts and third-party applications that interface with Red Hat Satellite.

Figure 3-95 shows the high-level deployment architecture with detailed system components of the Red Hat Satellite server. Also shown is protocol level view of how Red Hat Satellite works in managing the system components.

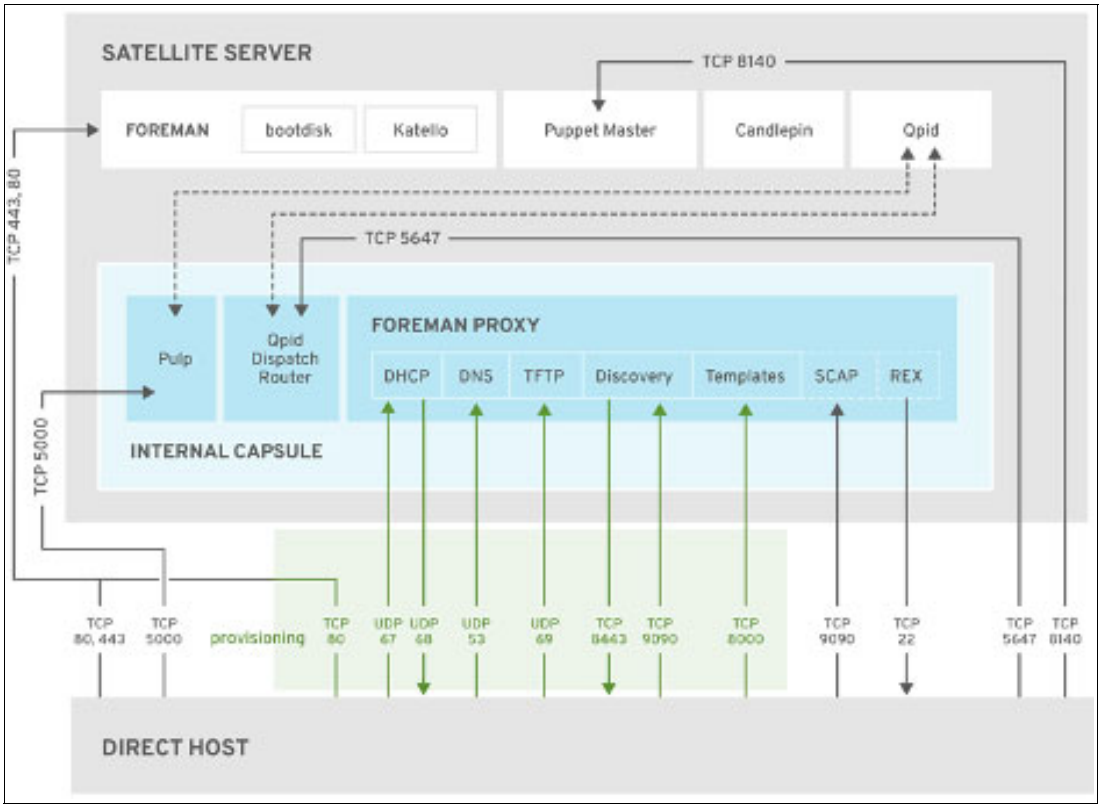


Figure 3-95 Red Hat Satellite server high level architecture

3.6.4 Prerequisites for deploying Red Hat Satellite

This section describes the requirements for deploying Red Hat Satellite. For this scenario, we selected a configuration to deploy Red Hat Satellite server on IBM Cloud with the hardware requirements that are listed in Table 3-9.

Table 3-9 Configuration for deploying Red Hat Satellite server

CPU	16 vCPU
RAM	32 GB
Storage	1x500 GB and 1x100GB
Network interfaces	Min 2x1G or 2x10G interfaces if 1G not available

The following components also are required:

- A unique host name, which can contain lowercase letters, numbers, dots (.), and hyphens (-)
- A valid Red Hat Satellite Server and Smart Management Subscription that must be activated and assigned to RHN ID
- Public and Private Network provisioned with internet or proxy connection
- NTP Sync

- Access to all TCP and UDP ports, as shown in Figure 3-95 on page 152
- RHSM access for subscribing Red Hat Enterprise Linux server
- Red Hat Enterprise Linux V7 OS ISO image to deploy on provisioned machine (physical or virtual)

3.6.5 Installing Red Hat Satellite

This section describes how Red Hat Satellite is deployed on the IBM cloud machine. Red Hat Satellite can be deployed on bare metal or a virtual server, depending on the number of managed nodes or cluster complexity within data center or across.

Downloading Red Hat Satellite and Red Hat Enterprise Linux server ISO

Complete the following steps:

1. At this website, log in to the Red Hat Customer Portal by using your customer account information, as shown in Figure 3-96.

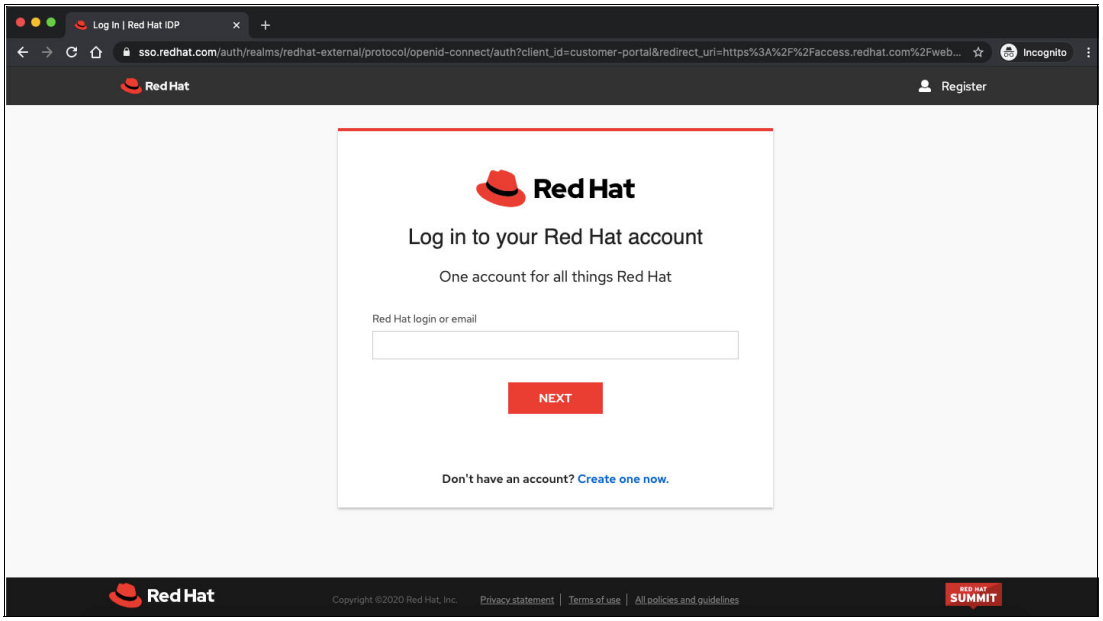


Figure 3-96 Red Hat Customer portal login page

2. Click **Downloads** in the menu bar, as shown in Figure 3-97.

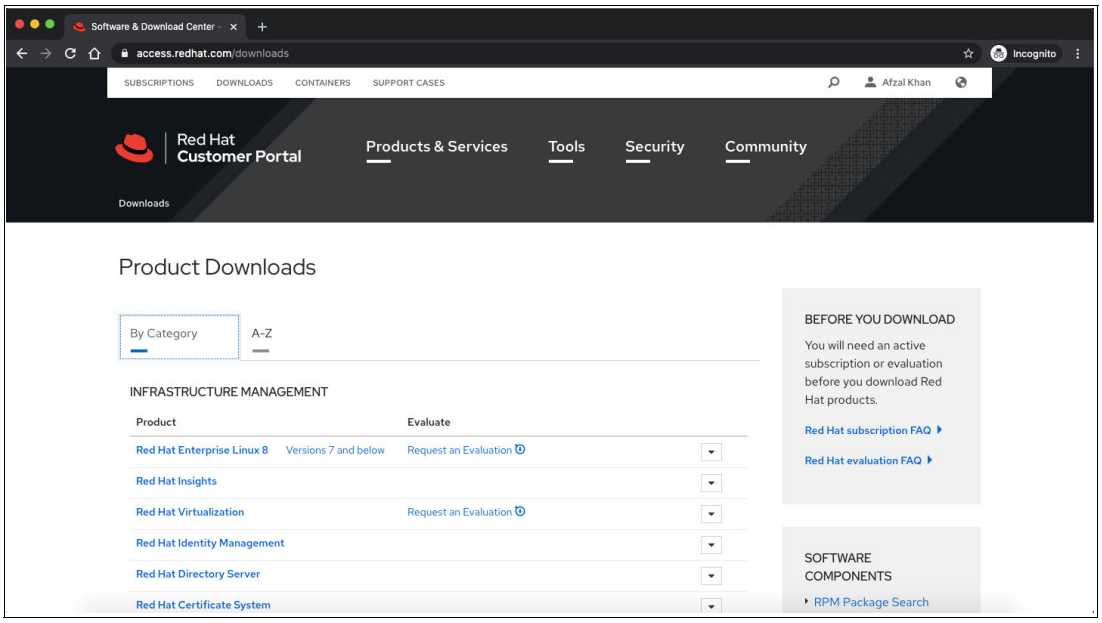


Figure 3-97 Red Hat Customer Portal Products Download Page

3. Click **Red Hat Satellite** to download Red Hat Satellite binary ISO, or select **Red Hat Enterprise Linux V7 ISO** to install on the server (see Figure 3-98).

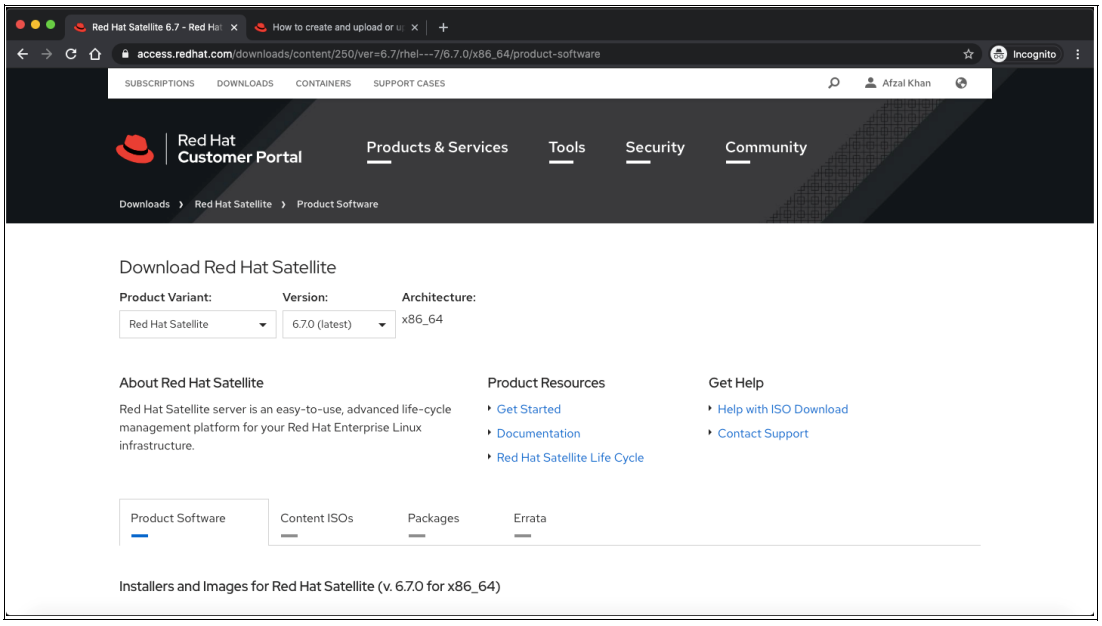


Figure 3-98 Red Hat Customer Portal: Download Red Hat Satellite

4. Select the ISO variant and click **Download Now**, as shown in Figure 3-99.

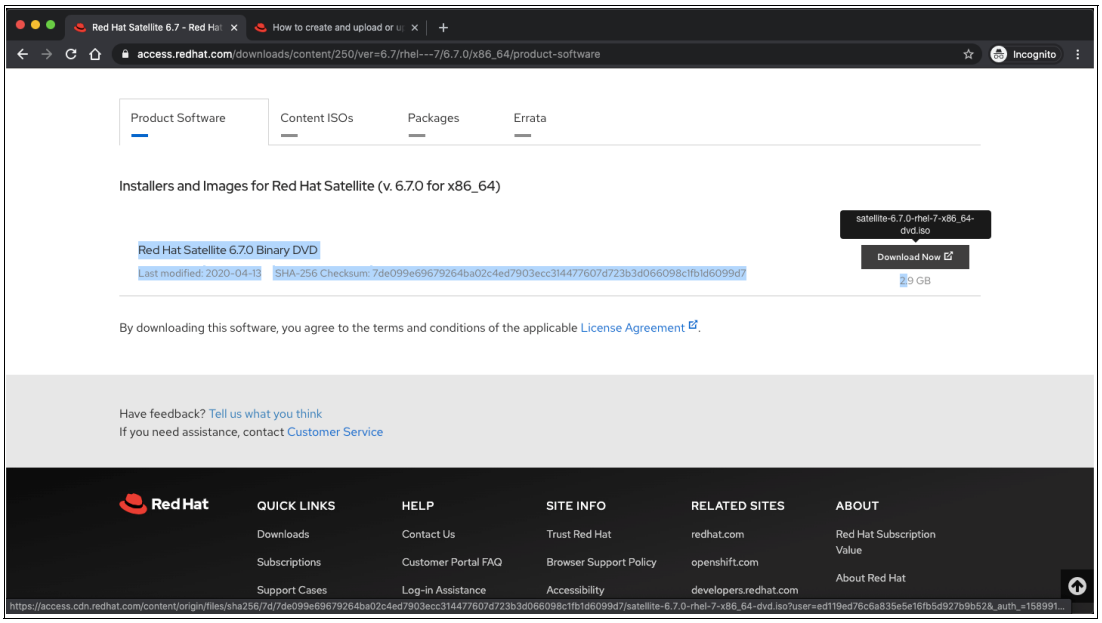


Figure 3-99 Installers and Images for Red Hat Satellite window

Note: You need a valid Red Hat Satellite or Red Hat Enterprise Linux System Management Subscription to download the images.

Server provisioning

Complete the following steps to create satellite VM and install Red Hat Enterprise Linux:

1. Provision a VM with the specified configuration in the prerequisite section.
2. Attach the downloaded Red Hat Enterprise Linux or Red Hat Satellite ISO image to virtual CD/DVD-ROM of the VM.
3. Set the boot order to CD/DVD-ROM as priority.
4. After the attach is successful, boot the VM with attached Red Hat Enterprise Linux V7 or higher.
5. On the console, press Enter to continue installing the operating system.
6. Set the user name and password for the root and user.
7. After the operating system installation is complete and machine is rebooted successfully, SSH to the machine by using the root credentials.

In the next section, you perform a subscription activation on the installed machine.

Subscription activation of Red Hat Enterprise Linux

Follow the procedure that is described in Appendix A, “Red Hat subscription activation process” on page 401 to subscribe the Red Hat Enterprise Linux host to the RHN, and enable required software repositories. For reference, high-level steps must be performed, as shown in Example 3-39.

Example 3-39 Steps to subscribe the host to the RHN

```
//SSH as root user to the machine
# subscription-manager register
Username: xxxxxx
Password: xxxxxx
The system has been registered with ID: 541084ff2-44cab-4eb1-9fa1-7683431bcf9a
# subscription-manager attach --pool=pool_id
# subscription-manager repos --disable "*"

# subscription-manager repos --enable=rhel-7-server-rpms
\--enable=rhel-7-server-satellite-6.6-rpms
\--enable=rhel-7-server-satellite-maintenance-6-rpms
\--enable=rhel-server-rhscl-7-rpms \
--enable=rhel-7-server-ansible-2.8-rpms
```

Installing the Red Hat Satellite server

To install the Red Hat Satellite server, you must install the required packages. Check that the required repositories are attached to Red Hat Enterprise Linux machine, as shown in “Subscription activation of Red Hat Enterprise Linux” on page 156.

Now, install the Red Hat Satellite packages. Use the satellite-installer to install the Red Hat Satellite and its components:

```
#yum install satellite
#satellite-installer --scenario satellite \
--foreman-initial-admin-username admin \
--foreman-initial-admin-password redhat \
--foreman-proxy-puppetca true \
--foreman-proxy-tftp true \
--enable-foreman-plugin-discovery
```

Access the Red Hat Satellite UI by using the public IP address of the Red Hat Enterprise Linux machine, and confirm the credentials passed to the satellite-installer worked.

Generating and importing Subscription Manifest in Red Hat Satellite

A Subscription Manifest is a set of encrypted files that contains subscription information. It is imported into the Red Hat Satellite server to access the Content Delivery Network and find the repositories available from the Products of the Subscriptions.

Complete the following steps to create a Manifest:

1. At the Red Hat Customer Portal,click **Subscriptions** → **Subscription Allocations** → **New Subscription Allocation** and enter the name and type.
2. Add the subscriptions by browsing to the **Subscriptions** tab and click **Add subscriptions**.
3. The filter can be used to search or manually browse to the required subscriptions and enter the number of entitlements needed. Click **Submit**.
4. Click **Export Manifest** to download it to the system.

5. Upload the manifest to the Red Hat Satellite by following the steps that are described at this web page.

Complete the following steps to update a manifest:

1. At the Red Hat Customer Portal, click **Subscriptions** → **Subscription Allocations**. Select the manifest or satellite profile and then, click **Subscriptions** → **Add Subscriptions**.
2. Manually browse to the required subscriptions or use the filter to search. Enter the number of entitlements that are needed. Click **Submit**.
3. Upload or refresh the updated manifest to the Red Hat Satellite by following the steps that are described at this web page.

In Red Hat Satellite V6.4, the manifest can also be managed directly in the Red Hat Satellite WebUI. Select **Content** → **Subscriptions** → **Add subscriptions**. Enter the number of entitlements that are needed in the Quantity to Allocate field. Click **Submit**.

The entitlements are now bound to the manifest and refreshes it.

For more information about generating and importing the manifest, see this web page.

3.6.6 Configuring Red Hat Satellite

This section describes configuring a deployed Red Hat Satellite server, including repository sync, creating content a view, publishing a content view, and creating activation keys.

Synchronizing repositories with RHN on Red Hat Satellite server

Complete the following steps to synchronize the repositories:

1. In the Red Hat Satellite web UI, select **Content** → **Red Hat Repositories**.
2. In the Search field, enter the following repository name: `Red Hat Enterprise Linux V7 server (RPMs)`.
3. In the **Available Repositories** pane, click **Red Hat Enterprise Linux V7 server (RPMs)** to expand the repository set.
4. For the x86_64 7.5 entry, click the **Enable** icon to enable the repository.
5. In the Search field, enter the following repository name: `Red Hat Satellite Tools V6.6 (for Red Hat Enterprise Linux V7 server) (RPMs)`.
6. In the **Available Repositories** pane, click **Red Hat Satellite Tools V6.6 (for Red Hat Enterprise Linux V7 server) (RPMs)** to expand the repository set.
7. For the x86_64 entry, click the **Enable** icon to enable the repository.
8. In the Search field, enter the following repository name: `Red Hat Enterprise Linux Server 7 (Kickstart)`.
9. In the **Available Repositories** pane, click **Red Hat Enterprise Linux V7 Server (Kickstart)** to expand the repository set.
10. For the x86_64 7.5 entry, click the **Enable** icon to enable the repository.

Creating content view on Red Hat Satellite

To create content view, complete the following steps:

1. Click **Content** → **Content Views** and then, click **Create New View**.
2. In the Name field, enter: `RHEL7 x86_64`.

   The label is automatically populated.
3. Clear the **Composite View?** check box and click **Save**.

Publishing a content view on Red Hat Satellite

To publish a content view, follow these steps:

1. Click **Content** → **Content Views**.
2. Click the name of the Content View that you want to publish.
3. Click **Publish New Version** to display the Publish New Version page.

   This selection specifies the version and allows you to enter a comment to reflect any changes that were made to the Content View.
4. Click **Save** to publish the Content View to the library.

   You can monitor the publication progress on the window that appears.
5. When the publishing process is complete, click **Promote**.

   The list of available promotion paths (**Library** → **Dev** → **QA**) displays.
6. Select the **Dev environment** check box, and then, click **Promote Version**.

Creating activation keys on Red Hat Satellite server

To create the activation keys, complete the following steps:

1. On the main menu, click **Content** → **Activation Keys** → **New Activation Key**.
2. In the Name field, enter a name.
3. If applicable, clear the **Content Host Limit** check box.

   You can use this field to control how many times a specific activation key is used. For example, if you associate the key with a subscription that has a limited quantity, you can set the limit on the activation key to eliminate exceeding that quantity.
4. Select the check box for your environment.
5. In the Content View drop-down list, select the **RHEL 7 x86_64** content view, and then, click **Save**.
6. In the Activation Keys page, click the **Subscriptions** tab, and then, click the **Add** tab to display the list of available subscriptions.
7. From the list of available subscriptions, select the subscriptions that you want to add.
8. Click **Add Selected**.

3.6.7 Administering Red Hat Satellite

In this section, we describe how to administer the Red Hat Satellite server to work with managed host and perform systems management operations for Red Hat systems that are registered with Red Hat Satellite and RHN. You can also see the subscription management of Red Hat Enterprise Linux machines from Red Hat Satellite.

Registering hosts

To register hosts with Red Hat Satellite and enable the needed repositories on it, follow the Red Hat Ansible playbook that can be run on the Red Hat Satellite server, as shown in Example 3-40.

Note: Red Hat Satellite features an integration with Red Hat Ansible Automation platform.

Example 3-40 Red Hat Ansible playbook to register existing hosts

```
---
- hosts: all
  vars:
    satelliteip: 169.38.105.150
  tasks:
    - name: Download and install a copy of the CA Certificate for the Red Hat
Satellite 6 server
      yum:
        disable_gpg_check: yes
        name:
http://{{satelliteip}}/pub/katello-ca-consumer-red-satellite.redbooks.info-1.0-1.n
oarch.rpm
        state: present

    - name: delete the rhn registration locally
      command: mv /etc/sysconfig/rhn/systemid /etc/sysconfig/rhn/systemid.bak
      ignore_errors: True

    - name: Register with activation key matching Red Hat Enterprise Server
Version 7
      redhat_subscription:
        state: present
        activationkey: <<activation key name>>
        org_id: Default_Organization

    - name: Enable all the needed repositories
      rhsm_repository:
        name: rhel-7-server-*
        state: enabled

    - name: Install katello-agent and katello-host-tools
      yum:
        name:
          - katello-agent
          - katello-host-tools
          - insights-client
          - ansible
        state: present

    - name: register insights client
      shell: insights-client --register
```

Red Hat Satellite UI overview

Login to Red Hat Satellite server UI by using the IP address and admin credentials, as shown in Figure 3-100.

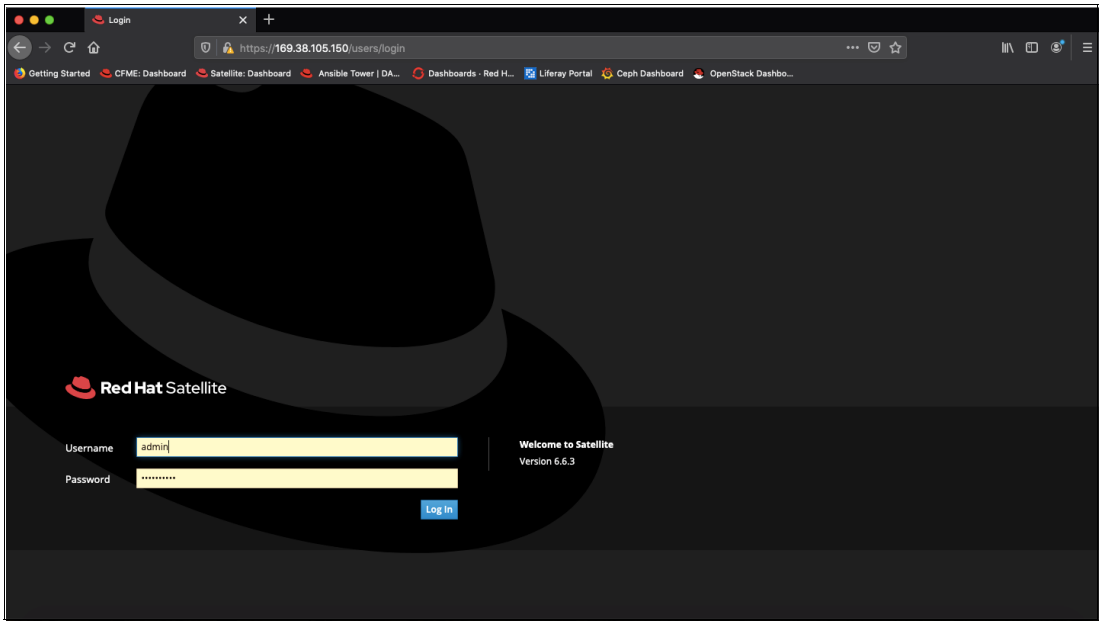


Figure 3-100 Red Hat Satellite: Login portal

After you are logged in, click the **Monitor** tab in the left control pane. You see the Overview page that includes information about Host Configurations Status, as shown in Figure 3-101. Click any of the configurations to perform a systems management task.

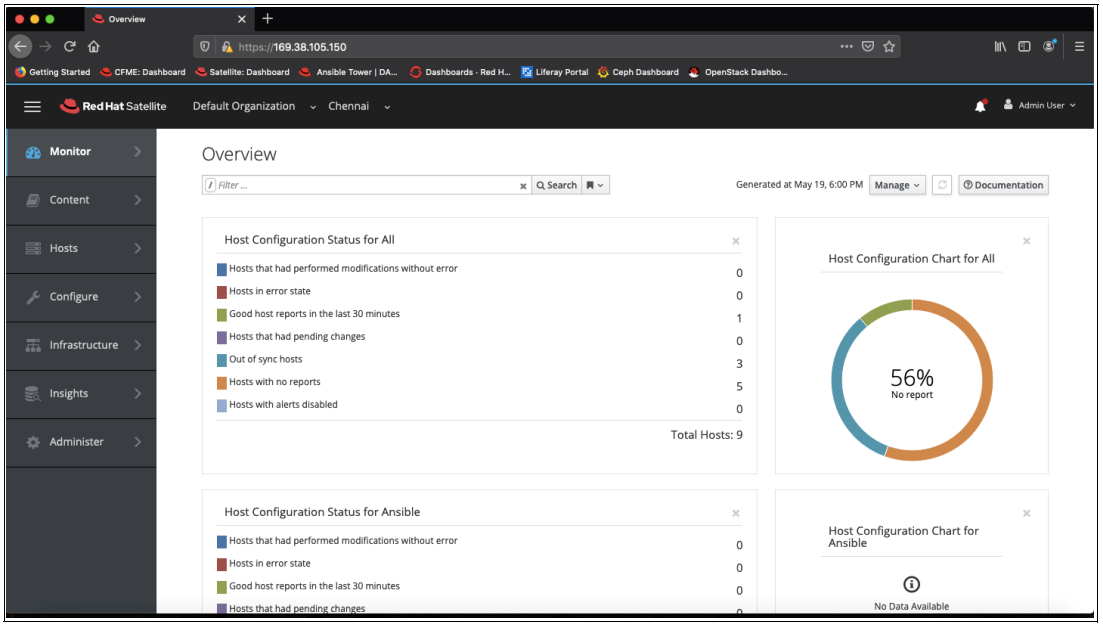


Figure 3-101 Red Hat Satellite: Overview window

In the **Monitor** tab, click **Statistics**. This view shows various statistics on the connected and registered hosts, such as CPUs and Memory, as shown in Figure 3-102.

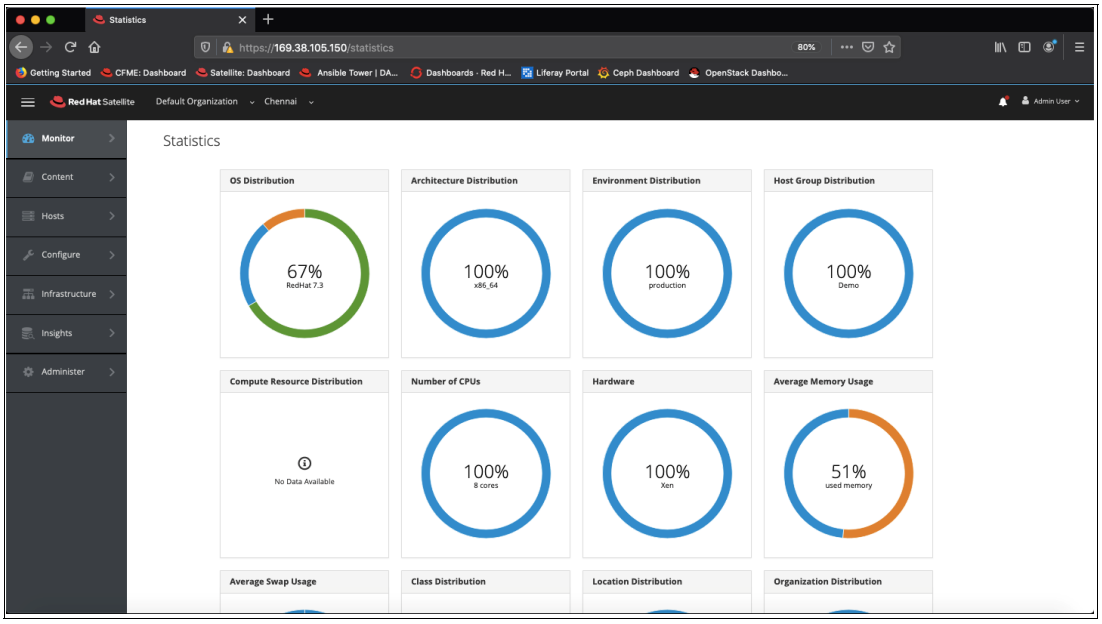


Figure 3-102 Red Hat Satellite: Monitor Statistics window

In the **Monitor** tab (see Figure 3-102), click the **Job Invocations** option to list all of the jobs that were started for the registered systems, as shown in Figure 3-103.

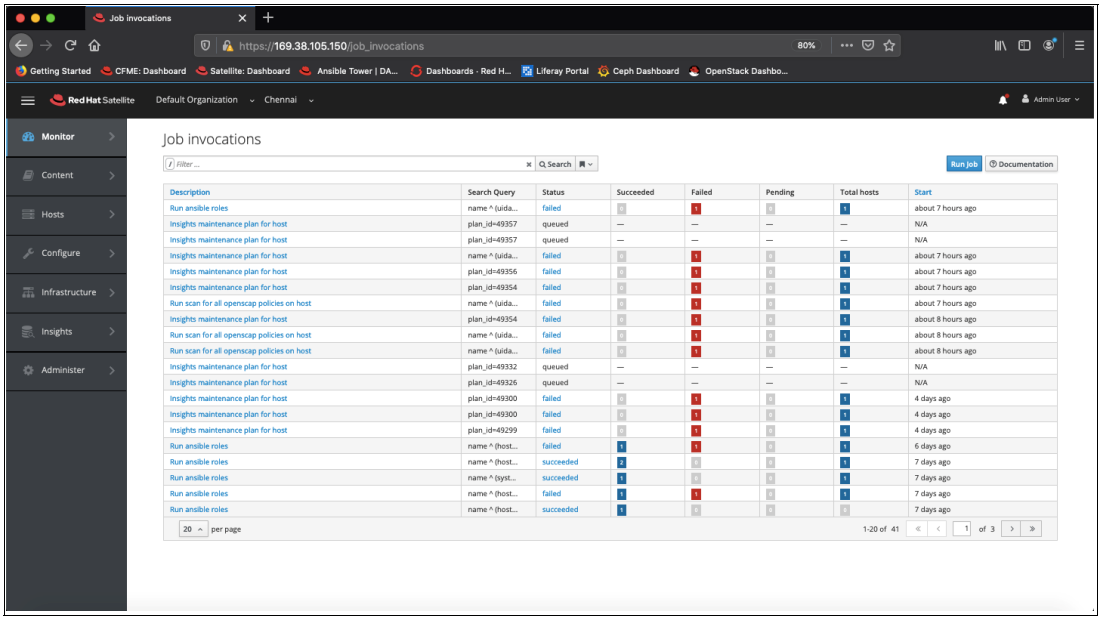


Figure 3-103 Red Hat Satellite: Monitor Job invocations window

Subscription management

Click the **Content** tab in the left control pane to view contents, such as subscriptions details, errata, advisories, and others, as shown in Figure 3-104. Add new subscriptions by clicking **Add Subscriptions** and enter details of the subscriptions.

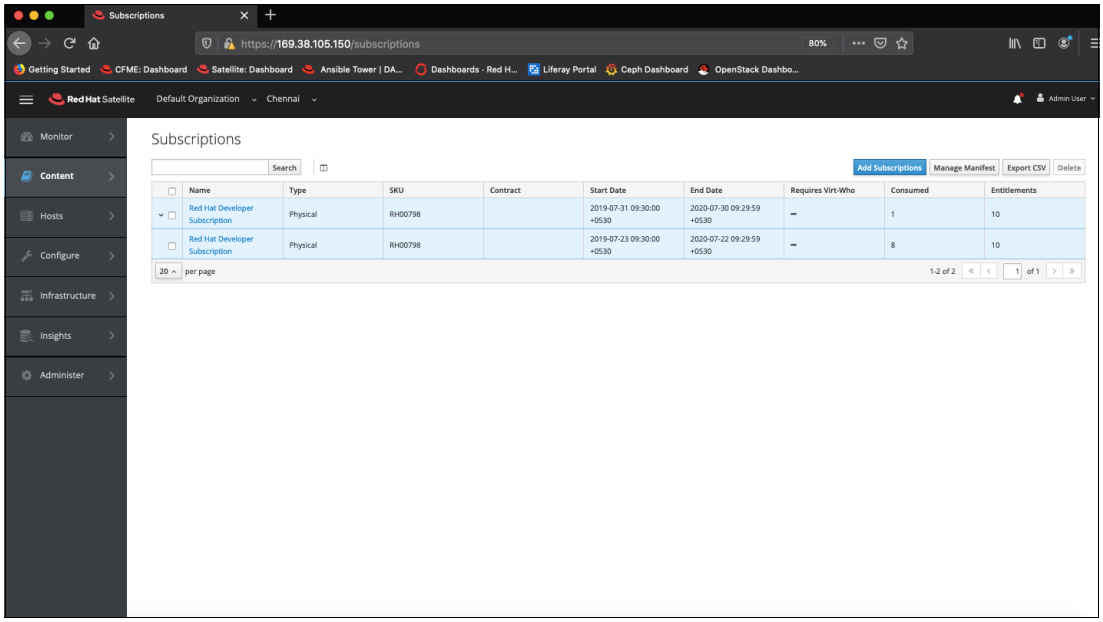


Figure 3-104 Red Hat Satellite: Content Subscriptions window

Errata management

Select **Errata** from the **Content** tab in left control pane. You see all of the Red Hat published errata. Select any errata and click **Apply** to apply a specific group of hosts or any specific host, as shown in Figure 3-105.

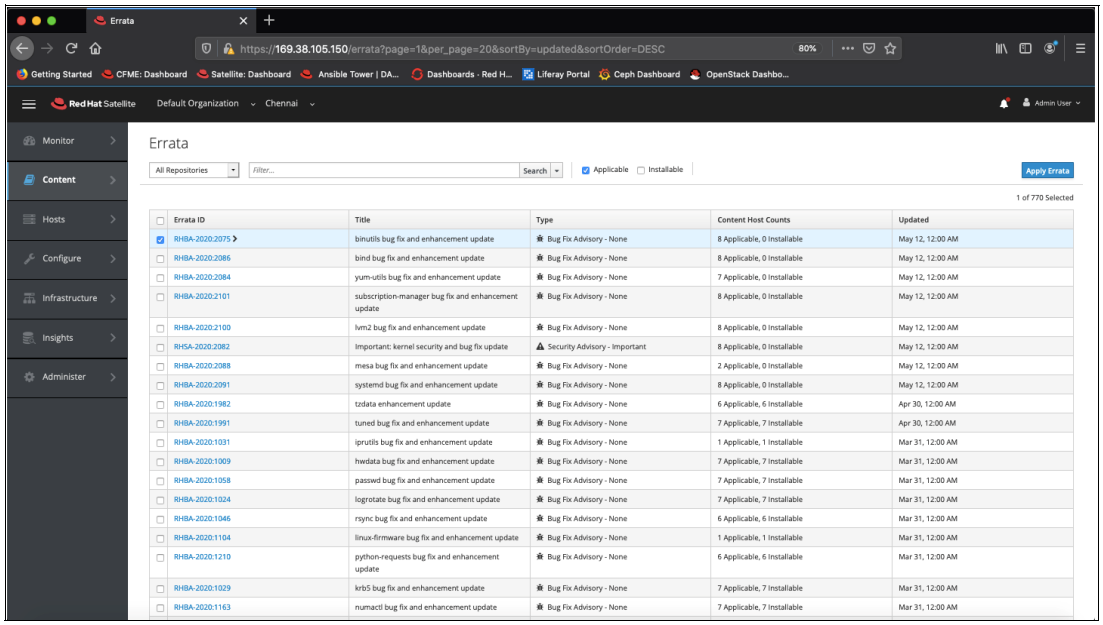


Figure 3-105 Red Hat Satellite: Content Errata window

Hosts management

Complete the following steps:

1. In the **Hosts** tab in left control pane, you see the list of hosts that are registered with the Red Hat Satellite server, as shown in Figure 3-106.

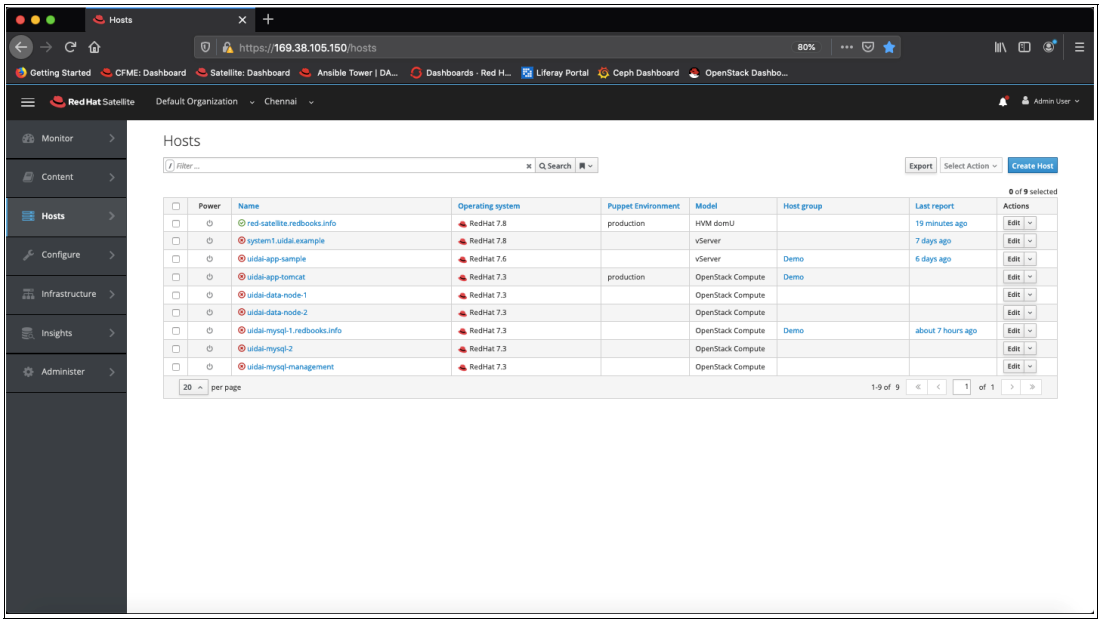


Figure 3-106 Red Hat Satellite: Hosts window

2. Select any host from the list. You see more information about the host, its runtime statistics, and its status, as shown in Figure 3-107.

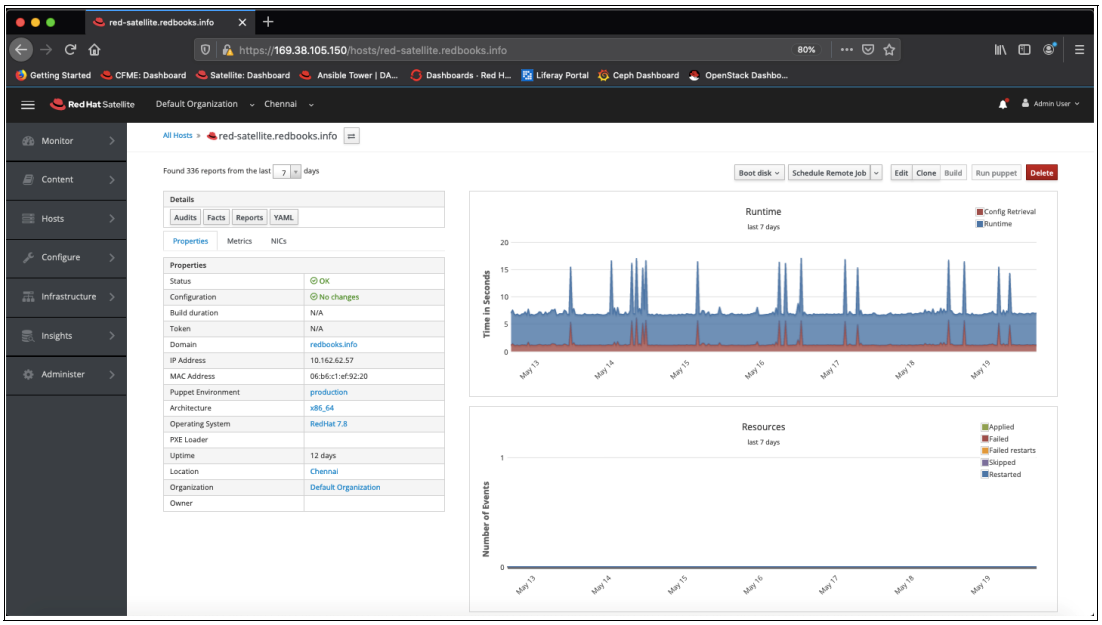


Figure 3-107 Red Hat Satellite: Host details pain

3. You can also provision a host by clicking **Create Host**. Enter the required details, such as configuration parameters, operating system, and interfaces, as shown in Figure 3-108. Red Hat Satellite interacts with the provider manager and starts a play to create a host.

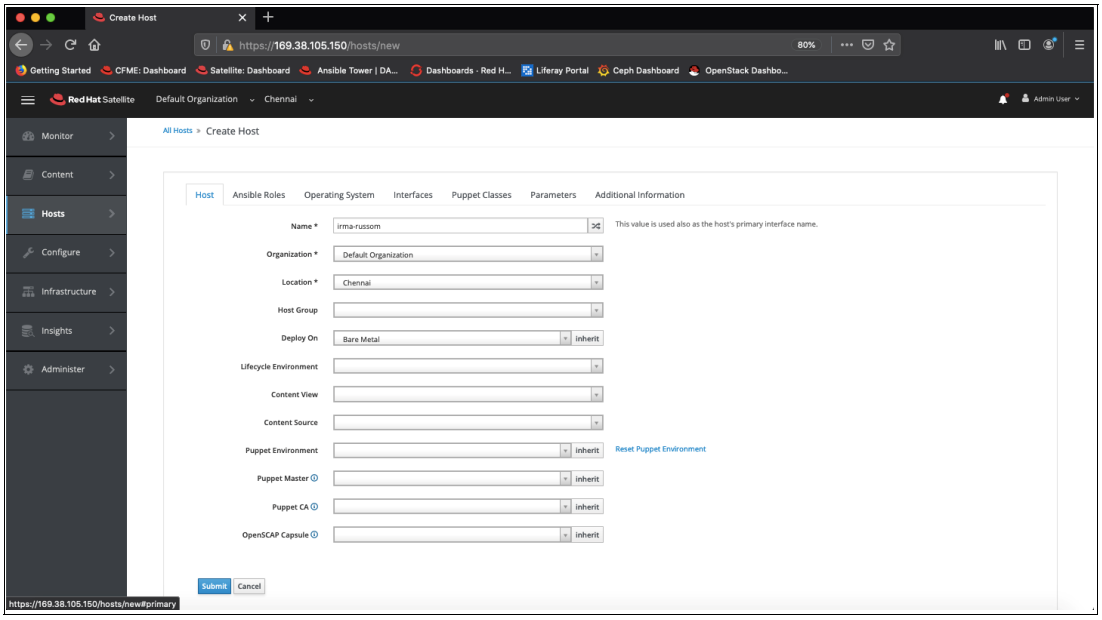


Figure 3-108 Red Hat Satellite: Create Host window

A host is created, as shown in Figure 3-109. After you select the host, a details page opens. You can view the compliancies or installable errata of the hosts by clicking any of it.

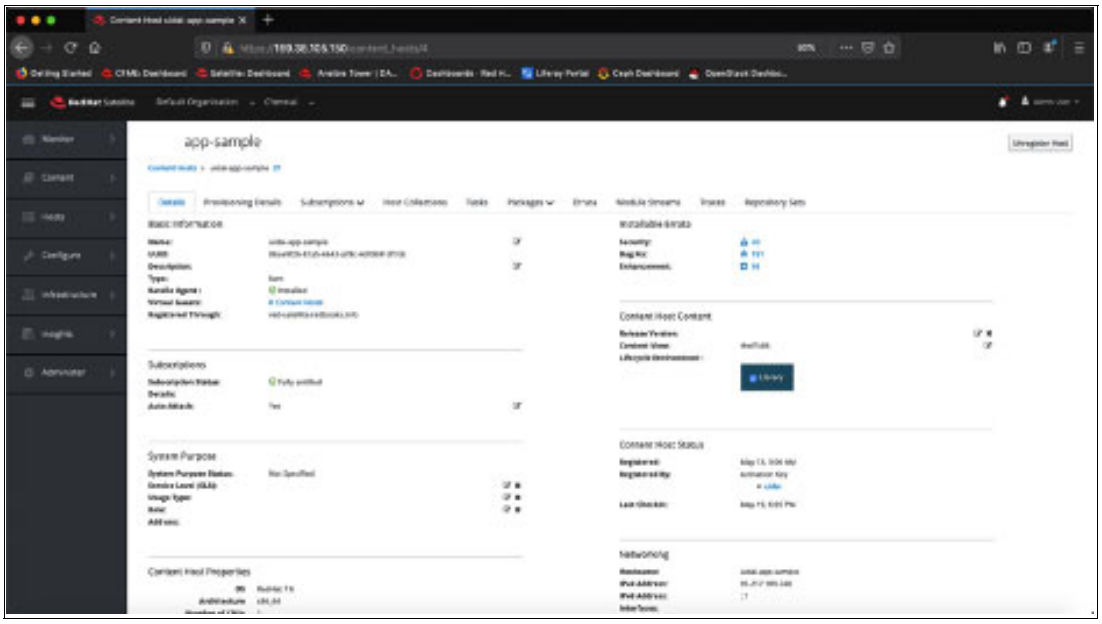


Figure 3-109 Red Hat Satellite: Content Hosts window

4. You see list of all the installable errata, as shown in Figure 3-110. Select any or all to install them to the host.

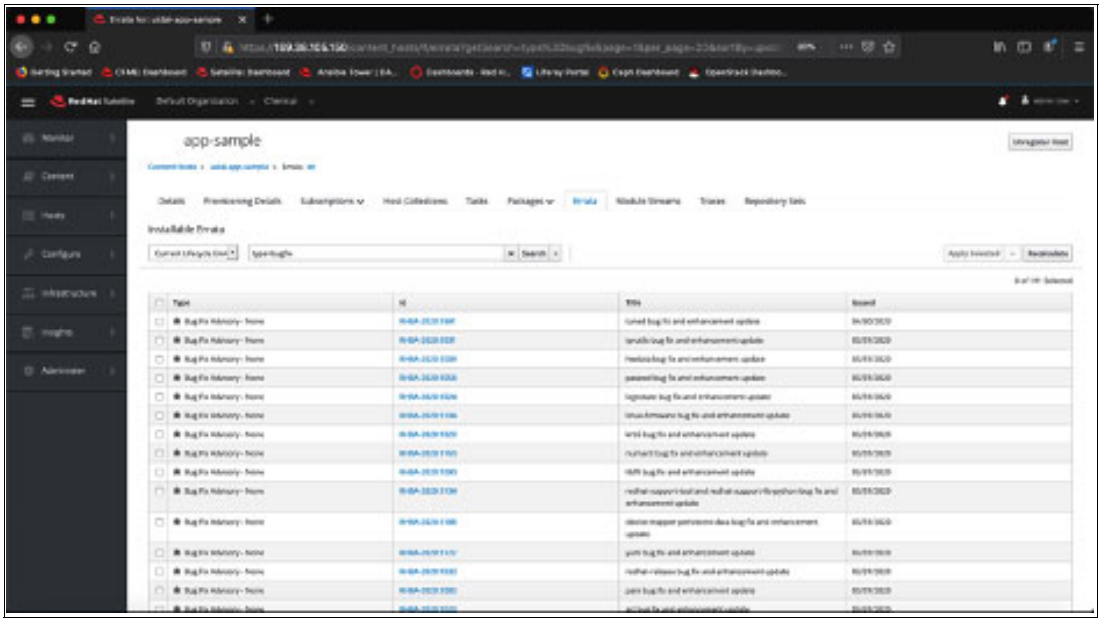


Figure 3-110 Red Hat Satellite: Installable Errata window

For more information, see 3.6.8, “Red Hat Satellite use-cases for deployed Red Hat Cloud” on page 168.

Provisioning management

Click **Hosts** → **Provisioning Templates** to list all the available provisioning templates on Red Hat Satellite. You can create a custom template by clicking **Create Template**, as shown in Figure 3-111.

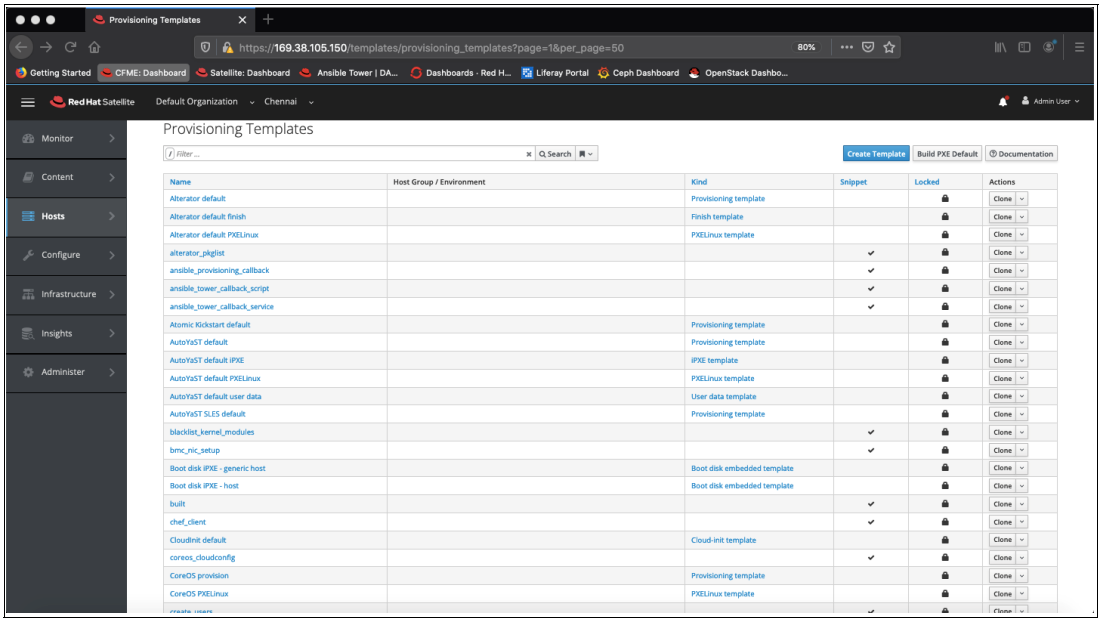


Figure 3-111 Red Hat Satellite: Provisioning Templates window

To edit any template, click the template to change the template and then save it, as shown in Figure 3-112.

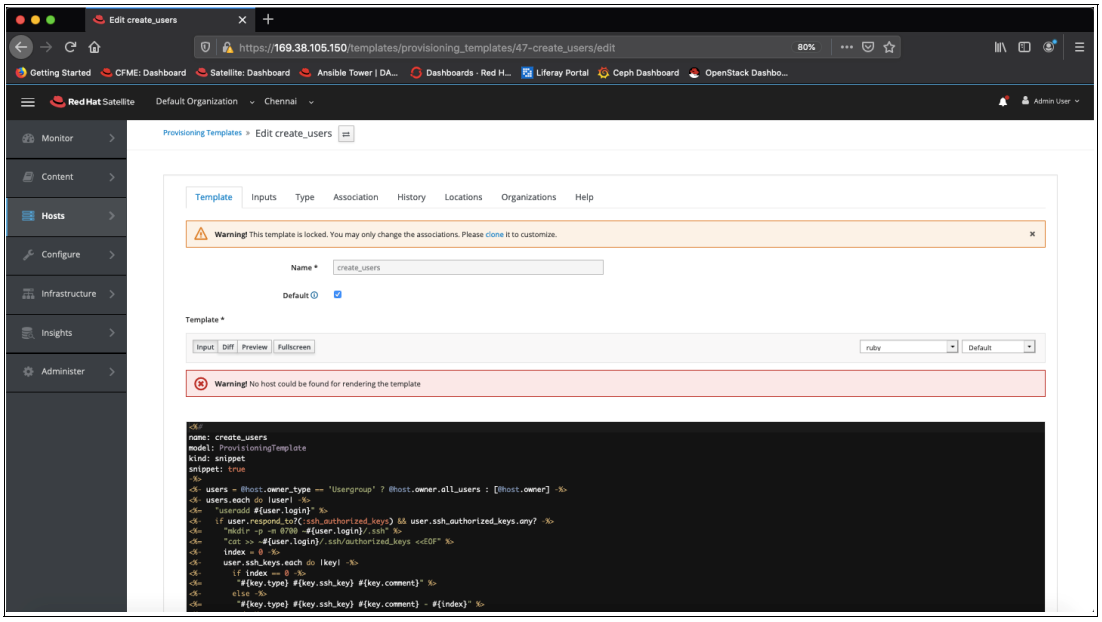


Figure 3-112 Red Hat Satellite: Editing templates window

You can view a list of capsules in the Red Hat Satellite server by clicking the **Infrastructure** tab in the left control pane and then selecting the **Capsules** option. Then, select the configured capsule to view its details, as shown in Figure 3-113.

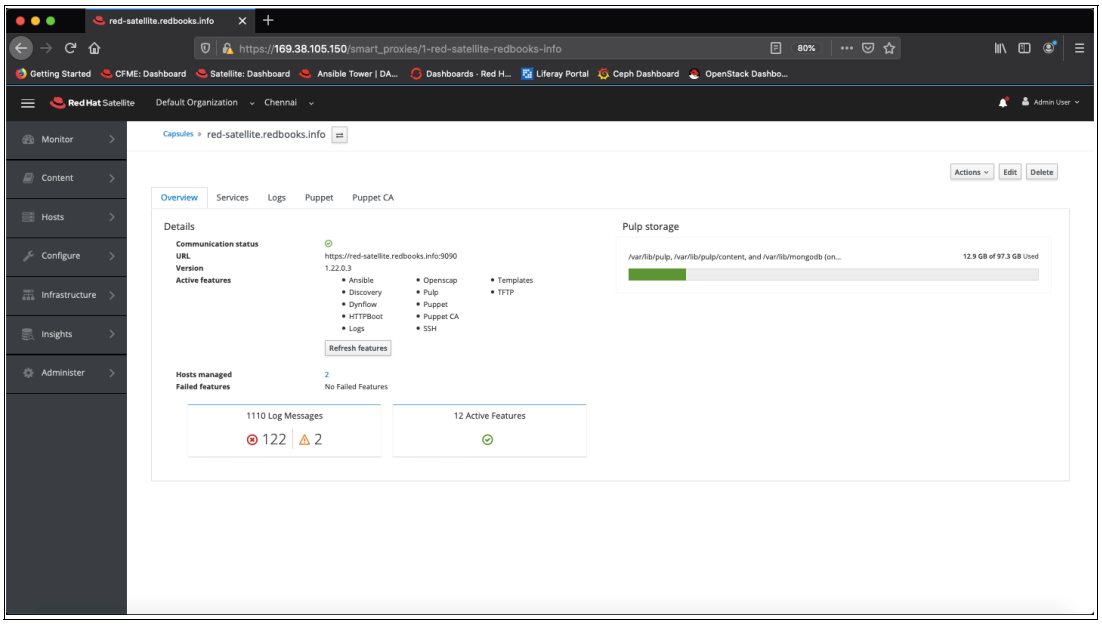


Figure 3-113 Red Hat Satellite: Capsules Overview pane

Red Hat Insights Overview window

Click the **Insights** tab in the left control pane to perform analytics and vulnerability scanning of the systems, as shown in Figure 3-114.

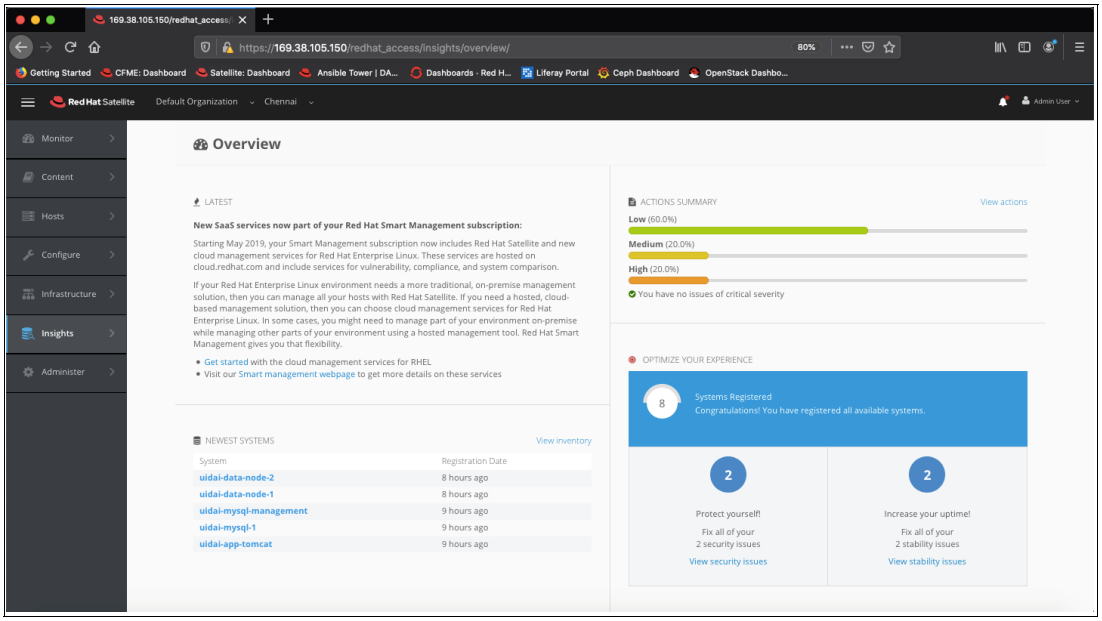


Figure 3-114 Red Hat Satellite: Insights Overview window

You can build the scanner plan, as shown in Figure 3-115.

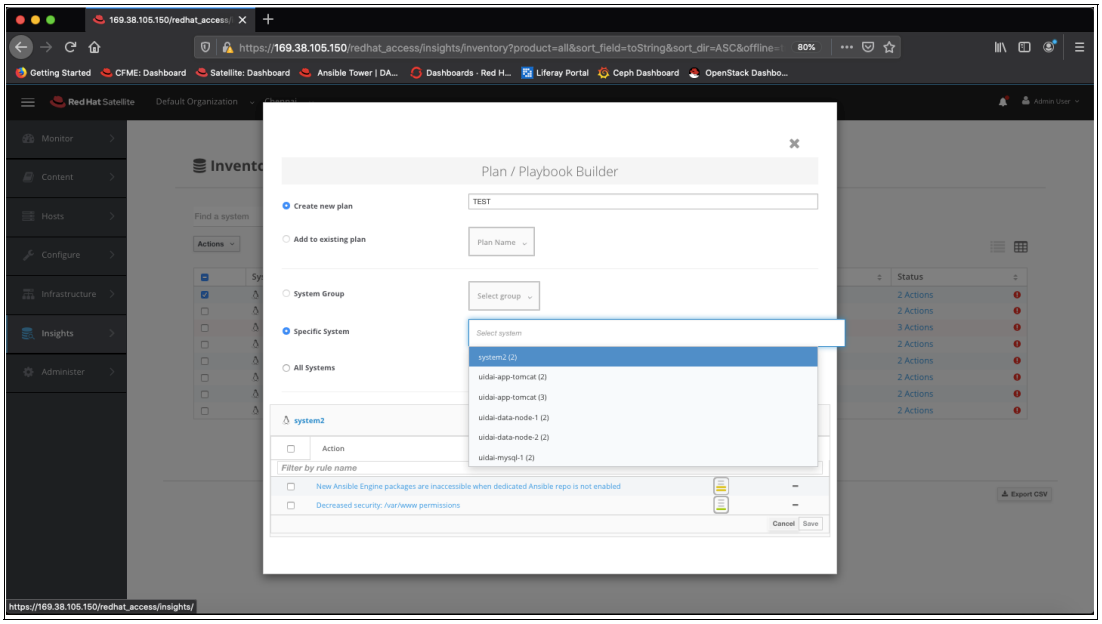


Figure 3-115 Red Hat Satellite: Plan and Playbook Builder window

Vulnerability scanning is described next.

3.6.8 Red Hat Satellite use-cases for deployed Red Hat Cloud

This section describes some use cases that use Red Hat Satellite.

Vulnerability scanning with OpenSCAP

The Insights client must be deployed on Red Hat Enterprise Linux guests to send data to Red Hat Satellite. Complete the following steps:

1. After logging in to the Red Hat Satellite server, click **Monitor** and select the **Dashboard** option to display a summary of vulnerabilities, as shown in Figure 3-116.

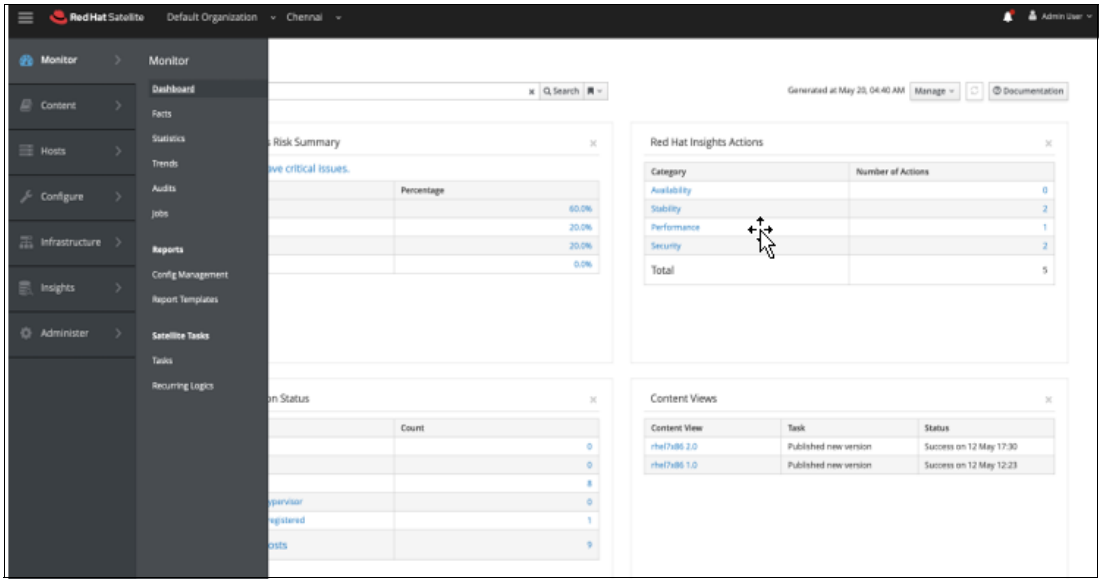


Figure 3-116 Red Hat Satellite: Monitor Dashboard window

Vulnerabilities are listed based on the Risk level and Category, as shown in Figure 3-117.

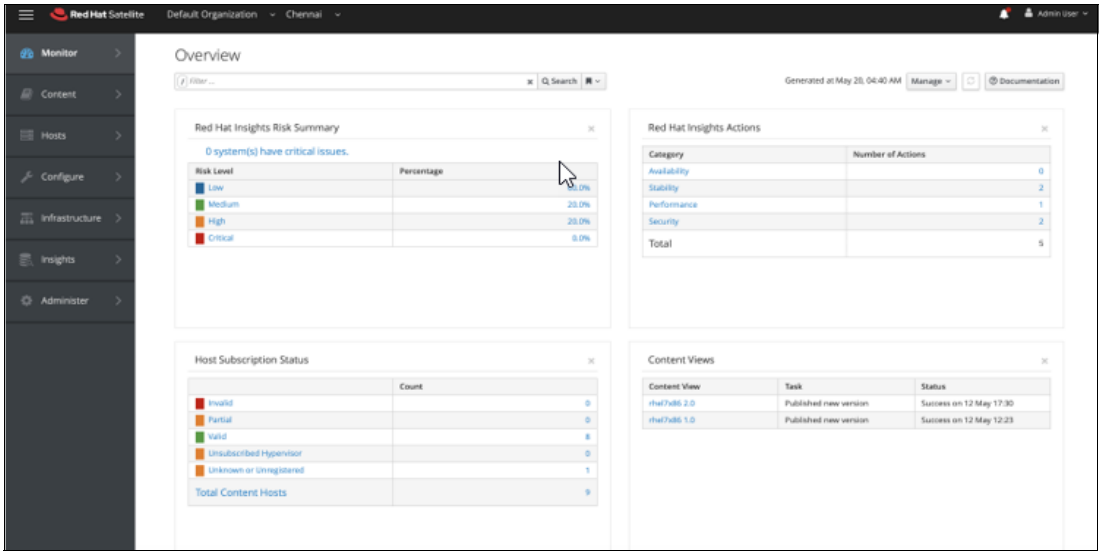


Figure 3-117 Red Hat Satellite: Monitor Overview

2. Select **Risk level** or **Category**. A list of vulnerabilities is shown. Then, select the vulnerabilities that you want to remediate, as shown in Figure 3-118.

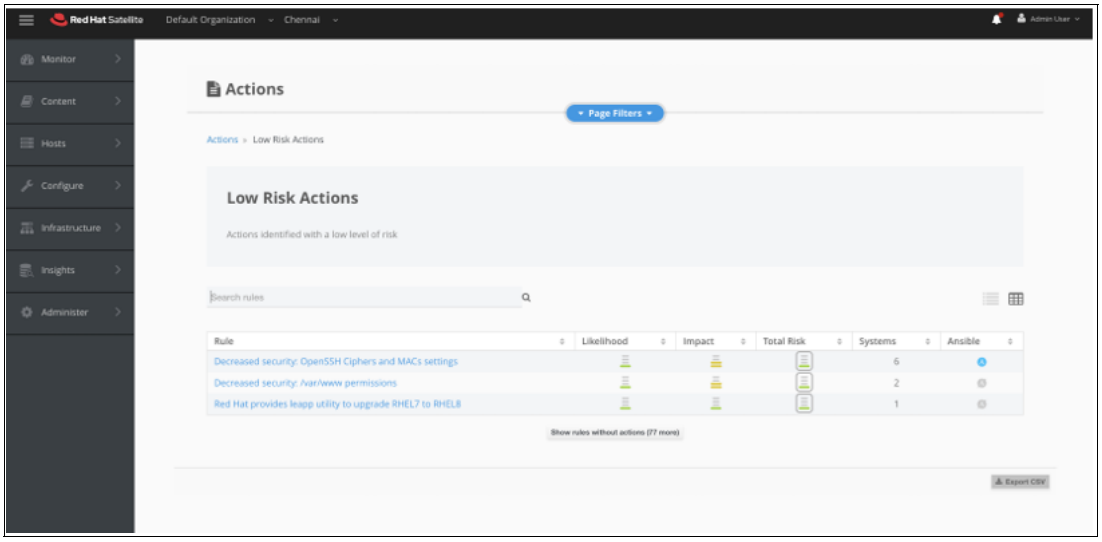


Figure 3-118 Red Hat Satellite: Low Risk Actions window

3. Select the hosts to remediate, as shown in Figure 3-119.

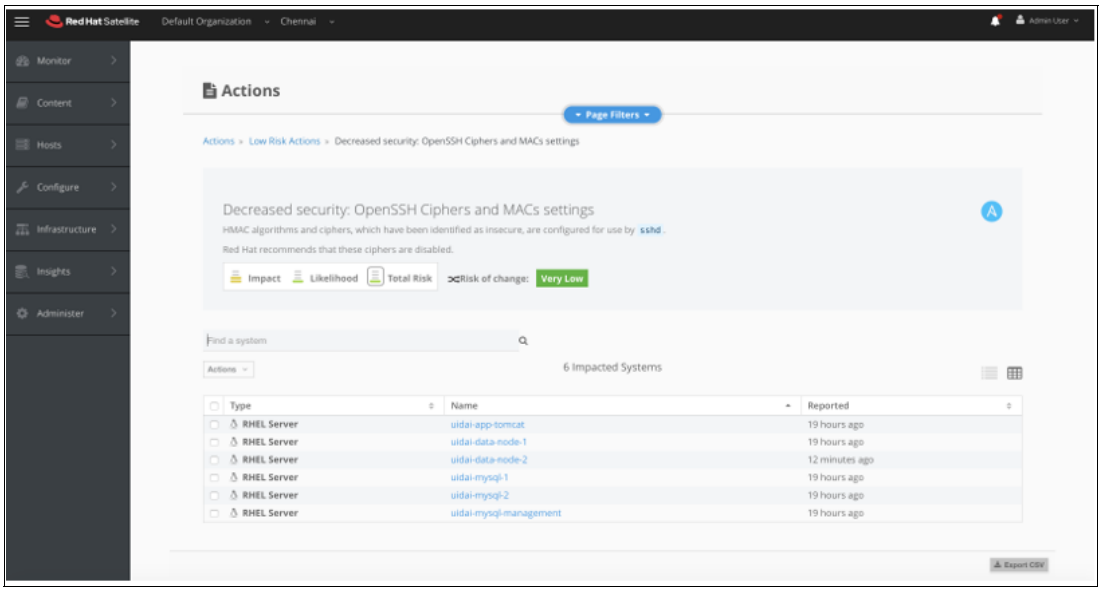


Figure 3-119 Red Hat Satellite: Actions window

4. Create a remediation plan by selecting **Action** → **Create a new Plan/Playbook**, as shown in Figure 3-120.

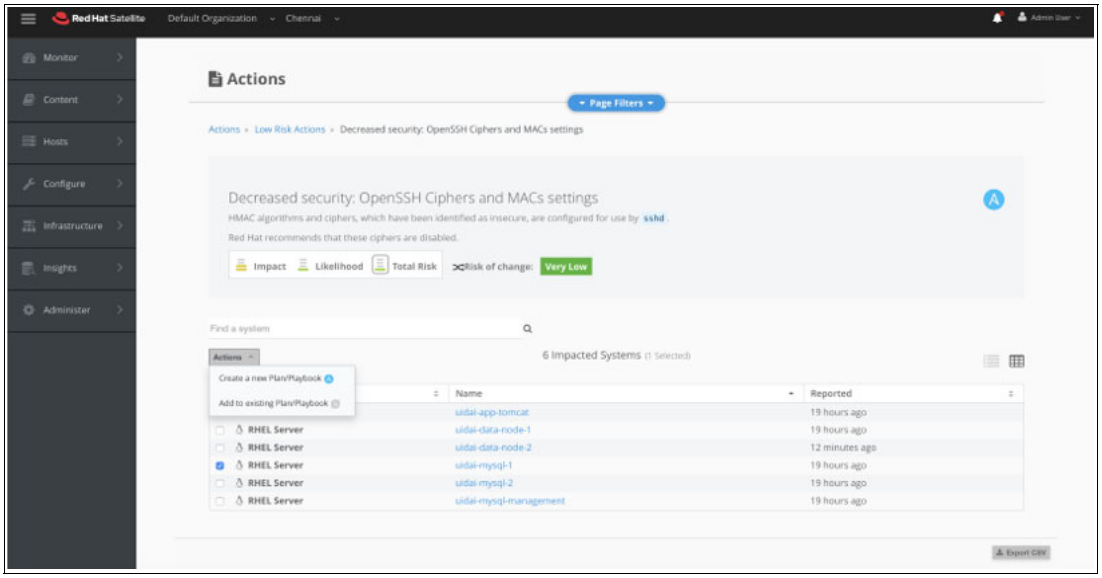


Figure 3-120 Red Hat Satellite Actions: Create window

5. Enter a name for the remediation plan and click **Save**, as shown in Example 3-121.

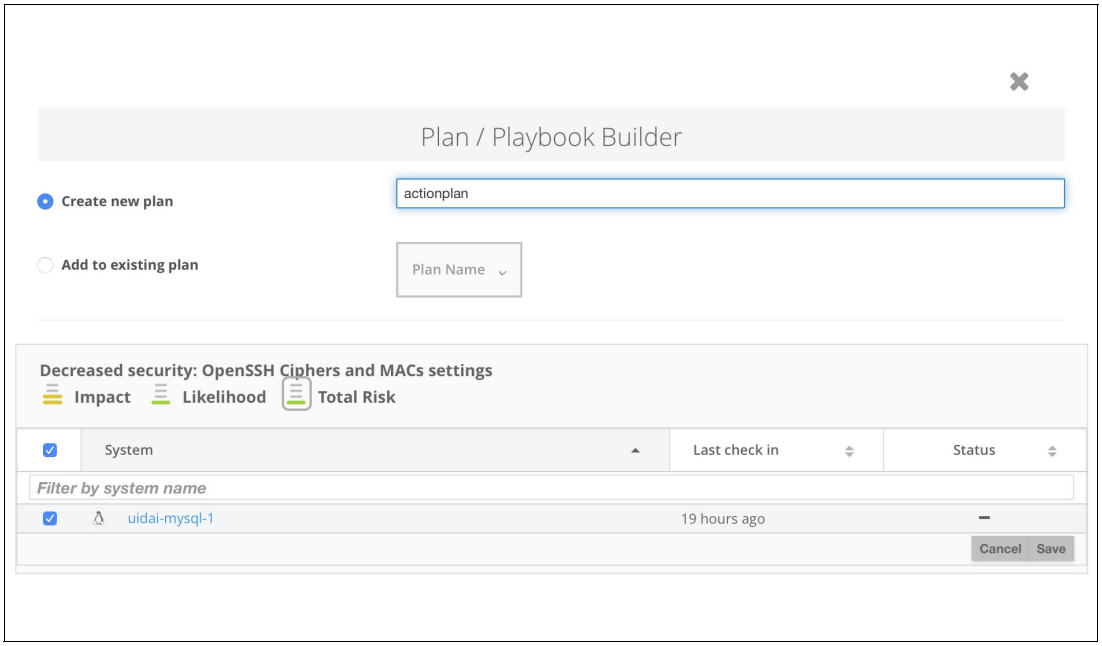


Figure 3-121 Plan / Playbook Builder window

6. Click **Remediation Plan** and select **Run Playbook**, which starts running the Red Hat Ansible playbook to remediate the vulnerabilities, as shown in Figure 3-122.

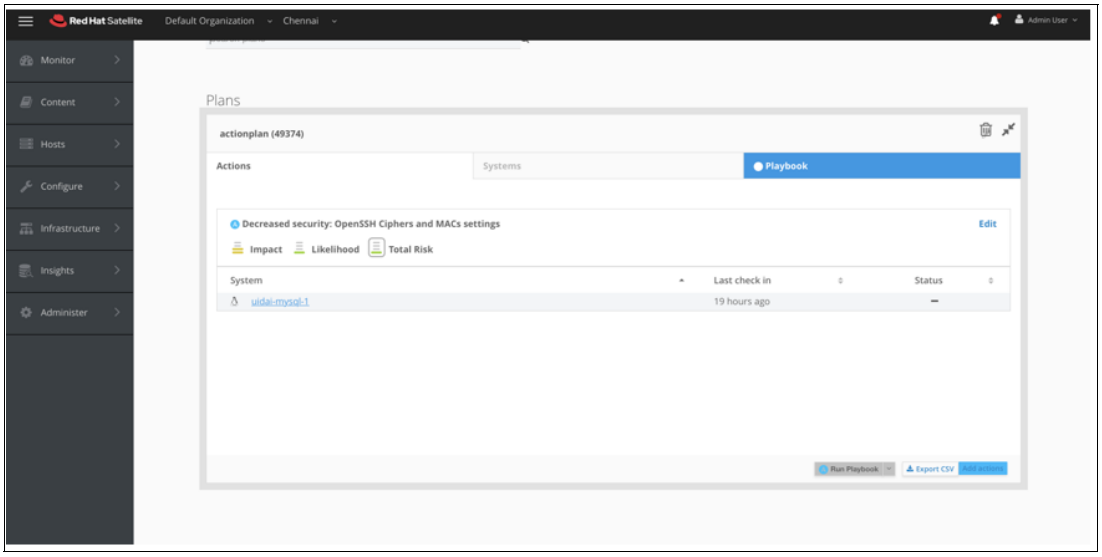


Figure 3-122 Red Hat Satellite: Plans window

Red Hat Systems Patch Management

Complete the following steps:

1. After your log into the Red Hat Satellite server, select the **Hosts** tab, which lists all the hosts that are managed by Red Hat Satellite server, as shown in Figure 3-123.

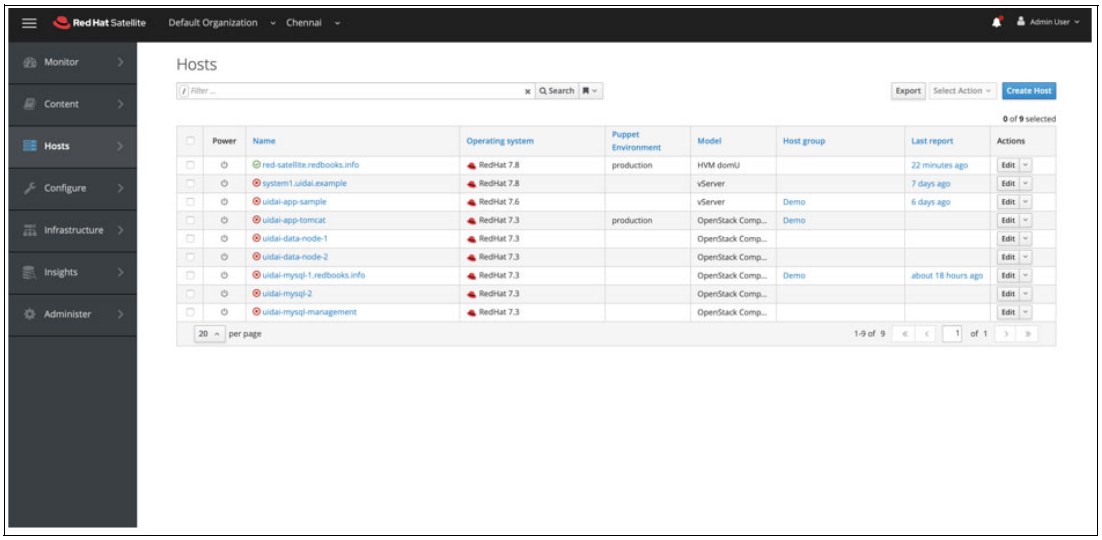


Figure 3-123 Red Hat Satellite: Hosts window

2. Click the host that you need to patch, as shown in Figure 3-124.

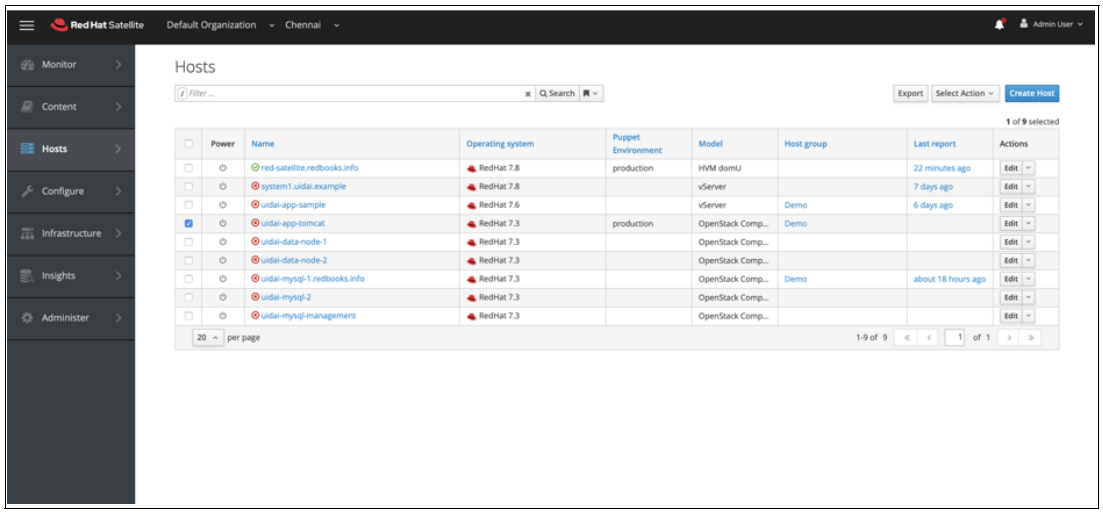


Figure 3-124 Red Hat Satellite: Selecting host

The details of the selected host are displayed, as shown in Figure 3-125.

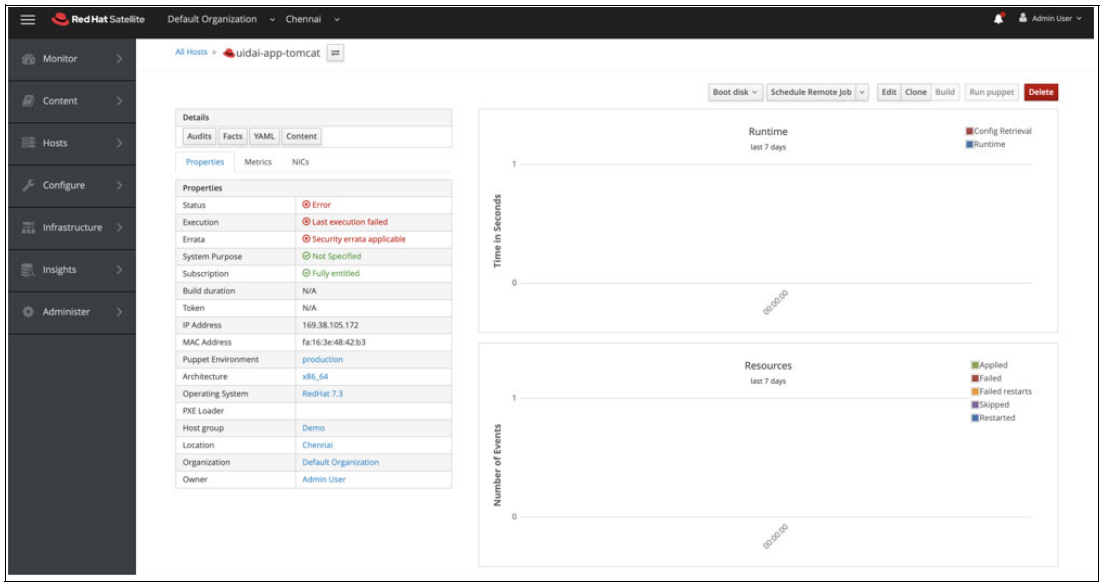


Figure 3-125 Red Hat Satellite: Host details

3. Click **Content**, which shows the installable Errata (patches), as shown in Figure 3-126. Three kinds of patches are available: Security, Bug Fix, and Enhancement.

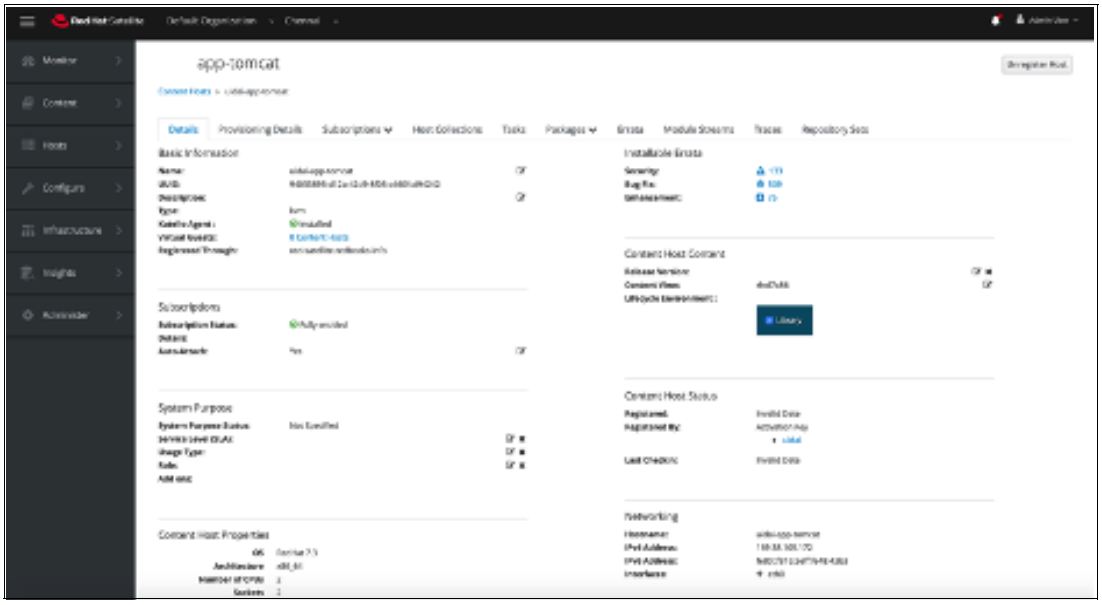


Figure 3-126 Red Hat Satellite: Content Hosts window

4. Select the suitable patch type. Figure 3-127 shows the selected security patch type, which then lists all of the security patches.

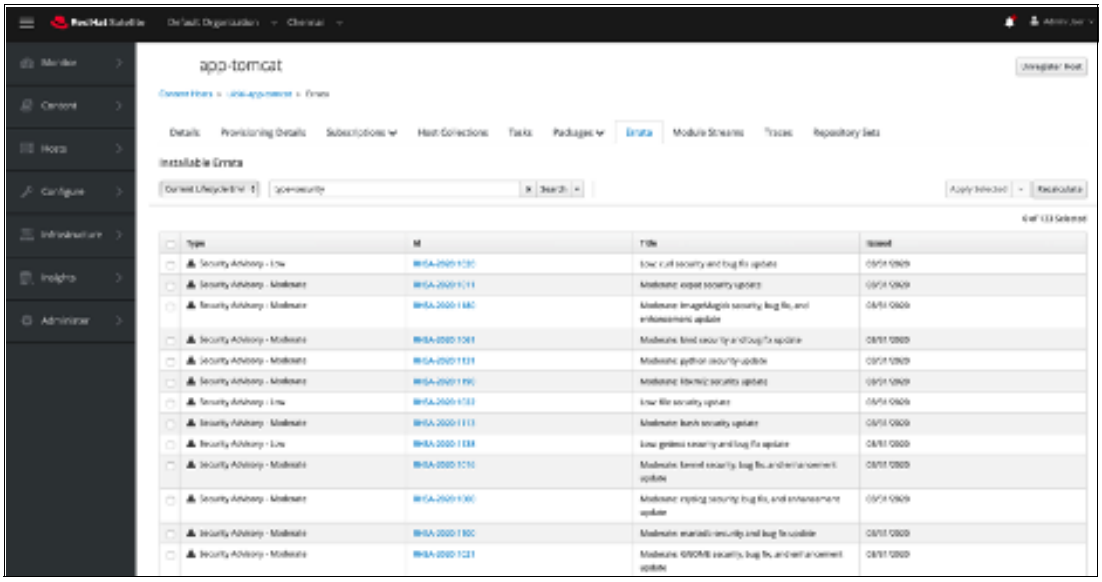


Figure 3-127 Red Hat Satellite: Content Hosts Installable Errata

5. Select the patch that must be applied and click **Apply Selected**, as shown in Figure 3-128.

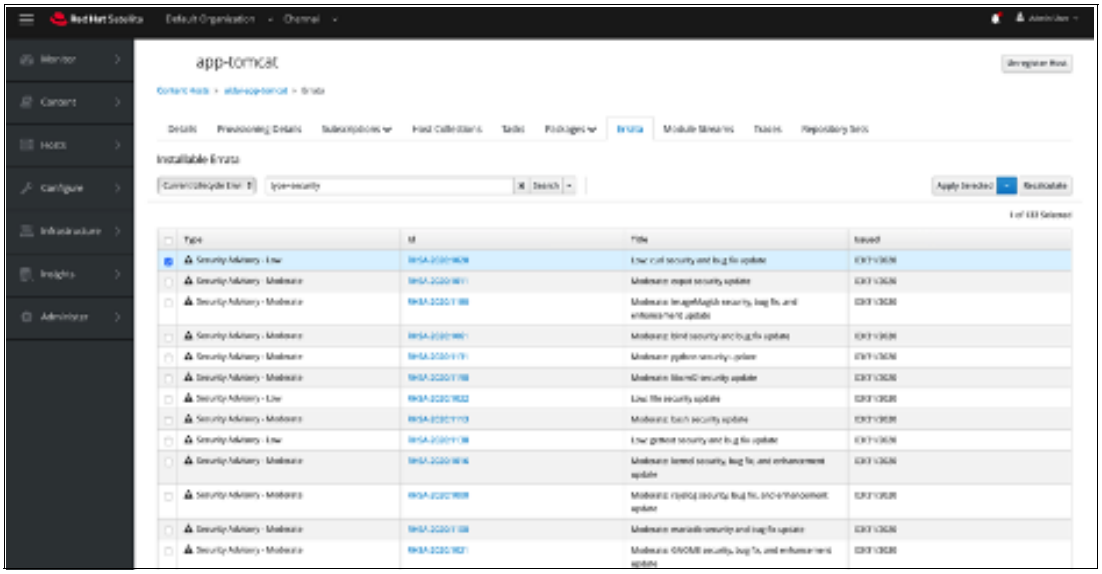


Figure 3-128 Red Hat Satellite: Selecting the patch

The security patch beings applying on the selected host, as shown in Figure 3-129.

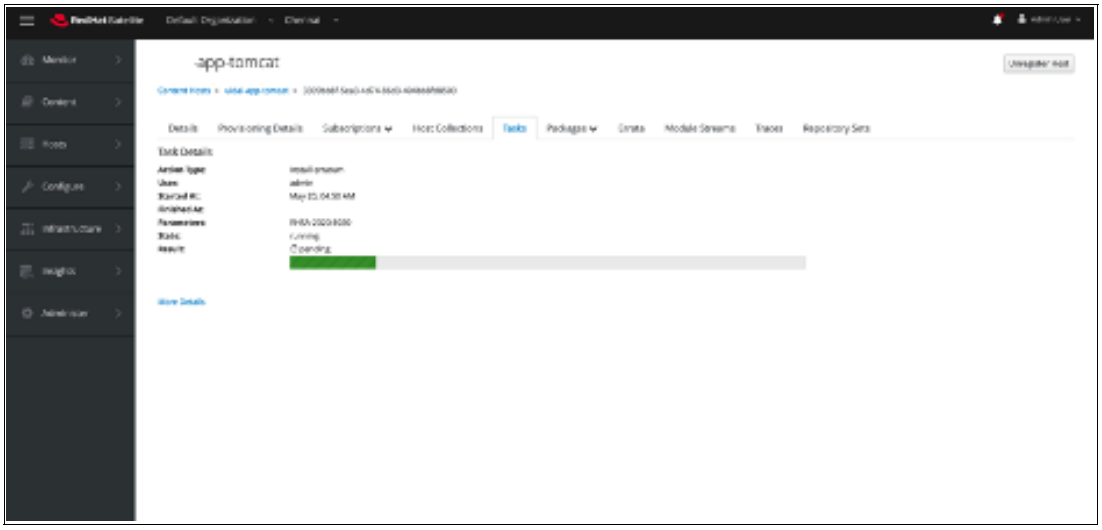


Figure 3-129 Red Hat Satellite: Applying patch window

3.7 Red Hat Ceph Storage

Data growth is a standard problem of the modern digital age. Being data driven to remain in business and increase profitability is not an option; rather, it is a necessity. In these complex times, tighter IT budgets, along with the need to shift the focus on business requirements, are putting added pressures on IT departments.

In recent times, suppliers brought many new technologies to the market that include, all-flash arrays, hyper converged infrastructure, and software-defined storage. As users seek to address specific workloads in their environments, they are looking to adopt many (if not all) of these new technologies in lieu of the traditional proprietary storage systems:

- Investment protection is enabled by modular design of software-defined systems and seamless nondisruptive process of adding and replacing individual nodes.
- Easy and quick deployments, upgrades, and refresh cycle of SDS save organizations time and money.
- Organizations can grow scale performance and capacity as required with ease based on requirements.
- Nondisruptive system or component upgrades or replacement allows businesses to function normally.
- Software investments can be preserved when refreshing hardware.

Red Hat's Ceph Storage is a unified Object Storage stack that supports block and Object Storage access.

3.7.1 Introduction

Software-defined storage is now seen as a critical need, as organizations struggle to manage unprecedented data growth while remaining agile and cost competitive. To manage petabytes of data at the speed and with the flexibility required by modern business environments, enterprises are increasingly turning to Red Hat Ceph Storage.

As a self-healing, self-managing platform with no single point of failure, Red Hat Ceph Storage significantly lowers the cost of storing enterprise data and helps companies manage exponential data growth in an automated fashion. Red Hat Ceph Storage is a robust, software-defined storage solution that offers the following benefits:

- Supports block, object, and file storage to serve as a single, efficient, unified storage platform.
- Provides an award-winning, web-scale object store for modern use cases.
- Decouples software from hardware to run cost-effectively on industry-standard servers and disks.
- Scales flexibly to support multi-petabyte deployments.
- Combines the most stable version of Ceph Storage from the open source community with a monitoring dashboard, easy-to-use deployment tools, and Red Hat support.

The power of Red Hat Ceph can transform your organization's IT infrastructure and your ability to manage vast amounts of data, especially for cloud computing platforms, such as Red Hat Enterprise Linux and Red Hat OpenStack. Red Hat Ceph delivers extraordinary scalability; that is, thousands of clients accessing petabytes to exabytes of data and beyond.

3.7.2 Red Hat Ceph system architecture

A Ceph storage cluster is built from many Ceph nodes for scalability, fault-tolerance, and performance. Each node is based on industry-standard hardware and uses intelligent Ceph daemons that communicate with each other to perform the following tasks:

- Store and retrieve data
- Replicate data
- Monitor and report on cluster health
- Redistribute data dynamically (remap and backfill)
- Ensure data integrity (scrubbing)
- Detecting and recovering from faults and failures

To the Ceph client interface that reads and writes data, a Ceph storage cluster appears as a simple pool where data is stored. However, the storage cluster performs many complex operations in a manner that is not apparent to the client interface.

Ceph clients and Ceph Object Storage daemons (Ceph OSD daemons, or OSDs) both use the CRUSH (controlled replication under scalable hashing) algorithm for storage and retrieval of objects. When a Ceph client reads or writes data (referred to as an I/O context), it connects to a logical storage pool in the Ceph cluster. Figure 3-130 shows the overall Ceph architecture, with concepts that are described next.

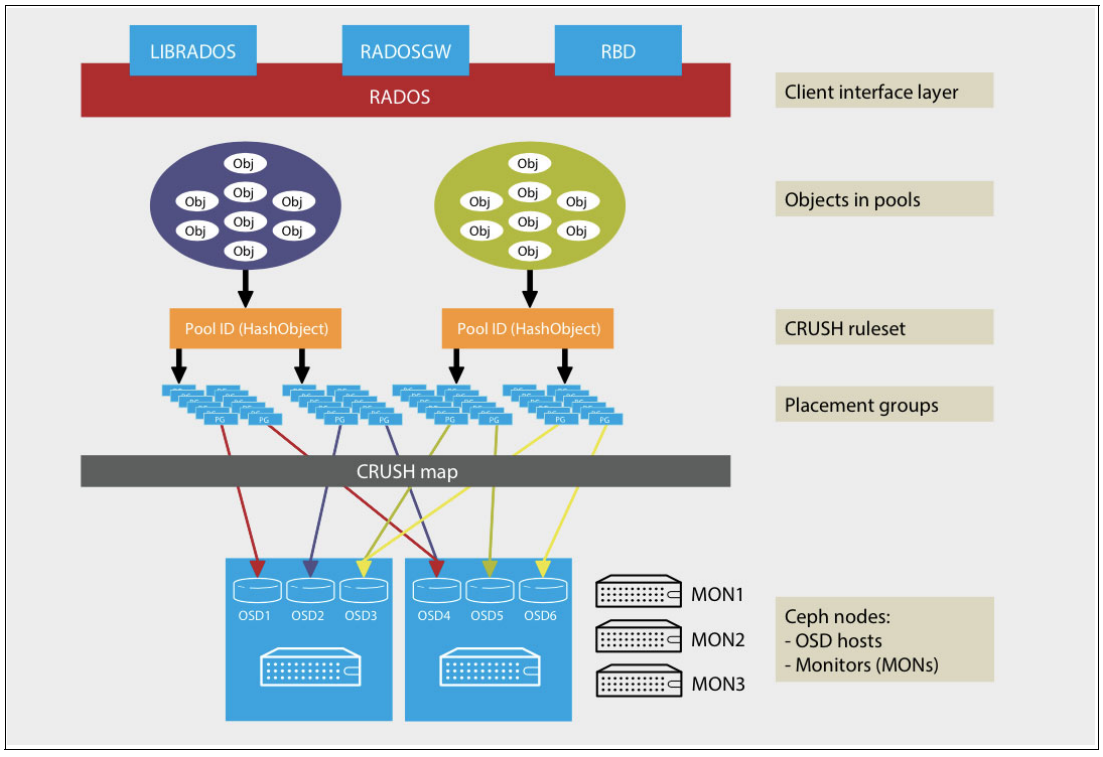


Figure 3-130 Red Hat Ceph logical architecture

Ceph architecture components

This section describes the architectural components for Ceph.

- CRUSH ruleset

  The CRUSH algorithm provides controlled, scalable, and de-clustered placement of replicated or erasure-coded data within Ceph and determines how to store and retrieve data by computing data storage locations. CRUSH empowers Ceph clients to communicate with OSDs directly, rather than through a centralized server or broker. By determining a method of storing and retrieving data by algorithm, Ceph avoids a single point of failure, performance bottleneck, and physical limit to scalability.

- Ceph monitors (MONs)

  Before Ceph clients can read or write data, they must contact a Ceph MON to obtain the current cluster map. A Ceph storage cluster can operate with a single monitor, but this introduces a single point of failure. For added reliability and fault tolerance, Ceph supports an odd number of monitors in a quorum (typically three or five for small to mid-sized clusters). Consensus among various monitor instances ensures consistent knowledge about the state of the cluster.

- Ceph OSD daemons

  In a Ceph cluster, Ceph OSDs store data and handle data replication, recovery, backfilling, and rebalancing. They also provide some cluster state information to Ceph monitors by checking other Ceph OSD daemons with a heartbeat mechanism. A Ceph storage cluster that is configured to keep three replicas of every object requires a minimum of three Ceph OSD daemons, two of which must be operational to successfully process write requests. Ceph OSD daemons roughly correspond to a file system on a physical hard disk drive (HDD) or flash storage. Multiple OSDs can exist on a physical OSD node.

- RADOS

  The Reliable Autonomic Distributed Object Store (RADOS) is the foundation of the Ceph storage cluster. A software-based reliable, autonomous, distributed object store consists of self-healing, self-managing, intelligent storage nodes and lightweight monitors

3.7.3 General Design Principles for Ceph Storage and Hardware Selection

Despite the flexibility of Ceph, no one cluster or pool configuration fits all applications or situations. Instead, successful configuration of a Ceph cluster requires carefully analyzing the general guidelines that are described next and choosing the correct hardware.

Identifying target workload IOPS profile

One of the key benefits of Ceph storage is the ability to support different types of workloads within the same cluster by using Ceph performance domains. Dramatically different hardware configurations can be associated with each performance domain, as listed in Table 3-10. Selecting suitably sized and optimized servers for these performance domains is an essential aspect of designing a Red Hat Ceph Storage cluster.

Table 3-10 Ceph storage configurations

Optimization criteria	Properties	Example use cases
IOPS-optimized storage cluster	► Lowest cost per IOPS. ► Highest IOPS per GB. ► 99th percentile latency consistency.	► Typically block storage. ► 3x replication for hard HDDs or 2x replication for solid state drives (SSDs). ► MySQL on Red Hat OpenStack clouds.

Optimization criteria	Properties	Example use cases
Throughput optimized	► Lowest cost per MBps (throughput). ► Highest MBps per TB ► Highest MBps per BTU. Highest MBps per Watt. ► 97th percentile latency consistency.	► Block or Object Storage. ► 3x replication. ► Active performance storage for video, audio, and images. ► Streaming media.
Cost and capacity optimized	► Lowest cost per TB. ► Lowest BTU per TB. ► Lowest Watts required per TB.	► Typically, Object Storage. ► Erasure coding common for maximizing usable capacity ► Object archive. ► Video, audio, and image object repositories.

The following hardware sizing guidelines are for each of the types of workloads that are listed in Table 3-10 on page 177:

- IOPS-optimized solutions: With the growing use of flash storage, organizations increasingly host IOPS-intensive workloads on Ceph storage clusters so that they can emulate high-performance public cloud solutions with private cloud storage. These workloads commonly involve structured data from MySQL, MariaDB, or PostgreSQL-based applications. NVMe SSDs with colocated Ceph write journals typically host OSDs. Typical servers include the following elements:
  - CPU: 10 cores per NVMe SSD, assuming a 2 GHz CPU.
  - RAM: 16 GB baseline, plus 5 GB per OSD.
  - Networking: 10 Gigabit Ethernet (GbE) per 12 OSDs (each for client- and cluster-facing networks).
  - OSD media: High-performance, high-endurance enterprise NVMe SSDs.
  - OSDs: Four per NVMe SSD.
  - Journal media: High-performance, high-endurance enterprise NVMe SSD, colocated with OSDs.
  - Controller: Native PCIe bus
- Throughput-optimized solutions: Throughput-optimized Ceph solutions often are centered around semi-structured or unstructured data. Large-block sequential I/O is typical. Storage media on OSD hosts is commonly HDDs with write journals on SSD-based volumes. A typical server includes the following elements:
  - CPU: 0.5 cores per HDD, assuming a 2 GHz CPU.
  - RAM: 16 GB baseline, plus 5 GB per OSD.
  - Networking: 10 GbE per 12 OSDs each for client- and cluster-facing networks.
  - OSD media: 7,200 RPM enterprise HDDs.
  - OSDs: One per HDD.
  - Journal media: High-endurance, high-performance enterprise serial-attached SCSI (SAS) or NVMe SSDs.
  - OSD-to-journal ratio: 4-5:1 for an SSD journal, or 12-18:1 for an NVMe journal.
  - Host bus adapter (HBA): Just a bunch of disks (JBOD).

- Cost and capacity-optimized solutions: These solutions typically focus on higher capacity, or longer archival scenarios. Data can be semi-structured or unstructured. Workloads include media archives, big data analytics archives, and machine image backups. Large-block sequential I/O is typical. For greater cost effectiveness, OSDs often are hosted on HDDs with Ceph write journals colocated on the HDDs. Solutions typically include the following elements:
  - CPU: 0.5 cores per HDD, assuming a 2 GHz CPU.
  - RAM: 16 GB baseline, plus 5 GB per OSD.
  - Networking: 10 GbE per 12 OSDs (each for client- and cluster-facing networks).
  - OSD media: 7,200 RPM enterprise HDDs.
  - OSDs: One per HDD.
  - Journal media: Colocated on the HDD.
  - HBA: JBOD.

Choosing the correct storage access method and data protection technology

Red Hat Ceph Storage provides the following storage access methods to clients, as shown in Figure 3-131:

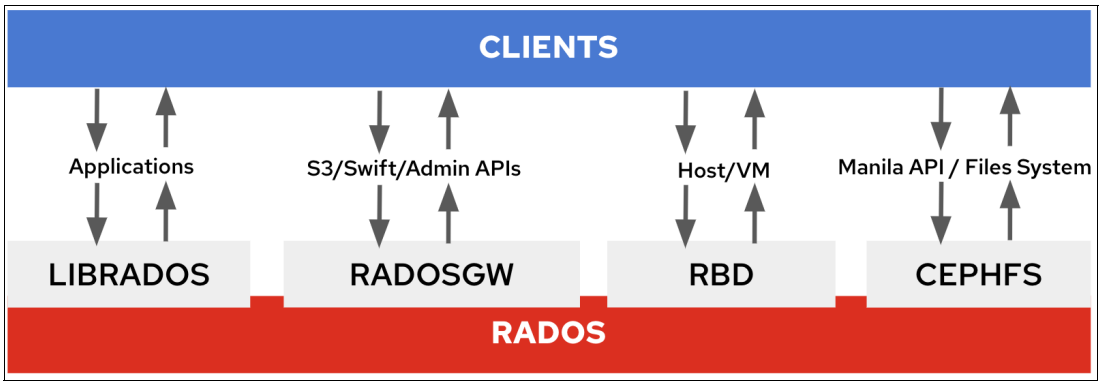


Figure 3-131 Red Hat Ceph logical architecture

- Object Storage:
  - LIBRADOS: A library that allows apps to directly access RADOS (C, C++, Java, Python, and Ruby).
  - Rados GateWay RGW): A Web Services Gateway for Object Storage, which is compatible with S3 and Swift. Support for authentication by using Active Directory, LDAP, and Red Hat OpenStack Keystone V3. Greater compatibility with the Amazon S3 and Red Hat OpenStack Swift Object Storage APIs.
- Block storage

  RBD is a reliable, fully distributed block device with cloud platform integration. RBD offers a Ceph block storage device that mounts like a physical storage drive for use by physical and virtual systems (with a Linux kernel driver, KVM/QEMU storage backend, or user-space libraries).
- File Storage:
  - CEPHFS: A distributed file system with POSIX semantics and scale-out metadata.

- LIBRADOS: A library allowing apps to directly access RADOS (C, C++, Java, Python, and Ruby) storage access method and data protection method (see 3.7.4, “Red Hat Ceph design guidelines” on page 181) are interrelated. For example, Ceph block storage is supported only on replicated pools, although Ceph Object Storage is supported on erasure-coded or replicated pools.

- Data Protection Algorithms

  As a design decision, choosing the data protection method can affect the solution’s total cost of ownership (TCO) more than any other factor. This issue occurs is because the chosen data protection method strongly affects the amount of raw storage capacity that must be purchased to yield the wanted amount of usable storage capacity. Applications feature diverse needs for performance and availability. As a result, Ceph provides data protection at the storage pool level:

  - Replicated storage pools: Replication makes full copies of stored objects and is ideal for quick recovery. In a replicated storage pool, Ceph configuration defaults to a replication factor of three, involving a primary OSD and two secondary OSDs. If two of the three OSDs in a placement group become unavailable, data can be read, but write operations are suspended until at least two OSDs are operational.
  - Erasure-coded storage pools: Erasure coding provides a single copy of data plus parity, and it is useful for archive storage and cost-effective durability and availability. With erasure coding, storage pool objects are divided into chunks by using the `n=k+m` notation, where `k` is the number data chunks that are created, `m` is the number of coding chunks that are created to provide data protection, and `n` is the total number of chunks that are placed by CRUSH after the erasure coding process.

Figure 3-132 shows the relationships among Ceph storage pools, PGs, and OSDs for replicated and erasure-coded pools.

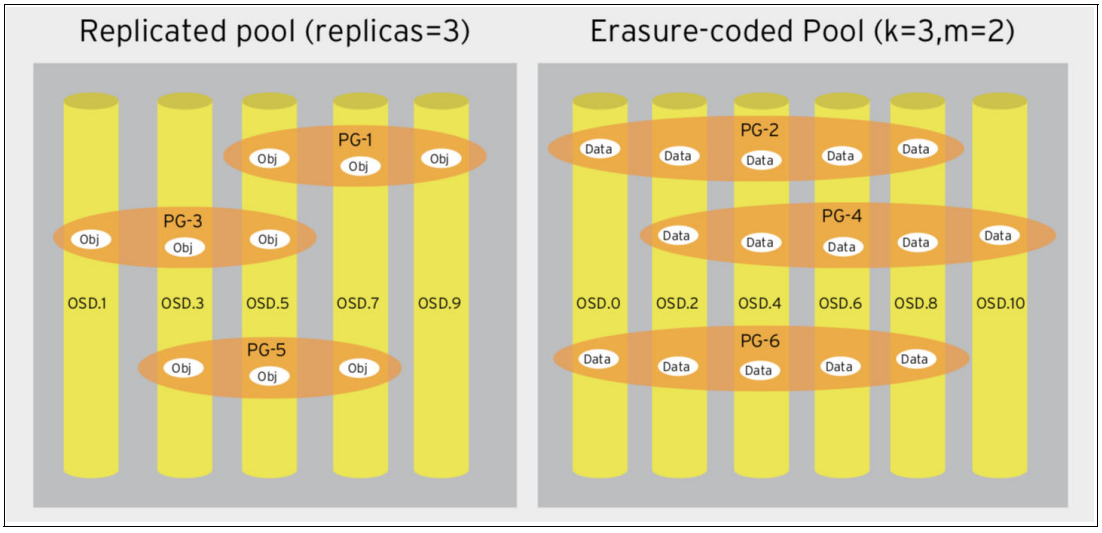


Figure 3-132 Ceph storage pools

Ceph block storage often is configured with 3x replicated pools and is not supported directly on erasure-coded pools. Ceph Object Storage is supported on replicated or erasure-coded pools. Depending on the performance needs and read/write mix of an Object Storage workload, an erasure-coded pool can provide a cost-effective solution and meet performance requirements.

3.7.4 Red Hat Ceph design guidelines

Red Hat Ceph Storage cluster is a distributed data object store that provides excellent performance, reliability, and scalability. Distributed object stores are the future of storage because they accommodate unstructured data, and clients can use modern object interfaces and traditional interfaces simultaneously (see Figure 3-133); for example:

- APIs in many languages (C/C++, Java, Python)
- RESTful interfaces (S3/Swift)
- Block device interface
- File system interface

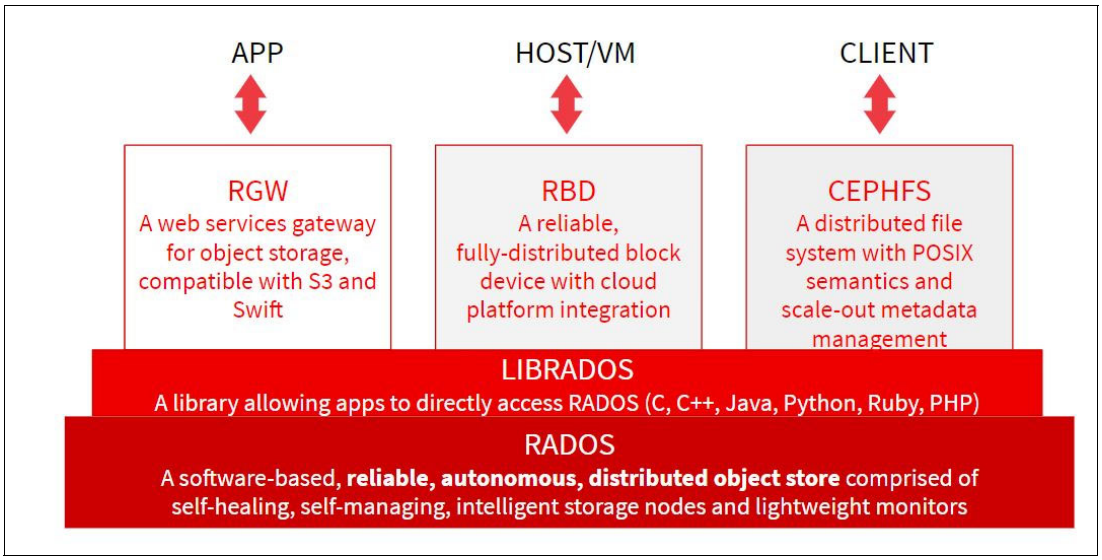


Figure 3-133 Ceph Storage Access Gateways

The power of Red Hat Ceph Storage cluster can transform your organization's IT infrastructure and your ability to manage vast amounts of data, especially for cloud computing platforms, such as Red Hat OpenStack. Red Hat Ceph Storage cluster delivers extraordinary scalability-thousands of clients accessing petabytes to exabytes of data and beyond.

At the heart of every Ceph deployment is the Red Hat Ceph Storage cluster. It consists of the following types of daemons (see Figure 3-134 on page 182):

- Ceph OSD daemon: Ceph OSDs store data on behalf of Ceph clients. Also, Ceph OSDs use the CPU, memory, and networking of Ceph nodes to perform data replication, erasure coding, rebalancing, recovery, monitoring, and reporting functions.
- Ceph Monitor: A Ceph Monitor maintains a master copy of the Red Hat Ceph Storage cluster map with the current state of the Red Hat Ceph Storage cluster. Monitors require high consistency and use Paxos to ensure agreement about the state of the Red Hat Ceph Storage cluster.
- Ceph Manager: The Ceph Manager maintains detailed information about placement groups, process metadata, and host metadata in lieu of the Ceph Monitor-significantly improving performance at scale. The Ceph Manager handles running many of the read-only Ceph CLI queries, such as placement group statistics. The Ceph Manager also provides the RESTful monitoring APIs.

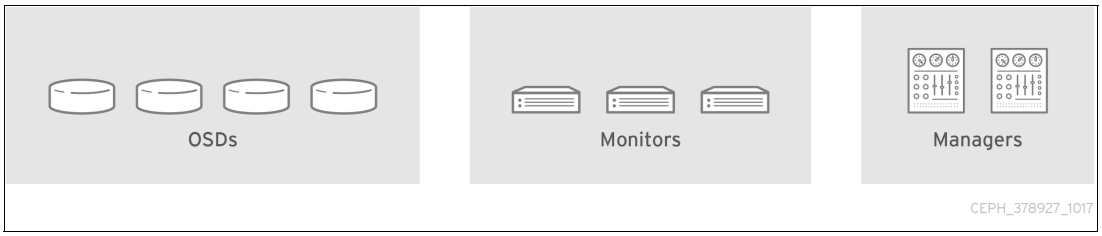


Figure 3-134 Ceph Storage logical components

Ceph client interfaces read data from and write data to the Red Hat Ceph Storage cluster. Clients need the following data to communicate with the Red Hat Ceph Storage cluster:

- Ceph configuration file, or the cluster name (often `ceph`) and the monitor address
- Pool name
- User name and the path to the secret key

Ceph clients maintain object IDs and the pool names where they store the objects. However, they do not need to maintain an object-to-OSD index or communicate with a centralized object index to look up object locations.

To store and retrieve data, Ceph clients access a Ceph Monitor and retrieve the latest copy of the Red Hat Ceph Storage cluster map. Then, Ceph clients provide an object name and pool name to librados, which computes an object's placement group and the primary OSD for storing and retrieving data by using the Controlled Replication Under Scalable Hashing (CRUSH) algorithm. The Ceph client connects to the primary OSD where it can perform read and write operations. No intermediary server, broker, or bus exists between the client and the OSD.

When an OSD stores data, it receives data from a Ceph client, whether the client is a Ceph Block Device, Ceph Object Gateway, Ceph file system, or another interface, and it stores the data as an object with the following characteristics:

- Collocated containerized storage: Isolate resources for peak load and recovery situations by collocating one of the scale out daemons in each storage node, which allows for the creation of a cluster that is comprised entirely of storage nodes. Because this simplified strategy reduces demand for resources and hardware costs, colocation is an ideal scenario for TCO workloads.
- Expanded workload reach: Migrate traditional workloads from a file-based application to an object store by using NFS v3 or v4 protocols. Platforms without a native driver (VMware and Windows) can access Ceph Storage by way of iSCSI, which delivers block-level access by way of an industry storage networking standard. Complement Ceph's block and Object Storage capabilities with CephFS, which provides stability, protection from data loss, HA, and full scale-out capabilities, such as multiple active MDS configurations.
- Day one management with Red Hat Ansible: Go from download to available storage quickly. Simplified deployment enables you to set up OSD, MON, and Ceph Manager nodes with an Red Hat Ansible playbook on Red Hat Enterprise Linux so that your cluster is immediately ready for your workloads. MDS, RGW, and the Ceph client role also can be optionally deployed with Red Hat Ansible.
- State-of-the-art monitoring and management: Gain insight into high-level cluster health, drill-down to specific nodes to identify performance gaps, manage volumes, or start cluster upgrades with the dashboard management interface.

A Red Hat Ceph Storage cluster can have many Ceph nodes for limitless scalability, HA, and performance. Each node uses non-proprietary hardware and intelligent Ceph daemons that communicate with each other to complete the following tasks:

- Write and read data.
- Ensure durability by replicating or erasure coding data.
- Monitor and report on cluster health (also referred to as *heartbeating*).
- Redistribute data dynamically (also referred to as *backfilling*).
- Ensure data integrity.
- Recover from failures: To the Ceph client interface that reads and writes data, a Red Hat Ceph Storage cluster resembles a simple pool where it stores data. However, librados and the storage cluster perform many complex operations in a manner that is not apparent to the client interface. Ceph clients and Ceph OSDs use the CRUSH algorithm.

Next, we describe how CRUSH enables Ceph to perform these operations seamlessly.

The Ceph storage cluster stores data objects in logical partitions called *pools*. Ceph administrators can create pools for particular types of data, such as for block devices, object gateways, or just to separate one group of users from another.

A Ceph pool is a logical partition for storing object data. It also plays a critical role in how the Ceph storage cluster distributes and stores data. However, these complex operations are not apparent to the Ceph client.

Ceph pools define the following factors:

- Pool type

  In early versions of Ceph, a pool maintained multiple deep copies of an object. Today, Ceph can maintain multiple copies of an object, or it can use erasure coding to ensure durability. The data durability method is pool-wide and does not change after creating the pool. The pool type defines the data durability method when creating the pool. Pool types are not apparent to the client.

- Placement groups

  In an exabyte scale storage cluster, a Ceph pool might store millions of data objects or more. Ceph must handle many types of operations, including data durability via replicas or erasure code chunks, data integrity by scrubbing or CRC checks, replication, rebalancing and recovery. Therefore, managing data on a per-object basis presents a scalability and performance bottleneck. Ceph addresses this bottleneck by sharding a pool into placement groups. The CRUSH algorithm computes the placement group for storing an object and computes the Acting Set of OSDs for the placement group. CRUSH puts each object into a placement group. Then, CRUSH stores each placement group in a set of OSDs. System administrators set the placement group count when creating or modifying a pool.

- CRUSH ruleset

  CRUSH plays another important role because it can detect failure domains and performance domains. CRUSH can identify OSDs by storage media type and organize OSDs hierarchically into nodes, racks, and rows. CRUSH enables Ceph OSDs to store object copies across failure domains. For example, copies of an object can be stored in different server rooms, aisles, racks, and nodes. If a large part of a cluster fails, such as a rack, the cluster can still operate in a degraded state until the cluster recovers.

Also, CRUSH enables clients to write data to particular types of hardware, such as SSDs, hard disk drives with SSD journals, or hard disk drives with journals on the same drive as the data. The CRUSH ruleset determines failure domains and performance domains for the pool. Administrators set the CRUSH ruleset when creating a pool:

- Durability

  In exabyte scale storage clusters, hardware failure is an expectation and not an exception. When data objects are used to represent larger-grained storage interfaces, such as a block device, losing one or more data objects for that larger-grained interface can compromise the integrity of the larger-grained storage entity-potentially rendering it useless.

  Therefore, data loss is intolerable. Ceph provides high data durability in two ways:

  – Replica pools store multiple deep copies of an object by using the CRUSH failure domain to physically separate one data object copy from another. That is, copies are distributed to separate physical hardware. This ability increases durability during hardware failures.

  – Erasure coded pools store each object as K+M chunks, where K represents data chunks and M represents coding chunks. The sum represents the number of OSDs that are used to store the object and the M value represents the number of OSDs that can fail and still restore data if the M number of OSDs fail.

  From the client perspective, Ceph is elegant and simple. The client reads from and writes to pools. However, pools play an important role in data durability, performance and HA.

3.7.5 Red Hat Ceph deployment architecture overview

An overview of the design of the Ceph cluster deployment is shown in Figure 3-135.

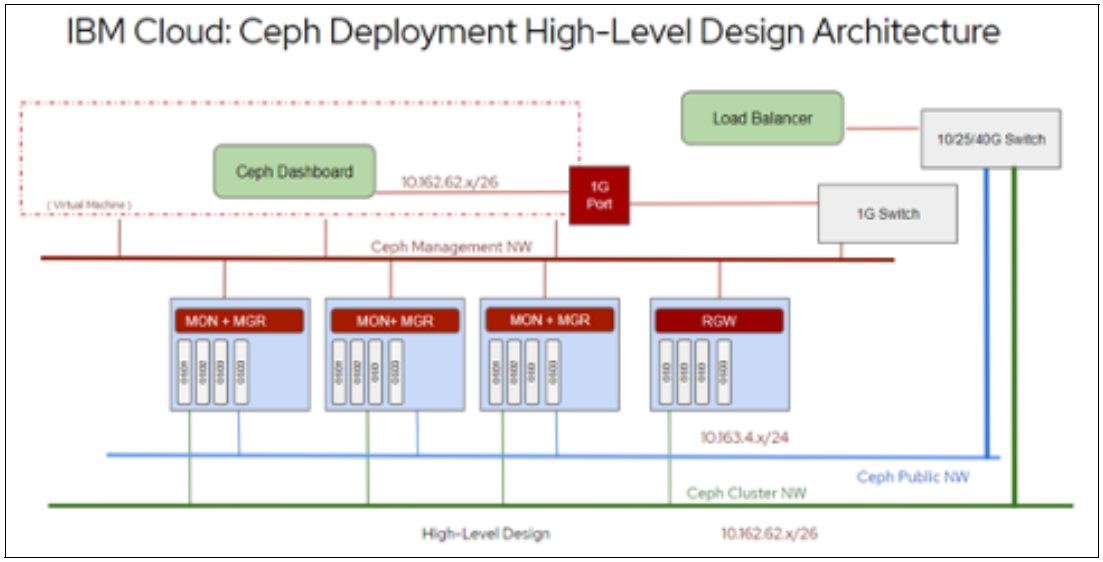


Figure 3-135 High level deployment diagram for storage deployment

BlueStore

BlueStore is the next generation storage implementation for Ceph. Because the market for storage devices now includes solid state drives (SSDs) and non-volatile memory over PCI Express (NVMe), their use in Ceph reveals some of the limitations of the FileStore storage implementation.

Although FileStore has many improvements to facilitate SSD and NVMe storage, other limitations remain, including increasing placement groups (which are computationally expensive) and the double write penalty. Consider the following points regarding BlueStore:

- Whereas FileStore interacts with a file system on a block device, BlueStore eliminates that layer of indirection and directly uses a raw block device for Object Storage.
- BlueStore uses the lightweight BlueFS file system on a small partition for its k/v databases.
- BlueStore eliminates the paradigm of a directory that represented a placement group, a file that represents an object, and file XATTRs that represent metadata.
- BlueStore eliminates the double write penalty of FileStore; therefore, write operations are nearly twice as fast with BlueStore under most workloads.

BlueStore stores data as the following formats:

- Object data

  In BlueStore, Ceph stores objects as blocks directly on a raw block device. The portion of the raw block device that stores object data does *not* contain a file system. The omission of the file system eliminates a layer of indirection, which improves performance. However, much of the BlueStore performance improvement comes from the block database and write-ahead log.

- Block database

  In BlueStore, the block database handles the object semantics to ensure consistency. An object's unique identifier is a key in the block database. The values in the block database consist of a series of block addresses that refer to the stored object data, the object's placement group, and object metadata.

  The block database can be on a BlueFS partition on the same raw block device that stores the object data, or on a separate block device, often when the primary block device is an HDD and an SSD or NVMe improves performance.

  The block database provides several improvements over FileStore; namely, the key/value semantics of BlueStore are not limited by the limitations of file system XATTRs. BlueStore can assign objects to other placement groups quickly within the block database without the overhead of moving files from one directory to another, as is the case in FileStore.

  BlueStore also introduces new features. The block database can store the checksum of the stored object data and its metadata, which allows full data checksum operations for each read. This process is more efficient than periodic scrubbing to detect bit rot. BlueStore can compress an object and the block database can store the algorithm that is used to compress an object-ensuring that read operations select the appropriate algorithm for decompression.

- Write-ahead Log

  In BlueStore, the write-ahead log ensures Atomicity, similar to the journaling functions of FileStore. As with FileStore, BlueStore logs all aspects of each transaction. However, the BlueStore write-ahead log or WAL can perform this function simultaneously, which eliminates the double write penalty of FileStore. Therefore, BlueStore is nearly twice as fast as FileStore on write operations for most workloads.

BlueStore also can deploy the WAL on the same device for storing object data, or it can deploy the WAL on another device, often when the primary block device is an HDD and an SSD or NVMe improves performance.

- Recommendations

  It is only helpful to store a block database or a write-ahead log on a separate block device if the separate device is faster than the primary storage device. For example, SSD and NVMe devices are faster than HDDs. Placing the block database and the WAL on separate devices can also have performance benefits due to differences in their workloads.

When an administrator adds a Ceph OSD to a Ceph storage cluster, Ceph updates the cluster map. This change to the cluster map also changes object placement because the modified cluster map changes an input for the CRUSH calculations. CRUSH places data evenly, but pseudo randomly. Therefore, only a small amount of data moves when an administrator adds an OSD. The amount of data is often the number of new OSDs divided by the total amount of data in the cluster. For example, in a cluster with 50 OSDs, 1/50th or 2% of the data might move when adding an OSD.

Figure 3-136 shows the rebalancing process where some, but not all, of the PGs migrate from OSDs (OSD 1 and 2 in Figure 3-136) to the new OSD (OSD 3 in Figure 3-136). Even when rebalancing, CRUSH is stable. Many of the placement groups remain in their original configuration, and each OSD gets some added capacity; therefore, no load spikes occur on the new OSD after the cluster rebalances.

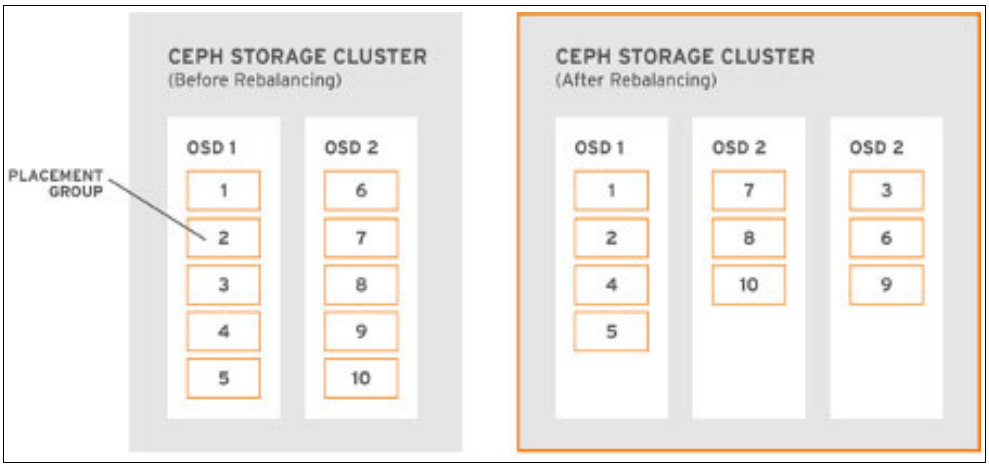


Figure 3-136 Ceph Storage rebalancing

3.7.6 Prerequisites and server environment

This section covers the deployment configuration for the Ceph nodes:

- Requirements:
  - 10 TB of usable space (5 TB for RBD and 5 TB for object)
  - Approximately 39 TB (15 TB raw and 15 TB object) of raw capacity
- Ceph cluster hardware configuration, as listed in Table 3-11.

Table 3-11 Ceph cluster hardware details

Physical server count	4
CPU cores	20 Cores (2.5 GHz+) or above

Physical server count	4
RAM memory	192 GB
Network interfaces	1G X1 10G X 4 (2 interface connecting towards Application/ Storage Data NW of Red Hat OpenStack Platform) and (2 interface for Ceph internal Replication)
Storage OSD nodes	5 X 1.6 TB SSD (Block Pool) 5 X 1.6 TB SSD (Object Storage)
RockDB and WAL storage	12 X 1.6 TB SSD

- Two VMs, as listed in Table 3-12 (one for the installer and the other for Ceph Dashboard).

Table 3-12 Usage configuration

Usage	Count	CPU cores/vCPUs	RAM	Internal storage	Other storage
Ceph Dashboard	1	8	32 GB	250 GB	300 GB
Ceph Ansible	1	8	32 GB	250 GB	N/A

- Ceph cluster network details, as listed in Table 3-13.

Note: For security reasons, we masked the IP addresses, and suggest that you configure the Ceph cluster per your network configurations.

Table 3-13 Cluster network details

Hostname	ILO IP address	Production /application IP (10G) (bond1)	Cluster replication IP (10G) (int2)	Public IP (bond1)
ceph-dashboard.redbooks.info	N/A	N/A	10.xx.xx.xx/26	169.xx.xx.xx
red-ceph1.redbooks.info	10.xx.xx.xx	10.xx.xx.xx/24	10.xx.xx.xx/26	169.xx.xx.xx
red-ceph2.redbooks.info	10.xx.xx.xx	10.xx.xx.xx/24	10.xx.xx.xx/26	169.xx.xx.xx
red-ceph4.redbooks.info	10.xx.xx.xx	10.xx.xx.xx/24	10.xx.xx.xx/26	169.xx.xx.xx
red-ceph3.redbooks.info	10.xx.xx.xx	10.xx.xx.xx/24	10.xx.xx.xx/26	169.xx.xx.xx

We also include a collocated containerized environment under which two services can be configured on one node. Table 3-14 lists the configuration with Bonding mode details.

Table 3-14 Node bonding configurations modes

Hostname	Setup	int0	Bond1 mode	int2
red-ceph1.redbooks.info	MON+OSD	LACP	LACP	LACP
red-ceph2.redbooks.info	MON+OSD	LACP	LACP	LACP
red-ceph4.redbooks.info	MON+OSD	LACP	LACP	LACP
red-ceph3.redbooks.info	OSD	LACP	LACP	LACP

- NTP and DNS (Table 3-15) details

  DNS was configured on dashboard host locally.

  Note: Because we have a single NTP server and it is a single point of failure (SPOF), it is advisable to *not* use a single NTP server. For more information, see this web page.

Table 3-15 DNS servers

DNS	DNS
10.0.80.11	10.0.80.12

- Versions and redundancy, as listed in Table 3-16.

Table 3-16 Versions configured in the cluster

Ceph version	Red Hat Enterprise Linux operating system version	Ceph failover domain
3.3	7.7	Host

Note: Ceph Cluster was deployed with Default Failover Domain as host because the minimum number of racks are not provided or available for satisfying the Ceph cluster rack level failover domain requirement.

3.7.7 Installing Red Hat Ceph cluster

Complete the following steps to install Red Hat Ceph cluster:

1. Registering nodes and attaching subscriptions.

   See product details by logging in to RHSM on RHN. For more information about logging in, see Appendix A, “Red Hat subscription activation process” on page 401.

   Example 3-41 shows how to register the host and attaching the subscription. Check that proxy is enabled and the host can access the internet.

Example 3-41 Register host and attach subscription

```
# subscription-manager register --username afkhan19 --password XXXXXXX
Registering to: subscription.rhsm.redhat.com:443/subscription
The system has been registered with ID: aa3d7f76-6adf-4208-9172-376da1776cc5
The registered system name is: ceph-dashboard.redbooks.in
#
# subscription-manager attach --pool=8a85f98c63842fef0163986288c45cd9
Successfully attached a subscription for: Red Hat Ceph Storage, Premium (Up to
512TB on a maximum of 25 Physical Nodes)
```

2. Enable repositories:

   a. Disable the default software repositories. Then, enable the Red Hat Enterprise Linux V7 server and Red Hat Enterprise Linux V7 server Extras repositories on all servers:

```
# subscription-manager repos --disable=*
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-7-server-extras-rpms
```

b. Update the system to receive the latest packages:

```
# yum update -y
```

3. Create a Red Hat Ansible user with sudo access.

   User `rhcs-user` was created for deployment on Red Hat Ansible host:

```
# useradd rhcs-user
# cat /etc/sudoers.d/rhcs-user
rhcs-user ALL = (root) NOPASSWD:ALL
#chmod 0440 /etc/sudoers.d/rhcs-user
```

4. Enable password-less SSH for Red Hat Ansible:

   a. Set up password-less from Red Hat Ansible to all nodes. From `rhcs-user`, copy to all nodes:

```
[rhcs-user ~]$ ssh-keygen
[rhcs-user ~]$ ssh-copy-id rhcs-user@allnodes
```

   b. Create the SSH configuration file and update user name and hostname for all nodes:

```
# touch ~/.ssh/config
# cat .ssh/config
```

5. Set up the Ceph cluster:

   a. Using the root user account on the Red Hat Ansible administration node, enable the Red Hat Ceph Storage 3 Tools repository and Red Hat Ansible repository. Red Hat Ansible was set up on the dashboard node:

```
# subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
--enable=rhel-7-server-ansible-2.6-rpms
# yum install ceph-ansible
```

   b. As the Red Hat Ansible user, create the `ceph-ansible-keys` directory where Red Hat Ansible stores temporary values that are generated by the ceph-ansible playbook:

```
$ mkdir ~/ceph-ansible-keys
```

   c. As root, create a symbolic link to the `/usr/share/ceph-ansible/group_vars` directory in the `/etc/ansible/` directory:

```
$ ln -s /usr/share/ceph-ansible/group_vars /etc/ansible/group_vars
```

   d. Browse to the `/usr/share/ceph-ansible/` directory:

```
$ cd /usr/share/ceph-ansible
```

   e. Create copies of the yml files:

```
# cp group_vars/all.yml.sample group_vars/all.yml
# cp group_vars/osds.yml.sample group_vars/osds.yml
# cp site-docker.yml.sample site-docker.yml
```

   f. Edit the `group_vars/all.yml` file and update the parameters, as shown in Example 3-42.

Example 3-42 Update parameters in group_vars/All.yml file

```
[rhcs-user@ceph-dashboard.redbooks.infoceph-ansible]$ cat group_vars/all.yml |
grep -v "#" | grep -v "^$"
---
dummy:
fetch_directory: ~/ceph-ansible-keys
cluster: ceph
ceph_repository_type: cdn
```

```
ceph_origin: repository
ceph_repository: rhcs
ceph_rhcs_version: 3
monitor_interface: int2
ip_version: ipv4
public_network: 10.163.4.0/24
cluster_network: 10.162.62.0/26
ceph_conf_overrides:
  global:
        mon_allow_pool_delete: True
ceph_docker_image: "rhceph/rhceph-3-rhel7"
ceph_docker_image_tag: "latest"
ceph_docker_registry: "registry.access.redhat.com"
containerized_deployment: True
fetch_directory: ~/ceph-ansible-keys
ceph_origin: repository
ceph_repository: rhcs
ceph_rhcs_version: 3
ceph_docker_image: "rhceph/rhceph-3-rhel7"
ceph_docker_image_tag: "latest"
ceph_docker_registry: "registry.access.redhat.com"
[rhcs-user@ceph-dashboard ceph-ansible]$
```

We created 5 SAS disks + 1 NVMe disk pool for CGF application and 1 NVMe disk pool for Oracle DB.

Example 3-43 shows the `osds.yml` reference for Red Hat OpenStack application pool with five SAS disks and one NVMe disk.

Example 3-43 osds.yml reference file

```
[rhcs-user@ceph-dashboard.redbooks.infoceph-ansible]$ cat group_vars/osds.yml
|grep -v "#"|grep -v "^$"
---
dummy:
osd_objectstore: bluestore
osd_scenario: lvm
devices:
- /dev/sdb
  - /dev/sdc

[rhcs-user@ceph-dashboard.redbooks.info]$
```

g. Edit the Red Hat Ansible inventory file at `/etc/ansible/hosts`, as shown Example 3-44. (Remember to comment out example hosts.)

Example 3-44 Edit Red Hat Ansible inventory file

```
[rhcs-user@ceph-dashboard.redbooks.info]$ cat /etc/ansible/hosts| egrep -v "#|^$"
[mons]
red-ceph1.redbooks.in
red-ceph2.redbooks.in
red-ceph4.redbooks.in
[mgrs]
red-ceph1.redbooks.in
red-ceph2.redbooks.in
red-ceph4.redbooks.in
```

```
[osds]
red-ceph1.redbooks.in
red-ceph2.redbooks.in
red-ceph4.redbooks.in
red-ceph3.redbooks.in
[rhcs-user@ceph-dashboard.redbooks.info]$
```

h. As the Red Hat Ansible user, ensure that Red Hat Ansible can reach the Ceph hosts:

```
$ ansible all -m ping
```

i. As root, create the `/var/log/ansible/` directory and assign the suitable permissions for the Red Hat Ansible user:

```
$ mkdir /var/log/ansible
$ chown rhcs-user:rhcs-user  /var/log/ansible
$ chmod 755 /var/log/ansible
```

j. Edit the `/usr/share/ceph-ansible/ansible.cfg` file, updating the `log_path` value:

```
$ log_path = /var/log/ansible/ansible.log
```

k. As the Red Hat Ansible user, switch to the `/usr/share/ceph-ansible/` directory:

```
$ cd /usr/share/ceph-ansible/
```

l. Run the ceph-ansible playbook for Ceph cluster deployment:

```
$ ansible-playbook site-docker.yml
```

Note: This playbook must be run twice; one with CGF `osds.yml` (five SAS disks and one NVMe). Then, run the `site-docker.yml`, update as was done with `osds.yml` (see Example 3-43 on page 190) again for oracle (one NVMe disk), and run `site-docker.yml` again.

6. Verify the cluster and OSD status, as shown in Example 3-45.

Example 3-45 Verify the cluster and OSD status

```
[rhcs-user@red-ceph1.redbooks.info ~]$ ceph -s
   cluster:
    id:     b4864a77-08fc-4d4e-ad27-7b13de46221d
    health: HEALTH_WARN
            1 pools have many more objects per pg than average
            application not enabled on 4 pool(s)

  services:
    mon: 3 daemons, quorum red-ceph2,red-ceph4,red-ceph1
    mgr: red-ceph2(active), standbys: red-ceph1, red-ceph4
    osd: 8 osds: 8 up, 8 in

  data:
    pools:   5 pools, 304 pgs
    objects: 13.55k objects, 34.4GiB
    usage:   112GiB used, 6.87TiB / 6.98TiB avail
    pgs:     304 active+clean

  io:
    client:   138KiB/s wr, 0op/s rd, 7op/s wr
```

```
[rhcs-user@red-ceph1.redbooks.info ~]$
```

Check the disk status of each node as shown in Example 3-46. The logical volumes are created automatically under Ceph deployment on all OSD nodes.

Example 3-46 Check disk status of each node

```
[rhcs-user@red-ceph1.redbooks.info ~]$ lsblk
NAME
MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda
8:00 278.5G  0 disk
??sda1
8:10 2G  0 part /boot
??sda2
8:20 276.5G  0 part
??rhel-root
253:00 260.5G  0 lvm  /
 ??rhel-swap
253:1016G  0 lvm  [SWAP]
sdb
8:16   0   7.3T  0 disk
??ceph--block--f10437f7--9faa--42d9--91a0--fc3fdc7f6b42-osd--block--949048df--36de
--4d41--9d22--0d0610acf493 253:13   0   7.3T  0 lvm
sdc
8:32   0   7.3T  0 disk
??ceph--block--52858db2--0df6--43b5--b0ed--fd1d3a5ebac4-osd--block--7ab723d6--24d0
--4e5c--afd8--1c4acf305f31 253:15   0   7.3T  0 lvm

[rhcs-user@red-ceph1.redbooks.info ~]$
```

3.7.8 Installing the Red Hat Ceph Storage Dashboard

DNS is configured on the dashboard for name resolution. Update `/etc/resolv.conf` on all hosts with the following reference to set up the DNS: `nameserver 10.0.80.11`.

Note: If DNS services go down, the Ceph clusters are not affected, except dashboard services.

Complete the following steps:

1. Run the following command on the Red Hat Ansible administration node as the root user to install the cephmetrics-ansible package:

   ```
   # yum install cephmetrics-ansible
   ```

2. Using the Ceph Ansible inventory as a base, add the Red Hat Ceph Storage Dashboard node under the `[ceph-grafana]` section of the Red Hat Ansible inventory file (which by default is at `/etc/ansible/hosts`), as shown in Example 3-47.

Example 3-47 Add the Red Hat Ceph Storage Dashboard node

```
[root@ceph-dashboard.redbooks.info~]# cat /etc/ansible/hosts| egrep -v "#|^$"
[mons]
red-ceph1.redbooks.in
red-ceph2.redbooks.in
```

```
red-ceph4.redbooks.in
[mgrs]
red-ceph1.redbooks.in
red-ceph2.redbooks.in
red-ceph4.redbooks.in
[osds]
red-ceph1.redbooks.in
red-ceph2.redbooks.in
red-ceph4.redbooks.in
red-ceph3.redbooks.in
[ceph-grafana]
ceph-dashboard.redbook.in
[root@ceph-dashboard.redbooks.info~]#
```

3. Change to the `/usr/share/cephmetrics-ansible/` directory and update all.yml for dashboard:

```
[root@ceph-dashboard.redbooks.info~]# cd /usr/share/cephmetrics-ansible
[root@ceph-dashboard.redbooks.info]#  cat group_vars/all.yml| egrep -v "^#|^$"
dummy:
cluster_name: ceph
containerized: true
[root@ceph-dashboard.redbooks.infocephmetrics-ansible]#
```

4. Run the Red Hat Ansible playbook for Dashboard deployment by way of root user:

> **Note:** Check that the time was synced with the NTP server. Also, check that the firewall was enabled on the dashboard host because the playbook automatically adds the open ports that are required for dashboard.

```
# ansible-playbook -v playbook.yml
```

1. After the playbook deployment successfully completes, access the dashboard (see Figure 3-137) by using port 3000, and with default user `admin` and password `admin`.

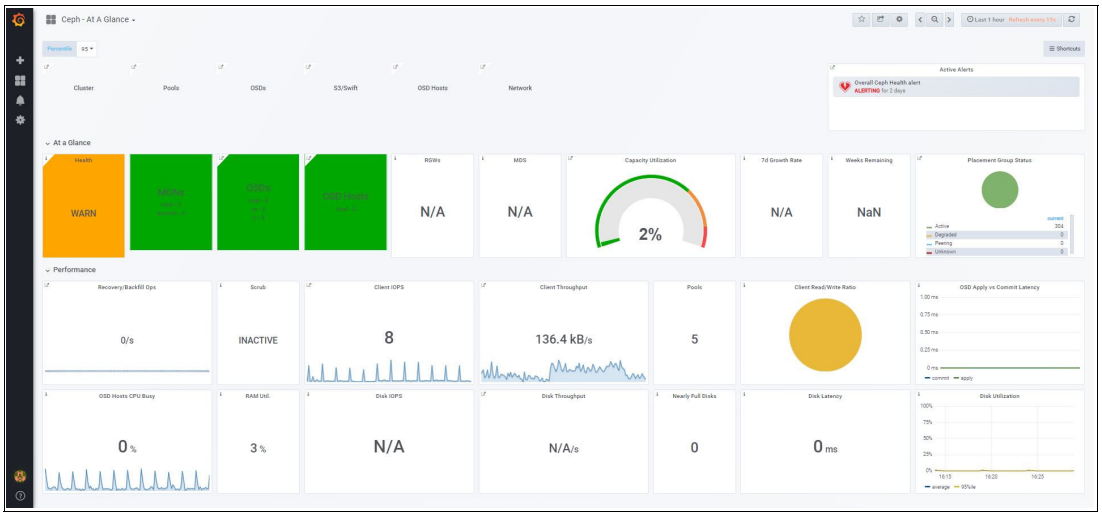


Figure 3-137 Ceph At A Glance window

Now, you can see three Mons and eight OSDs, as shown in Figure 3-138.

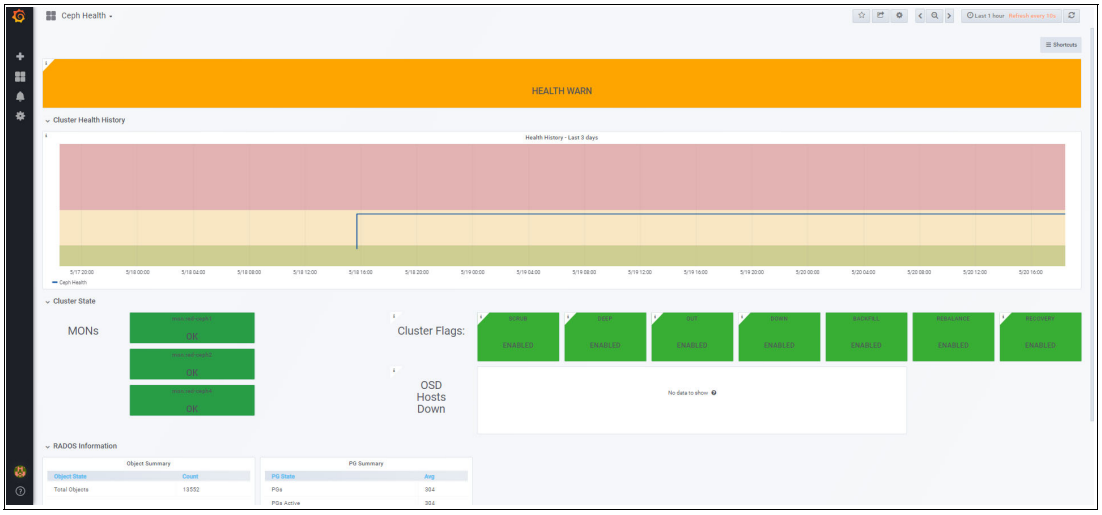


Figure 3-138 Ceph Health window

3.7.9 Configuring Red Hat Ceph storage cluster

Complete the following steps to configure the deployed Red Hat Ceph Cluster by creating pool and CRUSH rules:

1. In any MON node, create the CRUSH rule for SSD and HDD. Example 3-48 shows how to create `replicated_rule_ssd` and `replicated_rule_hdd` rule.

Example 3-48 Create replicated_rule_ssd and replicated_rule_hdd

```
[rhcs-user@red-ceph1.redbooks.info ~]$  ceph osd crush rule create-replicated
replicated_rule_ssd default host ssd
[rhcs-user@red-ceph1.redbooks.info ~]$
[rhcs-user@red-ceph1.redbooks.info ~]$  ceph osd crush rule create-replicated
replicated_rule_sata default host hdd
[rhcs-user@red-ceph1.redbooks.info ~]$
[rhcs-user@red-ceph1.redbooks.info ~]$ ceph osd crush rule ls
replicated_rule
replicated_rule_sata
replicated_rule_ssd
[rhcs-user@red-ceph1.redbooks.info ~]$
```

2. Create the pool for the application openstack 5 pool, as shown in Example 3-49.

Example 3-49 Create the pool

```
[rhcs-user@red-ceph1.redbooks.info ~]$ ceph osd pool create backups 8
ceph osd pool set backups size 3
while [ $(ceph -s | grep creating -c) -gt 0 ]; do echo -n .;sleep 1; done ~]$
[rhcs-user@red-ceph1.redbooks.info ~]$ ceph osd pool create volumes 128
ceph osd pool set volumes size 3
while [ $(ceph -s | grep creating -c) -gt 0 ]; do echo -n .;sleep 1; done
```

3. Set the replication of pool and map applications, as shown in Example 3-50.

Example 3-50 Set replication

```
[root@red-ceph1.redbooks.info ~]$ ceph osd pool create vms 128
ceph osd pool set vms size 3
while [ $(ceph -s | grep creating -c) -gt 0 ]; do echo -n .;sleep 1; done
[root@red-ceph1.redbooks.info ~]$ ceph osd pool create images 32
ceph osd pool set images size 3
while [ $(ceph -s | grep creating -c) -gt 0 ]; do echo -n .;sleep 1; done
[root@red-ceph1.redbooks.info ~]$
[root@red-ceph1.redbooks.info ~]$ ceph osd pool create metrics 8
ceph osd pool set metrics size 3
while [ $(ceph -s | grep creating -c) -gt 0 ]; do echo -n .;sleep 1; done
[root@red-ceph1.redbooks.info ~]$
```

4. Verify the pools and profile, as shown in Example 3-51.

Example 3-51 Verify pool and profile

```
[root@red-ceph1.redbooks.info ~]# [root@red-ceph1 ~]# ceph osd pool ls detail
pool 9 'backups' replicated size 3 min_size 2 crush_rule 0 object_hash rjenkins
pg_num 8 pgp_num 8 last_change 63 flags hashpspool stripe_width 0
pool 10 'volumes' replicated size 3 min_size 2 crush_rule 0 object_hash rjenkins
pg_num 128 pgp_num 128 last_change 81 flags hashpspool stripe_width 0
        removed_snaps [1~3]
pool 11 'images' replicated size 3 min_size 2 crush_rule 0 object_hash rjenkins
pg_num 32 pgp_num 32 last_change 89 flags hashpspool stripe_width 0
        removed_snaps [1~3]
pool 12 'metrics' replicated size 3 min_size 2 crush_rule 0 object_hash rjenkins
pg_num 8 pgp_num 8 last_change 72 flags hashpspool stripe_width 0
pool 13 'vms' replicated size 3 min_size 2 crush_rule 0 object_hash rjenkins
pg_num 128 pgp_num 128 last_change 85 flags hashpspool stripe_width 0
        removed_snaps [1~3]
[root@red-ceph1.redbooks.info ~]#
```

5. Verify both rules that were created for HDD, as shown in Example 3-52.

Example 3-52 Verify the rules for HDD

```
[rhcs-user@red-ceph1.redbooks.info ~]$ ceph osd crush rule dump
replicated_rule_sata
{
   "rule_id": 1,
   "rule_name": "replicated_rule_sata",
   "ruleset": 1,
   "type": 1,
   "min_size": 1,
   "max_size": 10,
   "steps": [
    {
        "op": "take",
        "item": -2,
        "item_name": "default~hdd"
    },
    {
        "op": "chooseleaf_firstn",
```

```
        "num": 0,
        "type": "host"
    },
    {
        "op": "emit"
    }
   ]
}

[rhcs-user@red-ceph1.redbooks.info ~]$
```

6. Assign rules to pools, as shown in Example 3-53.

Example 3-53 Assign rules to pool

```
[rhcs-user@red-ceph1.redbooks.info ~]$ ceph osd pool set bhubneshwar_sata
crush_rule replicated_rule_sata
set pool 2 crush_rule to replicated_rule_sata
[rhcs-user@red-ceph1.redbooks.info ~]$ ceph osd pool set bhubneshwar_ssd
crush_rule replicated_rule_ssd
set pool 1 crush_rule to replicated_rule_ssd
[rhcs-user@red-ceph1.redbooks.info ~]$
```

Docker containers information is on all nodes, as shown in Example 3-54.

Example 3-54 Docket container information in the nodes

```
[root@ceph-dashboard.redbooks.info~]#  ansible all -m shell -a "docker ps"
red-ceph1.redbooks.info | SUCCESS | rc=0 >>
CONTAINER ID    IMAGE                                                  COMMAND         CREATED
STATUS          PORTS           NAMES
8062adb9cd11    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-osd-33
520cd681cbdc    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-osd-11
4149152230bc    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-osd-24
fbc34f854604    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-osd-20
7252bc332ee4    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-osd-15
eaf0bfe65224    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-osd-3
bf263e99e7d0    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-osd-19
7445303aea58    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-osd-7
01af9b0159d4    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-osd-28
9ed38c5eaf2c    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-mgr-red-ceph1.redbooks.info
f8348124092d    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-mon-red-ceph1.redbooks.info

red-ceph3.redbooks.in | SUCCESS | rc=0 >>
CONTAINER ID    IMAGE                                                  COMMAND         CREATED
STATUS          PORTS           NAMES
6deb55569b00    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                       ceph-osd-23
```

```
a65bc105226c    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-31
94226dd1915e    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-1
cf195deee776    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-13
478fd11891b7    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-35
b1d0d605a2cb    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-18
361d6cfe5f37    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-27
ab6a90e21def    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-9
6c9a79cec635    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-5
7bc8be4c61e5    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-mgr-or-bhu-inf-ucs-ceph-osd-03
d9e91cbc01b5    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-mon-or-bhu-inf-ucs-ceph-osd-03

red-ceph2.redbooks.info | SUCCESS | rc=0 >>
CONTAINER ID    IMAGE                                                     COMMAND         CREATED
STATUS          PORTS           NAMES
285192c58ddb    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-6
ccbac22c3dc3    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-21
beb51676a276    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-2
f4a6a6693bf0    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-32
ff816436d575    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-25
f673893749ac    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-29
4bed5d38eaed    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-17
bba8dc35667e    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-10
03ffe064b0c7    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-14
38053bb88e10    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-mon-or-bhu-inf-ucs-ceph-osd-02
2f9593950151    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-mgr-or-bhu-inf-ucs-ceph-osd-02

red-ceph4.redbooks.info | SUCCESS | rc=0 >>
CONTAINER ID    IMAGE                                                     COMMAND         CREATED
STATUS          PORTS           NAMES
71661835c15b    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-12
f18f35d53a64    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-8
21e0674a0272    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-26
8a132c7757c7    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-22
813c63a00f2b    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-16
```

```
fa0bd354ffc9    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-34
0b633004eb1d    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-30
22551177534d    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-4
530c755738b1    registry.access.redhat.com/rhceph/rhceph-3-rhel7:latest   "/entrypoint.sh"3 days ago
Up 3 days                           ceph-osd-0

ceph-dashboard.redbooks.info | SUCCESS | rc=0 >>
CONTAINER ID    IMAGE                                                      COMMAND              CREATED
STATUS          PORTS                NAMES
c9553a261999    registry.access.redhat.com/rhceph/rhceph-3-dashboard-rhel7:3   "/run.sh"            5 days
ago      Up 3 days       0.0.0.0:3000->3000/tcp   grafana-server
7b02bc8f2b15    registry.access.redhat.com/openshift3/prometheus:v3.9      "/bin/prometheus -..."   5 days
ago      Up 3 days       0.0.0.0:9090->9090/tcp   prometheus

[root@ceph-dashboard.redbooks.info~]#
```

3.7.10 Integrating deployed Ceph with Red Hat OpenStack platform

This section reviews the Red Hat OpenStack and Red Hat Ceph Integration. The requirements for this integration with the Red Hat OpenStack overcloud also are described. This section also deploys an overcloud with a Ceph configuration.

Why Ceph storage for Red Hat OpenStack Cloud?

Red Hat Ceph Storage scales as Red Hat OpenStack scales (see Figure 3-139): quickly, reliably, and cost-effectively, on industry-standard servers and disks. It also maps to Red Hat OpenStack's modular architecture and components for secure storage of petabytes of data.

The Ceph Block Device (RBD) provides a single backend for Red Hat OpenStack's Nova, Glance, and Cinder services to efficiently store images, volumes, and snapshots. The CephFS file system provides the underlying support for the Manila file sharing service, while the Ceph Object Gateway (RGW) provides a REST interface that is compatible with applications written for Red Hat OpenStack Swift and supports the Keystone authentication service. RBD, CephFS, and RGW are all served by the same storage cluster, which supports all Red Hat OpenStack services.

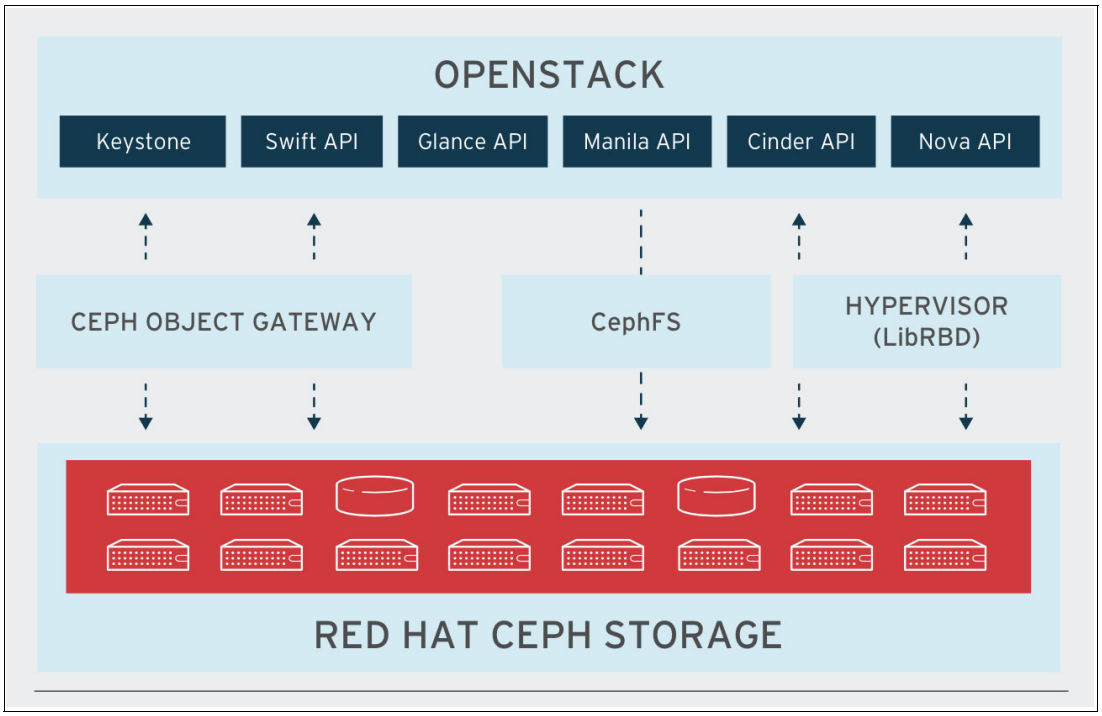


Figure 3-139 Red Hat Ceph and Red Hat OpenStack Integration

Using Red Hat Ceph Storage, Red Hat OpenStack users can start one or hundreds of VMs instantaneously, which are then readily accessible by cloud users. Backups are also instantaneous by using seamless integration between Ceph and the Glance, Cinder, and Nova services. Red Hat Ceph Storage also supports different storage tiers to optimize costs and performance ratios for different workloads on the same cluster, and multi-site replication for disaster recovery or archiving.

Integration scenario

Red Hat OpenStack Platform director creates a cloud environment called the *overcloud*. The director is used to configure extra features for an overcloud. One of these extra features includes integration with Red Hat Ceph Storage. This storage includes Ceph Storage clusters that are created with the director or existing Ceph Storage clusters.

This section describes integrating an existing Ceph Storage cluster with an overcloud. This means the director configures the overcloud to use the Ceph Storage cluster for storage needs. You manage and scale the cluster outside of the overcloud configuration.

Preparing overcloud nodes

An overcloud already is deployed in Red Hat OpenStack, as described in 3.2.3, “Overcloud” on page 41. The scenario in this section consists of six nodes in the overcloud:

- Three Controller nodes with HA
- Three Compute nodes

The director integrates a separate Ceph Storage Cluster with its own nodes into the overcloud. You manage this cluster independently from the overcloud.

Configuring the deployed Ceph Cluster for integration

Complete the following steps to configure the Ceph Cluster for integration:

1. Create the following pools in your Ceph cluster relevant to your environment:
   - Volumes: Storage for Red Hat OpenStack Block Storage (cinder).
   - Images: Storage for Red Hat OpenStack Image Storage (glance).
   - vms: Storage for instances.
   - Backups: Storage for Red Hat OpenStack Block Storage Backup (cinder-backup).
   - Metrics: Storage for Red Hat OpenStack Telemetry Metrics (gnocchi). Use the following commands as a guide:

     ```
     [root@ceph ~]# ceph osd pool create volumes PGNUM
     [root@ceph ~]# ceph osd pool create images PGNUM
     [root@ceph ~]# ceph osd pool create vms PGNUM
     [root@ceph ~]# ceph osd pool create backups PGNUM
     [root@ceph ~]# ceph osd pool create metricsPGNUM
     ```

     Note: Replace PGNUM with the number of placement groups. We recommend approximately 100 per OSD.

2. Create a `client.openstack` user in your Ceph cluster with the following capabilities:
   - cap_mon: Allow r.
   - cap_osd: Allow class-read object_prefix rbd_children, allow rwx pool=volumes, allow rwx pool=vms, allow rwx pool=images, allow rwx pool=backups, allow rwx pool=metrics. Use the following command as a guide:

     ```
     [root@ceph ~]# ceph auth add client.openstack mon 'allow r' osd 'allow
     class-read object_prefix rbd_children, allowrwx pool=volumes, allowrwx
     pool=vms, allowrwx pool=images, allowrwx pool=backups, allow rwx
     pool=metrics'
     ```

3. Make a note the Ceph client key that was created for the `client.openstack` user, as shown in Example 3-55.

Example 3-55 Create the Ceph client key

```
[root@ceph ~]# ceph auth list

...client.openstackkey: AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ==caps: [mon] allow
rcaps:[osd] allow class-read object_prefix rbd_children, allowrwx pool=volumes,
allowrwx pool=vms, allow rwx pool=images, allow rwx pool=backups, allow rwx
pool=metrics...
```

The key value here (`AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ==`) is your Ceph client key.

Note also the file system ID of your Ceph Storage cluster. This value is specified with the `fsid` setting in the configuration file of your cluster (under the `[global]` section):

```
[global]fsid = 4b5c8c0a-ff60-454b-a1b4-9747aa737d19...
```

The Ceph client key and file system ID are used later.

Initializing the stack user

Log in to the director host as the stack user and run the following command to initialize your director configuration:

```
$ source ~/stackrc
```

This process sets up environment variables that contain authentication details to access the director's CLI tools.

Registering nodes

A node definition template (`instackenv.json`) is a JSON format file and contains the hardware and power management details for registering nodes, as shown in Example 3-56.

Example 3-56 Node definition: JSON format file

```
{"nodes":[{"mac":["bb:bb:bb:bb:bb:bb"],
"cpu":"4","memory":"6144","disk":"40","arch":"x86_64", "pm_type":"pxe_ipmitool",
"pm_user":"admin", "pm_password":"p@55w0rd!",

"pm_addr":"192.0.2.205"},{"mac":["cc:cc:cc:cc:cc:cc"],
"cpu":"4","memory":"6144","disk":"40","arch":"x86_64", "pm_type":"pxe_ipmitool",
"pm_user":"admin", "pm_password":"p@55w0rd!",
"pm_addr":"192.0.2.206"},{"mac":["dd:dd:dd:dd:dd:dd"],
"cpu":"4","memory":"6144","disk":"40","arch":"x86_64", "pm_type":"pxe_ipmitool",
"pm_user":"admin", "pm_password":"p@55w0rd!",
"pm_addr":"192.0.2.207"},{"mac":["ee:ee:ee:ee:ee:ee"],
"cpu":"4","memory":"6144","disk":"40","arch":"x86_64", "pm_type":"pxe_ipmitool",
"pm_user":"admin", "pm_password":"p@55w0rd!",
"pm_addr":"192.0.2.208"}{"mac":["ff:ff:ff:ff:ff:ff"],
"cpu":"4","memory":"6144","disk":"40","arch":"x86_64", "pm_type":"pxe_ipmitool",
"pm_user":"admin", "pm_password":"p@55w0rd!",
"pm_addr":"192.0.2.209"}{"mac":["gg:gg:gg:gg:gg:gg"],"cpu":"4","memory":"6144","di
sk":"40","arch":"x86_64", "pm_type":"pxe_ipmitool", "pm_user":"admin",
"pm_password":"p@55w0rd!", "pm_addr":"192.0.2.210"}]}
```

After creating the template, save the file to the stack user's home directory (`/home/stack/instackenv.json`). Initialize the stack user; then, import `instackenv.json` into the directory:

```
$ source ~/stackrc$ openstack overcloud node import ~/instackenv.json
```

The use of this command imports the template and registers each node from the template into the director. Assign the kernel and ramdisk images to each node:

```
$ openstack overcloud node configure <node>
```

The nodes are now registered and configured in the directory.

Manually tagging the nodes

After registering each node, you must inspect the hardware and tag the node into a specific profile. Profile tags match your nodes to flavors, and in turn the flavors are assigned to a deployment role.

To inspect and tag new nodes, complete the following steps:

1. Trigger hardware introspection to retrieve the hardware attributes of each node:

   ```
   $ openstack overcloud node introspect --all-manageable --provide
   ```

   The **`--all-manageable`** option introspects only nodes in a managed state. In this case, it is all of them.

   The **`--provide`** option resets all nodes to an active state after introspection.

   Important: Check that this process runs to completion. This process usually takes 15 minutes for bare metal nodes.

2. Retrieve a list of your nodes to identify their UUIDs:

   ```
   $ openstack baremetal node list
   ```

3. Add a profile option to the properties/capabilities parameter for each node to manually tag a node to a specific profile.

   For example, to tag three nodes to use the control profile and another three nodes to use the compute profile, run the command as shown in Example 3-57.

Example 3-57 Tagging nodes for the control profile

```
$ ironic node-update 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 add
properties/capabilities='profile:control,boot_option:local'

$ ironic node-update 6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a add
properties/capabilities='profile:control,boot_option:local'

$ ironic node-update 5e3b2f50-fcd9-4404-b0a2-59d79924b38e add
properties/capabilities='profile:control,boot_option:local'

$ ironic node-update 484587b2-b3b3-40d5-925b-a26a2fa3036f add
properties/capabilities='profile:compute,boot_option:local'

$ ironic node-update d010460b-38f2-4800-9cc4-d69f0d067efe add
properties/capabilities='profile:compute,boot_option:local'

$ ironic node-update d930e613-3e14-44b9-8240-4f3559801ea6 add
properties/capabilities='profile:compute,boot_option:local'
```

```
The addition of the profile option tags the nodes into each respective profile.
```

Integrating the existing Ceph cluster

The Heat template collection directory contains the necessary templates and environment files to deploy an overcloud:

```
/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible
- external.yaml
```

The director uses ceph-ansible to integrate with a Ceph cluster, but ceph-ansible is not installed by default on the undercloud. Run the following command to install the ceph-ansible package on the undercloud:

```
sudo yum install -y ceph-ansible
```

To configure the integration, you must supply the details of your Ceph cluster to the director. To do this, use a custom environment file, which allows to override the default settings used by:

```
/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible
-external.yam
```

Complete the following steps:

1. Create the following custom environment file:

   ```
   /home/stack/templates/ceph-config.yaml
   ```

2. Add a `parameter_defaults` header to this file: `parameter_defaults`.

   Under this header (see Example 3-58), set all the parameters you want to override in `/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml`. At a minimum, you need to set:

   – CephClientKey: The Ceph client key of your Ceph Storage cluster.

   – CephClusterFSID: The file system ID of your Ceph Storage cluster. This is the value of `fsid` in your Ceph Storage cluster configuration file.

   – CephExternalMonHost: A comma-delimited list of the IPs of all MON hosts in your Ceph Storage cluster (for example, 172.16.1.7, 172.16.1.8).

Example 3-58 Add to the parameters_default

```
parameter_defaults:CephClientKey: AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ==
CephClusterFSID: 4b5c8c0a-ff60-454b-a1b4-9747aa737d19 CephExternalMonHost:
172.16.1.7, 172.16.1.8
```

3. If necessary, set the name of the Red Hat OpenStack pools and the client user by using the following parameters and values:

   ```
   *CephClientUserName: openstack
   *NovaRbdPoolName: vms
   *CinderRbdPoolName: volumes
   *GlanceRbdPoolName: images
   *CinderBackupRbdPoolName: backups
   *GnocchiRbdPoolName: metrics
   ```

You can also add overcloud parameters to your custom environment file. For example, to set vxlan as the neutron network type, add the following to `parameter_defaults`:

```
Neutron Network Type: vxlan
```

Backwards compatibility with older versions of Red Hat Ceph Storage

If you are integrating Red Hat OpenStack Platform with an external Ceph Storage Cluster from an earlier version, you might need to enable compatibility with earlier versions.

If you are running Red Hat Ceph Storage 1.3, you must add the following line to the `parameter_defaults` of your custom environment file (in this case, `/home/stack/templates/ceph-config`):

```
parameter_defaults:RbdDefaultFeatures: 1
```

If you are running an external Red Hat Ceph Storage 2.x cluster, it is not necessary to use this parameter. Though Red Hat OpenStack Platform 13 uses Red Hat Ceph Storage V3.x clients, those clients remain compatible with a Red Hat Ceph Storage V2.x server.

3.7.11 Deploying an overcloud

The creation of the overcloud requires more arguments for the **`openstack overcloud deploy`** command; for example:

```
$ openstack overcloud deploy --templates \-e
/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible
- external.yaml \-e /home/stack/templates/ceph-config.yaml \-e --ntp-server
pool.ntp.org \
```

The command uses the following options:

- `--templates`

  Creates the overcloud from the default Heat template collection (namely, `/usr/share/openstack-tripleo-heat-templates/`).

- `-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible- external.yaml`

  Sets the director to integrate a Ceph cluster to the overcloud.

- `-e /home/stack/templates/ceph-config.yaml`

  Adds a custom environment file to override the defaults set by:

  `-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml`

- `--ntp-server pool.ntp.org`

  Sets our NTP server.

The overcloud creation process begins and the director provisions your nodes. This process takes some time to complete. To view the status of the overcloud creation, open a separate terminal as the stack user and run:

```
$ source ~/stackrc$ openstack stack list --nested
```

Example 3-59 on page 205 configures the overcloud to use your external Ceph Storage cluster. You manage this cluster independently from the overcloud. For example, you scale the Ceph Storage cluster by using the Ceph management tools and not through the Red Hat OpenStack Platform director.

Example 3-59 Overcloud configuration for Ceph storage cluster

```
nohup openstack overcloud deploy --verbose --templates \
  -e /home/stack/templates/node-info.yaml \
  -e
/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible
-external.yaml \
  -e /home/stack/templates/ceph-config.yaml \
  -e /home/stack/templates/service_net.yaml \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/ips-from-pool-all.yaml \
  -e /home/stack/templates/network-environment.yaml \
  -r /home/stack/templates/roles_data.yaml \
  --ntp-server 10.0.77.54 &

[stack@red-director templates]$ cat ceph-config.yaml
```

Parameter_defaults:

```
#NOTE: These example parameters are required when using CephExternal
  CephClusterFSID: 'b4864a77-08fc-4d4e-ad27-7b13de46221d'
  CephClientKey: 'AQAJDsFeMCD9ERAAUg6z+P5EELf/YCqNnjzEUg=='
  CephExternalMonHost: '10.162.62.41,10.162.62.11,10.162.62.24'
```

3.7.12 Accessing the overcloud

The director generates a script to configure and help authenticate interactions with your overcloud from the director host. The director saves this file (overcloudrc) in your stack user's home directory. Run the following command to use this file:

```
$ source ~/overcloudrc
```

The necessary environment variables are loaded to interact with your overcloud from the director host's CLI. To return to interacting with the director's host, run the following command:

```
$ source ~/stackrc
```

3.8 Red Hat CloudForms

Red Hat CloudForms delivers the insight, control, and automation that enterprises need to address the challenges of managing virtual environments. This technology enables enterprises with existing virtual infrastructures to improve visibility and control, and those starting virtualization deployments to build and operate a well-managed virtual infrastructure.

Red Hat CloudForms provides the following feature sets:

- Insight: Discovery, Monitoring, Utilization, Performance, Reporting, Analytic, Chargeback, and Trending
- Control: Security, Compliance, Alerting, and Policy-Based Resource, and Configuration Enforcement
- Automate: IT Process, Task and Event, Provisioning, and Workload Management and Orchestration

- Integrate: Systems Management, Tools and Processes, Event Consoles, Configuration Management Database (CMDB), Role-based Administration (RBA), and Web Services.]

Note: For more information about installing Red Hat CloudForms, see 3.8.2, “Installing and configuring Red Hat CloudForms” on page 207.

3.8.1 Architecture components

Figure 3-140 describes the capabilities of Red Hat CloudForms. Its features are designed to work together to provide robust management and maintenance of your virtual infrastructure.

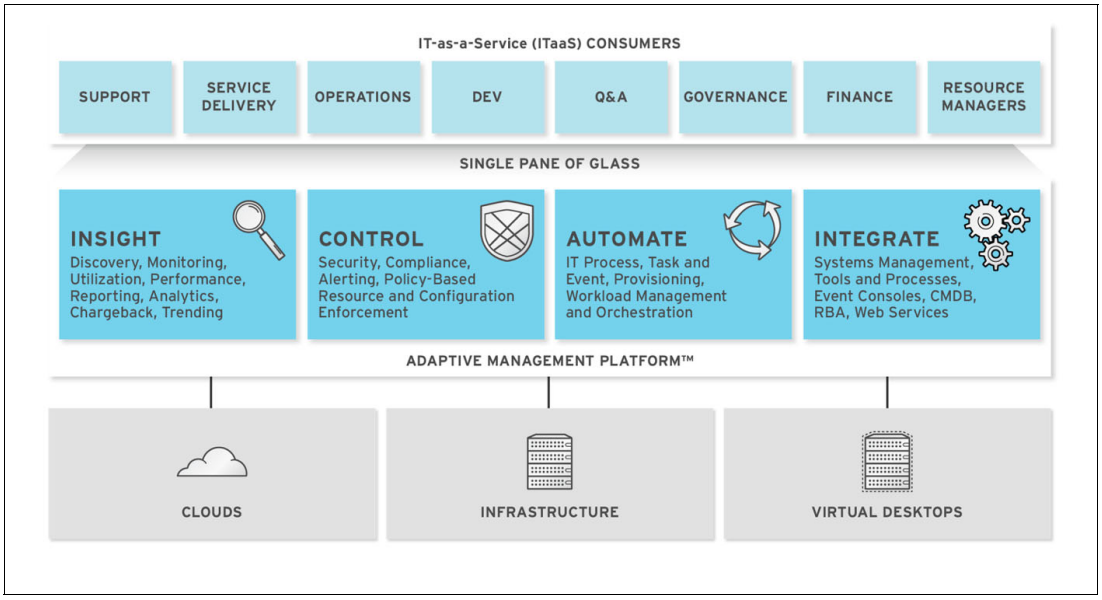


Figure 3-140 Red Hat CloudForms characteristics

The architecture consists of the following components (see Figure 3-141 on page 207):

- The Red Hat CloudForms appliance (appliance), which is supplied as a secure, high-performance, and preconfigured VM. It provides support for HTTPS communications.
- The Red Hat CloudForms Server (Server) is on the appliance. It is the software layer that communicates between the SmartProxy and the Virtual Management Database (VMDB). It includes support for HTTPS communications.
- The VMDB is on the appliance or another computer that is accessible to the appliance. It is the definitive source of intelligence that is collected about your virtual infrastructure. It also holds status information regarding appliance tasks.
- The Red Hat CloudForms Console (Console) is the Web interface that is used to view and control the server and appliance. It is uses through Web 2.0 mashups and web services (WS Management) interfaces.
- The SmartProxy can be on the appliance or on an ESX Server. If not embedded in the server, the SmartProxy can be deployed from the appliance. A SmartProxy agent must be configured in each storage location, and be visible to the appliance. The SmartProxy acts on behalf of the appliance that is communicating with it over HTTPS on standard port 443.

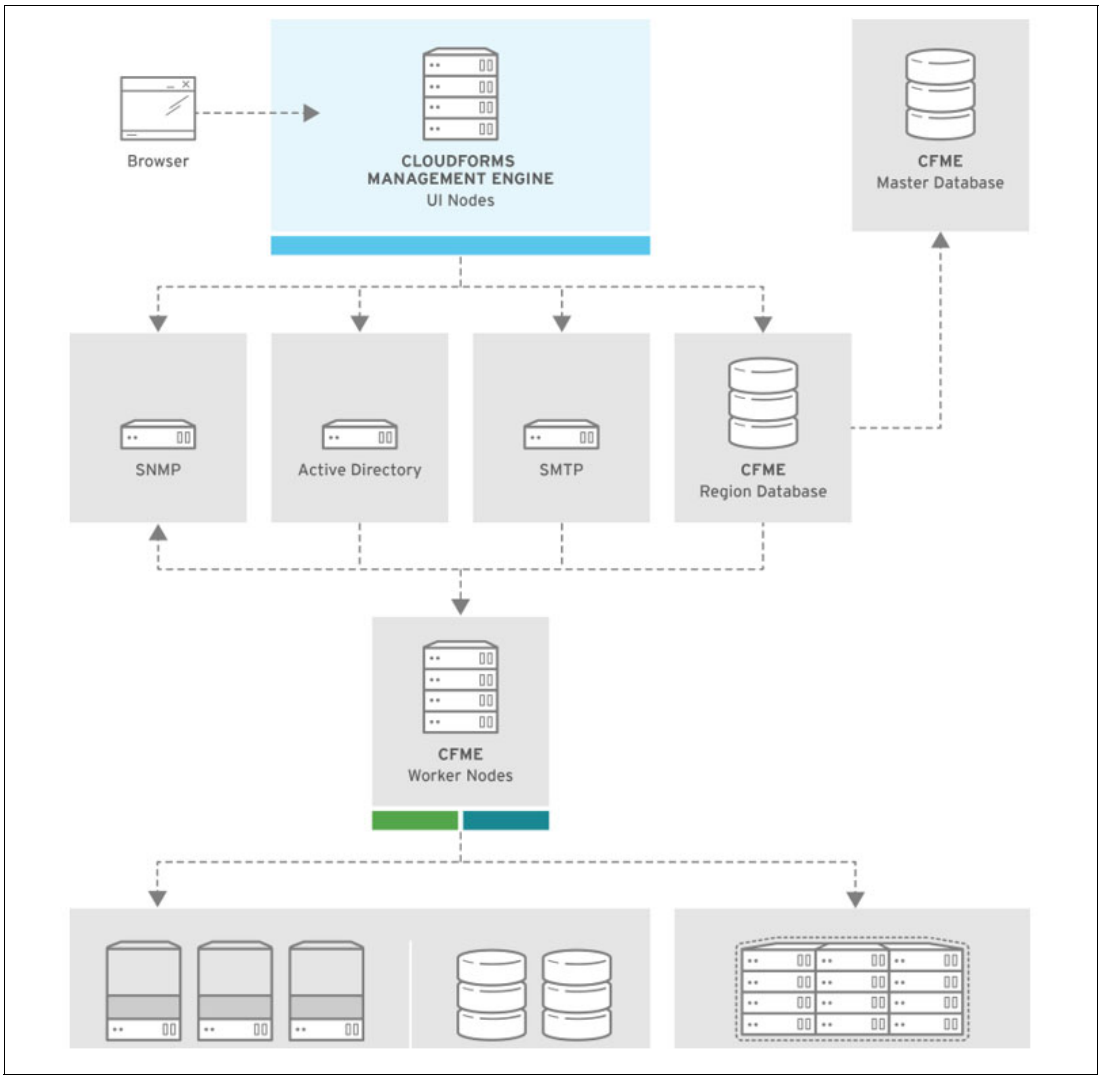


Figure 3-141 Red Hat CloudForms architecture

3.8.2 Installing and configuring Red Hat CloudForms

Installing Red Hat CloudForms consists of the following tasks:

- Downloading the appliance for your environment as a VM image template.
- Setting up a VM on the appliance.
- Installing the CFME Virtual Appliance.
- Configuring the CloudForms appliance:
  - Configuring a database for Red Hat CloudForms
  - Configuring the appliance.

Downloading the appliance

Complete the following steps:

1. At `access.redhat.com`, log in to the Red Hat Customer Portal by using your customer account information, as shown in Figure 3-142.

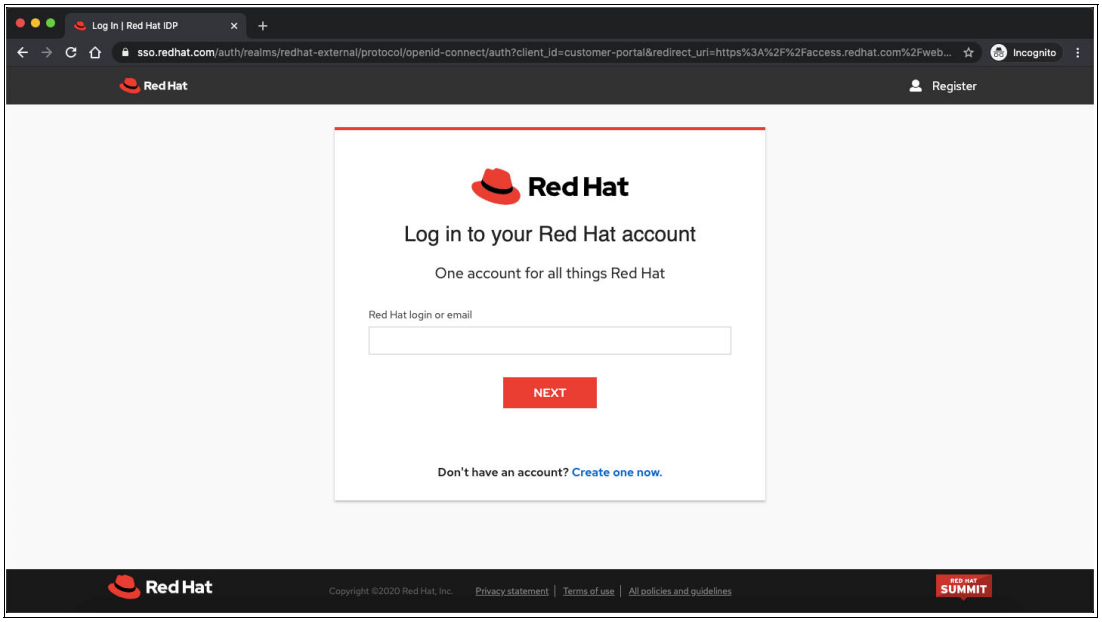


Figure 3-142 Red Hat Customer portal login page

2. Click **Downloads** in the menu bar, as shown in Figure 3-143.

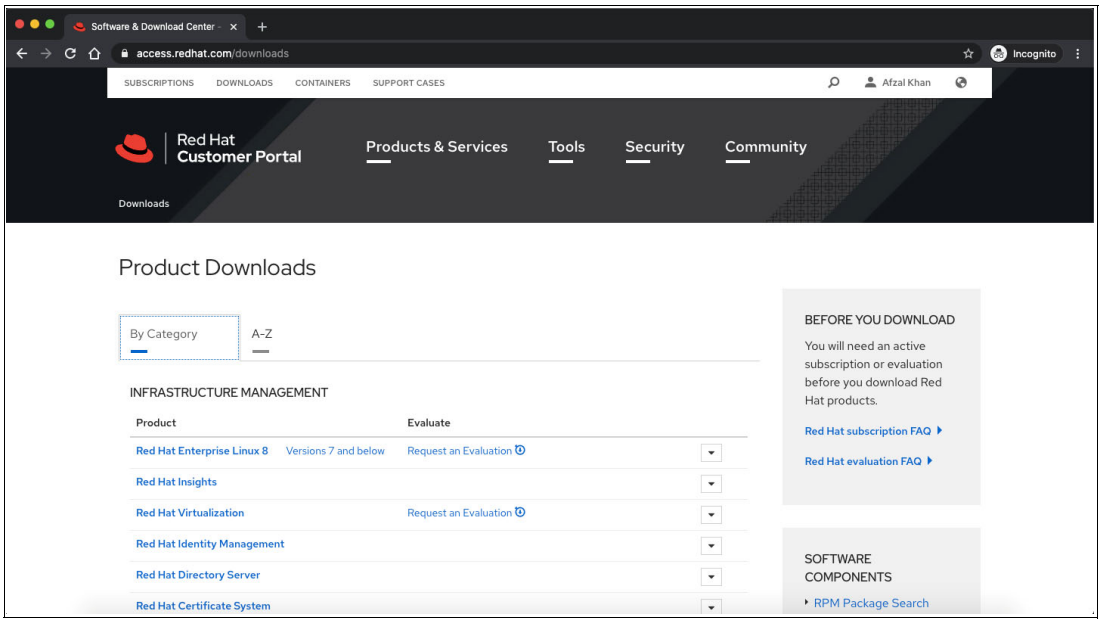


Figure 3-143 Red Hat Customer Portal Products Download Page

3. Click **Red Hat CloudForms** to access the product download page, as shown in Figure 3-144.

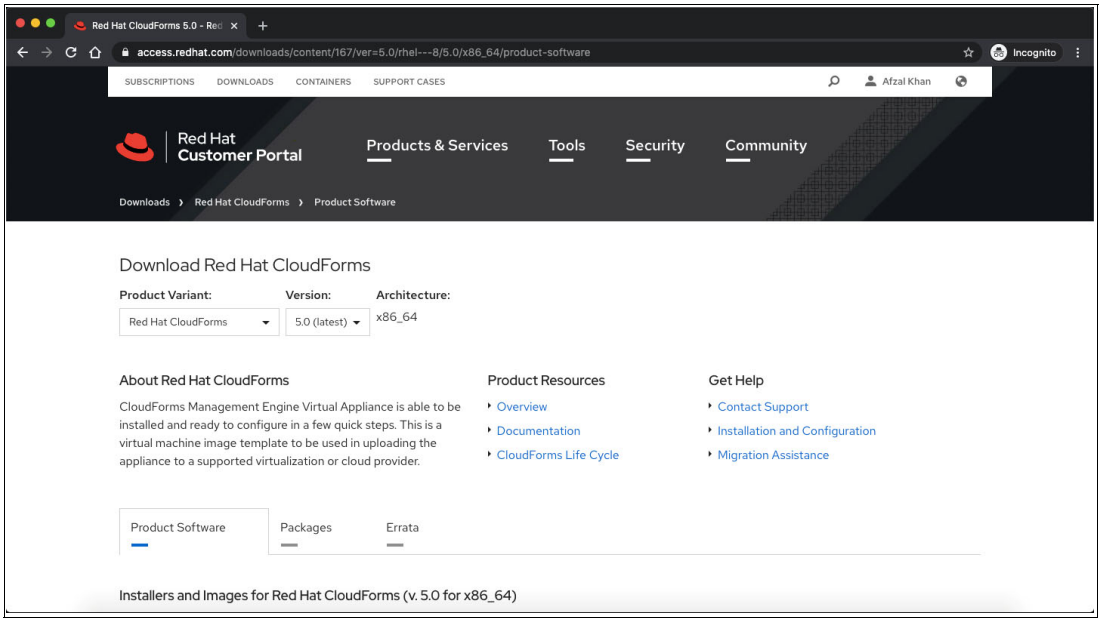


Figure 3-144 Red Hat CloudForms Product page

4. From the list of installers and images (see Figure 3-145), click **Download Now** for the Red Hat Virtual Appliance (`qcow`).

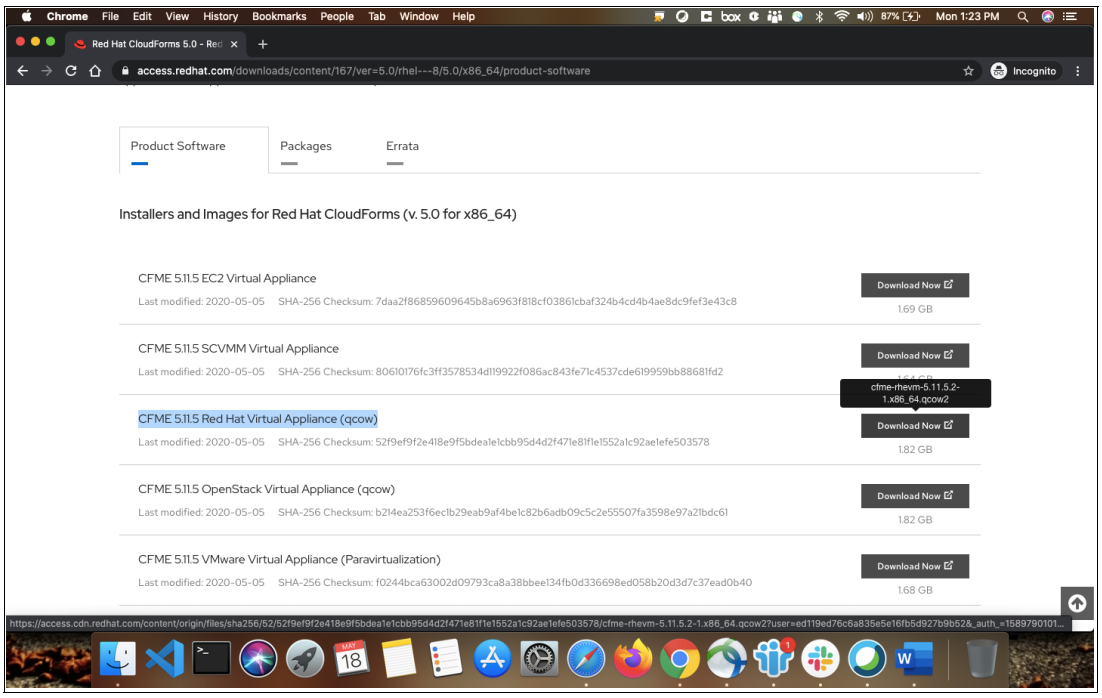


Figure 3-145 Red Hat CloudForms Appliance Download page

Creating VMs by using downloaded CFME Virtual Appliance

Complete the following steps:

1. Import the Red Hat Virtual Appliance (`qcow`) image in KVM virtualization server.
2. Copy the `qcow2` image to create four copies of the image.
3. Create four VMs with the virtual hardware configuration that is listed in Table 3-17 by using the copied images.

Table 3-17 VM configuration

Storage	50 GB
RAM	12 GB
CPUs	4 vCPU

4. Attach a 200 GB VDisk on the VM that acts as database server for Red Hat CloudForms.

Configuring a database for Red Hat CloudForms

Before Red Hat CloudForms are used, the database options must be configured for it. Red Hat CloudForms provides two options for database configuration:

- Install an internal PostgreSQL database to the appliance.
- Configure the appliance to use an external PostgreSQL database.

For this environment, an internal PostgreSQL database was selected. First, add a 100 GB VDisk to the VM tagged as database appliance in Red Hat CloudForms cluster. Then, complete the follow steps:

1. Start the appliance and open a terminal console.
2. After starting the appliance, log in with a user name of `root` and the default password of `smartvm`. The the Bash prompt is displayed for the root user.
3. Run the `appliance_console` command. The Red Hat CloudForms appliance summary pane displays, as shown in Figure 3-146.

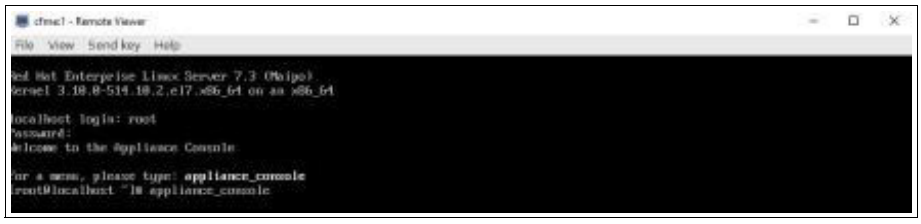


Figure 3-146 Red Hat CloudForms: Appliance summary pane

4. Press Enter to manually configure settings.
5. Select **5) Configure Database**.
6. You are prompted to create or fetch an encryption key:
   - If this appliance is the first Red Hat CloudForms appliance, select **1) Create key**.
   - If this appliance is not the first Red Hat CloudForms appliance, select **2) Fetch key from remote machine** to fetch the key from the first appliance. For worker and multi-region setups, use this option to copy key from another appliance (see Figure 3-147 on page 211).

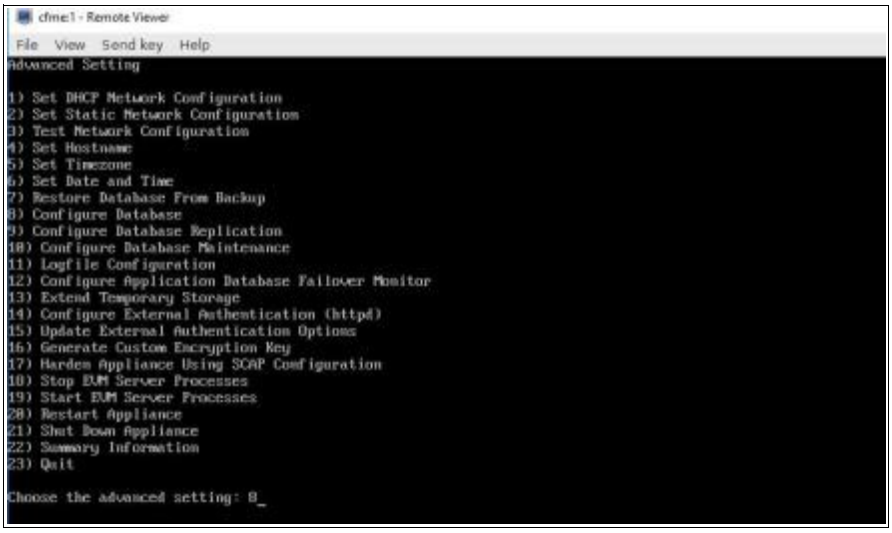


Figure 3-147 Enhance Settings selection pane

Note: All Red Hat CloudForms appliances in a multi-region deployment must use the same key.

7. Choose **1) Create Internal Database** for the database location. Choose a disk for the database, which can be a disk that you attached previously, or a partition on the current disk (see Figure 3-148).

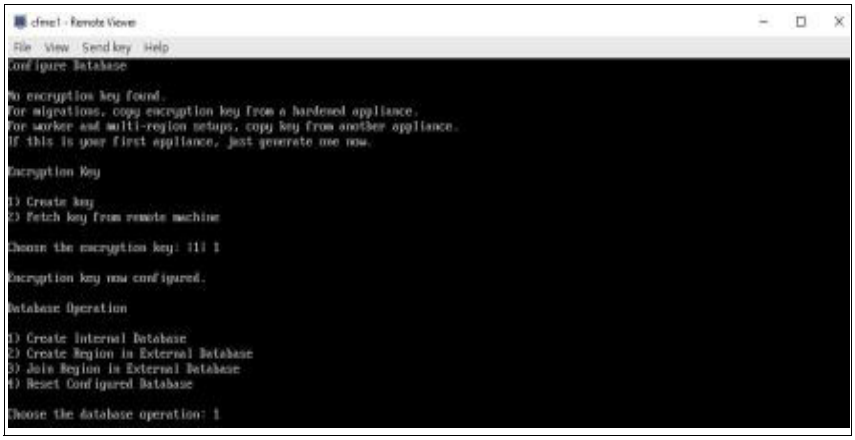


Figure 3-148 Configure database pane

8. Do not partition the disk (Figure 3-149 on page 212):
   - Enter 1 to choose `/dev/vdb` for the database location. This option creates a logical volume that uses this device and mounts the volume to the appliance in a location that is suitable for storing the database. The default location is `/var/opt/rh/rh-postgresql95/lib/pgsql`, which can be found in the environment variable $APPLIANCE_PG_MOUNT_POINT.
   - Enter 2 to continue without partitioning the disk. A second prompt confirm this choice. Selecting this option results in the use of the root file system for the data directory, which is *not* advised in most cases.

Figure 3-149 Database disk configuration pane

9. Enter Y or N for in response to the question: `Should this appliance run as a stand-alone database server?` (see Figure 3-150):
   - Select Y to configure the appliance as a database-only appliance. As a result, the appliance is configured as a basic PostgreSQL server, without a user interface.
   - Select N to configure the appliance with the full administrative user interface.

Figure 3-150 Database server configuration pane

10. When prompted, enter a unique number to create a region.

> **Important:** Creating a region destroys any data that is on the chosen database.

11. Create and confirm a password for the database, as shown in Figure 3-151.

Figure 3-151 Region and password database selection pane

Configuring worker appliances of Red Hat CloudForms

Complete the following steps for the remaining three appliances:

1. Start the appliance and open a terminal console.
2. After starting the appliance, log in with the user name `root` and the default password `smartvm`. The Bash prompt is displayed for the root user.
3. Run the **`appliance_console`** command. The Red Hat CloudForms appliance summary pane is displayed.
4. Press Enter to manually configure settings.
5. Select **5) Configure Database**.
6. You are prompted to create or fetch a security key. Because this appliance is not the first Red Hat CloudForms appliance, choose **2) Fetch key from remote machine**. For worker and multi-region setups, use this option to copy key from another appliance.

Note: All Red Hat CloudForms appliances in a multi-region deployment must use the same key.

7. Choose **3) Join Region in External Database** for the database location.
8. Enter the database hostname or IP address when prompted.
9. Enter the port number or leave blank for the default (`5432`).
10. Enter the database name or leave blank for the default (`vmdb_production`).
11. Enter the database user name or leave blank for the default (`root`).
12. Enter the chosen database user's password.
13. Confirm the configuration, if prompted.

After Red Hat CloudForms is installed, you can log in and perform administration tasks.

3.8.3 Administering and managing Red Hat OpenStack by using Red Hat CloudForms

This section shows how to administer and manage the Red Hat OpenStack from the Red Hat CloudForms UI. This section also describes how to add an Red Hat OpenStack Cloud as Provider to Red Hat CloudForms. Finally, you see how to view the orchestration stacks and Topology of deployment from Red Hat CloudForms UI.

Adding Red Hat OpenStack as cloud provider to Red Hat CloudForms

Complete the following steps:

1. Log in to Red Hat CloudForms for the first time after installing it.
2. Browse to the URL for the log in window, as shown in Figure 3-152 (`https://xx.xx.xx.xx` on the VM instance).

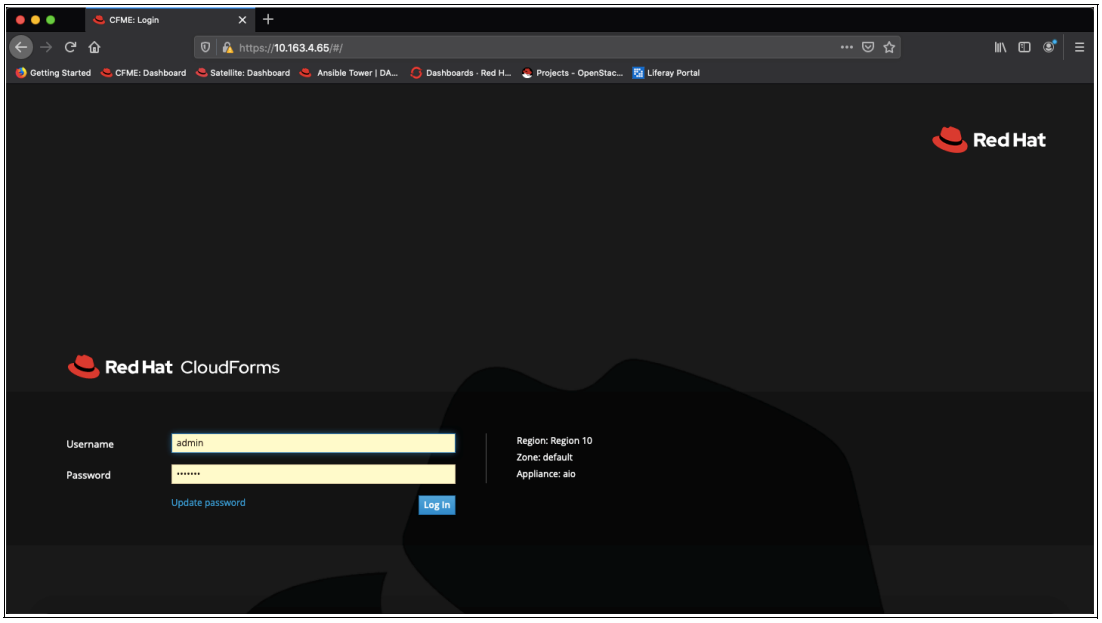


Figure 3-152 Red Hat CloudForms login pane

3. Enter the default credentials (user name is `admin`; password is `smartvm`) for the initial login.

4. Click **Login**. The Dashboard window opens, as shown in Figure 3-153.

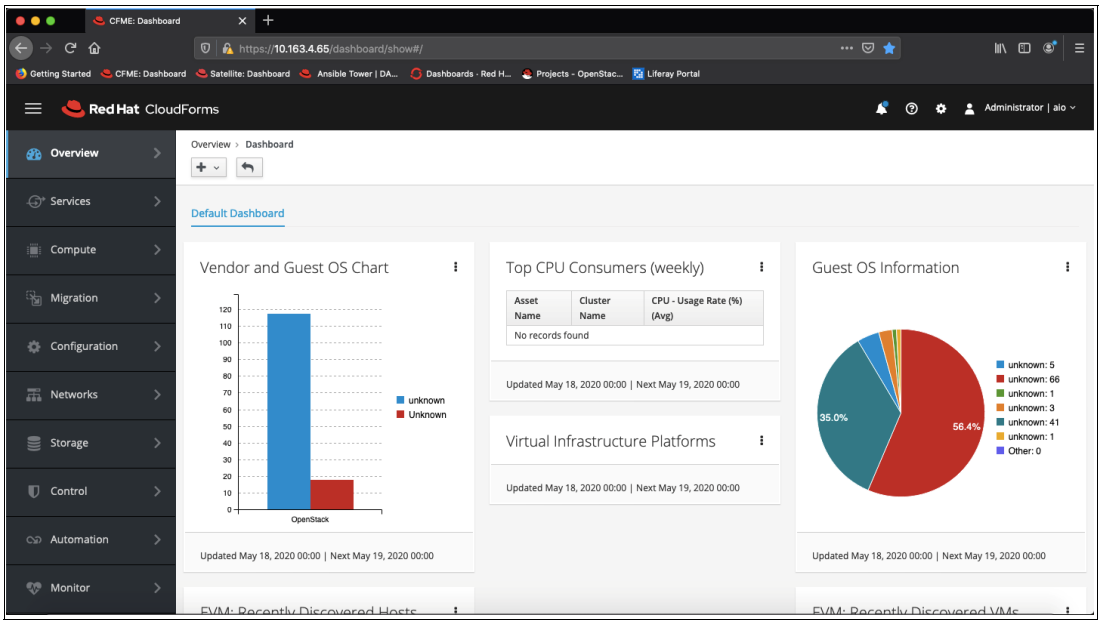


Figure 3-153 Red Hat CloudForms Dashboard pane

> **Note:** You can configure Red Hat CloudForms from the Tools option on the upper right corner of the page.

5. Select **Compute** → **Clouds** → **Providers** from the left control pane.
6. Click **Configuration** → **Add a New Cloud Provider**, as shown in Figure 3-154.

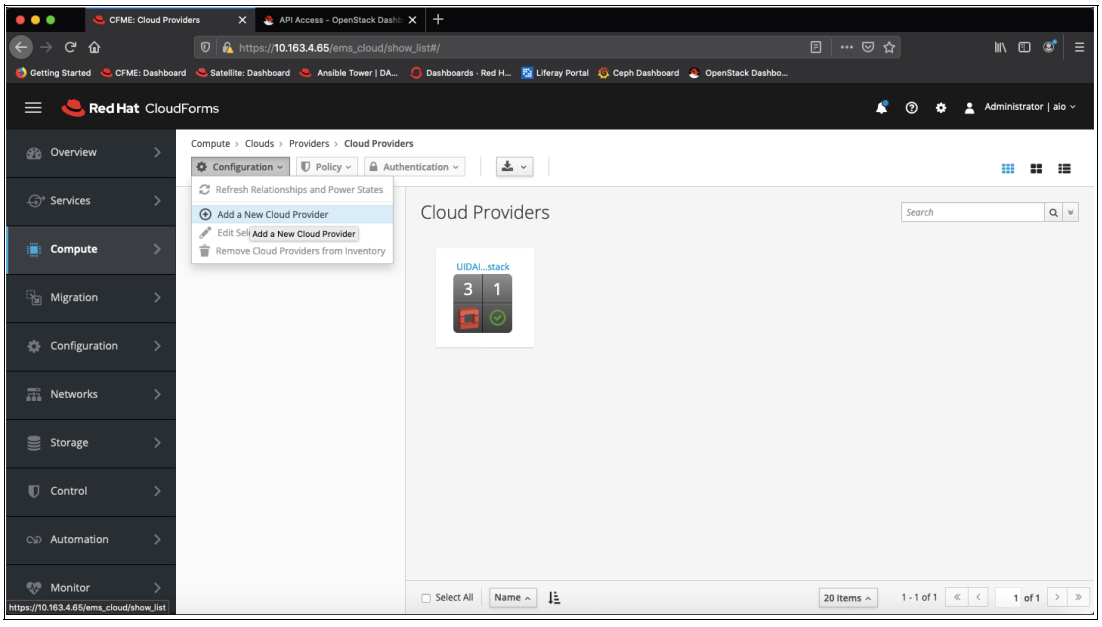


Figure 3-154 Red Hat CloudForms Configuration menu

7. In the Add a New Cloud Provider window, enter the name of the provider; then, select **Type** → **OpenStack**, as shown in Figure 3-155.

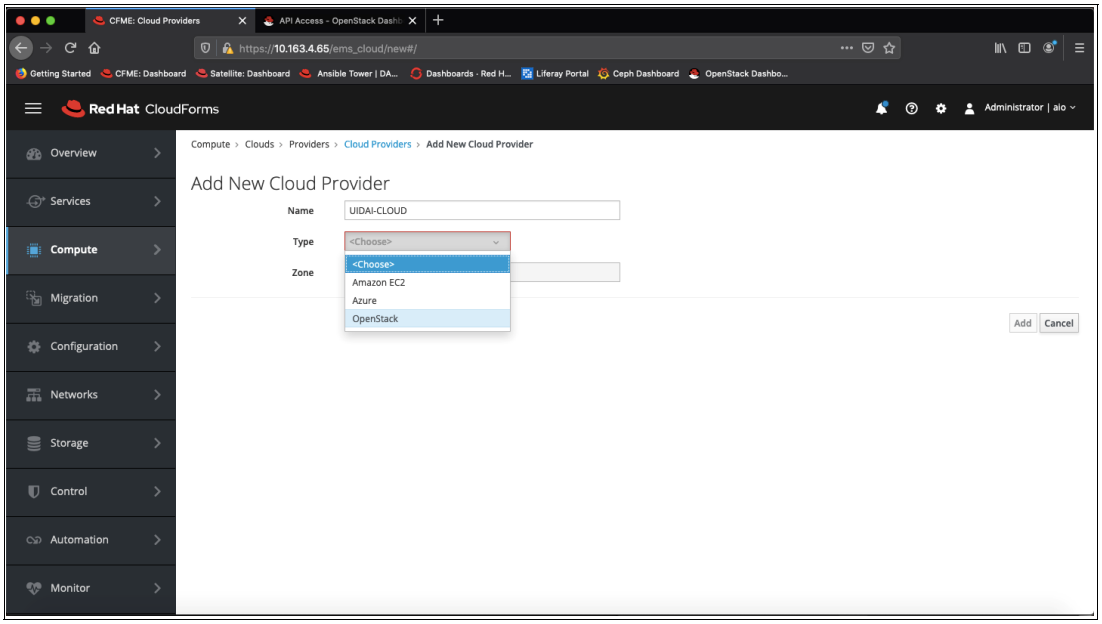


Figure 3-155 Red Hat CloudForms: Add New Cloud Provider

8. After you select **OpenStack** as the provider, the extended parameters are shown. Enter the IP and hostname information. Select **Security Protocol**, and enter the user name and password as shown in Figure 3-156.

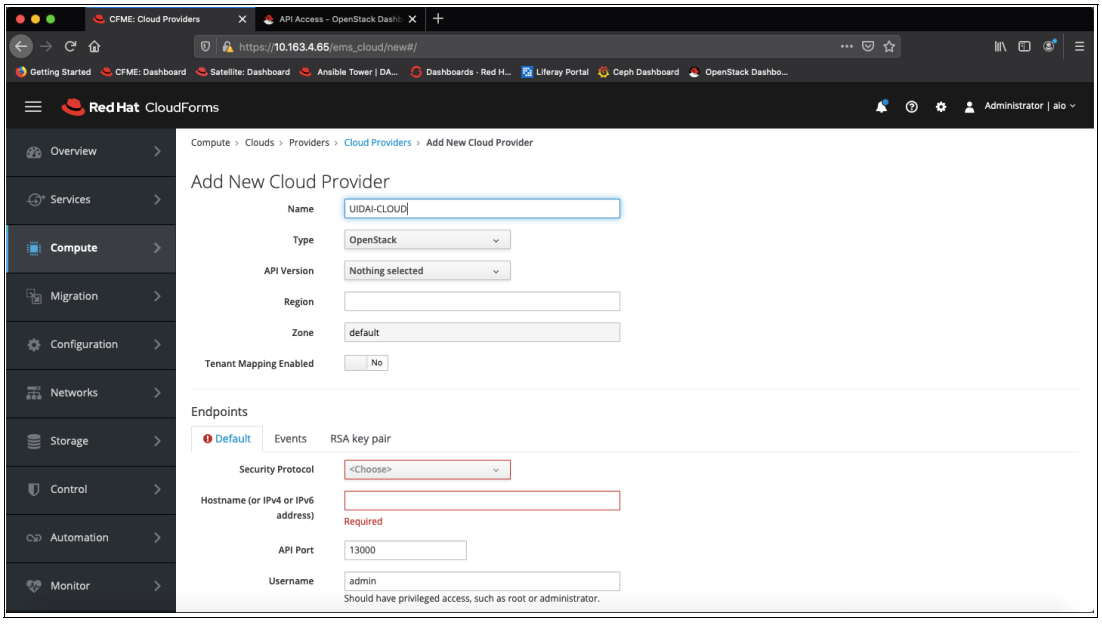


Figure 3-156 Red Hat CloudForms: Add New Cloud Provider continues

9. When all the required information is entered, click **Validate**. After the validation is successful, click **OK** to add the provider.

10. The provider is added. Click the added provider to view its objects (see Figure 3-157).

    You can click each object on the page to view the contents.

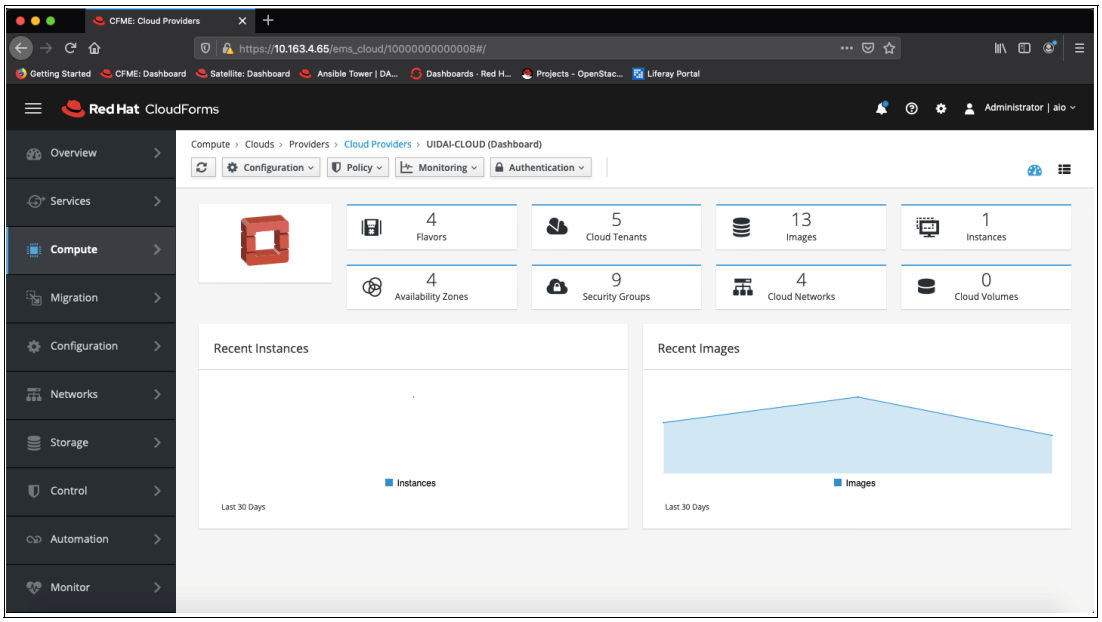


Figure 3-157 Red Hat CloudForms: Provider added pane

11. The cloud provider is now added and configured. If you want to include policy enforcement on the cloud provider, select **Policy**.

Stacks and topology view of deployment in Red Hat CloudForms

Complete the following steps:

1. Select **Compute** → **Clouds** → **Stacks** in the left control pane to view the orchestration stacks that were created for the added OpenStack provider (see Figure 3-158).

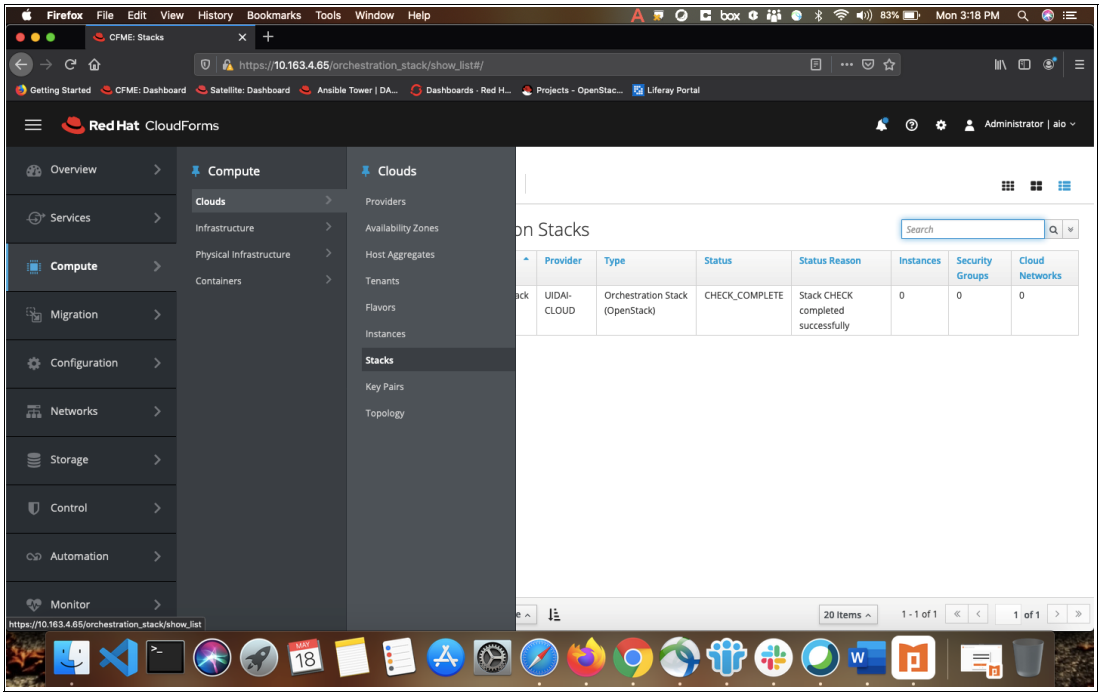


Figure 3-158 Red Hat CloudForms: Orchestration stacks

2. Click **Topology** to view the topology of the deployed Red Hat OpenStack Cloud, as shown in Figure 3-159.

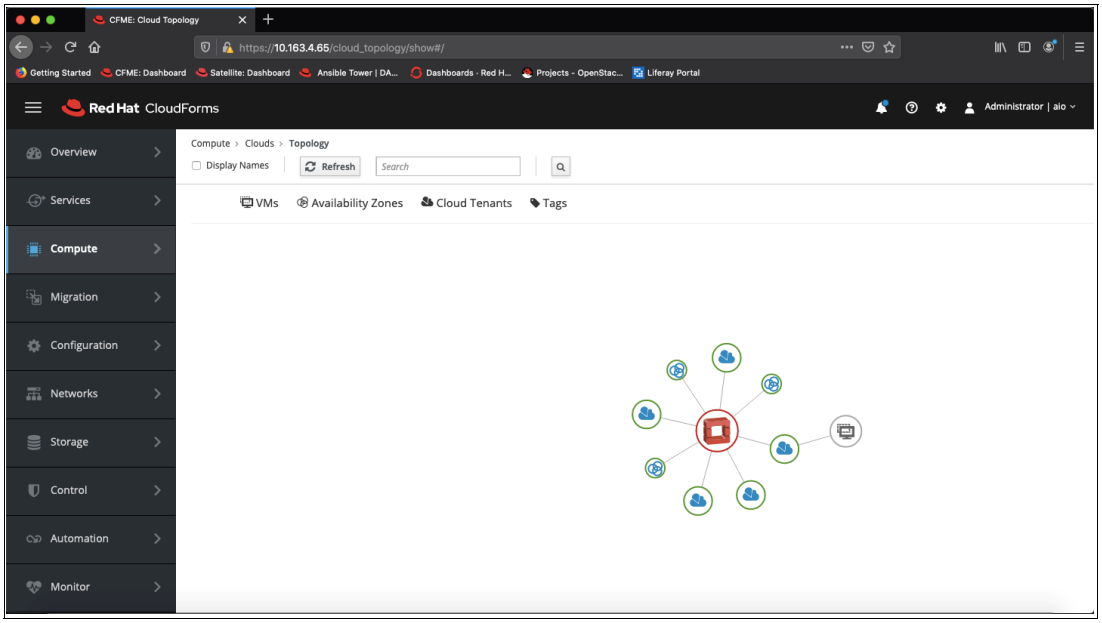


Figure 3-159 Red Hat CloudForms: Topology pane

Chargeback and reporting in Red Hat CloudForms

You can configure the chargeback rates and reporting for the cloud and infrastructure providers added in Red hat CloudForms as shown in Figure 3-160:

1. Select **Overview** → **Chargebacks** from the Left Control pane.
2. Click **Rates** → **Compute** → **Create / Use Default** for setting the chargeback rates. To configure reporting capabilities, select **Reports** on the same page and click **Configuration**.

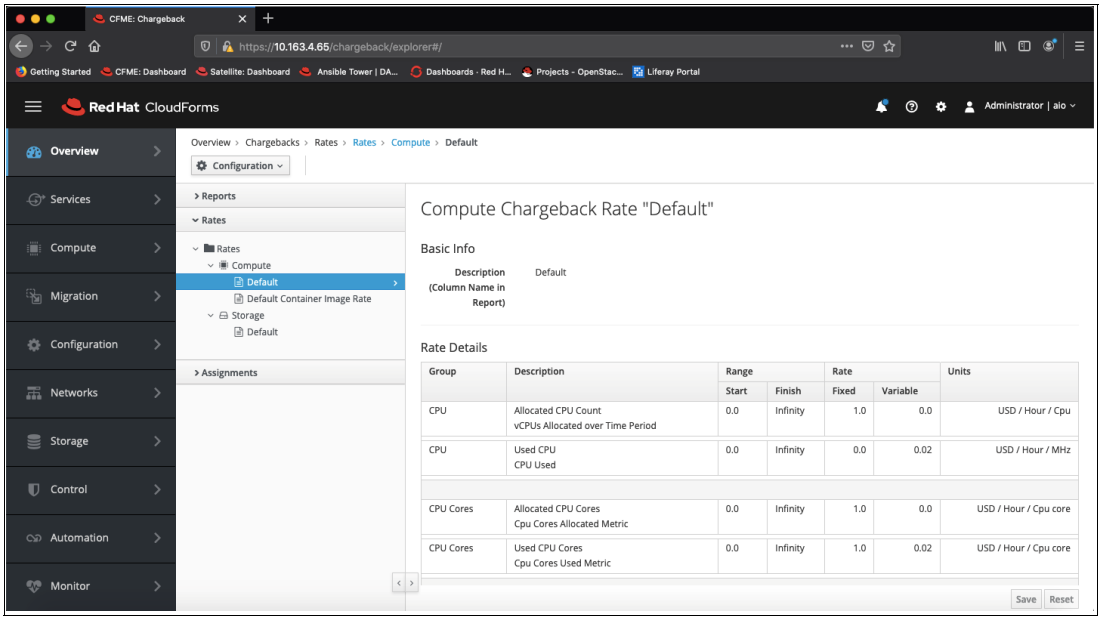


Figure 3-160 Red Hat CloudForms: Compute Chargeback Rate “Default” pane

3.8.4 Creating orchestration patterns in Red Hat CloudForms

This section describes the orchestration aspects of the Red Hat CloudForms of the hybrid cloud management platform and demonstrates the capability of orchestrating provisioning and deprovisioning on the cloud providers (Red Hat OpenStack, in this case). You can perform orchestration across multi-hybrid clouds, including Private and Public clouds from Red Hat CloudForms Management UI.

Complete the following steps:

1. Log in to Red Hat CloudForms Management UI and select **Services** → **Catalogs**. You are redirected to a page where all the service catalogs are listed. For example, we created a sample Heat Orchestration service catalog that is named `Hot_Sample`, as shown in Figure 3-161.

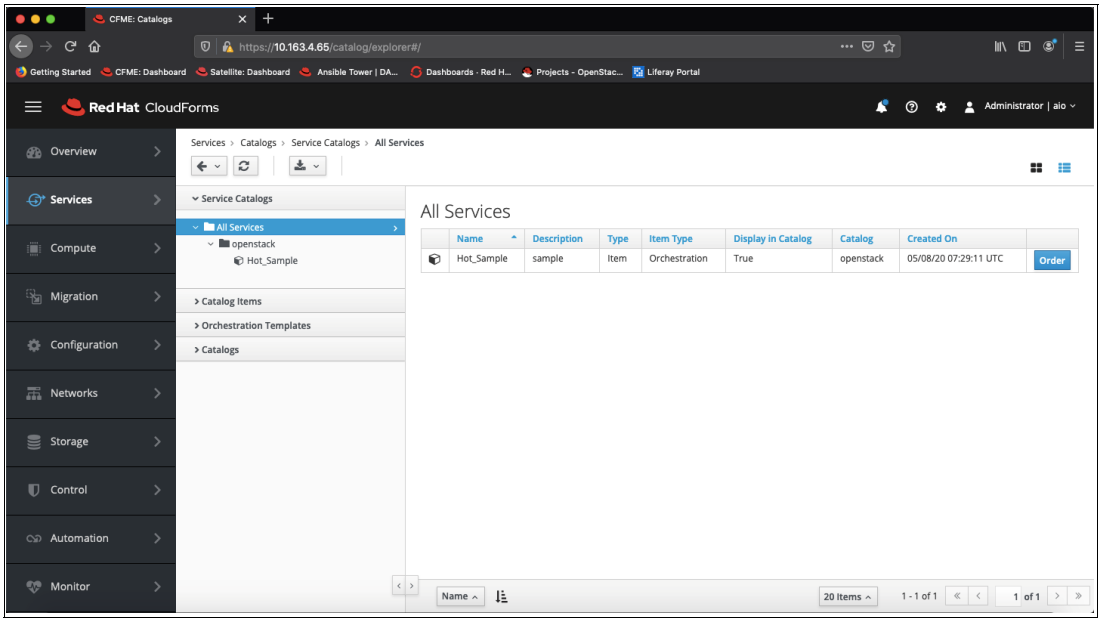


Figure 3-161 Red Hat CloudForms: All Services pane

2. To Provision the stack on the Red Hat OpenStack, click **Order** for single click provisioning.

3. Click **Catalog Item** to see the basic information about the service catalog that is named `Hot_Sample`, as shown in Figure 3-162.

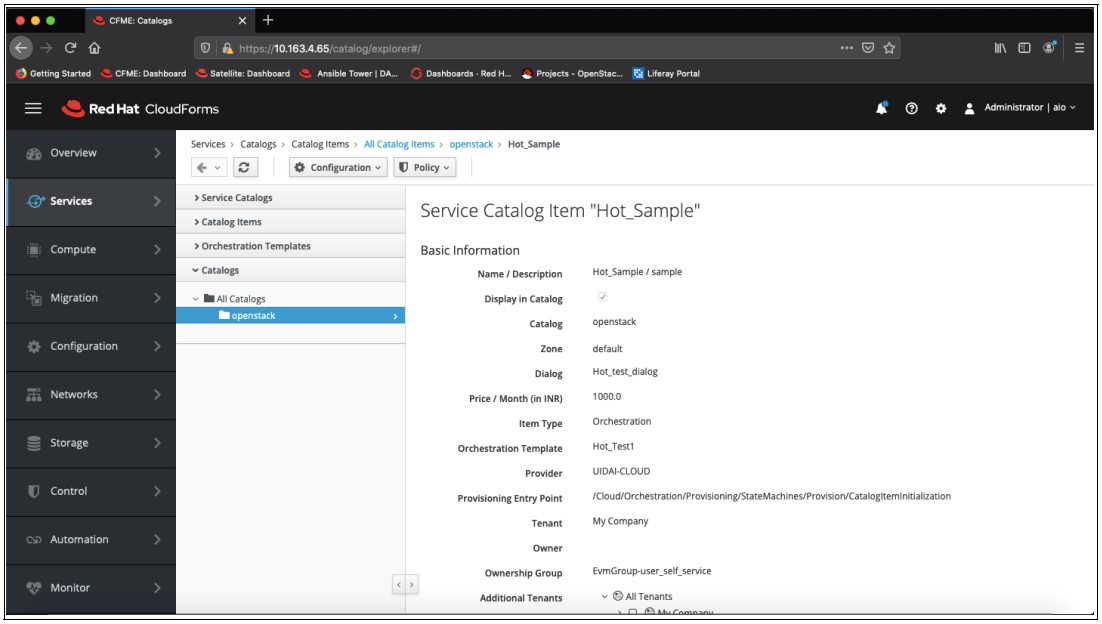


Figure 3-162 Red Hat CloudForms: Service Catalog Item “Hot Sample” pane

4. This Orchestration Self-Service Catalog also includes a heat template that is named `Hot_Test1`, which is used to perform the orchestration, as shown in Figure 3-162. To view the Template, select **All Orchestration Templates** → **Heat Templates** → **Hot_Test1**, as shown in Figure 3-163.

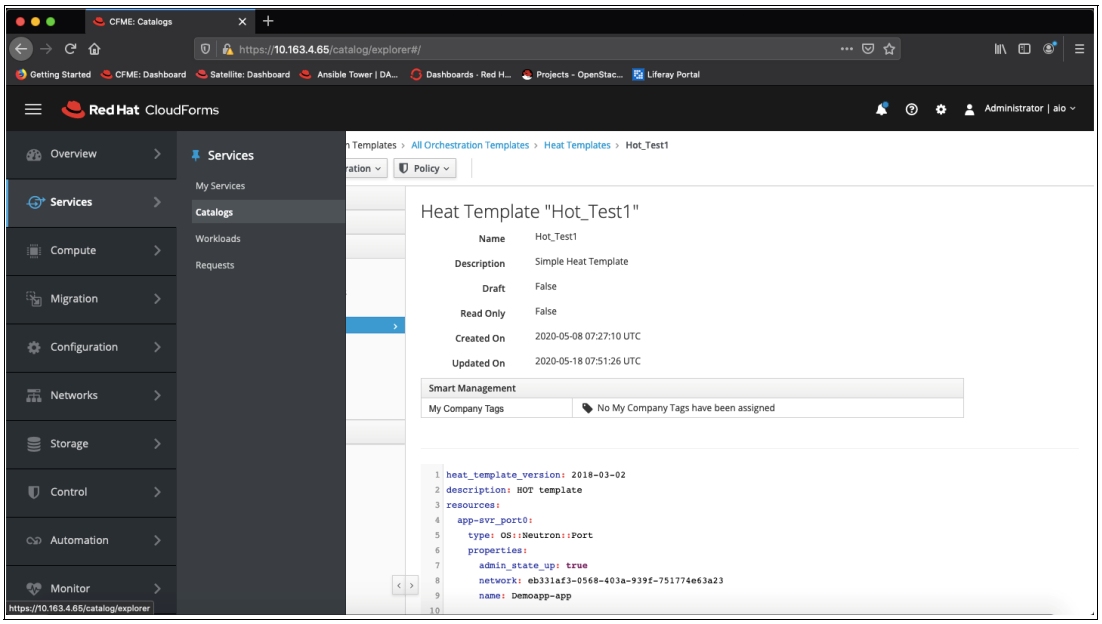


Figure 3-163 Red Hat CloudForms: Heat Template “Hot Test1”

5. You see the Heat Template source code. To edit the code, click **Settings** → **Configuration** → **Edit**.

4

Open source IT operations management

This chapter describes the solutions and tools that are deployed to help and maintain IT operations.

This chapter includes the topics:

4.1 Open source IT operations management overview

IT operations management is achieved through integrated open source tools that provide management of public, private, and hybrid cloud environments, as shown in Figure 4-1. This section discusses the minimum requirements for IT operations management, such as proactive monitoring with Prometheus, SLA dashboards in Grafana, log analysis and monitoring with ELK stack, IT Service Management with IT Operational Portal (iTop), and Unified visualization layer with Liferay portal.

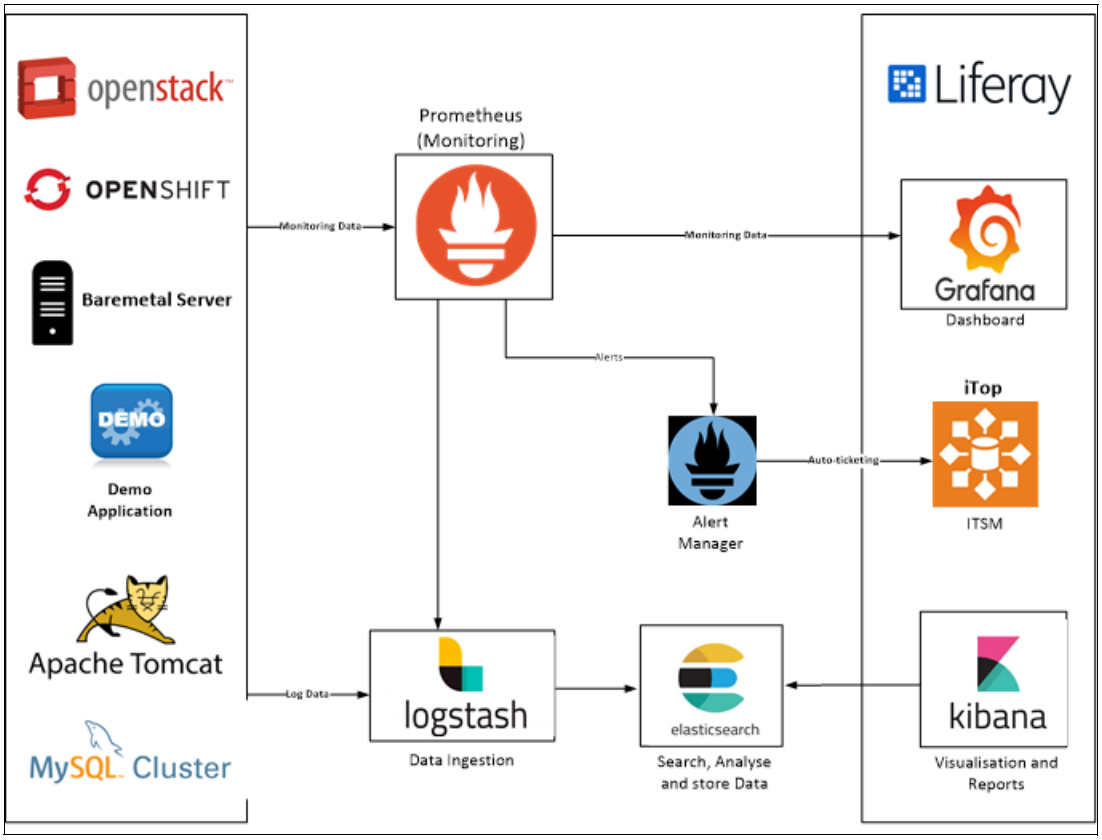


Figure 4-1 Cloud management-integrated solutions

IT operations require service management, such as incidents management, system monitoring, and log analysis as a key function and container platform and container image repository for agile project methodology and portability.

IT operations consist of a portal for integrated UI, service management for SLA and notification, Grafana for timeseries graphs, Prometheus for system collected data, and Kibana for analytics UI and analytics data.

IT operations use containers for prompt application development and easy management. Red Hat OpenShift is chosen as a most popular container platform.

The following tasks are considered to analyze open source before deploying applications on Red Hat OpenShift:

- Using containerized open source: Create an application that uses a container.

- Setting up storage configuration: A container that is required to store configuration `.yaml` files, file storage, and store the data in database. Red Hat OpenShift provides configMaps and PersistentVolumeClaim (PVC) to pod configuration.
- Managing updated containers and deploying a new application.

Quota setup, the number of replicas, autoscaling for containers, network, and security are required for a production environment.

Enterprise Management Systems (EMS) as shown in Figure 4-2 consists of a portal for the integrated UI, Service Management for SLA and notification, Grafana for timeseries graphs, Prometheus for system-collected data and Kibana for the analytics UI and analytics data.

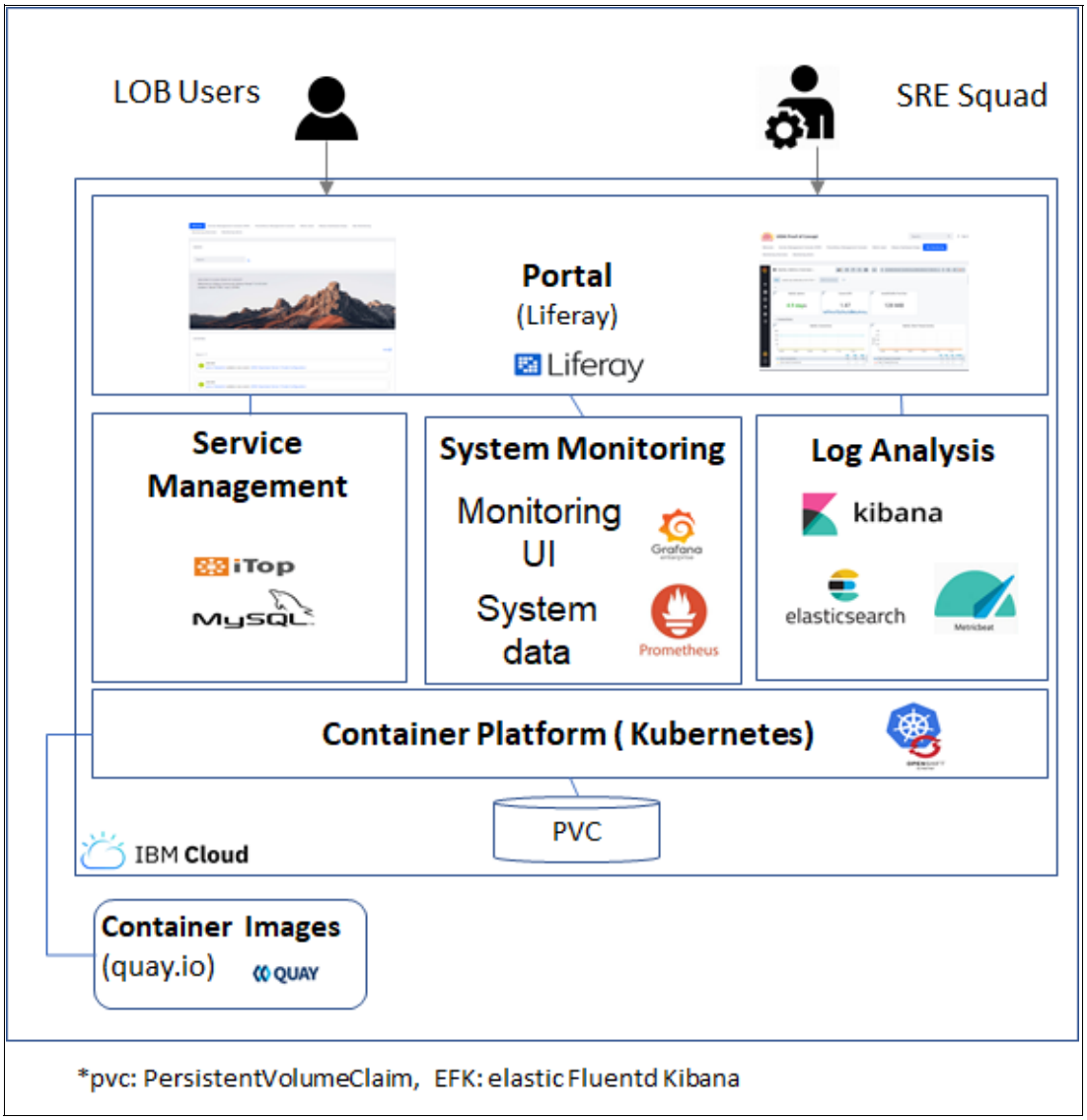


Figure 4-2 Enterprise Management Systems

EMS uses containerized applications for prompt application development and easy management. Deploying applications includes the following tasks:

- Using Containerized open source - Create an application that uses a container.

- Setting up storage configuration: A container that is required to store a configuration `.yaml` file, file storage, and store the data in database. Red Hat OpenShift provides configMaps and PersistentVolumeClaim (PVC) to pod configuration.
- Managing updated containers and deploying a new application.

4.2 Creating and managing applications on Red Hat OpenShift

This section explains how to create and manage applications on Red Hat OpenShift.

4.2.1 Creating an integrated portal by using Liferay

A new application uses Docker images from Docker Hub, sets up a configuration, pushes a customized image to Red Hat quay.io, and builds a Red Hat OpenShift container, as shown in Figure 4-3.

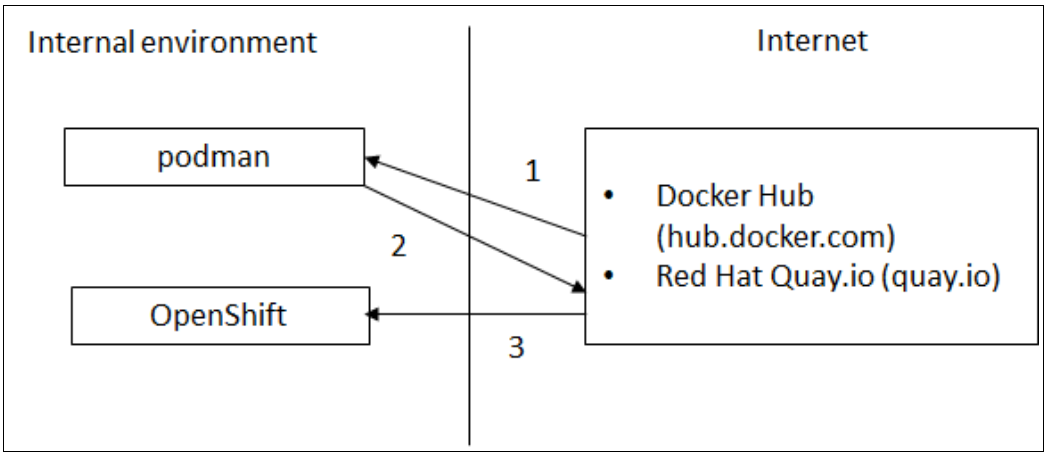


Figure 4-3 Sequence of new application creation

This process is designed for multiple application developers to configure and add features. Configurations and features are available for the portal, service management and analysis, user interface, and setting up Prometheus, exporters, fluentd, Elasticsearch, and Kibana (FLK).

Complete the following steps:

1. Create a portal by using the Liferay container.
2. Create an environment as follows:

```
$ podman run -it -p 8080:8080 liferay/portal:7.1.0-ga1-201809012030
$ podman ps

[root@bm2 ~]# podman ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
a509934a8439  docker.io/liferay/portal:7.1.0-ga1-201809012030  /bin/sh -c
/etc/l...  10 days ago  Up 7 seconds ago  0.0.0.0:8080->8080/tcp
nostalgic_mayer
```

3. Point your browser to `http://localhost:8080`, as shown in Figure 4-4.

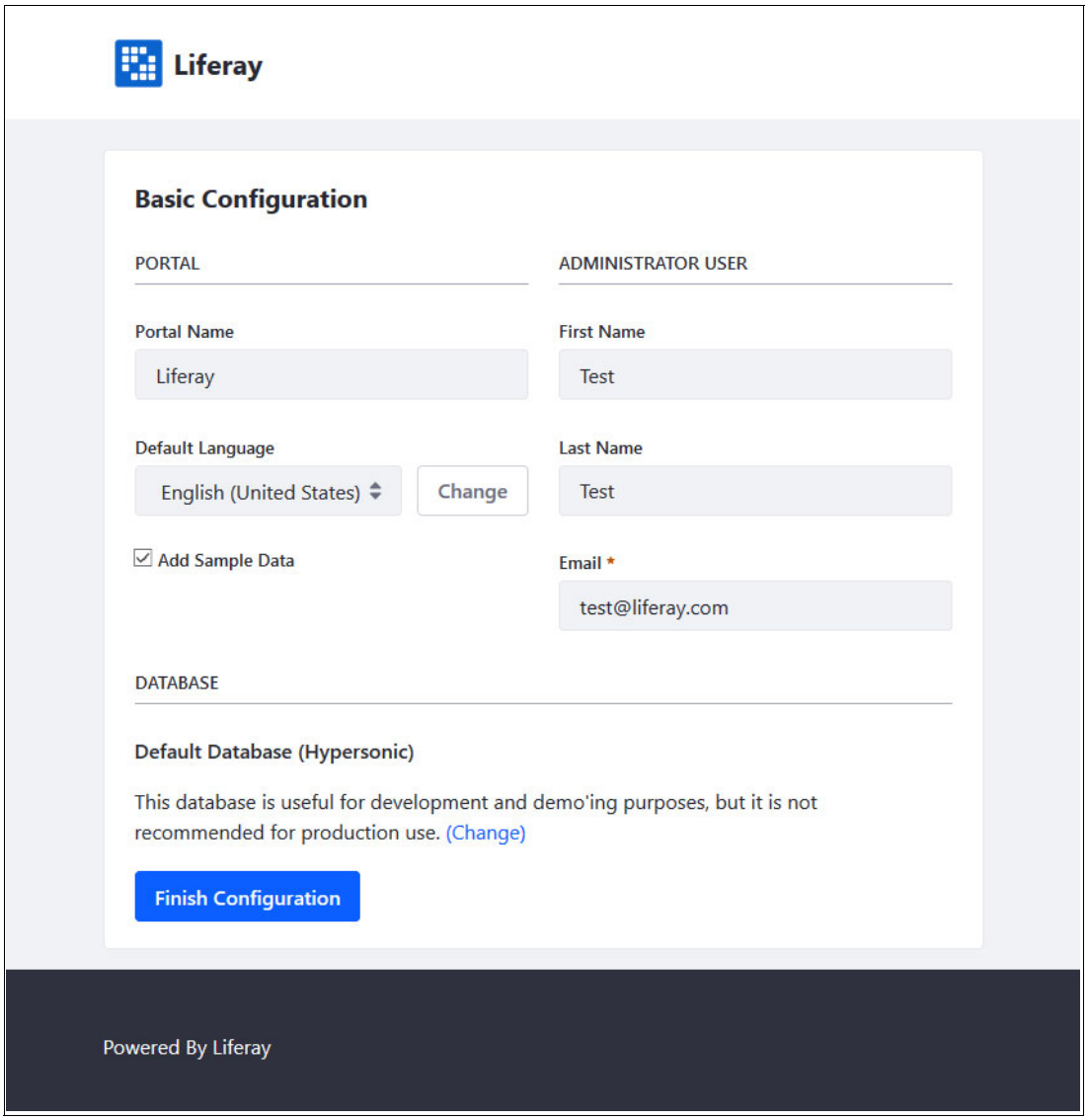


Figure 4-4 Liferay Basic Configuration window

After you set up the portal, complete the following steps:

1. Upload you customized container with your customized configuration, and then, commit and push the container (see Figure 4-5 and Figure 4-6):

```
$podman login quay.io
$podman commit container id quay.io/<quay-id>/<repository-name>
```

```
[root@codeReady ~]# podman commit bb24a6e3ec18 quay.io/doyoungim999/portal:test
Getting image source signatures
Copying blob 73046094a9b8 skipped: already exists
Copying blob 298c3bb2664f skipped: already exists
Copying blob 9335le248e6e skipped: already exists
Copying blob 951f0c24494c skipped: already exists
Copying blob f026b7ef5f63 skipped: already exists
Copying blob 846049cb1f22 skipped: already exists
Copying blob a137afled394 skipped: already exists
Copying blob 117c94579db6 skipped: already exists
Copying blob b22a0fee5724 done
Copying config d8977f1013 done
Writing manifest to image destination
Storing signatures
d8977f1013c21d3c4c450c07e0d0d478498416ac23ec680cbd16462745283434
```

Figure 4-5 Saving the container

```
$podman push quay.io/<quay-id>/<repository-name>:<tag>
```

```
[root@codeReady ~]# podman push quay.io/doyou      /portal:test
Getting image source signatures
Copying blob 73046094a9b8 done
Copying blob 951f0c24494c done
Copying blob 846049cb1f22 done
Copying blob f026b7ef5f63 done
Copying blob 298c3bb2664f done
Copying blob 9335le248e6e done
Copying blob 117c94579db6 done
Copying blob b22a0fee5724 done
Copying blob a137afled394 done
Copying config d8977f1013 done
Writing manifest to image destination
Copying config d8977f1013 done
Writing manifest to image destination
Storing signatures
```

Figure 4-6 Push the container

2. Access `quay.io` by using your quay user name and password, as shown in Figure 4-7.

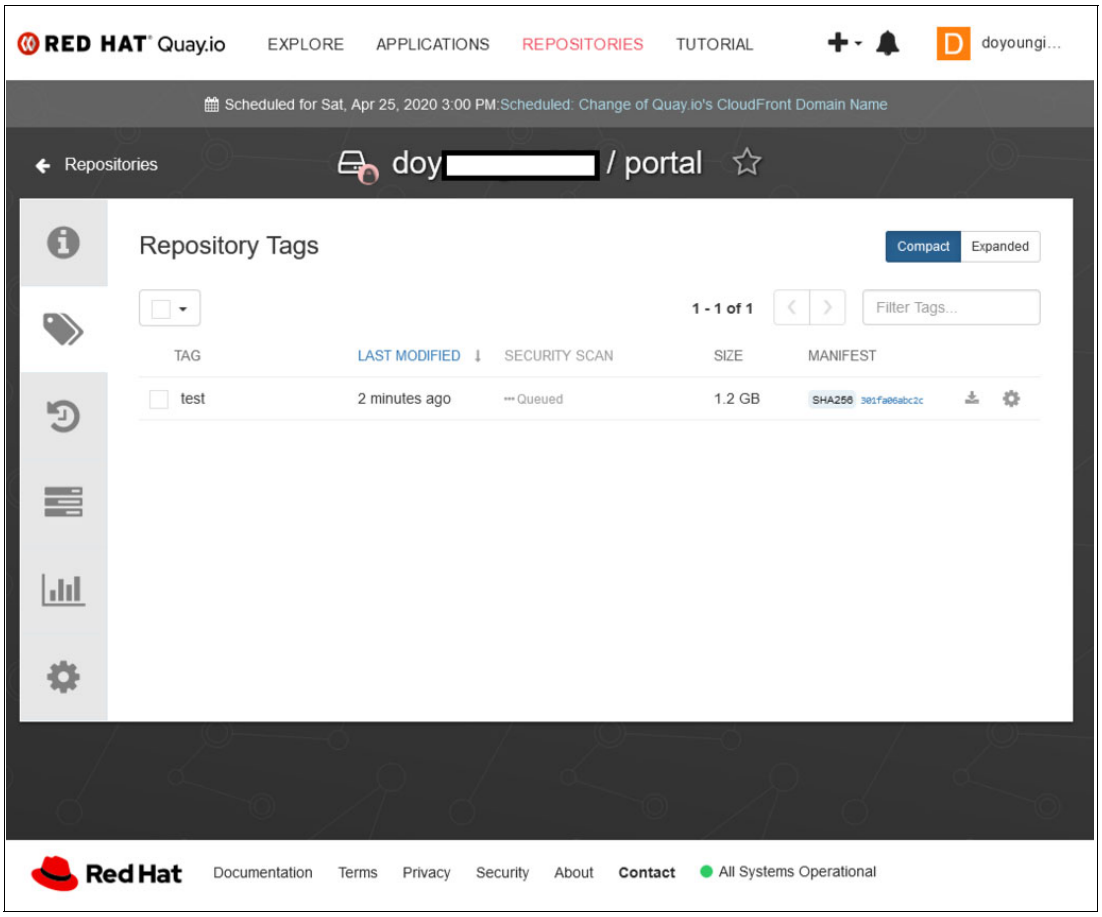


Figure 4-7 Quay.io window: Repository tags

3. Create a `new-app` to Red Hat OpenShift with customized container:

```
$ oc new-app portal  --name=portal  quay.io/manasmohsin/liferay:1
```

4. Create a service account to run this container:

```
$oc create serviceaccount useroot
```

5. Assign authority to the service account:

```
$oc adm policy add-scc-to-user anyuid -z useroot
```

6. Run the container as service account user root:

```
$oc edit dc/portal
```

7. Add serviceAccountName user root, as shown in Figure 4-8.

```
    spec:
      containers:
      - image: quay.io/manasmohsin/liferay@sha256:5c1b0c1197d08f8fb94e403f3fb382
b98707071cf20de5f71d50aa6d3fadba83
        imagePullPolicy: Always
        name: portal
        ports:
        - containerPort: 11311
          protocol: TCP
        - containerPort: 8080
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccountName: useroot
      terminationGracePeriodSeconds: 30
```

Figure 4-8 Add service account name user root

8. Set up the portal for access (see Figure 4-9):

```
$oc expose svc/portal
$oc get route
```

```
[user2@codeReady .vnc]$ oc get route
NAME     HOST/PORT                          PATH   SERVICES   PORT       TERMI
NATION   WILDCARD
portal   portal-emsproject.apps-crc.testing         portal     8080-tcp
         None
```

Figure 4-9 Route to access the portal

Browse the `portal-empproject.apps-crc.testing`, as shown in Figure 4-10.

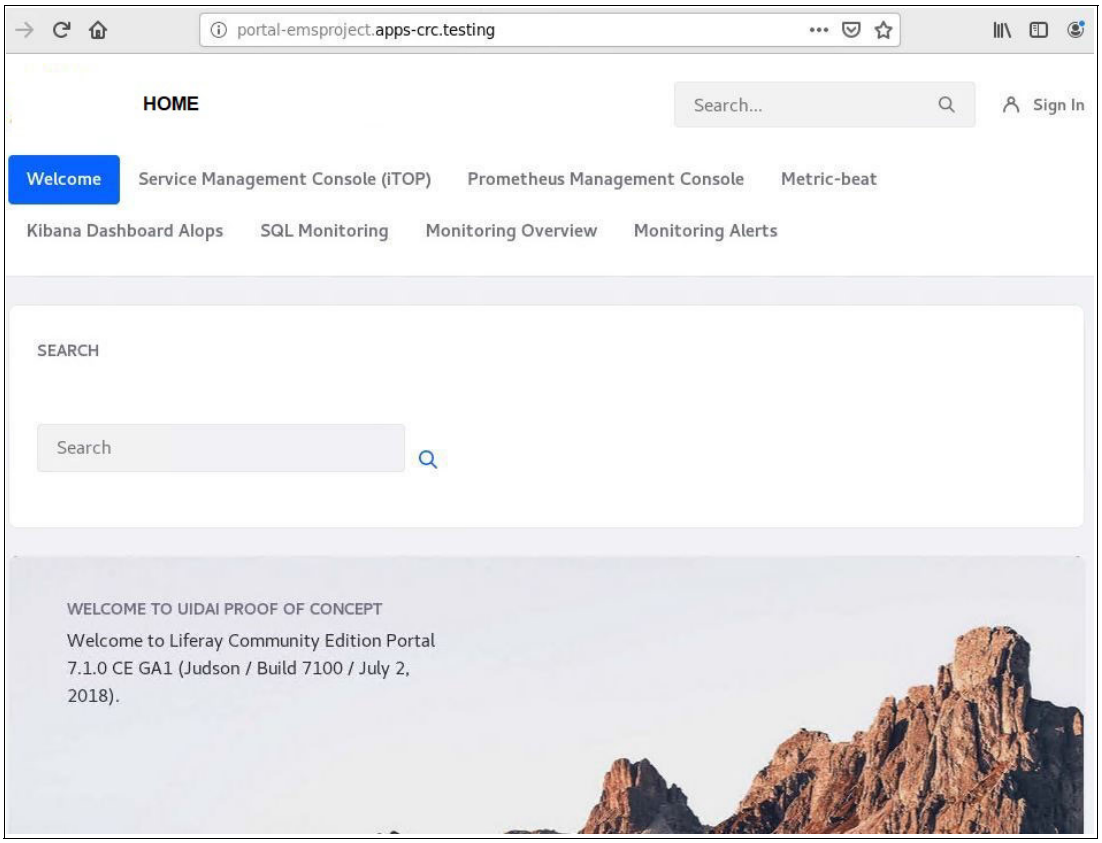


Figure 4-10 Portal window

4.2.2 Creating a Prometheus in Red Hat OpenShift

To create a Prometheus in Red Hat OpenShift, build a Prometheus container image by using the configuration files, as shown in Figure 4-11. You also can deploy a standard image from Docker Hub and add the configuration file to Red Hat OpenShift.

Before deploying a new application, the application must be analyzed to use Kubernetes features.

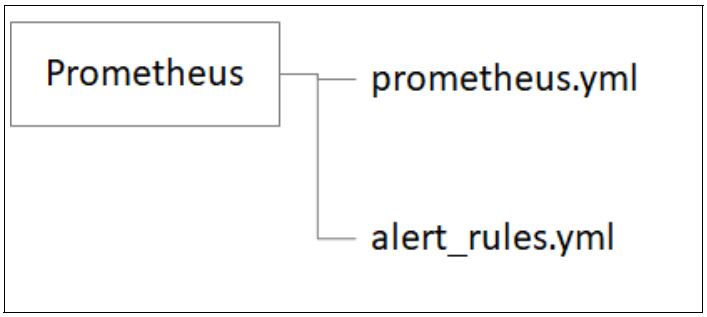


Figure 4-11 Prometheus configuration structure for this book

Create a Prometheus by using the command from Prometheus official image:

```
$oc new-app docker.io/prom/prometheus:latest
```

Red Hat OpenShift provides configmaps for the configuration files. Two files are used (`prometheus.yml` and `alert_rules.yml`) under the prometheus directory:

```
[user2@server prometheus]$ ll
total 8
-rwxr-xr-x. 1 user2 user2 2668 Apr 15 22:29 alert_rules.yml
-rwxr-xr-x. 1 user2 user2 3451 Apr 15 20:34 prometheus.yml

$oc create configmap prometheusdir-config --from-file=prometheus/
$ oc get configmap/prometheusdir-config

[user2@server ~]$ oc get configmap/prometheusdir-config
NAME                    DATA   AGE
prometheusdir-config    3      19h
```

List the Prometheus configuration by running the following command (see Example 4-1):

```
$oc describe configmap/prometheusdir-config
```

Example 4-1 List the configuration map for Prometheus

```
[user2@server prom]$ oc describe configmap/prometheusdir-config
Name:         prometheusdir-config
Namespace:    emsproject
Labels:       <none>
Annotations:  <none>

Data
====
alert_rules.yml:
----
groups:
  - name: example
    rules:
    - alert: InstanceDown2
      expr: up == 0
      for: 1m
      labels:
        severity: 1
        priority: 1
        summary: "Instance {{ $labels.instance }} down"
        description: "{{ $labels.instance }} of job {{ $labels.job }} has been
down for more than 5 minutes."
    - alert: NodeDiskReadCount
      expr: node_disk_reads_completed_total > 9134000
      for: 1m
      labels:
        severity: 4
        priority: 4
        summary: "Instance {{ $labels.instance }} has Node Disk Read Count
crossing threshold Issue"
        description: "{{ $labels.instance }} of job {{ $labels.job }} has Node
Disk Read Count > 1134000"
    - alert: NodeFileSystemFree
      expr: node_filesystem_files_free > 9043381
      for: 1m
      labels:
```

```
        severity: 2
        priority: 2
        summary: "Instance {{ $labels.instance }} has Node File System Free
crossing threshold Issue"
        description: "{{ $labels.instance }} of job {{ $labels.job }} has Node
File System Free > 1043381"
    - alert: NodeCPUSecondTotal
      expr: node_cpu_seconds_total{mode="iowait"} > 940095
      for: 1m
      labels:
        severity: 1
        priority: 1
        summary: "Instance {{ $labels.instance }} has Node CPU Second Total
crossing threshold Issue"
        description: "{{ $labels.instance }} of job {{ $labels.job }} has Node CPU
Second Total > 840095."
    - alert: HostHighCpuLoad
      expr: avg by(instance)(irate(node_cpu_seconds_total {mode="user"} [1m]) *
100) > 95
      for: 1m
      labels:
        severity: 1
        priority: 1
        summary: "Host high CPU load {{ $labels.instance }}"
        description: "CPU load is > 95%"
    - alert: HostHighCpuLoad
      expr: avg by(instance)(irate(node_cpu_seconds_total {mode="user"} [1m]) *
100) > 70
      for: 1m
      labels:
        severity: 2
        priority: 2
        summary: "Host high CPU load {{ $labels.instance }}"
        description: "CPU load is > 70%"
    - alert: HostOutOfMemory
      expr: node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes * 100 < 35
      for: 5m
      labels:
        severity: 3
        priority: 3
        summary: "Host out of memory {{ $labels.instance }}"
        description: "Node memory is filling up ...only 20% left)."
    - alert: HostOutOfDiskSpace
      expr: (node_filesystem_avail_bytes {device="tmpfs",mountpoint="/run"}  *
100) / node_filesystem_size_bytes{device="tmpfs",mountpoint="/run"} > 91
      for: 5m
      labels:
        severity: 1
        priority: 1
        summary: "Host out of disk space {{ $labels.instance }}"
        description: "Disk is almost full..only 9% left"

prometheus.yml:
----
# my global config
```

```
global:
  scrape_interval:     20s # Set the scrape interval to every 15 seconds. Default
is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every
1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    #path_prefix: /alertmanager
    - targets:
      - prom.redbooks.info:9093
      #- alertmanager:9093
      #- 169.38.67.66:9093

# Load rules once and periodically evaluate them according to the global
'evaluation_interval'.
rule_files:
  - 'alert_rules.yml'
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped
from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['prom.redbooks.info:9090']

  - job_name: 'node_exporter'
    static_configs:
    - targets: ['prom.redbooks.info:9100']
    - targets: ['134.209.22.37:9100']
    - targets: ['169.38.131.87:9100']
    - targets: ['tomcat.redbooks.info:9100']
    - targets: ['node1-sql.redbooks.info:9100']
    - targets: ['node2-sql.redbooks.info:9100']
    relabel_configs:
    - source_labels: [__address__]
      regex:  'prom.redbooks.info:9100'
      target_label: instance
      replacement: 'prometheus_server'
    - source_labels: [__address__]
      regex:  '134.209.22.37:9100'
      target_label: instance
      replacement: 'external_server'
    - source_labels: [__address__]
```

```
    regex:  '169.38.131.87:9100'
    target_label: instance
    replacement: 'elk_server'
  - source_labels: [__address__]
    regex:  'tomcat.redbooks.info:9100'
    target_label: instance
    replacement: 'tomcat_server'
  - source_labels: [__address__]
    regex:  'node1-sql.redbooks.info:9100'
    target_label: instance
    replacement: 'node1-sql_server'
  - source_labels: [__address__]
    regex:  'node2-sql.redbooks.info:9100'
    target_label: instance
    replacement: 'node2-sql_server'

- job_name: 'mysqld_exporter'
  static_configs:
  - targets: ['node1-sql.redbooks.info:9104']
  - targets: ['node2-sql.redbooks.info:9104']
  - targets: ['node1-sql.redbooks.info:9144']

- job_name: 'apache'
  static_configs:
  - targets: ['tomcat.redbooks.info:9117']
- job_name: 'java'
  static_configs:
  - targets: ['tomcat.redbooks.info:8181']
  relabel_configs:
  - source_labels: [__address__]
    regex: 'tomcat.redbooks.info:8181'
    target_label: instance
    replacement: 'tomcat'
- job_name: 'ipmi_exporter'
  scrape_interval: 1m
  scrape_timeout: 30s
  metrics_path: /ipmi
  scheme: http
  file_sd_configs:
  - files:
    - /root/prometheus-2.16.0.linux-amd64/targets.yml
    refresh_interval: 5m
  relabel_configs:
  - source_labels: [__address__]
    separator: ;
    regex: (.*)
    target_label: __param_target
    replacement: ${1}
    action: replace
  - source_labels: [__param_target]
    separator: ;
    regex: (.*)
    target_label: instance
    replacement: ${1}
    action: replace
```

```
    - separator: ;
      regex: .*
      target_label: __address__
      replacement: ipmi-exp.redbooks.info:9290
      action: replace


Events:  <none>
```

Set up configmap in a container:

```
$oc set volume dc/prometheus --add --mount-path=/etc/prometheus
--name=prometheusdir-volume-1 -t configmap --configmap-name=prometheusdir-config
```

To see the pod configuration, run the following command as shown in Example 4-2:

```
$oc get dc/prometheus -o yaml
```

Example 4-2 Shows the Pod configuration

```
[user2@server ~]$ oc get dc/prometheus -o yaml
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  annotations:
    openshift.io/generated-by: OpenShiftNewApp
  creationTimestamp: "2020-04-17T00:42:40Z"
  generation: 3
  labels:
    app: prometheus
    app.kubernetes.io/component: prometheus
    app.kubernetes.io/instance: prometheus
  name: prometheus
  namespace: emsproject
  resourceVersion: "1168910"
  selfLink:
/apis/apps.openshift.io/v1/namespaces/emsproject/deploymentconfigs/prometheus
  uid: 3f738a00-5551-4951-ae4b-29033a7cdd14
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    deploymentconfig: prometheus
  strategy:
    activeDeadlineSeconds: 21600
    resources: {}
    rollingParams:
      intervalSeconds: 1
      maxSurge: 25%
      maxUnavailable: 25%
      timeoutSeconds: 600
      updatePeriodSeconds: 1
    type: Rolling
  template:
    metadata:
      annotations:
        openshift.io/generated-by: OpenShiftNewApp
```

```
      creationTimestamp: null
      labels:
        deploymentconfig: prometheus
    spec:
      containers:
      - image:
docker.io/prom/prometheus@sha256:0ec13a50dfa65ec1cf8583ef640271e14392534f1d2c21f75
8eb0e0e441f7540
        imagePullPolicy: Always
        name: prometheus
        ports:
        - containerPort: 9090
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /prometheus
          name: prometheus-volume-1
        - mountPath: /etc/prometheus
          name: prometheusdir-volume-1
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
      - emptyDir: {}
        name: prometheus-volume-1
      - configMap:
          defaultMode: 420
          name: prometheusdir-config
        name: prometheusdir-volume-1
  test: false
  triggers:
  - type: ConfigChange
  - imageChangeParams:
      automatic: true
      containerNames:
      - prometheus
      from:
        kind: ImageStreamTag
        name: prometheus:latest
        namespace: emsproject
      lastTriggeredImage:
docker.io/prom/prometheus@sha256:0ec13a50dfa65ec1cf8583ef640271e14392534f1d2c21f75
8eb0e0e441f7540
    type: ImageChange
status:
  availableReplicas: 1
  conditions:
  - lastTransitionTime: "2020-04-17T00:42:53Z"
    lastUpdateTime: "2020-04-17T00:42:53Z"
    message: Deployment config has minimum availability.
    status: "True"
```

```
    type: Available
  - lastTransitionTime: "2020-04-17T00:55:18Z"
    lastUpdateTime: "2020-04-17T00:55:21Z"
    message: replication controller "prometheus-2" successfully rolled out
    reason: NewReplicationControllerAvailable
    status: "True"
    type: Progressing
  details:
    causes:
    - type: ConfigChange
    message: config change
  latestVersion: 2
  observedGeneration: 3
  readyReplicas: 1
  replicas: 1
  unavailableReplicas: 0
  updatedReplicas: 1
```

To see configMap from the Red Hat OpenShift console, click **Workloads** → **Config Maps** → **prometheusdir-config**, as shown in Figure 4-12.

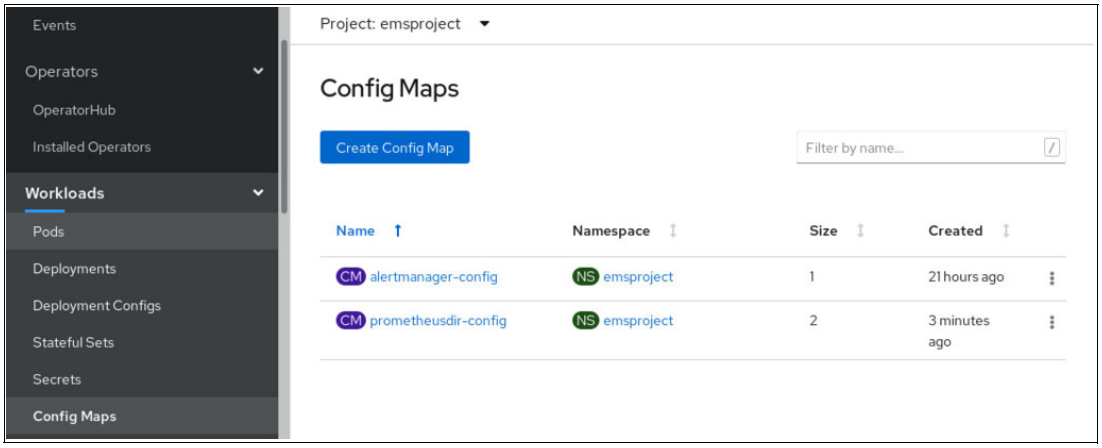


Figure 4-12 Config Maps window

To make the service available by using a URL, run the following command:

```
$oc expose svc/prometheus
```

To enable a URL to browser the service, run the following command:

```
$oc get route

[user2@server ~]$ oc get route/prometheus
NAME         HOST/PORT                                PATH   SERVICES
PORT       TERMINATION   WILDCARD
prometheus   prometheus-emsproject.apps-crc.testing          prometheus
9090-tcp                 None
```

Browse the URL to see the `prometheus.yml` configuration, as shown in Figure 4-13.

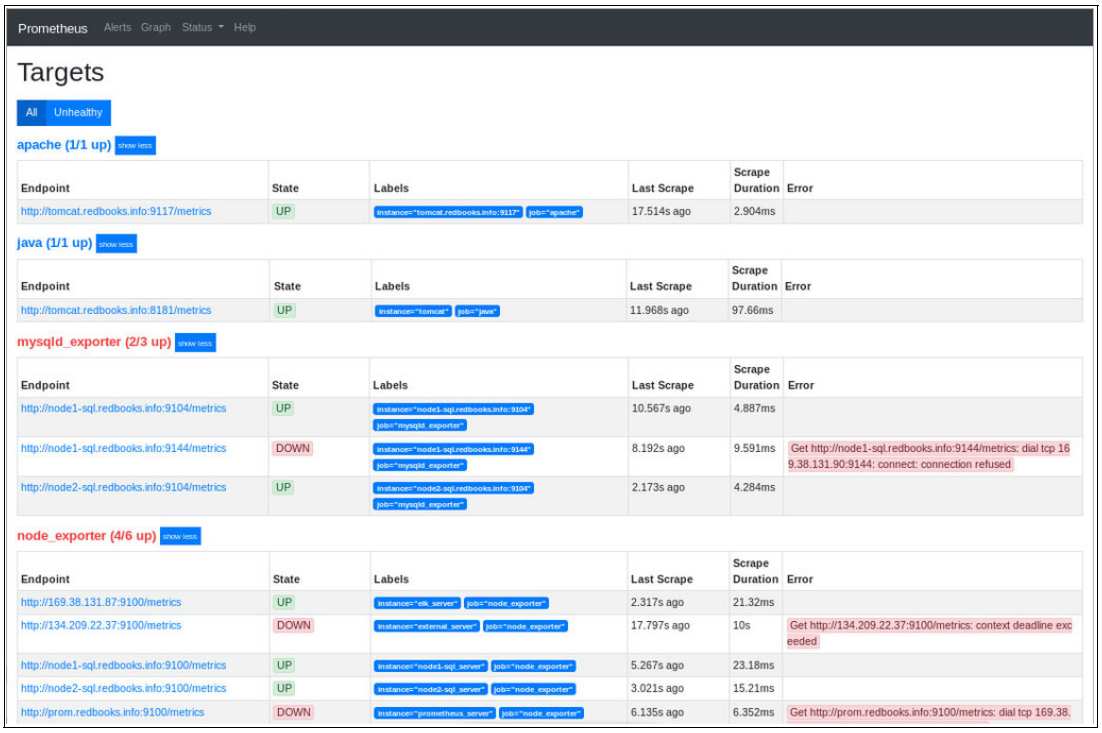


Figure 4-13 Prometheus Targets configuration window

Browse to see the `alert_rules.yml` configuration, as shown in Figure 4-14.

Figure 4-14 Prometheus Alerts window

4.2.3 Creating a Grafana in Red Hat OpenShift

To create a Grafana, the environment parameters and storage configuration that is shown in Figure 4-15 are required.

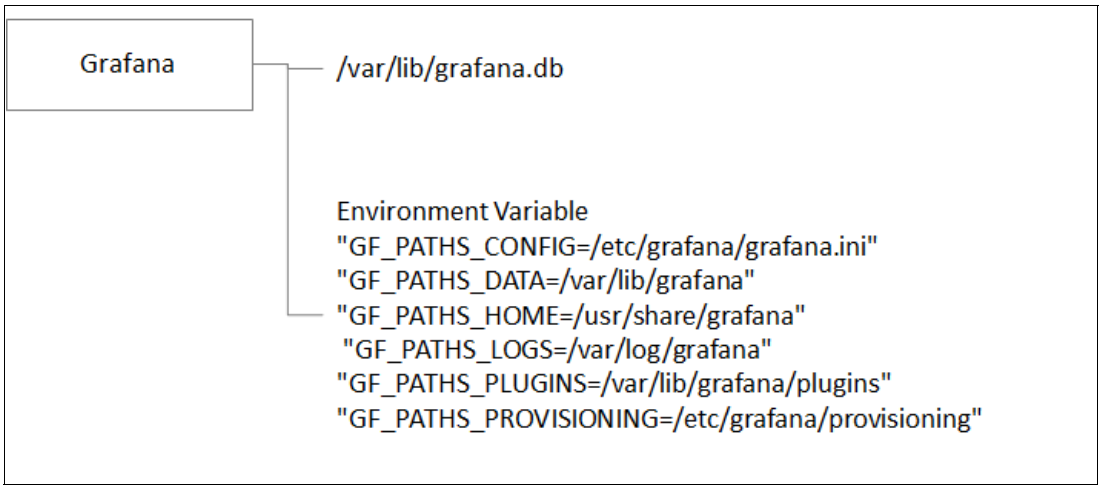


Figure 4-15 Grafana environment parameters

Complete the following steps:

1. Import the image, as shown in Example 4-3:

```
$ oc import-image  docker.io/grafana/grafana:latest --confirm
```

Example 4-3 Import image

```
[user2@codeReady mysql]$ oc import-image  docker.io/grafana/grafana:latest
--confirm
imagestream.image.openshift.io/grafana imported

Name:                   grafana
Namespace:              emsproject
Created:                Less than a second ago
Labels:                 <none>
Annotations:
openshift.io/image.dockerRepositoryCheck=2020-05-13T01:43:07Z
Image Repository:
image-registry.openshift-image-registry.svc:5000/emsproject/grafana
Image Lookup:           local=false
Unique Images:          1
Tags:                   1

latest
  tagged from docker.io/grafana/grafana:latest

  *
docker.io/grafana/grafana@sha256:44e41d00685a13783555fa1e8efc2a5e5e70e1fa16db55230
35cbedca8bd7eb7
      Less than a second ago

Image Name:     grafana:latest
```

```
Docker Image:
docker.io/grafana/grafana@sha256:44e41d00685a13783555fa1e8efc2a5e5e70e1fa16db55230
35cbedca8bd7eb7
Name:
sha256:44e41d00685a13783555fa1e8efc2a5e5e70e1fa16db5523035cbedca8bd7eb7
Created:        Less than a second ago
Annotations:    image.openshift.io/dockerLayersOrder=ascending
Image Size:     88.5MB in 8 layers
Layers:         2.787MB
sha256:4167d3e149762ea326c26fc2fd4e36fdeb7d4e639408ad30f37b8f25ac285a98
                175B
sha256:dd2bf2ad25d9d32fef2c99adb7ffd54d17080f96ad545b1bfd42f6f8c5c7c917
                3.79MB
sha256:bc3026833a3ad08f8733ebf2e57e7af5355403b99324ab3023dd4efa4cb915d0
                8.001MB
sha256:8789bc1f4250fc3b69bd532c4bf0279661e69f64433c517b224caf09a31024fb
                8.645MB
sha256:fc931efc1e7156b41bad35b7c72f556ce588bdbbb9b613584ae1208d77417f44
                65.26MB
sha256:09de0f0f5c9117c85fdbf7f26bdfb5ccebc4d6c5910fd344dde5475dde13747b
                10.42kB
sha256:b9833e14e8a252ae554575daaa553755ef239ebe6e5b6a8d74bf70a0aa3cc38a
                1.249kB
sha256:763366917f499837db1ef79ff416a87b3716623a9072cda4b429d5abe497a377
Image Created:  2 weeks ago
Author:         <none>
Arch:           amd64
Entrypoint:     /run.sh
Working Dir:    /usr/share/grafana
User:           grafana
Exposes Ports:  3000/tcp
Docker Labels:  <none>
Environment:
PATH=/usr/share/grafana/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbi
n:/bin
                GF_PATHS_CONFIG=/etc/grafana/grafana.ini
                GF_PATHS_DATA=/var/lib/grafana
                GF_PATHS_HOME=/usr/share/grafana
                GF_PATHS_LOGS=/var/log/grafana
                GF_PATHS_PLUGINS=/var/lib/grafana/plugins
                GF_PATHS_PROVISIONING=/etc/grafana/provisioning
```

2. Create a Grafana in Red Hat OpenShift by running the following command:

```
$oc new-app docker.io/grafana/grafana:latest \
 -e "GF_PATHS_CONFIG=/etc/grafana/grafana.ini" \
 -e "GF_PATHS_DATA=/var/lib/grafana" \
 -e "GF_PATHS_HOME=/usr/share/grafana" \
 -e "GF_PATHS_LOGS=/var/log/grafana" \
 -e "GF_PATHS_PLUGINS=/var/lib/grafana/plugins" \
 -e "GF_PATHS_PROVISIONING=/etc/grafana/provisioning" \
docker.io/grafana/grafana:latest
```

3. Create a physical volume storage (pvc) and mount pvc for Grafana by running the following command:

```
$oc set volume dc/grafana --add --mount-path=/var/lib/grafana/
--name=grafana-volume-1 -t pvc --claim-name=pvc-grafana-1 --claim-size=25G
```

4. If you configure an existing pvc, run the following command:

```
$oc set volume dc/grafana --add --mount-path=/var/lib/grafana/
--name=grafana-volume-1  -t pvc --claim-name=pvc-grafana-1
```

5. To see the pvc details, run the **$oc get pvc** command:

```
[user2@server data]$ oc get pvc
NAME            STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
pvc-grafana-1   Bound    pv0015   100Gi      RWO,ROX,RWX                   32m
```

6. To see the Grafana configuration details, run the **$oc get dc/grafana -o yaml** command, as shown in Example 4-4.

Example 4-4 Grafana yaml configuration file

```
[user2@server ~]$ oc get dc/grafana -o yaml
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  annotations:
    openshift.io/generated-by: OpenShiftNewApp
  creationTimestamp: "2020-04-17T03:16:39Z"
  generation: 4
  labels:
    app: grafana
    app.kubernetes.io/component: grafana
    app.kubernetes.io/instance: grafana
  name: grafana
  namespace: emsproject
  resourceVersion: "1198233"
  selfLink:
/apis/apps.openshift.io/v1/namespaces/emsproject/deploymentconfigs/grafana
  uid: e6c964fb-7441-42fc-bcb5-0f9b6eb90224
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    deploymentconfig: grafana
  strategy:
    activeDeadlineSeconds: 21600
    resources: {}
    rollingParams:
      intervalSeconds: 1
      maxSurge: 25%
      maxUnavailable: 25%
      timeoutSeconds: 600
      updatePeriodSeconds: 1
    type: Rolling
  template:
    metadata:
      annotations:
        openshift.io/generated-by: OpenShiftNewApp
```

```
      creationTimestamp: null
      labels:
        deploymentconfig: grafana
    spec:
      containers:
      - env:
        - name: GF_PATHS_CONFIG
          value: /etc/grafana/grafana.ini
        - name: GF_PATHS_DATA
          value: /var/lib/grafana
        - name: GF_PATHS_HOME
          value: /usr/share/grafana
        - name: GF_PATHS_LOGS
          value: /var/log/grafana
        - name: GF_PATHS_PLUGINS
          value: /var/lib/grafana/plugins
        - name: GF_PATHS_PROVISIONING
          value: /etc/grafana/provisioning
        image:
docker.io/grafana/grafana@sha256:bdef6f27255a09deb2f89741b3800a9a394a7e9eefa032570
760e5688dd00a2f
        imagePullPolicy: Always
        name: grafana
        ports:
        - containerPort: 3000
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /var/lib/grafana
          name: grafana-volume-1
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
      - name: grafana-volume-1
        persistentVolumeClaim:
          claimName: pvc-grafana-1
  test: false
  triggers:
  - type: ConfigChange
  - imageChangeParams:
      automatic: true
      containerNames:
      - grafana
      from:
        kind: ImageStreamTag
        name: grafana:latest
        namespace: emsproject
      lastTriggeredImage:
docker.io/grafana/grafana@sha256:bdef6f27255a09deb2f89741b3800a9a394a7e9eefa032570
760e5688dd00a2f
```

```
    type: ImageChange
status:
  availableReplicas: 1
  conditions:
  - lastTransitionTime: "2020-04-17T03:17:08Z"
    lastUpdateTime: "2020-04-17T03:17:08Z"
    message: Deployment config has minimum availability.
    status: "True"
    type: Available
  - lastTransitionTime: "2020-04-17T03:24:25Z"
    lastUpdateTime: "2020-04-17T03:24:28Z"
    message: replication controller "grafana-3" successfully rolled out
    reason: NewReplicationControllerAvailable
    status: "True"
    type: Progressing
  details:
    causes:
    - type: Manual
    message: manual change
  latestVersion: 3
  observedGeneration: 4
  readyReplicas: 1
  replicas: 1
  unavailableReplicas: 0
  updatedReplicas: 1
```

7. Set up Grafana access by running the following commands:

```
$oc expose svc/grafana
$oc get route/grafana

[user2@server ~]$ oc get route/grafana
NAME      HOST/PORT                              PATH   SERVICES   PORT
TERMINATION   WILDCARD
grafana   grafana-emsproject.apps-crc.testing          grafana
3000-tcp                 None
```

8. Create the configuration map (see Example 4-5):

```
$oc create configmap grafana-config --from-file=conf/
conf/grafana.ini
```

Example 4-5 Grafana configuration file

```
##################### Grafana Configuration Example #####################
#
# Everything has defaults so you only need to uncomment things you want to
# change

# possible values : production, development
;app_mode = production

# instance name, defaults to HOSTNAME environment variable value or hostname if
HOSTNAME var is empty
;instance_name = ${HOSTNAME}

################################### Paths ###################################
[paths]
```

```
# Path to where grafana can store temp files, sessions, and the sqlite3 db (if
that is used)
;data = /var/lib/grafana

# Temporary files in `data` directory older than given duration will be removed
;temp_data_lifetime = 24h

# Directory where grafana can store logs
;logs = /var/log/grafana

# Directory where grafana will automatically scan and look for plugins
;plugins = /var/lib/grafana/plugins

# folder that contains provisioning config files that grafana will apply on
startup and while running.
;provisioning = conf/provisioning

#################################### Server ####################################
[server]
# Protocol (http, https, h2, socket)
;protocol = http

# The ip address to bind to, empty will bind to all interfaces
;http_addr =

# The http port  to use
;http_port = 3000

# The public facing domain name used to access grafana from a browser
;domain = localhost

# Redirect to correct domain if host header does not match domain
# Prevents DNS rebinding attacks
;enforce_domain = false

# The full public facing url you use in browser, used for redirects and emails
# If you use reverse proxy and sub path specify full url (with sub path)
;root_url = %(protocol)s://%(domain)s:%(http_port)s/

# Serve Grafana from subpath specified in `root_url` setting. By default it is set
to `false` for compatibility reasons.
;serve_from_sub_path = false

# Log web requests
;router_logging = false

# the path relative working path
;static_root_path = public

# enable gzip
;enable_gzip = false

# https certs & key file
;cert_file =
;cert_key =
```

```
# Unix socket path
;socket =

#################################### Database #####################################
[database]
# You can configure the database connection by specifying type, host, name, user
and password
# as separate properties or as on string using the url properties.

# Either "mysql", "postgres" or "sqlite3", it's your choice
;type = sqlite3
;host = 127.0.0.1:3306
;name = grafana
;user = root
# If the password contains # or ; you have to wrap it with triple quotes. Ex
"""#password;"""
;password =

# Use either URL or the previous fields to configure the database
# Example: mysql://user:secret@host:port/database
;url =

# For "postgres" only, either "disable", "require" or "verify-full"
;ssl_mode = disable

;ca_cert_path =
;client_key_path =
;client_cert_path =
;server_cert_name =

# For "sqlite3" only, path relative to data_path setting
;path = grafana.db

# Max idle conn setting default is 2
;max_idle_conn = 2

# Max conn setting default is 0 (mean not set)
;max_open_conn =

# Connection Max Lifetime default is 14400 (means 14400 seconds or 4 hours)
;conn_max_lifetime = 14400

# Set to true to log the sql calls and execution times.
;log_queries =

# For "sqlite3" only. cache mode setting used for connecting to the database.
(private, shared)
;cache_mode = private

#################################### Cache server #############################
[remote_cache]
# Either "redis", "memcached" or "database" default is "database"
;type = database
```

```
# cache connectionstring options
# database: will use Grafana primary database.
# redis: config like redis server e.g.
`addr=127.0.0.1:6379,pool_size=100,db=0,ssl=false`. Only addr is required. ssl may
be 'true', 'false', or 'insecure'.
# memcache: 127.0.0.1:11211
;connstr =

################################### Data proxy ###########################
[dataproxy]

# This enables data proxy logging, default is false
;logging = false

# How long the data proxy should wait before timing out default is 30 (seconds)
;timeout = 30

# If enabled and user is not anonymous, data proxy will add X-Grafana-User header
with username into the request, default is false.
;send_user_header = false

################################### Analytics
###################################
[analytics]
# Server reporting, sends usage counters to stats.grafana.org every 24 hours.
# No ip addresses are being tracked, only simple counters to track
# running instances, dashboard and error counts. It is very helpful to us.
# Change this option to false to disable reporting.
;reporting_enabled = true

# Set to false to disable all checks to https://grafana.net
# for new vesions (grafana itself and plugins), check is used
# in some UI views to notify that grafana or plugin update exists
# This option does not cause any auto updates, nor send any information
# only a GET request to http://grafana.com to get latest versions
;check_for_updates = true

# Google Analytics universal tracking code, only enabled if you specify an id here
;google_analytics_ua_id =

# Google Tag Manager ID, only enabled if you specify an id here
;google_tag_manager_id =

################################### Security ###################################
[security]
# disable creation of admin user on first start of grafana
;disable_initial_admin_creation = false

# default admin user, created on startup
;admin_user = admin

# default admin password, can be changed before first start of grafana,  or in
profile settings
;admin_password = admin
```

```
# used for signing
;secret_key = SW2YcwTIb9zpOOhoPsMm

# disable gravatar profile images
;disable_gravatar = false

# data source proxy whitelist (ip_or_domain:port separated by spaces)
;data_source_proxy_whitelist =

# disable protection against brute force login attempts
;disable_brute_force_login_protection = false

# set to true if you host Grafana behind HTTPS. default is false.
;cookie_secure = false

# set cookie SameSite attribute. defaults to `lax`. can be set to "lax", "strict",
"none" and "disabled"
;cookie_samesite = lax

# set to true if you want to allow browsers to render Grafana in a <frame>,
<iframe>, <embed> or <object>. default is false.
allow_embedding = true

# Set to true if you want to enable http strict transport security (HSTS) response
header.
# This is only sent when HTTPS is enabled in this configuration.
# HSTS tells browsers that the site should only be accessed using HTTPS.
# The default version will change to true in the next minor release, 6.3.
;strict_transport_security = false

# Sets how long a browser should cache HSTS. Only applied if
strict_transport_security is enabled.
;strict_transport_security_max_age_seconds = 86400

# Set to true if to enable HSTS preloading option. Only applied if
strict_transport_security is enabled.
;strict_transport_security_preload = false

# Set to true if to enable the HSTS includeSubDomains option. Only applied if
strict_transport_security is enabled.
;strict_transport_security_subdomains = false

# Set to true to enable the X-Content-Type-Options response header.
# The X-Content-Type-Options response HTTP header is a marker used by the server
to indicate that the MIME types advertised
# in the Content-Type headers should not be changed and be followed. The default
will change to true in the next minor release, 6.3.
;x_content_type_options = false

# Set to true to enable the X-XSS-Protection header, which tells browsers to stop
pages from loading
# when they detect reflected cross-site scripting (XSS) attacks. The default will
change to true in the next minor release, 6.3.
;x_xss_protection = false
```

```
#################################### Snapshots ##########################
[snapshots]
# snapshot sharing options
;external_enabled = true
;external_snapshot_url = https://snapshots-origin.raintank.io
;external_snapshot_name = Publish to snapshot.raintank.io

# Set to true to enable this Grafana instance act as an external snapshot server
and allow unauthenticated requests for
# creating and deleting snapshots.
;public_mode = false

# remove expired snapshot
;snapshot_remove_expired = true

#################################### Dashboards History #################
[dashboards]
# Number dashboard versions to keep (per dashboard). Default: 20, Minimum: 1
;versions_to_keep = 20

# Minimum dashboard refresh interval. When set, this will restrict users to set
the refresh interval of a dashboard lower than given interval. Per default this is
not set/unrestricted.
# The interval string is a possibly signed sequence of decimal numbers, followed
by a unit suffix (ms, s, m, h, d), e.g. 30s or 1m.
;min_refresh_interval =

#################################### Users ###############################
[users]
# disable user signup / registration
;allow_sign_up = true

# Allow non admin users to create organizations
;allow_org_create = true

# Set to true to automatically assign new users to the default organization (id 1)
;auto_assign_org = true

# Set this value to automatically add new users to the provided organization (if
auto_assign_org above is set to true)
;auto_assign_org_id = 1

# Default role new users will be automatically assigned (if disabled above is set
to true)
;auto_assign_org_role = Viewer

# Require email validation before sign up completes
;verify_email_enabled = false

# Background text for the user field on the login page
;login_hint = email or username
;password_hint = password

# Default UI theme ("dark" or "light")
;default_theme = dark
```

```
# External user management, these options affect the organization users view
;external_manage_link_url =
;external_manage_link_name =
;external_manage_info =

# Viewers can edit/inspect dashboard settings in the browser. But not save the
dashboard.
;viewers_can_edit = false

# Editors can administrate dashboard, folders and teams they create
;editors_can_admin = false

[auth]
# Login cookie name
;login_cookie_name = grafana_session

# The lifetime (days) an authenticated user can be inactive before being required
to login at next visit. Default is 7 days,
;login_maximum_inactive_lifetime_days = 7

# The maximum lifetime (days) an authenticated user can be logged in since login
time before being required to login. Default is 30 days.
;login_maximum_lifetime_days = 30

# How often should auth tokens be rotated for authenticated users when being
active. The default is each 10 minutes.
;token_rotation_interval_minutes = 10

# Set to true to disable (hide) the login form, useful if you use OAuth, defaults
to false
;disable_login_form = false

# Set to true to disable the signout link in the side menu. useful if you use
auth.proxy, defaults to false
;disable_signout_menu = false

# URL to redirect the user to after sign out
;signout_redirect_url =

# Set to true to attempt login with OAuth automatically, skipping the login
screen.
# This setting is ignored if multiple OAuth providers are configured.
;oauth_auto_login = false

# limit of api_key seconds to live before expiration
;api_key_max_seconds_to_live = -1

################################### Anonymous Auth ######################
[auth.anonymous]
# enable anonymous access
;enabled = false

# specify organization name that should be used for unauthenticated users
;org_name = Main Org.
```

```
# specify role for unauthenticated users
;org_role = Viewer

#################################### Github Auth ##########################
[auth.github]
;enabled = false
;allow_sign_up = true
;client_id = some_id
;client_secret = some_secret
;scopes = user:email,read:org
;auth_url = https://github.com/login/oauth/authorize
;token_url = https://github.com/login/oauth/access_token
;api_url = https://api.github.com/user
;allowed_domains =
;team_ids =
;allowed_organizations =

#################################### GitLab Auth ##########################
[auth.gitlab]
;enabled = false
;allow_sign_up = true
;client_id = some_id
;client_secret = some_secret
;scopes = api
;auth_url = https://gitlab.com/oauth/authorize
;token_url = https://gitlab.com/oauth/token
;api_url = https://gitlab.com/api/v4
;allowed_domains =
;allowed_groups =

#################################### Google Auth ##########################
[auth.google]
;enabled = false
;allow_sign_up = true
;client_id = some_client_id
;client_secret = some_client_secret
;scopes = https://www.googleapis.com/auth/userinfo.profile
https://www.googleapis.com/auth/userinfo.email
;auth_url = https://accounts.google.com/o/oauth2/auth
;token_url = https://accounts.google.com/o/oauth2/token
;api_url = https://www.googleapis.com/oauth2/v1/userinfo
;allowed_domains =
;hosted_domain =

#################################### Grafana.com Auth ####################
[auth.grafana_com]
;enabled = false
;allow_sign_up = true
;client_id = some_id
;client_secret = some_secret
;scopes = user:email
;allowed_organizations =

#################################### Azure AD OAuth #######################
```

```
[auth.azuread]
;name = Azure AD
;enabled = false
;allow_sign_up = true
;client_id = some_client_id
;client_secret = some_client_secret
;scopes = openid email profile
;auth_url = https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/authorize
;token_url = https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/token
;allowed_domains =
;allowed_groups =

################################### Generic OAuth ##########################
[auth.generic_oauth]
;enabled = false
;name = OAuth
;allow_sign_up = true
;client_id = some_id
;client_secret = some_secret
;scopes = user:email,read:org
;email_attribute_name = email:primary
;email_attribute_path =
;auth_url = https://foo.bar/login/oauth/authorize
;token_url = https://foo.bar/login/oauth/access_token
;api_url = https://foo.bar/user
;allowed_domains =
;team_ids =
;allowed_organizations =
;role_attribute_path =
;tls_skip_verify_insecure = false

;tls_client_cert =
;tls_client_key =
;tls_client_ca =

################################### SAML Auth ##############################
[auth.saml] # Enterprise only
# Defaults to false. If true, the feature is enabled.
;enabled = false

# Base64-encoded public X.509 certificate. Used to sign requests to the IdP
;certificate =

# Path to the public X.509 certificate. Used to sign requests to the IdP
;certificate_path =

# Base64-encoded private key. Used to decrypt assertions from the IdP
;private_key =

;# Path to the private key. Used to decrypt assertions from the IdP
;private_key_path =

# Base64-encoded IdP SAML metadata XML. Used to verify and obtain binding
locations from the IdP
;idp_metadata =
```

```
# Path to the SAML metadata XML. Used to verify and obtain binding locations from
the IdP
;idp_metadata_path =

# URL to fetch SAML IdP metadata. Used to verify and obtain binding locations from
the IdP
;idp_metadata_url =

# Duration, since the IdP issued a response and the SP is allowed to process it.
Defaults to 90 seconds.
;max_issue_delay = 90s

# Duration, for how long the SP's metadata should be valid. Defaults to 48 hours.

# Friendly name or name of the attribute within the SAML assertion to use as the
user's name
;assertion_attribute_name = displayName

# Friendly name or name of the attribute within the SAML assertion to use as the
user's login handle
;assertion_attribute_login = mail

# Friendly name or name of the attribute within the SAML assertion to use as the
user's email
;assertion_attribute_email = mail

#################################### Basic Auth ##########################
[auth.basic]
;enabled = true

#################################### Auth Proxy ##########################
[auth.proxy]
;enabled = false
;header_name = X-WEBAUTH-USER
;header_property = username
;auto_sign_up = true
;sync_ttl = 60
;whitelist = 192.168.1.1, 192.168.2.1
;headers = Email:X-User-Email, Name:X-User-Name
# Read the auth proxy docs for details on what the setting below enables
;enable_login_token = false

#################################### Auth LDAP ##########################
[auth.ldap]
;enabled = false
;config_file = /etc/grafana/ldap.toml
;allow_sign_up = true

# LDAP backround sync (Enterprise only)
# At 1 am every day
;sync_cron = "0 0 1 * * *"
;active_sync_enabled = true

#################################### SMTP / Emailing ##########################
```

```
[smtp]
;enabled = false
;host = localhost:25
;user =
# If the password contains # or ; you have to wrap it with triple quotes. Ex
"""#password;"""
;password =
;cert_file =
;key_file =
;skip_verify = false
;from_address = admin@grafana.localhost
;from_name = Grafana
# EHLO identity in SMTP dialog (defaults to instance_name)
;ehlo_identity = dashboard.example.com

[emails]
;welcome_email_on_sign_up = false
;templates_pattern = emails/*.html

#################################### Logging ##########################
[log]
# Either "console", "file", "syslog". Default is console and  file
# Use space to separate multiple modes, e.g. "console file"
;mode = console file

# Either "debug", "info", "warn", "error", "critical", default is "info"
;level = info

# optional settings to set different levels for specific loggers. Ex filters =
sqlstore:debug
;filters =

# For "console" mode only
[log.console]
;level =

# log line format, valid options are text, console and json
;format = console

# For "file" mode only
[log.file]
;level =

# log line format, valid options are text, console and json
;format = text

# This enables automated log rotate(switch of following options), default is true
;log_rotate = true

# Max line number of single file, default is 1000000
;max_lines = 1000000

# Max size shift of single file, default is 28 means 1 << 28, 256MB
;max_size_shift = 28
```

```
# Segment log daily, default is true
;daily_rotate = true

# Expired days of log file(delete after max days), default is 7
;max_days = 7

[log.syslog]
;level =

# log line format, valid options are text, console and json
;format = text

# Syslog network type and address. This can be udp, tcp, or unix. If left blank,
the default unix endpoints will be used.
;network =
;address =

# Syslog facility. user, daemon and local0 through local7 are valid.
;facility =

# Syslog tag. By default, the process' argv[0] is used.
;tag =

#################################### Usage Quotas ########################
[quota]
; enabled = false

#### set quotas to -1 to make unlimited. ####
# limit number of users per Org.
; org_user = 10

# limit number of dashboards per Org.
; org_dashboard = 100

# limit number of data_sources per Org.
; org_data_source = 10

# limit number of api_keys per Org.
; org_api_key = 10

# limit number of orgs a user can create.
; user_org = 10

# Global limit of users.
; global_user = -1

# global limit of orgs.
; global_org = -1

# global limit of dashboards
; global_dashboard = -1

# global limit of api_keys
; global_api_key = -1
```

```
# global limit on number of logged in users.
; global_session = -1

################################### Alerting ############################
[alerting]
# Disable alerting engine & UI features
;enabled = true
# Makes it possible to turn off alert rule execution but alerting UI is visible
;execute_alerts = true

# Default setting for new alert rules. Defaults to categorize error and timeouts
as alerting. (alerting, keep_state)
;error_or_timeout = alerting

# Default setting for how Grafana handles nodata or null values in alerting.
(alerting, no_data, keep_state, ok)
;nodata_or_nullvalues = no_data

# Alert notifications can include images, but rendering many images at the same
time can overload the server
# This limit will protect the server from render overloading and make sure
notifications are sent out quickly
;concurrent_render_limit = 5


# Default setting for alert calculation timeout. Default value is 30
;evaluation_timeout_seconds = 30

# Default setting for alert notification timeout. Default value is 30
;notification_timeout_seconds = 30

# Default setting for max attempts to sending alert notifications. Default value
is 3
;max_attempts = 3

# Makes it possible to enforce a minimal interval between evaluations, to reduce
load on the backend
;min_interval_seconds = 1

################################### Explore #############################
[explore]
# Enable the Explore section
;enabled = true

################################### Internal Grafana Metrics
#########################
# Metrics available at HTTP API Url /metrics
[metrics]
# Disable / Enable internal metrics
;enabled           = true
# Graphite Publish interval
;interval_seconds  = 10
# Disable total stats (stat_totals_*) metrics to be generated
;disable_total_stats = false
```

```
#If both are set, basic auth will be required for the metrics endpoint.
; basic_auth_username =
; basic_auth_password =

# Send internal metrics to Graphite
[metrics.graphite]
# Enable by setting the address setting (ex localhost:2003)
;address =
;prefix = prod.grafana.%(instance_name)s.

################################### Grafana.com integration
#########################
# Url used to import dashboards directly from Grafana.com
[grafana_com]
;url = https://grafana.com

################################### Distributed tracing ############
[tracing.jaeger]
# Enable by setting the address sending traces to jaeger (ex localhost:6831)
;address = localhost:6831
# Tag that will always be included in when creating new spans. ex
(tag1:value1,tag2:value2)
;always_included_tag = tag1:value1
# Type specifies the type of the sampler: const, probabilistic, rateLimiting, or
remote
;sampler_type = const
# jaeger samplerconfig param
# for "const" sampler, 0 or 1 for always false/true respectively
# for "probabilistic" sampler, a probability between 0 and 1
# for "rateLimiting" sampler, the number of spans per second
# for "remote" sampler, param is the same as for "probabilistic"
# and indicates the initial sampling rate before the actual one
# is received from the mothership
;sampler_param = 1
# Whether or not to use Zipkin propagation (x-b3- HTTP headers).
;zipkin_propagation = false
# Setting this to true disables shared RPC spans.
# Not disabling is the most common setting when using Zipkin elsewhere in your
infrastructure.
;disable_shared_zipkin_spans = false

################################### External image storage
#########################
[external_image_storage]
# Used for uploading images to public servers so they can be included in
slack/email messages.
# you can choose between (s3, webdav, gcs, azure_blob, local)
;provider =

[external_image_storage.s3]
;endpoint =
;path_style_access =
;bucket =
;region =
;path =
```

```
;access_key =
;secret_key =

[external_image_storage.webdav]
;url =
;public_url =
;username =
;password =

[external_image_storage.gcs]
;key_file =
;bucket =
;path =

[external_image_storage.azure_blob]
;account_name =
;account_key =
;container_name =

[external_image_storage.local]
# does not require any configuration

[rendering]
# Options to configure a remote HTTP image rendering service, e.g. using
https://github.com/grafana/grafana-image-renderer.
# URL to a remote HTTP image renderer service, e.g. http://localhost:8081/render,
will enable Grafana to render panels and dashboards to PNG-images using HTTP
requests to an external service.
;server_url =
# If the remote HTTP image renderer service runs on a different server than the
Grafana server you may have to configure this to a URL where Grafana is reachable,
e.g. http://grafana.domain/.
;callback_url =

[panels]
# If set to true Grafana will allow script tags in text panels. Not recommended as
it enable XSS vulnerabilities.
;disable_sanitize_html = false

[plugins]
;enable_alpha = false
;app_tls_skip_verify_insecure = false

[enterprise]
# Path to a valid Grafana Enterprise license.jwt file
;license_path =

[feature_toggles]
# enable features, separated by spaces
;enable =
```

```
$oc set volume dc/grafana --add --mount-path=/etc/grafana/conf
--name=grafana-volume-2 -t configmap --configmap-name=grafana-config
```

9. If you have Grafana storage from an existing Podman or Docker, copy it to the container and rollout the Pod:

   ```
   $ oc cp grafana.db <container-name>:/var/lib/grafana
   ```

   For example:

   ```
   [user2@server data]$ oc cp grafana.db grafana-3-2m5m4:/var/lib/grafana

   $oc rollout latest dc/grafana
   ```

10. Browse Grafana from the data, as shown in Figure 4-16.

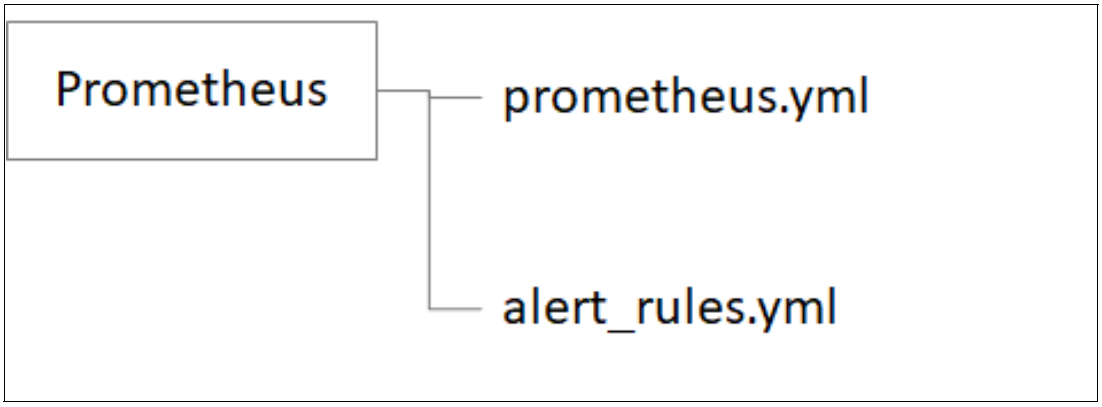


Figure 4-16 Loading from existing Grafana data

4.2.4 Creating an alertmanager in Red Hat OpenShift

To create an alert manager in Red Hat OpenShift, you must deploy an alertmanager container and update the `prometheus.yml` file to point to the alertmanager URL, as shown in Figure 4-17.

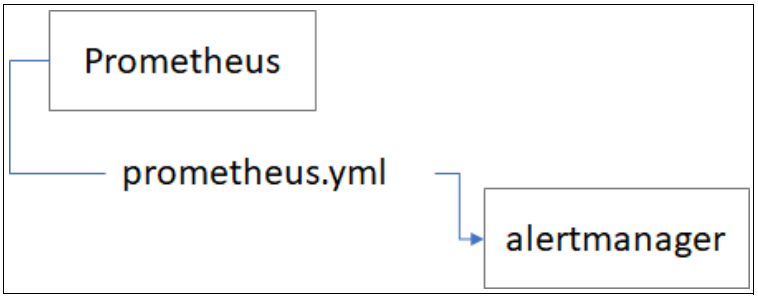


Figure 4-17 Prometheus.yml pointing to alertmanager

To create an alertmanager, run the following commands:

```
$oc new-app docker.io/prom/alertmanager:latest
$oc expose svc/alertmanager
$oc get route
```

For example:

```
[user2@server prometheus]$ oc get route|grep alert
alertmanager   alertmanager-emsproject.apps-crc.testing alertmanager   9093-tcp
```

To browse alert manager, change in the `prometheus.yml` file the alertmanagers targets with the alert manager URL; for example, `ex.alertmanager-emsproject.apps-crc.testing`.

Check the .yml file, as follows (see Example 4-6):

```
$cat prometheus.yml
```

Example 4-6 Check contents prometheus.yml file

```
[user2@server prometheus]$ cat prometheus.yml
# my global config
global:
  scrape_interval:     20s # Set the scrape interval to every 15 seconds. Default
is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every
1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    #path_prefix: /alertmanager
    - targets:
      - alertmanager-emsproject.apps-crc.testing
      #- alertmanager:9093
      #- 169.38.67.66:9093

# Load rules once and periodically evaluate them according to the global
'evaluation_interval'.
rule_files:
  - 'alert_rules.yml'
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped
from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['prom.redbooks.info:9090']

  - job_name: 'node_exporter'
    static_configs:
    - targets: ['prom.redbooks.info:9100']
    - targets: ['134.209.22.37:9100']
    - targets: ['169.38.131.87:9100']
    - targets: ['tomcat.redbooks.info:9100']
    - targets: ['node1-sql.redbooks.info:9100']
    - targets: ['node2-sql.redbooks.info:9100']
    relabel_configs:
    - source_labels: [__address__]
      regex:  'prom.redbooks.info:9100'
      target_label: instance
```

```
    replacement: 'prometheus_server'
  - source_labels: [__address__]
    regex:  '134.209.22.37:9100'
    target_label: instance
    replacement: 'external_server'
  - source_labels: [__address__]
    regex:  '169.38.131.87:9100'
    target_label: instance
    replacement: 'elk_server'
  - source_labels: [__address__]
    regex:  'tomcat.redbooks.info:9100'
    target_label: instance
    replacement: 'tomcat_server'
  - source_labels: [__address__]
    regex:  'node1-sql.redbooks.info:9100'
    target_label: instance
    replacement: 'node1-sql_server'
  - source_labels: [__address__]
    regex:  'node2-sql.redbooks.info:9100'
    target_label: instance
    replacement: 'node2-sql_server'

- job_name: 'mysqld_exporter'
  static_configs:
  - targets: ['node1-sql.redbooks.info:9104']
  - targets: ['node2-sql.redbooks.info:9104']
  - targets: ['node1-sql.redbooks.info:9144']

- job_name: 'apache'
  static_configs:
  - targets: ['tomcat.redbooks.info:9117']
- job_name: 'java'
  static_configs:
  - targets: ['tomcat.redbooks.info:8181']
  relabel_configs:
  - source_labels: [__address__]
    regex: 'tomcat.redbooks.info:8181'
    target_label: instance
    replacement: 'tomcat'
- job_name: 'ipmi_exporter'
  scrape_interval: 1m
  scrape_timeout: 30s
  metrics_path: /ipmi
  scheme: http
  file_sd_configs:
  - files:
    - /root/prometheus-2.16.0.linux-amd64/targets.yml
    refresh_interval: 5m
  relabel_configs:
  - source_labels: [__address__]
    separator: ;
    regex: (.*)
    target_label: __param_target
    replacement: ${1}
    action: replace
```

```
- source_labels: [__param_target]
  separator: ;
  regex: (.*)
  target_label: instance
  replacement: ${1}
  action: replace
- separator: ;
  regex: .*
  target_label: __address__
  replacement: ipmi-exp.redbooks.info:9290
  action: replace
```

When a prometheusAlert is received, alertmanager receives the incident, as shown in Figure 4-18, Figure 4-19 on page 261, and Figure 4-20 on page 261.

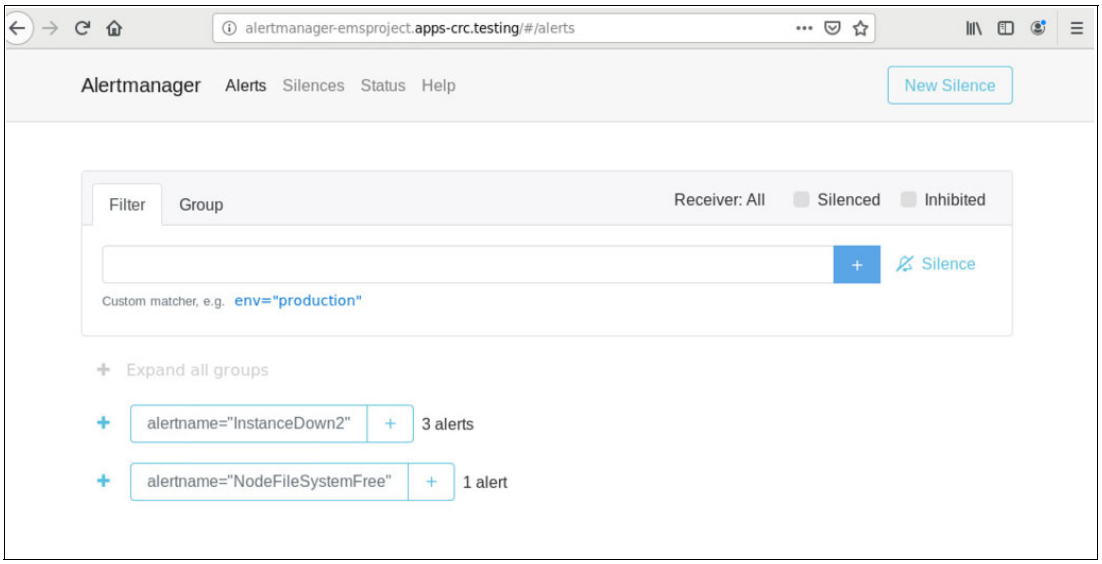


Figure 4-18 Alertmanager Alerts Filter

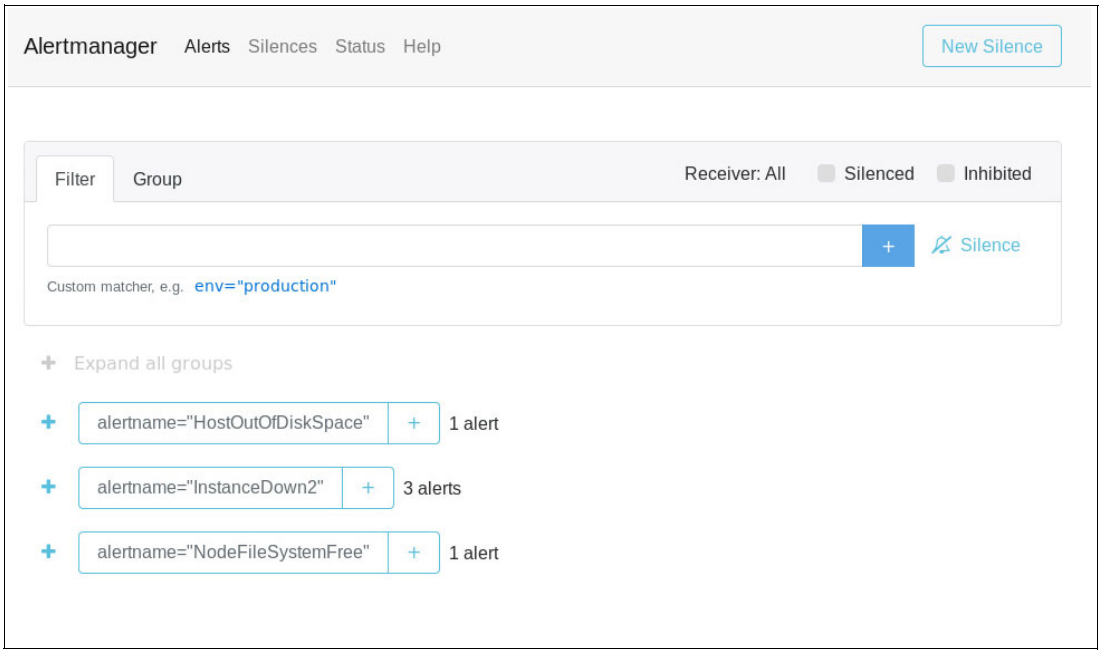


Figure 4-19 Alertmanager Filter: alertname HostOutOfDiskSpace

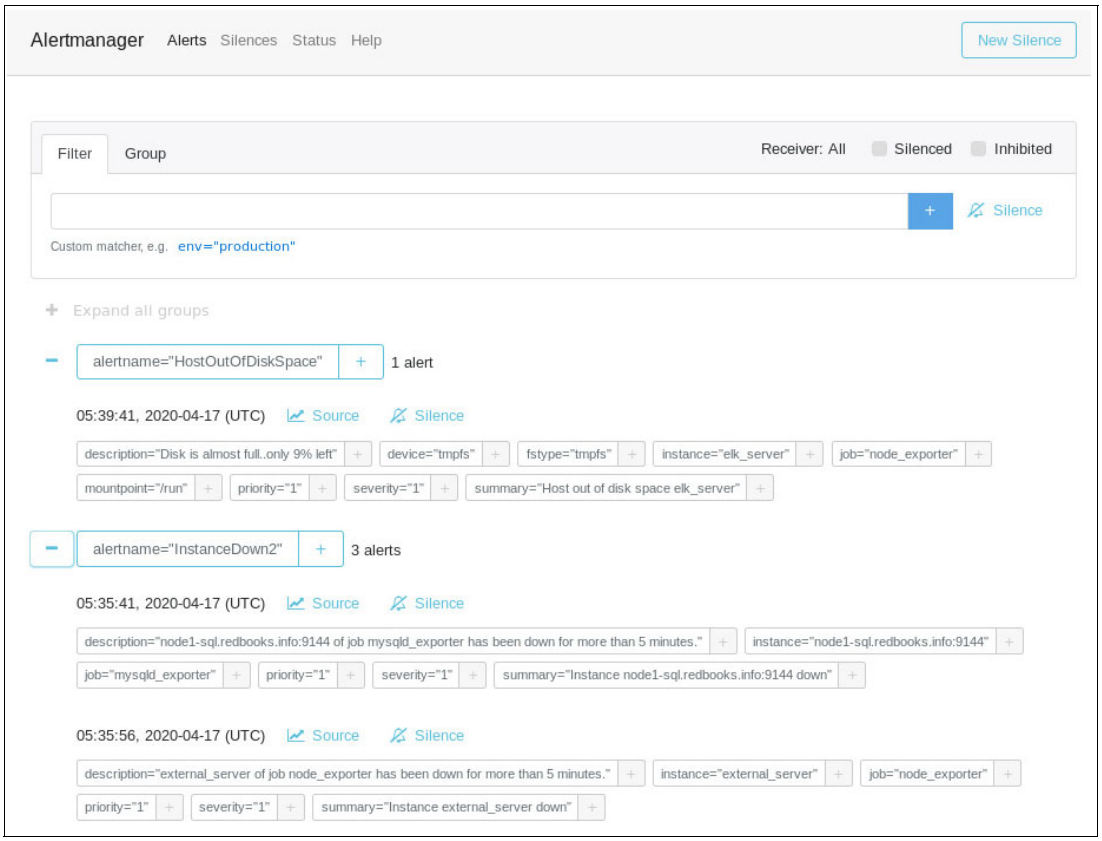


Figure 4-20 Alertmanager Filter: alertname description

4.2.5 Creating an iTop in Red Hat OpenShift

To create an iTop in Red Hat OpenShift, deploy the iTop container and create a persistence directory to store mysql database and web content (see Figure 4-21).

Figure 4-21 iTop directory structure

To create an iTop container in Red Hat OpenShift, run the following command:

```
$oc new-app docker.io/vbkunin/itop
```

To make the service available, run the following command:

```
$oc expose svc/itop
```

To create two Persistent volume claims (PVC) in the `.yml` file (one for the database and one for web content) run the following command (see Example 4-7 and Example 4-8 on page 263):

```
$oc create -f itop-pvc.yml
```

Example 4-7 Persistent volume claim: pvc-itop-1

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
  name: pvc-itop-1
  namespace: emsproject
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100G
status:
  accessModes:
  - ReadWriteOnce
  - ReadWriteMany
  - ReadOnlyMany
  capacity:
    storage: 100Gi
  phase: Bound
```

Example 4-8 Persistent volume claim: pvc-itop-3

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
  name: pvc-itop-3
  namespace: emsproject
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100G
status:
  accessModes:
  - ReadWriteOnce
  - ReadWriteMany
  - ReadOnlyMany
  capacity:
    storage: 100Gi
  phase: Bound
```

The following commands set the persistent volumes and mounting paths for the database and the web content:

```
$oc set volume dc/itop --add --mount-path=/var/lib/mysql/itop_db
--name=itop-volume-1 -t pvc --claim-name=pvc-itop-1 --overwrite

$oc set volume dc/itop --add --mount-path=/var/www/html --name=itop-volume-3 -t
pvc --claim-name=pvc-itop-3 --overwrite
```

To import from the environment, copy the web content to the container that is deployed on persistence:

```
$oc get pods
$oc cp /home/user2/itop/backup/html itop-3-n6h2s:/var/www/
itop-3-n6h2s is container id from the output of "oc get pods"
```

If a mysql database is available from an environment, export the database:

```
$oc cp /home/user2/itop/itop_db.sql  itop-6-qg7bl:/tmp
itop-6-qg7b is the container id  from "oc get pods"
```

To export the database data, run the following command:

```
$mysqldump --add-drop-table -uroot itop_db > itop_db.sql
```

Then, import the data by running the following command:

```
$mysql -uroot itop_db< itop_db.sql
```

Example 4-9 shows the final iTop deployment configuration content.

Example 4-9 iTop configuration

```
[user2@codeReady itop]$ oc get dc/itop -o yaml
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
```

```
  annotations:
    openshift.io/generated-by: OpenShiftNewApp
  creationTimestamp: "2020-04-21T02:13:25Z"
  generation: 6
  labels:
    app: itop
    app.kubernetes.io/component: itop
    app.kubernetes.io/instance: itop
  name: itop
  namespace: emsproject
  resourceVersion: "577573"
  selfLink:
/apis/apps.openshift.io/v1/namespaces/emsproject/deploymentconfigs/itop
  uid: 2d5554de-2506-4ca6-b214-6d2977ff0cd4
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    deploymentconfig: itop
  strategy:
    activeDeadlineSeconds: 21600
    resources: {}
    rollingParams:
      intervalSeconds: 1
      maxSurge: 25%
      maxUnavailable: 25%
      timeoutSeconds: 600
      updatePeriodSeconds: 1
    type: Rolling
  template:
    metadata:
      annotations:
        openshift.io/generated-by: OpenShiftNewApp
      creationTimestamp: null
      labels:
        deploymentconfig: itop
    spec:
      containers:
      - image:
docker.io/vbkunin/itop@sha256:d402e866384b2c74c75cfaaeca9e21f945798e4b310c79fa0239
62fe01f0f249
        imagePullPolicy: Always
        name: itop
        ports:
        - containerPort: 3306
          protocol: TCP
        - containerPort: 80
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /var/www/html
          name: itop-volume-3
        - mountPath: /var/lib/mysql/itop_db
```

```
          name: itop-volume-1
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: useroot
      serviceAccountName: useroot
      terminationGracePeriodSeconds: 30
      volumes:
      - name: itop-volume-3
        persistentVolumeClaim:
          claimName: pvc-itop-3
      - name: itop-volume-1
        persistentVolumeClaim:
          claimName: pvc-itop-1
  test: false
  triggers:
  - type: ConfigChange
  - imageChangeParams:
      automatic: true
      containerNames:
      - itop
      from:
        kind: ImageStreamTag
        name: itop:latest
        namespace: emsproject
      lastTriggeredImage:
docker.io/vbkunin/itop@sha256:d402e866384b2c74c75cfaaeca9e21f945798e4b310c79fa0239
62fe01f0f249
    type: ImageChange
status:
  availableReplicas: 1
  conditions:
  - lastTransitionTime: "2020-04-21T02:13:51Z"
    lastUpdateTime: "2020-04-21T02:13:51Z"
    message: Deployment config has minimum availability.
    status: "True"
    type: Available
  - lastTransitionTime: "2020-04-21T02:17:05Z"
    lastUpdateTime: "2020-04-21T02:17:08Z"
    message: replication controller "itop-4" successfully rolled out
    reason: NewReplicationControllerAvailable
    status: "True"
    type: Progressing
  details:
    causes:
    - type: Manual
    message: manual change
  latestVersion: 4
  observedGeneration: 6
  readyReplicas: 1
  replicas: 1
  unavailableReplicas: 0
  updatedReplicas: 1
```

Browse iTop (see Figure 4-22). Find the URL by running the following command:

```
$oc get route itop
```

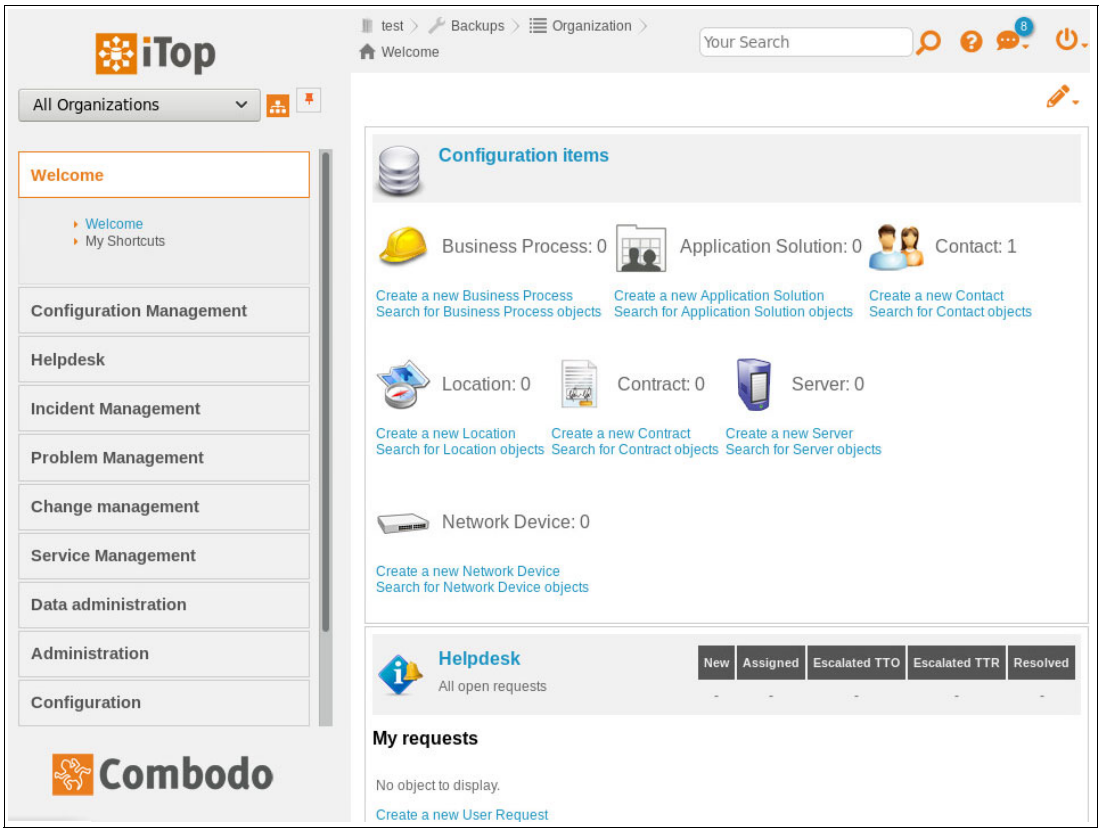


Figure 4-22 iTop Welcome window

In a production environment, the database is running as a separate pod and service.

To use mysql as a name for the pod, use mysql during the database configuration, for example:

```
$oc new-app docker.io/mysql:5.7.29 -e MYSQL_ROOT_PASSWORD=$password
```

To check whether the configuration was successful and to find the two pods, run the following command:

```
$oc get pods
```

For example:

```
[user2@server ]$ oc get pods
NAME             READY   STATUS      RESTARTS   AGE
itop-3-deploy    0/1     Completed   0          3h46m
itop-4-c7drm     1/1     Running     1          76m
itop-4-deploy    0/1     Completed   0          76m
mysql-1-deploy   0/1     Completed   0          3h24m
mysql-1-gmqsb    1/1     Running     0          3h23m
```

4.2.6 Creating a webhook in Red Hat OpenShift

To create a webhook in Red Hat OpenShift, create a webhook container. A configuration file that is named `hook.json` is included with webhook, as shown in Figure 4-23.

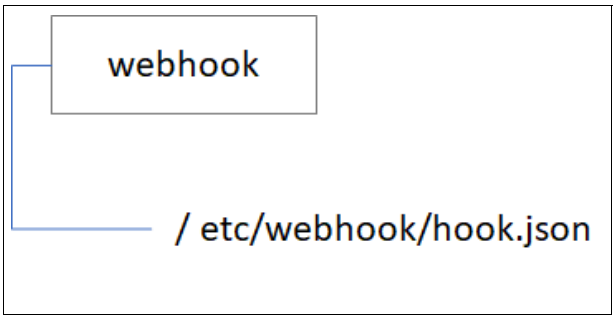


Figure 4-23 The webhook JSON file

In this scenario, webhook starts a send message, which is written in bash and adds modules that were pulled from the webhook image.

The webhook_j3 is a customized image:

```
$oc new-app quay.io/manasmohsin/webhook_j3:v3
```

Create a configmap to store `webhook.json` (see Example 4-10):

```
$oc create configmap webhook-config --from-file=/home/user2/webhook/hooks.json
$oc get configmap/webhook-config -o yaml
```

Example 4-10 The webhook configuration yaml file

```
[user2@codeReady ~]$ oc get configmap/webhook-config -o yaml
apiVersion: v1
data:
  hooks.json: |
    [
    {
    "id": "runscript",
    "execute-command": "/home/j123/sendmessage.sh",
    "command-working-directory": "/home/j123/",
    "pass-arguments-to-command":
    [
    {
      "source": "payload",
      "name": "alerts.0.labels.instance"
    },
    {
      "source": "payload",
      "name": "alerts.0.labels.alertname"
    },
    {
      "source": "string",
      "name": "is above threshold"
    },
    {
      "source": "string",
      "name": "HARD"
```

```
    },
    {
      "source": "payload",
      "name": "alerts.0.labels.severity"
    },
    {
      "source": "payload",
      "name": "alerts.0.labels.priority"
    },
    {
      "source": "payload",
      "name": "alerts.0.labels.summary"
    },
    {
      "source": "payload",
      "name": "alerts.0.labels.description"
    }
    ]
    }
    ]
kind: ConfigMap
metadata:
  creationTimestamp: "2020-04-22T05:56:18Z"
  name: webhook-config
  namespace: emsproject
  resourceVersion: "1025467"
  selfLink: /api/v1/namespaces/emsproject/configmaps/webhook-config
  uid: 0c2fb58e-ee91-4e29-b587-71c7737b68cf
```

Remove the empty volume and add the configmap volume:

```
$oc set volume dc/webhookj3 --remove  --mount-path=/etc/webhook
--name=webhookj3-volume-1 --confirm

$ oc set volume dc/webhookj3 --add --mount-path=/etc/webhook
--name=webhook-volume-1 -t configmap --configmap-name=webhook-config --overwrite
```

Find volumemount and volume in the deployment descriptor (see Example 4-11):

```
$oc get dc/webhookj3 -o yaml
```

Example 4-11 Search deployment descriptor

```
[user2@codeReady ~]$ oc get dc/webhookj3 -o yaml
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  annotations:
    openshift.io/generated-by: OpenShiftNewApp
  creationTimestamp: "2020-04-23T06:42:40Z"
  generation: 4
  labels:
    app: webhookj3
    app.kubernetes.io/component: webhookj3
    app.kubernetes.io/instance: webhookj3
  name: webhookj3
  namespace: emsproject
```

```
  resourceVersion: "1029372"
  selfLink:
/apis/apps.openshift.io/v1/namespaces/emsproject/deploymentconfigs/webhookj3
  uid: b2844165-8244-453d-976d-8be9c5e46f34
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    deploymentconfig: webhookj3
  strategy:
    activeDeadlineSeconds: 21600
    resources: {}
    rollingParams:
      intervalSeconds: 1
      maxSurge: 25%
      maxUnavailable: 25%
      timeoutSeconds: 600
      updatePeriodSeconds: 1
    type: Rolling
  template:
    metadata:
      annotations:
        openshift.io/generated-by: OpenShiftNewApp
      creationTimestamp: null
      labels:
        deploymentconfig: webhookj3
    spec:
      containers:
      - image:
quay.io/manasmohsin/webhook_j3@sha256:0cfa40a5572aa08dec1b290a74f56baf7e2a3538c702
fb38d16f47ff42340065
        imagePullPolicy: IfNotPresent
        name: webhookj3
        ports:
        - containerPort: 9000
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /etc/webhook
          name: webhookj3-volume-1
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
      - configMap:
          defaultMode: 420
          name: webhook-config
        name: webhookj3-volume-1
  test: false
  triggers:
  - type: ConfigChange
```

```
  - imageChangeParams:
      automatic: true
      containerNames:
      - webhookj3
      from:
        kind: ImageStreamTag
        name: webhookj3:v6
        namespace: emsproject
      lastTriggeredImage:
quay.io/manasmohsin/webhook_j3@sha256:0cfa40a5572aa08dec1b290a74f56baf7e2a3538c702
fb38d16f47ff42340065
    type: ImageChange
status:
  availableReplicas: 1
  conditions:
  - lastTransitionTime: "2020-04-23T06:42:54Z"
    lastUpdateTime: "2020-04-23T06:42:54Z"
    message: Deployment config has minimum availability.
    status: "True"
    type: Available
  - lastTransitionTime: "2020-04-23T06:43:39Z"
    lastUpdateTime: "2020-04-23T06:43:41Z"
    message: replication controller "webhookj3-3" successfully rolled out
    reason: NewReplicationControllerAvailable
    status: "True"
    type: Progressing
  details:
    causes:
    - type: ConfigChange
    message: config change
  latestVersion: 3
  observedGeneration: 4
  readyReplicas: 1
  replicas: 1
  unavailableReplicas: 0
  updatedReplicas: 1
```

Find configmap from the Red Hat OpenShift console, as shown in Figure 4-24.

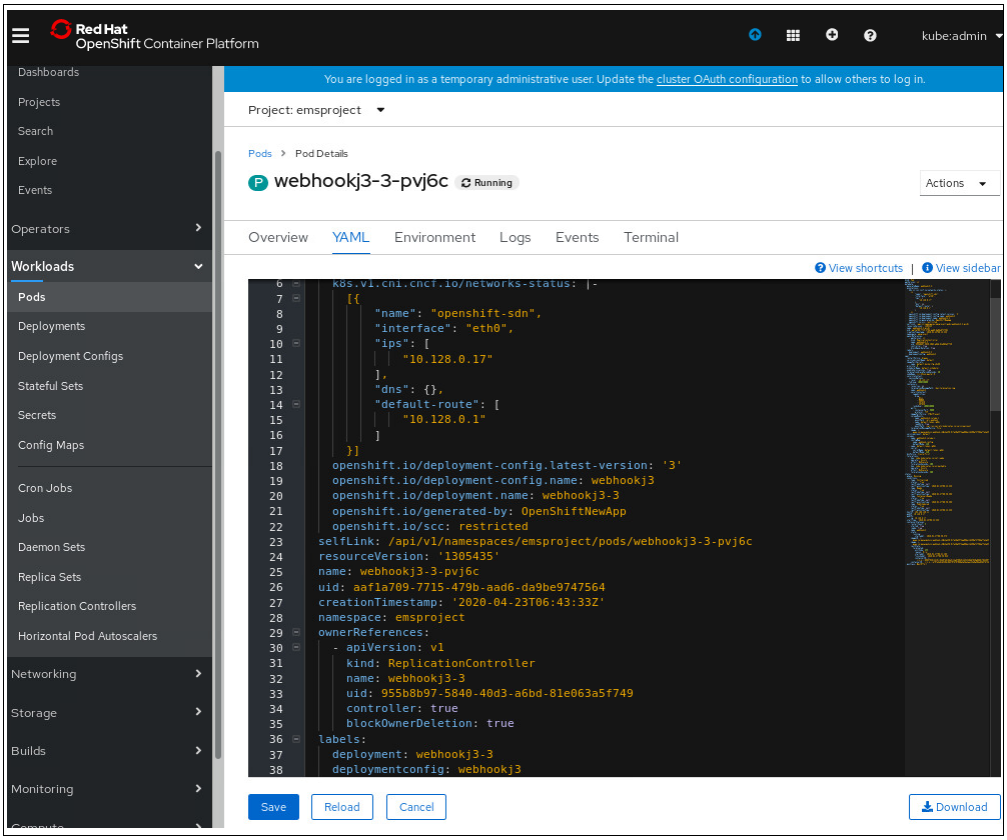


Figure 4-24 Red Hat OpenShift web console: $oc expose svc/webhookj3

Update the alertmanager if the webhook URL is not configured correctly.

Click **alertmanager** → **status** to find the URL in the Status window (see Figure 4-25).

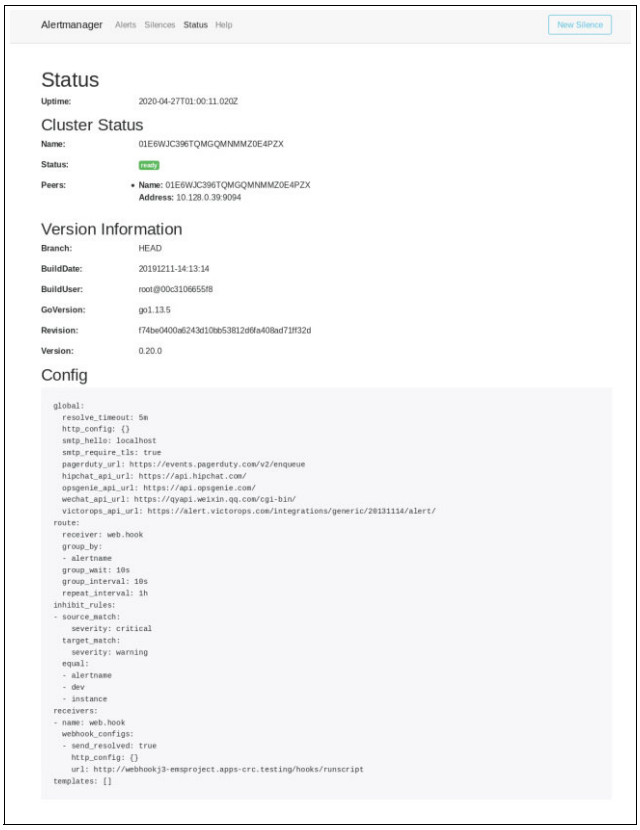


Figure 4-25 alertmanager status

4.2.7 Creating an ELK in Red Hat OpenShift

This section describes how to create an ELK deployment.

Elasticsearch deployment

To create an Elasticsearch application, run the following command:

```
$oc import-image docker.elastic.co/elasticsearch/elasticsearch:7.6.2 --confirm
```

For example:

```
[user2@codeReady html]$ oc get is elasticsearch
NAME IMAGE REPOSITORY TAGS UPDATED
elasticsearch   image-registry.openshift-image-registry.svc:5000/emsproject/elasticsearch
7.6.2   33 seconds ago

$oc new-app -e "discovery.type=single-node" \
    docker.elastic.co/elasticsearch/elasticsearch:7.6.2

$oc set volume dc/elasticsearch --add --mount-path=/usr/share/elasticsearch/data
--name=elasticsearch-volume-1 -t pvc --claim-name=pvc-elasticsearch-1
--claim-size=100G --overwrite

$oc expose svc elasticsearch
$oc get route elasticsearch
```

For example:

```
[user2@codeReady ~]$ oc get route elasticsearch
NAME        HOST/PORT PATH   SERVICES        PORT       TERMINATION   WILDCARD
elasticsearch elasticsearch-emsproject.apps-crc.testing elasticsearch 9200-tcp None
```

Kibana deployment

To create the Kibana application, the Elasticsearch URL is required as an input parameter (see Example 4-12):

```
$oc new-app -e ELASTICSEARCH_HOST=[ELASTIC_ROUTE]
docker.elastic.co/kibana/kibana:7.6.0
```

Example 4-12 Create Kibana application

```
$oc new-app -e
ELASTICSEARCH_HOST="http://elasticsearch-emsproject.apps-crc.testing/"
docker.elastic.co/kibana/kibana:7.6.0
```

```
$oc expose svc kibana
$oc get route kibana
```

For example:

```
[user2@codeReady tmp]$ oc get route kibana
NAME     HOST/PORT        PATH   SERVICES           PORT    TERMINATION   WILDCARD
kibana   kibana-emsproject.apps-crc.testing kibana 5601-tcp    None
```

Metricbeat deployment

To create Metricbeat to push the data from Prometheus to Elasticsearch, run the following command:

```
$oc create configmap metricbeat-config \
--from-file=/home/user2/metricbeat/metricbeat.yml
```

Example 4-13 shows the content of `metricbeat.yml`.

Example 4-13 metricbeat.yml file

```
[user2@codeReady tmp]$ cat metricbeat.yml
metricbeat.modules:
- module: system
  period: 10s
  metricsets:
    - cpu
    - load
    - memory
    - network
    - process
    - process_summary
    - socket_summary
    #- entropy
    #- core
    #- diskio
    #- socket
    #- services
  process.include_top_n:
    by_cpu: 5      # include top 5 processes by CPU
    by_memory: 5   # include top 5 processes by memory
```

```
- module: system
  period: 1m
  metricsets:
    - filesystem
    - fsstat
  processors:
  - drop_event.when.regexp:
      system.filesystem.mount_point: '^/(sys|cgroup|proc|dev|etc|host|lib)($|/)'
- module: system
  period: 15m
  metricsets:
    - uptime
- module: prometheus
  period: 10s
  hosts: ["http://prometheus-emsproject.apps-crc.testing"]
  metrics_path: /metrics
- module: prometheus
  metricsets: ["collector"]
  enabled: true
  period: 10s
  hosts: ["http://prometheus-emsproject.apps-crc.testing"]
  metrics_path: /metrics
- module: prometheus
  period: 10s
  hosts: ["http://prometheus-emsproject.apps-crc.testing"]
  metrics_path: '/federate'
  query:
    'match[]': '{__name__!=""}'
processors:
- add_cloud_metadata: ~
- add_docker_metadata: ~
output.elasticsearch:
  hosts: '${ELASTICSEARCH_HOSTS:elasticsearch:9200}'
  username: '${ELASTICSEARCH_USERNAME:}'
  password: '${ELASTICSEARCH_PASSWORD:}'
```

You must add the serviceAccountName in the deployment descriptor:

```
$oc new-app -e
output.elasticsearch.hosts="http://elasticsearch-emsproject.apps-crc.testing/"   \
 -e setup.kibana.host=”http://elasticsearch-emsproject.apps-crc.testing/”\
 docker.elastic.co/beats/metricbeat:7.6.2
```

To configure configmap in `dc/metricbeat`, run the **`oc set volume`**, command as shown in Example 4-14.

Example 4-14 Configure configmap

```
$oc set volume dc/metricbeat --add
--mount-path=/usr/share/metricbeat/metricbeat.yml \
   --sub-path=metricbeat.yml \
  --name=metricbeat-volume-1 -t configmap --configmap-name=metricbeat-config
```

From the `metricbeat.yml` file, two modules are enabled: System and Prometheus.

Verify that the respective module data is ingested or not using the discover feature of Kibana, as shown in Figure 4-26 and Figure 4-27.

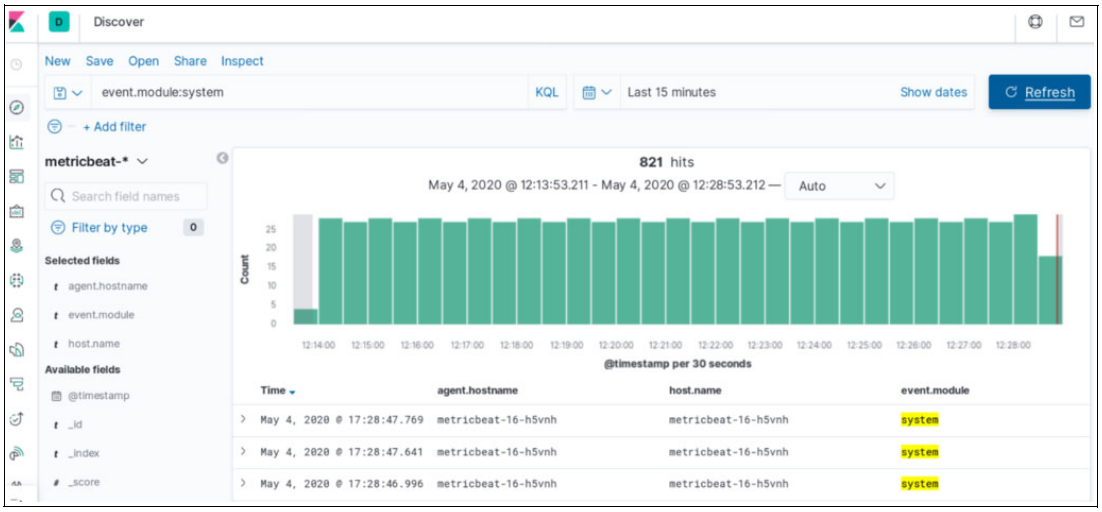


Figure 4-26 System module data

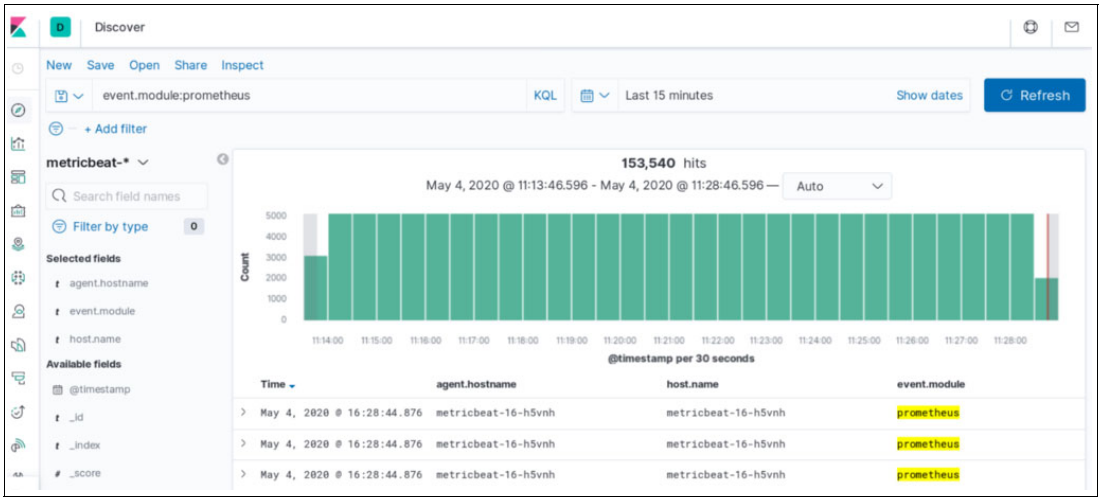


Figure 4-27 Prometheus module data

4.3 Monitoring by using Prometheus and Grafana

This section describes monitoring by using Prometheus and Grafana.

4.3.1 Prometheus

Prometheus is a systems and service time-series monitoring system. It is 100% open source and community driven. It collects metrics from configured targets at specified intervals, evaluates rule expressions, displays the results, and triggers alerts if some condition is observed to be true.

Note: For more information, including documentation, examples, and guides, see this website.

Prometheus includes the following main distinguishing features as compared to other monitoring systems:

- A multi-dimensional data model (timeseries that are defined by metric name and set of key and value dimensions).
- A flexible query language to take advantage of this dimensionality.
- No dependency on distributed storage; single-server nodes are autonomous.
- TimeSeries collection occurs by way of a pull model over HTTP.
- Pushing timeseries is supported by way of an intermediary gateway.
- Targets are discovered by way of service discovery or static configuration.
- Multiple modes of graphing and dashboarding support.
- Support for hierarchical and horizontal federation.

Figure 4-28 shows an architectural overview of Prometheus.

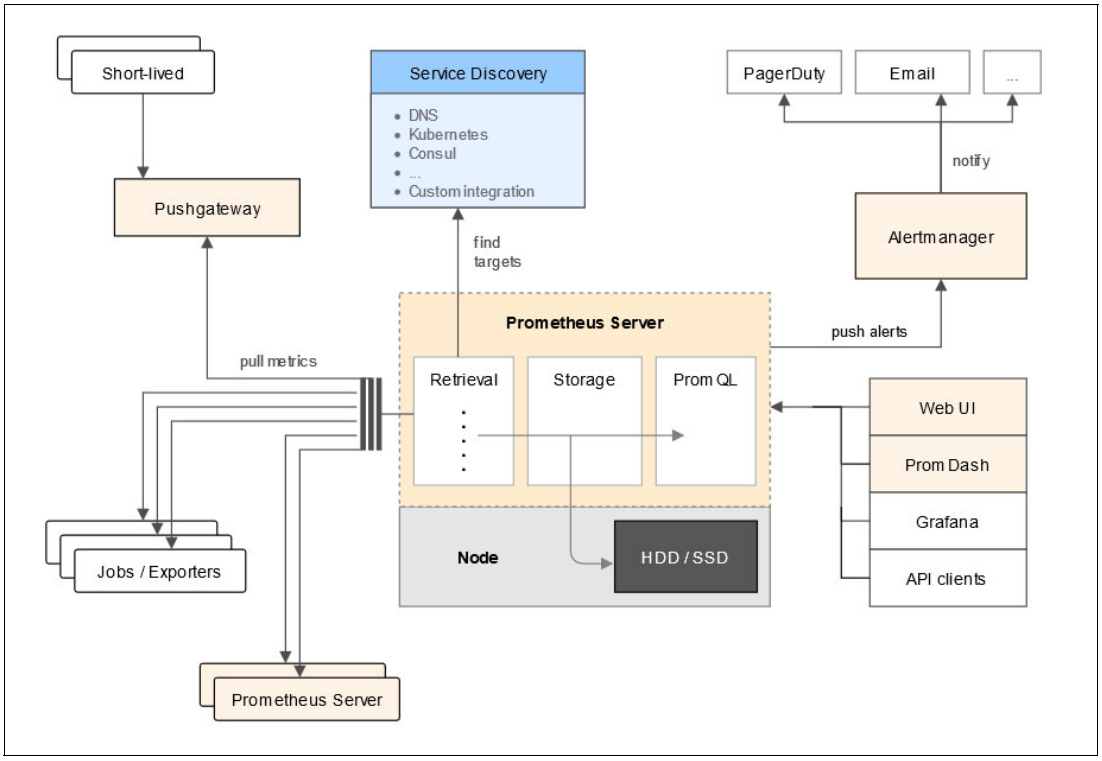


Figure 4-28 Architecture overview

4.3.2 Installing Prometheus server

Complete the following steps to install Prometheus V2.17.1 on Red Hat Enterprise Linux server V7.7:

1. Create a system user and group for Prometheus by running the following command (see Figure 4-29):

   ```
   # useradd -m -s /bin/false prometheus
   ```

```
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# df -k
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs         1926448       0   1926448   0% /dev
tmpfs            1937316       0   1937316   0% /dev/shm
tmpfs            1937316    8768   1928548   1% /run
tmpfs            1937316       0   1937316   0% /sys/fs/cgroup
/dev/xvda2      24638844 2090648  21289956   9% /
/dev/xvda1        999320  131352    815540  14% /boot
tmpfs             387464       0    387464   0% /run/user/0
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# pwd
/root
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# useradd -m -s /bin/false prometheus
[root@prometheus ~]#
[root@prometheus ~]#
```

Figure 4-29 Create Prometheus user and group

2. Check that the user was created by running the following command (see Figure 4-30):

   ```
   # id prometheus
   ```

```
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# id prometheus
uid=1000(prometheus) gid=1000(prometheus) groups=1000(prometheus)
[root@prometheus ~]#
[root@prometheus ~]#
```

Figure 4-30 Prometheus ID

3. Create the following directories and provide the required permissions: Create `prometheus` configuration directory under `/etc/` (see Figure 4-31) and data directory under `/var/lib/` (see Figure 4-32); then, change the directory permissions (see Figure 4-33).

   ```
   # mkdir /etc/prometheus
   ```

```
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# mkdir /etc/prometheus
You have mail in /var/spool/mail/root
[root@prometheus ~]#
[root@prometheus ~]#
```

Figure 4-31 Prometheus configuration directory

   ```
   # mkdir /var/lib/prometheus
   ```

```
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# mkdir /var/lib/prometheus
```

Figure 4-32 Prometheus data directory

   ```
   # chown prometheus:prometheus /var/lib/prometheus
   ```

```
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]# chown prometheus:prometheus /var/lib/prometheus/
[root@prometheus prometheus]#
[root@prometheus prometheus]#
```

Figure 4-33 Directory permissions

4. Run the **curl** command to download the latest release of Prometheus archive and extract it to get the binary files in a Red Hat Enterprise Linux server, as shown in Figure 4-34.

```
# curl -s https://api.github.com/repos/prometheus/prometheus/releases/latest | grep browser_download_url | grep linux-amd64 | cut -d '"' -f 4 | wget -qi -
```

```
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# curl -s https://api.github.com/repos/prometheus/prometheus/releases/latest | grep browser_download_url | grep linux-amd64 | cut -d '"' -
f 4 | wget -qi -
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]#
```

Figure 4-34 Download Prometheus archive and extract it

5. Validate the downloaded `.tar` file, as shown in Figure 4-35.

```
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# ls -l
total 58796
-rw-r--r--. 1 root root 60140415 Mar 26 13:27 prometheus-2.17.1.linux-amd64.tar.gz
[root@prometheus ~]#
[root@prometheus ~]#
```

Figure 4-35 Validate the tar file

6. Extract the downloaded archive `.tar` file and validate the content files by running the following command (see Figure 4-36):

```
# tar -xvf prometheus-2.17.1.linux-amd64.tar.gz
```

```
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# tar -xvf prometheus-2.17.1.linux-amd64.tar.gz
prometheus-2.17.1.linux-amd64/
prometheus-2.17.1.linux-amd64/NOTICE
prometheus-2.17.1.linux-amd64/LICENSE
prometheus-2.17.1.linux-amd64/prometheus.yml
prometheus-2.17.1.linux-amd64/prometheus
prometheus-2.17.1.linux-amd64/promtool
prometheus-2.17.1.linux-amd64/console_libraries/
prometheus-2.17.1.linux-amd64/console_libraries/menu.lib
prometheus-2.17.1.linux-amd64/console_libraries/prom.lib
prometheus-2.17.1.linux-amd64/consoles/
prometheus-2.17.1.linux-amd64/consoles/prometheus-overview.html
prometheus-2.17.1.linux-amd64/consoles/index.html.example
prometheus-2.17.1.linux-amd64/consoles/node-cpu.html
prometheus-2.17.1.linux-amd64/consoles/node-overview.html
prometheus-2.17.1.linux-amd64/consoles/node.html
prometheus-2.17.1.linux-amd64/consoles/node-disk.html
prometheus-2.17.1.linux-amd64/consoles/prometheus.html
prometheus-2.17.1.linux-amd64/tsdb
[root@prometheus ~]#
[root@prometheus ~]#
```

Figure 4-36 Extract the archive

7. Validate the extracted directory, as shown in Figure 4-37.

```
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# ls -l
total 58800
drwxr-xr-x. 4 3434 3434     4096 Mar 26 13:24 prometheus-2.17.1.linux-amd64
-rw-r--r--. 1 root root 60140415 Mar 26 13:27 prometheus-2.17.1.linux-amd64.tar.gz
[root@prometheus ~]#
```

Figure 4-37 Validate the directory

8. Check the content that was created, as shown in Figure 4-38.

```
[root@prometheus ~]#
[root@prometheus ~]#
[root@prometheus ~]# cd prometheus-2.17.1.linux-amd64
[root@prometheus prometheus-2.17.1.linux-amd64]# ls -l
total 143072
drwxr-xr-x. 2 3434 3434     4096 Mar 26 13:22 console_libraries
drwxr-xr-x. 2 3434 3434     4096 Mar 26 13:22 consoles
-rw-r--r--. 1 3434 3434    11357 Mar 26 13:22 LICENSE
-rw-r--r--. 1 3434 3434     3184 Mar 26 13:22 NOTICE
-rwxr-xr-x. 1 3434 3434 84338005 Mar 26 11:20 prometheus
-rw-r--r--. 1 3434 3434      926 Mar 26 13:22 prometheus.yml
-rwxr-xr-x. 1 3434 3434 48235996 Mar 26 11:22 promtool
-rwxr-xr-x. 1 3434 3434 13732141 Mar 26 11:22 tsdb
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]#
```

Figure 4-38 Check the content

9. Place these directories into your binary $PATH location and under the configuration folder. Copy the directories (see Figure 4-39) and place them as follows (see Figure 4-40):

```
# cp prometheus promtool /usr/local/bin/
```

```
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]# cp prometheus promtool /usr/local/bin/
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]#
```

Figure 4-39 Copy the directories

```
# cp -r consoles/ console_libraries/ /etc/prometheus/
```

```
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]# cp -r consoles/ console_libraries/ /etc/prometheus/
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]# _
```

Figure 4-40 Place them in the configuration folder

10. Copy the yml file from the extracted directory to the Prometheus configuration directory (see Figure 4-41):

```
# cp prometheus.yml /etc/prometheus/
```

```
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]# cp prometheus.yml /etc/prometheus/
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]#
```

Figure 4-41 Copy the yml file to the Prometheus configuration

11. Validate the contents of the directory (see Figure 4-42):

```
# ls -l /etc/prometheus/prometheus.yml
```

```
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]# ls -l /etc/prometheus/prometheus.yml
-rw-r--r--. 1 root root 926 Apr  6 03:53 /etc/prometheus/prometheus.yml
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]#
```

Figure 4-42 List the directory contents

12. If needed, edit the sample configuration file by running the **vi** editor (see Figure 4-43):

```
# vi /etc/prometheus/prometheus.yml
```

```
# my global config
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['localhost:9090']
```

Figure 4-43 Edit the prometheus.yml file

13. Create `systemd` service for Prometheus. Edit the `/etc/system/system/prometheus.service` file as follows (Figure 4-44):

```
# vi /etc/systemd/system/prometheus.service
```

```
[Unit]
Description=Prometheus
Documentation=https://prometheus.io/docs/introduction/overview/
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
Environment="GOMAXPROCS=2"
User=prometheus
Group=prometheus
ExecReload=/bin/kill -HUP $MAINPID
ExecStart=/usr/local/bin/prometheus \
  --config.file=/etc/prometheus/prometheus.yml \
  --storage.tsdb.path=/var/lib/prometheus \
  --web.console.templates=/etc/prometheus/consoles \
  --web.console.libraries=/etc/prometheus/console_libraries \
  --web.listen-address=0.0.0.0:9090 \
  --web.external-url=

SyslogIdentifier=prometheus
Restart=always

[Install]
WantedBy=multi-user.target
```

Figure 4-44 Edit prometheus.service file

14. Set the required directory permissions as shown in Figure 4-45, Figure 4-46, Figure 4-47, and Figure 4-48.

```
# cd /etc/prometheus
```

```
[root@prometheus prometheus-2.17.1.linux-amd64]#
[root@prometheus prometheus-2.17.1.linux-amd64]# cd /etc/prometheus
[root@prometheus prometheus]# ls -l
total 12
drwxr-xr-x. 2 root root 4096 Apr  6 03:49 console_libraries
drwxr-xr-x. 2 root root 4096 Apr  6 03:49 consoles
-rw-r--r--. 1 root root  926 Apr  6 03:53 prometheus.yml
[root@prometheus prometheus]#
[root@prometheus prometheus]#
```

Figure 4-45 Check the contents of the /etc/prometheus directory

```
# chown -R prometheus:prometheus /etc/prometheus
```

```
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]# chown -R prometheus:prometheus /etc/prometheus
[root@prometheus prometheus]#
```

Figure 4-46 Change the directory permissions

```
# chmod -R 775 /etc/Prometheus
```

```
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]# chown -R prometheus:prometheus /etc/prometheus
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]# chmod -R 775 /etc/prometheus
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]# ls -l
total 12
drwxrwxr-x. 2 prometheus prometheus 4096 Apr  6 03:49 console_libraries
drwxrwxr-x. 2 prometheus prometheus 4096 Apr  6 03:49 consoles
-rwxrwxr-x. 1 prometheus prometheus  926 Apr  6 03:53 prometheus.yml
[root@prometheus prometheus]#
[root@prometheus prometheus]#
```

Figure 4-47 Modify the permission access

```
# chown -R prometheus:prometheus /var/lib/prometheus/
```

```
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]# chown -R prometheus:prometheus /var/lib/prometheus/
[root@prometheus prometheus]#
[root@prometheus prometheus]#
```

Figure 4-48 Modify the /var/lib/prometheus/ directory access

15. Reload the daemon and start the service (see Figure 4-49 and Figure 4-50):

```
# systemctl daemon-reload
# systemctl start prometheus
```

```
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]# systemctl daemon-reload
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]# systemctl start prometheus
[root@prometheus prometheus]#
[root@prometheus prometheus]#
```

Figure 4-49 Reload daemon and start service

```
# systemctl status prometheus.service
```

```
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]# systemctl status prometheus.service
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2020-04-06 04:16:56 CDT; 1min 2s ago
     Docs: https://prometheus.io/docs/introduction/overview/
 Main PID: 21526 (prometheus)
   CGroup: /system.slice/prometheus.service
           └─21526 /usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml --storage.tsdb.path=/var/lib/prometheus --web.console.templates=...

Apr 06 04:16:56 prometheus.redbooks.info prometheus[21526]: level=info ts=2020-04-06T09:16:56.615Z caller=main.go:667 msg="Starting TSDB ..."
Apr 06 04:16:56 prometheus.redbooks.info prometheus[21526]: level=info ts=2020-04-06T09:16:56.615Z caller=web.go:514 component=web msg="Start liste....0:9090
Apr 06 04:16:56 prometheus.redbooks.info prometheus[21526]: level=info ts=2020-04-06T09:16:56.618Z caller=head.go:575 component=tsdb msg="replaying...awhile"
Apr 06 04:16:56 prometheus.redbooks.info prometheus[21526]: level=info ts=2020-04-06T09:16:56.620Z caller=head.go:624 component=tsdb msg="WAL segme...gment=0
Apr 06 04:16:56 prometheus.redbooks.info prometheus[21526]: level=info ts=2020-04-06T09:16:56.620Z caller=head.go:627 component=tsdb msg="WAL repla...24569ms
Apr 06 04:16:56 prometheus.redbooks.info prometheus[21526]: level=info ts=2020-04-06T09:16:56.622Z caller=main.go:683 fs_type=EXT4_SUPER_MAGIC
Apr 06 04:16:56 prometheus.redbooks.info prometheus[21526]: level=info ts=2020-04-06T09:16:56.622Z caller=main.go:684 msg="TSDB started"
Apr 06 04:16:56 prometheus.redbooks.info prometheus[21526]: level=info ts=2020-04-06T09:16:56.622Z caller=main.go:788 msg="Loading configuration fi...eus.yml
Apr 06 04:16:56 prometheus.redbooks.info prometheus[21526]: level=info ts=2020-04-06T09:16:56.624Z caller=main.go:816 msg="Completed loading of con...eus.yml
Apr 06 04:16:56 prometheus.redbooks.info prometheus[21526]: level=info ts=2020-04-06T09:16:56.624Z caller=main.go:635 msg="Server is ready to recei...uests."
Hint: Some lines were ellipsized, use -l to show in full.
[root@prometheus prometheus]#
[root@prometheus prometheus]#
[root@prometheus prometheus]#
```

Figure 4-50 Check service status

16. Check the Prometheus server in your browser. By default, Prometheus is running on port 9090. Verify the tool installation (see Figure 4-51):

```
http://<server-ip>:9090
```

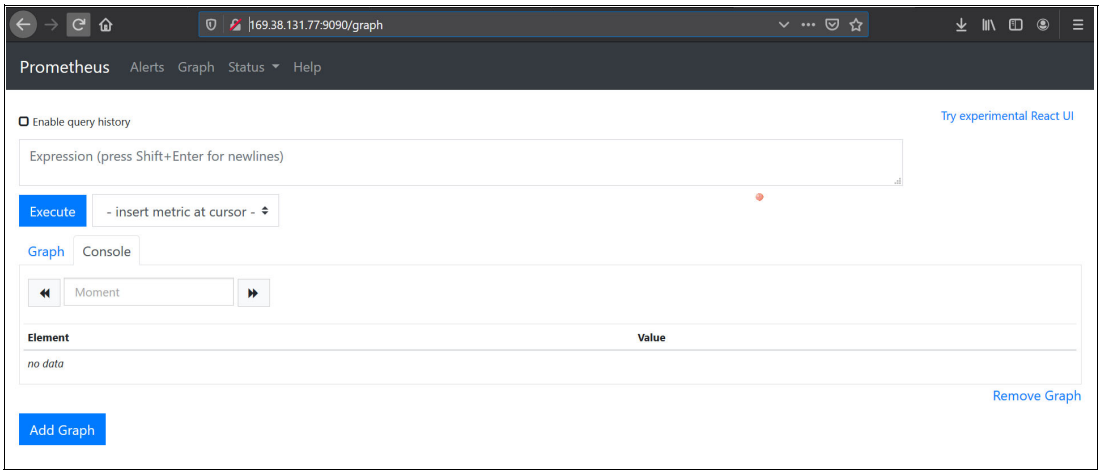


Figure 4-51 Prometheus server

Complete the following steps to install Prometheus as Red Hat Enterprise Linux container:

1. Podman is a tool for running Linux containers. Subscribe, then enable extras channel to install Podman (see Figure 4-52 on page 282):

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-7-server-extras-rpms
# subscription-manager repos --enable=rhel-7-server-optional-rpms
```

```
[root@ems ~]#
[root@ems ~]# subscription-manager repos --enable=rhel-7-server-rpms
Repository 'rhel-7-server-rpms' is enabled for this system.
[root@ems ~]#
[root@ems ~]#
[root@ems ~]#
[root@ems ~]# subscription-manager repos --enable=rhel-7-server-extras-rpms
Repository 'rhel-7-server-extras-rpms' is enabled for this system.
[root@ems ~]#
[root@ems ~]#
[root@ems ~]# subscription-manager repos --enable=rhel-7-server-optional-rpms
Repository 'rhel-7-server-optional-rpms' is enabled for this system.
[root@ems ~]#
```

Figure 4-52 Enable Podman

2. Install Podman by using the following command (for the installation output, see Figure 4-53, Figure 4-54, Figure 4-55 on page 283, Figure 4-56 on page 283, and Figure 4-57 on page 284):

   ```
   # yum install podman -y
   ```

```
[root@ems ~]#
[root@ems ~]# yum install podman -y
Loaded plugins: product-id, search-disabled-repos, subscription-manager
rhel-7-server-extras-rpms                                                          | 2.0 kB  00:00:00
rhel-7-server-optional-rpms                                                        | 1.8 kB  00:00:00
rhel-7-server-rpms                                                                 | 2.0 kB  00:00:00
rhel-7-server-supplementary-rpms                                                   | 2.0 kB  00:00:00
(1/3): rhel-7-server-extras-rpms/x86_64/group                                      |  147 B  00:00:00
(2/3): rhel-7-server-extras-rpms/x86_64/updateinfo                                 | 230 kB  00:00:00
(3/3): rhel-7-server-extras-rpms/x86_64/primary                                    | 389 kB  00:00:00
rhel-7-server-extras-rpms                                                                    1275/1275
Resolving Dependencies
--> Running transaction check
---> Package podman.x86_64 0:1.6.4-16.el7_8 will be installed
--> Processing Dependency: slirp4netns >= 0.4.0-1 for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: runc >= 1.0.0-57 for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: containers-common >= 0.1.29-3 for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: containernetworking-plugins >= 0.8.1-1 for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: nftables for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: libseccomp for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: fuse-overlayfs for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: container-selinux for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: conmon for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: libseccomp.so.2()(64bit) for package: podman-1.6.4-16.el7_8.x86_64
--> Running transaction check
---> Package conmon.x86_64 2:2.0.8-1.el7 will be installed
---> Package container-selinux.noarch 2:2.119.1-1.c57a6f9.el7 will be installed
--> Processing Dependency: policycoreutils-python for package: 2:container-selinux-2.119.1-1.c57a6f9.el7.noarch
---> Package containernetworking-plugins.x86_64 0:0.8.3-2.el7 will be installed
---> Package containers-common.x86_64 1:0.1.40-7.el7_8 will be installed
---> Package fuse-overlayfs.x86_64 0:0.7.2-6.el7_8 will be installed
--> Processing Dependency: libfuse3.so.3(FUSE_3.2)(64bit) for package: fuse-overlayfs-0.7.2-6.el7_8.x86_64
--> Processing Dependency: libfuse3.so.3(FUSE_3.0)(64bit) for package: fuse-overlayfs-0.7.2-6.el7_8.x86_64
--> Processing Dependency: libfuse3.so.3()(64bit) for package: fuse-overlayfs-0.7.2-6.el7_8.x86_64
---> Package libseccomp.x86_64 0:2.3.1-4.el7 will be installed
---> Package nftables.x86_64 1:0.8-14.el7 will be installed
--> Processing Dependency: libnftnl.so.7(LIBNFTNL_5)(64bit) for package: 1:nftables-0.8-14.el7.x86_64
--> Processing Dependency: libnftnl.so.7()(64bit) for package: 1:nftables-0.8-14.el7.x86_64
---> Package runc.x86_64 0:1.0.0-67.rc10.el7_8 will be installed
--> Processing Dependency: criu for package: runc-1.0.0-67.rc10.el7_8.x86_64
---> Package slirp4netns.x86_64 0:0.4.3-4.el7_8 will be installed
--> Running transaction check
---> Package criu.x86_64 0:3.12-2.el7 will be installed
--> Processing Dependency: libprotobuf-c.so.1(LIBPROTOBUF_C_1.0.0)(64bit) for package: criu-3.12-2.el7.x86_64
--> Processing Dependency: libprotobuf-c.so.1()(64bit) for package: criu-3.12-2.el7.x86_64
```

Figure 4-53 Install Podman

```
--> Processing Dependency: libnet.so.1()(64bit) for package: criu-3.12-2.el7.x86_64
---> Package fuse3-libs.x86_64 0:3.6.1-4.el7 will be installed
---> Package libnftnl.x86_64 0:1.0.8-3.el7 will be installed
---> Package policycoreutils-python.x86_64 0:2.5-34.el7 will be installed
--> Processing Dependency: policycoreutils = 2.5-34.el7 for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: setools-libs >= 3.3.8-4 for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: libsemanage-python >= 2.5-14 for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: audit-libs-python >= 2.1.3-4 for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: python-IPy for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: libqpol.so.1(VERS_1.4)(64bit) for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: libqpol.so.1(VERS_1.2)(64bit) for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: libcgroup for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: libapol.so.4(VERS_4.0)(64bit) for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: checkpolicy for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: libqpol.so.1()(64bit) for package: policycoreutils-python-2.5-34.el7.x86_64
--> Processing Dependency: libapol.so.4()(64bit) for package: policycoreutils-python-2.5-34.el7.x86_64
--> Running transaction check
---> Package audit-libs-python.x86_64 0:2.8.5-4.el7 will be installed
---> Package checkpolicy.x86_64 0:2.5-8.el7 will be installed
---> Package libcgroup.x86_64 0:0.41-21.el7 will be installed
---> Package libnet.x86_64 0:1.1.6-7.el7 will be installed
---> Package libsemanage-python.x86_64 0:2.5-14.el7 will be installed
---> Package policycoreutils.x86_64 0:2.5-33.el7 will be updated
---> Package policycoreutils.x86_64 0:2.5-34.el7 will be an update
---> Package protobuf-c.x86_64 0:1.0.2-3.el7 will be installed
---> Package python-IPy.noarch 0:0.75-6.el7 will be installed
---> Package setools-libs.x86_64 0:3.3.8-4.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=================================================================================================================
 Package                            Arch            Version                       Repository                 Size
=================================================================================================================
Installing:
 podman                             x86_64          1.6.4-16.el7_8                rhel-7-server-extras-rpms  13 M
Installing for dependencies:
 audit-libs-python                  x86_64          2.8.5-4.el7                   rhel-7-server-rpms         77 k
 checkpolicy                        x86_64          2.5-8.el7                     rhel-7-server-rpms        295 k
 conmon                             x86_64          2:2.0.8-1.el7                 rhel-7-server-extras-rpms  31 k
 container-selinux                  noarch          2:2.119.1-1.c57a6f9.el7       rhel-7-server-extras-rpms  40 k
 containernetworking-plugins        x86_64          0.8.3-2.el7                   rhel-7-server-extras-rpms  20 M
 containers-common                  x86_64          1:0.1.40-7.el7_8              rhel-7-server-extras-rpms  42 k
 criu                               x86_64          3.12-2.el7                    rhel-7-server-rpms        453 k
 fuse-overlayfs                     x86_64          0.7.2-6.el7_8                 rhel-7-server-extras-rpms  55 k
```

Figure 4-54 Podman installation continues

```
 fuse3-libs                                x86_64              3.6.1-4.el7                 rhel-7-server-extras-rpms                82 k
 libcgroup                                 x86_64              0.41-21.el7                 rhel-7-server-rpms                       66 k
 libnet                                    x86_64              1.1.6-7.el7                 rhel-7-server-optional-rpms              59 k
 libnftnl                                  x86_64              1.0.8-3.el7                 rhel-7-server-rpms                       78 k
 libseccomp                                x86_64              2.3.1-4.el7                 rhel-7-server-rpms                       56 k
 libsemanage-python                        x86_64              2.5-14.el7                  rhel-7-server-rpms                      113 k
 nftables                                  x86_64              1:0.8-14.el7                rhel-7-server-rpms                      186 k
 policycoreutils-python                    x86_64              2.5-34.el7                  rhel-7-server-rpms                      457 k
 protobuf-c                                x86_64              1.0.2-3.el7                 rhel-7-server-rpms                       28 k
 python-IPy                                noarch              0.75-6.el7                  rhel-7-server-rpms                       32 k
 runc                                      x86_64              1.0.0-67.rc10.el7_8         rhel-7-server-extras-rpms               2.7 M
 setools-libs                              x86_64              3.3.8-4.el7                 rhel-7-server-rpms                      620 k
 slirp4netns                               x86_64              0.4.3-4.el7_8               rhel-7-server-extras-rpms                82 k
Updating for dependencies:
 policycoreutils                           x86_64              2.5-34.el7                  rhel-7-server-rpms                      917 k

Transaction Summary
================================================================================================================================================
Install  1 Package  (+21 Dependent packages)
Upgrade             (  1 Dependent package)

Total download size: 39 M
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
(1/23): checkpolicy-2.5-8.el7.x86_64.rpm                                                 | 295 kB  00:00:00
(2/23): audit-libs-python-2.8.5-4.el7.x86_64.rpm                                         |  77 kB  00:00:00
(3/23): conmon-2.0.8-1.el7.x86_64.rpm                                                    |  31 kB  00:00:00
(4/23): container-selinux-2.119.1-1.c57a6f9.el7.noarch.rpm                               |  40 kB  00:00:00
(5/23): containers-common-0.1.40-7.el7_8.x86_64.rpm                                      |  42 kB  00:00:00
(6/23): fuse-overlayfs-0.7.2-6.el7_8.x86_64.rpm                                          |  55 kB  00:00:00
(7/23): criu-3.12-2.el7.x86_64.rpm                                                       | 453 kB  00:00:00
(8/23): containernetworking-plugins-0.8.3-2.el7.x86_64.rpm                               |  20 MB  00:00:00
(9/23): fuse3-libs-3.6.1-4.el7.x86_64.rpm                                                |  82 kB  00:00:00
(10/23): libnftnl-1.0.8-3.el7.x86_64.rpm                                                 |  78 kB  00:00:00
(11/23): libseccomp-2.3.1-4.el7.x86_64.rpm                                               |  56 kB  00:00:00
(12/23): libnet-1.1.6-7.el7.x86_64.rpm                                                   |  59 kB  00:00:00
(13/23): libcgroup-0.41-21.el7.x86_64.rpm                                                |  66 kB  00:00:00
(14/23): libsemanage-python-2.5-14.el7.x86_64.rpm                                        | 113 kB  00:00:00
(15/23): nftables-0.8-14.el7.x86_64.rpm                                                  | 186 kB  00:00:00
(16/23): policycoreutils-2.5-34.el7.x86_64.rpm                                           | 917 kB  00:00:00
(17/23): policycoreutils-python-2.5-34.el7.x86_64.rpm                                    | 457 kB  00:00:00
(18/23): python-IPy-0.75-6.el7.noarch.rpm                                                |  32 kB  00:00:00
(19/23): protobuf-c-1.0.2-3.el7.x86_64.rpm                                               |  28 kB  00:00:00
(20/23): podman-1.6.4-16.el7_8.x86_64.rpm                                                |  13 MB  00:00:00
(21/23): setools-libs-3.3.8-4.el7.x86_64.rpm                                             | 620 kB  00:00:00
```

Figure 4-55 Podman install continues

```
(22/23): slirp4netns-0.4.3-4.el7_8.x86_64.rpm                                            |  82 kB  00:00:00
(23/23): runc-1.0.0-67.rc10.el7_8.x86_64.rpm                                             | 2.7 MB  00:00:00
------------------------------------------------------------------------------------------------------------------------------------------------
Total                                                                    20 MB/s |  39 MB  00:00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : libseccomp-2.3.1-4.el7.x86_64                                                                                    1/24
  Installing : slirp4netns-0.4.3-4.el7_8.x86_64                                                                                 2/24
  Updating   : policycoreutils-2.5-34.el7.x86_64                                                                                3/24
  Installing : containernetworking-plugins-0.8.3-2.el7.x86_64                                                                   4/24
  Installing : libcgroup-0.41-21.el7.x86_64                                                                                     5/24
  Installing : libnftnl-1.0.8-3.el7.x86_64                                                                                      6/24
  Installing : 1:nftables-0.8-14.el7.x86_64                                                                                     7/24
  Installing : 2:conmon-2.0.8-1.el7.x86_64                                                                                      8/24
  Installing : libnet-1.1.6-7.el7.x86_64                                                                                        9/24
  Installing : libsemanage-python-2.5-14.el7.x86_64                                                                            10/24
  Installing : protobuf-c-1.0.2-3.el7.x86_64                                                                                   11/24
  Installing : criu-3.12-2.el7.x86_64                                                                                          12/24
  Installing : setools-libs-3.3.8-4.el7.x86_64                                                                                 13/24
  Installing : python-IPy-0.75-6.el7.noarch                                                                                    14/24
  Installing : checkpolicy-2.5-8.el7.x86_64                                                                                    15/24
  Installing : audit-libs-python-2.8.5-4.el7.x86_64                                                                            16/24
  Installing : policycoreutils-python-2.5-34.el7.x86_64                                                                        17/24
  Installing : 2:container-selinux-2.119.1-1.c57a6f9.el7.noarch                                                                18/24
  Installing : runc-1.0.0-67.rc10.el7_8.x86_64                                                                                 19/24
  Installing : fuse3-libs-3.6.1-4.el7.x86_64                                                                                   20/24
  Installing : fuse-overlayfs-0.7.2-6.el7_8.x86_64                                                                             21/24
  Installing : 1:containers-common-0.1.40-7.el7_8.x86_64                                                                       22/24
  Installing : podman-1.6.4-16.el7_8.x86_64                                                                                    23/24
  Cleanup    : policycoreutils-2.5-33.el7.x86_64                                                                               24/24
Loaded plugins: product-id, subscription-manager
  Verifying  : 2:container-selinux-2.119.1-1.c57a6f9.el7.noarch                                                                 1/24
  Verifying  : fuse-overlayfs-0.7.2-6.el7_8.x86_64                                                                              2/24
  Verifying  : 1:nftables-0.8-14.el7.x86_64                                                                                     3/24
  Verifying  : fuse3-libs-3.6.1-4.el7.x86_64                                                                                    4/24
  Verifying  : audit-libs-python-2.8.5-4.el7.x86_64                                                                             5/24
  Verifying  : checkpolicy-2.5-8.el7.x86_64                                                                                     6/24
  Verifying  : policycoreutils-2.5-34.el7.x86_64                                                                                7/24
  Verifying  : python-IPy-0.75-6.el7.noarch                                                                                     8/24
  Verifying  : libseccomp-2.3.1-4.el7.x86_64                                                                                    9/24
  Verifying  : policycoreutils-python-2.5-34.el7.x86_64                                                                        10/24
  Verifying  : setools-libs-3.3.8-4.el7.x86_64                                                                                 11/24
  Verifying  : protobuf-c-1.0.2-3.el7.x86_64                                                                                   12/24
```

Figure 4-56 Installation continues

```
  Verifying  : protobuf-c-1.0.2-3.el7.x86_64                                                                                  12/24
  Verifying  : runc-1.0.0-67.rc10.el7_8.x86_64                                                                                13/24
  Verifying  : 1:containers-common-0.1.40-7.el7_8.x86_64                                                                      14/24
  Verifying  : libsemanage-python-2.5-14.el7.x86_64                                                                           15/24
  Verifying  : slirp4netns-0.4.3-4.el7_8.x86_64                                                                               16/24
  Verifying  : criu-3.12-2.el7.x86_64                                                                                         17/24
  Verifying  : libnet-1.1.6-7.el7.x86_64                                                                                      18/24
  Verifying  : podman-1.6.4-16.el7_8.x86_64                                                                                   19/24
  Verifying  : 2:conmon-2.0.8-1.el7.x86_64                                                                                    20/24
  Verifying  : libnftnl-1.1.0.8-3.el7.x86_64                                                                                  21/24
  Verifying  : libcgroup-0.41-21.el7.x86_64                                                                                   22/24
  Verifying  : containernetworking-plugins-0.8.3-2.el7.x86_64                                                                 23/24
  Verifying  : policycoreutils-2.5-33.el7.x86_64                                                                              24/24
rhel-7-server-extras-rpms/x86_64/productid                                                                  | 2.1 kB  00:00:00
rhel-7-server-supplementary-rpms/7Server/x86_64/productid                                                   | 2.1 kB  00:00:00

Installed:
  podman.x86_64 0:1.6.4-16.el7_8

Dependency Installed:
  audit-libs-python.x86_64 0:2.8.5-4.el7              checkpolicy.x86_64 0:2.5-8.el7                      conmon.x86_64 2:2.0.8-1.el7
  container-selinux.noarch 2:2.119.1-1.c57a6f9.el7    containernetworking-plugins.x86_64 0:0.8.3-2.el7    containers-common.x86_64 1:0.1.40-7.el7_8
  criu.x86_64 0:3.12-2.el7                            fuse-overlayfs.x86_64 0:0.7.2-6.el7_8               fuse3-libs.x86_64 0:3.6.1-4.el7
  libcgroup.x86_64 0:0.41-21.el7                      libnet.x86_64 0:1.1.6-7.el7                         libnftnl.x86_64 0:1.0.8-3.el7
  libseccomp.x86_64 0:2.3.1-4.el7                     libsemanage-python.x86_64 0:2.5-14.el7              nftables.x86_64 1:0.8-14.el7
  policycoreutils-python.x86_64 0:2.5-34.el7          protobuf-c.x86_64 0:1.0.2-3.el7                     python-IPy.noarch 0:0.75-6.el7
  runc.x86_64 0:1.0.0-67.rc10.el7_8                   setools-libs.x86_64 0:3.3.8-4.el7                   slirp4netns.x86_64 0:0.4.3-4.el7_8

Dependency Updated:
  policycoreutils.x86_64 0:2.5-34.el7

Complete!
[root@ems ~]#
[root@ems ~]#
```

Figure 4-57 Podman install completes

3. Install the latest Prometheus container image (see Figure 4-58):

```
# podman pull prom/prometheus
```

```
[root@ems ~]#
[root@ems ~]#
[root@ems ~]# podman pull prom/prometheus
Trying to pull registry.access.redhat.com/prom/prometheus...
  name unknown: Repo not found
Trying to pull registry.fedoraproject.org/prom/prometheus...
  manifest unknown: manifest unknown
Trying to pull registry.centos.org/prom/prometheus...
  manifest unknown: manifest unknown
Trying to pull docker.io/prom/prometheus...
Getting image source signatures
Copying blob 626a2a3fee8c done
Copying blob e8af9b0860a9 done
Copying blob 10888ee4f64b done
Copying blob 0f8c40e1270f done
Copying blob 954c05febdeb done
Copying blob c28155f7eb79 done
Copying blob 1000ceb203a8 done
Copying blob 9d1b4f52acab done
Copying blob 59b2f7d48510 done
Copying blob e0f68e2bc6ce done
Copying blob c914ff888424 done
Copying blob 907ddadc7a02 done
Copying config 358a0d2395 done
Writing manifest to image destination
Storing signatures
358a0d2395fe711bb8258e8fb4b2d7865c0a9a6463969bcd1452ee8869ea6653
[root@ems ~]#
[root@ems ~]#
```

Figure 4-58 Install latest updates

4. Verify the container image (see Figure 4-59):

```
# podman images
```

```
[root@ems ~]# podman images
REPOSITORY                  TAG      IMAGE ID       CREATED       SIZE
docker.io/prom/prometheus   latest   358a0d2395fe   2 weeks ago   137 MB
[root@ems ~]#
[root@ems ~]#
[root@ems ~]#
```

Figure 4-59 Verify image

5. Run the container and verify the status (see Figure 4-60):

```
# podman run -d -p 9090:9090 prom/prometheus
# podman ps
```

```
[root@ems ~]#
[root@ems ~]#
[root@ems ~]# podman run -d  -p 9090:9090  prom/prometheus
9a2c1cd386673eba100539606733387b4b77ac16c85ea011b69f5a763343e766
[root@ems ~]#
[root@ems ~]#
[root@ems ~]# podman ps
CONTAINER ID  IMAGE                              COMMAND               CREATED        STATUS            PORTS                    NAMES
9a2c1cd38667  docker.io/prom/prometheus:latest  --config.file=/et...  9 seconds ago  Up 8 seconds ago  0.0.0.0:9090->9090/tcp  great_goodall
[root@ems ~]#
[root@ems ~]#
[root@ems ~]#
```

Figure 4-60 Run and verify status

6. Check the Prometheus server in your browser (see Figure 4-61):

   `http://<server-ip>:9090`

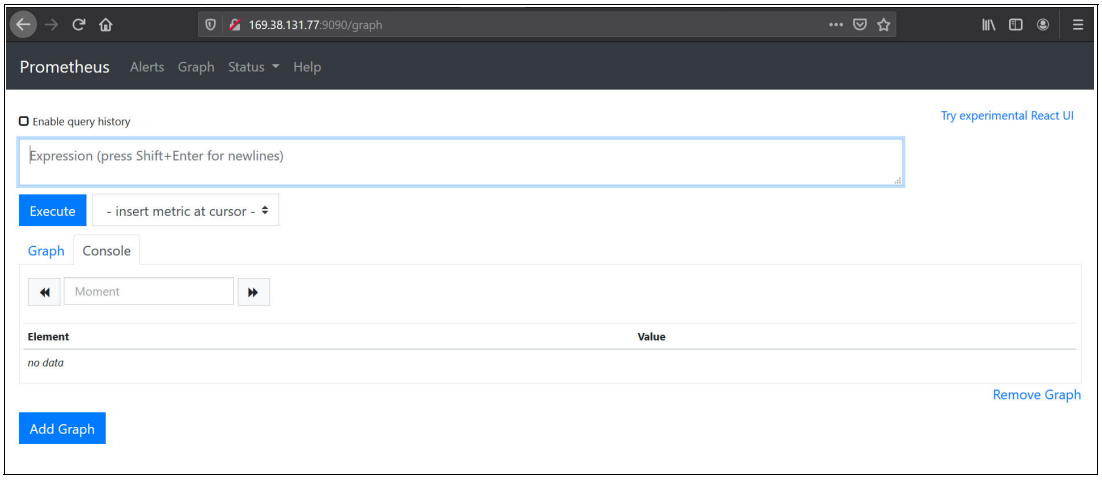


Figure 4-61 Prometheus server window

4.3.3 Installing node exporter (node exporter, ipmi exporter, mysql exporter, JMX exporter

This section describes installing node exporter and its components.

Installing the node exporter

Node exporters are components that send various hardware and kernel-related metrics from the system to the Prometheus server for monitoring. They are deployed at each client and configured to emit the data on a port of client system. Prometheus listens to the port and receives all the data.

Complete the following steps to download and install the node exporter on a Linux system:

1. SSH to the server, as shown in Figure 4-62.

```
rks@DESKTOP-L4EQ7H7:~$ ssh root@169.38.131.68
root@169.38.131.68's password:
Last failed login: Mon Apr  6 03:35:18 CDT 2020 from 193.142.146.21 on ssh:notty
There were 12 failed login attempts since the last successful login.
Last login: Mon Apr  6 02:05:06 2020 from 49.206.10.179
[root@node2-sql ~]#
```

Figure 4-62 Log in to the system

2. Run the following commands to create a folder and download the node exporter file into the folder, as shown in Figure 4-63 on page 286:

   ```
   mkdir <dir_name>
   cd <dir_name>
   wget https://bit.ly/3e5lK8M
   ```

```
[root@node2-sql tmp]# cd exporters/
[root@node2-sql exporters]# wget https://github.com/prometheus/node_exporter/releases/download/v0.18.1/node_exporter-0.18.1.linux-amd64.tar.gz
--2020-04-06 04:15:53--  https://github.com/prometheus/node_exporter/releases/download/v0.18.1/node_exporter-0.18.1.linux-amd64.tar.gz
Resolving github.com (github.com)... 13.234.176.102
Connecting to github.com (github.com)|13.234.176.102|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/9524057/5dc5df80-86f1-11e9-924c-ef392e7300e3?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKI
AIWNJYAX4CSVEH53A%2F20200406%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200406T091457Z&X-Amz-Expires=300&X-Amz-Signature=fea03235e4c93f07af5cf7dc9b79642a76f343b3f47f2f995
ffd2b77848cf57f&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dnode_exporter-0.18.1.linux-amd64.tar.gz&response-content-type=ap
plication%2Foctet-stream [following]
--2020-04-06 04:15:54--  https://github-production-release-asset-2e65be.s3.amazonaws.com/9524057/5dc5df80-86f1-11e9-924c-ef392e7300e3?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz
-Credential=AKIAIWNJYAX4CSVEH53A%2F20200406%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200406T091457Z&X-Amz-Expires=300&X-Amz-Signature=fea03235e4c93f07af5cf7dc9b79642a76
f343b3f47f2f995ffd2b77848cf57f&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dnode_exporter-0.18.1.linux-amd64.tar.gz&response-
content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 52.216.171.51
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|52.216.171.51|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8083296 (7.7M) [application/octet-stream]
Saving to: 'node_exporter-0.18.1.linux-amd64.tar.gz'

100%[=============================================================================================================================>] 8,083,296   3.05MB/s   in 2.5s

2020-04-06 04:15:58 (3.05 MB/s) - 'node_exporter-0.18.1.linux-amd64.tar.gz' saved [8083296/8083296]

[root@node2-sql exporters]#
```

Figure 4-63 Download the node exporter

3. Run the following command to extract the downloaded `tar.gz` file (see Figure 4-64):

   ```
   tar -xzvf node_exporter-0.18.1.linux-amd64.tar.gz
   ```

```
[root@node2-sql exporters]# tar -xvzf node_exporter-0.18.1.linux-amd64.tar.gz
node_exporter-0.18.1.linux-amd64/
node_exporter-0.18.1.linux-amd64/node_exporter
node_exporter-0.18.1.linux-amd64/NOTICE
node_exporter-0.18.1.linux-amd64/LICENSE
[root@node2-sql exporters]#
```

Figure 4-64 Extract the node exporter tar file

4. Copy the node_exporter file to bin by running the following commands:

   ```
   cd node_exporter-0.18.1.linux-amd64
   cp node_exporter /usr/local/bin
   ```

5. Create a user (`node_exporter`) to run the `node_exporter` as service, and give full access to the `node_exporter` file to the user that was created:

   ```
   useradd --no-create-home --shell /bin/false node_exporter
   chown node_exporter:node_exporter /usr/local/bin/node_exporter
   ```

6. Create an entry in the system to run node exporter as service:

   ```
   echo '[Unit]
   Description=Node Exporter
   Wants=network-online.target
   After=network-online.target

   [Service]
   User=node_exporter
   Group=node_exporter
   Type=simple
   ExecStart=/usr/local/bin/node_exporter

   [Install]
   WantedBy=multi-user.target' > /etc/systemd/system/node_exporter.service
   ```

7. To enable node_exporter as service, run the following commands:

   ```
   systemctl daemon-reload
   systemctl start node_exporter
   systemctl enable node_exporter
   ```

8. Check whether the service is running by running the following command (see Figure 4-65 on page 287):

   ```
   ps -uax | grep node_exporter
   ```

```
[root@node2-sql node_exporter-0.18.1.linux-amd64]# cp node_exporter /usr/local/bin
[root@node2-sql node_exporter-0.18.1.linux-amd64]# useradd --no-create-home --shell /bin/false node_exporter
[root@node2-sql node_exporter-0.18.1.linux-amd64]# chown node_exporter:node_exporter /usr/local/bin/node_exporter
[root@node2-sql node_exporter-0.18.1.linux-amd64]# echo '[Unit]
> Description=Node Exporter
> Wants=network-online.target
> After=network-online.target
>
> [Service]
> User=node_exporter
> Group=node_exporter
> Type=simple
> ExecStart=/usr/local/bin/node_exporter
>
> [Install]
> WantedBy=multi-user.target' > /etc/systemd/system/node_exporter.service
[root@node2-sql node_exporter-0.18.1.linux-amd64]# systemctl daemon-reload
[root@node2-sql node_exporter-0.18.1.linux-amd64]# systemctl start node_exporter
[root@node2-sql node_exporter-0.18.1.linux-amd64]# systemctl enable node_exporter
Created symlink from /etc/systemd/system/multi-user.target.wants/node_exporter.service to /etc/systemd/system/node_exporter.service.
[root@node2-sql node_exporter-0.18.1.linux-amd64]# ps | grep -i node_exporter
[root@node2-sql node_exporter-0.18.1.linux-amd64]# ps uax | grep node
node_ex+   2672  0.0  0.1 115084  5184 ?        Ss1  04:32   0:00 /usr/local/bin/node_exporter
root       2699  0.0  0.0 112684   700 pts/2    S+   04:32   0:00 grep --color=auto node
[root@node2-sql node_exporter-0.18.1.linux-amd64]#
```

Figure 4-65 Check running service

9. After validating that the service is running, the Prometheus settings can be updated and Prometheus can be restarted.

   The Prometheus settings are updated as follows:

```
job_name: 'node_exporter'
scrape_interval: 5s
static_configs:
targets:
<IP:9100>
relabel_configs:
source_labels: [__address__]
regex: <IP:9100>
target_label: instance
replacement: <new_name>
action: replace
```

JMX exporter

JMX exporter is one of the exporters that is custom built for Java applications. As in this use case, the application is Java-based and deployed in Apache Tomcat as a web application to monitor the application metrics that JMX used.

Setting up JMX

To set up JMX, complete the following steps:

1. Log in to the server where the application is going to be deployed as root.
2. Check whether tomcat is running:

```
ps -eaf | grep tomcat
```

3. If tomcat is running, stop it by running the following command:

```
PATH_TOMCAT/bin/shutdown.sh
```

4. Run the following commands to download the JAR file and configuration file for jmx:

```
mkdir /opt/monitoring
cd /opt/monitoring

wget
https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.12.
0/jmx_prometheus_javaagent-0.12.0.jar
```

```
wget
https://github.com/prometheus/jmx_exporter/blob/master/example_configs/tomcat.y
ml
```

5. Run the following command to run JMX as javaagent when tomcat runs:

```
export JAVA_OPTS="$JAVA_OPTS
-javaagent:/opt/monitoring/jmx_prometheus_javaagent-0.3.0.jar=9095:/opt/monitor
ing/tomcat.yml"
```

6. Start tomcat again by running the following command:

```
PATH_TOMCAT/bin/startup.sh
```

The command runs tomcat and the JMX metrics are available at:

```
http://<IP>:9095/metrics
```

The configuration in Prometheus is similar to that of node exporter, for example:

```
-job_name: 'java_metrics'
scrape_interval: 5s
static_configs:
targets:
- <IP>:9095
  relabel_configs:
  - source_labels: [__address__]
    regex: <IP>:9095
    target_label: instance
    replacement: <new_name>
    action: replace
```

IPMI exporter

IPMI exporter is one of the exporters that uses freeIPMI software to fetch the IPMI metrics. The IPMI exporter is installed on a server that can connect to other bare metal servers over the private IP network.

Setting up IPMI

The freeIPMI application must be installed and configured to fetch bmc, chassis, and IPMI data. Complete the following steps:

1. Log in to the server where the application is going to be deployed as root.
2. Check whether ipmimonitoring is accessible, as shown in Figure 4-66.

```
[root@ipmi-exp ~]# which ipmimonitoring
/usr/local/sbin/ipmimonitoring
[root@ipmi-exp ~]#
```

Figure 4-66 Check ipmimonitoring availability

3. If it is not accessible, run the following command after checking that freeIPMI was installed:

```
ipmi-exporter ALL = NOPASSWD:/usr/sbin/ipmimonitoring, /usr/sbin/ipmi-sensors,
/usr/sbin/ipmi-dcmi, /usr/sbin/bmc-info, /usr/sbin/ipmi-chassis
```

4. RUn the following commands to download the `ipmi_exporter` tar file:

```
mkdir /opt/monitoring (if monitoring is not existing)
cd /opt/monitoring
```

```
wget
https://github.com/soundcloud/ipmi_exporter/releases/download/v1.1.0/ipmi_expor
ter-v1.1.0.linux-amd64.tar.gz
tar -xvf ipmi_exporter-v1.1.0.linux-amd64.tar.gz
cd ipmi_exporter-v1.1.0.linux-amd64
cp ipmi.conf ipmi.conf.bak
```

5. Update the `ipmi.conf` file with the content that is shown in Example 4-15.

Example 4-15 Update the ipmi.conf file

```
# Configuration file for ipmi_exporter
# This is an example config for scraping remote hosts via IPMI.
# Information required to access remote IPMI interfaces can be supplied in the
# 'modules' section. A scrape can request the usage of a given config by
# setting the `module` URL parameter.

modules:
  default:
# These settings are used if no module is specified, the
# specified module doesn't exist, or of course if
# module default is specified.
    user: "user_name"
    pass: "_password_"
# The below settings correspond to driver-type, privilege-level, and
# session-timeout respectively, see `man 5 freeipmi.conf` (and e.g.
# `man 8 ipmi-sensors` for a list of driver types).
    driver: "LAN_2_0"
    privilege: "user"
# The session timeout is in milliseconds. Note that a scrape can take up
# to (session-timeout * #-of-collectors) milliseconds, so set the scrape
# timeout in Prometheus accordingly.
    timeout: 10000
# Available collectors are bmc, ipmi, chassis, and dcmi
# If not specified, all three are used
    collectors:
    - bmc
    - ipmi
    - chassis
# Got any sensors you don't care about? Add them here.
#    exclude_sensor_ids:
  dcmi:
# Use these settings when scraped with module=dcmi.
    user: "user_name"
    pass: "_password_"
    privilege: "admin"
    driver: "LAN_2_0"
    collectors:
    - dcmi
  thatspecialhost:
# Use these settings when scraped with module=thatspecialhost.
    user: "user_name"
    pass: "_password_"
    privilege: "admin"
    driver: "LAN"
    collectors:
    - ipmi
```

```
# Need any special workaround flags set? Add them here.
# Workaround flags might be needed to address issues with specific vendor
implementations
# e.g.
https://www.gnu.org/software/freeipmi/freeipmi-faq.html#Why-is-the-output-from-Fre
eIPMI-different-than-another-software_003f
# For a full list of flags, refer to:
# https://www.gnu.org/software/freeipmi/manpages/man8/ipmi-sensors.8.html#lbAL
    workaround_flags:
    - discretereading
```

6. Run the `ipmi_exporter` by running the following command:

   ```
   ./ipmi_exporter --config.file=./ipmi.conf
   ```

   Exporter is up and running at `http://<IP>:9290/metrics`.

7. A `targets.yml` file must be added to the Prometheus server with following content:

   ```
   ---
   - targets:
     - IPMI_server_1_Private_IP
     - IPMI_server_2_Private_IP
   .
   .
     - IPMI_server_n_Private_IP
     labels:
        job: ipmi_exporter
   ```

The targets file must contain all of the IPMI servers' private IPs for which the metrics must be scrapped. It is often kept with the Prometheus `.yml` file.

The configuration in Prometheus must be updated as shown in Example 4-16.

Example 4-16 Prometheus configuration file updates

```
- job_name: ipmi_exporter
  honor_timestamps: true
  scrape_interval: 1m
  scrape_timeout: 30s
  metrics_path: /ipmi
  scheme: http
  file_sd_configs:
  - files:
    - <path_to_targets.yml>
    refresh_interval: 5m
  relabel_configs:
  - source_labels: [__address__]
    separator: ;
    regex: (.*)
    target_label: __param_target
    replacement: ${1}
    action: replace
  - source_labels: [__param_target]
    separator: ;
    regex: (.*)
    target_label: instance
    replacement: ${1}
    action: replace
```

```
  - separator: ;
    regex: .*
    target_label: __address__
    replacement: <IP/DNS_Name>:9290
    action: replace
```

4.3.4 Alertmanager

Alertmanager triggers webhook when the alert is received from Prometheus. The Alertmanager configuration is in `alertmanager.yml` (`$HOME/Alertmanager/alertmanager-0.20.0.linux-amd64/alertmanager.yml`), as shown in Example 4-17.

Example 4-17 alertmanager.yml

```
global:
  resolve_timeout: 1m

route:
  group_by: ['alertname','instance']
  group_wait: 10s
  group_interval: 15s
  repeat_interval: 1h
  #receiver: 'web.hook'
  #receiver: 'default-receiver'
  receiver: email-me
inhibit_rules:
- source_match:
    severity: '1'
  target_match:
    severity: '2'
  # Apply inhibition if the alertname is the same.
  equal: ['alertname','summary','instance']
receivers:
- name: email-me
  email_configs:
  - to: 219.ajit@gmail.com
    from: root@vsi-servicemanagement.doy.com
    smarthost: smtp.gmail.com:587
    auth_username: cloudibm2020@gmail.com
    auth_identity: cloudibm2020@gmail.com
    auth_password: uylugagvfofasqgt
    send_resolved: true
    require_tls: true
  webhook_configs:
  - url: 'http://169.38.67.66:9000/hooks/runscript'
    send_resolved: true
```

4.3.5 Webhook

Webhook is started by Alertmanager to run actions when an alert is received from Prometheus. The `sendmessage.sh` that is included in the webhook configuration sends a message to iTop, as shown in Example 4-18 on page 292.

Example 4-18 /home/j123/hooks.json

```
[
{
"id": "runscript",
"execute-command": "/home/j123/sendmessage.sh",
"command-working-directory": "/home/j123/",
"pass-arguments-to-command":
[
{
  "source": "payload",
  "name": "alerts.0.labels.instance"
},
{
  "source": "payload",
  "name": "alerts.0.labels.alertname"
},
{
  "source": "string",
  "name": "is above threshold"
},
{
  "source": "string",
  "name": "HARD"
},
{
  "source": "payload",
  "name": "alerts.0.labels.severity"
},
{
  "source": "payload",
  "name": "alerts.0.labels.priority"
},
{
  "source": "payload",
  "name": "alerts.0.labels.summary"
},
{
  "source": "payload",
  "name": "alerts.0.labels.description"
}
]
}
]
```

Run webhook by using the following command:

```
$ /root/Alertmanager/webhook -hooks /home/j123/hooks.json -verbose
```

Webhook logs show that webhook is triggered and creates a ticket in iTop by using REST API. To send a message to iTop for service management, the running script (`sendmessage.sh`) is run, as shown in Example 4-19.

Example 4-19 /home/j123/sendmessage.sh

```
#!/bin/bash
################################################################################
```

```
#                                                                              #
# Example script for creating a UserRequest ticket via the REST/JSON webservices #
#                                                                              #
################################################################################

# iTop location and credentials, change them to suit your iTop installation
ITOP_URL="http://169.38.131.81/itop/"
ITOP_USER="admin"
ITOP_PWD="p@sswOrd"


# Parameters checking, see below for default values
if [ "$1" == "" ]; then
                echo "Missing parameter 1: host"
                exit -1
else
                HOST="$1"
fi

if [ "$2" == "" ]; then
                echo "Missing parameter 2: Service"
                exit -1
else
                SERVICE="$2"
fi

if [ "$3" == "" ]; then
                echo "Missing parameter 3: Service Status"
                exit -1
else
                SERVICE_STATUS="$3"
fi

if [ "$4" == "" ]; then
                echo "Missing parameter 4: Service State Type"
                exit -1
else
                SERVICE_STATUS_TYPE="$4"
fi

if [ "$5" == "" ]; then
                echo "Missing parameter 5: Urgency"
                exit -1
else
                URGENCY="$5"
fi

if [ "$6" == "" ]; then
                echo "Missing parameter 6: Priority"
                exit -1
else
                PRIORITY="$6"
fi

if [ "$7" == "" ]; then
```

```
                echo "Missing parameter 7: Summary"
                exit -1
else
                SUMMARY="$7"
fi

if [ "$8" == "" ]; then
                echo "Missing parameter 8: Description"
                exit -1
else
                DESCRIPTION="$8"
fi
# Default values, adapt them to your configuration
TICKET_CLASS="Incident"
ORGANIZATION="SELECT Organization JOIN FunctionalCI AS CI ON
CI.org_id=Organization.id WHERE CI.name='"${HOST}"'"
#ORGANIZATION="Demo"
TITLE="$SUMMARY - is raised for metric - $2"
DESCRIPTION="$DESCRIPTION - is raised on $HOST"
#CALLER="prometheus"

# Let's create the ticket via the REST/JSON API
if [[ ( "$SERVICE_STATUS" != "OK" ) && ( "$SERVICE_STATUS" != "UP" ) && (
"$SERVICE_STATUS_TYPE" == "HARD" ) ]]; then
                CIS_LIST='[{"functionalci_id":"SELECT FunctionalCI WHERE
name=\"'"$1"'\"", "impact_code": "manual"}]'
                JSON_DATA='{"operation":"core/create",
"class":"'"${TICKET_CLASS}"'", "fields": {"functionalcis_list":'"${CIS_LIST}"',
"org_id":"'"${ORGANIZATION}"'", "title":"'"$SUMMARY"'",
"urgency":"'"$URGENCY"'","priority":"'"$PRIORITY"'","description":"'"$DESCRIPTION"
'"}, "comment": "Created by the Monitoring", "output_fields": "id"}'

                RESULT=`wget -q
--post-data='auth_user='"${ITOP_USER}"'&auth_pwd='"${ITOP_PWD}"'&json_data='"${JSO
N_DATA}" --no-check-certificate -O -
"${ITOP_URL}/webservices/rest.php?version=1.0"`

                PATTERN='"key":"([0-9])+"'
                if [[ $RESULT =~ $PATTERN ]]; then
                                echo "Ticket created successfully"
                                echo $PATTERN
                else
                                echo "ERROR: failed to create ticket"
                                echo $RESULT
                fi
else
                echo "Service State Type != HARD, doing nothing"
fi
```

Figure 4-67 shows the log of running `sendmessage.sh`.

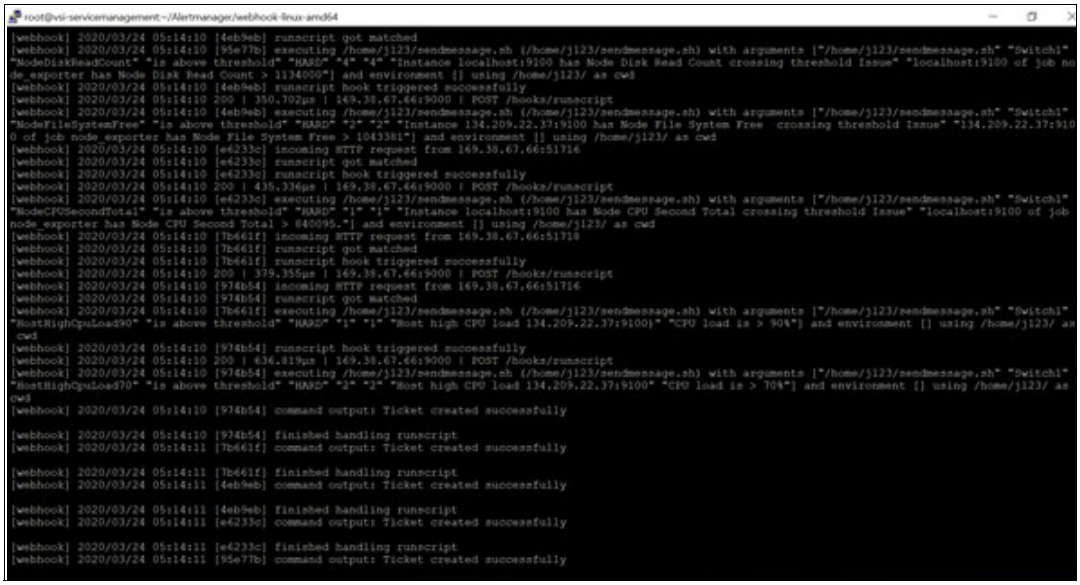


Figure 4-67 sendmessage.sh

To customize the webhook container, a Dockerfile is created, as shown in Example 4-20.

Example 4-20 /home/j123/webhook/Dockerfile

```
FROM almir/webhook:latest
LABEL maintainer="ajityada@in.ibm.com"
USER root
RUN mkdir /home/j123/
WORKDIR /home/j123/contianer
COPY hooks.json /etc/webhook/
COPY sendmessage.sh /home/j123/
CMD ["-verbose", "-hooks=/etc/webhook/hooks.json", "-hotreload"]
```

In the Dockerfile directory, build the container image and push the container image to `quay.io` as follows:

```
$cd /home/j123/webhook/Dockerfile
$docker build  -t  quay.io/manasmohsin/webhook_j3:v6  .
$docker push quay.io/manasmohsin/webhook_j3:v6
```

After you push the image, you can find it by browsing to the following link (in our scenario):

`https://quay.io/repository/manasmohsin/webhook_j3?tab=tags`

The `quay.io` is providing a vulnerability test for the image. Figure 4-68 shows that this custom image passed the vulnerability test.

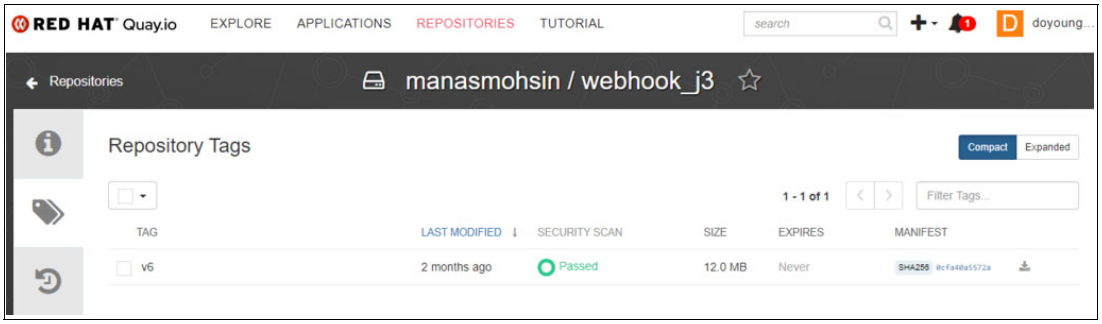


Figure 4-68 Red Hat Quay.io: Repository Tags window

4.3.6 Grafana

By using Grafana, users can query, visualize, understand, and send alerts about metric data that was collected from servers or applications. Various storage options are available that can be used to store the metrics and Grafana is adaptive in terms of data sources. It creates dashboards of the metrics that can be used to understand the on-going historical behavior. It allows users to create filters and manage with the data based on the filters.

Complete the following steps to deploy Grafana on Linux:

1. Define inbound rules.

   Check that you have port 3000 (Grafana runs to allow communication at this port) open for access from the defined sources. For the sake of easing the process, all of the ICMP, TCP, and UDP ports are open for the source, as shown in Figure 4-69. Specific rules can be added.

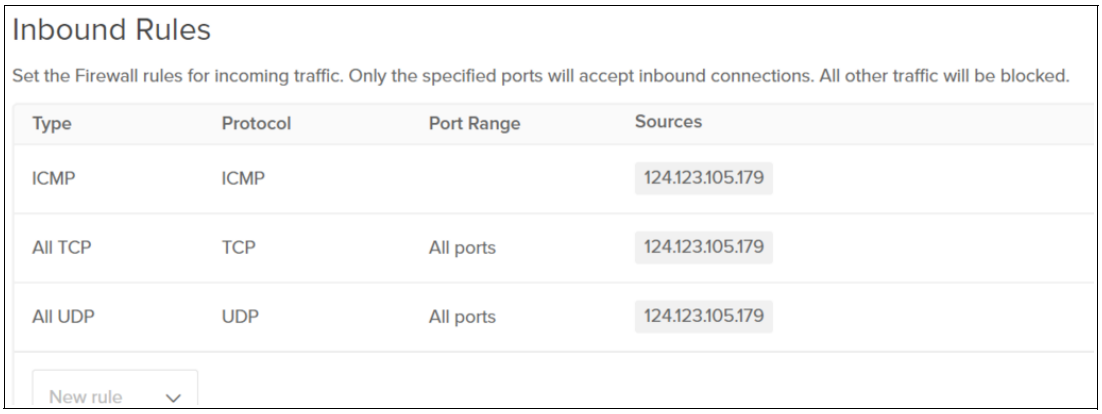


Figure 4-69 Define inbound rules

2. Define outbound rules.

   Check whether this system can communicate with other devices. For ease, all of the ports are open to access from all of the IPs, as shown in Figure 4-70.

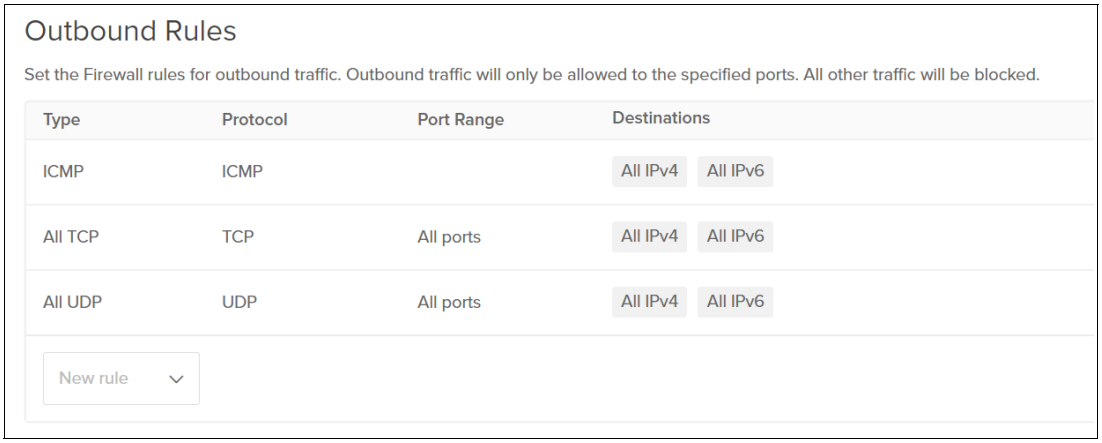


Figure 4-70 Define outbound rules

3. Update and upgrade the server:

```
sudo apt-get update
sudo apt-get upgrade
```

4. Reboot the system:

```
sudo reboot
```

5. Install wget if not available already:

```
sudo apt-get -y install wget
```

6. Install the latest Grafana:

```
echo 'deb https://packages.grafana.com/oss/deb stable main' >>
/etc/apt/sources.list
curl https://packages.grafana.com/gpg.key | sudo apt-key add -
sudo apt-get update
sudo apt-get -y install grafana
```

7. Start and enable the Grafana server:

```
systemctl daemon-reload
systemctl start grafana-server
```

8. Enable to start the Grafana server on boot:

```
systemctl enable grafana-server.service
```

9. Check the server status:

```
systemctl status grafana-server
```

The Grafana server is now active (that is, running).

Note: The Grafana configuration is stored in `/etc/grafana/grafana.ini`. This configuration must *not* be changed. If changes are required, copy the original file and change that copied file.

10. Access Grafana.

    Go to `http://<IP>:3000`.

11. Enter the user name (`admin`) and password (`admin`), as shown in Figure 4-71.

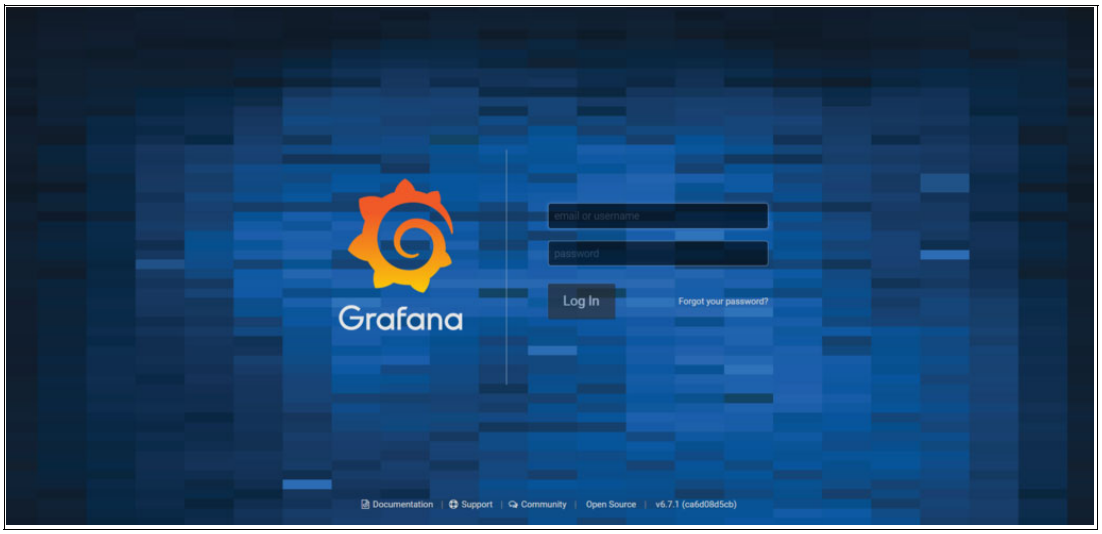


Figure 4-71 Grafana log in window

After these credentials are entered, the password can be changed or can be skipped and changed later.

Prometheus data source in Grafana

Grafana supports multiple data sources and for our example, Prometheus is used as the data source. Because metrics can be read and stored in Prometheus, it makes life easier when we Prometheus support is in Grafana. This support allows for the creation of dashboards from metrics data and understanding the server state by using dashboards at run time easy and efficient.

Setting up Prometheus as the data source

Complete the following steps:

1. Log in to Grafana at `http://<IP>:3000`. Enter your credentials, as shown in Figure 4-71 on page 298.
2. Add a data source.

   After logging in to Grafana, the Welcome window looks as shown in Figure 4-72. This window is where the Create a data source option is green and enabled. Also, no dashboards were starred or recently accessed.

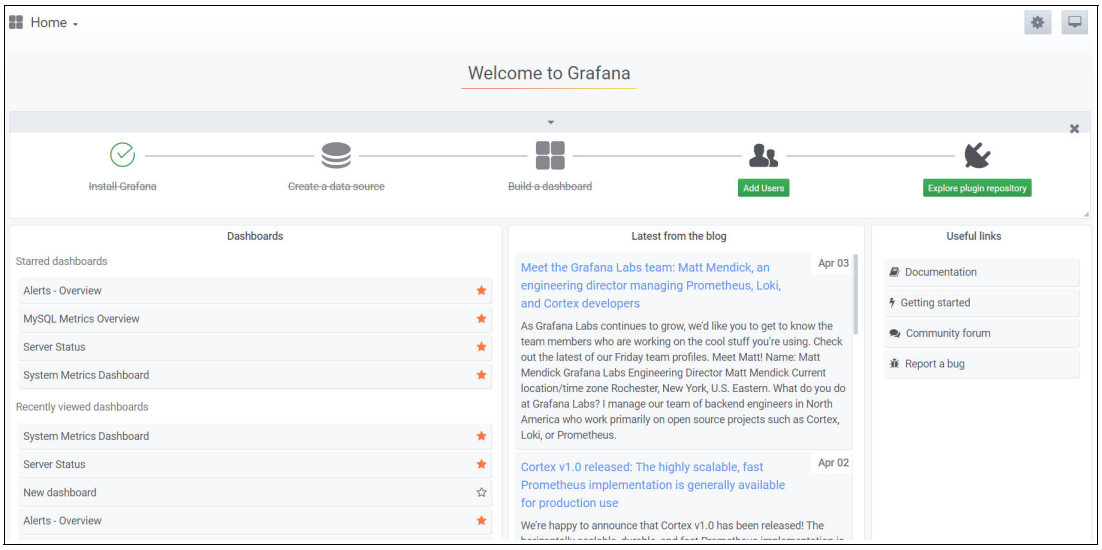


Figure 4-72 Grafana welcome pane

3. Click **Add Data Source** or go to Configuration and select **Data Source** to add data source. For the current use case, select **Prometheus**, which is shown under the time series databases as the data source, as shown in Figure 4-73.

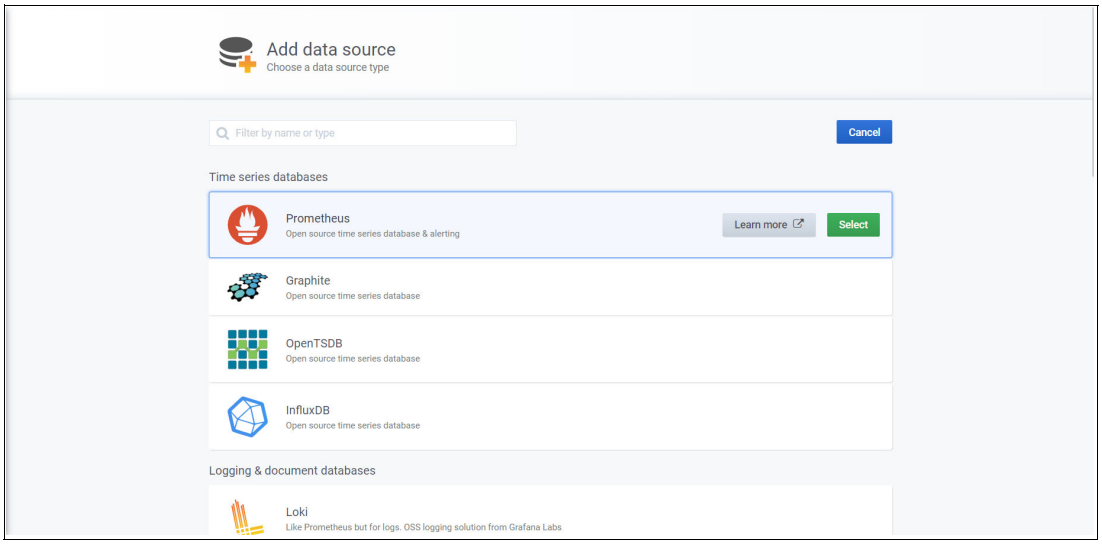


Figure 4-73 Add data source pane

The Data Sources addition page is shown in Figure 4-74.

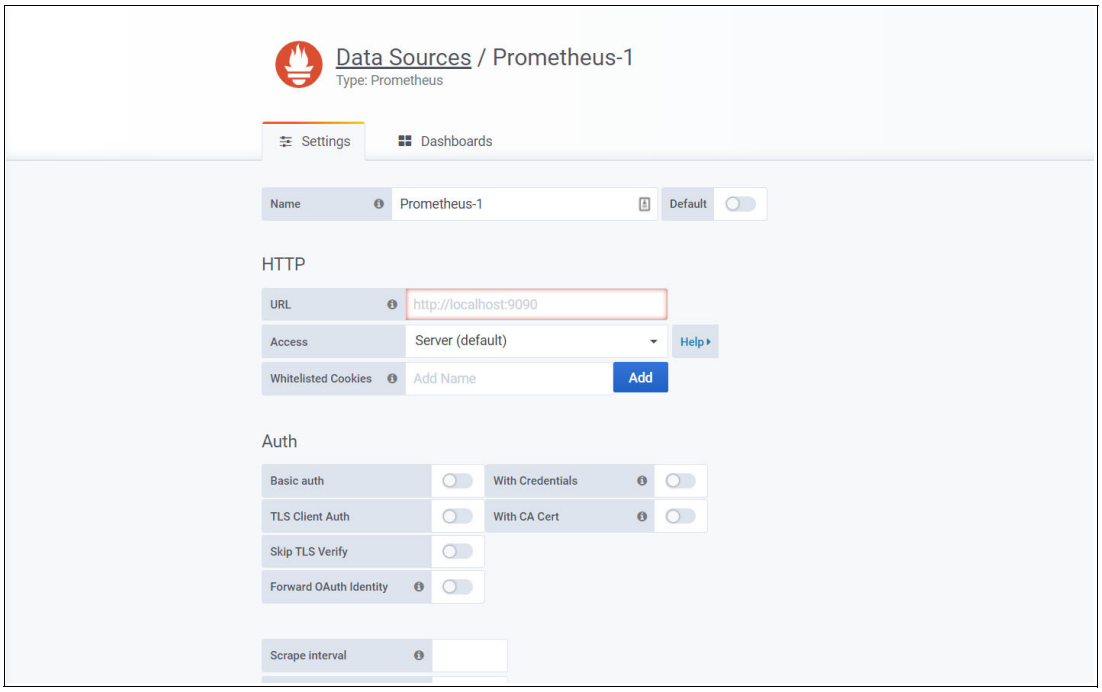


Figure 4-74 Data Sources pane: Type Prometheus

4. Enter a Unique Name, URL (Prometheus URL), and set the HTTP Method as GET.

   These settings are sufficient to integrate Prometheus. However, if some level of authentication is needed for Prometheus, you might have to provide it in the settings, as shown in Figure 4-74.

1. Click **Save and Test**. The message, “Data source is working”, is shown (see Figure 4-75).

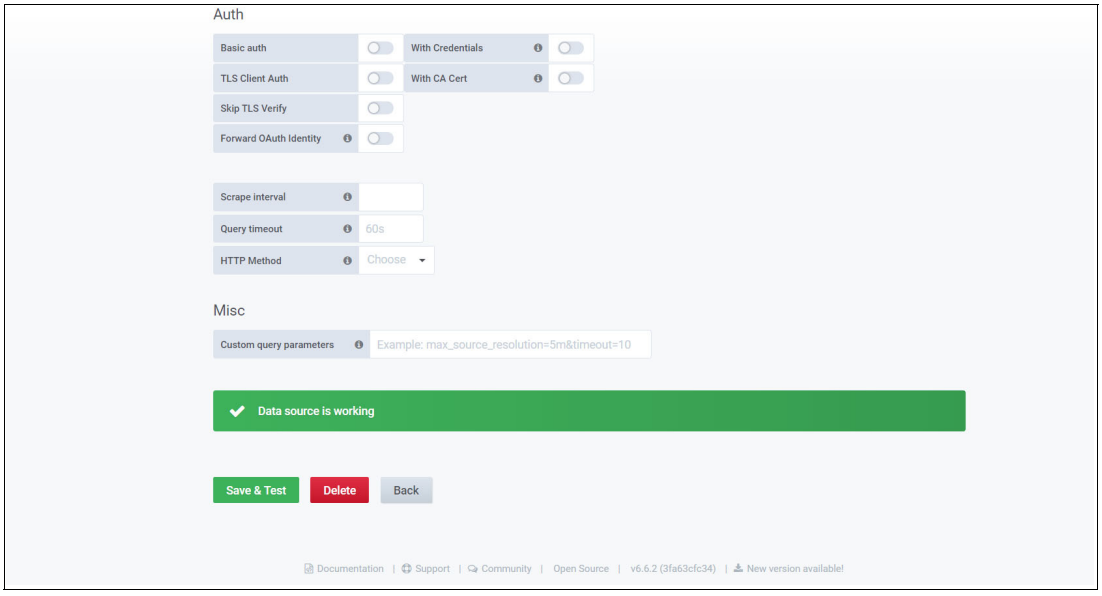


Figure 4-75 Successful data source integration message

After successful addition, the data source appears in the Data Sources list, as shown in Figure 4-76.

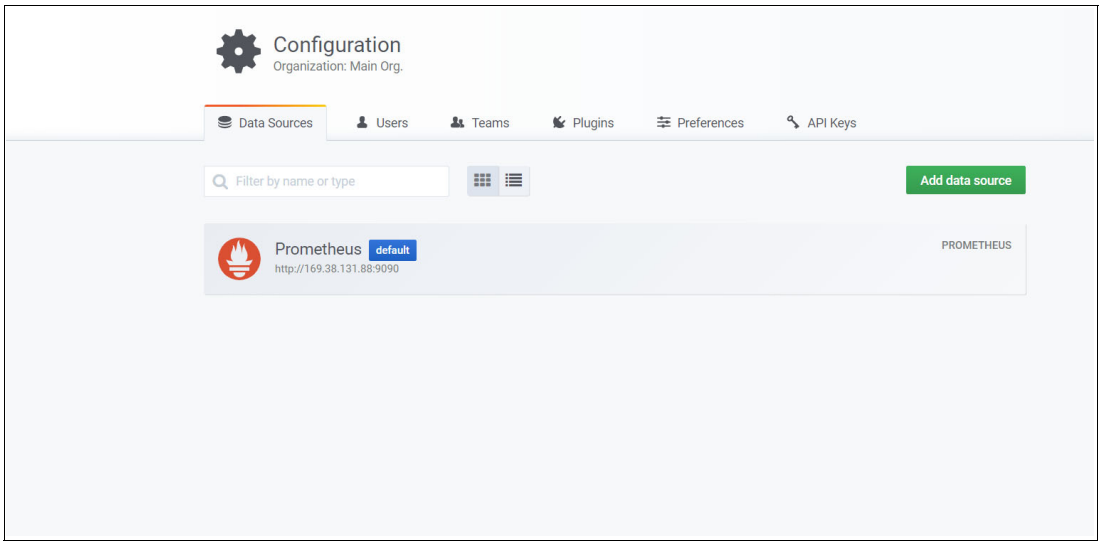


Figure 4-76 Data Source: Configuration Organization pane

The data now can be fetched from the Prometheus server and plotted in dashboard windows.

4.3.7 Adding SLA dashboard

This section describes how to add a SLA in the Grafana dashboard.

Setting up SLA dashboards

Dashboards in Grafana can be set up in various ways. This flexibility enables quick setup and variety. Options also are available that allow users to tweak an existing concept to their need instead of redeveloping the concept from scratch.

Consider the following points:

- It can be developed from scratch; for example, by going through metrics individually and adding windows per needed in a dashboard.
- User can use dashboards that are developed by others and uploaded to Grafana labs.
- User can import a JSON form of a dashboard.

Complete the following steps:

1. After logging in to Grafana UI, select the plus sign (**+**) and select **Dashboard**.
2. Import from a `.json` file by selecting the **Manage** tab, as shown in Figure 4-77.

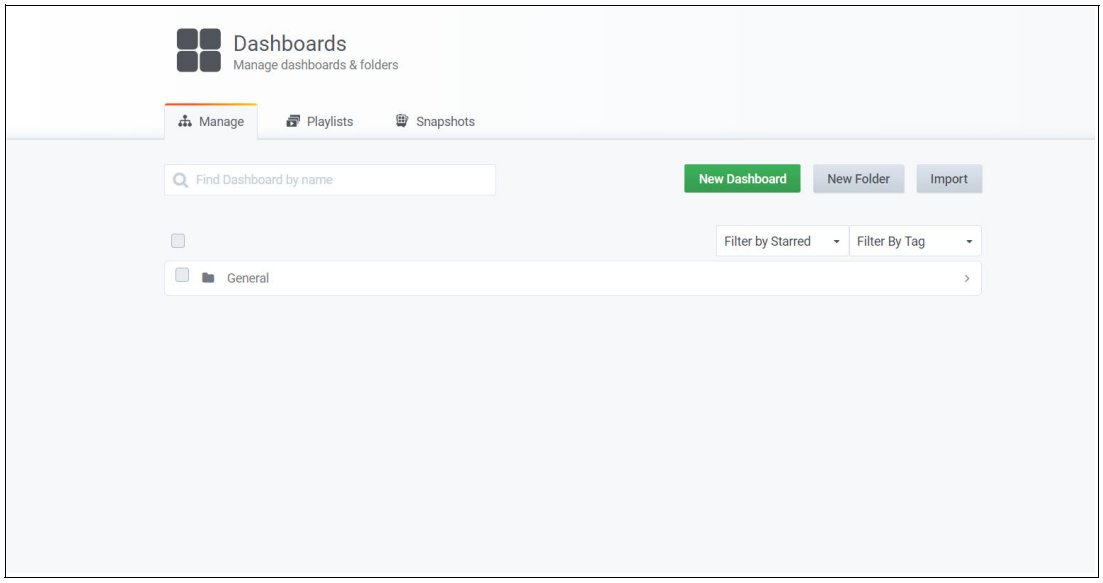


Figure 4-77 Manage dashboards and folders

3. Click **Import** (see Figure 4-77). The window in which you import a dashboard from JSON opens, as shown in Figure 4-78.

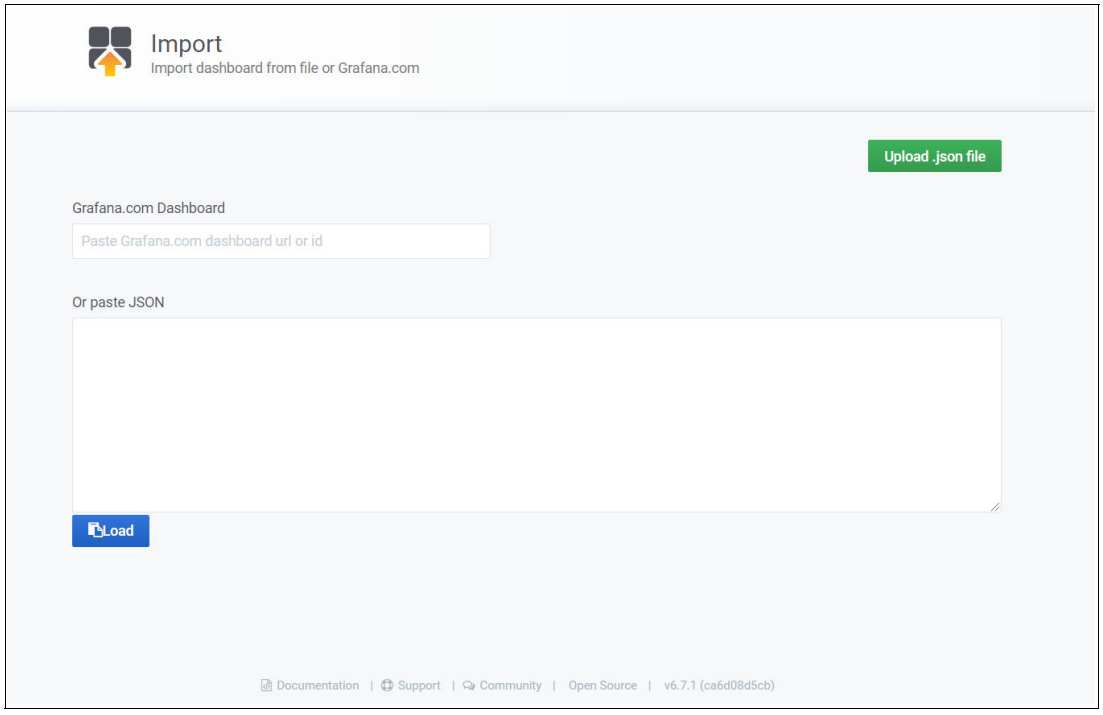


Figure 4-78 Import pane: Upload dashboard

4. Click **Upload .json file** (see Figure 4-78 on page 302) to choose the JSON file and upload it from a file selector window. After the file is selected and uploaded, a few properties can be updated, such as Name, Folder, and Prometheus Data Source.

   By default, the dashboard is named when it is created. Although it is not necessary to update the folder, it is suggested that you select it and create it if it is not available. The data source must be added because it is a mandatory field to import the dashboard.

5. After all the values are set, click **Import** (see Figure 4-79).

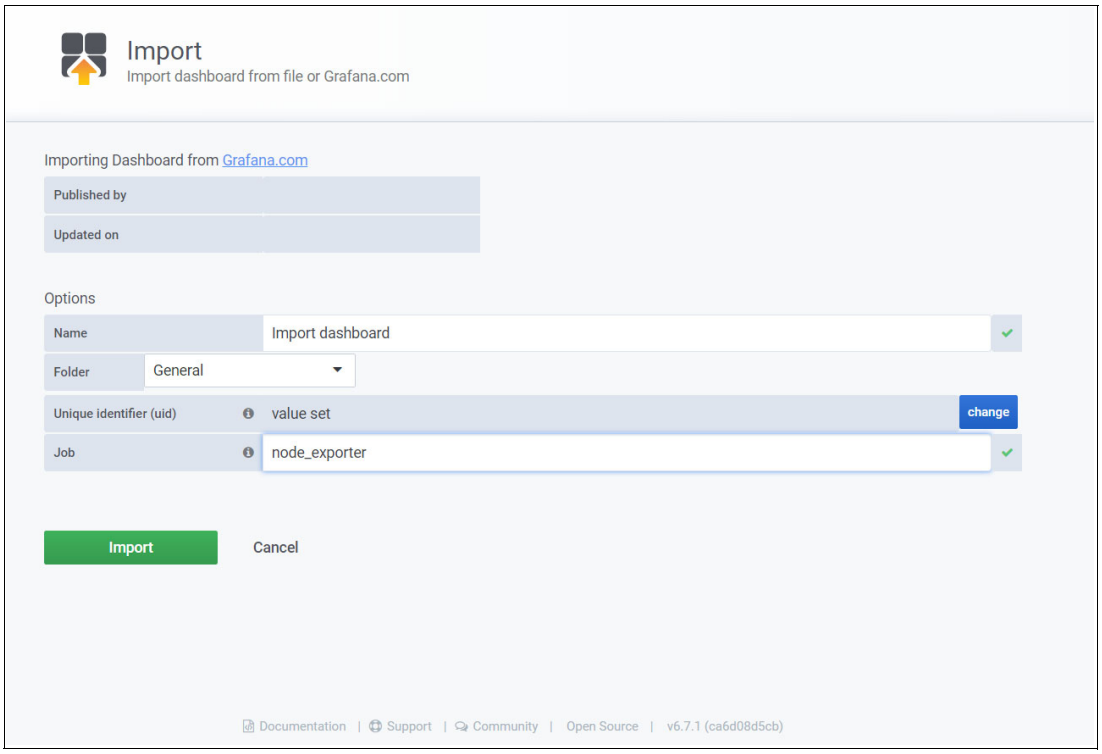


Figure 4-79 Import pane: Importing Dashboard

After the import process is complete, the dashboard is updated with the graphs and plots, as shown in Figure 4-80.

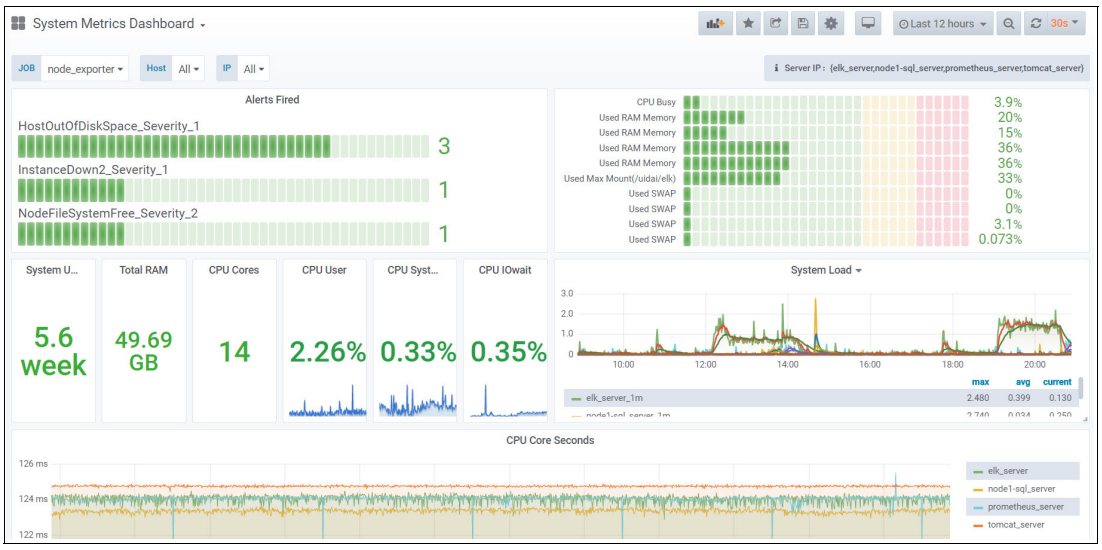


Figure 4-80 Systems Metrics Dashboard

The dashboard includes options to update the data every 5, 10, or 30 seconds. By default, 30 seconds is selected.

The option is available to select the metrics of the last *n* hours or minutes to show in the graph. By default, 1 hour of data is shown and it is updated every 30 seconds.

If multiple jobs exist, they are listed in JOB menu. Similarly, drop-down menus for hosts and IPs are also provided. Selecting any combination or individual displays filtered data that is based on the selection, and updates the plots with the available data.

4.4 Service management by using IT Operational Portal

This section describes how to perform system management with iTop.

4.4.1 iTop overview

iTop is an open source web-based IT Service Management tool. It is ITIL oriented and allows users to manage user requests, incidents, problems, changes, and service catalog in a single platform and configuration items and their relationships in a flexible CMDB.

Features

iTop includes the following features:

- Fully configurable Configuration Management (CMDB)
- HelpDesk and incident management
- Service and contract management
- Change management
- Configurable SLA management
- Graphical impact analysis
- CSV import tool for any data
- Consistency audit to check data quality
- Data synchronization (for data federation)

Installation of iTop

iTop relies on Apache/IIS, MySQL, and PHP as prerequisites for installation.

To install iTop V2.6.3 on Red Hat Enterprise Linux server V7.7, install Apache server.

Apache is an open source multi-platform web server. It provides a full range of web server features including CGI, SSL, and virtual domains.

To install Apache, enter the following command (see Figure 4-81 on page 305):

```
# yum install httpd -y
```

```
[root@itop-new ~]#
[root@itop-new ~]# yum install httpd -y
Loaded plugins: product-id, search-disabled-repos, subscription-manager
rhel-7-server-optional-rpms                                | 1.8 kB     00:00
rhel-7-server-rpms                                         | 2.0 kB     00:00
rhel-7-server-supplementary-rpms                           | 2.0 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.6-93.el7 will be installed
--> Processing Dependency: httpd-tools = 2.4.6-93.el7 for package: httpd-2.4.6-9
3.el7.x86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.6-93.el7.x86_6
4
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.4.6-9
3.el7.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.6-93.el
7.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.4.8-5.el7 will be installed
---> Package apr-util.x86_64 0:1.5.2-6.el7 will be installed
---> Package httpd-tools.x86_64 0:2.4.6-93.el7 will be installed
---> Package mailcap.noarch 0:2.1.41-2.el7 will be installed
--> Finished Dependency Resolution
```

Figure 4-81 Install Apache server

Figure 4-82 shows dependencies that are resolved during the installation.

```
Dependencies Resolved

================================================================================
 Package          Arch        Version              Repository               Size
================================================================================
Installing:
 httpd            x86_64      2.4.6-93.el7         rhel-7-server-rpms      1.2 M
Installing for dependencies:
 apr              x86_64      1.4.8-5.el7          rhel-7-server-rpms      104 k
 apr-util         x86_64      1.5.2-6.el7          rhel-7-server-rpms       92 k
 httpd-tools      x86_64      2.4.6-93.el7         rhel-7-server-rpms       92 k
 mailcap          noarch      2.1.41-2.el7         rhel-7-server-rpms       31 k

Transaction Summary
================================================================================
Install  1 Package (+4 Dependent packages)

Total download size: 1.5 M
Installed size: 4.3 M
Downloading packages:
(1/5): apr-1.4.8-5.el7.x86_64.rpm                          | 104 kB   00:00
(2/5): apr-util-1.5.2-6.el7.x86_64.rpm                     |  92 kB   00:00
(3/5): httpd-2.4.6-93.el7.x86_64.rpm                       | 1.2 MB   00:00
(4/5): httpd-tools-2.4.6-93.el7.x86_64.rpm                 |  92 kB   00:00
```

Figure 4-82 Solving dependencies

Figure 4-83 shows the continuation of the installation of Apache server.

```
(5/5): mailcap-2.1.41-2.el7.noarch.rpm                    |  31 kB   00:00
--------------------------------------------------------------------------------
Total                                           3.2 MB/s | 1.5 MB  00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : apr-1.4.8-5.el7.x86_64                                      1/5
  Installing : apr-util-1.5.2-6.el7.x86_64                                 2/5
  Installing : httpd-tools-2.4.6-93.el7.x86_64                             3/5
  Installing : mailcap-2.1.41-2.el7.noarch                                 4/5
  Installing : httpd-2.4.6-93.el7.x86_64                                   5/5
Loaded plugins: product-id, subscription-manager
  Verifying  : apr-1.4.8-5.el7.x86_64
  Verifying  : mailcap-2.1.41-2.el7.noarch
  Verifying  : httpd-2.4.6-93.el7.x86_64
  Verifying  : apr-util-1.5.2-6.el7.x86_64
  Verifying  : httpd-tools-2.4.6-93.el7.x86_64
rhel-7-server-supplementary-rpms/7Server/x86_64/productid

Installed:
  httpd.x86_64 0:2.4.6-93.el7
```

Figure 4-83 Installation continues

Figure 4-84 shows the completed installation message.

```
Dependency Installed:
  apr.x86_64 0:1.4.8-5.el7          apr-util.x86_64 0:1.5.2-6.el7          httpd

Complete!
[root@itop-new ~]#
[root@itop-new ~]# _
```

Figure 4-84 Install complete

Installing httpd-devel package

Install the `httpd-devel` package by running the following command (see Figure 4-85):

```
# yum install httpd-devel
```

```
[root@itop-new itop]# yum install httpd-devel
Loaded plugins: product-id, search-disabled-repos, subscription-manager
rhel-7-server-optional-rpms                                | 1.8 kB     00:00
rhel-7-server-rpms                                         | 2.0 kB     00:00
rhel-7-server-supplementary-rpms                           | 2.0 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package httpd-devel.x86_64 0:2.4.6-93.el7 will be installed
--> Processing Dependency: apr-util-devel for package: httpd-devel-2.4.6-93.el7.
x86_64
--> Processing Dependency: apr-devel for package: httpd-devel-2.4.6-93.el7.x86_6
4
--> Running transaction check
---> Package apr-devel.x86_64 0:1.4.8-5.el7 will be installed
---> Package apr-util-devel.x86_64 0:1.5.2-6.el7 will be installed
--> Processing Dependency: openldap-devel(x86-64) for package: apr-util-devel-1.
5.2-6.el7.x86_64
--> Processing Dependency: libdb-devel(x86-64) for package: apr-util-devel-1.5.2
-6.el7.x86_64
--> Processing Dependency: expat-devel(x86-64) for package: apr-util-devel-1.5.2
-6.el7.x86_64
--> Running transaction check
---> Package expat-devel.x86_64 0:2.1.0-11.el7 will be installed
--> Processing Dependency: expat = 2.1.0-11.el7 for package: expat-devel-2.1.0-1
```

Figure 4-85 Install httpd-devel

Figure 4-86 shows the dependencies check and continuing with the installation.

```
1.el7.x86_64
---> Package libdb-devel.x86_64 0:5.3.21-25.el7 will be installed
---> Package openldap-devel.x86_64 0:2.4.44-21.el7_6 will be installed
--> Processing Dependency: cyrus-sasl-devel(x86-64) for package: openldap-devel-
2.4.44-21.el7_6.x86_64
--> Running transaction check
---> Package cyrus-sasl-devel.x86_64 0:2.1.26-23.el7 will be installed
--> Processing Dependency: cyrus-sasl(x86-64) = 2.1.26-23.el7 for package: cyrus
-sasl-devel-2.1.26-23.el7.x86_64
---> Package expat.x86_64 0:2.1.0-10.el7_3 will be updated
---> Package expat.x86_64 0:2.1.0-11.el7 will be an update
--> Running transaction check
---> Package cyrus-sasl.x86_64 0:2.1.26-23.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package             Arch      Version               Repository             Size
================================================================================
Installing:
 httpd-devel         x86_64    2.4.6-93.el7          rhel-7-server-rpms    198 k
Installing for dependencies:
 apr-devel           x86_64    1.4.8-5.el7           rhel-7-server-rpms    188 k
```

Figure 4-86 Checking dependencies

Figure 4-87 shows the continuation of the dependencies updating process.

```
 apr-util-devel      x86_64    1.5.2-6.el7          rhel-7-server-rpms     76 k
 cyrus-sasl          x86_64    2.1.26-23.el7        rhel-7-server-rpms     88 k
 cyrus-sasl-devel    x86_64    2.1.26-23.el7        rhel-7-server-rpms    310 k
 expat-devel         x86_64    2.1.0-11.el7         rhel-7-server-rpms     57 k
 libdb-devel         x86_64    5.3.21-25.el7        rhel-7-server-rpms     39 k
 openldap-devel      x86_64    2.4.44-21.el7_6      rhel-7-server-rpms    804 k
Updating for dependencies:
 expat               x86_64    2.1.0-11.el7         rhel-7-server-rpms     81 k

Transaction Summary
================================================================================
Install  1 Package  (+7 Dependent packages)
Upgrade             ( 1 Dependent package)

Total download size: 1.8 M
Is this ok [y/d/N]: y
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
(1/9): apr-devel-1.4.8-5.el7.x86_64.rpm                    | 188 kB   00:00
(2/9): apr-util-devel-1.5.2-6.el7.x86_64.rpm               |  76 kB   00:00
(3/9): cyrus-sasl-2.1.26-23.el7.x86_64.rpm                 |  88 kB   00:00
(4/9): cyrus-sasl-devel-2.1.26-23.el7.x86_64.rpm           | 310 kB   00:00
(5/9): expat-2.1.0-11.el7.x86_64.rpm                       |  81 kB   00:00
(6/9): expat-devel-2.1.0-11.el7.x86 64.rpm                 |  57 kB   00:00
```

Figure 4-87 Updating dependencies

Figure 4-88 shows the continuation of the installation, verification, and updating procedure.

```
(7/9): httpd-devel-2.4.6-93.el7.x86_64.rpm                 | 198 kB   00:00
(8/9): libdb-devel-5.3.21-25.el7.x86_64.rpm                |  39 kB   00:00
(9/9): openldap-devel-2.4.44-21.el7_6.x86_64.rpm           | 804 kB   00:00
--------------------------------------------------------------------------------
Total                                           2.8 MB/s | 1.8 MB  00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : apr-devel-1.4.8-5.el7.x86_64                                 1/10
  Installing : libdb-devel-5.3.21-25.el7.x86_64                             2/10
  Installing : cyrus-sasl-2.1.26-23.el7.x86_64                              3/10
  Installing : cyrus-sasl-devel-2.1.26-23.el7.x86_64                        4/10
  Installing : openldap-devel-2.4.44-21.el7_6.x86_64                        5/10
  Updating   : expat-2.1.0-11.el7.x86_64                                    6/10
  Installing : expat-devel-2.1.0-11.el7.x86_64                              7/10
  Installing : apr-util-devel-1.5.2-6.el7.x86_64                            8/10
  Installing : httpd-devel-2.4.6-93.el7.x86_64                              9/10
  Cleanup    : expat-2.1.0-10.el7_3.x86_64                                 10/10
Loaded plugins: product-id, subscription-manager
  Verifying  : expat-2.1.0-11.el7.x86_64                                    1/10
  Verifying  : apr-util-devel-1.5.2-6.el7.x86_64                            2/10
  Verifying  : httpd-devel-2.4.6-93.el7.x86_64                              3/10
  Verifying  : expat-devel-2.1.0-11.el7.x86 64                              4/10
```

Figure 4-88 Installation continues

Figure 4-89 shows the installation completion.

```
  Verifying  : openldap-devel-2.4.44-21.el7_6.x86_64                        5/10
  Verifying  : cyrus-sasl-2.1.26-23.el7.x86_64                              6/10
  Verifying  : apr-devel-1.4.8-5.el7.x86_64                                 7/10
  Verifying  : libdb-devel-5.3.21-25.el7.x86_64                             8/10
  Verifying  : cyrus-sasl-devel-2.1.26-23.el7.x86_64                        9/10
  Verifying  : expat-2.1.0-10.el7_3.x86_64                                 10/10

Installed:
  httpd-devel.x86_64 0:2.4.6-93.el7

Dependency Installed:
  apr-devel.x86_64 0:1.4.8-5.el7
  apr-util-devel.x86_64 0:1.5.2-6.el7
  cyrus-sasl.x86_64 0:2.1.26-23.el7
  cyrus-sasl-devel.x86_64 0:2.1.26-23.el7
  expat-devel.x86_64 0:2.1.0-11.el7
  libdb-devel.x86_64 0:5.3.21-25.el7
  openldap-devel.x86_64 0:2.4.44-21.el7_6

Dependency Updated:
  expat.x86_64 0:2.1.0-11.el7

Complete!
[root@itop-new itop]#
```

Figure 4-89 Installation complete

Starting Apache Server

Start the Apache service and configure it to start automatically on every restart (see Figure 4-90) by running the following command:

```
# service httpd restart
# chkconfig httpd on
```

```
[root@itop-new ~]#
[root@itop-new ~]# service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[root@itop-new ~]#
[root@itop-new ~]# chkconfig httpd on
Note: Forwarding request to 'systemctl enable httpd.service'.
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service t
o /usr/lib/systemd/system/httpd.service.
[root@itop-new ~]#
```

Figure 4-90 Configure the Apache service

Testing Apache installation

Open your web browser and browse to `http://localhost/` or `http://server-ip-address/`, as shown in Figure 4-91.

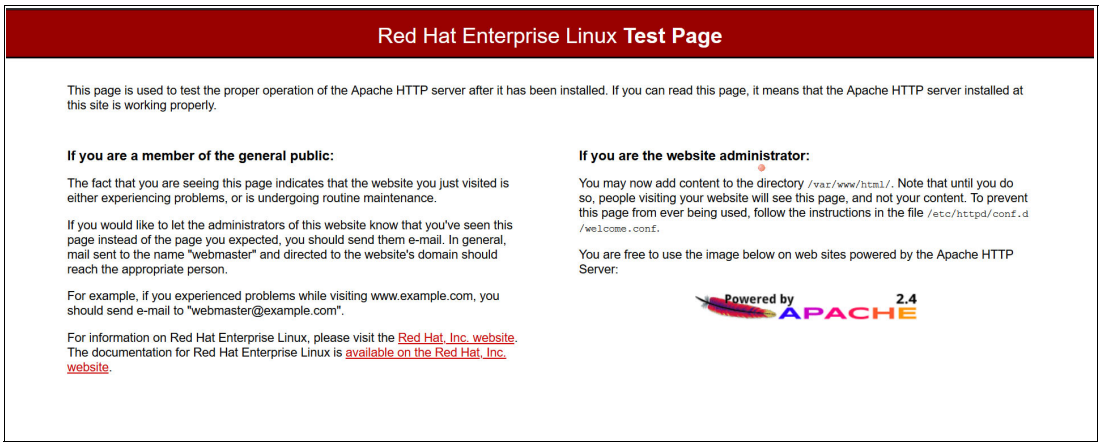


Figure 4-91 Red Hat Enterprise Linux: Test Page

Installing MySQL server

MySQL is an enterprise class open source database. Complete the following steps to install MySQL server:

1. Download the binaries for MySQL Server 5.7.29 (see Figure 4-92):

```
# wget https://dev.mysql.com/get/mysql-5.7.29-1.el7.x86_64.rpm-bundle.tar
```

```
[root@itop-new tmp]# wget https://dev.mysql.com/get/mysql-5.7.29-1.el7.x86_64.rpm-bundle.tar
```

Figure 4-92 Download binaries

2. Untar the RPM package.

   Browse to the location where the `mysql-5.7.29-1.el7.x86_64.rpm-bundle.tar` file was downloaded and run the following command (see Figure 4-93):

```
# tar -xvf mysql-5.7.29-1.el7.x86_64.rpm-bundle.tar
```

```
[root@itop-new tmp]# tar -xvf mysql-5.7.29-1.el7.x86_64.rpm-bundle.tar
```

Figure 4-93 Untar the binaries

After completing the untar, these files are available, as shown in Figure 4-94.

```
[root@itop-new tmp]# ll
total 1067184
-rw-r--r--. 1 root root  545832960 Dec 19 02:20 mysql-5.7.29-1.el7.x86_64.rpm-bundle.tar
-rw-r--r--. 1 7155 31415  27768112 Dec 19 02:12 mysql-community-client-5.7.29-1.el7.x86_64.rpm
-rw-r--r--. 1 7155 31415    318972 Dec 19 02:12 mysql-community-common-5.7.29-1.el7.x86_64.rpm
-rw-r--r--. 1 7155 31415   4085448 Dec 19 02:12 mysql-community-devel-5.7.29-1.el7.x86_64.rpm
-rw-r--r--. 1 7155 31415  47521016 Dec 19 02:12 mysql-community-embedded-5.7.29-1.el7.x86_64.rpm
-rw-r--r--. 1 7155 31415  23354680 Dec 19 02:12 mysql-community-embedded-compat-5.7.29-1.el7.x86_64.rpm
-rw-r--r--. 1 7155 31415 131015588 Dec 19 02:12 mysql-community-embedded-devel-5.7.29-1.el7.x86_64.rpm
-rw-r--r--. 1 7155 31415   2596180 Dec 19 02:12 mysql-community-libs-5.7.29-1.el7.x86_64.rpm
-rw-r--r--. 1 7155 31415   1353080 Dec 19 02:12 mysql-community-libs-compat-5.7.29-1.el7.x86_64.rpm
-rw-r--r--. 1 7155 31415 183618644 Dec 19 02:12 mysql-community-server-5.7.29-1.el7.x86_64.rpm
-rw-r--r--. 1 7155 31415 124193252 Dec 19 02:12 mysql-community-test-5.7.29-1.el7.x86_64.rpm
```

Figure 4-94 Binaries files after untar

3. Install the `mysql-community-common` package.

   This package contains common files that are needed by MySQL Client Library and MySQL Server (see Figure 4-95):

   ```
   # yum install mysql-community-common-5.7.29-1.el7.x86_64.rpm
   ```

```
[root@itop-new tmp]# yum install -y mysql-community-common-5.7.29-1.el7.x86_64.rpm
```

Figure 4-95 Install of mysql-community-common package

4. Install the `mysql-community-libs` package.

   This package contains the shared libraries for MySQL Client Applications (see Figure 4-96):

   ```
   # yum install mysql-community-libs-5.7.29-1.el7.x86_64.rpm
   ```

```
[root@itop-new tmp]# yum install -y mysql-community-libs-5.7.29-1.el7.x86_64.rpm
```

Figure 4-96 Install mysql-community-libs package

5. Install the `mysql-community-client` package by running the following command (Figure 4-97):

   ```
   # yum install mysql-community-client-5.7.29-1.el7.x86_64.rpm
   ```

```
[root@itop-new tmp]# yum install -y mysql-community-client-5.7.29-1.el7.x86_64.rpm
```

Figure 4-97 Install mysql-community-client package

6. Install the `mysql-community-server` package (see Figure 4-98):

   ```
   # yum install mysql-community-server-5.7.29-1.el7.x86_64.rpm
   ```

```
[root@itop-new tmp]# yum install -y mysql-community-server-5.7.29-1.el7.x86_64.rpm
```

Figure 4-98 Install mysql-community-server package

7. Initialize the database (see Figure 4-99):

   ```
   # mysqld -initialize -user=mysql
   ```

```
[root@itop-new tmp]# mysqld --initialize --user=mysql
```

Figure 4-99 Initialize the database

8. After the installation is complete, browse to the data directory that is specified in `/etc/my.cnf` and check whether it is populated with data, as shown in Figure 4-100.

```
[root@itop-new mysql]# ll
total 123080
-rw-r-----. 1 mysql mysql       56 Apr 13 00:51 auto.cnf
-rw-------. 1 mysql mysql     1676 Apr 13 00:51 ca-key.pem
-rw-r--r--. 1 mysql mysql     1112 Apr 13 00:51 ca.pem
-rw-r--r--. 1 mysql mysql     1112 Apr 13 00:51 client-cert.pem
-rw-------. 1 mysql mysql     1676 Apr 13 00:51 client-key.pem
-rw-r-----. 1 mysql mysql      302 Apr 13 01:04 ib_buffer_pool
-rw-r-----. 1 mysql mysql 12582912 Apr 13 01:04 ibdata1
-rw-r-----. 1 mysql mysql 50331648 Apr 13 01:04 ib_logfile0
-rw-r-----. 1 mysql mysql 50331648 Apr 13 00:51 ib_logfile1
-rw-r-----. 1 mysql mysql 12582912 Apr 13 01:04 ibtmp1
drwxr-x---. 2 mysql mysql     4096 Apr 13 00:51 mysql
srwxrwxrwx. 1 mysql mysql        0 Apr 13 01:04 mysql.sock
-rw-------. 1 mysql mysql        5 Apr 13 01:04 mysql.sock.lock
drwxr-x---. 2 mysql mysql     4096 Apr 13 00:51 performance_schema
-rw-------. 1 mysql mysql     1676 Apr 13 00:51 private_key.pem
-rw-r--r--. 1 mysql mysql      452 Apr 13 00:51 public_key.pem
-rw-r--r--. 1 mysql mysql     1112 Apr 13 00:51 server-cert.pem
-rw-------. 1 mysql mysql     1680 Apr 13 00:51 server-key.pem
drwxr-x---. 2 mysql mysql    12288 Apr 13 00:51 sys
```

Figure 4-100 Postinstallation check

9. Retrieve the temporary password for the MySQL Login.

   The **`mysqld -initialize`** command generates an initial temporary password for root user. This password is stored in the `mysql.log` file (as shown in Figure 4-101) that is specified in the `my.cnf`. Copy this password for logging in to MySQL Service.

> **Note:** Without this password, it is not possible to log in to MySQL for the first time.

```
[root@itop-new mysql]# cat /var/log/mysqld.log
2020-04-13T05:51:27.262450Z 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit_defaults_for_t
imestamp server option (see documentation for more details).
2020-04-13T05:51:28.831651Z 0 [Warning] InnoDB: New log files created, LSN=45790
2020-04-13T05:51:29.010799Z 0 [Warning] InnoDB: Creating foreign key constraint system tables.
2020-04-13T05:51:29.082434Z 0 [Warning] No existing UUID has been found, so we assume that this is the first time that this serve
r has been started. Generating a new UUID: d1e699db-7d4a-11ea-9235-067d4d055b27.
2020-04-13T05:51:29.087865Z 0 [Warning] Gtid table is not ready to be used. Table 'mysql.gtid_executed' cannot be opened.
2020-04-13T05:51:29.970968Z 0 [Warning] CA certificate ca.pem is self signed.
2020-04-13T05:51:30.314350Z 1 [Note] A temporary password is generated for root@localhost: :e6Tac-!t.b,
```

Figure 4-101 Password stored in mysql.log file

> **Note:** The password can be found on the last line of the file or search the file for string `temporary password is generated`.

10. Start the MySQL service (see Figure 4-102):

```
# service mysqld start
```

```
[root@itop-new mysql]# service mysqld start
```

Figure 4-102 Start MySQL service

11. Check if MySQL process is running (see Figure 4-103):

```
# ps -ef | grep mysql
```

```
[root@itop-new mysql]# ps -ef| grep mysql
mysql     6657     1  0 01:04 ?        00:00:01 /usr/sbin/mysqld --daemonize --pid-file=/var/run/mysqld/mysqld.pid
```

Figure 4-103 Check MySQL process

12. Run the **mysql_secure_configuration** command for securing the mysql:

```
# mysql_secure_configuration
```

Change the password for the root user, as shown in Figure 4-104.

> **Note:** When prompted by the message `Enter password for user root`, enter the temporary password that is generated in the `mysql.log` file. This process sets the new password as required.

```
[root@itop-new mysql]# mysql_secure_installation

Securing the MySQL server deployment.

Enter password for user root:

The existing password for the user account root has expired. Please set a new password.

New password:

Re-enter new password:
```

Figure 4-104 Changing the root user password

After the password is set, follow the instructions to secure mysql service. Log in to MySQL service by using the new root password.

13. Install PHP.

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely used open source general purpose scripting language that is well-suited for web development and can be embedded into HTML.

a. Install PHP packages (see Figure 4-105 on page 314):

```
# yum -y install php php-mysql php-common php-gd php-mbstring php-mcrypt
php-devel php-xml php-imap php-ldap php-mbstring php-odbc php-pear php-xmlrpc
php-soap php-cli graphviz
```

> **Note:** For brevity, only the initial installation output and the completion output are shown.

```
[root@itop-new ~]#
[root@itop-new ~]# yum -y install php php-mysql php-common php-gd php-mbstring p
hp-mcrypt php-devel \
>    php-xml php-imap php-ldap php-mbstring php-odbc php-pear php-xmlrpc php-soa
p \
>    php-cli graphviz
Loaded plugins: product-id, search-disabled-repos, subscription-manager
rhel-7-server-optional-rpms                                | 1.8 kB     00:00
rhel-7-server-rpms                                         | 2.0 kB     00:00
rhel-7-server-supplementary-rpms                           | 2.0 kB     00:00
No package php-mcrypt available.
No package php-imap available.
Resolving Dependencies
--> Running transaction check
---> Package graphviz.x86_64 0:2.30.1-21.el7 will be installed
--> Processing Dependency: urw-fonts for package: graphviz-2.30.1-21.el7.x86_64
--> Processing Dependency: librsvg-2.so.2()(64bit) for package: graphviz-2.30.1-
21.el7.x86_64
--> Processing Dependency: libpangoft2-1.0.so.0()(64bit) for package: graphviz-2
.30.1-21.el7.x86_64
--> Processing Dependency: libpangocairo-1.0.so.0()(64bit) for package: graphviz
-2.30.1-21.el7.x86_64
--> Processing Dependency: libpango-1.0.so.0()(64bit) for package: graphviz-2.30
.1-21.el7.x86_64
```

Figure 4-105 Install PHP packages

Figure 4-106 shows the output of the completion of the install of the PHP packages.

```
  urw-base35-nimbus-mono-ps-fonts.noarch 0:20170801-10.el7
  urw-base35-nimbus-roman-fonts.noarch 0:20170801-10.el7
  urw-base35-nimbus-sans-fonts.noarch 0:20170801-10.el7
  urw-base35-p052-fonts.noarch 0:20170801-10.el7
  urw-base35-standard-symbols-ps-fonts.noarch 0:20170801-10.el7
  urw-base35-z003-fonts.noarch 0:20170801-10.el7
  xorg-x11-font-utils.x86_64 1:7.5-21.el7
  xorg-x11-server-utils.x86_64 0:7.7-20.el7

Complete!
[root@itop-new ~]#
```

Figure 4-106 Completion of the installation of the PHP packages

b. Adjust the following PHP settings (see Figure 4-107):

```
# vi /etc/php.ini
post_max_size = 32M
-- Save & Quit (:wq)
```

```
; Most likely, you won't want to disable this option globally. It causes $_POST
; and $_FILES to always be empty; the only way you will be able to read the
; POST data will be through the php://input stream wrapper. This can be useful
; to proxy requests or to process the POST data in a memory efficient fashion.
; http://php.net/enable-post-data-reading
;enable_post_data_reading = Off

; Maximum size of POST data that PHP will accept.
; Its value may be 0 to disable the limit. It is ignored if POST data reading
; is disabled through enable_post_data_reading.
; http://php.net/post-max-size
post_max_size = 32M

; Automatically add files before PHP document.
; http://php.net/auto-prepend-file
auto_prepend_file =

; Automatically add files after PHP document.
; http://php.net/auto-append-file
auto_append_file =

; By default, PHP will output a character encoding using
; the Content-type: header.  To disable sending of the charset, simply
:wq!
```

Figure 4-107 Adjusting a PHP setting

c. Restart Apache server to load the new configuration (see Figure 4-108):

```
# service httpd restart
```

```
[root@itop-new ~]#
[root@itop-new ~]# service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[root@itop-new ~]#
[root@itop-new ~]#
```

Figure 4-108 Restart the Apache server

14. Download and install iTop:

   a. Install zip and extract packages (see Figure 4-109 and Figure 4-110):

   ```
   # yum -y install zip unzip
   ```

```
[root@itop-new ~]# yum -y install zip unzip
Loaded plugins: product-id, search-disabled-repos, subscription-manager
rhel-7-server-optional-rpms                                | 1.8 kB     00:00
rhel-7-server-rpms                                         | 2.0 kB     00:00
rhel-7-server-supplementary-rpms                           | 2.0 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package unzip.x86_64 0:6.0-21.el7 will be installed
---> Package zip.x86_64 0:3.0-11.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package      Arch          Version              Repository                 Size
================================================================================
Installing:
 unzip        x86_64        6.0-21.el7           rhel-7-server-rpms         171 k
 zip          x86_64        3.0-11.el7           rhel-7-server-rpms         260 k

Transaction Summary
================================================================================
Install  2 Packages
```

Figure 4-109 Install zip

```
Total download size: 431 k
Installed size: 1.1 M
Downloading packages:
(1/2): unzip-6.0-21.el7.x86_64.rpm                         | 171 kB   00:00
(2/2): zip-3.0-11.el7.x86_64.rpm                           | 260 kB   00:00
--------------------------------------------------------------------------------
Total                                              1.2 MB/s | 431 kB  00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : zip-3.0-11.el7.x86_64                                       1/2
  Installing : unzip-6.0-21.el7.x86_64                                     2/2
Loaded plugins: product-id, subscription-manager
  Verifying  : unzip-6.0-21.el7.x86_64                                     1/2
  Verifying  : zip-3.0-11.el7.x86_64                                       2/2

Installed:
  unzip.x86_64 0:6.0-21.el7              zip.x86_64 0:3.0-11.el7

Complete!
[root@itop-new ~]# _
```

Figure 4-110 Extract files

b. Download iTop package under `/var/www/html` (see Figure 4-111):

```
# cd /var/www/html
# wget
https://sourceforge.net/projects/itop/files/itop/2.6.3/iTop-2.6.3-5092.zip
```

```
[root@itop-new html]#
[root@itop-new html]# wget https://sourceforge.net/projects/itop/files/itop/2.6.
3/iTop-2.6.3-5092.zip
--2020-04-11 07:15:11--  https://sourceforge.net/projects/itop/files/itop/2.6.3/
iTop-2.6.3-5092.zip
Resolving sourceforge.net (sourceforge.net)... 216.105.38.13
Connecting to sourceforge.net (sourceforge.net)|216.105.38.13|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://sourceforge.net/projects/itop/files/itop/2.6.3/iTop-2.6.3-5092
.zip/ [following]
--2020-04-11 07:15:13--  https://sourceforge.net/projects/itop/files/itop/2.6.3/
iTop-2.6.3-5092.zip/
Connecting to sourceforge.net (sourceforge.net)|216.105.38.13|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://sourceforge.net/projects/itop/files/itop/2.6.3/iTop-2.6.3-5092
.zip/download [following]
--2020-04-11 07:15:14--  https://sourceforge.net/projects/itop/files/itop/2.6.3/
iTop-2.6.3-5092.zip/download
Connecting to sourceforge.net (sourceforge.net)|216.105.38.13|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://downloads.sourceforge.net/project/itop/itop/2.6.3/iTop-2.6.3-5
092.zip?r=&ts=1586607314&use_mirror=gigenet [following]
--2020-04-11 07:15:15--  https://downloads.sourceforge.net/project/itop/itop/2.6
.3/iTop-2.6.3-5092.zip?r=&ts=1586607314&use_mirror=gigenet
```

Figure 4-111 Download iTop

Figure 4-112 shows the completion of the download.

```
Resolving downloads.sourceforge.net (downloads.sourceforge.net)... 216.105.38.13
Connecting to downloads.sourceforge.net (downloads.sourceforge.net)|216.105.38.1
3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://gigenet.dl.sourceforge.net/project/itop/itop/2.6.3/iTop-2.6.3-
5092.zip [following]
--2020-04-11 07:15:16--  https://gigenet.dl.sourceforge.net/project/itop/itop/2.
6.3/iTop-2.6.3-5092.zip
Resolving gigenet.dl.sourceforge.net (gigenet.dl.sourceforge.net)... 69.65.16.14
2
Connecting to gigenet.dl.sourceforge.net (gigenet.dl.sourceforge.net)|69.65.16.1
42|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12568355 (12M) [application/octet-stream]
Saving to: 'iTop-2.6.3-5092.zip'

100%[======================================>] 12,568,355  2.50MB/s   in 6.7s

2020-04-11 07:15:24 (1.78 MB/s) - 'iTop-2.6.3-5092.zip' saved [12568355/12568355
]

[root@itop-new html]#
```

Figure 4-112 Downloading iTop continues

c. Extract the package and rename the web directory as iTop (see Figure 4-113):

```
# unzip iTop-2.6.3-5092.zip
# mv web itop
```

```
[root@itop-new html]#
[root@itop-new html]# ls -l
total 12340
-rw-r--r--.  1 root root      106 Jan 22 09:55 INSTALL
-rw-r--r--.  1 root root 12568355 Feb  5 04:02 iTop-2.6.3-5092.zip
-rw-r--r--.  1 root root    34520 Oct 18  2018 LICENSE
-rw-r--r--.  1 root root     2069 Jan 22 09:55 README
drwxr-xr-x. 21 root root     4096 Apr 11 07:32 web
[root@itop-new html]#
[root@itop-new html]#
[root@itop-new html]# mv web itop
[root@itop-new html]#
[root@itop-new html]#
[root@itop-new html]# ls -l
total 12340
-rw-r--r--.  1 root root      106 Jan 22 09:55 INSTALL
drwxr-xr-x. 21 root root     4096 Apr 11 07:32 itop
-rw-r--r--.  1 root root 12568355 Feb  5 04:02 iTop-2.6.3-5092.zip
-rw-r--r--.  1 root root    34520 Oct 18  2018 LICENSE
-rw-r--r--.  1 root root     2069 Jan 22 09:55 README
[root@itop-new html]#
[root@itop-new html]# _
```

Figure 4-113 Rename the web directory

d. Create the following directory and make them writable:

```
# mkdir /var/www/html/itop/conf
# mkdir /var/www/html/itop/env-production
# chmod 777 /var/www/html/itop/conf/
# chmod 777 /var/www/html/itop/data
# chmod 777 /var/www/html/itop/env-production/
# chmod 777 /var/www/html/itop/log
```

e. Complete the installation by using a web browser:

```
http://<Server_ip_address>/itop
```

Installing iTop V2.7.0-beta as Red Hat Enterprise Linux container

Complete the following steps to install iTop V2.7.0:

1. Podman is a tool for running Linux containers. Subscribe, then enable extras channel to install Podman (see Figure 4-114):

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-7-server-extras-rpms
# subscription-manager repos --enable=rhel-7-server-optional-rpms
```

```
[root@ems ~]#
[root@ems ~]# subscription-manager repos --enable=rhel-7-server-rpms
Repository 'rhel-7-server-rpms' is enabled for this system.
[root@ems ~]#
[root@ems ~]#
[root@ems ~]#
[root@ems ~]# subscription-manager repos --enable=rhel-7-server-extras-rpms
Repository 'rhel-7-server-extras-rpms' is enabled for this system.
[root@ems ~]#
[root@ems ~]#
[root@ems ~]# subscription-manager repos --enable=rhel-7-server-optional-rpms
Repository 'rhel-7-server-optional-rpms' is enabled for this system.
[root@ems ~]#
```

Figure 4-114 Subscribe and enable extras channels

2. Install Podman (see Figure 4-115):

```
# yum install podman -y
```

```
[root@ems ~]#
[root@ems ~]# yum install podman -y
Loaded plugins: product-id, search-disabled-repos, subscription-manager
rhel-7-server-extras-rpms                                                                                                                                | 2.0 kB  00:00:00
rhel-7-server-optional-rpms                                                                                                                              | 1.8 kB  00:00:00
rhel-7-server-rpms                                                                                                                                       | 2.0 kB  00:00:00
rhel-7-server-supplementary-rpms                                                                                                                         | 2.0 kB  00:00:00
(1/3): rhel-7-server-extras-rpms/x86_64/group                                                                                                            |  147 B  00:00:00
(2/3): rhel-7-server-extras-rpms/x86_64/updateinfo                                                                                                       | 230 kB  00:00:00
(3/3): rhel-7-server-extras-rpms/x86_64/primary                                                                                                          | 389 kB  00:00:00
rhel-7-server-extras-rpms                                                                                                                                                1275/1275
Resolving Dependencies
--> Running transaction check
---> Package podman.x86_64 0:1.6.4-16.el7_8 will be installed
--> Processing Dependency: slirp4netns >= 0.4.0-1 for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: runc >= 1.0.0-57 for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: containers-common >= 0.1.29-3 for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: containernetworking-plugins >= 0.8.1-1 for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: nftables for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: libseccomp for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: fuse-overlayfs for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: container-selinux for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: conmon for package: podman-1.6.4-16.el7_8.x86_64
--> Processing Dependency: libseccomp.so.2()(64bit) for package: podman-1.6.4-16.el7_8.x86_64
--> Running transaction check
---> Package conmon.x86_64 2:2.0.8-1.el7 will be installed
---> Package container-selinux.noarch 2:2.119.1-1.c57a6f9.el7 will be installed
--> Processing Dependency: policycoreutils-python for package: 2:container-selinux-2.119.1-1.c57a6f9.el7.noarch
---> Package containernetworking-plugins.x86_64 0:0.8.3-2.el7 will be installed
---> Package containers-common.x86_64 1:0.1.40-7.el7_8 will be installed
---> Package fuse-overlayfs.x86_64 0:0.7.2-6.el7_8 will be installed
--> Processing Dependency: libfuse3.so.3(FUSE_3.2)(64bit) for package: fuse-overlayfs-0.7.2-6.el7_8.x86_64
--> Processing Dependency: libfuse3.so.3(FUSE_3.0)(64bit) for package: fuse-overlayfs-0.7.2-6.el7_8.x86_64
--> Processing Dependency: libfuse3.so.3()(64bit) for package: fuse-overlayfs-0.7.2-6.el7_8.x86_64
---> Package libseccomp.x86_64 0:2.3.1-4.el7 will be installed
---> Package nftables.x86_64 1:0.8-14.el7 will be installed
--> Processing Dependency: libnftnl.so.7(LIBNFTNL_5)(64bit) for package: 1:nftables-0.8-14.el7.x86_64
--> Processing Dependency: libnftnl.so.7()(64bit) for package: 1:nftables-0.8-14.el7.x86_64
---> Package runc.x86_64 0:1.0.0-67.rc10.el7_8 will be installed
--> Processing Dependency: criu for package: runc-1.0.0-67.rc10.el7_8.x86_64
---> Package slirp4netns.x86_64 0:0.4.3-4.el7_8 will be installed
--> Running transaction check
---> Package criu.x86_64 0:3.12-2.el7 will be installed
--> Processing Dependency: libprotobuf-c.so.1(LIBPROTOBUF_C_1.0.0)(64bit) for package: criu-3.12-2.el7.x86_64
--> Processing Dependency: libprotobuf-c.so.1()(64bit) for package: criu-3.12-2.el7.x86_64
```

Figure 4-115 Install Podman

Figure 4-116 shows the completion of the installation of Podman. A few output windows were omitted.

```
  Verifying  : protobuf-c-1.0.2-3.el7.x86_64                                                                                                                                12/24
  Verifying  : runc-1.0.0-67.rc10.el7_8.x86_64                                                                                                                              13/24
  Verifying  : 1:containers-common-0.1.40-7.el7_8.x86_64                                                                                                                    14/24
  Verifying  : libsemanage-python-2.5-14.el7.x86_64                                                                                                                         15/24
  Verifying  : slirp4netns-0.4.3-4.el7_8.x86_64                                                                                                                             16/24
  Verifying  : criu-3.12-2.el7.x86_64                                                                                                                                       17/24
  Verifying  : libnet-1.1.6-7.el7.x86_64                                                                                                                                    18/24
  Verifying  : podman-1.6.4-16.el7_8.x86_64                                                                                                                                 19/24
  Verifying  : 2:conmon-2.0.8-1.el7.x86_64                                                                                                                                  20/24
  Verifying  : libnftnl-1.0.8-3.el7.x86_64                                                                                                                                  21/24
  Verifying  : libcgroup-0.41-21.el7.x86_64                                                                                                                                 22/24
  Verifying  : containernetworking-plugins-0.8.3-2.el7.x86_64                                                                                                               23/24
  Verifying  : policycoreutils-2.5-33.el7.x86_64                                                                                                                            24/24
rhel-7-server-extras-rpms/x86_64/productid                                                                                                               | 2.1 kB  00:00:00
rhel-7-server-supplementary-rpms/7Server/x86_64/productid                                                                                                | 2.1 kB  00:00:00

Installed:
  podman.x86_64 0:1.6.4-16.el7_8

Dependency Installed:
  audit-libs-python.x86_64 0:2.8.5-4.el7                checkpolicy.x86_64 0:2.5-8.el7                        conmon.x86_64 2:2.0.8-1.el7
  container-selinux.noarch 2:2.119.1-1.c57a6f9.el7      containernetworking-plugins.x86_64 0:0.8.3-2.el7      containers-common.x86_64 1:0.1.40-7.el7_8
  criu.x86_64 0:3.12-2.el7                              fuse-overlayfs.x86_64 0:0.7.2-6.el7_8                 fuse3-libs.x86_64 0:3.6.1-4.el7
  libcgroup.x86_64 0:0.41-21.el7                        libnet.x86_64 0:1.1.6-7.el7                           libnftnl.x86_64 0:1.0.8-3.el7
  libseccomp.x86_64 0:2.3.1-4.el7                       libsemanage-python.x86_64 0:2.5-14.el7                nftables.x86_64 1:0.8-14.el7
  policycoreutils-python.x86_64 0:2.5-34.el7            protobuf-c.x86_64 0:1.0.2-3.el7                       python-IPy.noarch 0:0.75-6.el7
  runc.x86_64 0:1.0.0-67.rc10.el7_8                     setools-libs.x86_64 0:3.3.8-4.el7                     slirp4netns.x86_64 0:0.4.3-4.el7_8

Dependency Updated:
  policycoreutils.x86_64 0:2.5-34.el7

Complete!
[root@ems ~]#
[root@ems ~]#
```

Figure 4-116 Completing the installation of Podman

3. Install the latest Prometheus container image (see Figure 4-117):

```
# podman pull vbkunin/itop
```

```
[root@ems ~]#
[root@ems ~]# podman pull vbkunin/itop
Trying to pull registry.access.redhat.com/vbkunin/itop...
  name unknown: Repo not found
Trying to pull registry.fedoraproject.org/vbkunin/itop...
  manifest unknown: manifest unknown
Trying to pull registry.centos.org/vbkunin/itop...
  manifest unknown: manifest unknown
Trying to pull docker.io/vbkunin/itop...
Getting image source signatures
Copying blob a3ed95caeb02 skipped: already exists
Copying blob 2aea1b77cff7 skipped: already exists
Copying blob 281a73dee007 skipped: already exists
Copying blob 0218064da0a9 skipped: already exists
Copying blob a3ed95caeb02 skipped: already exists
Copying blob ebac621dcea3 skipped: already exists
Copying blob b5807316443cc skipped: already exists
Copying blob faa5fbdba239 skipped: already exists
Copying blob 59a714b7d8bf skipped: already exists
Copying blob a3ed95caeb02 skipped: already exists
Copying blob 64744e45912d skipped: already exists
Copying blob e6e648699d59 skipped: already exists
Copying blob 494fead19a8e skipped: already exists
Copying blob f18365dc4e03 skipped: already exists
Copying blob d7413153377l skipped: already exists
Copying blob 364493d13210 skipped: already exists
Copying blob 36dbdaba5fcc skipped: already exists
Copying blob c094f88699db skipped: already exists
Copying blob 03d1f2bc5774 skipped: already exists
Copying blob 0e6ffdcdbc5a skipped: already exists
Copying blob 2c2b82b3db0f skipped: already exists
Copying blob 73e866103fb8 skipped: already exists
Copying blob bc7bf2113b25 skipped: already exists
Copying blob 97bc2628ee3b skipped: already exists
Copying blob 57c655e75303 skipped: already exists
Copying config f2ad9745b2 done
Writing manifest to image destination
Storing signatures
f2ad9745b22a93b69e2f98b98637810a6185de33205619fe06c225789b3dbaf6
[root@ems ~]#
[root@ems ~]#
```

Figure 4-117 Install Prometheus container image

4. Run iTop container (see Figure 4-118):

```
# podman run -d -p 8000:80 --name=my-itop vbkunin/itop:2.7.0-beta
```

```
[root@ems ~]#
[root@ems ~]# podman run -d -p 8000:80 --name=my-itop vbkunin/itop:2.7.0-beta
45ab469c9b02e0268a4c65c701f18e4fba255f24034dafa6a73552de0deac870
[root@ems ~]#
[root@ems ~]#
```

Figure 4-118 Running iTop container

5. Check the status (see Figure 4-119):

```
# podman ps
```

```
[root@ems ~]#
[root@ems ~]#
[root@ems ~]#
[root@ems ~]# podman ps
CONTAINER ID  IMAGE                              COMMAND               CREATED         STATUS             PORTS                   NAMES
45ab469c9b02  docker.io/vbkunin/itop:2.7.0-beta                        31 seconds ago  Up 30 seconds ago  0.0.0.0:8000->80/tcp    my-itop
```

Figure 4-119 Status check

6. Complete the installation by using a web browser:

```
http://<Server_ip_address>/itop
```

iTop installation wizard

This section shows the iTop installation wizard:

1. Assess the iTop installation setup by using a web browser:

```
http://<Server_ip_address>/itop
```

2. Click **Continue** (see Figure 4-120).

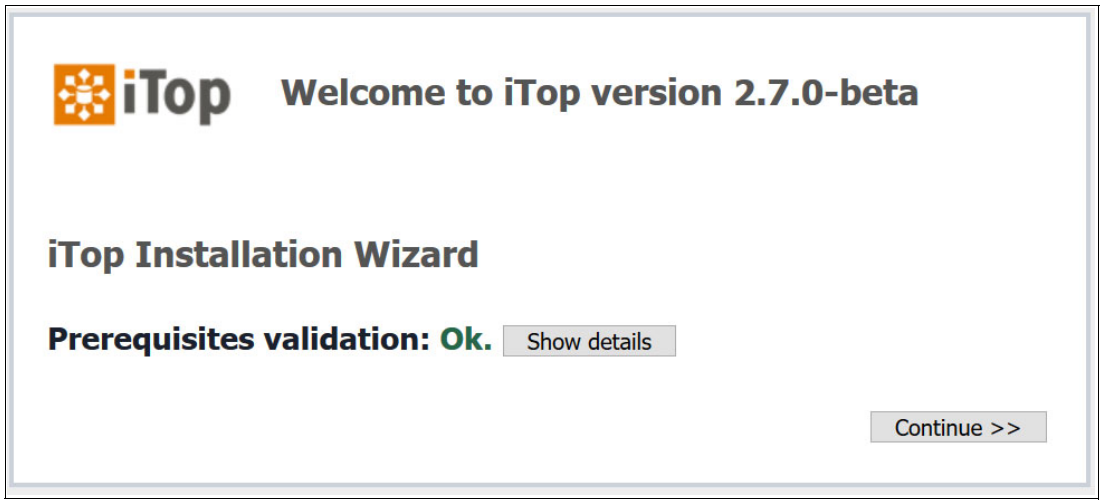


Figure 4-120 iTop Installation Wizard window

3. Select **Install a New iTop** and then, click **Next**, as shown in Figure 4-121.

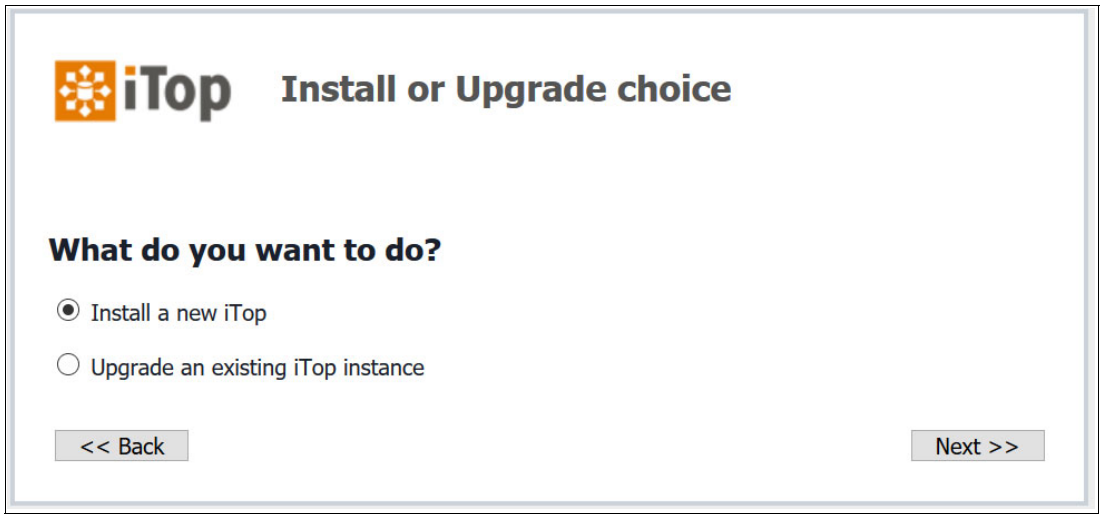


Figure 4-121 iTop: Install or Upgrade choice window

4. Select the **I Accept the terms of the licenses of the 59 components mentioned above** box and then, click **Next**, as shown in Figure 4-122.

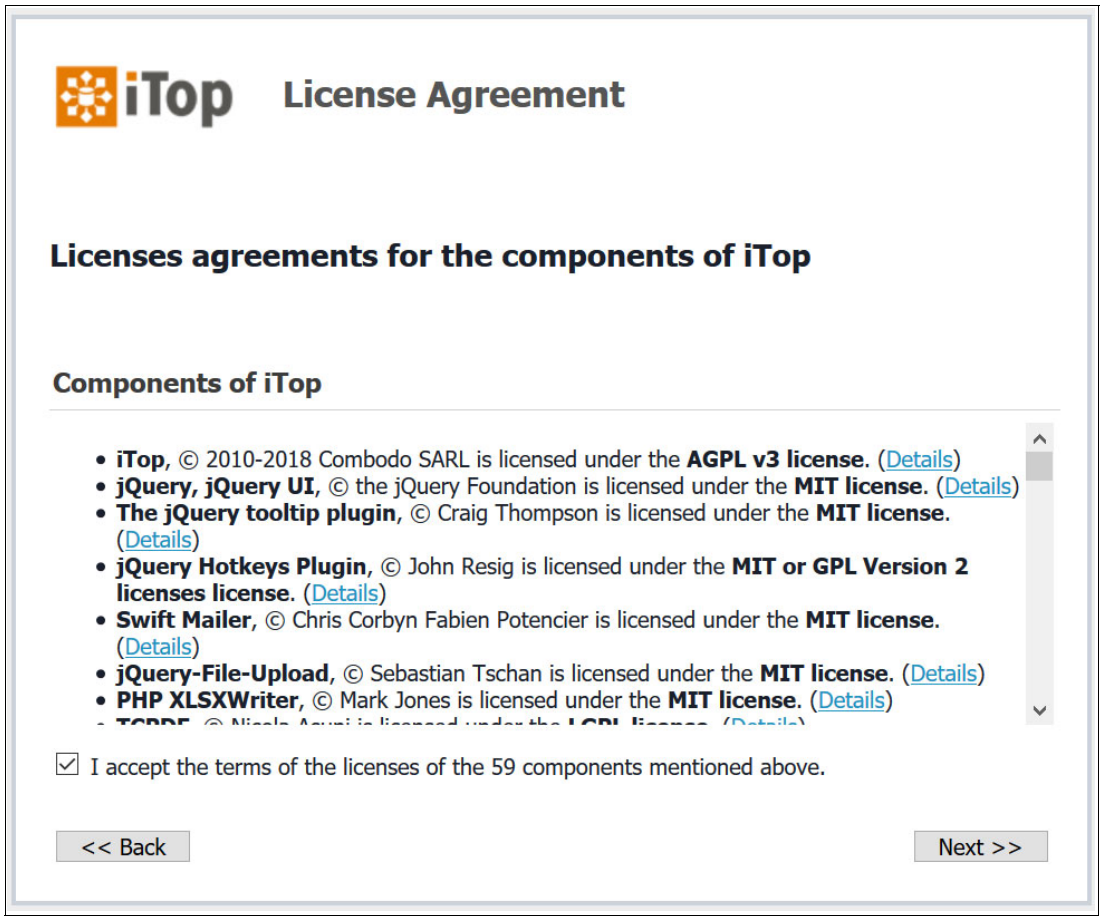


Figure 4-122 iTop License Agreement window

5. Enter the following database server connection:
   - Server Name: `localhost`
   - Login: `root`
   - Password: Default password is blank. You can use the password you created.
   - In the Database field, select **Create a new Database**: `itop_db`

6. Click **Next**, as shown in Figure 4-123.

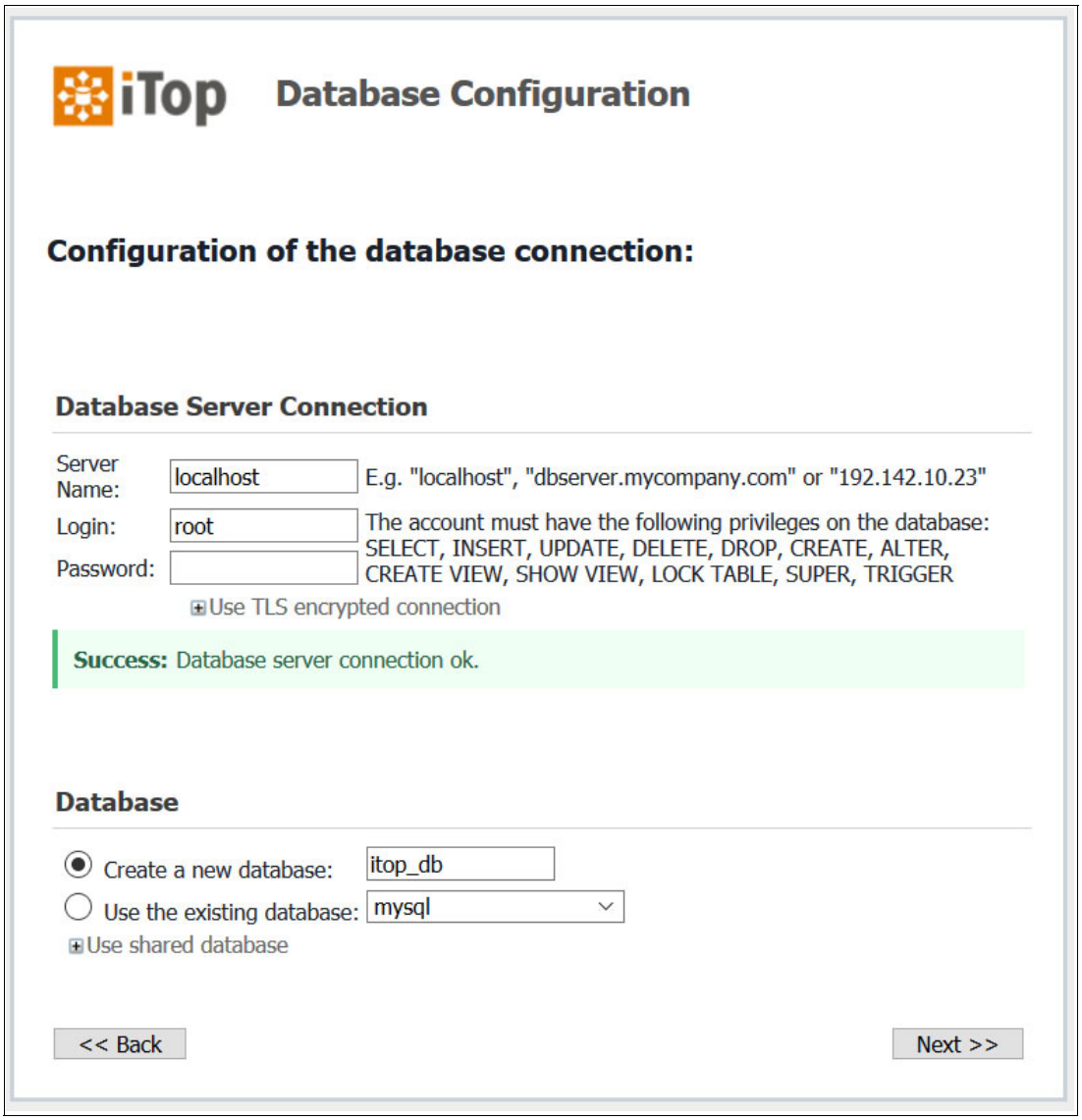


Figure 4-123 iTop Database Configuration window

7. Enter the following Administrator Account information:
   - Login: `admin`
   - Password: `<Password>`
   - Confirm password: `<Password>`
   - Language: `English`

8. Click **Next**, as shown in Figure 4-124.

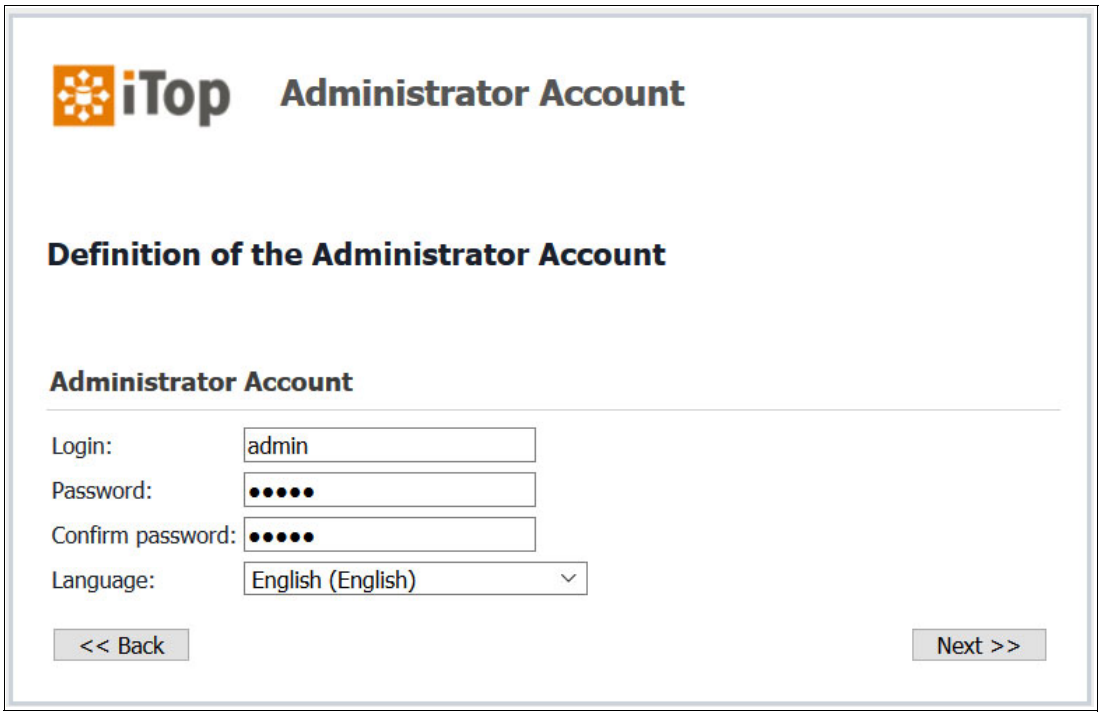


Figure 4-124 iTop Administrator Account window

9. For the Sample Data option, if you directly use it in production environment, select the **I am installing a production instance, create an empty database to start from** option and click **Next** (see Figure 4-125).

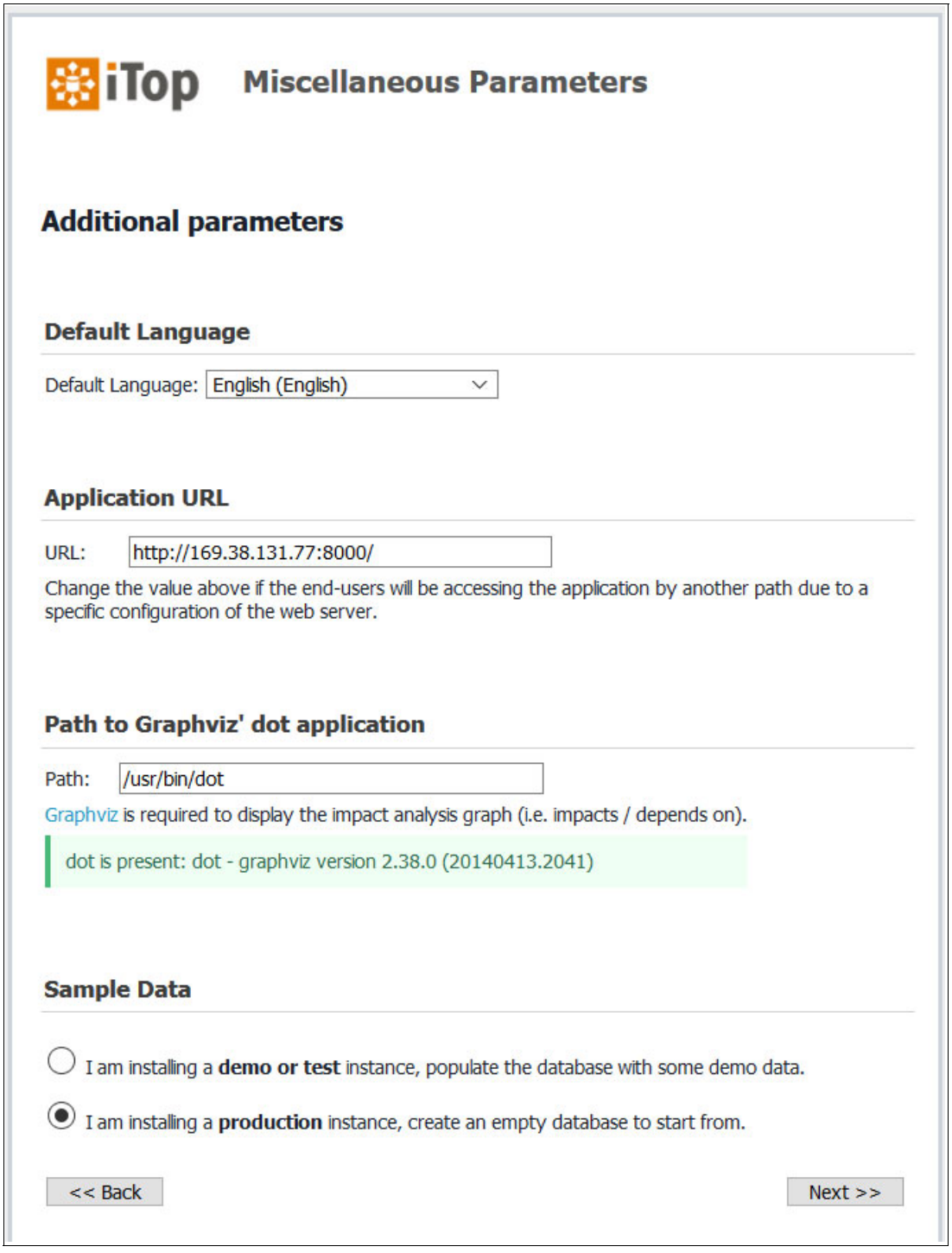


Figure 4-125 iTop Miscellaneous Parameters window

10. Select the type of CIs you need inside your CMDB and click **Next**, as shown in Figure 4-126.

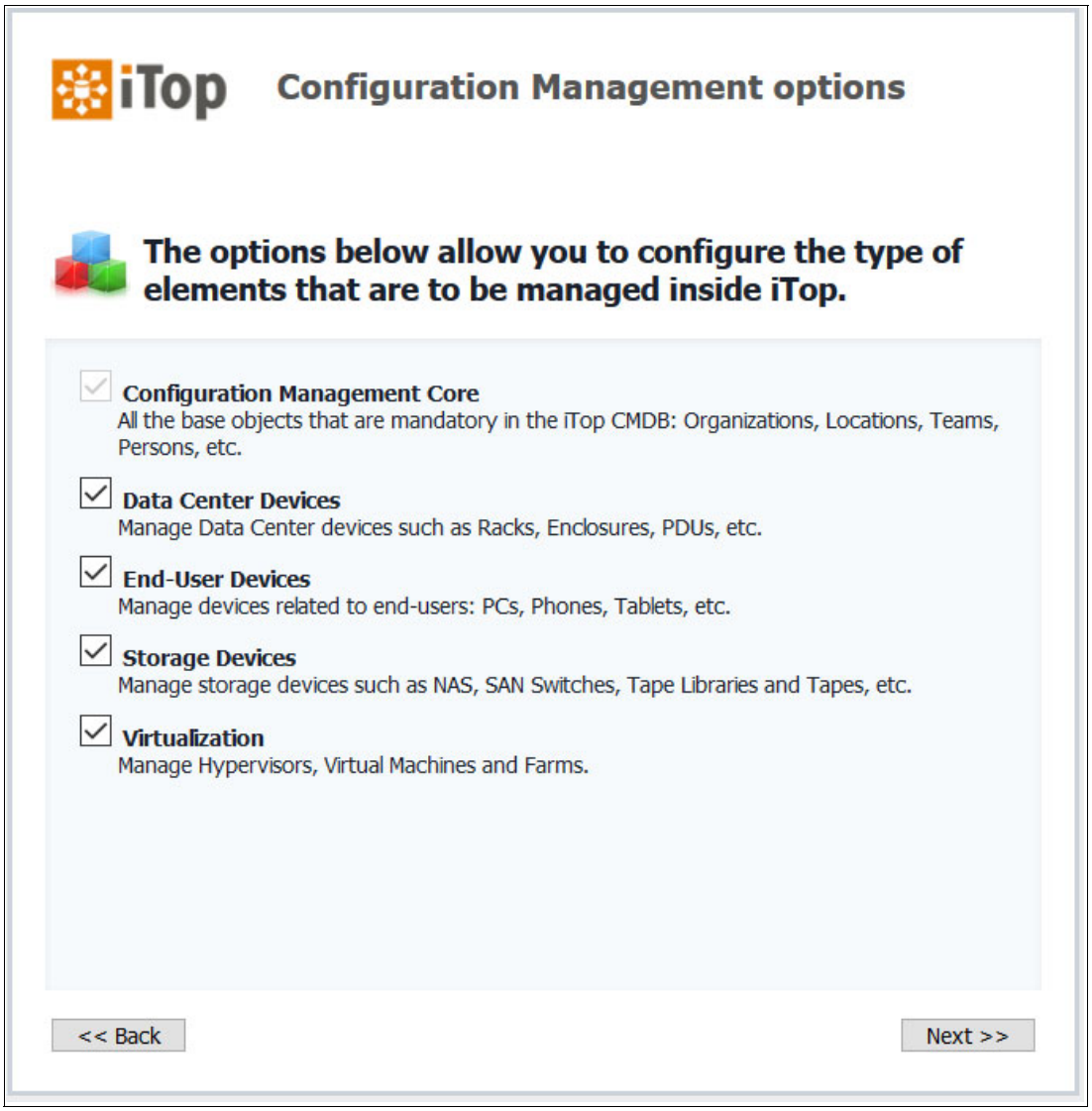


Figure 4-126 iTop Configuration Management options window

11. Select **Service Management for Enterprises** and click **Next**, as shown in Figure 4-127.

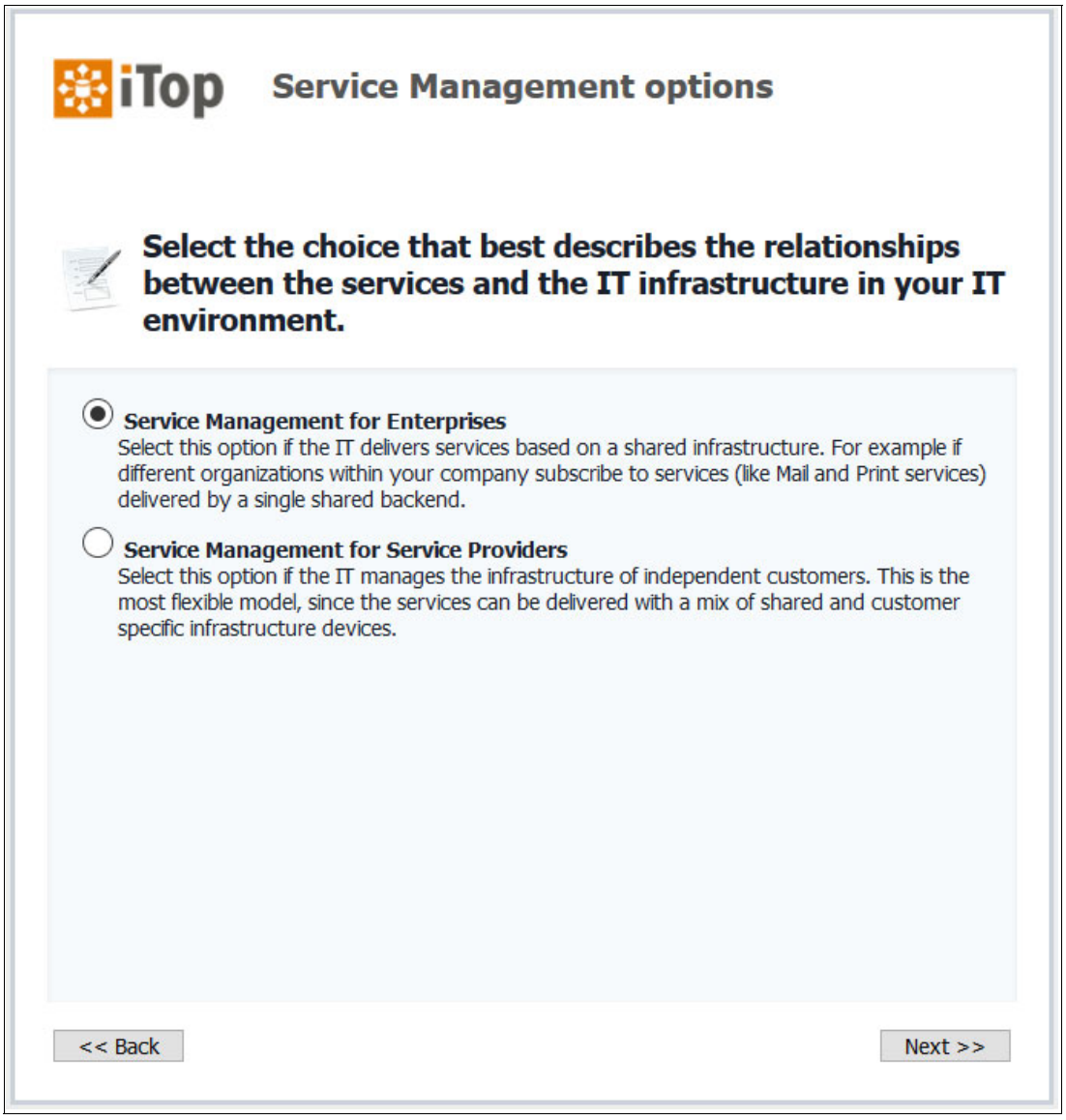


Figure 4-127 iTop Service Management options window

12. Select **ITIL Compliant Tickets Management** → **User Request Management**, **Incident Management**, and **Customer Portal**. Click **Next**, as shown in Figure 4-128.

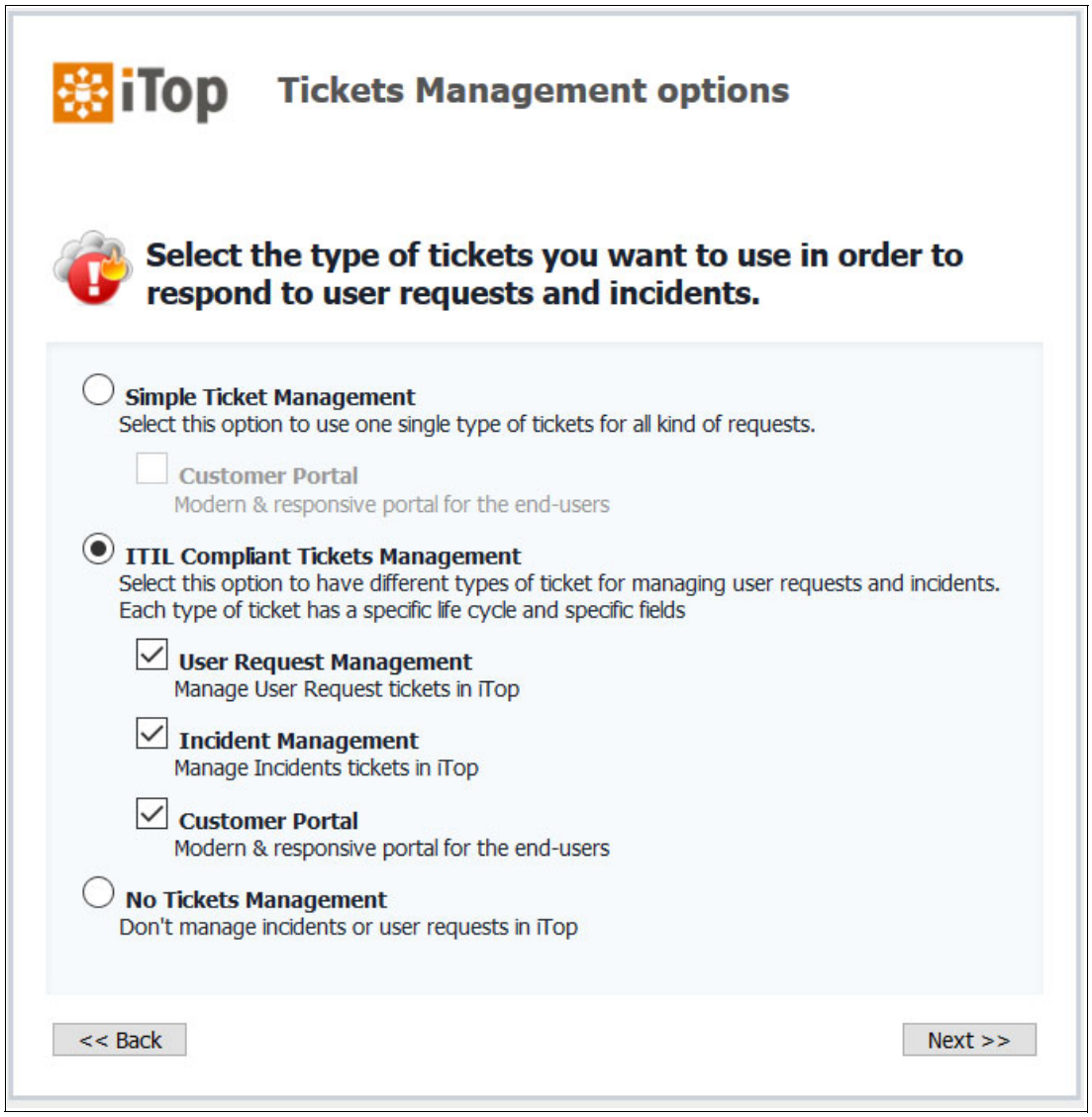


Figure 4-128 iTop Tickets Management options window

13. Select **ITIL Change Management** and click **Next**, as shown in Figure 4-129.

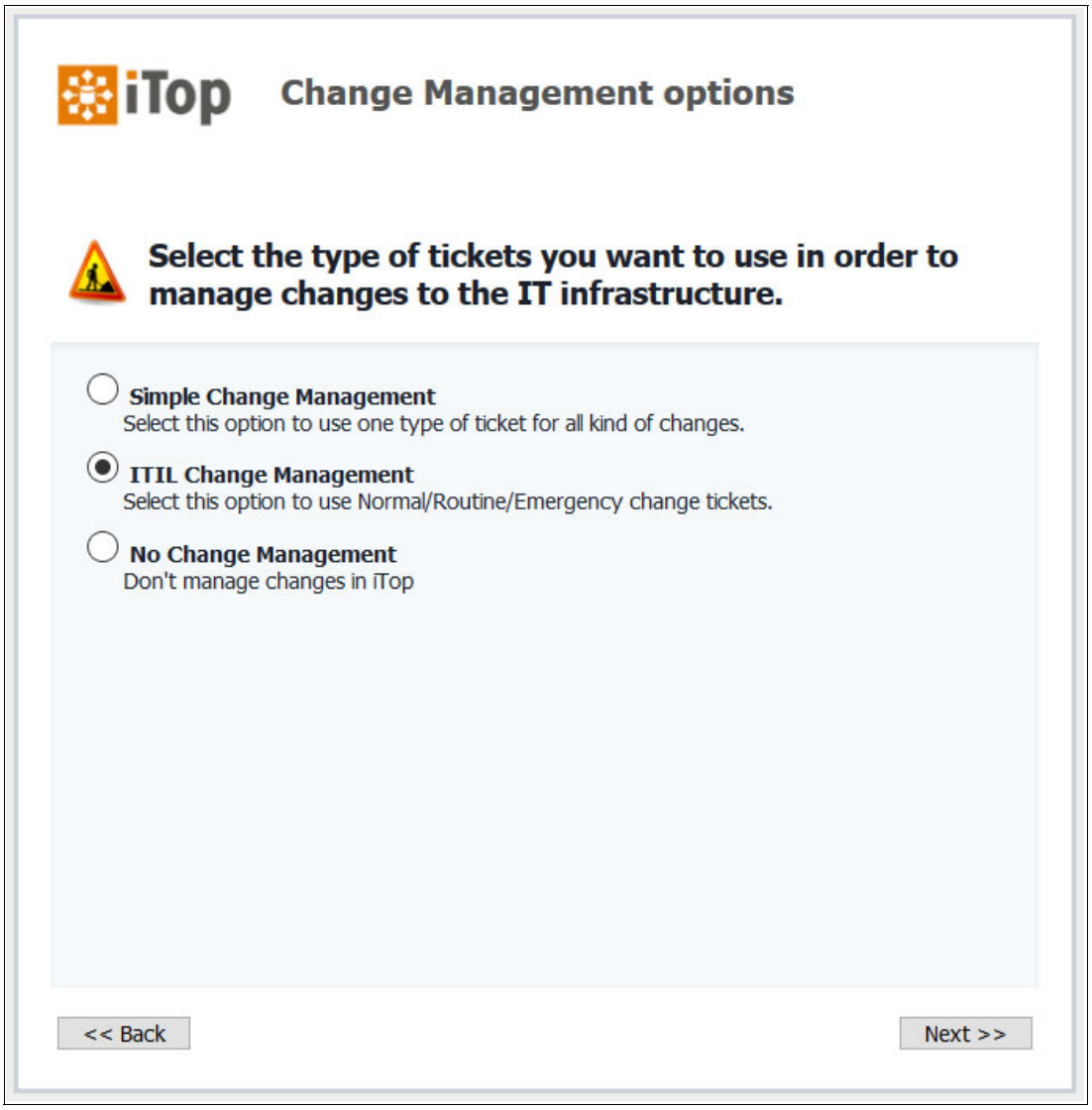


Figure 4-129 iTop Change Management options window

14. Select the **Known Errors Management** and **Problem Management** options. Click **Next**, as shown in Figure 4-130.

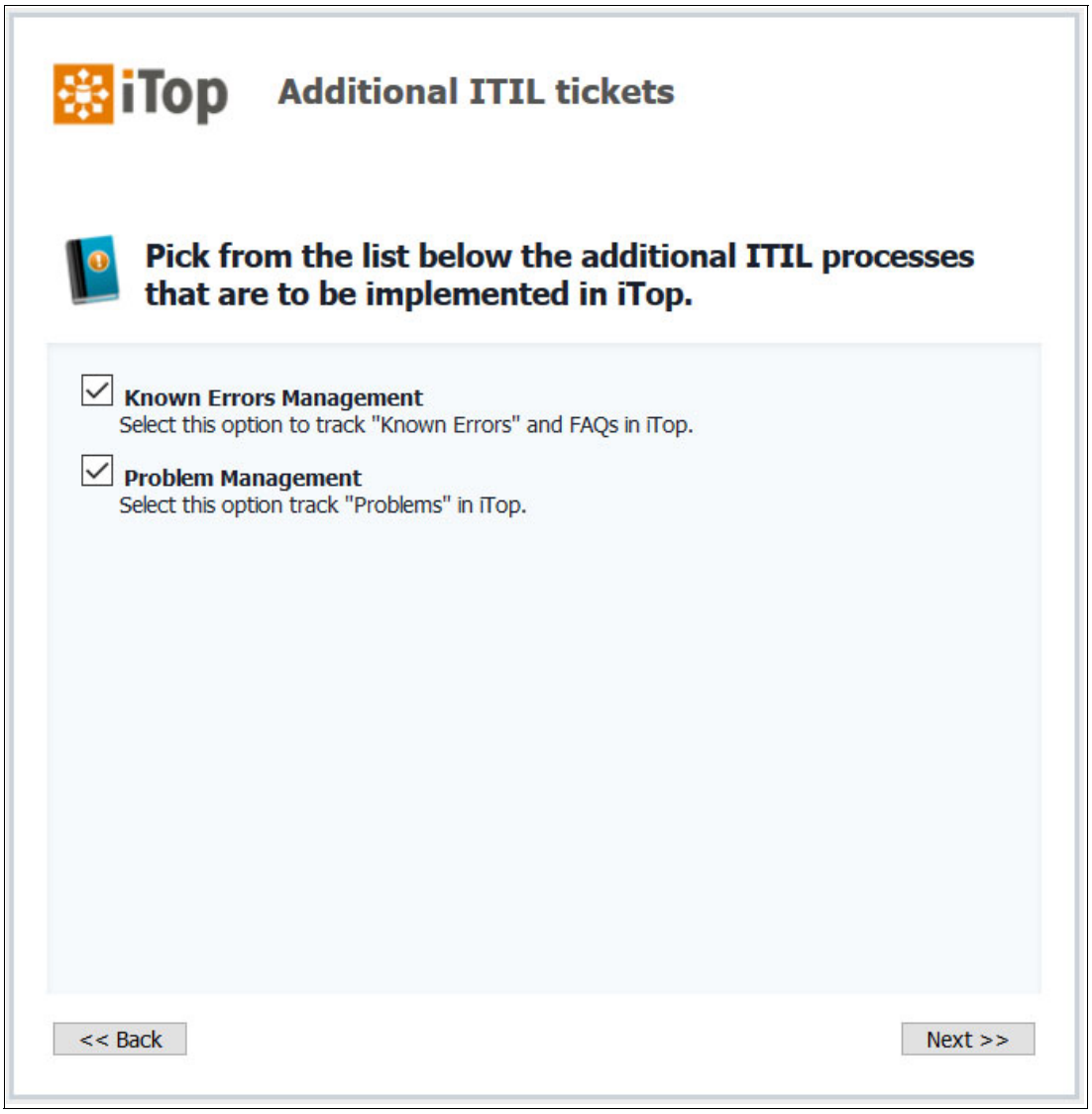


Figure 4-130 iTop Additional ITIL tickets window

15. Click **Install**, as shown in Figure 4-131.

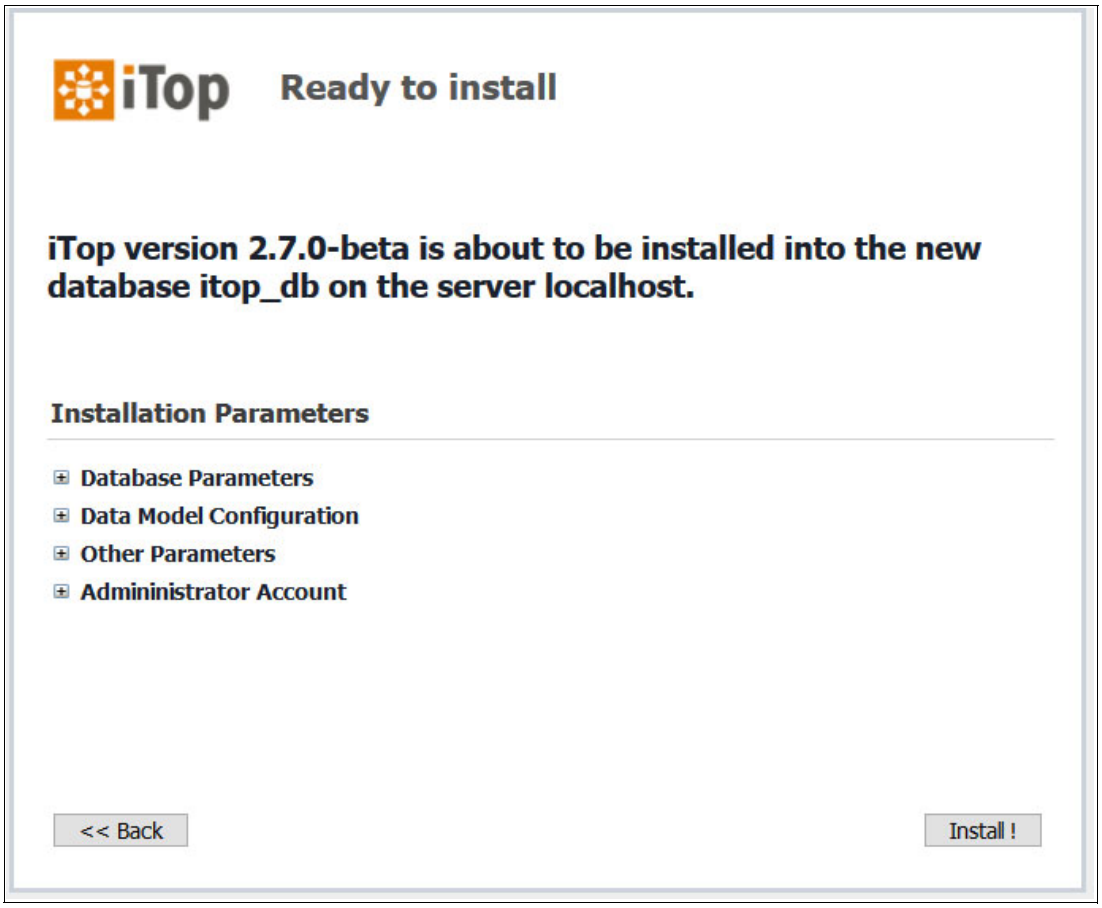


Figure 4-131 iTop Ready to install window

Figure 4-132 shows the completion of the installation. Click **Enter iTop**.

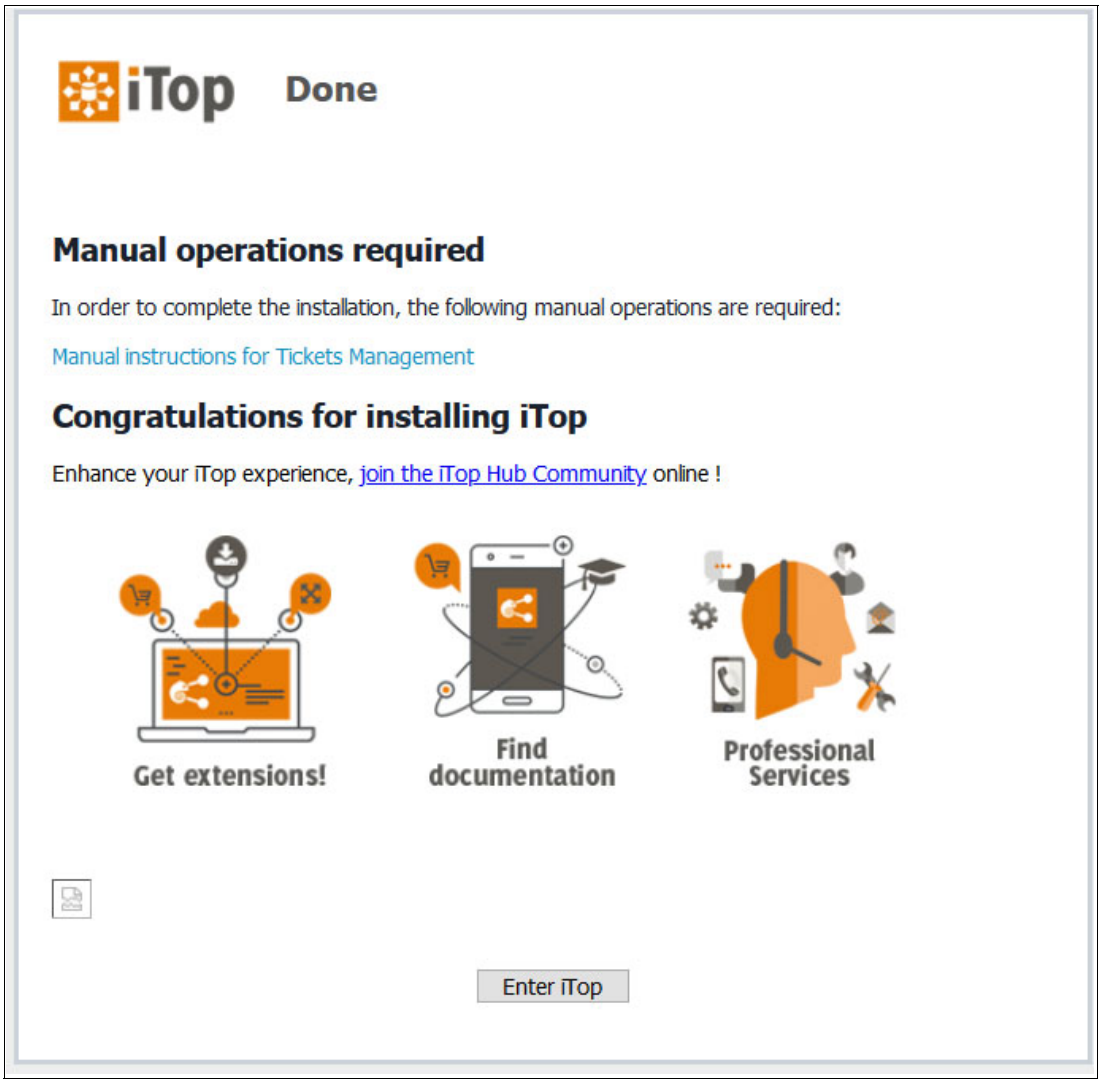


Figure 4-132 iTop Done window

The iTop dashboard window is displayed as shown in Figure 4-133.

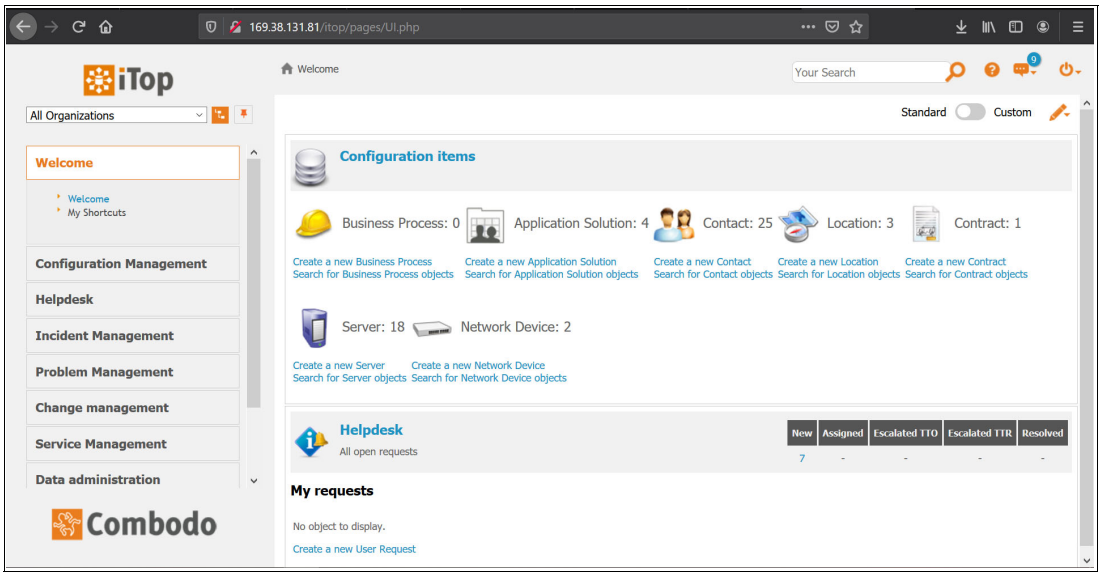


Figure 4-133 iTop dashboard

4.4.2 Generating an App password for Postfix to access Google accounts

This section explains how to generate an App password for third-party applications; for example, Postfix to access Google accounts.

Prerequisites

To generate a password for Postfix, you must have a valid Google Gmail account.

The solution

Complete the following steps to generate a password for Postfix:

1. Log in to Google Gmail, as shown in Figure 4-134.

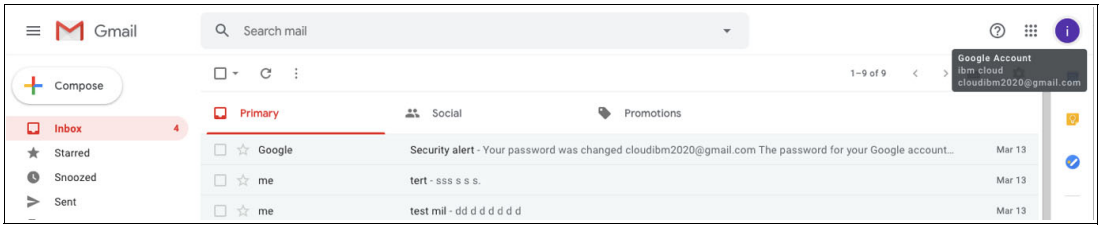


Figure 4-134 Shows access to an email account

2. Click **Account Setting** (upper right corner) and then, click the **Manage Your Google Account** tab, as shown in Figure 4-135.

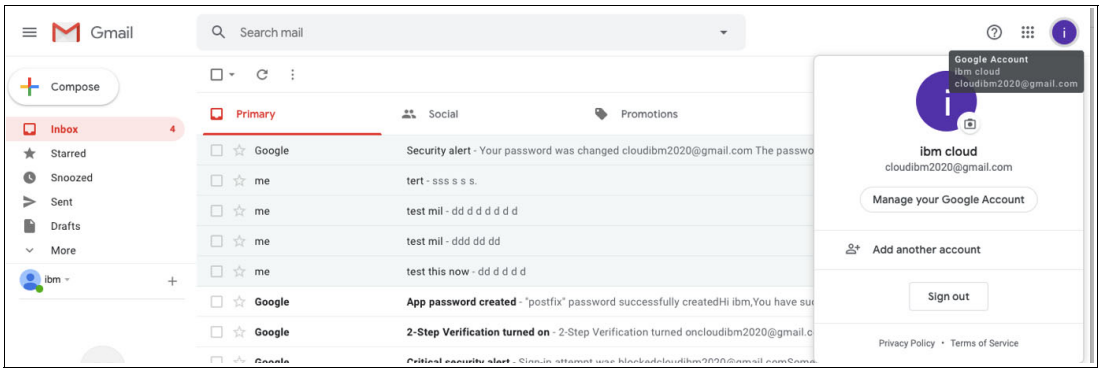


Figure 4-135 Account configuration settings: Manage Google account

3. Click **Security** in the upper left corner of the window, as shown in Figure 4-136.

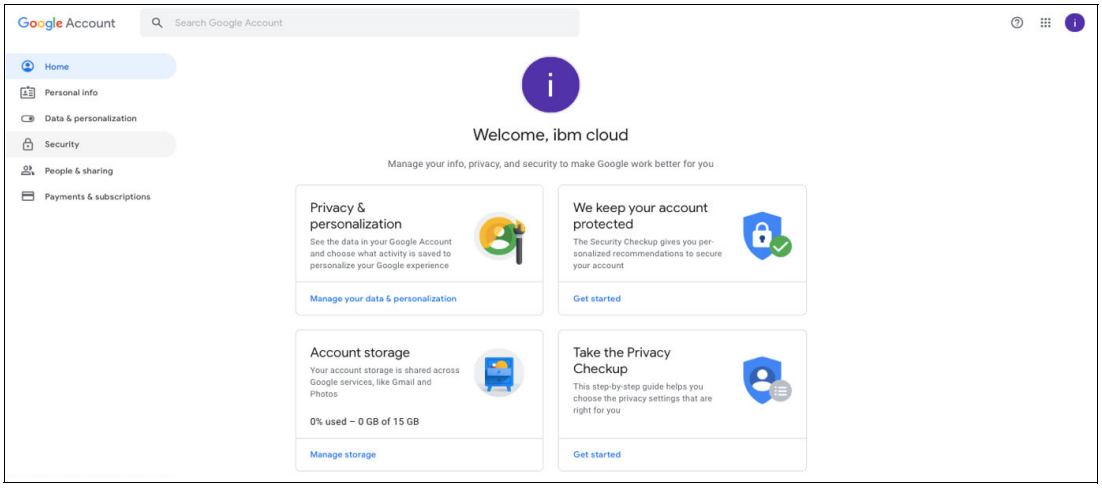


Figure 4-136 Google account: Security configuration

The Security settings window opens, as shown in Figure 4-137.

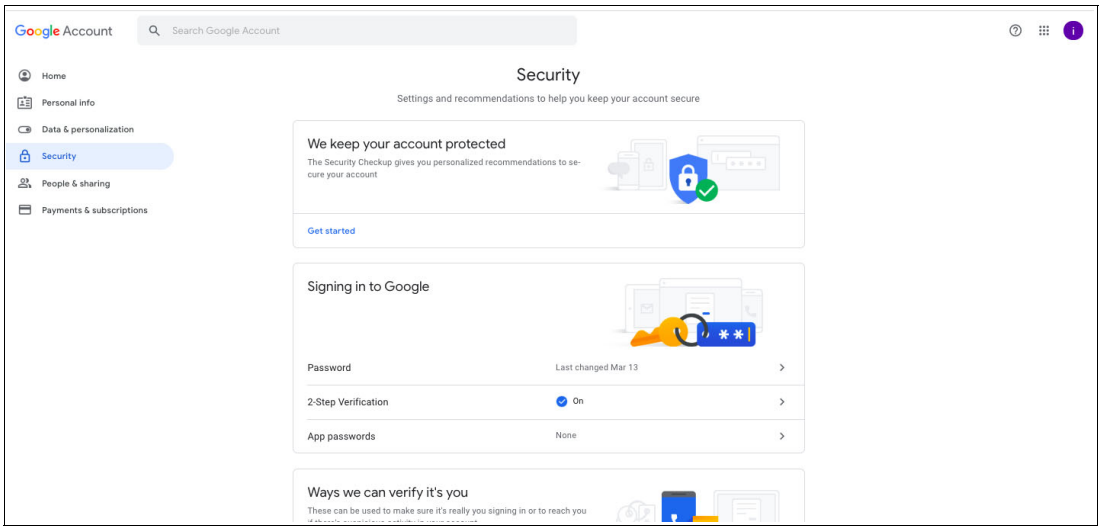


Figure 4-137 Google account: Security settings pane

4. Click **App passwords** and enter the Gmail account password and click **Sign In** (see Figure 4-138).

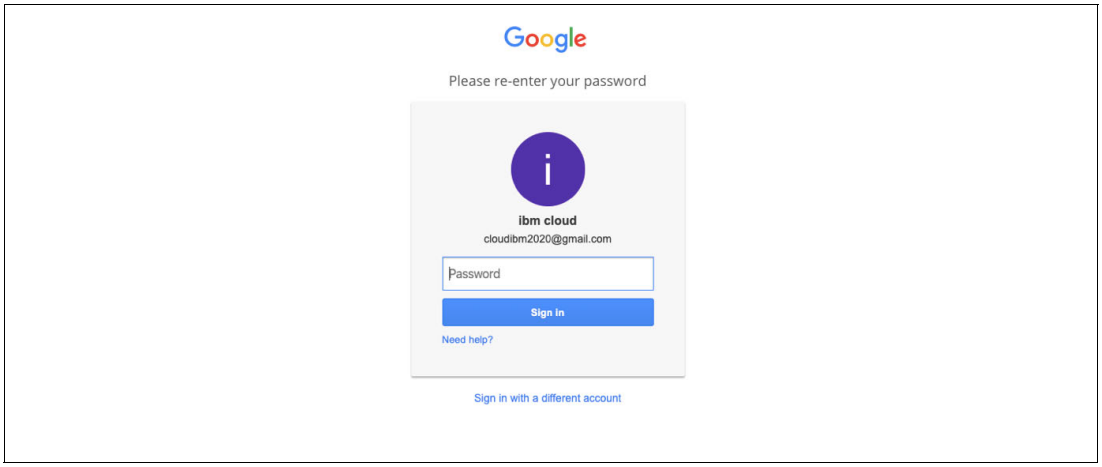


Figure 4-138 Enter account password

The App passwords window opens, as shown in Figure 4-139.

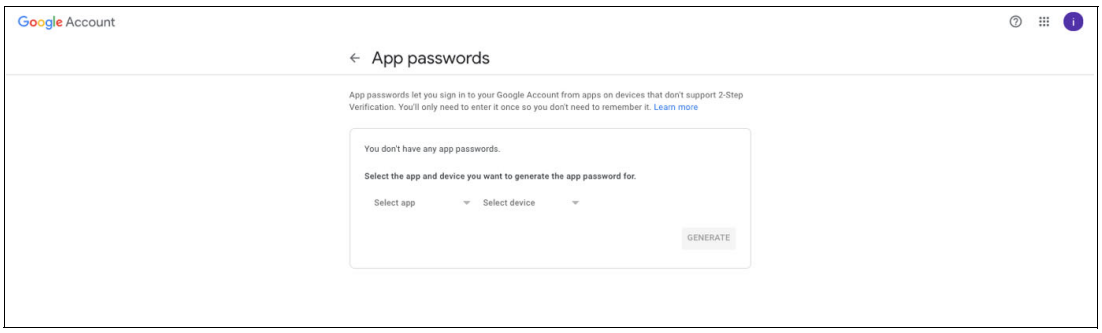


Figure 4-139 Google Account: App password pane

5. In the Select app section, select the **other (Custom Name)** option and enter `postfix`, as shown in Figure 4-140.

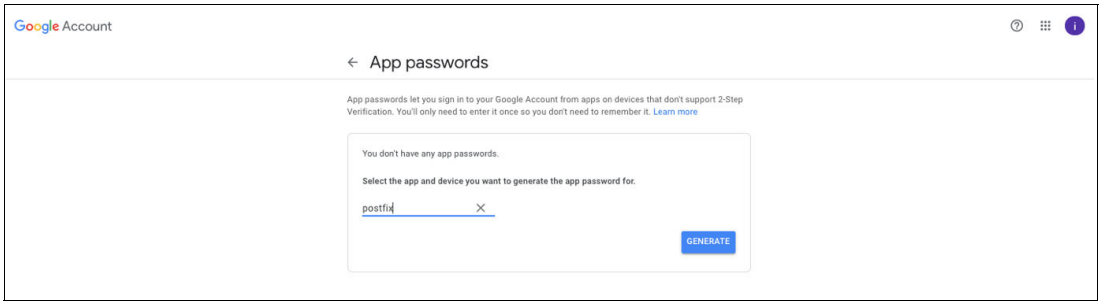


Figure 4-140 App password window: Select application to generate password

6. Click **GENERATE** to get the app password, as shown in Figure 4-141.

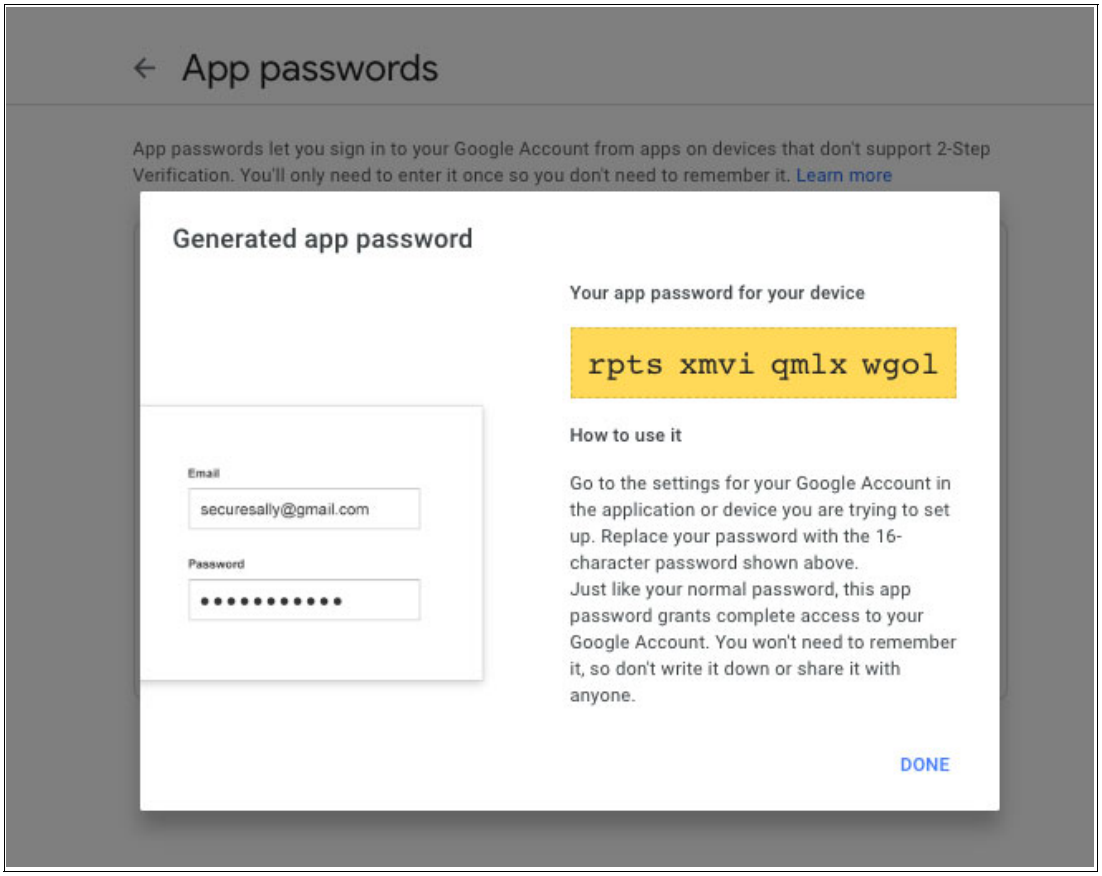


Figure 4-141 App password generated

The generated password is added to the `sasl_password` file during the Postfix configuration. For more information, see 4.4.3, “Mail extension with iTop” on page 337.

4.4.3 Mail extension with iTop

This section describes how to configure postfix for email messaging.

Traditional email transport

How the mail setup works when a mail server is in your environment is shown in Figure 4-142.

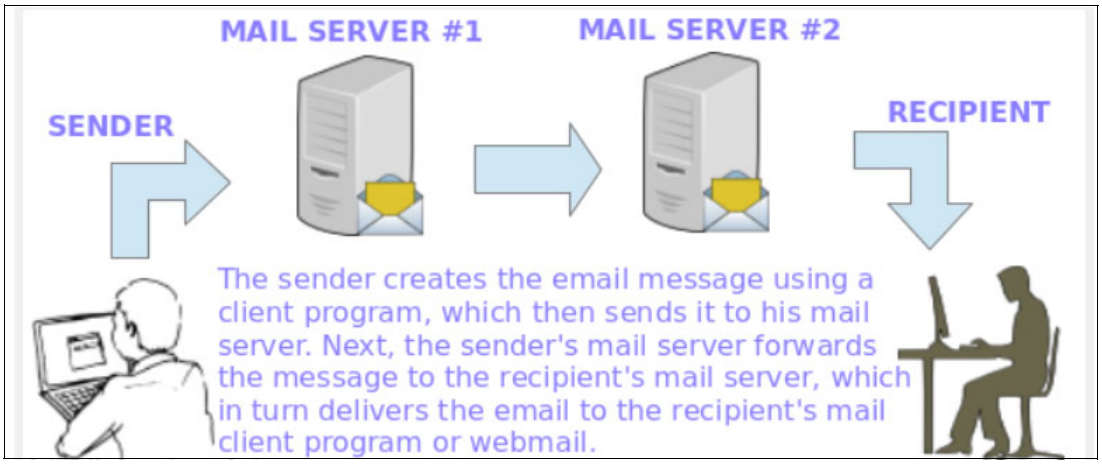


Figure 4-142 email framework

Problem statement

You can see a traditional mail transport in Figure 4-142, but this case requires to route mail from a machine in the cloud where no mail server exists. Therefore, the question is: How is the mail routed without an SMTP server?

Postfix must be configured to send email by using an external SMTP provider, such as Gmail, Mandrill, SendGrid, Amazon SES, or any other SMTP server, for several reasons. One reason is to avoid getting your mail flagged as spam if your current server's IP was added to a spam list.

Prerequisites

To implement Postfix, the following prerequisites must be met:

- Your fully qualified domain name (FQDN)
- All updates are installed
- A valid user name and password for the SMTP mail provider (Mandrill or SendGrid)
- The libsasl2-modules package is installed and current
- Postfix package is installed
- The cyrus-saslp package is installed

The solution

Complete the following steps to deploy the Postfix solution:

1. Install the Postfix by running **yum**, as shown in Figure 4-143.

```
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]# yum install postfix
```

Figure 4-143 Install Postfix

2. View the installed package, as shown in Figure 4-144.

```
[root@vsi-servicemanagement ~]# yum list postfix

Failed to set locale, defaulting to C
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Installed Packages
postfix.x86_64                                          2:2.10.1-7.el7                                          @base
[root@vsi-servicemanagement ~]#
```

Figure 4-144 Check the installed package

3. Install the `cyrus-sasl` packages by running **`yum`**, as shown in Figure 4-145.

```
[root@vsi-servicemanagement ~]# yum install cyrus-sasl-*
Failed to set locale, defaulting to C
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Package cyrus-sasl-lib-2.1.26-23.el7.x86_64 already installed and latest version
Package cyrus-sasl-plain-2.1.26-23.el7.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
---> Package cyrus-sasl.x86_64 0:2.1.26-23.el7 will be installed
---> Package cyrus-sasl-devel.x86_64 0:2.1.26-23.el7 will be installed
---> Package cyrus-sasl-gs2.x86_64 0:2.1.26-23.el7 will be installed
---> Package cyrus-sasl-gssapi.x86_64 0:2.1.26-23.el7 will be installed
---> Package cyrus-sasl-ldap.x86_64 0:2.1.26-23.el7 will be installed
---> Package cyrus-sasl-md5.x86_64 0:2.1.26-23.el7 will be installed
---> Package cyrus-sasl-ntlm.x86_64 0:2.1.26-23.el7 will be installed
---> Package cyrus-sasl-scram.x86_64 0:2.1.26-23.el7 will be installed
---> Package cyrus-sasl-sql.x86_64 0:2.1.26-23.el7 will be installed
--> Processing Dependency: libpq.so.5()(64bit) for package: cyrus-sasl-sql-2.1.26-23.el7.x86_64
--> Running transaction check
---> Package postgresql-libs.x86_64 0:9.2.24-2.el7_7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================================================
 Package                      Arch                 Version                          Repository                Size
================================================================================================================================
Installing:
 cyrus-sasl                   x86_64               2.1.26-23.el7                    base                      88 k
 cyrus-sasl-devel             x86_64               2.1.26-23.el7                    base                     310 k
 cyrus-sasl-gs2               x86_64               2.1.26-23.el7                    base                      41 k
 cyrus-sasl-gssapi            x86_64               2.1.26-23.el7                    base                      41 k
 cyrus-sasl-ldap              x86_64               2.1.26-23.el7                    base                      36 k
 cyrus-sasl-md5               x86_64               2.1.26-23.el7                    base                      57 k
 cyrus-sasl-ntlm              x86_64               2.1.26-23.el7                    base                      42 k
 cyrus-sasl-scram             x86_64               2.1.26-23.el7                    base                      43 k
 cyrus-sasl-sql               x86_64               2.1.26-23.el7                    base                      38 k
Installing for dependencies:
 postgresql-libs              x86_64               9.2.24-2.el7_7                   updates                  234 k

Transaction Summary
================================================================================================================================
Install  9 Packages (+1 Dependent package)
```

Figure 4-145 Install cyrus-sasl packages

4. View the installed package, as shown in Figure 4-146.

```
[root@vsi-servicemanagement ~]# yum list cyrus-sasl-*
Failed to set locale, defaulting to C
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Installed Packages
cyrus-sasl-lib.x86_64                                   2.1.26-23.el7                                   @anaconda
cyrus-sasl-plain.x86_64                                 2.1.26-23.el7                                   @base
Available Packages
cyrus-sasl.i686                                         2.1.26-23.el7                                   base
cyrus-sasl.x86_64                                       2.1.26-23.el7                                   base
cyrus-sasl-devel.i686                                   2.1.26-23.el7                                   base
cyrus-sasl-devel.x86_64                                 2.1.26-23.el7                                   base
cyrus-sasl-gs2.i686                                     2.1.26-23.el7                                   base
cyrus-sasl-gs2.x86_64                                   2.1.26-23.el7                                   base
cyrus-sasl-gssapi.i686                                  2.1.26-23.el7                                   base
cyrus-sasl-gssapi.x86_64                                2.1.26-23.el7                                   base
cyrus-sasl-ldap.i686                                    2.1.26-23.el7                                   base
cyrus-sasl-ldap.x86_64                                  2.1.26-23.el7                                   base
cyrus-sasl-lib.i686                                     2.1.26-23.el7                                   base
cyrus-sasl-md5.i686                                     2.1.26-23.el7                                   base
cyrus-sasl-md5.x86_64                                   2.1.26-23.el7                                   base
cyrus-sasl-ntlm.i686                                    2.1.26-23.el7                                   base
cyrus-sasl-ntlm.x86_64                                  2.1.26-23.el7                                   base
cyrus-sasl-plain.i686                                   2.1.26-23.el7                                   base
cyrus-sasl-scram.i686                                   2.1.26-23.el7                                   base
cyrus-sasl-scram.x86_64                                 2.1.26-23.el7                                   base
cyrus-sasl-sql.i686                                     2.1.26-23.el7                                   base
cyrus-sasl-sql.x86_64                                   2.1.26-23.el7                                   base
```

Figure 4-146 Installed packages

5. Configure Postfix by adding or changing the following parameters in the `/etc/postfix/main.cf` file (see Figure 4-147):
   - relayhost = [smtp.gmail.com]:587
   - smtp_sasl_auth_enable = yes
   - smtp_sasl_auth_enable = yes
   - smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
   - smtp_sasl_security_options = noanonymous
   - smtp_sasl_tls_security_options = noanonymous
   - smtp_use_tls = yes
   - smtp_tls_policy_maps = hash:/etc/postfix/tls_policy

```
[root@vsi-servicemanagement ~]# vi /etc/postfix/main.cf
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]# cat /etc/postfix/main.cf |grep -i smtp |grep -v "#"
relayhost = [smtp.gmail.com]:587
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_sasl_tls_security_options = noanonymous
smtp_use_tls = yes
smtp_tls_policy_maps = hash:/etc/postfix/tls_policy
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
```

Figure 4-147 Configure Postfix

6. Add Gmail's user name and password in the `/etc/postfix/sasl_passwd` file `[smtp.gmail.com]:587 cloudibm2020@gmail.com:uylugagvfofasqgt`, as shown in Figure 4-148.

```
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]# vi /etc/postfix/sasl_passwd
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]# cat /etc/postfix/sasl_passwd
[smtp.gmail.com]:587 cloudibm2020@gmail.com:uylugagvfofasqgt
[root@vsi-servicemanagement ~]#
[root@vsi-servicemanagement ~]#
```

Figure 4-148 Add email credentials

> **Note:** The password is the App password with which you sign in to your Google account from apps on devices that do not support 2-step verification.

For more information about generating an App password generation, see this web page.

7. Google requires TLS in the Postfix's SMTP client. This TLS is added in the `/etc/postfix/main.cf` file, which is related to `tls_policy`:

   smtp_tls_policy_maps = hash:/etc/postfix/tls_policy

8. Create the database files by running the following commands:

```
#chmod 600 /etc/postfix/sasl_passwd
#postmap /etc/postfix/sasl_passwd
#postmap /etc/postfix/tls_policy
```

9. Start the postfix service by running the following command:

```
#systemctl start postfix
```

10. Enable the postfix service by running the following command:

```
#systemctl enable postfix
```

11. Check the status of Postfix service, as shown in Figure 4-149.

Figure 4-149 Check status of postfix service

12. Send an email to the remote user and verify the status in the relayhost in the `/var/log/maillog` file, as shown in Figure 4-150.

Figure 4-150 Text email configuration

A test mail arrives in your Gmail inbox, as shown in Figure 4-151.

Figure 4-151 Test email arrives

4.4.4 Fixing Python-pip package installation issues

When you cannot install python modules or manager on server (for example, 169.38.67.66) whereas you can install the modules on another server (for example, 169.38.122.215), complete the following steps to address the issue:

1. List the required packages on both the servers, as shown in Figure 4-152 and Figure 4-153.

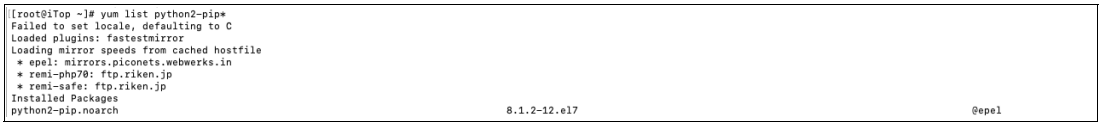


Figure 4-152 Server (iTop) on which the package is already installed

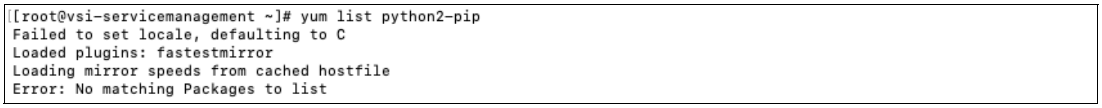


Figure 4-153 Server on which the package is not installed

2. Attempt to install the package on the server (`vsi-servicemanagement`). No package is available, as shown in Figure 4-154.

```
[root@vsi-servicemanagement ~]# yum install python-pip
Failed to set locale, defaulting to C
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
No package python-pip available.
Error: Nothing to do
[root@vsi-servicemanagement ~]# yum list python2-pip
Failed to set locale, defaulting to C
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Error: No matching Packages to list
```

Figure 4-154 Attempt installation: python package

Listing the contents show that the required package is not in the software repository; therefore, it is not available in the system, as shown in Figure 4-155.

```
[[root@vsi-servicemanagement ~]# cd /etc/yum.repos.d/
[[root@vsi-servicemanagement yum.repos.d]# ls -ltr
total 36
-rw-r--r--. 1 root root 1708 Oct 28  2014 CentOS-Base.repo
-rw-r--r--. 1 root root 1618 Oct 28  2014 CentOS-Base.repo.orig
-rw-r--r--. 1 root root  314 Sep  5  2019 CentOS-fasttrack.repo
-rw-r--r--. 1 root root 6639 Sep  5  2019 CentOS-Vault.repo
-rw-r--r--. 1 root root 1331 Sep  5  2019 CentOS-Sources.repo
-rw-r--r--. 1 root root  630 Sep  5  2019 CentOS-Media.repo
-rw-r--r--. 1 root root  649 Sep  5  2019 CentOS-Debuginfo.repo
-rw-r--r--. 1 root root 1309 Sep  5  2019 CentOS-CR.repo
[[root@vsi-servicemanagement yum.repos.d]#
```

Figure 4-155 List repository contents on vsi-servicemanagement

Listing the contents of the second system (iTop) shows that the required package is available in the software repositories, as shown in Figure 4-156.

```
[[root@iTop ~]# cd /etc/yum.repos.d/
[root@iTop yum.repos.d]# ls -ltr
total 112
-rw-r--r--. 1 root root 1209 Jan 29  2014 mysql-community.repo
-rw-r--r--. 1 root root 1060 Jan 29  2014 mysql-community-source.repo
-rw-r--r--. 1 root root 1708 Oct 28  2014 CentOS-Base.repo
-rw-r--r--. 1 root root 1618 Oct 28  2014 CentOS-Base.repo.orig
-rw-r--r--. 1 root root  947 Apr  2  2017 webtatic-archive.repo
-rw-r--r--. 1 root root  314 Sep  5  2019 CentOS-fasttrack.repo
-rw-r--r--. 1 root root 6639 Sep  5  2019 CentOS-Vault.repo
-rw-r--r--. 1 root root 1331 Sep  5  2019 CentOS-Sources.repo
-rw-r--r--. 1 root root  630 Sep  5  2019 CentOS-Media.repo
-rw-r--r--. 1 root root  649 Sep  5  2019 CentOS-Debuginfo.repo
-rw-r--r--. 1 root root 1309 Sep  5  2019 CentOS-CR.repo
-rw-r--r--. 1 root root 1050 Sep 17 18:25 epel.repo
-rw-r--r--. 1 root root 1149 Sep 17 18:25 epel-testing.repo
-rw-r--r--. 1 root root 2605 Feb 17 10:07 remi.repo
-rw-r--r--. 1 root root  750 Feb 17 10:07 remi-safe.repo
-rw-r--r--. 1 root root 1314 Feb 17 10:07 remi-php74.repo
-rw-r--r--. 1 root root 1314 Feb 17 10:07 remi-php73.repo
-rw-r--r--. 1 root root 1314 Feb 17 10:07 remi-php72.repo
-rw-r--r--. 1 root root 1314 Feb 17 10:07 remi-php71.repo
-rw-r--r--. 1 root root  456 Feb 17 10:07 remi-php54.repo
-rw-r--r--. 1 root root  855 Feb 17 10:07 remi-modular.repo
-rw-r--r--. 1 root root  446 Feb 17 10:07 remi-glpi94.repo
-rw-r--r--. 1 root root  446 Feb 17 10:07 remi-glpi93.repo
-rw-r--r--. 1 root root  446 Feb 17 10:07 remi-glpi92.repo
-rw-r--r--. 1 root root  446 Feb 17 10:07 remi-glpi91.repo
-rw-r--r--. 1 root root 1314 Mar 11 04:51 remi-php70.repo
-rw-r--r--. 1 root root  141 Mar 11 08:30 MariaDB.repo
```

Figure 4-156 List repository contents on iTop

3. Copy the required repository from the system iTop to the `vsi-servicemanagement` system, as shown in Figure 4-157.

```
[root@iTop yum.repos.d]# scp -rp * remi* 169.38.67.66:/etc/yum.repos.d
The authenticity of host '169.38.67.66 (169.38.67.66)' can't be established.
ECDSA key fingerprint is SHA256:h826+sQe3667JwbryvgYlgENoNL04vgdeENvmQtw+Vs.
ECDSA key fingerprint is MD5:ad:d6:2a:d8:2a:b6:e4:14:de:3b:af:80:36:d7:e4:dc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '169.38.67.66' (ECDSA) to the list of known hosts.
root@169.38.67.66's password:
CentOS-Base.repo                                                                                    100% 1708     1.7MB/s   00:00
CentOS-Base.repo.orig                                                                               100% 1618    61.5KB/s   00:00
CentOS-CR.repo                                                                                      100% 1309     1.7MB/s   00:00
CentOS-Debuginfo.repo                                                                               100%  649   813.9KB/s   00:00
CentOS-fasttrack.repo                                                                               100%  314   393.7KB/s   00:00
CentOS-Media.repo                                                                                   100%  630   947.7KB/s   00:00
CentOS-Sources.repo                                                                                 100% 1331     1.7MB/s   00:00
CentOS-Vault.repo                                                                                   100% 6639     8.7MB/s   00:00
epel.repo                                                                                           100% 1050     1.2MB/s   00:00
epel-testing.repo                                                                                   100% 1149     1.3MB/s   00:00
MariaDB.repo                                                                                        100%  141   165.4KB/s   00:00
mysql-community.repo                                                                                100% 1209     1.6MB/s   00:00
mysql-community-source.repo                                                                         100% 1060     1.5MB/s   00:00
remi-glpi91.repo                                                                                    100%  446   658.4KB/s   00:00
remi-glpi92.repo                                                                                    100%  446   626.5KB/s   00:00
remi-glpi93.repo                                                                                    100%  446   719.5KB/s   00:00
remi-glpi94.repo                                                                                    100%  446   733.8KB/s   00:00
remi-modular.repo                                                                                   100%  855     1.5MB/s   00:00
remi-php54.repo                                                                                     100%  456   797.0KB/s   00:00
remi-php70.repo                                                                                     100% 1314     1.6MB/s   00:00
remi-php71.repo                                                                                     100% 1314     1.8MB/s   00:00
remi-php72.repo                                                                                     100% 1314     1.8MB/s   00:00
remi-php73.repo                                                                                     100% 1314     1.9MB/s   00:00
remi-php74.repo                                                                                     100% 1314     1.9MB/s   00:00
remi.repo                                                                                           100% 2605     3.8MB/s   00:00
remi-safe.repo                                                                                      100%  750     1.2MB/s   00:00
webtatic-archive.repo                                                                               100%  947     1.3MB/s   00:00
remi-glpi91.repo                                                                                    100%  446   672.4KB/s   00:00
remi-glpi92.repo                                                                                    100%  446   632.6KB/s   00:00
remi-glpi93.repo                                                                                    100%  446   867.6KB/s   00:00
remi-glpi94.repo                                                                                    100%  446   685.8KB/s   00:00
remi-modular.repo                                                                                   100%  855     1.4MB/s   00:00
remi-php54.repo                                                                                     100%  456   767.7KB/s   00:00
remi-php70.repo                                                                                     100% 1314     2.3MB/s   00:00
```

Figure 4-157 Copy the repository

4. After copying the repository, rerun the package installation command, as shown in Figure 4-158.

```
[root@vsi-servicemanagement yum.repos.d]# yum install python2-pip
Failed to set locale, defaulting to C
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * epel: mirrors.aliyun.com
 * remi-php70: ftp.riken.jp
 * remi-safe: ftp.riken.jp
Resolving Dependencies
--> Running transaction check
---> Package python2-pip.noarch 0:8.1.2-12.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================================================================
 Package                          Arch                          Version                               Repository                      Size
================================================================================================================================================
Installing:
 python2-pip                      noarch                        8.1.2-12.el7                          epel                           1.7 M

Transaction Summary
================================================================================================================================================
Install  1 Package

Total download size: 1.7 M
Installed size: 7.2 M
Is this ok [y/d/N]: y
Downloading packages:
python2-pip-8.1.2-12.el7.noarch.rpm                                                                 | 1.7 MB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : python2-pip-8.1.2-12.el7.noarch                                                                   1/1
```

Figure 4-158 Installing the Python software packages

The required Python module and package are installed in the `vsi-servicemanagement` system, as shown in Figure 4-159.

```
[root@vsi-servicemanagement yum.repos.d]# date
Mon Mar 16 07:47:05 CDT 2020
[root@vsi-servicemanagement yum.repos.d]# yum list python2-pip
Failed to set locale, defaulting to C
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * epel: mirrors.piconets.webwerks.in
 * remi-php70: ftp.riken.jp
 * remi-safe: ftp.riken.jp
Installed Packages
python2-pip.noarch                                              8.1.2-12.el7                                            @epel
[root@vsi-servicemanagement yum.repos.d]# packet_write_wait: Connection to 169.38.67.66 port 22: Broken pipe
Baldeeps-MacBook-Air:~ baldeepsingh$
```

Figure 4-159 Python installation completed

4.5 Log monitoring and analysis

This section shows the installation and configuration of the tools to be used for log monitoring and analysis.

4.5.1 Elasticsearch installation and configuration

This section describes the installation and configuration of Elasticsearch.

Node and server preparation

Complete the following steps to prepare the node and server:

1. Verify the current file system on the server, as shown in Figure 4-160.

```
[root@virtualserver01 ]# df -h
Filesystem                Size  Used Avail Use% Mounted on
devtmpfs                   16G     0   16G   0% /dev
tmpfs                      16G     0   16G   0% /dev/shm
tmpfs                      16G   57M   16G   1% /run
tmpfs                      16G     0   16G   0% /sys/fs/cgroup
/dev/xvda2                 24G  4.3G   19G  20% /
/dev/xvda1                976M  129M  797M  14% /boot
tmpfs                     3.2G     0  3.2G   0% /run/user/0
```

Figure 4-160 Check the server's file system

2. Create a file system that is separate from the operating system's file system. To do this, verify whether a disk is available by running the command that is shown in Figure 4-161.

```
[root@virtualserver01 ]# fdisk -l|grep 'Disk /dev/xv'
Disk /dev/xvdb: 2147 MB, 2147483648 bytes, 4194304 sectors
Disk /dev/xvda: 26.8 GB, 26843545600 bytes, 52428800 sectors
Disk /dev/xvdc: 107.4 GB, 107374182400 bytes, 209715200 sectors
```

Figure 4-161 Checking for an available disk

Figure 4-161 shows a disk is available for use: `Disk /dev/xvdc`.

3. To initialize the physical volume (PV) for later use by the Logical Volume Manager (LVM), run the following command:

   ```
   pvcreate /dev/xvdc
   ```

4. Construct an XFS file system by running the following command:

   ```
   mkfs.xfs /dev/xvdc
   ```

5. Create a volume group (VG) on the block devices by running the following command. A new VG that is named apps is created on `/dev/xvdc`:

   ```
   vgcreate apps /dev/xvdc
   ```

6. Create a logical volume in a VG `apps_lv` with disk space that us allocated by running the following command:

   ```
   lvcreate -n apps_lv -L 99.9G apps
   ```

7. Construct an XFS file system by running the command as follows:

   ```
   mkfs.xfs /dev/apps/apps_lv
   ```

8. Run the following command to get UUID for `/dev/apps/apps_lv`:

   ```
   blkid /dev/apps/apps_lv
   ```

9. The `/etc/fstab` system configuration file contains all available disks, disk partitions, and their options. Add the following line to get the `/<dir>/elk` file system mounted:

   ```
   UUID=8ae781ac-d6b2-41de-b5ab-99feb420e675 /<dir>/elk xfs defaults,noatime 1 2
   ```

10. Run the **`mount`** command to mount the file system in sequence under the server main root (/) partition per partition required:

    ```
    mount -a
    ```

11. Verify that the file system is mounted with allocated file system space by running the display file system command, as shown in Figure 4-162.

```
[root@virtualserver01 home]# df -h
Filesystem                Size  Used Avail Use% Mounted on
devtmpfs                   16G     0   16G   0% /dev
tmpfs                      16G     0   16G   0% /dev/shm
tmpfs                      16G  1.6G   15G  10% /run
tmpfs                      16G     0   16G   0% /sys/fs/cgroup
/dev/xvda2                 24G  6.2G   17G  28% /
/dev/xvda1                976M  129M  797M  14% /boot
/dev/mapper/apps-apps_lv  100G  2.9G   97G   3% /uidai/elk
tmpfs                     3.2G     0  3.2G   0% /run/user/0
[root@virtualserver01 home]#
```

Figure 4-162 Check mounted file systems

Support matrix

For more information about the various possible compatibility matrixes (see Figure 4-163), see this web page.

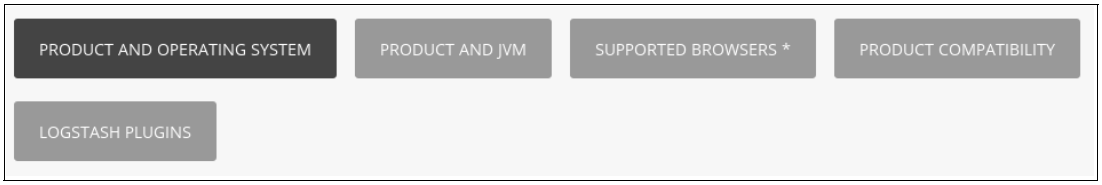


Figure 4-163 Matrix support

For example:

- Product compatibility: Elasticsearch versus Kibana versus Beats
- Product and operating system compatibility

Java (JVM) version

This section describes the Java version requirements:

- Elasticsearch is built by using Java and requires at least Java 8 to run. Only Oracle's Java and the OpenJDK are supported.
- The same JVM version must be used on all Elasticsearch nodes and clients.

- Recommend installing Java version 1.8.0_131 or later. Elasticsearch refuses to start if the wrong Java version is used.
- The version of Java that Elasticsearch uses can be configured by setting the JAVA_HOME environment variable.
- Open JDK rpm can be downloaded from this web page or follow the instructions at this web page to install Java according to the operating system that is used.
- After Java is installed, verify the Java version by running the following command:

```
[root@localhost ~]# java -version
openjdk version "1.8.0_31"
OpenJDK Runtime Environment (build 1.8.0_31-b13)
OpenJDK 64-Bit Server VM (build 25.31-b07, mixed mode)
```

User ID details

The following user ID were created while rpm was installing ELK:

- `logstash:x:485:480:logstash:/opt/logstash:/sbin/nologin`
- `elasticsearch:x:496:491:elasticsearch user:/home/elasticsearch:/sbin/nologin`
- `kibana:x:495:490::/home/kibana:/bin/bash`

RPM installation

Complete the following steps to install Elasticsearch:

1. Click **Install Elasticsearch RPM** at this web page.

   Note: The RPM package is suitable for installation on Red Hat, Centos, SUSE Linux Enterprise Server, openSUSE, and other RPM-based systems. RPMs can be downloaded from the Elasticsearch website or from the Elasticsearch RPM repository.

   The latest version of Elasticsearch can be found in the Download Elasticsearch web page.

   Previous versions can be found in the Past Releases web page.
2. Select the suitable version, click **Download** to download the rpm package to the local machine, and then, SFTP it to the servers.
3. Run the following command across all the nodes and complete the installation:

   ```
   rpm -ivh elasticsearch-7.6.0-x86_64.rpm
   ```

If Elasticsearch is not starting on installation, run the following statements to configure the Elasticsearch service to start automatically by using systemd:

```
sudo systemctl daemon-reload
sudo systemctl enable elasticsearch.service
```

You can start the Elasticsearch service by running the following command:

```
sudo systemctl start elasticsearch.service
```

For more information about RPM installation, see this web page.

At this web page, more information about the initial configuration of Elasticsearch, running Elasticsearch in your system, and checking whether Elasticsearch is running in your system, also is available.

Running Elasticsearch in your system

To configure Elasticsearch to start automatically when your system starts, run the following commands across all nodes:

```
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable elasticsearch.service
```

Elasticsearch can be started and stopped as follows:

```
sudo systemctl start elasticsearch.service
sudo systemctl stop elasticsearch.service
```

These commands provide no feedback as to whether Elasticsearch was started successfully. Instead, this information is written to the log files that are in `/var/log/elasticsearch/` (the `elasticsearch.log` is the log file with this information).

By default, the Elasticsearch service does not log information in the systemd journal. To enable **`journalctl`** logging, the **`--quiet`** option must be removed from the **`ExecStart`** command line in the `elasticsearch.service` file.

When systemd logging is enabled, the logging information is available by running the **`journalctl`** commands, as shown in the following examples:

- To tail the journal:

  ```
  sudo journalctl -f
  ```

- To list journal entries for the Elasticsearch service:

  ```
  sudo journalctl --unit elasticsearch
  ```

- To list journal entries for the Elasticsearch service starting from a specific time:

  ```
  sudo journalctl --unit elasticsearch --since  "2020-03-16 18:17:16"
  ```

You can use the **`man journalctl help`** command for more options or see this web page.

Directory layout of the data

Table 4-1 lists the location structure for the data.

Table 4-1 Data location structures

Type	Description	Default location	Setting
home	Elasticshare home directory or $ES_HOME.	`/usr/share/elasticsearch`	
bin	Binary script,s including Elasticsearch to start a node and elasticsearch-plugin to install plug-ins.	`/usr/share/elasticshare/bin`	
conf	Configuration files, including elasticsearch.yml.	`/etc/elasticshared`	path.conf
conf	Environment variables including heap size and file descriptors.	`/etc/config/elasticsearch`	
data	The location of the data files of each index and shard allocated on the node. Can hold multiple locations.	`/var/lib/elasticsearch`	path.data
logs	Log files location.	`/var/log/elasticsearch`	path.logs
plugins	Plugin files location. Each plug-in is contained in a subdirectory.	`/usr/share/elasticshare/plugins`	

Type	Description	Default location	Setting
repo	Shared file systems repository locations. Can hold multiple locations. A file system repository can be placed in any subdirectory of any directory that is specified here.	Not configured	path.repo
scripts	Location of script files.	/etc/elasticsearch/scripts	path.scripts

Configuring Elasticsearch

Elasticsearch loads its configuration from the /etc/elasticsearch/elasticsearch.yml file by default. The format is available at this web page.

The configuration files contain settings that are node-specific (such as node.name and paths), or settings that a node requires to join a cluster, such as cluster.name and network.host.

Important Elasticsearch configurations

Check the configuration file details that are available in /etc/elasticsearch/:

- elasticsearch.yml for configuring Elasticsearch
- log4j2.properties for configuring Elasticsearch logging

The configuration file format is in YAML format. The following parameters are changed in the configuration files:

> **Note:** Check that all the nodes include similar changes to their configuration files. For more information, see this web page.

- cluster.name: elasticsearch: The cluster name that was decided.
- node.name: ${HOSTNAME}: The host name can be specified or a reference to the host name provided (by specifying the host name variable).
- path.data: /<dir>/elk/data: The path where Elasticsearch stores its data.
- path.logs: /var/log/elasticsearch/: The path where the logs are stored.
- bootstrap.memory_lock: true: To avoid memory issues, when this parameter is set to true, the MAX_LOCKED_MEMORY parameter in /etc/sysconfig/elasticsearch and LimitMEMLOCK=infinity parameter in the elasticsearch.conf file must be uncommented or set if these do not have these values assigned to them (see Example 4-21). For more about this parameter setting, see “Important system configurations” on page 348.
- network.host: 0.0.0.0/169.38.122.221: For a single node.
- http.port: 9200: Default port that is used.

Example 4-21 elasticsearch.yml uncommented line

```
[root@virtualserver01 elasticsearch]#  grep "^[^#;]" elasticsearch.yml
node.name: virtualserver01.IBM-PoC-MSIP.cloud
path.data: /<dir>/elk/data
path.logs: /var/log/elasticsearch
network.host: 169.38.122.221
http.port: 9200
discovery.seed_hosts: ["169.38.122.221"]
cluster.initial_master_nodes: ["169.38.122.221"]
bootstrap.memory_lock: true
```

```
bootstrap.system_call_filter: false
```

The log configuration file `Log4j2.properties` includes all of the details that are set up correctly. Confirm that the following parameters are set:

- `appender.rolling.type = RollingFile`
- `appender.rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.logs.cluster_name}.log`
- `appender.rolling.filePattern = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.logs.cluster_name}-%d{yyyy-MM-dd}.log`
- `appender.rolling.policies.time.type = TimeBasedTriggeringPolicy`
- `appender.rolling.policies.time.interval = 1`
- `appender.rolling.policies.time.modulate = true`

Distributions that use systemd require that system resource limits be configured by way of systemd rather than the `/etc/sysconfig/elasticsearch` file. For more information about the configuration, see this web page.

Important system configurations

The following settings must be addressed before going to production:

- JVM heap size: The `jvm.options` file in the `etc/elasticsearch` path must be updated as follows:
  - Elasticsearch assigns the entire heap that is specified in `jvm.options` by way of the Xms (minimum heap size) and Xmx (maximum heap size) settings:
    - Set the minimum heap size (Xms) and maximum heap size (Xmx) to be equal to each other.
    - The more heap available to Elasticsearch, the more memory it can use for caching. However, too much heap can subject you to long garbage collection pauses.
    - Set Xmx to no more than 50% of your physical RAM to ensure that enough physical RAM exists for kernel file system caches.
  - Because the plan is to run multiple components, such as ELK stack on the same set of servers, it is better to use the default 2 GB values for these Xms and Xmx parameters.
- Systemd-related `.conf` file: For systems that use systemd, the parameter for the number of open files and the LimitMEMLOCK parameter are updated in the `elasticssearch.conf` file. It is created manually after the `elasticsearch.service.d` file is created under path `/etc/systemd/system/`. This `.conf` file is an updated version of the default `elasticsearch.service` file in the `/usr/lib/systemd/system/` path.
- Disable swapping: Because configuration changes were made, you should not see any swapping that occurs in the Linux boxes that are running Elasticsearch.
- File descriptors: Elasticsearch uses many file descriptors or file handles. Running out of file descriptors can be disastrous and most likely lead to data loss. This setting for a system with systemd option is done in the `.conf` file that is in `/etc/systemd/system/elasticsearch.service.d`. The parameter is `LimitNOFILE=65536`.
- Virtual memory: Elasticsearch uses a hybrid `mmapfs/niofs` directory by default to store its indexes. The default operating system limits on map counts is likely to be too low, which can result in out of memory exceptions. To set this value permanently, update and add the `vm.max_map_count` (`vm.max_map_count=262144`) setting in `/etc/sysctl.conf`.

- Number of threads: Elasticsearch uses several thread pools for different types of operations. It is important that it can create threads whenever needed. Make sure that the number of threads that the Elasticsearch user can create is at least 2048. This setting is managed in the `.conf` file that is created in the systemd-related `.conf` file. You also can make sure that the `/etc/security/limits.conf` file is updated with the following lines:

```
# allow user 'elasticsearch' mlockall
elasticsearch soft memlock unlimited
elasticsearch hard memlock unlimited
```

After all of these settings are completed, it is best to restart the Elasticsearch service. It must up before moving on to the next step.

4.5.2 Kibana

Complete the following steps to perform an RPM installation of Kibana on one of the Elasticsearch server nodes:

1. Download the Kibana software from this web page.
2. Select the suitable version per the compatibility matrix.

For more information about setting up Kibana, see this web page.

Installing Kibana

Complete the following steps:

1. Download Kibana V5.5.0 from this web page.
2. Move the `kibana-5.5.0-x86_64.rpm` package file to the designated server.
3. As the root user, run the **`rpm -ivh kibana-7.6.0-x86_64.rpm`** command. After the installation is complete, see “Running Kibana in the system”.

Running Kibana in the system

To configure Kibana to start automatically when the system starts, run the following commands:

```
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable kibana.service
```

Kibana can be started and stopped by running the following commands:

```
sudo systemctl start kibana.service
sudo systemctl stop kibana.service
```

These commands provide no feedback as to whether Kibana was started successfully. Instead, this information is written to the log files in `/var/log/kibana/`.

Directory layout for Kibana

The RPM places configuration files, logs, and the data directory in the suitable locations for an RPM-based system, as listed in Table 4-2.

Table 4-2 Data directory layout for Kibana

Type	Description	Default location
home	Kibana home directory or $KIBANA_HOME.	`/usr/share/kibana`
bin	Binary scripts including Kibana to start the Kibana server and kibana-plugin to install plug-ins.	`/usr/share/kibana/bin`
conf	Configuration files including `kibana.yml`.	`/etc/kibana`
data	The location of the data files that are written to disk by Kibana and its plug-ins.	`/var/lib/kibana`
optimize	Transpile source code. Specific administrative actions (for example, installing plug-in) result in the source code being retranspile dynamically.	`/usr/share/kibana/optimize`
plugins	Plugin files location. Each plugin is contained in a subdirectory.	`/usr/share/kibana/plugins`

Configuring Kibana by way of the configuration file

The Kibana server reads properties from the `kibana.yml` file on start. The default settings configure Kibana to run on localhost:5601. To change the host, port number, or to connect to Elasticsearch running on a different machine, you must update your `kibana.yml` file (see Example 4-22).

Open the `/etc/kibana/kibana.yml` file in vi edit mode and update the following parameters. Uncomment the following parameters within the `.yml` file:

- `server.port: 5601`: Check that this port is open.
- `server.host: "169.38.122.221"`: The server name to which the remote connections connect.
- `server.name: "virtualserver01"`: A more readable name of the Kibana server.
- `elasticsearch.url: "http://169.38.122.221:9200"`: One of the Elasticsearch server details.
- `elasticsearch.requestTimeout: 30000`: Time in milliseconds that Kibana waits for response from Elasticsearch.

Example 4-22 kibana.yml uncommented lines

```
[root@virtualserver01 kibana]#  grep "^[^#;]" kibana.yml
server.port: 5601
server.host: "169.38.122.221"
server.name: "virtualserver01"
elasticsearch.hosts: ["http://169.38.122.221:9200"]
elasticsearch.requestTimeout: 30000
logging.dest: /var/log/kibana/kibana.log
```

Configuring Kibana with Elasticsearch

Before you can start using Kibana, you must configure which Elasticsearch indexes to explore. The first time you access Kibana, you are prompted to define an index pattern that matches the name of one or more of your indexes; that is, all you need to configure to start using Kibana. For more information, see this web page.

Kibana can be started and stopped by running the following commands (logged as root user):

```
systemctl start kibana.service
systemctl status kibana.service
journalctl -u kibana.service → This is the option to check for any error log messages
systemctl stop kibana.service
```

Accessing Kibana

Kibana is a web application that you access through port 5601. By using your web browser, browse to the machine where Kibana is running and specify the port number.

You can reach the Kibana server's status page by browsing to `localhost:5601/status`. The status page displays information about the server's resource usage and lists the installed plug-ins.

4.5.3 Logstash

This section describes how to install Logstash in only one of the nodes. The logs are ingested into the setup from this node.

Begin by downloading the suitable version of Logstash (per the compatibility matrix), from this web page.

Move the RPM package `logstash-7.6.0.rpm` to the suitable server for installation. For example, under `/<dir>/elk/rpms` within the server, as performed with the previous software.

Installing Logstash

This section describes how to install RPM for Logstash. As a root user, run the command from the directory where the Logstash rpm package is located:

```
rpm -ivh  logstash-7.6.0.rpm
```

Directory layout

After installing Logstash, the directory layout is as listed Table 4-3.

Table 4-3 Directory layout for Logstash

Type	Description	Default location	Setting
home	Home directory of the Logstash installation.	`/usr/share/logstash`	
bin	Binary scripts, including Logstash to start Logstash and logstash-plugin to install plug-ins.	`/usr/share/logstash/bin`	
settings	Configuration files, including logstash.yml, jvm options, and start options.	`/etc/logstash`	path.settings
conf	Logstash pipeline configuration files.	`/etc/logstash/configd`	path.config

Type	Description	Default location	Setting
logs	Log files.	`/var/log/logstash`	path.logs
plugins	Local, non-Ruby-Gem plug-in files. Each plug-in is contained in a subdirectory. Recommended for development only.	`/usr/share/logstash/plugins`	path.plugins

Logstash configuration files

Logstash has the following types of configuration files:

- Pipeline configuration files, which define the Logstash processing pipeline.
- Settings files, which specify options that control Logstash startup and execution.

Pipeline configuration files

You create pipeline configuration files when you define the stages of the Logstash processing pipeline. On the RPM, you place the pipeline configuration files in the `/etc/logstash/conf.d` directory. For more information about the configuration file, see this web page.

Setting files

The settings files are defined in the Logstash installation. Logstash includes the following settings files:

- `logstash.yml`: Contains Logstash configuration flags. You can set flags in this file instead of passing the flags at the command line. Any flags that you set at the command line override the corresponding settings in the logstash.yml file.
- `jvm.options`: Contains JVM configuration flags. Specify each flag on a separate line. You can also use this file to set the locale for Logstash.
- `startup.options` (Linux): Contains options used by the system-install script in `/usr/share/logstash/bin` to build the suitable startup script for the system.

4.5.4 Elasticsearch Filebeat integration

Filebeat is a lightweight shipper for forwarding and centralizing log data. Filebeat is installed as an agent on your servers. Filebeat monitors the log files or locations that you specify, collects log events, and forwards them to Elasticsearch or Logstash for indexing.

Download the suitable version of Filebeat (per the compatibility matrix), from this web page.

Move the downloaded RPM package `filebeat-7.6.0-x86_64.rpm` to the suitable server for installation. For example, under `/<dir>/elk/rpms` within the server.

Installing Filebeat

As root user, run the following command from the directory where filebeat rpm package is located:

```
rpm -ivh  filebeat-7.6.0-x86_64.rpm
```

Directory layout for Filebeat

Table 4-4 lists the directory layout after the installation of Filebeat.

Table 4-4 Filebeat directory layout

Type	Description	Default location	Configuration option
home	Home of the Filebeat installation.	N/A	path.home
bin	The location of the binary files.	{path.home}/bin	N/A
config	The location for configuration files.	{path.home}	path.config
data	The location for persistent data files.	{path.home}/data	path.data
logs	The location of the logs created by Filebeat.	{path.home}/logs	path.logs

Configuring Filebeat

To configure Filebeat, edit the configuration file. For rpm and deb, you find the configuration file at `/etc/filebeat/filebeat.yml`. A full example configuration file also is available at `/etc/filebeat/filebeat.reference.yml`.

Filebeat input selection

The main configuration unit in Filebeat is the inputs. They are responsible for locating specific files and applying basic processing to them.

The main configuration that is applied to inputs is path (or paths) to the file you want to track, but you can use other configuration options, such as defining the input type and encoding to use for reading the file, excluding and including specific lines, and adding custom fields.

Filebeat can be configured to track multiple files or define multiple inputs if you have input-specific configurations that you want to apply.

Filebeat support multiple modules. For more information, see this web page.

The following example filebeat input section monitors all of the files that end with `.log` in location `/var/log/`, and send the log messages to Elasticsearch or to Logstash, depending on the filebeat output section:

```
filebeat.inputs:
- type: log
  #Change value to true to activate the input configuration
  enabled: true
  paths:
    - "/var/log/*.log"
```

Filebeat output section

Filebeat configuration files define to where you want to ship the data. Many supported output options are available, including console, file, cloud, Redis, and Kafka. However, in most cases, you use the Logstash or Elasticsearch output types. Define a Logstash instance for more advanced processing and data enhancement.

For forwarding logs to Elasticsearch:

```
output.elasticsearch:
  hosts: ["localhost:9200"]
```

For forwarding logs to Logstash:

```
output.logstash:
  hosts: ["localhost:5044"]
```

For more information about multiline logs that are related to configurations, see this web page.

4.5.5 Elasticsearch Metricbeat installation

This section describes how to install elastic Metricbeats in only one node. Metricbeat collects the metrics from the Prometheus instance remotely.

Download the suitable version of Logstash (per the compatibility matrix), from this web page.

Move the RPM package *m*`etricbeat-7.6.0-x86_64.rpm` to the suitable server for installation. For example, under `/<dir>/elk/rpms` within the server.

Installing Metricbeat

This section describes how to perform the RPM install of Metricbeat. As root user, run the following command from the directory where metricbeat rpm package is located:

```
rpm -ivh  metricbeat-7.6.0-x86_64.rpm
```

Directory layout for Metricbeat

Table 4-5 lists the directory layout after the installation of Metricbeat.

Table 4-5 Metricbeat directory layout

Type	Description	Location
home	Home of the metric installation.	/usr/share/metricbeat
bin	The location of the binary files.	/usr/share/metricbeat/bin
conf	The location for configuration files.	/etc/metricbeat
data	The location for persistent data files.	/var/lib/metricbeat
logs	The location for the logs created by metricbeat	/var/log/metricbeat

Metricbeat module

Metricbeat is a lightweight shipper that can be installed on a specific server to periodically collect metrics from the operating system and services that are running on the server. Metricbeat takes the metrics and statistics that it collects and ships them to the output that you specify, such as Elasticsearch or Logstash.

Metricbeat start and stop command are as follows:

```
service metricbeat start
service metricbeat stop
```

The unit file that is included in the packages sets the **-e** flag by default. This flag makes Metricbeat log to stderr and disables other log outputs. Systemd stores all output sent to stderr in journald.

To review the Metricbeat logs, ruin the following command:

```
journalctl -u metricbeat.service
```

To enable default logging to /var/log/metricbeat, the metricbeat.service file under /lib/systemd/system must be modified to enable Metricbeat logging to default location /var/log/metricbeat by commenting the following line:

```
#Environment="BEAT_LOG_OPTS=-e"
```

Metricbeat helps to monitor servers by collecting metrics from the system and services running on the server. Multiple modules are available for monitoring,

Metricbeat system module

The system module enables monitoring servers. The following default metricsets are available:

- cpu
- load
- memory
- network
- process
- process_summary

In the /etc/metricbeat/moduls.d folder, enable the Metricbeat system module by renaming the module from system.yml.disable to system.yml.

Consider the following points:

- Metricbeat must be installed on each server from which metricdata must be collected.
- The following important settings must be updated in the metricbeat.yml file:

```
#Modules Section
metricbeat.config.modules:
  # Glob pattern for configuration loading
  path: ${path.config}/modules.d/*.yml

# Dashboard Section
setup.dashboards.enabled: true
#KIBANA Section
setup.kibana:
  host: "169.38.131.87:5601"
#Output Section as below to send data directly to elasticsearch.
output.elasticsearch:
  hosts: ["169.38.131.87:9200"]
```

- Metricbeat includes a default dashboard for each module. Default dashboards can be enabled by using the metricbeat.yml file in the /etc/metricbeat/metricbeat.yml directory.

- After the data is loaded into Elasticsearch and the default dashboard is loaded into Kibana, its sample dashboard view for the host overview and system dashboard are available, as shown in Figure 4-164 and Figure 4-165.

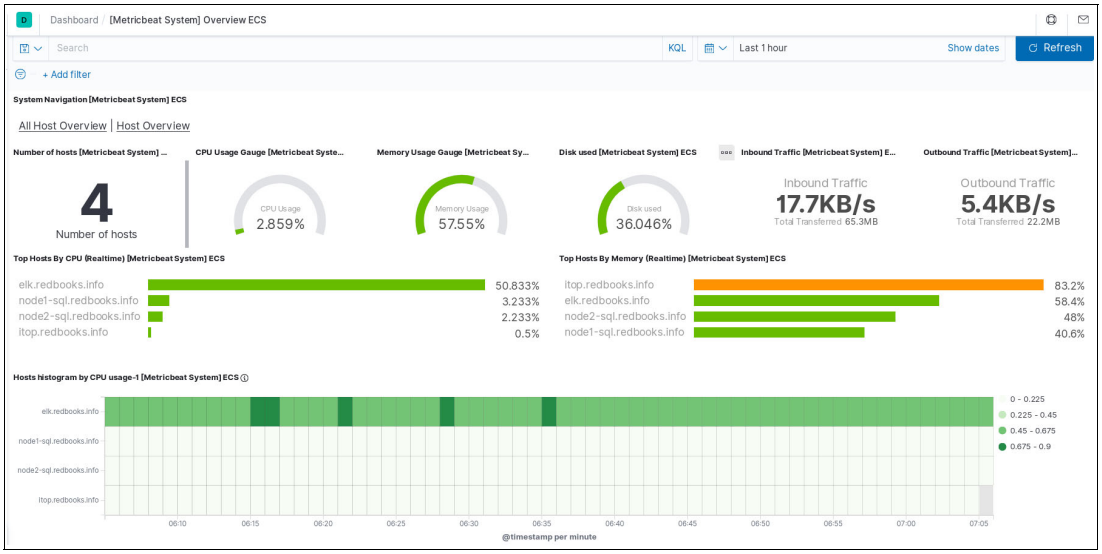


Figure 4-164 All hosts overview dashboard

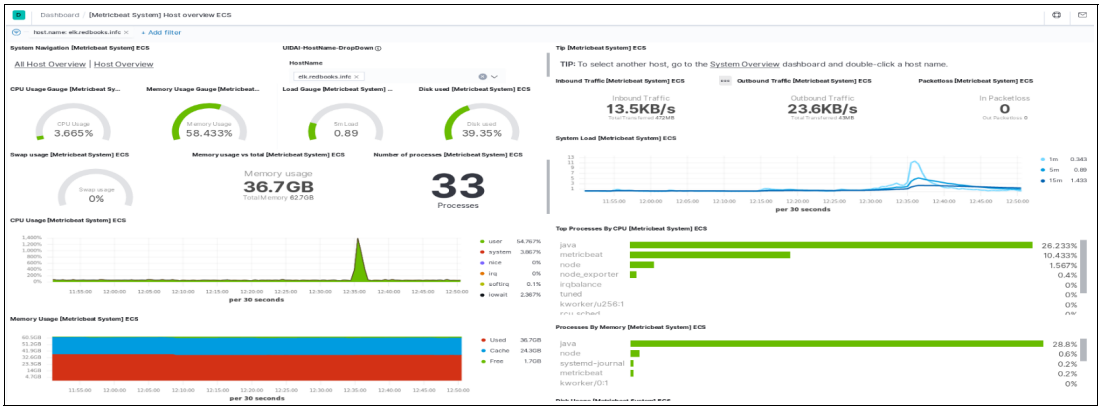


Figure 4-165 Host overview dashboard

Metricbeat Prometheus module

The Prometheus collector metricset scrapes data from Prometheus exporters. To scrape metrics from a Prometheus exporter, configure the hosts field to it. The path to retrieve the metrics from (`/metrics` by default) can be configured with `metrics_path`. This module can scrape all metrics that are stored in a Prometheus server by using the federation API and pointing this configuration to the Prometheus server.

Example 4-23 shows a `prometheus.yml` file that includes modules metricset and federation API.

Example 4-23 Sample prometheus.yml

```
# Module: prometheus
# Docs:
https://www.elastic.co/guide/en/beats/metricbeat/7.6/metricbeat-module-prometheus.
html
- module: prometheus
```

```
  period: 10s
  hosts: ["169.38.131.88:9090"]
  metrics_path: /metrics
  #username: "user"
  #password: "secret"
- module: prometheus
  metricsets: ["collector"]
  enabled: true
  period: 10s
  hosts: ["169.38.131.88:9090"]
  metrics_path: /metrics
- module: prometheus
  period: 10s
  hosts: ["169.38.131.88:9090"]
  metrics_path: '/federate'
  query:
    'match[]': '{__name__!=""}'
```

After data is loaded into Elasticsearch and the default dashboard is loaded into Kibana, the view in Kibana and the sample dashboard for Prometheus is displayed, as shown in Figure 4-166.

Figure 4-166 Metricboard Prometheus overview dashboard

Metricbeat for MYSQL module

This module periodically fetches metrics from MySQL servers. The default metricset is status. The MySQL module supports the standard configuration options.

The following example is a sample `mysql.yml` file:

```
- module: mysql
  metricsets:
    - status
  #  - galera_status
  period: 10s
  #hosts: ["root:secret@tcp(127.0.0.1:3306)/"]
  hosts: ["prom_exporter:prom_exporter@tcp(169.38.131.90:3306)/"]
```

After the data is loaded into Elasticsearch and default dashboard is loaded into Kibana, the view in Kibana and the sample dashboard for Host overview and system dashboard is displayed, as shown in Figure 4-167.

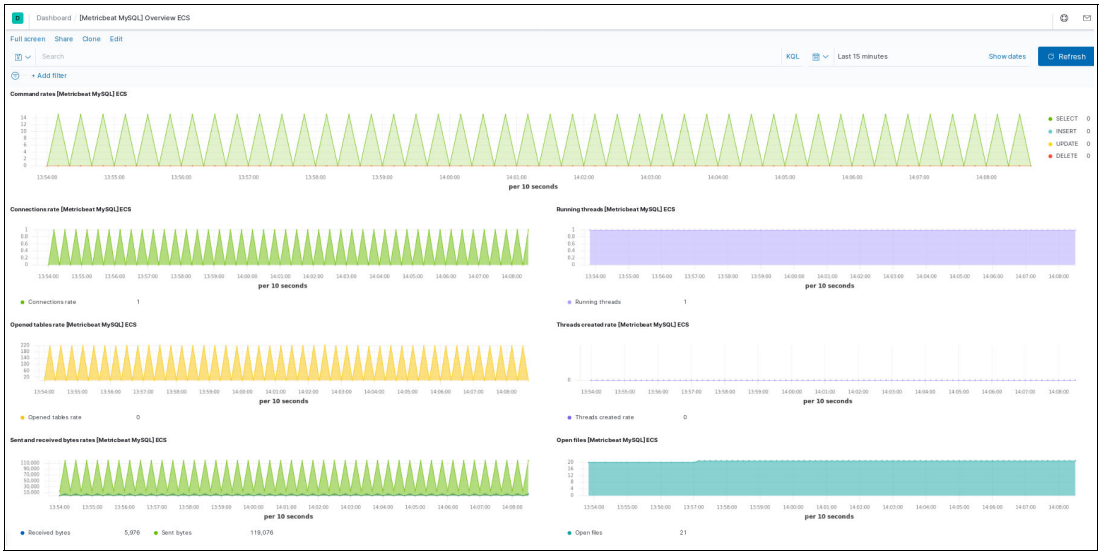


Figure 4-167 Metricbeat MYSQL dashboard overview

4.6 Unified dashboards and portals with Liferay

This section describes how to integrate the components into the Liferay portal.

4.6.1 Configuring and integrating all Cloud components and tools into the portal

Complete the following steps:

1. Get the portal URL after creating the portal application by running the following command:

   ```
   $oc get route
   ```

2. Browse to the URL and then, click **Sign In**. Click the menu in the left top, as shown in Figure 4-168. Then, click New (**+**) on the right side of the menu.

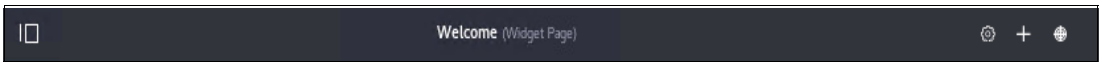


Figure 4-168 Welcome page

3. To add to the portal page, click **Build** → **Pages**, as shown in Figure 4-169.

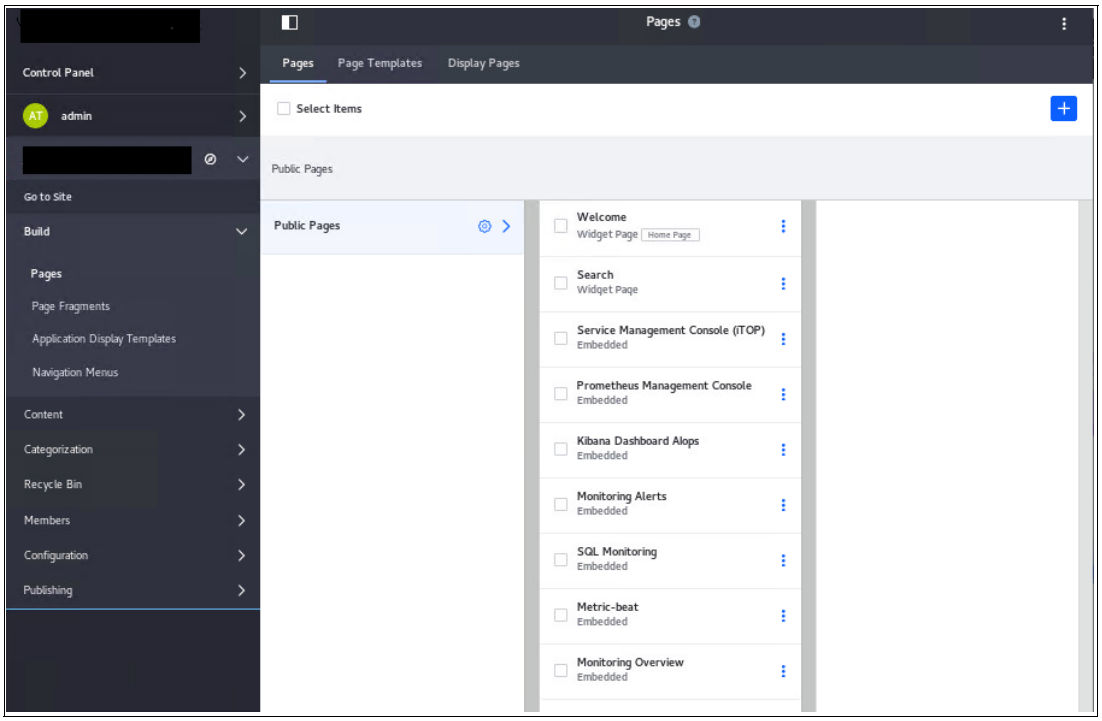


Figure 4-169 Control window to add pages to the portal

4. Click **Public page** → **Embedded** (see Figure 4-170).

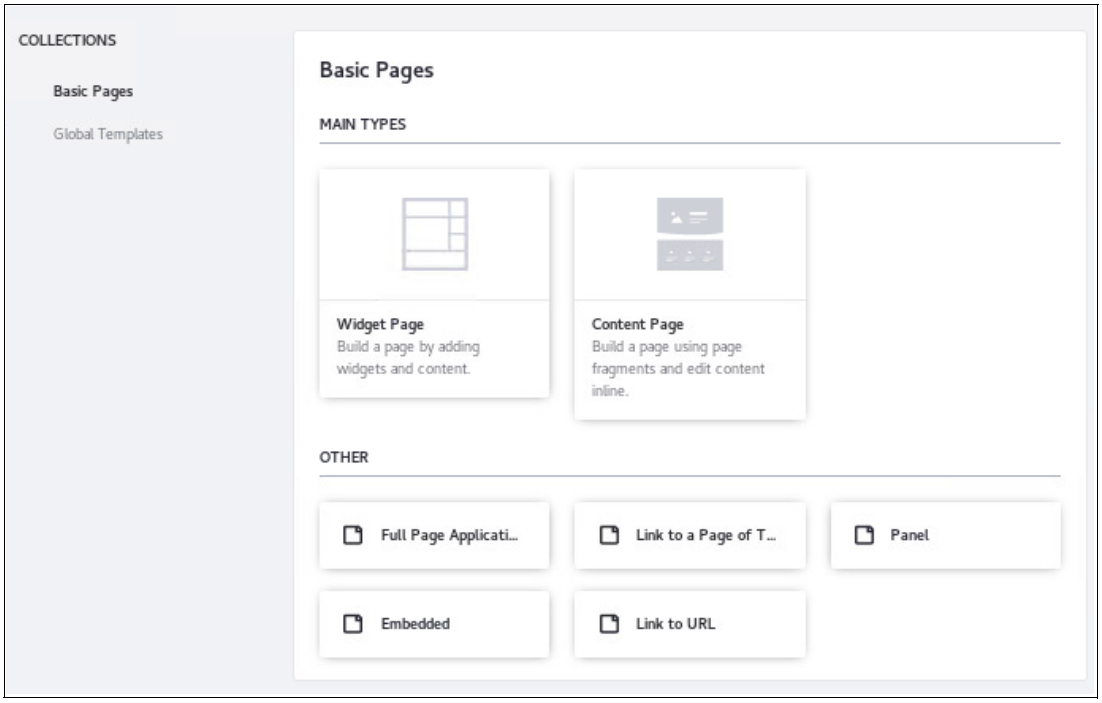


Figure 4-170 Collections pane: Adding to the portal

5. In the Add Page window, enter a name in the Name field (see Figure 4-171). Click **Save**.

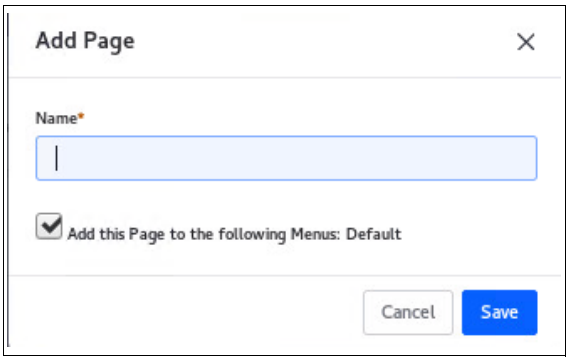


Figure 4-171 Add Page pane: Name

6. In the next window, add the URL, as shown in Figure 4-172. Click **Save**.

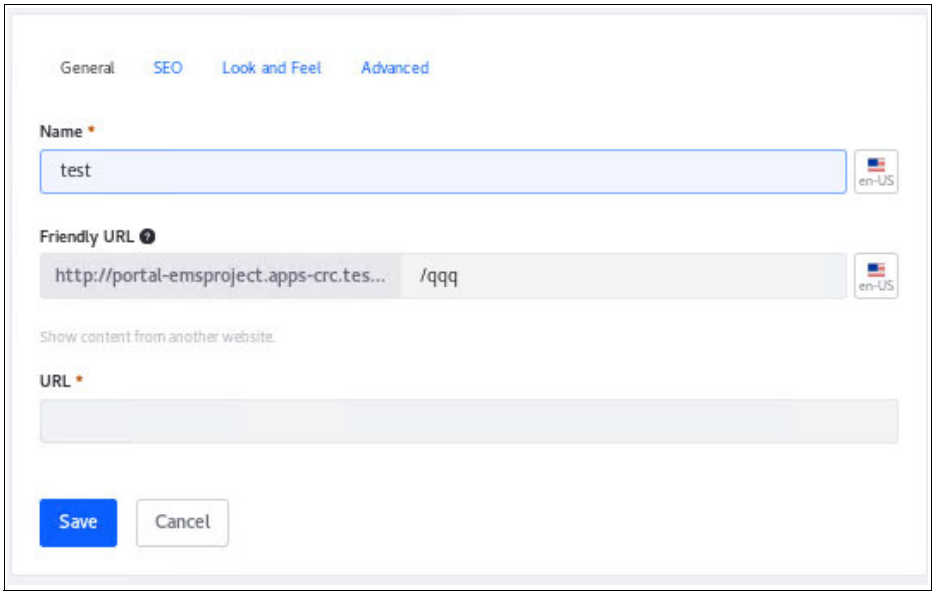


Figure 4-172 General pane: Add the URL

7. Review the page by clicking the three vertical dots. Then, click **Configure** (see Figure 4-173).

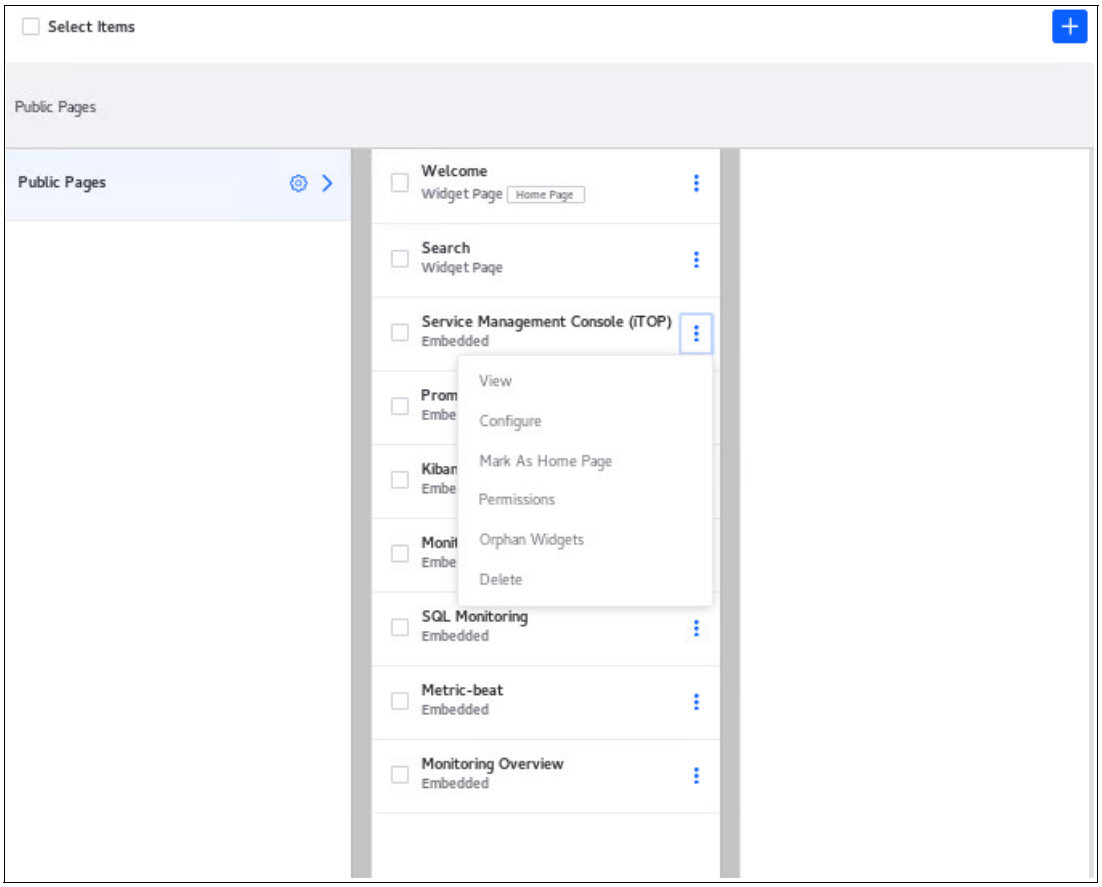


Figure 4-173 Public Pages pane: Selecting the items

Pages can be added in the Liferay portal by using the same steps.

5

Use cases

This chapter describes several use case scenarios that are deployed in the cluster.

This chapter includes the following topics:

5.1 Red Hat OpenStack tenancy and isolation architecture and Ceph Dashboard review

This use case shows steps to demonstrate Red Hat OpenStack tenancy isolation and architecture.

Figure 5-1 shows the list of bare metal servers part of the Red Hat OpenStack setup.

```
(undercloud) [stack@red-director ~]$ openstack baremetal node list
+--------------------------------------+-------------------------------+--------------------------------------+-------------+--------------------+-------------+
| UUID                                 | Name                          | Instance UUID                        | Power State | Provisioning State | Maintenance |
+--------------------------------------+-------------------------------+--------------------------------------+-------------+--------------------+-------------+
| 6fa72a36-0a0c-4249-8ac7-891603a6fa87 | red-compute1.redbooks.info    | 4588c7e6-1ff9-42fe-b9bd-5d32934da843 | power on    | active             | False       |
| 96c3c42e-16d1-49f6-bf2a-e05d56bd3f00 | blue-controller2.redbooks.info| 1829d594-22d2-42d4-b5de-3b8cba729e7e | power on    | active             | False       |
| e564423f-354b-4c8e-af4f-ae328a91b87f | red-compute7.redbooks.info    | 838b8fb9-5d5b-4894-a60f-2ce329d28baf | power on    | active             | False       |
| 8a945039-c44f-4e10-b759-485bc5499a40 | red-compute2.redbooks.com     | 9cc4f6fb-2ca3-479a-b882-ca1514be10e4 | power on    | active             | False       |
| 589df7ce-1b8b-4dc4-8d68-5e8c67e9b04b | red-compute5.redbooks.info    | 40eabce3-d933-41a8-845b-f49f8f2d2256 | power on    | active             | False       |
| a457f0d6-abe3-4ae5-8b19-8669c86bcb45 | red-compute6.redbooks.info    | 01cc1e55-9ae4-40c0-8b28-f22e794bd192 | power on    | active             | False       |
| 256d9a74-7267-43c9-b4a6-fc9f918fc751 | red-controller1.redbooks.info | 75036a84-9b1b-454b-9ff4-8a50dfd0242e | power on    | active             | False       |
| a4d1c346-136a-4f9b-9c6e-242623e4647a | red-computenode.redbooks.info | 3090420c-bd9c-4289-9afd-b37dc33f1b14 | power on    | active             | False       |
| c4e8df7d-5849-40cc-a7de-3ebe9d49165c | red-computenode2.redbooks.info| None                                 | None        | enroll             | False       |
| 25ba1c65-fe68-4f0f-b2b3-d472eab28922 | red-computenode3.redbooks.info| 8ce50693-e920-4778-a84f-67f5d1831722 | power on    | active             | False       |
| b9beb650-2757-4731-9e60-8ae777bc4c76 | red-computenode4.redbooks.info| ddf4ae17-608d-4b19-a005-0ea7ea233ca1 | power on    | active             | False       |
| 072129ed-ad85-4758-9619-fa16cfbfb468 | red-computenode5.redbooks.info| 5b17b841-e212-478c-b692-1f7cbe5ed64c | power on    | active             | False       |
| a6364a22-7ffe-4d24-a9d2-2c56445de23f | red-computenode6.redbooks.info| None                                 | None        | enroll             | False       |
| b9d0c1a1-80a3-4b44-981c-7dc3a728486e | red-computenode7.redbooks.info| None                                 | None        | enroll             | False       |
| 046151d9-7742-4cd8-ac46-d64574cfa0b0 | red-computenode8.redbooks.info| None                                 | None        | enroll             | False       |
+--------------------------------------+-------------------------------+--------------------------------------+-------------+--------------------+-------------+
```

Figure 5-1 Bare metal servers list

Figure 5-2 shows the list of services that are running on Red Hat OpenStack.

```
(overcloud) [stack@red-director ~]$ openstack service list
+----------------------------------+------------+----------------+
| ID                               | Name       | Type           |
+----------------------------------+------------+----------------+
| 08af1315329040fe8b47c64aa3a60174 | panko      | event          |
| 0dab695d3dae452393c5a499f6401c03 | glance     | image          |
| 1b84b7afc88d458a9752e559ad30bde3 | nova       | compute        |
| 1d12e392f4ec43c28b3e98589586b900 | placement  | placement      |
| 215942fa30db4281a142c0a17edc2c8e | keystone   | identity       |
| 4240523e45c8487ba40344def20e2bb7 | aodh       | alarming       |
| 502d4fc5f3b645de825cdca33b4a87e6 | neutron    | network        |
| 54dae50aaff743cd94b891dfc1041efb | heat-cfn   | cloudformation |
| 90b11b229e66412bb7948c148b6f61ea | cinderv3   | volume         |
| 9577820f96274c0d8656ffd52b088ba5 | ceilometer | metering       |
| 9713d201fef5425a8b073dea3b9d62c1 | swift      | object-store   |
| b7c5eacf8dd441cc8e6eb71acd042ff3 | cinderv2   | volumev2       |
| bb0d20e4f3014297927e2390ffef7055 | cinderv3   | volumev3       |
| e8ed34e035bb4c379cab9b722cb5fc2c | heat       | orchestration  |
| f8cbd3e3b14a431f81766385f613ec52 | gnocchi    | metric         |
+----------------------------------+------------+----------------+
```

Figure 5-2 Red Hat OpenStack services list

After you log in to the Red Hat OpenStack UI, you see a list of projects that are only Tenants, as shown in Figure 5-3.

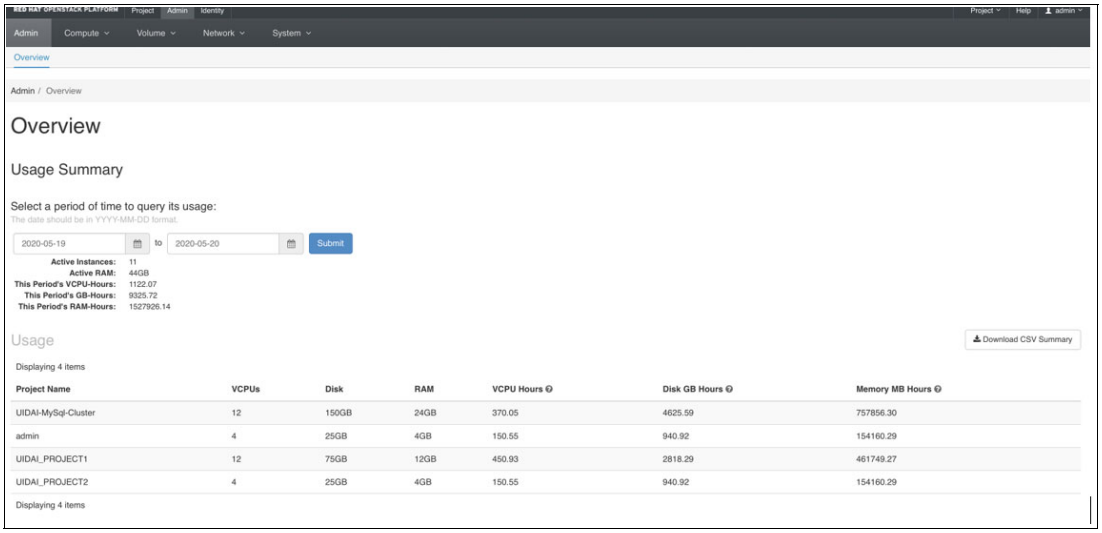


Figure 5-3 Red Hat OpenStack Platform: Overview window

In Red Hat OpenStack, you can create compute level isolation by creating host aggregates and present them as availability zones, as shown in Figure 5-4.

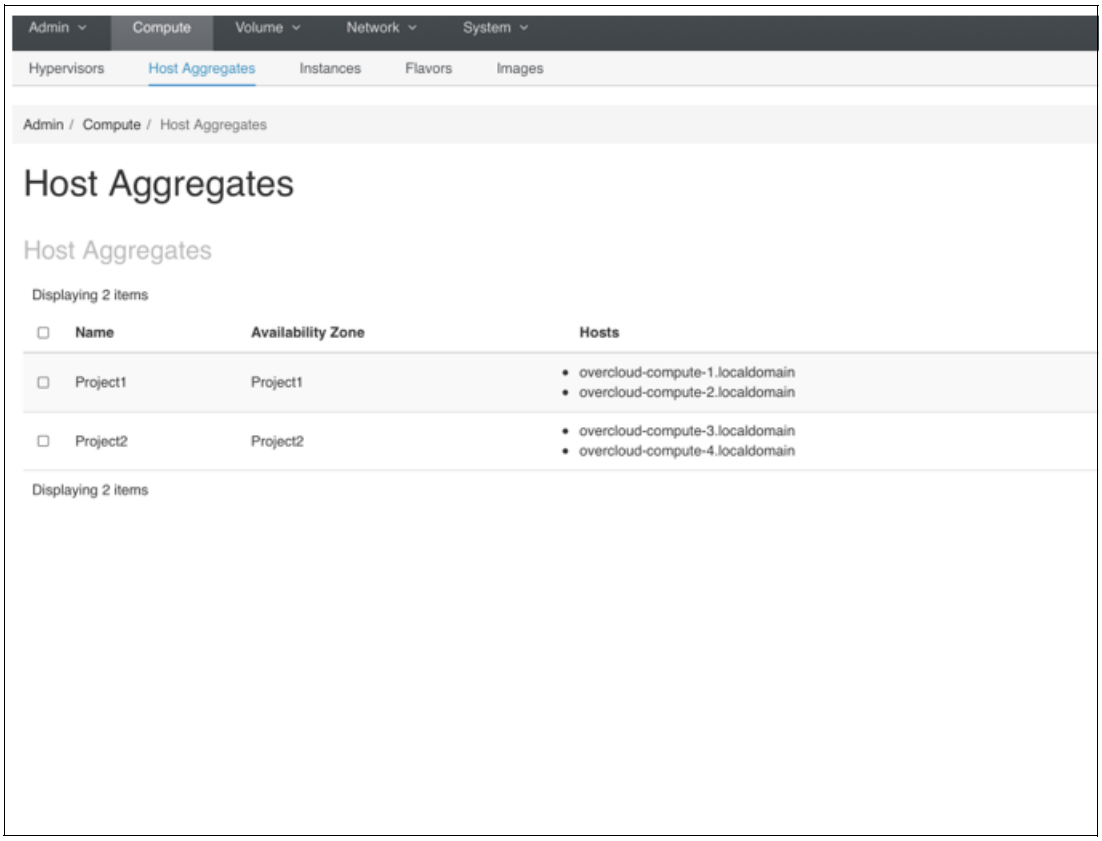


Figure 5-4 Red Hat OpenStack Platform: Computer: Host Aggregates window

You can create network level isolation by creating private networks, as shown in Figure 5-5.

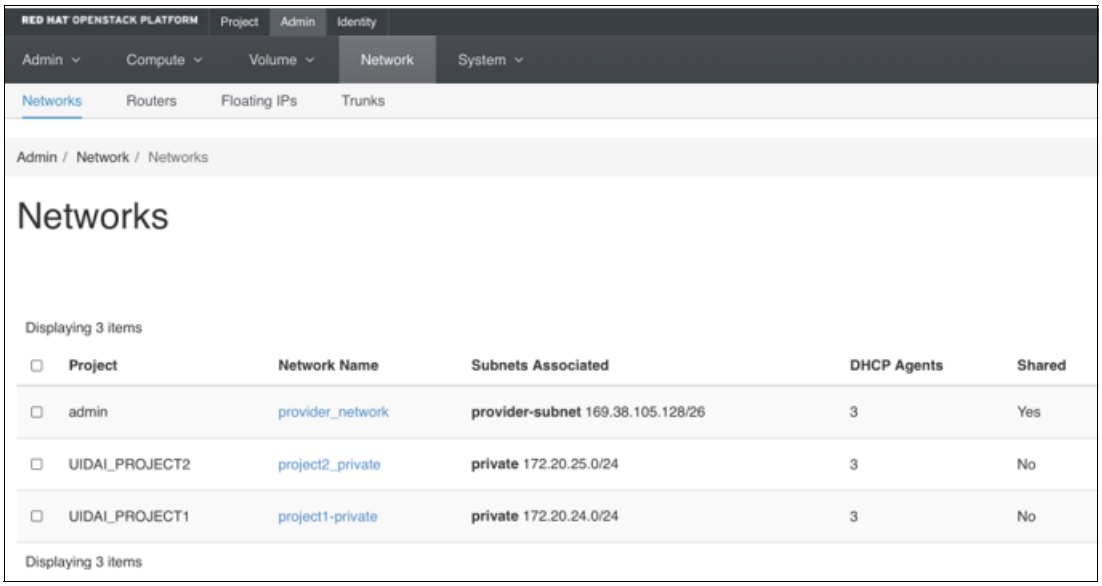


Figure 5-5 Red Hat OpenStack Platform: Networks window

Ceph is a software defined storage. The Ceph Dashboard is shown in Figure 5-6.

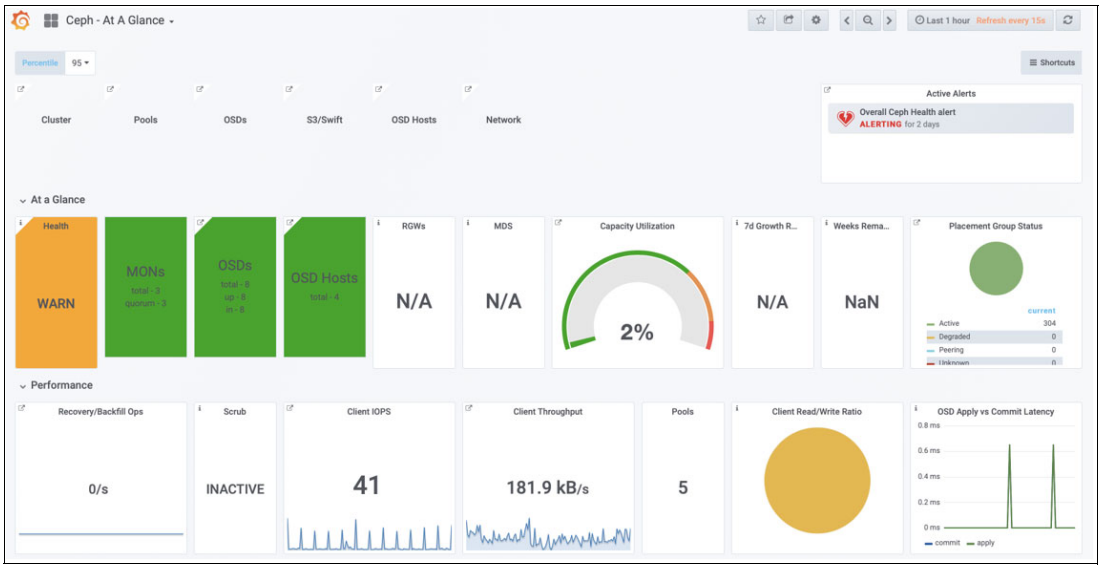


Figure 5-6 Ceph At a Glance dashboard

5.2 High availability in Red Hat OpenStack

In Red Hat, OpenStack V3 controllers are deployed to achieve high availability of various Red Hat OpenStack services.

For example, restart one of the controllers to test the high availability, as shown in Figure 5-7.

```
[root@overcloud-controller-1 ~]# pcs cluster status
Cluster Status:
 Stack: corosync
 Current DC: overcloud-controller-0 (version 1.1.21-4.el7-f14e36fd43) - partition with quorum
 Last updated: Wed May 20 13:33:02 2020
 Last change: Mon May 18 11:04:35 2020 by root via cibadmin on overcloud-controller-0
 12 nodes configured
 33 resources configured

PCSD Status:
  overcloud-controller-0: Online
  overcloud-controller-2: Online
  overcloud-controller-1: Online
[root@overcloud-controller-1 ~]# reboot
```

Figure 5-7 Rebooting one controller to test high availability

Even if one controller is down, you can still access the Red Hat OpenStack UI and access all the tenants, as shown in Figure 5-8.

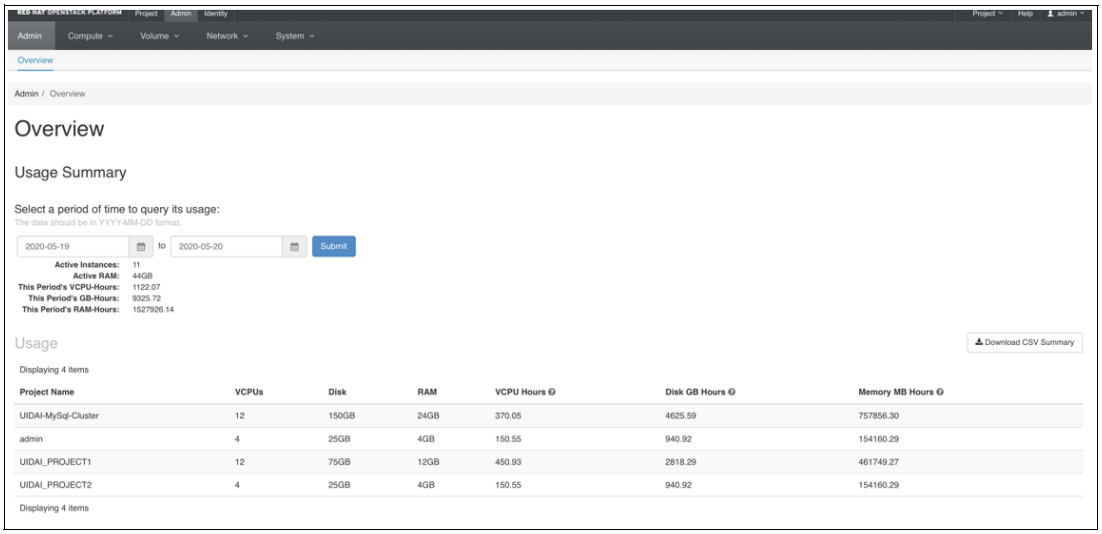


Figure 5-8 Red Hat OpenStack Platform: Overview summary window

5.3 Bare Metal provisioning using Ironic (manual and automated) in Red Hat OpenStack

Create and generate a JSON configuration file (as shown in Figure 5-9) with bare metal servers IPMI details, and server details, such as MAC addresses, CPU, memory, and disk.

```
{
    "nodes": [
        {
            "mac":[
                "0c:c4:7a:39:85:0c"
            ],
            "name":"red-compute3.redbooks.info",
            "cpu":"2",
            "memory":"65536",
            "disk":"1000",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"root",
            "pm_password":"h25NBPgpPf",
            "pm_addr":"10.162.62.9"
        }
    ]

}
```

Figure 5-9 JSON configuration file

Import the bare metal node to overcloud as shown in Figure 5-10.

```
(undercloud) [stack@red-director script]$ openstack overcloud node import test_introspection.json
Started Mistral Workflow tripleo.baremetal.v1.register_or_update. Execution ID: c1a2b25d-32c9-40a2-b791-21327ee7b4ab
Waiting for messages on queue 'tripleo' with no timeout.


1 node(s) successfully moved to the "manageable" state.
Successfully registered node UUID 84403a9a-0d02-48a6-9761-ee08a0bbbd38
(undercloud) [stack@red-director script]$
```

Figure 5-10 Importing the bare metal node

After the bare metal node is successfully imported, it shows as manageable in provisioning state, as shown in Figure 5-11.

```
(undercloud) [stack@red-director script]$ openstack baremetal node list
+--------------------------------------+-----------------------------+--------------------------------------+-------------+--------------------+-------------+
| UUID                                 | Name                        | Instance UUID                        | Power State | Provisioning State | Maintenance |
+--------------------------------------+-----------------------------+--------------------------------------+-------------+--------------------+-------------+
| 6fa72a36-0a0c-4249-8ac7-891603a6fa87 | red-compute1.redbooks.info  | 4588c7e6-1ff9-42fe-b9bd-5d32934da843 | power on    | active             | False       |
| 96c3c42e-16d1-49f6-bf2a-e05d56bd3f00 | blue-controller2.redbooks.info | 1829d594-22d2-42d4-b5de-3b8cba729e7e | power on    | active             | False       |
| e564423f-354b-4c8e-af4f-ae328a91b87f | red-compute7.redbooks.info  | 838b8fb9-5d5b-4894-a60f-2ce329d28baf | power on    | active             | False       |
| 8a945039-c44f-4e10-b759-485bc5499a40 | red-compute2.redbooks.com   | 9cc4f6fb-2ca3-479a-b882-ca1514be10e4 | power on    | active             | False       |
| 589df7ce-1b8b-4dc4-8d68-5e8c67e9b04b | red-compute5.redbooks.info  | 40eabce3-d933-41a8-845b-f49f8f2d2256 | power on    | active             | False       |
| a457f0d6-abe3-4ae5-8b19-8669c86bcb45 | red-compute6.redbooks.info  | 01cc1e55-9ae4-40c0-8b28-f22e794bd192 | power on    | active             | False       |
| 256d9a74-7267-43c9-b4a6-fc9f918fc751 | red-controller1.redbooks.info | 75036a84-9b1b-454b-9ff4-8a50dfd0242e | power on    | active             | False       |
| a4d1c346-136a-4f9b-9c6e-242623e4647a | red-computenode.redbooks.info | 3090420c-bd9c-4289-9afd-b37dc33f1b14 | power on    | active             | False       |
| c4e8df7d-5849-40cc-a7de-3ebe9d49165c | red-computenode2.redbooks.info | None                                 | None        | enroll             | False       |
| 25ba1c65-fe68-4f0f-b2b3-d472eab28922 | red-computenode3.redbooks.info | 8ce50693-e920-4778-a84f-67f5d1831722 | power on    | active             | False       |
| b9beb650-2757-4731-9e60-8ae777bc4c76 | red-computenode4.redbooks.info | ddf4ae17-608d-4b19-a005-0ea7ea233ca1 | power on    | active             | False       |
| 072129ed-ad85-4758-9619-fa16cfbfb468 | red-computenode5.redbooks.info | 5b17b841-e212-478c-b692-1f7cbe5ed64c | power on    | active             | False       |
| a6364a22-7ffe-4d24-a9d2-2c56445de23f | red-computenode6.redbooks.info | None                                 | None        | enroll             | False       |
| b9d0c1a1-80a3-4b44-981c-7dc3a728486e | red-computenode7.redbooks.info | None                                 | None        | enroll             | False       |
| 046151d9-7742-4cd8-ac46-d64574cfa0b0 | red-computenode8.redbooks.info | None                                 | None        | enroll             | False       |
| 84403a9a-0d02-48a6-9761-ee08a0bbbd38 | red-compute3.redbooks.info  | None                                 | power off   | manageable         | False       |
+--------------------------------------+-----------------------------+--------------------------------------+-------------+--------------------+-------------+
```

Figure 5-11 Bare metal node list after importing it

Introspection (bare metal operating system provisioning) takes 20 minutes. After successfully Introspected, the operating system is deployed to the bare metal server and changes provision state from manageable to available, as shown in Figure 5-12.

```
(undercloud) [stack@red-director script]$ openstack overcloud node introspect --all-manageable --provide
Waiting for introspection to finish...
Started Mistral Workflow tripleo.baremetal.v1.introspect_manageable_nodes. Execution ID: 801559cc-c048-481d-bfca-6225dd3bbd02
Waiting for messages on queue 'tripleo' with no timeout.
```

Figure 5-12 Red Hat OpenStack introspect provisioning

You can automate multiple bare metal server provisionings by using a shell script, as shown in Figure 5-13.

```
(undercloud) [stack@red-director script]$ vi baremetalprovisioning.sh
(undercloud) [stack@red-director script]$ ./baremetalprovisioning.sh
Started Mistral Workflow tripleo.baremetal.v1.register_or_update. Execution ID: 50edab6e-e1f6-4389-b13e-8f4717469b11
Waiting for messages on queue 'tripleo' with no timeout.


0 node(s) successfully moved to the "manageable" state.
Successfully registered node UUID 9dfc9708-9c12-4f1a-ba01-c2cbd8f3e170
+--------------------------------------+---------------------------+---------------+-------------+--------------------+-------------+
| UUID                                 | Name                      | Instance UUID | Power State | Provisioning State | Maintenance |
+--------------------------------------+---------------------------+---------------+-------------+--------------------+-------------+
| 9dfc9708-9c12-4f1a-ba01-c2cbd8f3e170 | red-compute3.redbooks.info | None          | power on    | manageable         | False       |
+--------------------------------------+---------------------------+---------------+-------------+--------------------+-------------+
Successfully added this baremetal for provisioning
Do you wish to provision this baremetal
1) Yes
2) No
```

Figure 5-13 Shell script automation

5.4 Red Hat Quay Registry Images Vulnerability Scanning using Clair Scanner

The repository is an important part of DevSecOps. The Quay is a secured private registry that is integrated with the Clair image Vulnerability scan. We use this registry for storing Docker images and scanning the images for vulnerability. Therefore, whenever a user pushes Docker images from their local development environment to the quay registry, the Clair starts scanning the image immediately. Refer to Figure 5-14 and Figure 5-15.

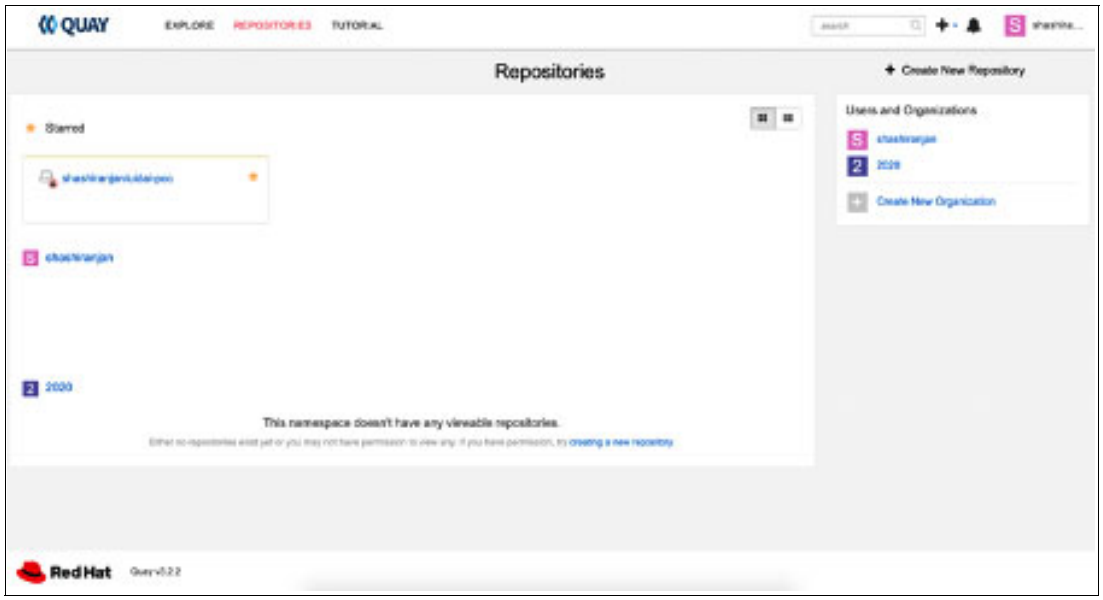


Figure 5-14 Red Hat Quay: Repositories window

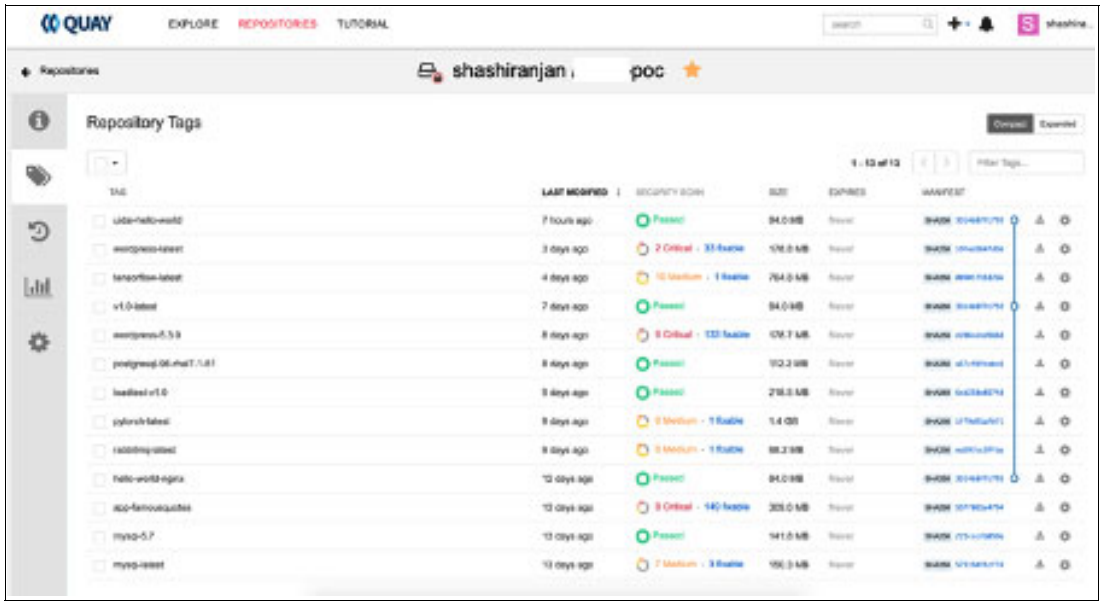


Figure 5-15 Red Hat Quay: Repository Tags window

You can configure Red Hat Quay to use the Clair as its security scanner, as shown in Figure 5-16. The Clair open source project is an image scanning engine that analyses the packages in the image layers. Red Hat only provides the Clair as a container image. When the Clair starts (and at regular intervals), it retrieves vulnerability metadata from various sources on the internet and stores them in its database.

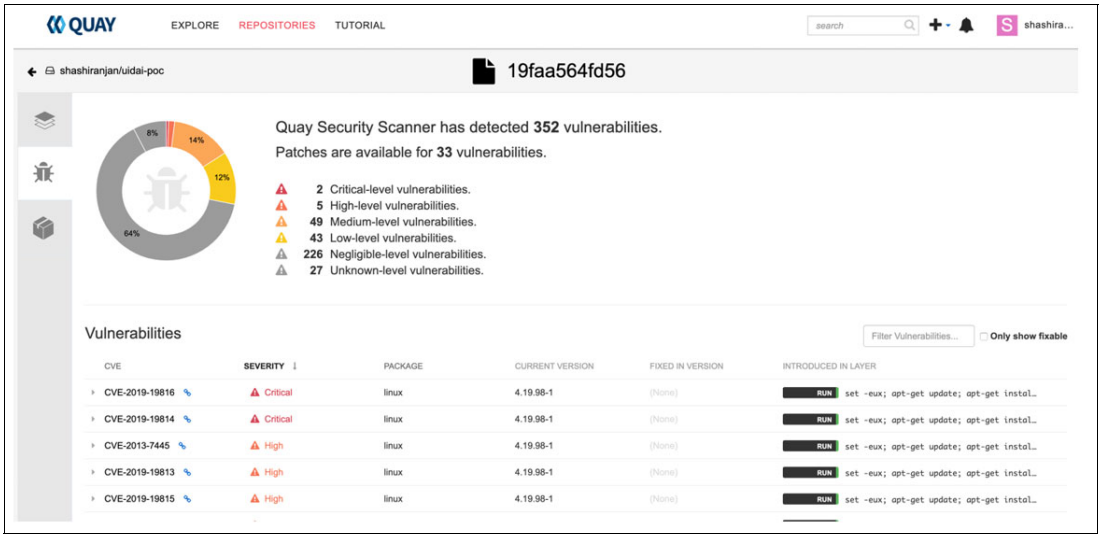


Figure 5-16 Red Hat Quay: Repositories: Vulnerabilities window

5.5 Two tier application deployment in Red Hat OpenShift container platform

Red Hat OpenShift is an enterprise container management platform that is based on Kubernetes. Red Hat OpenShift includes everything that is needed for hybrid cloud.

Red Hat OpenShift web console provides a graphical user interface to perform administrative, management, and troubleshooting tasks.

It supports Administrator and Developer perspectives. The task that we can perform on the UI also can be achieved through the operating system command line.

The dashboard that provides a quick overview of the cluster is shown in Figure 5-17 on page 372, including health metrics, resource counts, and a streaming list of events, such as machine updates or pod failures.

The left side of the dashboard provides a tool to manage the application on the cluster:

- Storage
- Networking
- Operators
- Compute
- User Management
- Administration

The developer view gives you an overview of the application setup:

- Build
- Create
- Topology

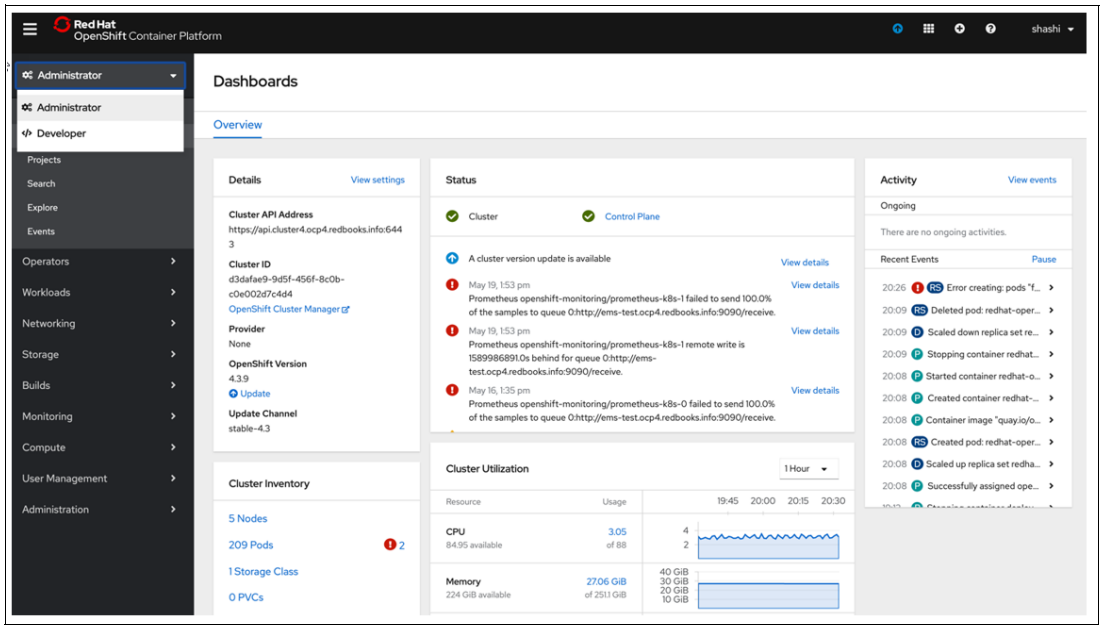


Figure 5-17 Red Hat OpenShift Container Platform: Dashboards window

This Red Hat OpenShift architecture has three masters and two worker nodes. Figure 5-18 shows the master controlling and managing the cluster and the worker node where the application was deployed.

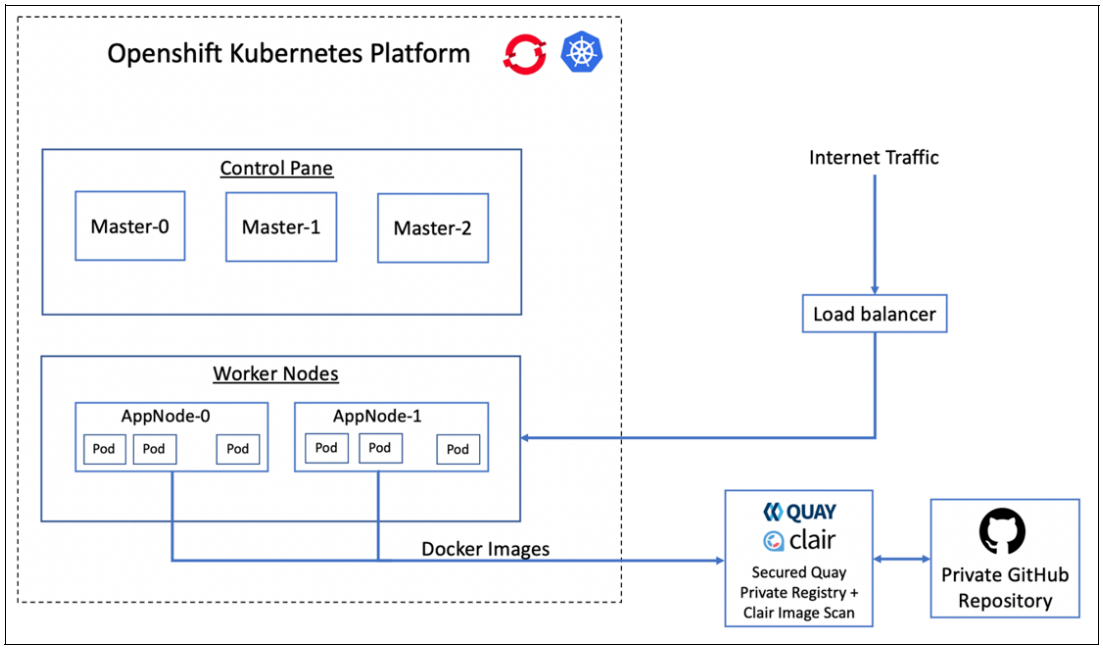


Figure 5-18 Red Hat OpenShift architecture

Figure 5-19 shows the dashboard view of the nodes in the cluster.

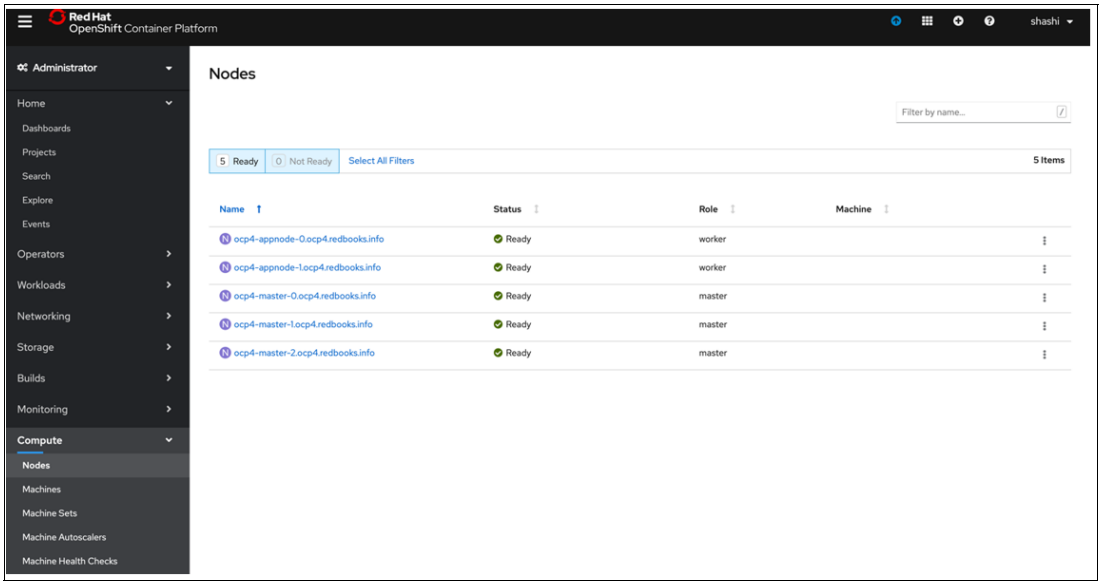


Figure 5-19 Red Hat OpenShift Container Platform: Nodes window

The WordPress application with MySQL is deployed from the Docker image, which was pushed to our private registry quay. Figure 5-20 shows the topology where there are two deployment configurations: one for WordPress and one for MySQL.

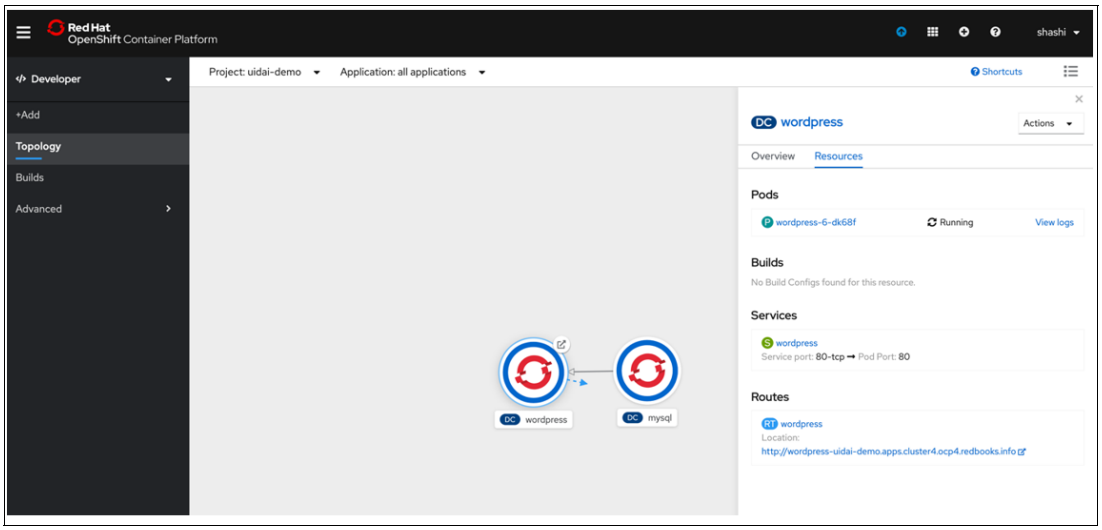


Figure 5-20 Red Hat OpenShift Container Platform: Topology window

Click **WordPress DC** and all of the information and URL of the WordPress application is displayed, as shown in Figure 5-21.

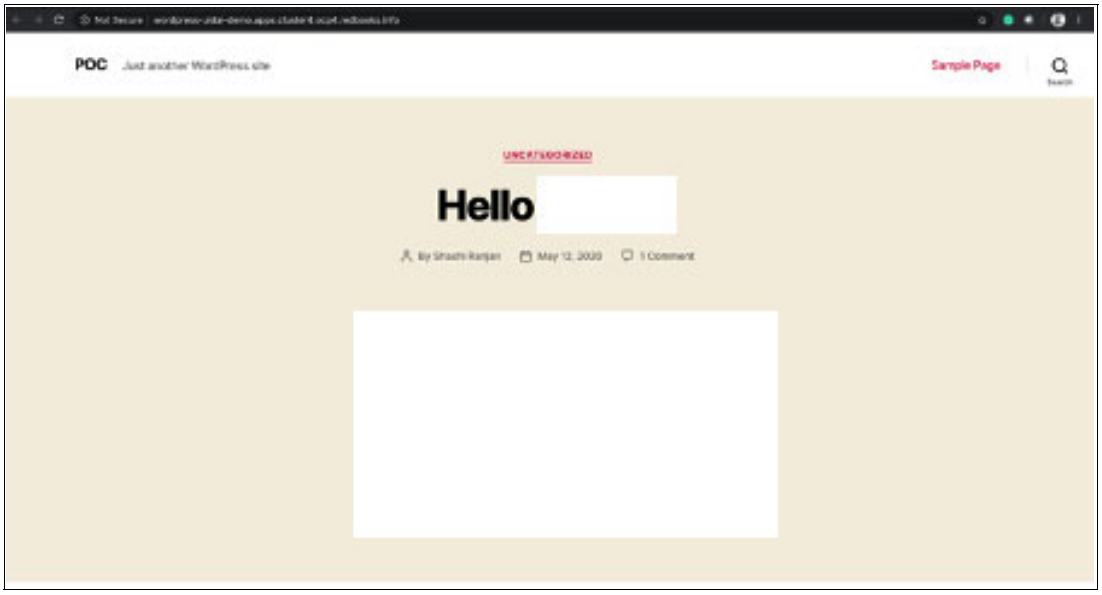


Figure 5-21 WordPress application window

5.6 Auto-scaling applications in Red Hat OpenShift container platform when load increases

We configured autoscaling on the pod so that whenever the CPU load increases above 50%, the pod begins scaling, as shown in Figure 5-22.

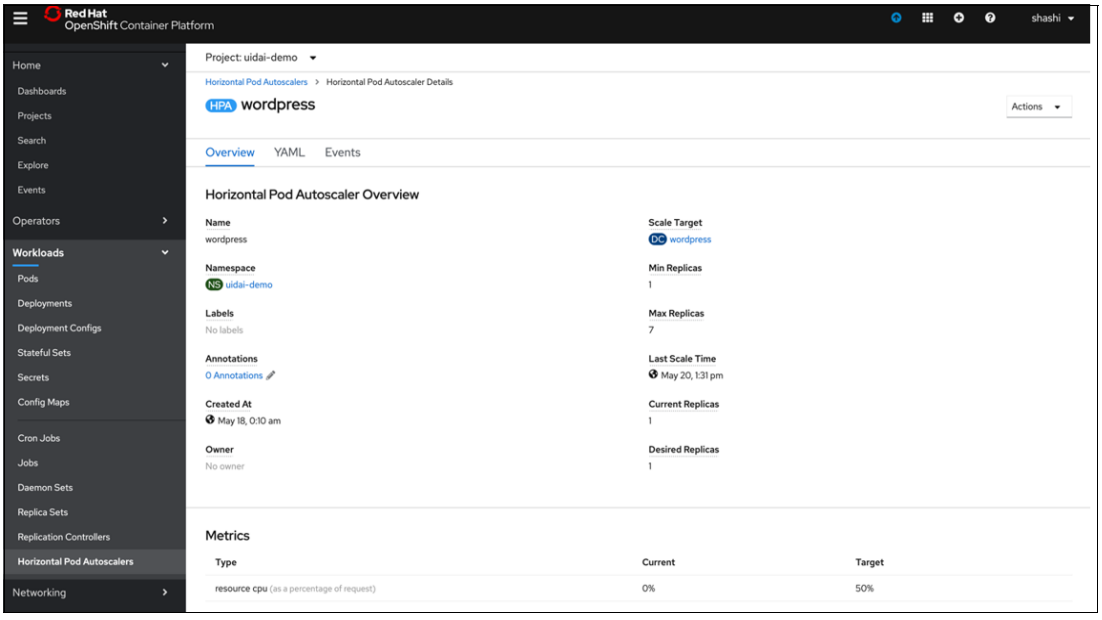


Figure 5-22 Red Hat OpenShift Container Platform: Workload: Horizontal Pod Autoscaling overview

5.7 Simulating a tier-2 application running Wordpress and mySQL

Apache Tomcat is an open source Java servlet container that implements several core Java enterprise specs, namely the Java servlet, JavaServer Pages (JSP), and WebSockets APIs. Tomcat is one of the most widely used web servers by Java developers. It is used to run the web applications.

5.7.1 Installing Tomcat

Complete the following steps to install Tomcat:

1. Check the server Java version.

   ```
   java -version
   ```

   The JAVA version must be V1.7 or V1.8 for Tomcat V9. If the Java version is older than version V1.7 or not present, download and install Java V1.8.

2. Download the Tomcat distribution (see Figure 5-23):

   ```
   wget
   https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.8/bin/apache-tomcat-9.0.8.
   tar.gz
   ```

```
[root@tomcat tmp]# wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.8/bin/apache-tomcat-9.0.8.tar.gz
```

Figure 5-23 Download Tomcat

3. Unpack the Tomcat distribution (see Figure 5-24):

   ```
   tar -xzf apache-tomcat-9.0.8.tar.gz
   ```

```
[root@tomcat tmp]# tar -xzf apache-tomcat-9.0.8.tar.gz
```

Figure 5-24 Extract the Tomcat files

4. Move the distribution to the `/usr/local/tomcat` folder (see Figure 5-25):

   ```
   mv apache-tomcat-9.0.8 /usr/local/tomcat9
   ```

```
[root@tomcat tmp]# mv apache-tomcat-9.0.8 /usr/local/tomcat9
```

Figure 5-25 Move the Tomcat distribution

5. Add a user to access the admin and management console:

   a. Open the `tomcat-users.xml` file, as shown in Figure 5-26.

```
[root@tomcat tomcat9]# vi /usr/local/tomcat9/conf/tomcat-users.xml
```

Figure 5-26 Open the xml file

b. Edit the file to add the admin user (see Figure 5-27).

Note: The role and user entry are within the `<tomcat-users>` tag.

```
<role rolename="admin-gui" />
<user username="admin" password="admin" roles="manager-gui,admin-gui" />
</tomcat-users>
```

Figure 5-27 Add the admin user

6. Edit the `context.xml` to allow users to access the management console:
   a. Open the `context.xml` file, as shown in Figure 5-28. The path of the file is `/usr/local/tomcat9/webapps/manager/META-INF/context.xml`.

```
[root@tomcat tomcat9]# vi /usr/local/tomcat9/webapps/manager/META-INF/context.xml
```

Figure 5-28 Edit the context.xml file

   b. Edit the file to comment `<Valve Classname >` tag (see Figure 5-29).

```
<!--
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
-->
```

Figure 5-29 Add comment tag

7. Start the Tomcat server by running the following command (see Figure 5-30):

   `/usr/local/tomcat9/bin/startup.sh`

```
[root@tomcat tomcat9]# /usr/local/tomcat9/bin/startup.sh
Using CATALINA_BASE:   /usr/local/tomcat9
Using CATALINA_HOME:   /usr/local/tomcat9
Using CATALINA_TMPDIR: /usr/local/tomcat9/temp
Using JRE_HOME:        /opt/jdk1.8.0_241/jre/
Using CLASSPATH:       /usr/local/tomcat9/bin/bootstrap.jar:/usr/local/tomcat9/bin/tomcat-juli.jar
Tomcat started.
[root@tomcat tomcat9]#
```

Figure 5-30 Start Tomcat server

8. To test Tomcat, open the Tomcat home page in the browser, as shown in Figure 5-31.

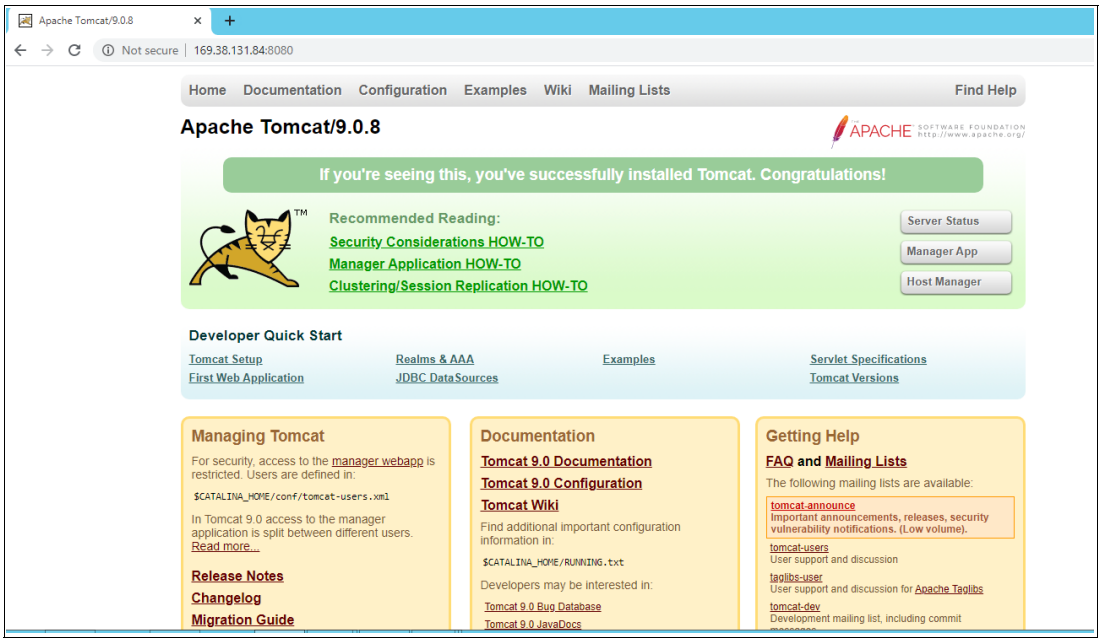


Figure 5-31 Tomcat home page

5.8 Open source automated alert and auto ticket

This section shows the configuration files building blocks for the auto ticket solution.

5.8.1 Configuration files

Figure 5-32 shows the `alertmanager.yml` configuration file.

```
[root@vsi-servicemanagement ~]# more Alertmanager/alertmanager-0.20.0.linux-amd64/alertmanager.yml
global:
  resolve_timeout: 1m

route:
  group_by: ['alertname','instance']
  group_wait: 10s
  group_interval: 5m
  repeat_interval: 1h
  #receiver: 'web.hook'
  #receiver: 'default-receiver'
  receiver: email-me
inhibit_rules:
- source_match:
    severity: '1'
  target_match:
    severity: '2'
  # Apply inhibition if the alertname is the same.
  equal: ['alertname','summary','instance']
receivers:
- name: email-me
  email_configs:
  - to: 219.ajit@gmail.com
    from: root@vsi-servicemanagement.doy.com
    smarthost: smtp.gmail.com:587
    auth_username: cloudibm2020@gmail.com
    auth_identity: cloudibm2020@gmail.com
    auth_password: uylugagvfofasggt
    send_resolved: true
    require_tls: true
  webhook_configs:
  - url: 'http://169.38.67.66:9000/hooks/runscript'
    send_resolved: true
[root@vsi-servicemanagement ~]#
```

Figure 5-32 alertmanager.yml file

Figure 5-33 shows the `Prometheus.yml` configuration file.

```
[root@vsi-servicemanagement ~]# more prometheus/prometheus.yml
# my global config
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    #path_prefix: /alertmanager
    - targets:
      - localhost:9093
      #- alertmanager:9093
      #- 169.38.67.66:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  - 'alert_rules.yml'
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['localhost:9090']
    - targets: ['server1:9100']
    - targets: ['server2:9100']
    - targets: ['134.209.22.37:9090']

  - job_name: 'node_exporter'
    static_configs:
    - targets: ['169.38.67.66:9100']
    - targets: ['134.209.22.37:9100']
    - targets: ['169.38.67.67:9100']
[root@vsi-servicemanagement ~]#
```

Figure 5-33 prometheus.yml file

Figure 5-34 shows the alert and metric configuration rules file.

```
[root@vsi-servicemanagement ~]# more prometheus/alert_rules.yml
groups:
  - name: example
    rules:
    - alert: InstanceDown2
      expr: up == 0
      for: 1m
      labels:
        severity: 1
        priority: 1
        summary: "Instance {{ $labels.instance }} down"
        description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 5 minutes."
    - alert: NodeDiskReadCount
      expr: node_disk_reads_completed_total > 9134000
      for: 1m
      labels:
        severity: 4
        priority: 4
        summary: "Instance {{ $labels.instance }} has Node Disk Read Count crossing threshold Issue"
        description: "{{ $labels.instance }} of job {{ $labels.job }} has Node Disk Read Count > 1134000"
    - alert: NodeFileSystemFree
      expr: node_filesystem_files_free > 9043381
      for: 1m
      labels:
        severity: 2
        priority: 2
        summary: "Instance {{ $labels.instance }} has Node File System Free  crossing threshold Issue"
        description: "{{ $labels.instance }} of job {{ $labels.job }} has Node File System Free > 1043381"
    - alert: NodeCPUSecondTotal
      expr: node_cpu_seconds_total{mode="iowait"} > 940095
      for: 1m
      labels:
        severity: 1
        priority: 1
        summary: "Instance {{ $labels.instance }} has Node CPU Second Total crossing threshold Issue"
        description: "{{ $labels.instance }} of job {{ $labels.job }} has Node CPU Second Total > 840095."
    - alert: HostHighCpuLoad
      expr: avg by(instance)(irate(node_cpu_seconds_total {mode="user"} [1m]) * 100) > 95
      for: 1m
      labels:
        severity: 1
        priority: 1
        summary: "Host high CPU load {{ $labels.instance }}"
        description: "CPU load is > 95%"
```

Figure 5-34 alert_rules.yml file

Figure 5-35 shows the webhook configuration file (`hooks.json`).

```
[root@vsi-servicemanagement j123]# more hooks.json
[
{
"id": "runscript",
"execute-command": "/home/j123/sendmessage.sh",
"command-working-directory": "/home/j123/",
"pass-arguments-to-command":
[
{
  "source": "payload",
  "name": "alerts.0.labels.instance"
},
{
  "source": "payload",
  "name": "alerts.0.labels.alertname"
},
{
  "source": "string",
  "name": "is above threshold"
},
{
  "source": "string",
  "name": "HARD"
},
{
  "source": "payload",
  "name": "alerts.0.labels.severity"
},
{
  "source": "payload",
  "name": "alerts.0.labels.priority"
},
{
  "source": "payload",
  "name": "alerts.0.labels.summary"
},
{
  "source": "payload",
  "name": "alerts.0.labels.description"
}
]
}
]
[root@vsi-servicemanagement j123]#
```

Figure 5-35 hooks.json file

5.8.2 Alert generation

Figure 5-36 shows all the configured metrics. The inactive state means that the metric value did not cross the threshold. The pending state means that the metric value crossed the threshold value and waits for a number of x seconds for the alert to persist before firing the alert to ensure it is not a temporary spike.

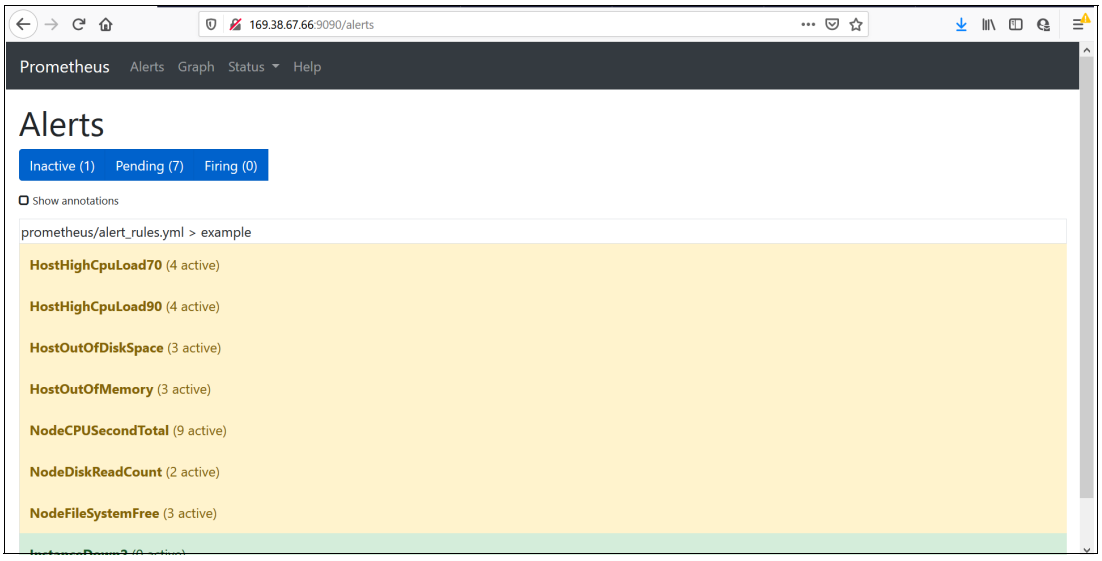


Figure 5-36 Prometheus Alerts window: Inactive state

Figure 5-37 shows metrics are in a firing state, which means alerts are fired to Alertmanager.

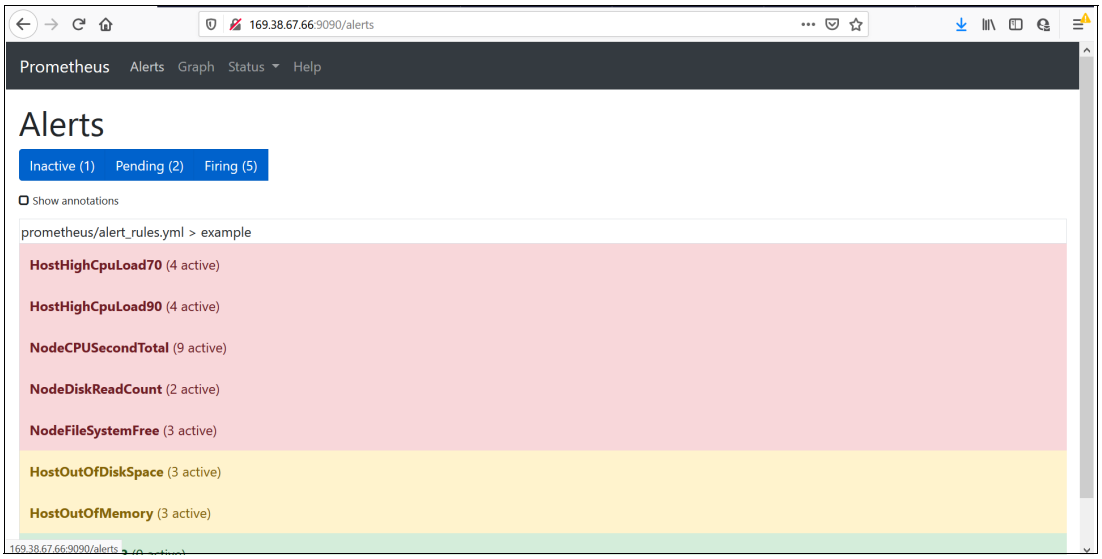


Figure 5-37 Prometheus Alerts window: Firing state

Figure 5-38 shows seven firing alerts are tiriggered and shown in the Prometheus window.

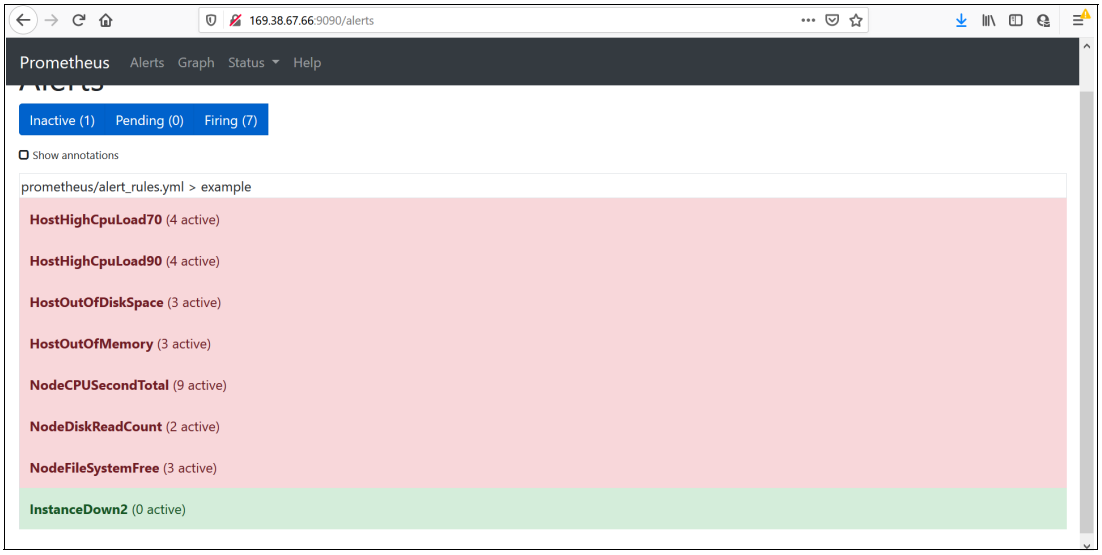


Figure 5-38 Prometheus Alerts window: Firing state shows seven

Figure 5-39 shows Alert Manager snapshot showing alerts reached Alert Manager.

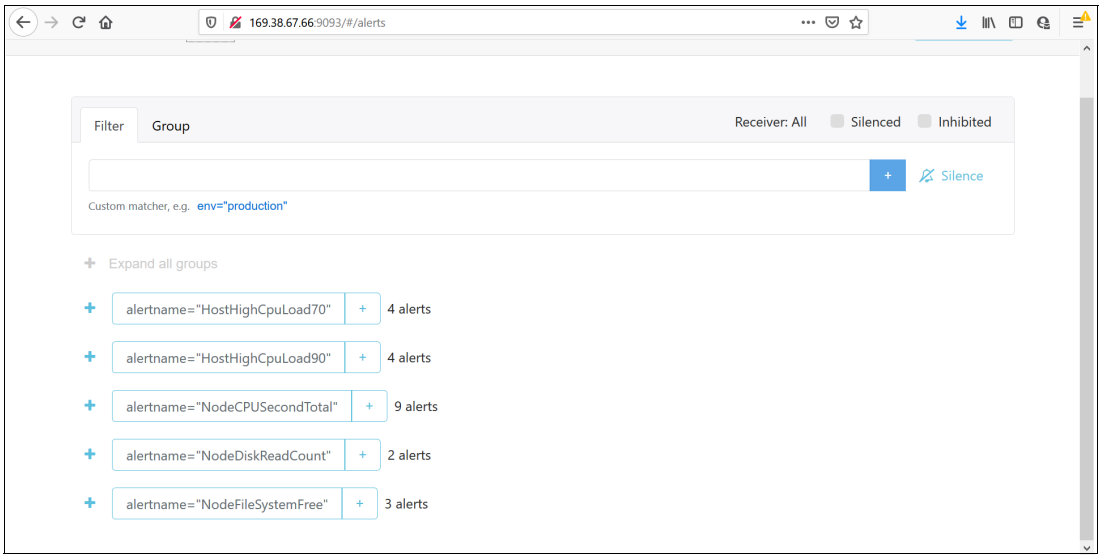


Figure 5-39 Prometheus Alert Manager: Filter alerts for production environment

5.8.3 Auto-Ticket (Alert to Ticket creation)

Figure 5-40 shows the webhook logs that show how each alert is processed and converted to tickets in iTop (see Figure 5-41 on page 382).

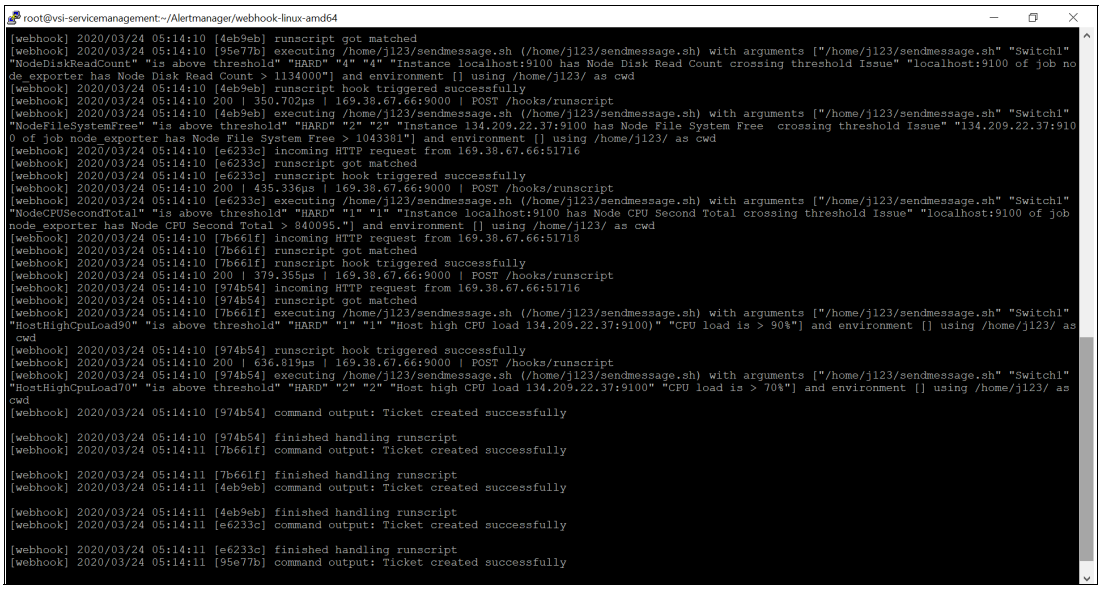


Figure 5-40 Logs created for the alert messages

Figure 5-41 shows the iTop window displaying the tickets that were created from the alerts (Alert Manager).

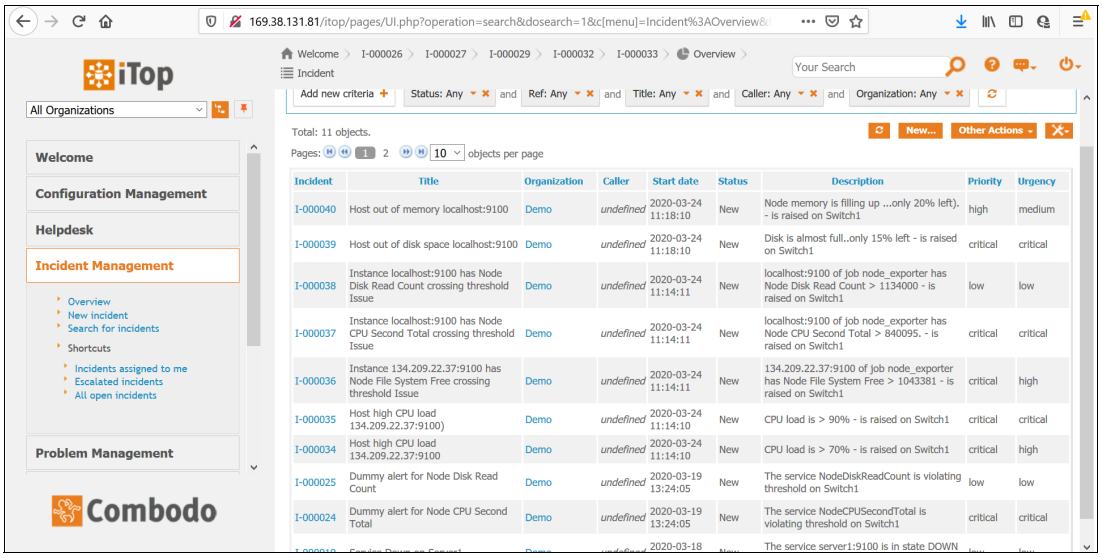


Figure 5-41 iTop Incident Management window

5.8.4 Middleware (Apache) monitoring

Complete the following steps to install the Apache middleware for monitoring:

1. Download `apache_exporter` from GitHub, as shown in Figure 5-42.

Figure 5-42 Get apache_exporter

2. Configure `apache_exporter` and start as a system service as shown in Figure 5-43 and Figure 5-44.

Figure 5-43 Configure and starting apache_exporter

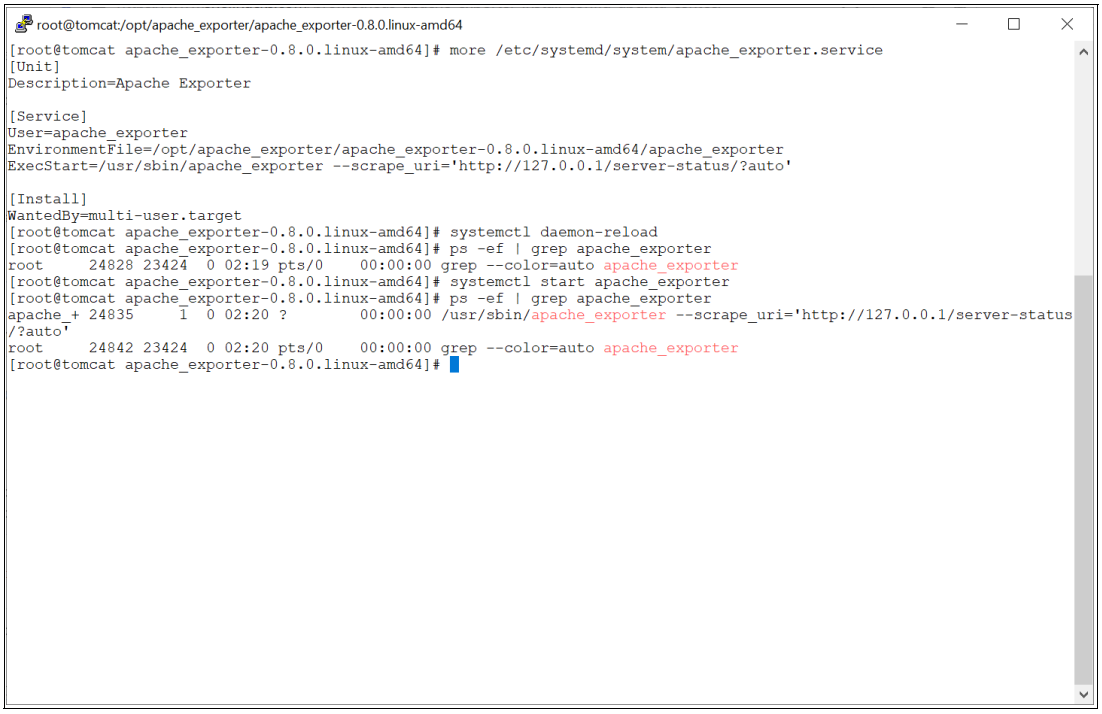


Figure 5-44 Check configuration: apache_exporter

3. Verify that `apache_exporter` is collecting the metrics, as shown in Figure 5-45.

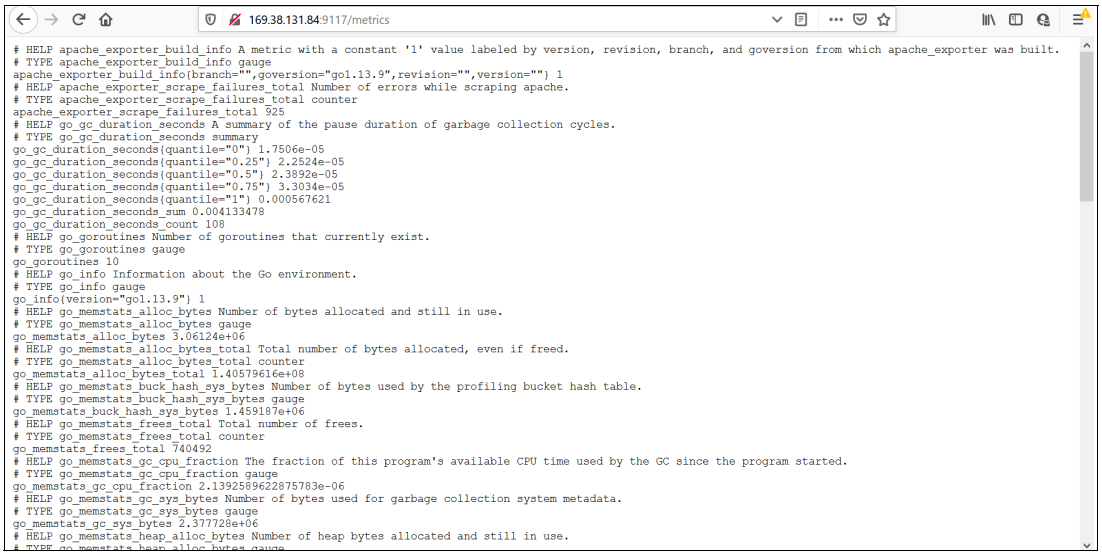


Figure 5-45 Verify apache_exporter

4. Update the Prometheus configuration to collect metrics from `apache_collector`, as shown in Figure 5-46.

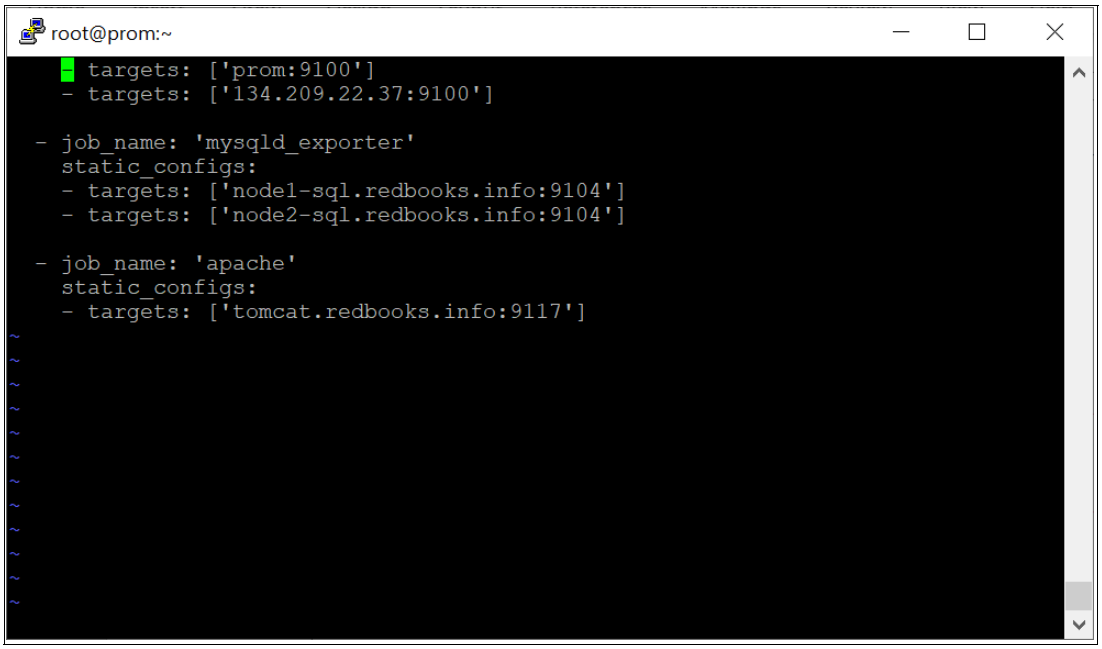


Figure 5-46 Prometheus configuration updates

5. Verify that the `apache_exporter` endpoint is available in Prometheus, as shown in Figure 5-47.

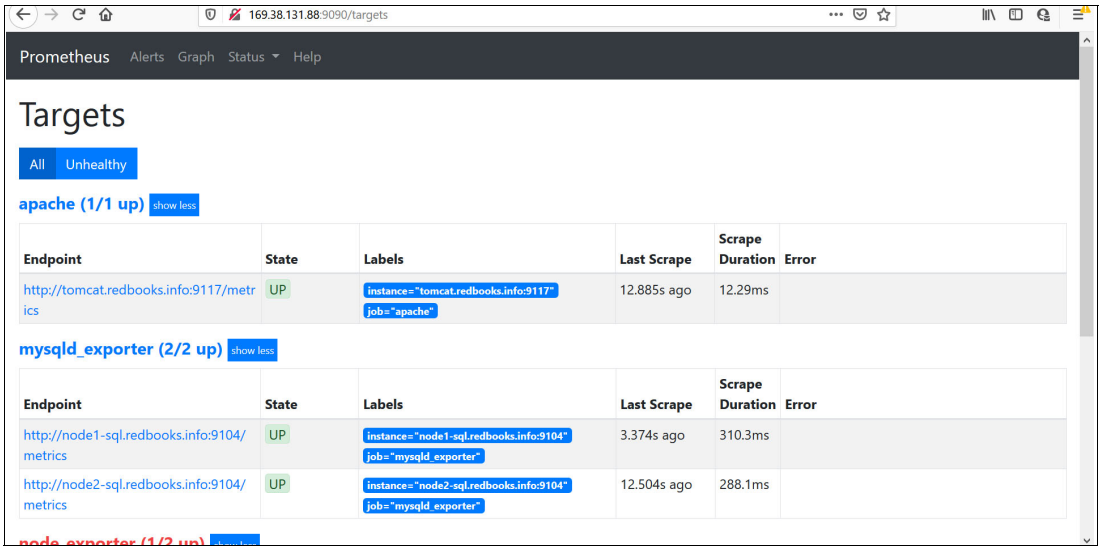


Figure 5-47 Prometheus: Targets window

5.8.5 Configuring Mysqld_exporter

Complete the following steps to configure `mysqld_exporter`:

1. Download the `mysqld_exporter` from GitHub, as shown in Figure 5-48.

Figure 5-48 Get mysqld_exporter

2. Configure `mysqld_exporter` as a service, as shown in Figure 5-49.

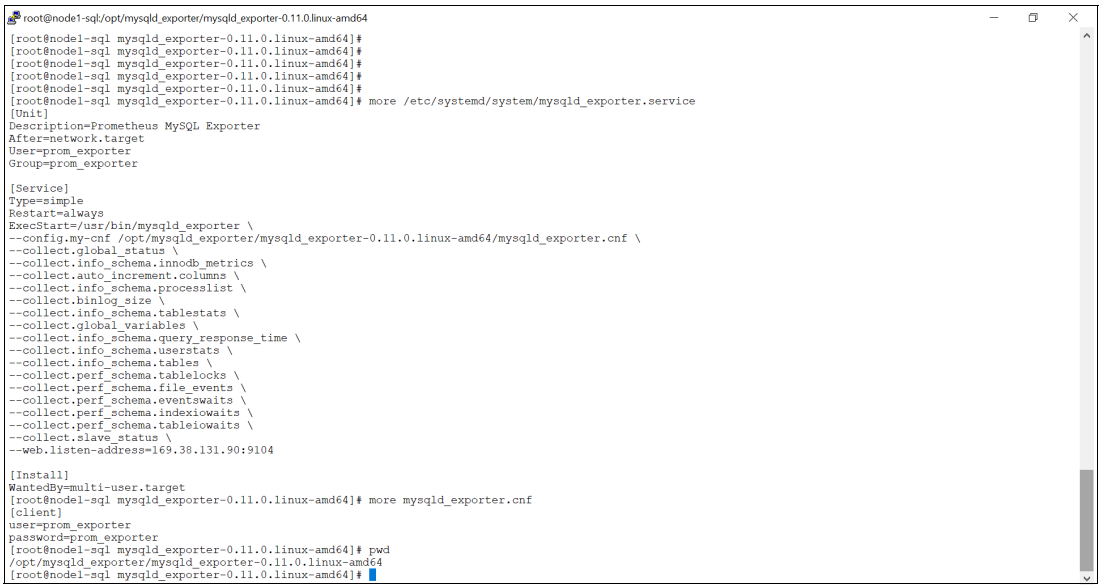


Figure 5-49 Configure mysqld_exporter

3. Start `mysqld_exporter` service, as shown in Figure 5-50.

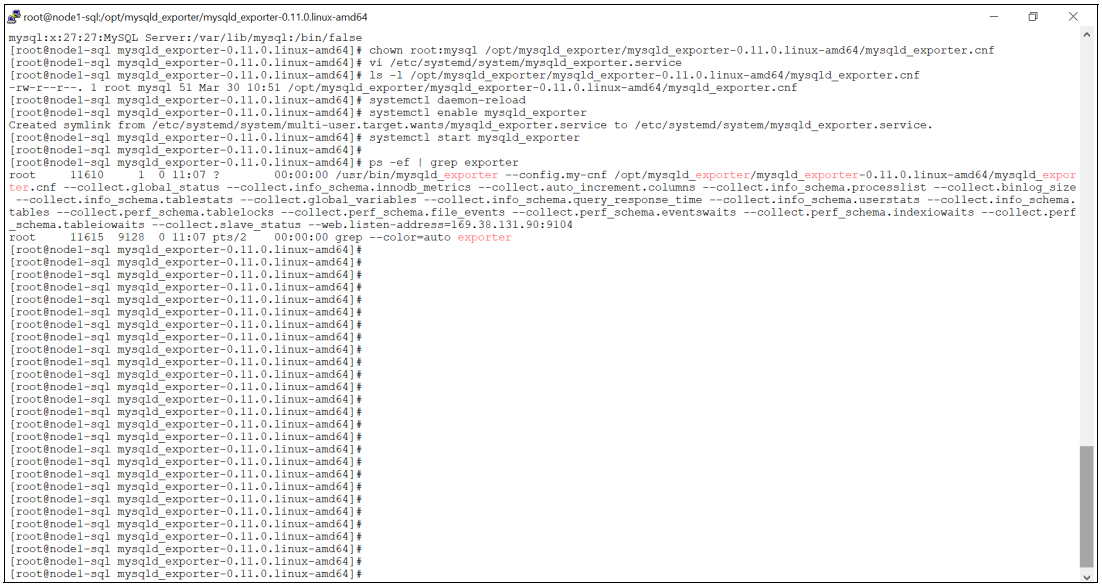


Figure 5-50 Start the service

4. Verify that `mysqld_exporter` is collecting the metrics, as shown in Figure 5-51.

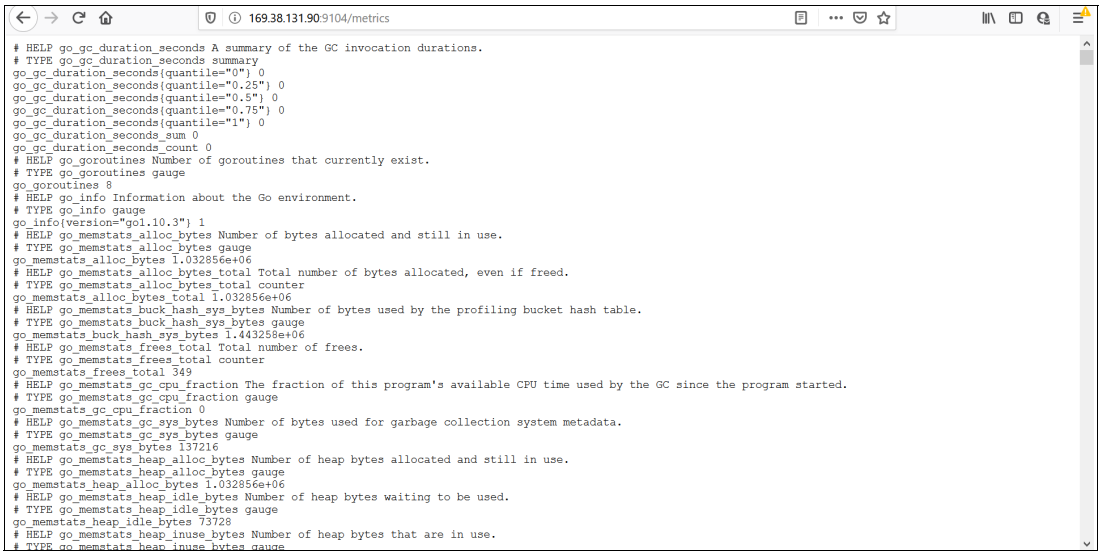


Figure 5-51 Verify mysqld_exporter collects metrics

5. Update the Prometheus configuration to pull the metrics from `mysqld_exporter`, as shown in Figure 5-52.

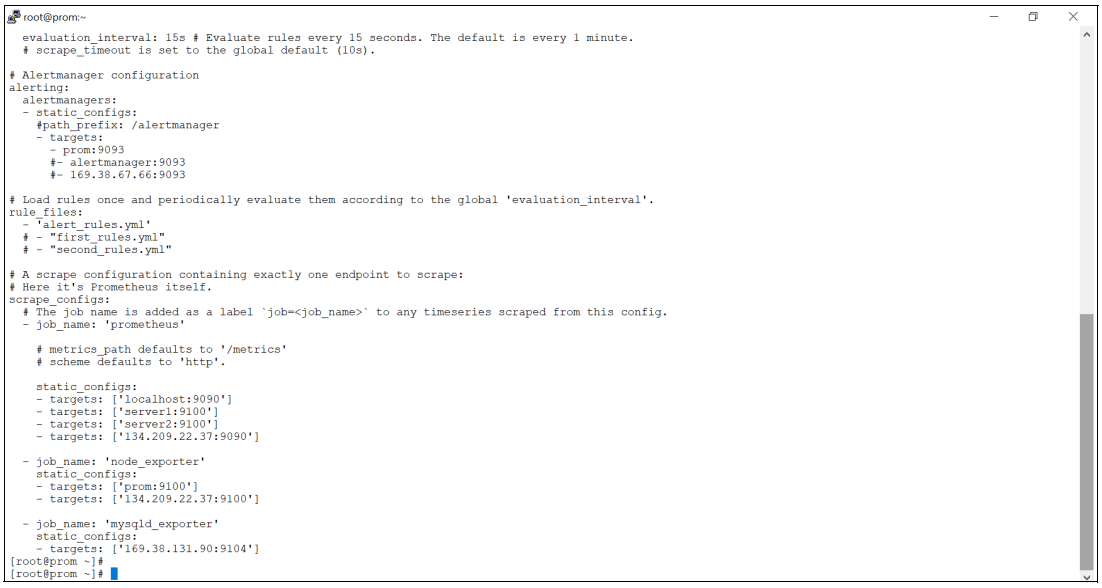


Figure 5-52 Prometheus configuration updates

Now, `mysqld_exporter` is available as endpoint in the Prometheus target list, as shown in Figure 5-53.

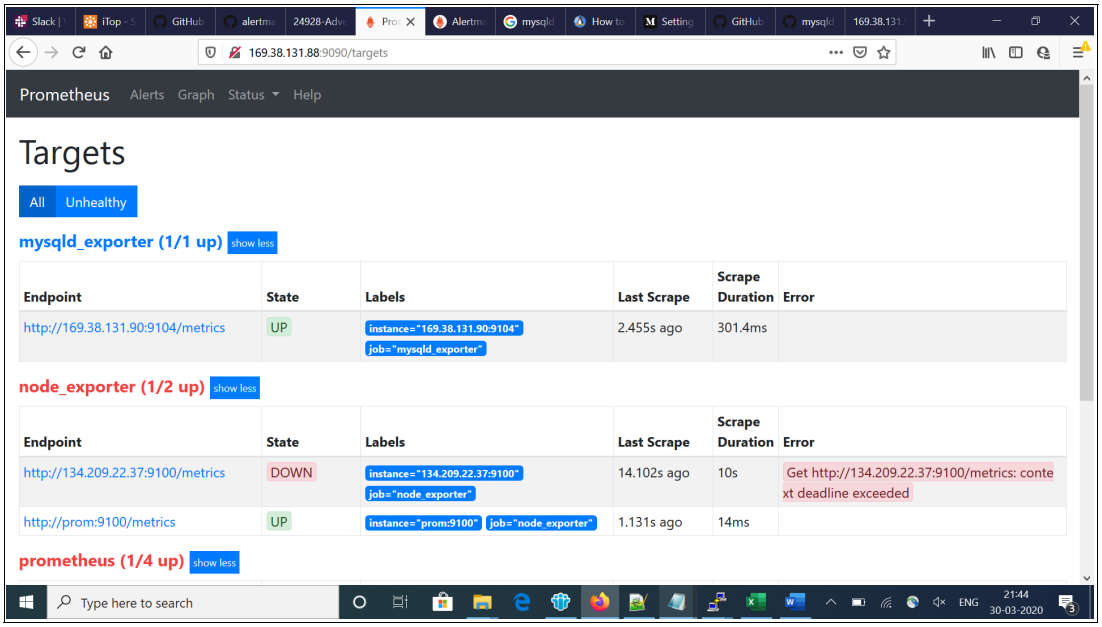


Figure 5-53 Prometheus Targets window: Endpoints

5.9 Anomaly detection using Elasticsearch and Python

Performance data is gathered into Prometheus (by using exporters) and Elasticsearch (by using beats). When various plots are displayed in Grafana by using Prometheus data source, no convenient way is available to detect anomalies.

In Elasticsearch, x-packs are supported for Anomaly Detection, but they are available with subscription and therefore Anomaly Detection must use Python.

Elasticsearch APIs are available, which were used in Python to collect data and later simple Anomaly Detection technique were used.

5.9.1 Setup

A python environment can be set up by using a python environment or a conda environment. Figure 5-54 and Figure 5-55 show the list of python libraries that were used to perform anomaly detection, and must be installed in the python environment by using pip and conda.

```
import pyodbc
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import json
from elasticsearch import Elasticsearch
from elasticsearch_dsl import connections, Search
from elasticsearch.helpers import bulk
from datetime import datetime, timedelta
```

Figure 5-54 Phyton libraries

```
from sklearn.metrics import r2_score, median_absolute_error, mean_absolute_error
from sklearn.metrics import median_absolute_error, mean_squared_error, mean_squared_log_error
```

Figure 5-55 Phyton libraries, continued

After the libraries are installed, a script can be developed to run the following commands to detect anomalies:

- Elasticsearch shows the search API that can be used to filter the data. For example, `node_load1` is considered as the metrics for which the anomaly is detected.
- In Elasticsearch, data is stored in indexes and to fetch the data you need to know the index name.

Figure 5-56 shows the script that is used to connect to Elasticsearch.

```
dsl_conn = connections.create_connection(hosts=['169.38.131.87:9200'], timeout=20)
```

Figure 5-56 Script to connect to Elasticsearch

Figure 5-57 on page 390 shows the scripts that are used to perform the search index that is specified and fetch data locally.

Note: The host can require authentication and the same must be mentioned in the connection.

```
body={
        "query": {
            "bool": {
                "must": [
                    {"exists": {"field":"prometheus.metrics.node_load1"}}
                ],
                "filter": [
                    {"range": {"@timestamp": {"gt": "now-1d", "lt": "now"}}},
                ],
            }
        }
    }
```

Figure 5-57 Script to perform the index search

Figure 5-58 shows the script that is used as function to fetch the matched result.

```
# valid client instance for Elasticsearch
docs=[]
if dsl_conn != None:

    # keep track of the number of the documents returned
    doc_count = 0

    # make a search() request to get all docs in the index
    resp = dsl_conn.search(index=index,
                           scroll='2s', size=100, body=body)

    # keep track of pass scroll _id
    old_scroll_id = resp['_scroll_id']
    while len(resp['hits']['hits']):

        # check if there's a new scroll ID
        if old_scroll_id != resp['_scroll_id']:
            print ("NEW SCROLL ID:", resp['_scroll_id'])

        # keep track of pass scroll _id
        old_scroll_id = resp['_scroll_id']

        # iterate over the document hits for each 'scroll'
        for doc in resp['hits']['hits']:
            docs.append(doc['_source'])
            doc_count += 1
#         print ("DOC COUNT:", doc_count)


        # make a request using the Scroll API
        resp = dsl_conn.scroll(
            scroll_id = old_scroll_id,
            scroll = '2s' # length of time to keep search context
        )
```

Figure 5-58 Script to fetch the matched results

Figure 5-59 shows the script that was used to create the data frame for analysis. Time for `node_load1`, `node_load5`, `node_load15`, and `hostname` were fetched.

```
cpu_util = pd.DataFrame({}, columns=['time', 'hostname', 'node_load1', 'node_load5', 'node_load15'])

counter = -1
for doc in docs:
    counter = counter+1
    cpu_util.loc[counter] = [datetime.strptime(doc['@timestamp'], '%Y-%m-%dT%H:%M:%S.%fZ'),
                             doc['prometheus']['labels']['instance'],
                             doc['prometheus']['metrics']['node_load1'],
                             doc['prometheus']['metrics']['node_load5'],
                             doc['prometheus']['metrics']['node_load15']]
```

Figure 5-59 Script to create the data frame for analysis

Metrics for `node_load5` and `node_load15` can be skipped because these metrics were added to perform same anomaly detection as `node_load1`. The time stamp is required because anomaly detection must be done on time series data.

The host name was considered because the same metrics are coming from multiple instances and hosts. This field also can be used to further filter the data.

Note: The host name filter can be applied to the Elasticsearch query. However, to keep the Elasticsearch query simple, it was used and a comparison of the number of anomalies can give an estimate of which host is overburdened or not performing optimally.

Figure 5-60 shows the script filtering data that was scraped from SQL server node2, and sorts the data based on time.

```
node2_data = cpu_util.loc[cpu_util['hostname']=='node2-sql_server', :].sort_values('time')
```

Figure 5-60 Script to filter data from scrapped nodes

Figure 5-61 shows the script where the lower and upper boundaries are evaluated. Based on these boundaries, the anomalies are reviewed.

```
def anomalyDetector(data, window, scale=1.5):

    for col in data.columns:
        if col!='time':
            rolling_mean = data[col].rolling(window=window, center=False).mean()

            plt.figure(figsize=(20,8))
            plt.title(f"Moving average for {col}\n window size = {window}s")
            plt.plot(data.time, rolling_mean, "g", label="Rolling mean values")

            # for each point evaluate the mean absoulute error between actual data and rolled data points
            mae = mean_absolute_error(data[col][window:], rolling_mean[window:])
            deviation = np.std(data[col][window:] - rolling_mean[window:])

            # Using scale on the mae the lower and upper boundary are decided
            lower_boundary = rolling_mean - (mae + (scale * deviation))
            upper_boundary = rolling_mean + (mae + (scale * deviation))

            #the boundaries are added to dataframe and plotted

            plt.plot(data.time, upper_boundary, "r--", label="Upper Bond / Lower Bond")
            plt.plot(data.time, lower_boundary, "r--")

            #any point that has crossed the boundaries becomes anomaly
            anomalies = data.loc[(data[col] < lower_boundary) | (data[col] > upper_boundary), col].index
            data[f'{col}_anomaly'] = data.loc[anomalies, col]

            #pointing out the anomanlies in the plot
            plt.plot(data.loc[anomalies,'time'], data.loc[anomalies, col], "ro", markersize=10)

            plt.plot(data.time, data[col], "b" ,label="Actual values")
            plt.legend(loc="upper left")
            plt.grid(True)
```

Figure 5-61 Script for anomalies detection

The upper and lower boundaries are based on mean absolute error value and standard deviation between normal value and rolled value, as shown in Figure 5-62.

```
sample = node2_data[['time', 'node_load1']].copy()
anomalyDetector(sample, 30, scale=1.5)
```

Figure 5-62 Boundaries set

Figure 5-63 shows a plot that is the result of the script that is shown in Figure 5-61 on page 392. The red circles represent the anomalies.

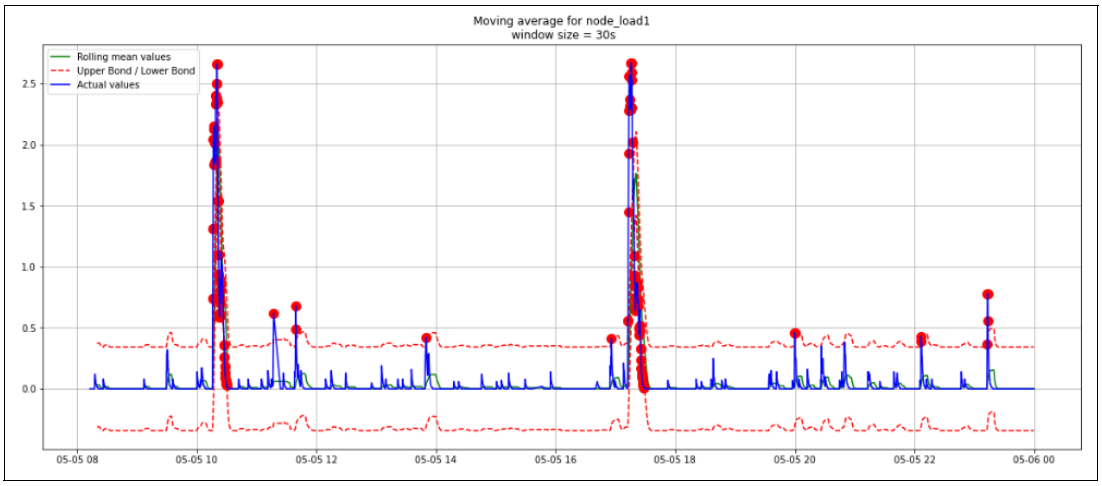


Figure 5-63 Anomalies plot

5.10 ELK stack for centralize monitoring

The ELK stack is the most popular solution, particularly in the open source world, for centralized logging. Elasticsearch is a full-text search and analysis engine where your data is stored.

Logstash aggregates logs and processes them before sending them on to Elasticsearch for indexing. Kibana provides an easy way to query the data and build dashboards.

One of the benefits of the use of the ELK stack and centralized logging for server log analysis is the ability to visualize the data in a single dashboard. For example, Figure 5-64 shows MYSQL dashboard that provides system metrics, system logs, and MYSQL application logs to be viewed in a single dashboard.

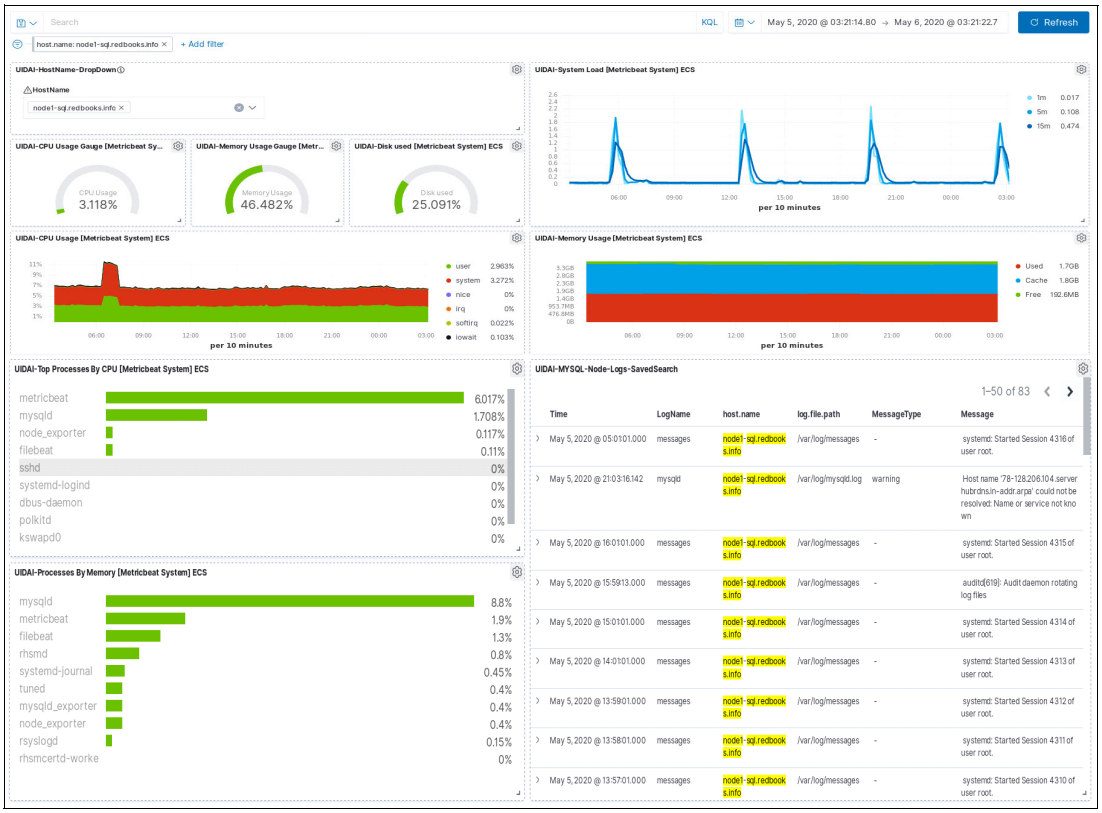


Figure 5-64 MYSQL dashboard

6

Site Reliability Engineering delivery model

This chapter discusses the Site Reliability Engineering (SRE) delivery model and includes the following topics:

6.1 Introduction

With today's collective knowledge and understanding, it is key to accept that traditional IT support is silo-based and limited from crossing barriers from one silo to another. Hence the reason why integration positions exist to try to enhance a more precise understanding of how an application works.

Today, major tech corporations run most of the systems with SRE practices by using clean and precise knowledge. But what about those enterprises that their core business is not cloud or a provider of technology services?

IBM grabs core principles and applies them to IT enterprises. But, how do we apply the sysadmin model that is running complex computing systems and inject principles that are too dramatic in running IT workloads in a new service management model?

Consider an example of what traditional means for a service integrator. Today, a sysadmin includes building software components and running a deployment of these components to efficiently work together. The service integrator is responsible for providing such a function and in return, the enterprise receives what we call *service*.

After these services are running, the service integrator is responsible for managing these workloads and operating them to keep them running, and provide key performance indicators, service level operators, and service level agreements that they need to meet. However, one of the essential distinctions of the traditional way is that operations are the responsibility of the service integrator is now divided from developers that are still are on the enterprise or outsourced to another service provider that owns the application development.

Service integrators are well-suited for running these operations because they include most of the expertise and skills in the market and can staff quick requests from enterprises.

In principle, a service integrator with site reliability engineering must have SRE teams that are focused on having software engineers to run products and create systems that accomplish the work that are run by sysadmins.

SRE applies the principle that developers must spend most their time not focused on design or implementations, but on run activities that are related to the product they created.

For technology companies, not all systems are what we call workload and application candidates for SRE. The expertise and the skills that exist are limited with that end-to-end knowledge.

The service integrator, with the enterprise, must evaluate the workloads schemes and select which workloads are the core workloads and require management with SRE management practices. This team owns end-to-end service, which starts at design (participate), implement (execution with assistance from the design team), and manage with assistance from the design team.

SREs become a part of IT operations (ITOPS), which is one of the critical activities to assure that SREs solve complex and common problems by using software development practices and new automation scripting to handle the repetitive tasks that a sysadmin can handle with manually operated tasks.

One of the key focus areas of the Service Integrator is to retain this SRE team intact, and for that, there is a critical requirement that the SREs do not get bored while doing the work. In

principle, SREs are individuals that are dynamic and have many skills that can consistently change over time.

The Service Integrator SRE is responsible for handling the essential responsibilities for the services they support. This SRE and its team make the joint squad 100% responsible for the service management practice that includes service availability, performance, change management, monitoring, latency, efficiency, emergency response, and capacity planning of the service they manage.

6.2 Managing a hybrid cloud

Automated management systems have long existed in IT departments and are part of ITOPS today. However, the core design principle of systems management deviated to a point where operations stand between the evolution of systems and innovation.

Service management has gotten to the point that it slows down the development of systems. One of the critical requirements in ITOPS is that developers and programmers are working towards new business opportunities, but with the key to deploying code in a faster an accelerated DevOps delivery model. The ability to run a self-automated delivery model is essential to support a Hybrid cloud. The IT service operations mandate that IBM service solution for ITOPS changes from keep systems running with embedded service levels and operational levels to a managed service that can run the environment and have integrated steps to avoid pitfalls during the implementation of new software releases and upgrades.

Therefore, processes were in place, such as development releases and test environment releases to a point where pre-production releases exist before any change occurs to the environment. As part of this delivery practice, IBM uses the proposed service management platform to interlock service delivery with DevOps delivery practices by cross skills and site reliability engineers.

Enterprises' high demand for infrastructure consumption ranges from network to compute, from compute to storage, and so on. To provide the increased demand from the new accelerated development workloads, network utilization throughput, and application interconnectivity, the complexity of enterprise systems had IBM generate an agile DevOps delivery model that contains service keepers (SREs).

As part of the service delivery, the SRE's proposition for management of hybrid cloud distributes them across three squads that are based on the following primary services that IBM provides:

- Infrastructure as a Service (Red Hat OpenStack)
- Platform as a Service (Red Hat OpenShift)
- Service Management System (SMS)

Each of these services are measured with a set of service level indicators that can be adjusted iteratively. In IBM service delivery for ITOPS, the SLA is a generated output of the suitable level of service levels that are required to be measured. For this, the service SRE leader generates an extensive list of service level indicators and objectives that in return meet the required service level agreement that is required by ITOPS.

These parameters permit the measurement of the required performance and availability of the service. SRE enhances over an extensive period the measurement of these parameters and allows them to improve gradually.

We recommend the following metrics for starting SLIs for ITOPS:

- Availability and uptime of the service
- Number of successful transactions and requests
- Consistency and durability of the data
- Latency
- Error rate
- Throughput
- Availability
- Response times

All of these metrics are captured by the SMS solution by using the Prometheus monitoring tool.

Service Level Objectives (SLO) for ITOPS define the acceptable downtime of each individualized service. To expand in this service level objective in ITOPS, multiple components exist as part of the service. For these items, SREs define the different parameters that are acceptable for downtime. IBM starts ITOPS service delivery with median SLOs and gradually increases it through the contract. IBM services set low-end and upper-end boundaries for many of these SLOs during measurement of service objectives.

The following starting SLOs are available for ITOPS:

- Durability of disks must be 99.9%
- Availability of service must be 99.95%
- Service must successfully serve 99.999% requests and transactions

With these three starting SLOs, IBM seeks to meet the minimum viable service level and still deliver acceptable quality to ITOPS.

As a result, the required SLA that is requested in the RFP is met by the establishment of the suitable SLOs and SLIs. It is the SRE's core responsibility to assure that all the required SLIs and SLOs are met to achieve the SLAs.

ITOPS SRE team are responsible for the following tasks:

- System availability
- Application performance and infrastructure performance
- Latency
- Efficiency
- Change management
- Monitoring
- Emergency response
- Capacity planning

This delivery model adjusts to the ask of ITOPS of integrating cross-functional teams with application development cycles. It also creates an Infrastructure team that has a clear understanding of how the application works.

The IBM proposed delivery model grabs core principles and applies them to ITOPS. With the intent to clear gaps in the understanding of the delivery model, consider the following example:

In an ITSM traditional process and IT, the organization provides a sysadmin with tasks, which includes building software components and running a deployment of these components to efficiently work together. The IT resource is responsible for providing such a function and in return, ITOPS receives and operational IT service delivers this IT resource.

However, one of the essential distinctions of the traditional way is that operations that are responsible for the service that is provided is divided between application developers that often are still on ITOPS or outsourced. IBM for ITOPS is a service integrator with site reliability engineering. With this service, the delivery teams align in cross-functional squads that are lead by one SRE as leader of the service that was delivered for ITOPS.

Now, the delivery teams focus on having software engineers and infrastructure engineers to run the IaaS, PaaS, and EMS. This combination allows the squad to support the Infrastructure and automate the work that is regularly accomplished by step-by-step sysadmin work.

In ITOPS, not all systems are what we call workload and application candidates for SRE. The expertise and skills are still limited by that end-to-end knowledge. For this reason, IBM selected the three primary services that IBM provides to ITOPS.

As part of the delivery, key activities are practiced to assure that SREs solve complex problems and joint problems with software development and new automation scripting to handle the repetitive tasks that a sysadmin handle with manually operated tasks.

These SREs are assigned one or multiple cross-functional squads that support the services.

The teams are distributed based on cross-functional skills. Therefore, all team members of the squad have vertical expertise but common knowledge on all other relevant areas that compose the provided service.

The squads for ITOPS are composed of cross skilled resources that total 64 resources to achieve 24 x 7 service delivery continuation.

For IaaS, the following skills are available with the squad:

- Hardware network
- Hardware server infrastructure backup
- Hardware storage infrastructure and Ceph - SAN
- Red Hat OpenStack - Red Hat Satellite - Red Hat CloudForms
- KVM - RHV
- Red Hat Ansible patch configuration - compliance
- Site Reliability Engineer
- Security and compliance

This IaaS squad is composed of 13 total resources across all the mapped skills. This single squad focus is to provide the IaaS services that encompass the private cloud for ITOPS.

For PaaS, the following skills are available with the squad:

- Hardware network
- Hardware server infrastructure specialist backup
- Hardware storage infrastructure and Ceph - SAN
- Red Hat OpenShift - Red Hat CloudForms
- Red Hat Ansible Patch Configuration - Compliance
- Site Reliability Engineer
- Security and Compliance
- Red Hat OpenStack, Red Hat Satellite, Red Hat CloudForms
- Runtime Applications
- Git: Jenkins Automation
- Service Offering Catalog

This PaaS squad is composed of 18 total resources at a particular point during the provided service. This squad is responsible for all services the container-based platform provides and its catalog.

6.3 Service Management System

The Service Management System (SMS) service squad features the following skills:

- LDAP: Red Hat Enterprise Linux operating system containers
- Python
- Liferay Portal: iTop
- Elasticsearch, Kibana, Grafana, and Logstash
- Site Reliability Engineer
- Snipe-IT: Open Audit and Red Hat Ansible Facts
- Jira
- Prometheus
- Influx DB, MySQL, and Kafka
- Git, Jenkins Automation
- MariaDB
- HTML, Java
- API
- Perl code, Simple Event Correlator
- Red Hat Ansible
- Security and Compliance

SMS is a sophisticated squad that is a full end-to-end product stack squad that delivers the entire SMS service.

The relevance of including an SRE as part of each squad is to recognize each as a service where all critical SLIs, SLOs, and SLAs are evaluated and optimized in their entirety for the required automation. For this reason, these three services do not provide a level 1, 2, and 3 agreement. It is instead replaced with a team working on all elements towards the elimination of more work in the service. A vital principle of the SRE in the squad is fostering the automation and toil elimination to focus on all areas of service improvement of the ITOPS cloud.

A

Red Hat subscription activation process

This appendix discusses how the Red Hat subscriptions activation process works. It includes the following topics:

Subscription activation process

For testing purposes, request an Evaluation/NFR subscription or undergo with purchase of the required Red Hat solution (see this web page), as shown in Figure A-1.

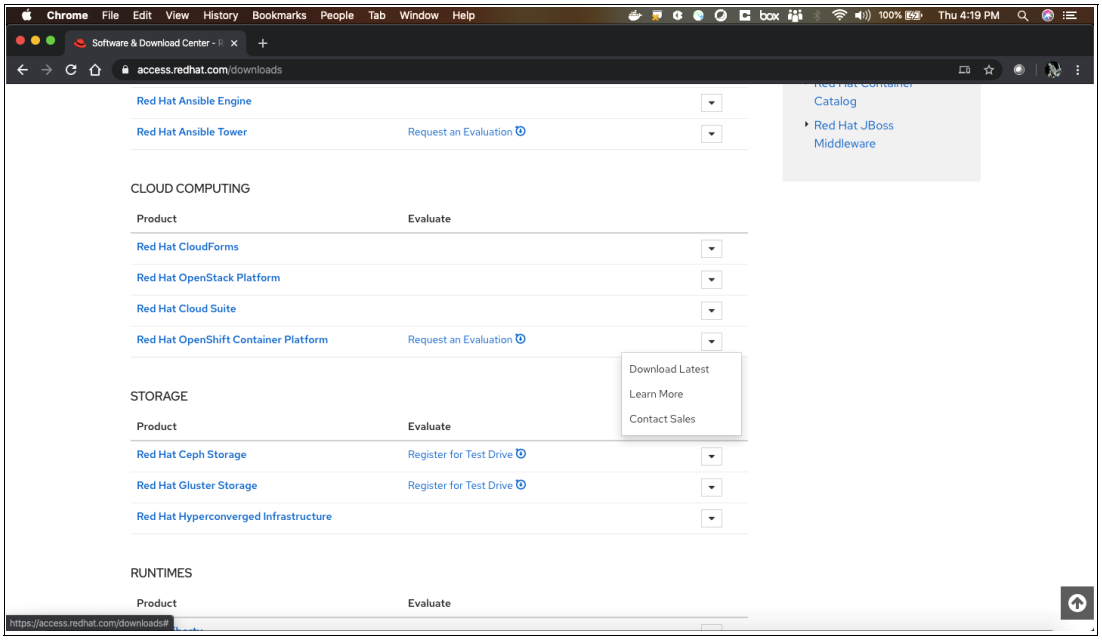


Figure A-1 Subscription Request Portal: Product download page

After the request is successful and approved, you receive an email (see Figure A-2), which includes the activation link and activation serial number attachment.

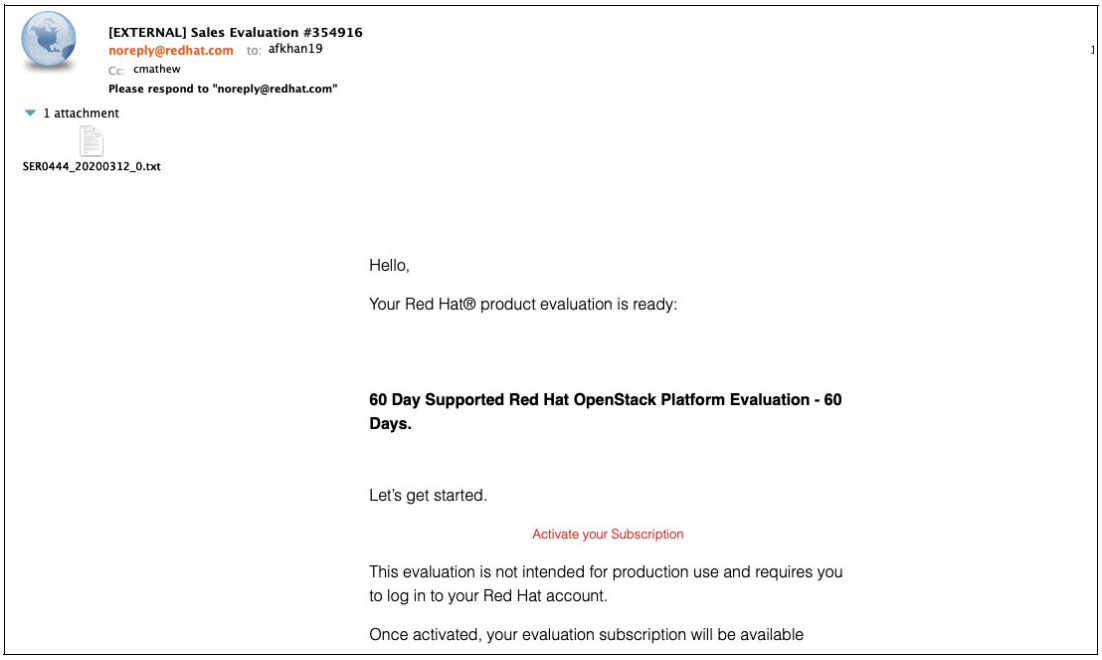


Figure A-2 Activation email

Click the activation link. You are redirected to the Red Hat Customer Portal, as shown in Figure A-3. Log in to the Portal by using the same user ID with which the subscription was requested.

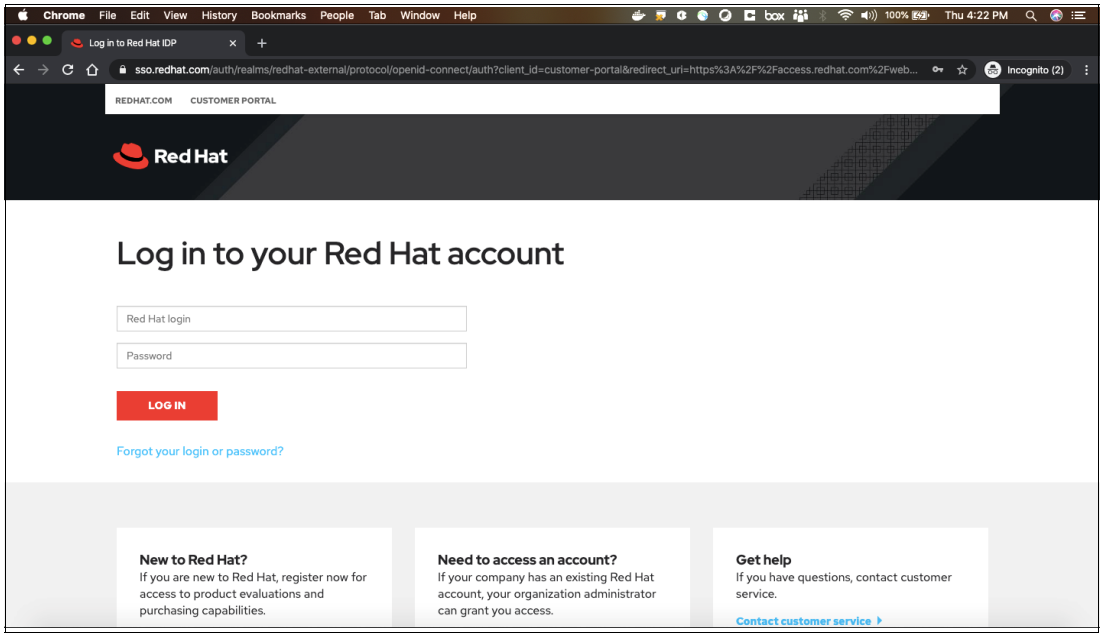


Figure A-3 Red Hat Customer Portal login page

After a successful login, you receive a message in the portal that the subscription was activated, as shown in Figure A-4. Then, you are redirected to the product download page.

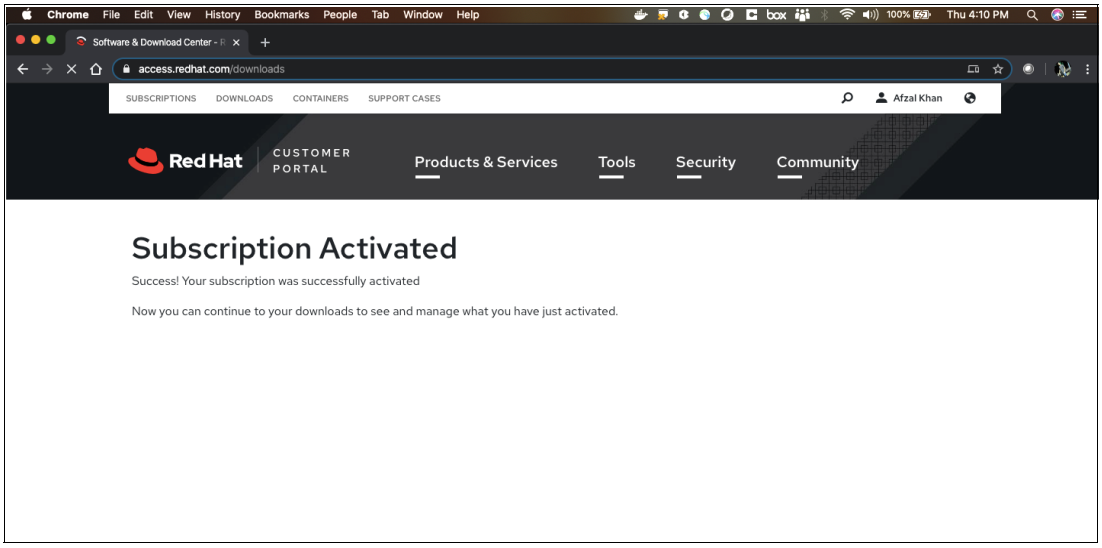


Figure A-4 Subscription Activation Message: Browser

Figure A-5 shows the email that is received with the subscription activated.

Figure A-5 Subscription Activation Message: Email

Verifying activated subscription

Complete the following steps to verify that the subscription was activated from the Red Hat Subscription Portal:

1. At the Red Hat Customer Portal, log in to the portal by clicking **Login**.
2. After successfully login, click **SUBSCRIPTIONS** in the upper left corner of the window, as shown in Figure A-6.

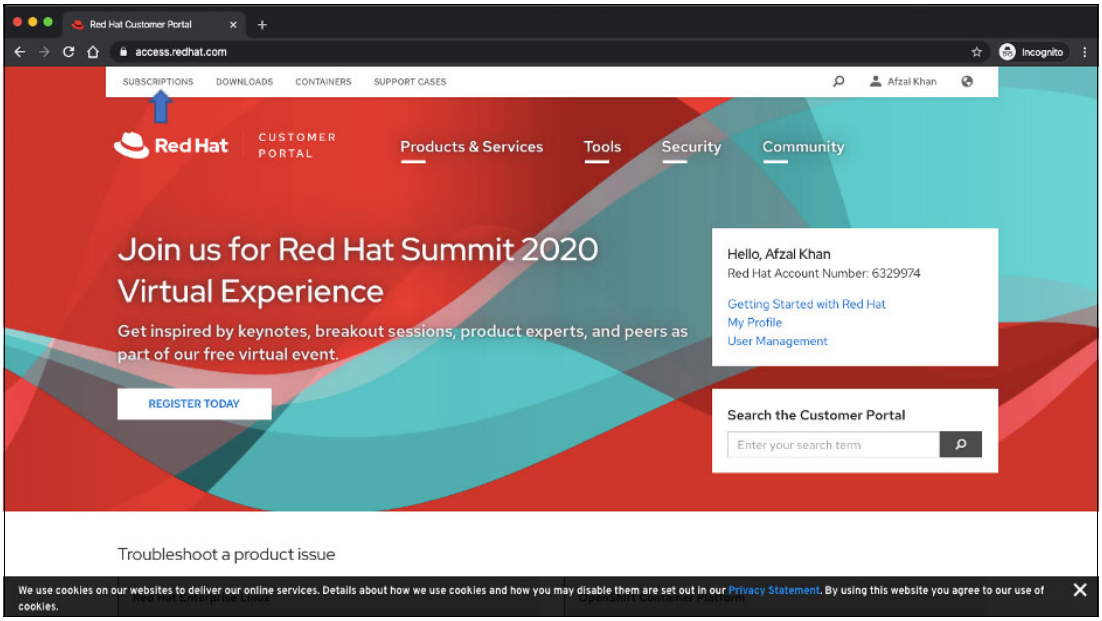


Figure A-6 Subscriptions Manager user portal login

3. The Overview dashboard of the Subscription Manager Portal opens, as shown in Figure A-7. Click **Subscriptions** for more information about active subscriptions that are assigned to your RHN ID.

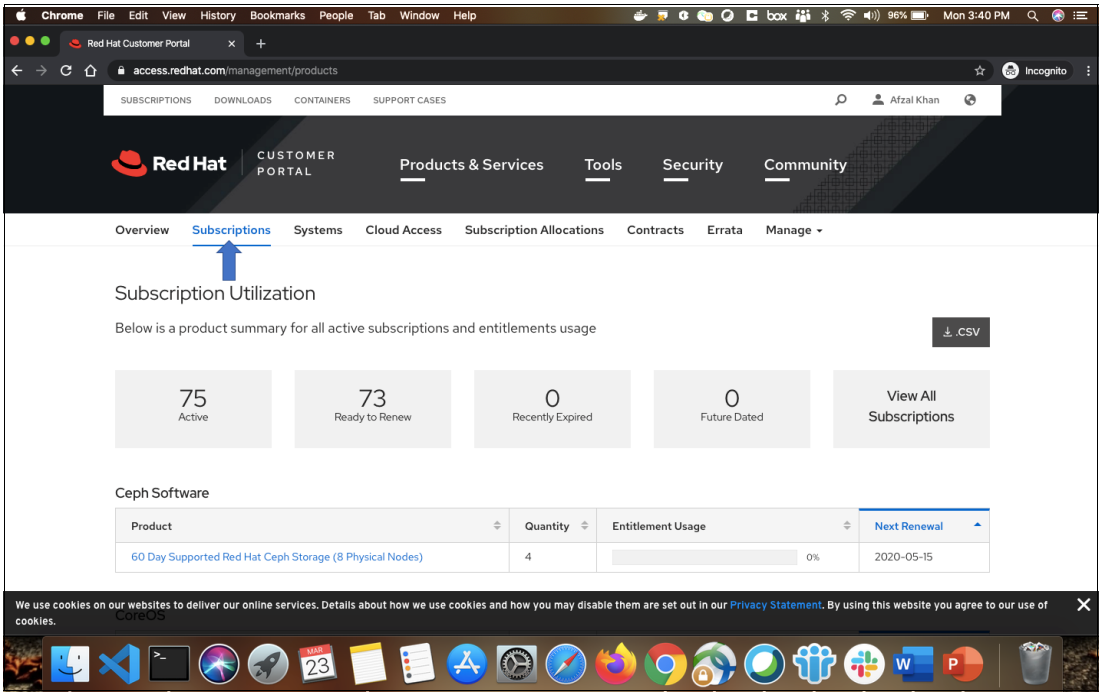


Figure A-7 Subscriptions overview page

4. Click any product for more information about that product, including active subscription state, entitlement usage, subscription identifier, and systems details where this subscription is attached, as shown in Figure A-8.

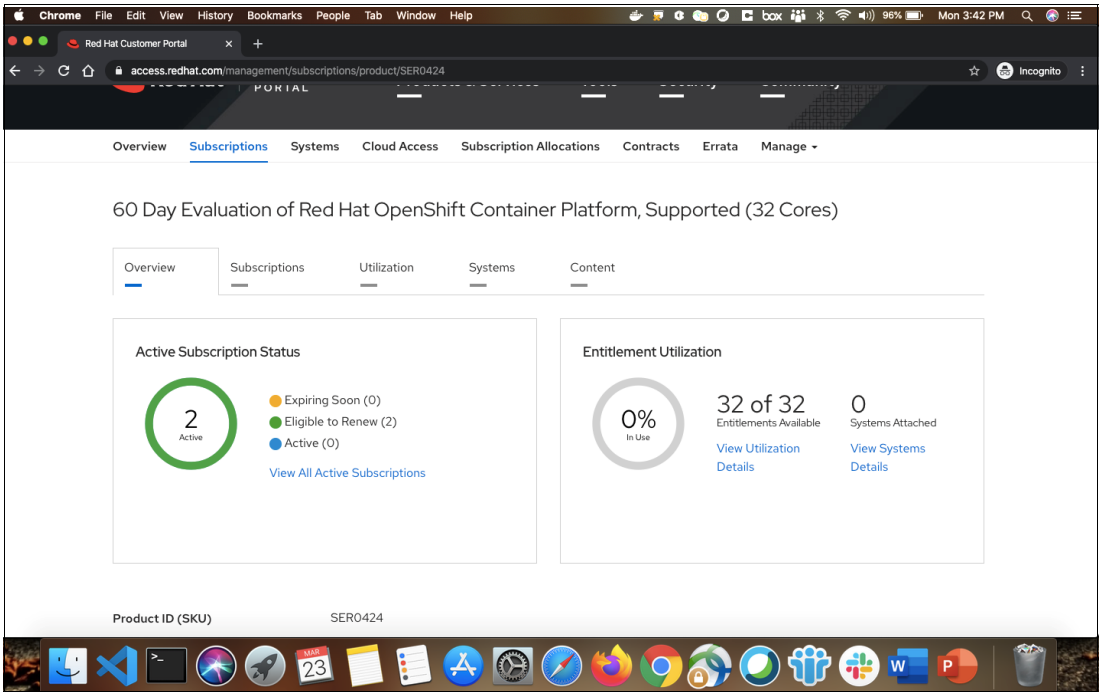


Figure A-8 Subscriptions Entitlement Details

5. Click the **Subscriptions** tab for more information about the subscription, such as pool IDs that are used to subscribe machines. Also, you can download a paper certificate of the subscription as PDF document (see Figure A-9).

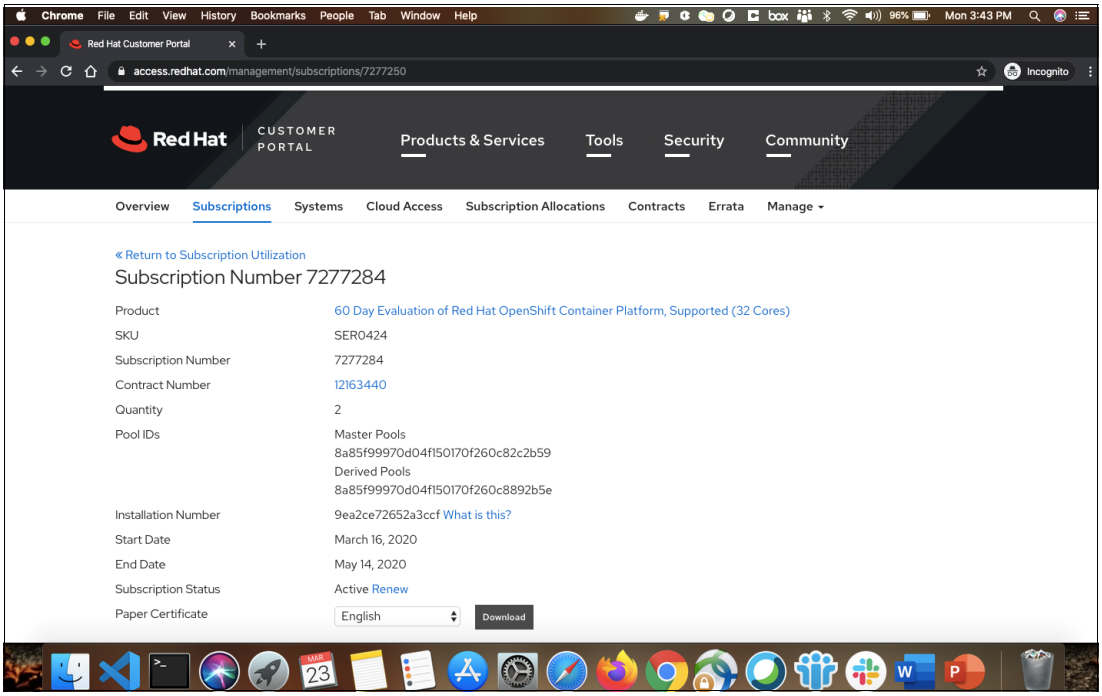


Figure A-9 Specific Subscriptions Number Description

Next, we describe how to subscribe your machine by using Red Hat Subscription Manager Tool (RHSM).

Applying subscription on the Red Hat Enterprise Linux machine

Complete the following steps to subscribe your machine:

1. Open a terminal (macOS or Linux) or PuTTY (windows) on your workstation.
2. SSH to the Red Hat Enterprise Linux server by using valid credentials by running the **$ ssh root@<Server MGMT IP>** command, as shown in Figure A-10.

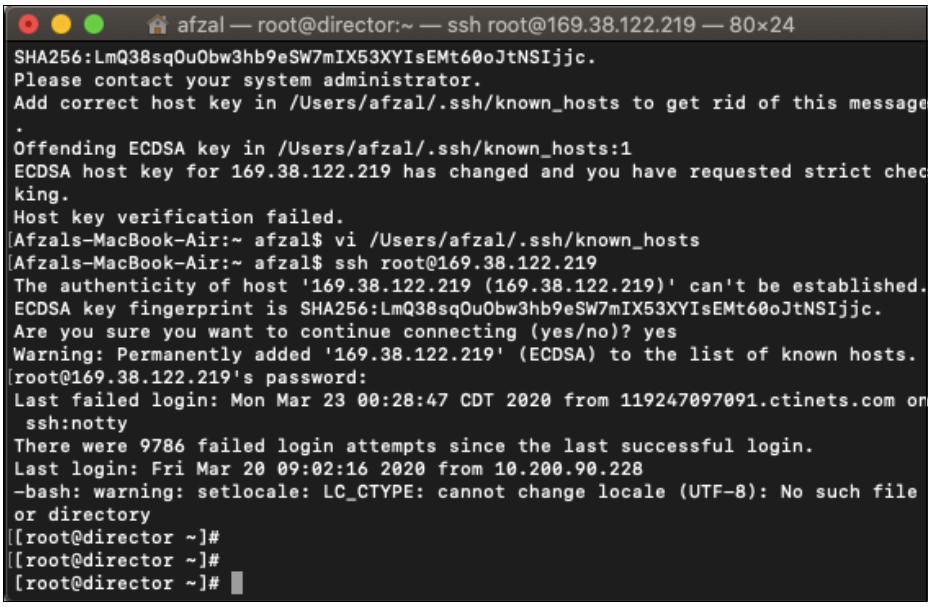


Figure A-10 SSH to Red Hat Enterprise Linux server from terminal

3. Check for the installed products by running the following commands (see Figure A-11 on page 408):

```
[root@director ~]# subscription-manager list
You are attempting to use a locale that is not installed.
+-------------------------------------------+
    Installed Product Status
+-------------------------------------------+
Product Name:   Red Hat Enterprise Linux Server
Product ID:     69
Version:        7.7
Arch:           x86_64
Status:         Unknown
Status Details:
Starts:
Ends:
```

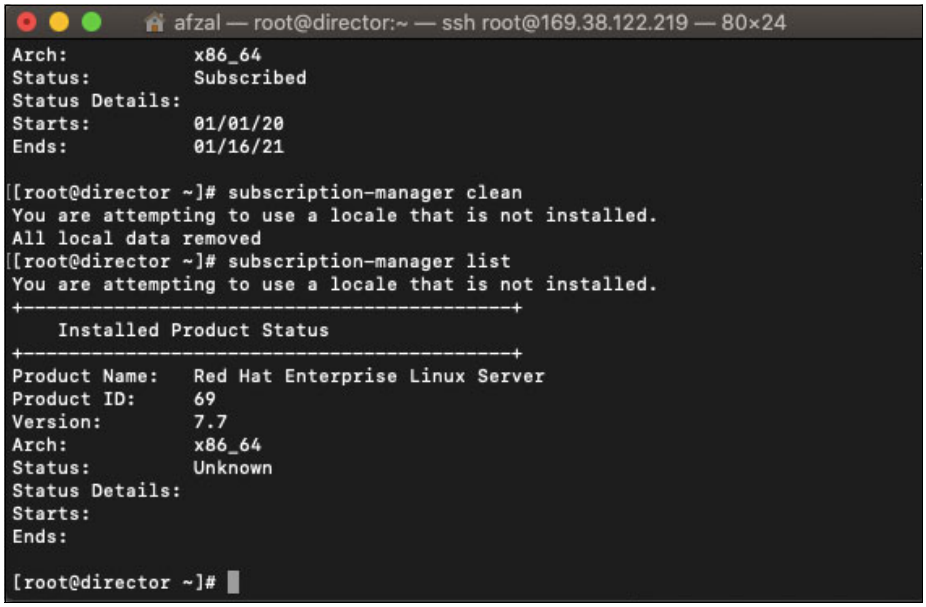


Figure A-11 Checking Installed Product Subscription Status

4. Register the machine with RHSM Server by running the following command (see Figure A-12):

```
[root@director ~]# subscription-manager register
You are attempting to use a locale that is not installed.
Registering to: subscription.rhsm.redhat.com:443/subscription
Username: <RHN Username>
Password:
The system has been registered with ID: 09ca14c4-efdd-4684-aa62-e1705cbef9a3
The registered system name is: director.redbooks.info
```

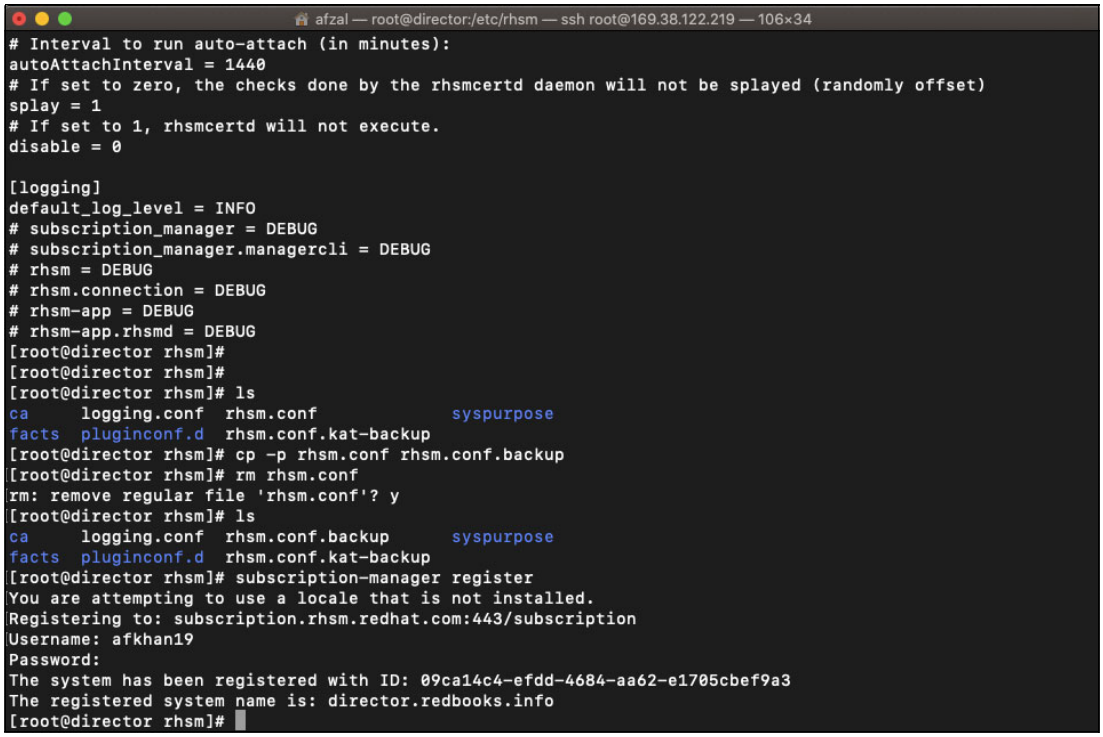


Figure A-12 Registering server with RHSM

5. When the machine is successfully registered to RHSM server, check for the available subscriptions that are attached with your RHN user, as shown in Figure A-13 on page 410. Copy the Pool ID associated subscription for next command input (highlighted in blue) as shown in Example A-1.

Example: A-1 Showing available subscriptions

```
[root@director ~]# subscription-manager list --available --all

You are attempting to use a locale that is not installed.
+-------------------------------------------+
    Available Subscriptions
+-------------------------------------------+
Subscription Name:   60 Day Supported Red Hat OpenStack Platform Evaluation
Provides:            Oracle Java (for RHEL Server)
                     Red Hat Software Collections (for RHEL Server)
                     Red Hat CodeReady Linux Builder for x86_64
                     Red Hat Ansible Engine
                     Red Hat Ceph Storage
                     Red Hat OpenStack
                     Red Hat Enterprise Linux Atomic Host Beta
                     Red Hat Enterprise Linux Fast Datapath
                     Red Hat OpenStack Beta
                     Red Hat CloudForms
                     Red Hat Enterprise Linux Atomic Host
                     Red Hat Enterprise Linux Load Balancer (for RHEL Server)
                     Red Hat Enterprise Linux Fast Datapath Beta for x86_64
                     Red Hat Enterprise Linux Advanced Virtualization Beta
                     Red Hat Beta
                     Red Hat Software Collections Beta (for RHEL Server)
                     Red Hat Enterprise MRG Messaging
                     Red Hat Enterprise Linux Server
                     Red Hat Enterprise Linux for x86_64
                     Red Hat Ceph Storage MON
                     Red Hat Enterprise Linux High Availability for x86_64
                     Red Hat Single Sign-On
                     Red Hat Ceph Storage Calamari
                     Red Hat Enterprise Linux Advanced Virtualization
SKU:                 SER0444
Contract:            12158604
Pool ID:             8a85f99c70d0557f0170d285485278ee
Provides Management: No
Available:           72
Suggested:           1
Service Level:       Premium
Service Type:        L1-L3
Subscription Type:   Standard
Starts:              03/12/20
Ends:                05/11/20
System Type:         Physical
```

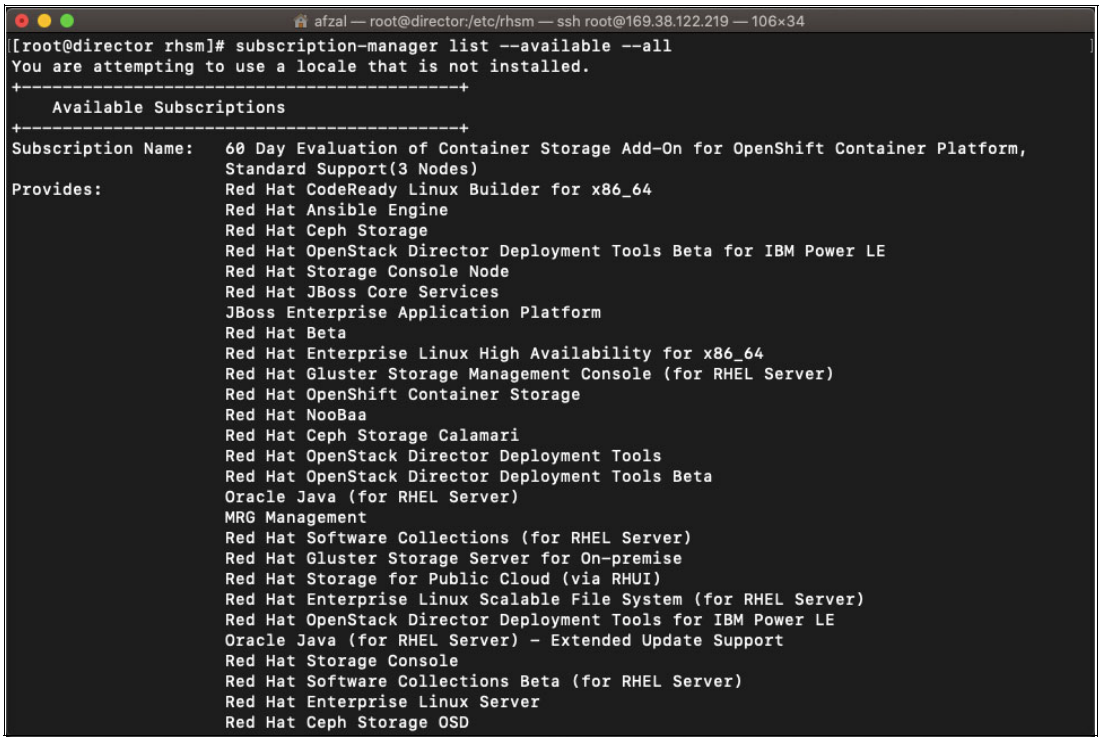


Figure A-13 Checking available subscriptions that can be attached with server

6. Attach the relevant pool to the server with the copied pool ID (see Figure A-14 on page 411). After the pool is attached successfully, you receive the following success message in the command output:

```
[root@director ~]# subscription-manager attach --pool=
8a85f99c70d0557f0170d285485278ee
You are attempting to use a locale that is not installed.
Successfully attached a subscription for: 60 Day Supported Red Hat OpenStack
Platform Evaluation
```

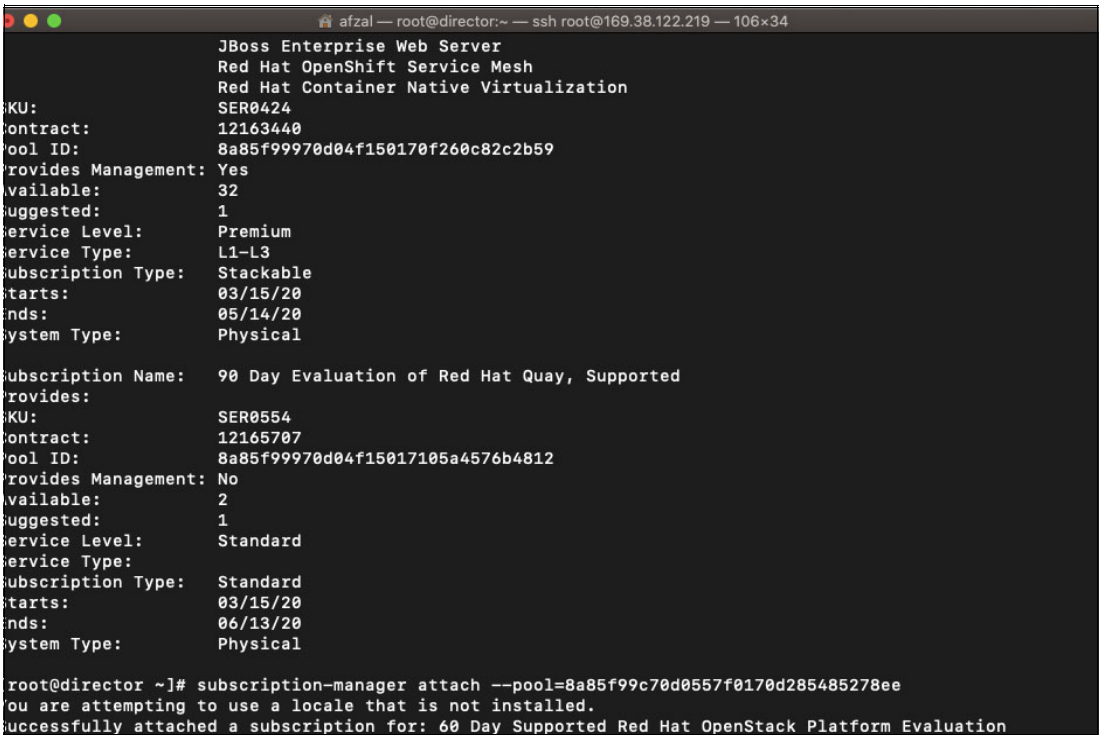


Figure A-14 Attaching specific pool that is required for product installation

7. Verify that the required pool is attached successfully (see Figure A-15 on page 412) by running the command as shown in Example A-2.

Example: A-2 Verify attached pool

```
[root@director ~]# subscription-manager list

You are attempting to use a locale that is not installed.
+-------------------------------------------+
    Installed Product Status
+-------------------------------------------+
Product Name:   Red Hat OpenStack
Product ID:     191
Version:        13.0
Arch:           x86_64
Status:         Subscribed
Status Details:
Starts:         03/12/20
Ends:           05/11/20

Product Name:   Red Hat Enterprise Linux Server
Product ID:     69
Version:        7.7
Arch:           x86_64
Status:         Subscribed
Status Details:
Starts:         03/12/20
Ends:           05/11/20
```

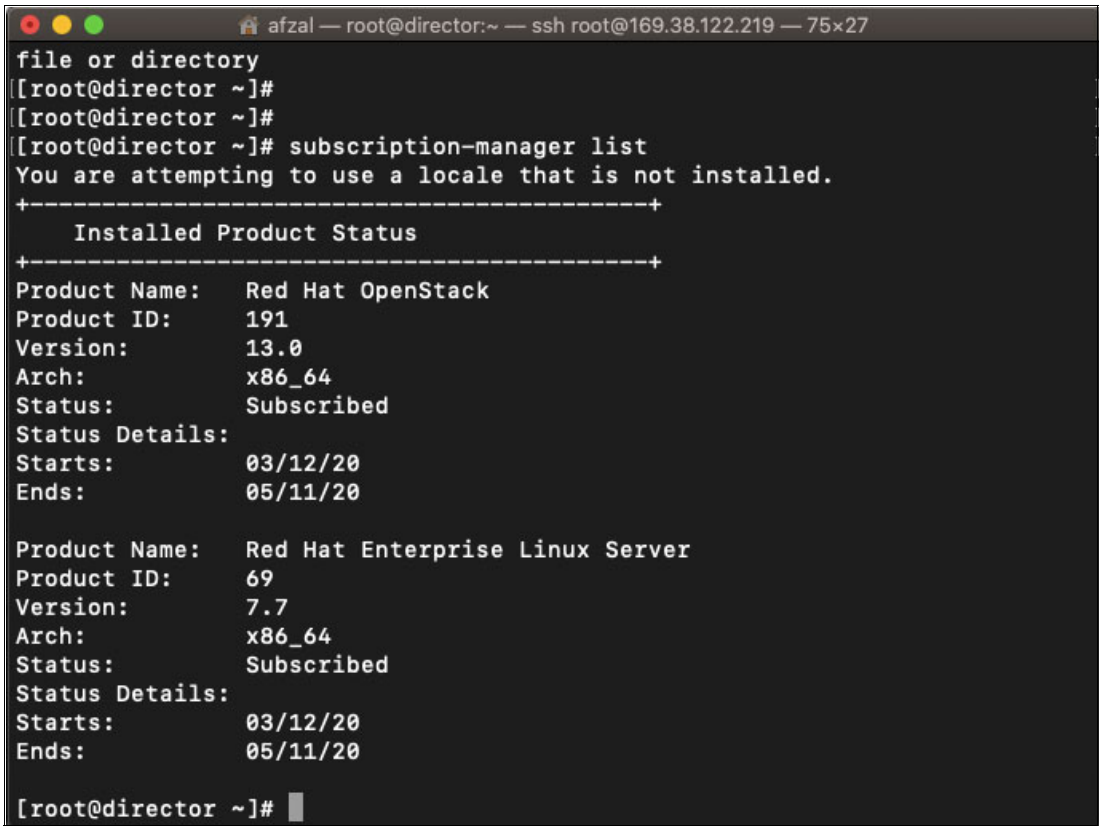


Figure A-15 Verifying successful pool attachment

B

Operations and executive dashboards in Grafana

This appendix discusses dashboard operations with Grafana and includes the following topics:

Dashboards

Grafana features options to connect to various data sources. After the matrixes are stored in the data source, it becomes available to build the dashboards.

For more information about how to import a dashboard by using JSON, see 4.3.7, “Adding SLA dashboard” on page 301.

In this section, we describe how to create a dashboard.

Creating a dashboard

Complete the following steps:

1. In the Grafana Welcome window, click **Home**. All of the stared dashboards and recently accessed dashboards are shown (see Figure B-1).

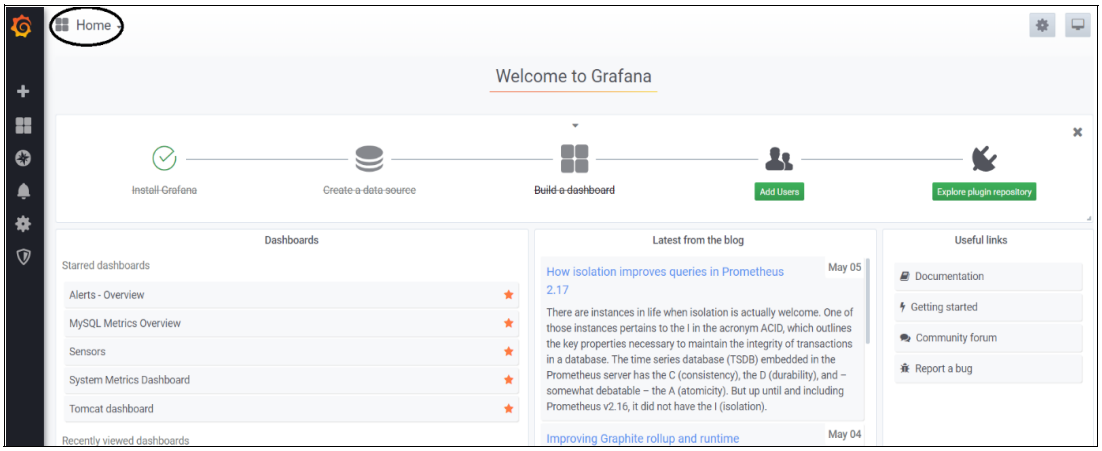


Figure B-1 Grafana welcome window

2. Click **New dashboard** to create a dashboard, as shown in Figure B-2.

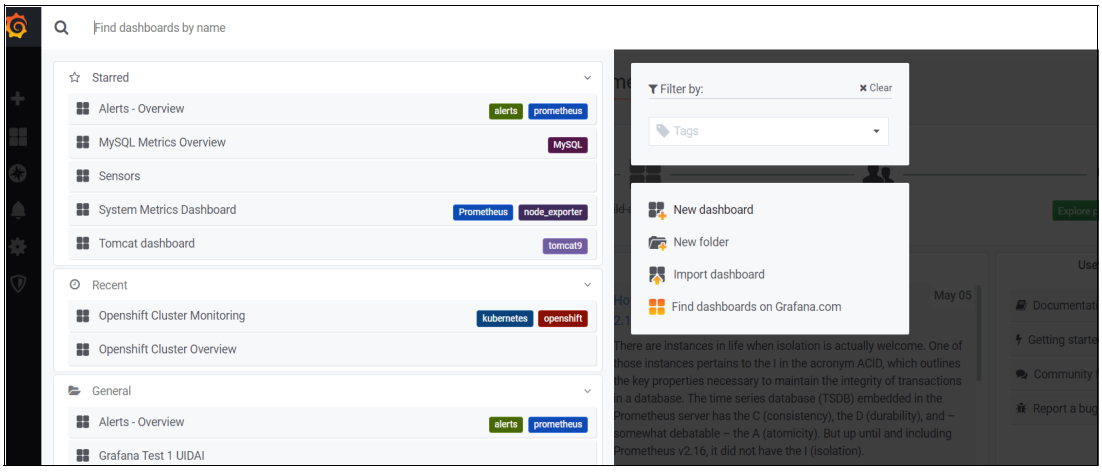


Figure B-2 Grafana dashboard window

The new dashboard is shown in Figure B-3.

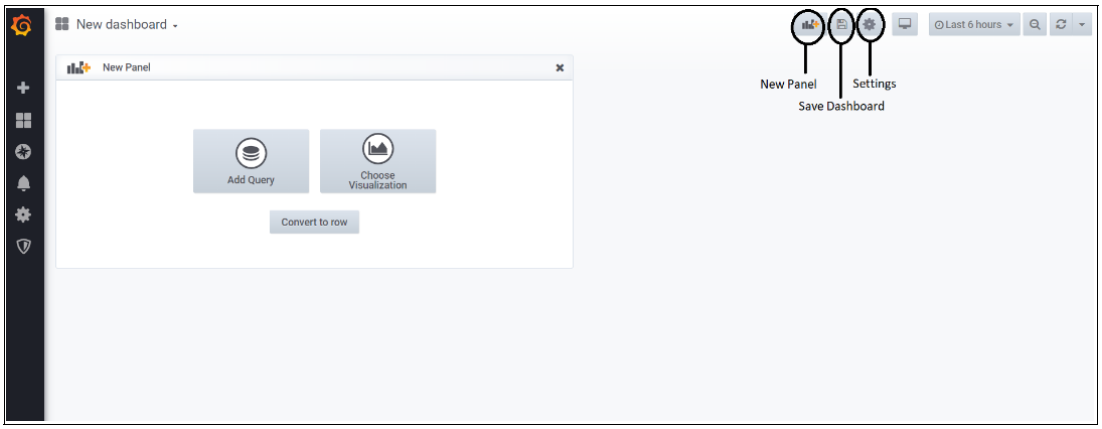


Figure B-3 Grafana new dashboard window

3. In the dashboard window (see Figure B-3), click **Settings** and the window that is shown in Figure B-4 shows all of the settings that are associated with the dashboard. The name, description, and other options can be modified in this window.

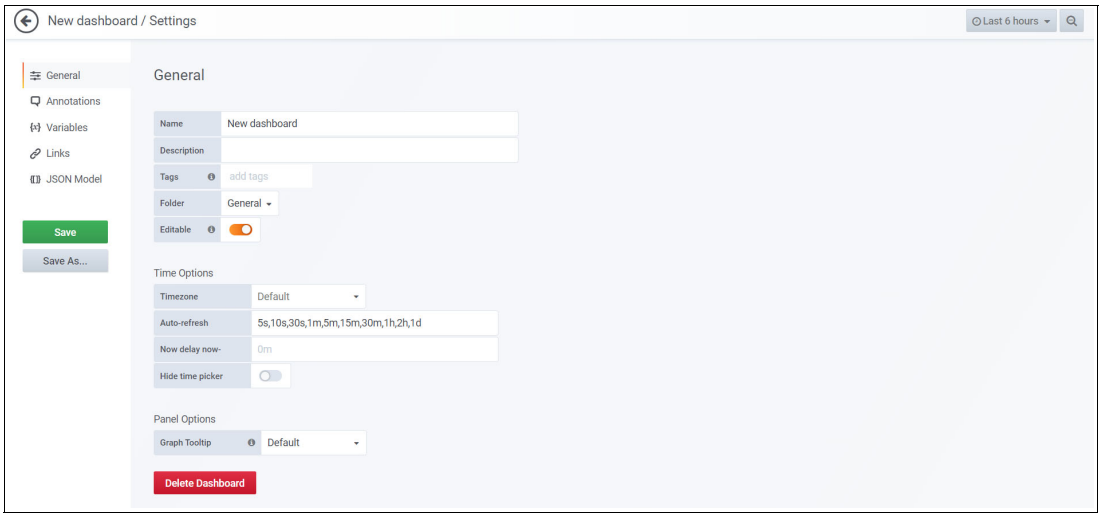


Figure B-4 Grafana New dashboard Settings window

In the settings page (see Figure B-4), an option is available to store variables that can be used to store data that you need to use repeatedly in the dashboard or queries. These variables can be used to build the drop-down in a dashboard.

A dashboard consists of multiple panes and when a dashboard is created, an empty pane is added by default. A dashboard can be divided into several sections by creating rows.

4. In a new pane, click **Convert to row**. A section is created, and under that section, several panes can be added, as shown in Figure B-5.

Figure B-5 Grafana New dashboard window

5. Click **Add Query** and the window refreshes (see Figure B-6).

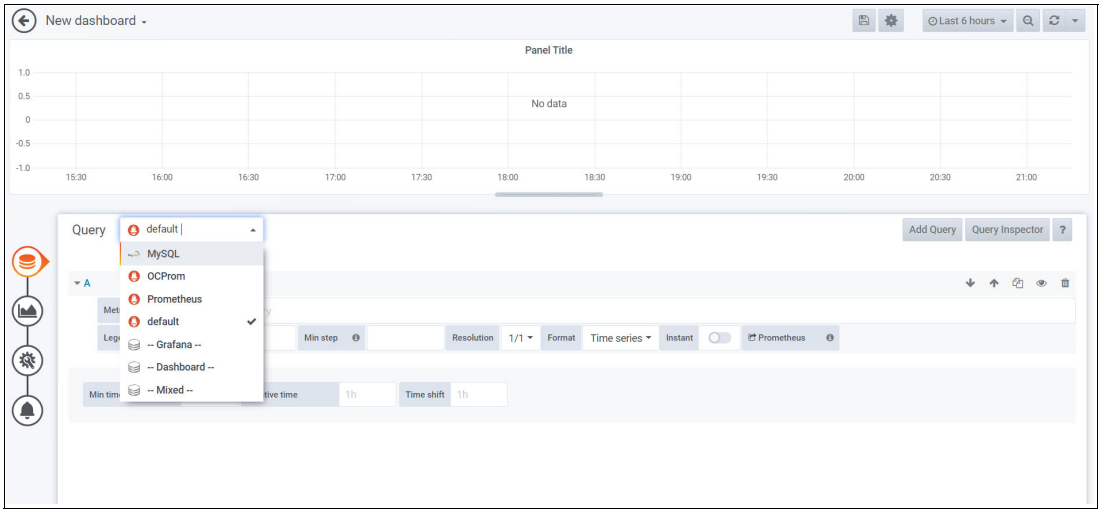


Figure B-6 Grafana New dashboard: Query window

6. In this section of the window, the query can be used to fetch data and metrics and display them based on the type of visualization. By default, graph visualization is selected.

   Click **Query** to display all data sources that were added to Grafana. By default, data source is selected, which is set in Grafana data sources settings.

Based on the data source, the query language changes and the data can be fetched to display as shown in Figure B-7.

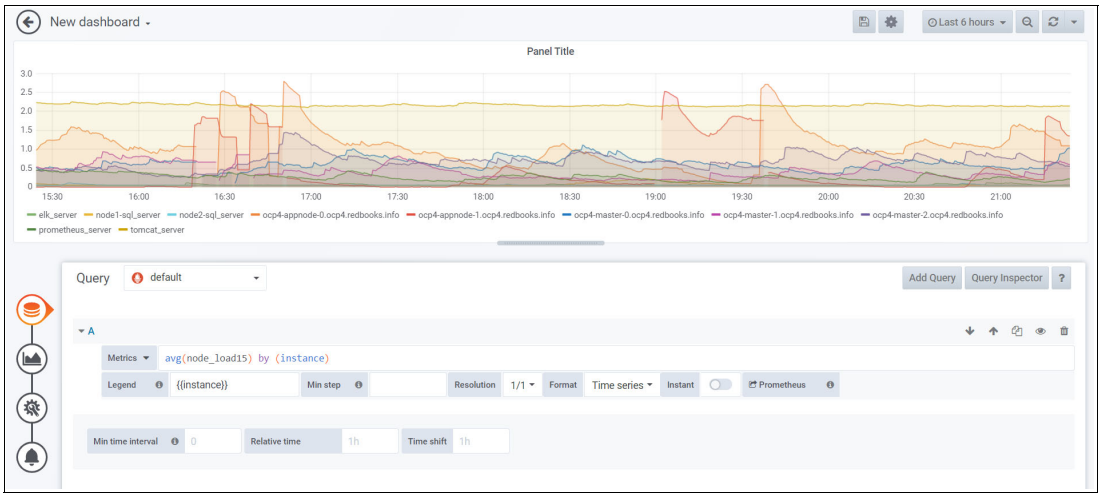


Figure B-7 Grafana graphics visualization window

Metrics can be selected by clicking the drop-down with Metrics name or a custom query can be added to display the plots. Several options are available here to along with the query, such as the legend, which can be used to name each line.

A query can be enabled or disabled, moved up or down, or copied. Also, new queries can be added (by clicking **Add Query)** and queries can be inspected (by clicking **Query Inspector**). All of the queries and plots are drawn on the same plane, which allows users to compare data. Similarly, various options are available in visualization for each type of visualization that can be used to make the plot more appealing.

7. Select **Choose Visualization** and the page refreshes, as shown in Figure B-8.

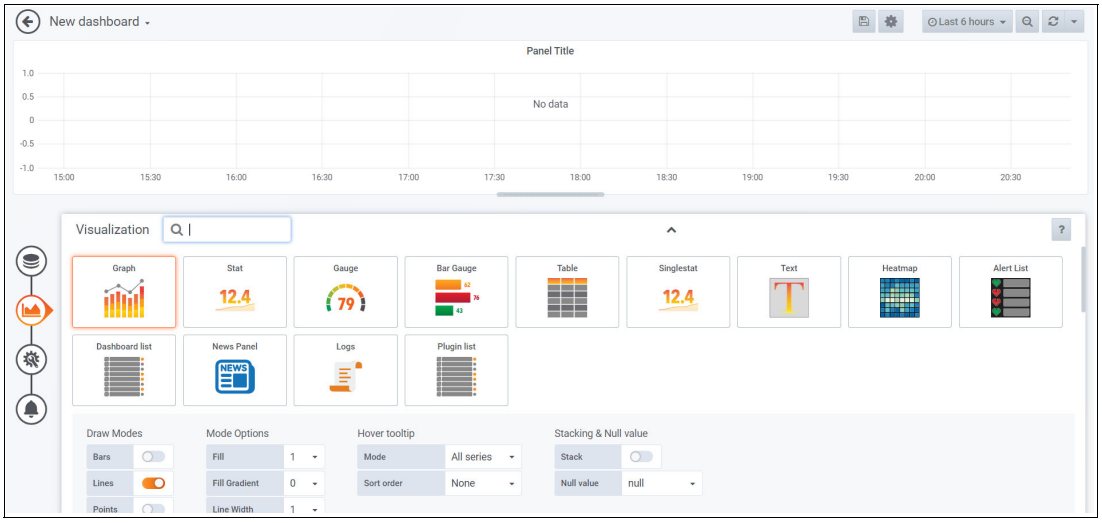


Figure B-8 Grafana New dashboard: Visualization window

Both panes are part of one function; for example, for creating a plot. A query can be required to fetch data and a visualization can be required to show the data. The query or visualization can be switched by using the options that are shown in Figure B-9.

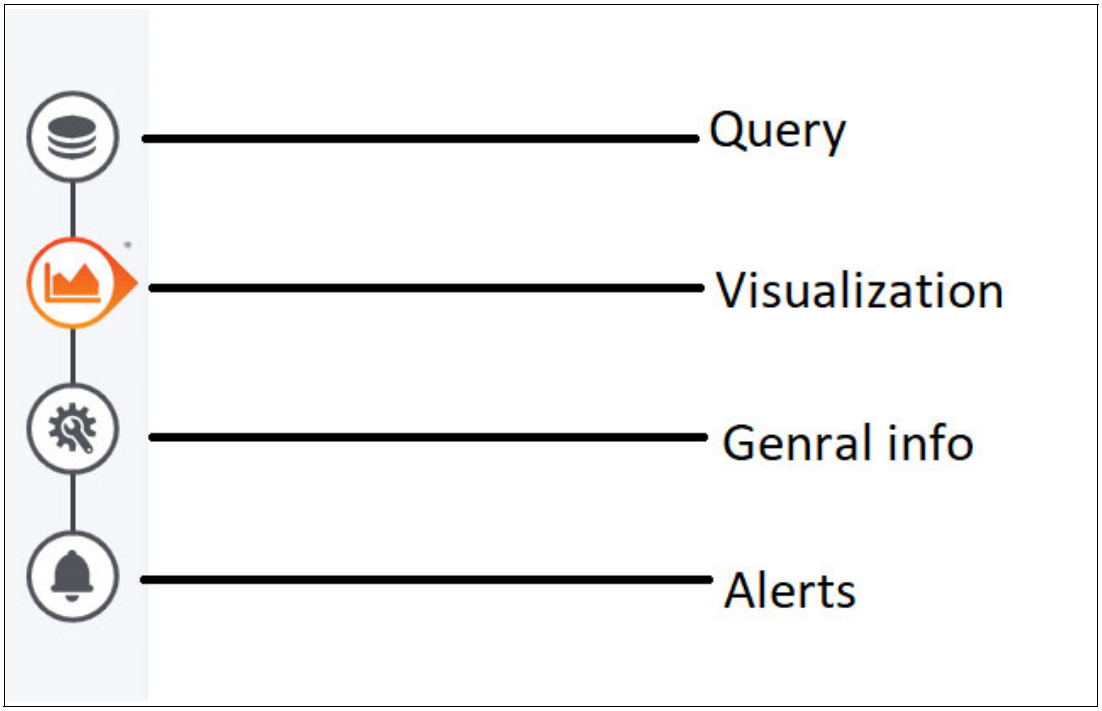


Figure B-9 Visualization query options

When multiple panes are added, they create a dashboard. When data is added, the windows looks as shown in the example in Figure B-10.

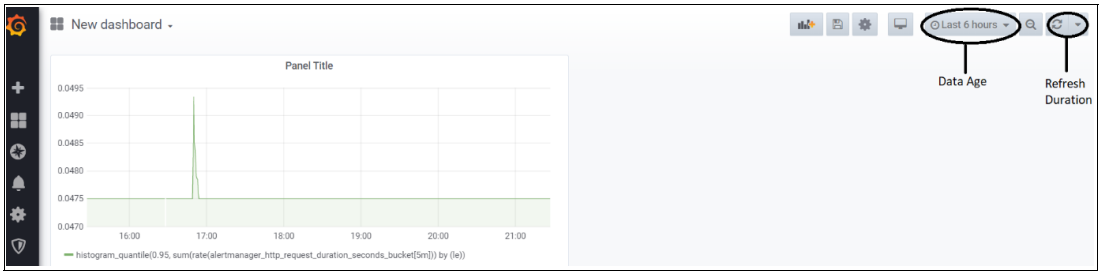


Figure B-10 Grafana New dashboard: Data Age and Refresh Duration options

8. Select **Data Age** to select the amount of historical data that is displayed, and **Refresh Duration** to refresh the incoming data. The refresh duration data is fetched from settings in the dashboard.

Grafana shows how the data is behaving in real time along with comparison with immediate past behavior.

Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this book.

Online resources

The following websites are also relevant as further information sources:

- Red Hat Customer Portal - Product Documentation:

  https://access.redhat.com/documentation/en-us/
- Red Hat Ansible Tower Quick Installation Guide V3.7.1:

  https://docs.ansible.com/ansible-tower/latest/html/quickinstall/index.html
- Red Hat OpenShift Container Platform 4.3 Documentation:

  https://docs.openshift.com/container-platform/4.3/welcome/index.html
- Product Documentation for Red Hat OpenShift Container Platform 4.3:

  https://access.redhat.com/documentation/en-us/openshift_container_platform/4.3/
- Product Documentation for Red Hat CloudForms 5.0:

  https://access.redhat.com/documentation/en-us/red_hat_cloudforms/5.0/
- Product Documentation for Red Hat Satellite 6.7:

  https://access.redhat.com/documentation/en-us/red_hat_satellite/6.7/
- Product Documentation for Red Hat Quay 3.3:

  https://access.redhat.com/documentation/en-us/red_hat_quay/3.3/
- How to register and subscribe a system to the Red Hat Customer Portal using Red Hat Subscription-Manager:

  https://access.redhat.com/solutions/253273
- Using and Configuring Red Hat Subscription Manager:

  https://red.ht/3bNbyQm
- Red Hat Ceph Storage Installation Guide:

  https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/4/html/installation_guide/index
- Red Hat Ceph Storage Architecture Guide:

  https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/4/html/architecture_guide/index
- Red Hat Ceph Storage Configuration Guide:

  https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/4/html/configuration_guide/index
- Red Hat Ceph Storage Commands Cheat Sheet:

  https://access.redhat.com/articles/3250281

- Red Hat Ceph Hardware Guide:

  https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/4/pdf/hardware_guide/Red_Hat_Ceph_Storage-4-Hardware_Guide-en-US.pdf
- Red Hat Ceph Compatibility Matrix:

  https://access.redhat.com/articles/1548993
- Red Hat Ceph Architecture Guide:

  https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/4/pdf/architecture_guide/Red_Hat_Ceph_Storage-4-Architecture_Guide-en-US.pdf
- Red Hat Ceph 4.0 Documentation:

  https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/4/

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

Software Defined Data Center with Red Hat Cloud and Open

SG24-8473-00

ISBN 0738459151

IBM

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

SG24-8473-00

ISBN 0738459151

Printed in U.S.A.