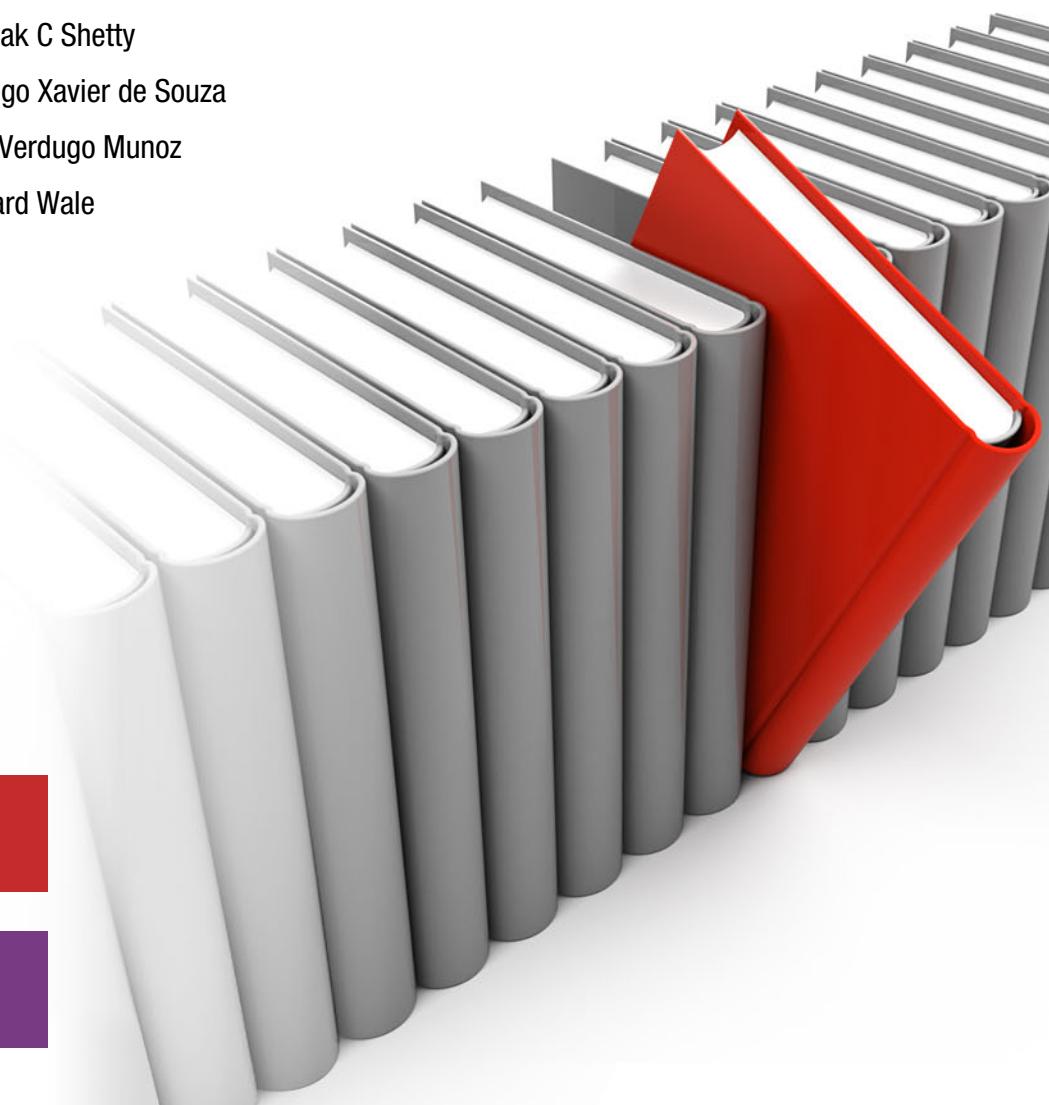


Red Hat OpenShift V4.X and IBM Cloud Pak on IBM Power Systems Volume 2

Dino Quintero
Jose Martin Abeleira
Marcelo Avalos
Agustin Barreto
Paul Chapman
Michael Easlon
Luis Ferreira
Gualberto Esteban Ferrer
Alain Fisher
Federico Fros
Miguel Gomez Gonzalez
Felix Gonzalez Sugasti
Abhishek Jain
Paulo Sergio Lemes
Gabriel Padilla Jimenez

Sudipto Pal
Bogdan Savu
Deepak C Shetty
Rodrigo Xavier de Souza
Alan Verdugo Munoz
Richard Wale



 Cloud

Power Systems

IBM
®

Redbooks



IBM Redbooks

Red Hat OpenShift V4.X and IBM Cloud Pak on IBM Power Systems Volume 2

April 2021

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

First Edition (April 2021)

This edition applies to:

Red Hat® OpenShift V4.5.4
Red Hat CoreOS V4.5.4
Red Hat Enterprise Linux V8.2
Terraform V0.12.29
IBM Cloud Shell V1.0.4
IBM Cloud Pak for Multicloud Management V1.3. Enterprise
Terraform V0.12.21
IBM Cloud Terraform Provider V1.12
Terraform Provider OpenStack V1.17.0
IBM i V7.3. RS730-K
IBM i Access Client Solutions V1.1.8.0
IBM PowerVC Standard Edition V1.4.4.1

© Copyright International Business Machines Corporation 2021. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
Preface	ix
Authors	ix
Now you can become a published author, too!	xiii
Comments welcome	xiv
Stay connected to IBM Redbooks	xiv
Chapter 1. Introduction.....	1
1.1 Introduction	2
1.2 Scope	2
Chapter 2. Reference architecture.....	3
2.1 A typical reference architecture for Red Hat OpenShift	4
2.2 Storage requirements	6
2.2.1 Container-ready storage	7
2.2.2 Container-native storage	8
2.2.3 Container storage use cases	10
2.2.4 Container storage on Power Systems servers	11
2.2.5 IBM Power Virtualization Center Container Storage Interface driver	12
2.2.6 IBM Spectrum Scale Container Native Storage	12
2.2.7 Red Hat OpenShift Container Storage	13
2.2.8 IBM Storage Suite for IBM Cloud Paks.....	13
Chapter 3. IBM Cloud Paks	15
3.1 IBM Cloud Paks	16
3.2 IBM Cloud Paks offering	17
3.2.1 Benefits of IBM Cloud Paks	17
3.2.2 IBM Cloud Pak for Application	18
3.2.3 IBM Cloud Pak for Data	18
3.2.4 IBM Cloud Pak for Integration	18
3.2.5 IBM Cloud Pak for Automation	19
3.2.6 IBM Cloud Pak for Multicloud Management	19
3.3 IBM Cloud Pak for Data	19
3.3.1 Key features of IBM Cloud Pak for Data	19
3.3.2 IBM Cloud Pak for Data features	20
3.3.3 IBM Cloud Pak for Data: Journey with data	21
3.4 IBM Cloud Pak for Multicloud Management	26
3.4.1 Overview	26
3.4.2 Multicloud management fundamentals	27
3.4.3 Component view of IBM Cloud Pak for Multicloud Management	28
3.4.4 IBM Cloud Pak for Multicloud Management: Modules and features	34
3.5 Use cases: IBM Cloud Pak for Data	34
3.5.1 Automated AI lifecycle: Organize and analyze data	34
Chapter 4. Solution scenarios	39
4.1 Installing Red Hat OpenShift in an OpenStack environment	40
4.1.1 Deploying by using an OpenStack Heat template.....	40

4.1.2 Deploying by using the Red Hat OpenShift installer	43
4.1.3 Deploying by using Terraform	44
4.2 Banking architecture use case scenarios	44
4.2.1 Requirements and challenges	44
4.2.2 Private cloud, public cloud, or hybrid cloud for banking	45
4.2.3 Modernization and cloud journey for Power Systems	46
4.2.4 Benefits of open hybrid cloud for banking	47
4.2.5 Use cases	47
Chapter 5. Red Hat OpenShift V4.x on IBM Power Systems cluster administration and monitoring	53
5.1 Logging with Cluster Logging Operator	54
5.1.1 Deploying cluster logging	54
5.2 Monitoring with Prometheus	55
5.2.1 Configuration	56
5.3 Monitoring the Hardware Management Console Performance Counter Monitoring data collection in Grafana dashboards	56
5.3.1 How it works	56
5.3.2 Deploying components	57
5.3.3 Configuring the Grafana interface	67
Chapter 6. Deployment scenarios	73
6.1 Introducing deployment scenarios	74
6.2 Deploying Red Hat OpenShift Container Platform V4.5 in an IBM Power Systems Virtual Server on IBM Cloud	75
6.2.1 IBM Power Systems Virtual Server on IBM Cloud	75
6.2.2 Getting started with deployment in IBM Power Systems Virtual Server	76
6.2.3 Prerequisites	78
6.2.4 Setting up the IBM Cloud environment	79
6.2.5 Setting up the deployment host	84
6.2.6 Deploying the Red Hat OpenShift Container Platform	86
6.3 Bare metal deployment by using Red Hat Enterprise Linux CoreOS LPARs on a Virtual I/O Server	91
6.4 Installing Red Hat OpenShift V4.3 and V4.4: Tips and tricks	93
6.4.1 Preparing your environment	93
6.4.2 PowerVM configuration for network installation	94
6.4.3 Configuring the DHCP server	95
6.4.4 Starting the servers to install Red Hat Enterprise Linux CoreOS	97
6.4.5 Installing from ISO instead of tftp	99
6.4.6 Configuring the DNS server	102
6.4.7 Configuring the load balancer	103
6.4.8 Creating the SSH key	105
6.4.9 Installation process	106
6.4.10 Checking the installation	109
6.4.11 Backing up your cluster	112
Appendix A. IBM Power Systems in the hybrid cloud world	115
IBM Power Systems: Journey to hybrid cloud	116
Introducing Red Hat Ansible for Power Systems	117
Overview	117
What is Red Hat Ansible	118
Why Red Hat Ansible on Red Hat OpenShift	118
Red Hat Ansible architecture	119
Red Hat Ansible automation with IBM Power Systems	120

IBM Cloud Automation Manager and Red Hat Ansible.....	120
Introducing Red Hat Ansible for IBM i	120
Overview.....	120
Red Hat Ansible Collections for IBM i.....	122
IBM i specific Red Hat Ansible modules	123
Installing Red Hat Ansible for IBM i	124
Running your first Red Hat Ansible example for IBM i	126
Running from the CLI	127
Running from Red Hat Ansible Tower.....	128
Common use cases and integrated examples	128
Sample labs: Deploying IBM virtual machines into a hybrid cloud.....	129
Deploying an IBM virtual machine by using IBM Cloud Pak for Multicloud Management in IBM PowerVC	129
Overview.....	129
Configuring a cloud connection.....	130
Creating a template for IBM i deployment.....	131
Moving a VM to IBM Cloud Object Storage in IBM Cloud from IBM PowerVC	146
Overview.....	146
Capturing a virtual machine from IBM PowerVC.....	146
Configuring IBM Cloud Object Storage in IBM Cloud.....	149
Deploying IBM VM into IBM Power Systems Virtual Server in IBM Cloud by using IBM Cloud Pak for Multicloud Management.....	155
Overview.....	155
Creating an IBM Power Systems Virtual Server service.....	156
Generating an IBM Cloud IAM token by using an API key	158
Creating a public and private key to connect to an IBM Power Systems Virtual Server instance.....	159
Importing a boot image into IBM Power Systems Virtual Server from IBM Cloud Object Storage	161
Creating a template to deploy a virtual machine into an IBM Power Systems Virtual Server instance.....	164
Deploying a virtual machine into IBM Power Systems Virtual Server by using IBM Cloud Pak for Multicloud Management	174
Summary.....	186
Appendix B. Red Hat OpenShift and Kubernetes for the developer	187
Red Hat OpenShift from the developer standpoint	188
The odo CLI	188
Installing odo.....	188
Creating an application.....	189
Modifying an application	193
Cleaning up	194
Conclusion	195
Appendix C. Red Hat OpenShift and IBM Cloud Paks entitlements	197
What you get with a Red Hat OpenShift V4.x subscription	198
What you get with an IBM Cloud Paks license.....	199
Appendix D. Virtual I/O Server and Shared Ethernet Adapter configuration	201
Configuration steps	202
Appendix E. Where to start with Red Hat OpenShift on IBM Power Systems	205
Red Hat OpenShift on IBM Power Systems support	206
Red Hat support information	206

Glossary	207
Related publications	209
IBM Redbooks	209
Online resources	209
Help from IBM	210

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.

AIX®	IBM Spectrum®	PowerVM®
Aspera®	IBM Watson®	PureData®
Cognos®	IBM Z®	Redbooks®
DataStage®	InfoSphere®	Redbooks (logo)  ®
Db2®	Netezza®	SPSS®
Global Business Services®	OpenScale™	SystemMirror®
IBM®	Power Architecture®	Tivoli®
IBM Cloud®	POWER8®	Watson OpenScale™
IBM Cloud Pak®	POWER9™	WebSphere®
IBM Elastic Storage®	PowerHA®	XIV®
IBM Garage™	PowerPC®	z/OS®

The following terms are trademarks of other companies:

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

RStudio, and the RStudio logo are registered trademarks of RStudio, Inc.

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 educates and prepares its readers to understand and enter the multicloud era.

This book takes you on a journey to understand the multicloud concept. This book provides a snapshot of the IBM Cloud® Solution, and it uses generally available and supported software and hardware resources to help you understand the following items:

- ▶ The rationale and methodology of this publication.
- ▶ New concepts and terminology.
- ▶ Reasons for moving native workloads to the cloud.
- ▶ Red Hat OpenShift V4.x on Power Systems.
- ▶ Reasons for using IBM Power Systems for moving to the cloud.
- ▶ Reference architecture for Red Hat OpenShift on IBM Power Systems.
- ▶ Installation planning, considerations, and guidelines to help provide an optimal system configuration and implementation.
- ▶ Implementation details for IBM Cloud Pak® for Multicloud Management, and IBM Cloud Pak for Data.
- ▶ Use cases to complement the theoretical concepts.

The goal of this publication is to describe the journey to implement an IBM Cloud Solution by using Red Hat OpenShift and IBM Cloud Paks on IBM Power Systems and theoretical knowledge to learn the concepts, perform hands-on exercises to practice the theory, and document these findings in sample scenarios.

The book addresses topics for developers, IT architects, IT specialists, sellers, and anyone looking to implement Red Hat OpenShift and IBM Cloud Paks on IBM Power Systems. Moreover, this book provides technical content to transfer how-to-skills to the support teams, and solution guidance to the sales team. This book complements the documentation that is available at IBM Knowledge Center and aligns with the educational offerings that are provided by IBM Systems Education.

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 (AI) 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.

Jose Martin Abeleira is a Senior Systems/Storage Administrator at DGI Uruguay. He worked for IBM, and has written Redbooks publications. He is a Certified Consulting IT Specialist, and does IBM Certified Systems Expert Enterprise Technical Support for IBM AIX® and Linux in Montevideo, Uruguay. He worked with IBM for 8 years and has 16 years of AIX experience. He holds an Information Systems degree from Universidad Ort Uruguay. His areas of expertise include Power Systems, AIX, UNIX, and LINUX, Live Partition Mobility (LPM), IBM PowerHA® SystemMirror®, SAN and Storage on the IBM DS line, IBM V7000, HITACHI HUSVM, and G200/G400. He also teaches Infrastructure Administration for the joint venture between Universidad de la Republica (UDELAR), Universidad del trabajo del Uruguay (UTU) and Universidad Tecnologica (UTEC).

Marcelo Avalos is an Infrastructure Architect for IBM i who is based at IBM Uruguay. Before working for IBM, he worked DATEC LTDA (an IBM Business Partner). During his 7+ years in IT, he has held leadership roles and worked on IBM customer accounts, critical situations, decision-making capability. He has experience delivering IBM semiconductors, IBM Power Systems, mid-range and high-end storage systems, and system software. He has worked in the banking industry in South America. His current work is hybrid cloud that is aligned to IBM strategy. He received his Electronic Systems Engineering degree from Escuela Militar de Ingeniería in Bolivia. He also holds a Postgraduate Coursework of Higher Education Teaching degree from Universidad Mayor de San Andrés, and is pursuing a master's degree in Project Management at Universidad para la Cooperación Internacional - Costa Rica.

Agustin Barreto is a software specialist consultant who works at Interamericana de Cómputos S.A, one of the top IBM vendors in Uruguay. His work is focused on discovering new technologies and helping customers in their modernization process. He also worked with DevOps tools and Service Desk solutions. He is studying system engineering at Universidad de la República. He also worked on several open source projects as a cloud native developer consultant. Currently, he works on Lift and Shift modernization projects for migrate services to the cloud.

Paul Chapman is a Consulting Client Technical Specialist for IBM Systems with 25 years of technical and management experience working for IBM and IBM Business Partners. He works with other IBM business units (IBM Cloud, IBM Global Business Services®, and IBM Global Technologies Services) across Europe to provide technical presales for customers, IBM Business Partners, and IBM personnel while designing open, hybrid, and multicloud Red Hat solutions with Power Systems. Paul received an Outstanding Technician SAP solution for a Wide World Energy company.

Michael Easlon is an IT Management Consultant within IBM Garage™ for Systems. He has 20 years of experience with IBM Power Systems and storage. His areas of focus include hybrid cloud, Linux, system management, Red Hat Ansible, Terraform, and Red Hat OpenShift Container Platform.

Luis Ferreira (Luix) is a Senior Software Engineer at IBM Austin who works on cloud containers, Kubernetes related products, and cloud computing design and architecture. He holds a Master of Science degree from Universidade Federal do Rio de Janeiro in Brazil. Before joining IBM Austin, Luis worked at IBM Tivoli® Systems as a Certified Tivoli Consultant, at IBM Brasil as a Certified IT Specialist, and at Cobra Computadores as a kernel developer and software designer for the SOX (UNIX X/Open Compatible) operating system (OS).

Gualberto Esteban Ferrer is a Senior Application Server Administrator at DGI Uruguay. He worked for IBM as an IT Specialist, where he took part in various projects with many people, and as a specialist in Java Platform, Enterprise Edition technology. He worked at IBM for 14 years: 8 years developing and designing application solutions, and 6 years managing applications servers while acting a team leader. He holds a Computer Engineer degree from Universidad de la Republica. He has expertise in IBM WebSphere® Family products, such as WebSphere Application Servers, Tivoli Monitoring, Data Power, and IBM MQ.

Alain Fisher is an IT specialist and DevOps Developer who supports many IBM development teams at the IBM Hursley Lab, UK. He holds a B.Sc. (Hons) degree in Computer Science from The Open University, England. He joined IBM in 2001 supporting IBM Db2® middleware products before moving to the Hursley Lab. His areas of expertise include the OpenStack cloud platform, cloud deployments, automation, and virtualization. He has participated in two previous IBM Redbooks residencies since 2005.

Federico Fros is an IT Specialist who works for IBM Global Technologies Services leading the Distributed Service Line team in Uruguay. He has worked at IBM for more than 16 years, including 13 years of experience in IBM Power Systems and IBM Storage. He is an IBM Certified Systems Expert for UNIX and High Availability and Red Hat Linux Certified Specialist. He is a Redbooks Gold Author, and his previous publications includes a high availability (HA) Redbooks publication on IBM Power and a Redbooks book on OpenShift V3.11 on IBM Power Systems. His areas of expertise include AIX, Linux, PowerHA, IBM PowerVM®, SAN networks, cloud computing, and IBM Storage Family, including IBM Spectrum® Scale.

Miguel Gomez Gonzalez is an IBM Power Systems Integration engineer who is based in Mexico. He has over 11 years of experience in Linux and IBM Power Systems technologies. He has extensive experience in implementing several IBM Linux and AIX clustering solutions. His areas of expertise are Power Systems performance boost, virtualization, HA, system administration, and test design. Miguel holds a Master in Information Technology Management degree from ITESM.

Felix Gonzalez Sugasti is an IT Specialist who works with the UNIX and Storage Team for IBM Global Technologies Services in Uruguay. He studies Electronic Engineering at Universidad de la República in Uruguay. He is a Red Hat Linux Certified Specialist with 5 years of experience in UNIX systems, including FreeBSD, Linux, and AIX. His areas of expertise include AIX, Linux, FreeBSD, Red Hat Ansible, IBM PowerVM, SAN, cloud computing, and IBM Storage.

Abhishek Jain is a Master Inventor and Software Engineer with IBM Systems in Pune, India. He holds a Bachelor of Technology degree in Computer Engineering. His current interests are in hybrid multicloud and container technology solutions. Abhishek has more than 8 years of experience working on various scale-out file systems and block/file/object storage topologies. He started working in his current role as a Test Lead Engineer for IBM Spectrum Scale Container Storage Interface (CSI) Driver and Cloud enablement deliveries.

Paulo Sergio Lemes is a Systems Consultant for AI Solutions on Power Systems with IBM Brazil. He has 19 years of experience in UNIX and Linux, ranging from systems design and development to systems support. His areas of expertise include AIX Power Systems, AIX, IBM GPFS, RHCS, Kernel-based Virtual Machine (KVM), and Linux. He is a Certified Advanced Technical Expert for Power Systems with AIX (CATE) and a Red Hat Certified Engineer (RHCE).

Gabriel Padilla Jimenez is a Test Architect for hardware on Power Systems. He is involved in the early phases of most hardware general availability, and works closely with development to achieve quality products. Gabriel previously worked on manufacturing lines at IBM Mexico as Test Lead for IBM XIV® Storage and Power Systems. Gabriel has a master's degree in information technology and a Bachelor of Science degree in Electronic Engineering.

Sudipto Pal is an Application Architect and Technical Project manager in IBM Global Business Services with over 20 years of leadership experience in designing innovative business solutions for customers from various industry, domain, and geographical locations. He is skilled in cloud computing, big data, application development, and virtualization. He successfully delivered several critical projects with IBM customers from the US and Europe. He led IBM Cognos® administration competency and mentored several candidates. He co-authored *Exploiting IBM PowerVM Virtualization Features with IBM Cognos 8 Business Intelligence*, SG24-7842.

Bogdan Savu is a Cloud Infrastructure Architect for IBM Systems. He has over 13 years of experience in designing, developing, and implementing cloud computing, virtualization, automatization, and infrastructure solutions. Bogdan holds a bachelor's degree in Computer Science from the Polytechnic University of Bucharest. He is a Red Hat Certified Architect, IBM Certified Advanced Technical Expert for Power Systems, and VMware Certified Professional. He holds TOGAF 9 Certified and ITIL 4 Managing Professional Certificate certificates. His areas of expertise include cloud computing, virtualization, DevOps, and scripting.

Deepak C Shetty has over 22 years of IT experience that spans various domains from application development, systems programming, OpenStack and cloud development, and presales consulting on IBM Cognitive and Red Hat OpenShift, multicloud, and hybrid cloud offerings. He has worked in both closed and open source development models. In his current role, he leads Pre-Sales Technical Consulting for IBM Cognitive Offerings at IBM Techline. His primary job role is to provide hardware design configurations, solution sizing, and technical consulting on all IBM Cognitive Offerings for IBM sellers, IBM Business Partners, and distributors. He also leads the Red Hat OpenShift focal team at IBM Techline and spends much time helping the field teams position, size, and sell Red Hat OpenShift on IBM Power Systems. He holds four patents, has authored publications, and has several technical awards, and is an IBM Certified Technical Specialist.

Rodrigo Xavier de Souza is an IBM Power Systems and IBM Cloud on Power Systems consultant with over 14 years of experience with UNIX and Linux systems management and administration. He performs consulting services that are supported by a strong background in AIX and Linux systems, in addition to virtualization on multiple platforms (Power Systems and x86) with SAN and storage knowledge. Rodrigo is a member of the IBM Systems Lab Services Team in Brazil helping customers with engagements for cloud adoption, on-premises (IBM Power Virtualization Center (IBM PowerVC) alone or integrating it with vRA Compose Multiplatform Cloud) or off-premises (IBM Cloud). Rodrigo has been testing and deploying Red Hat OpenShift on Power Systems, exploring their potential and helping to spread these technologies.

Alan Verdugo Munoz is a software engineer who currently works as a Python developer for the Sales and Marketing group of the CIO at IBM Mexico. He has over 10 years of experience writing code professionally and 15 years of experience working with GNU and Linux systems. He holds CompTIA Linux+ and LPIC-1 certifications.

Richard Wale is a Senior IT Specialist who supports many IBM development teams at IBM Hursley Lab, UK. He holds a B.Sc. (Hons) degree in Computer Science from Portsmouth University, England. He joined IBM in 1996 and has supported production IBM AIX systems since 1998. His areas of expertise include IBM Power Systems, PowerVM, AIX, and IBM i.

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

Wade Wallace
IBM Redbooks, Austin Center

Daniel de Souza Casali, Chuck Bryan, Si Win, Joe Cropper, Bruce Anthony, Manoj Kumar,
Bhaskar Reddy Ravula
IBM US

Li Jun BJ Zhu and Yun CDL Wang
IBM China

Benjamin Haubeck
IBM Germany

Abhishek Soni
IBM Singapore

Pradipta Banerjee
IBM India

Bhadri Madapusi
IBM Canada

Benoit Marolleau
IBM France

Juan Jose Floristan
Red Hat US

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

- ▶ 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>



Introduction

This chapter provides an overview of the scope of this publication.

This chapter contains the following topics:

- ▶ Introduction
- ▶ Scope

1.1 Introduction

This publication provides an overview of the Red Hat OpenShift Platform on IBM Power Systems. It is the successor to [Red Hat OpenShift and IBM Cloud Paks on IBM Power Systems: Volume 1, SG24-8459](#), which provided a summary overview of containers, Kubernetes, and the introduction of Red Hat OpenShift onto IBM Power Systems.

With over 25 years since the release of the original models, Power Systems continues to be designed for both traditional enterprise workloads and the most demanding, data-intensive computing. The range of models offer flexibility, scalability, and innovation.

This book is also a successor to [Red Hat OpenShift V4.3 on IBM Power Systems Reference Guide, REDP-5599](#). There were many changes since Volume 1 and Red Hat OpenShift V3.11 that the paper covered that provided an update of what was new with Red Hat OpenShift V4.3 on Power Systems, including installation instructions and sizing guides.

Since that paper, both Red Hat OpenShift V4.4 and V4.5 were released, and the landscape continued to broaden with the availability of more IBM Cloud Paks (on Power Systems) and updated versions of existing IBM Cloud Paks.

This book describes the differences between Red Hat OpenShift V3.x and Red Hat OpenShift V4.x. Red Hat OpenShift on IBM Power Systems is still a collection of logical partitions (LPARs) that is hosted on either IBM POWER8® or IBM POWER9™ processor-based systems. This book helps you to bring the advantages of the cloud to your enterprise applications running on your own hardware, which can be a more preferable modernization route than bringing your enterprise data out to the cloud.

1.2 Scope

This publication encompasses many topics similar to its predecessor publications. However, the document provides updated references for Red Hat OpenShift V4.x and IBM Cloud Paks on IBM Power Systems, including architecture references and storage options.

This book also provides updated content around deployment planning and suggested sizing guides and requirements.

More content is provided that illustrates scenarios from a technology and industry viewpoint. The book also illustrates best practices and current tuning recommendations for Red Hat OpenShift on Power Systems, including ongoing monitoring and administration of Red Hat OpenShift.

The later chapters of this book describe deployment scenarios from an installation viewpoint, including IBM Power Virtualization Center (IBM PowerVC) updates to support Red Hat OpenShift V4.x deployments, Red Hat OpenShift Container Platform v4.5 deployment in IBM Power Systems Virtual Server on IBM Cloud, and others.

This book documents many areas that were researched, studied, implemented, and tried during the residency, which complements documentation that is available in many IBM products and solutions sites, including IBM Knowledge Center.



Reference architecture

This chapter describes a typical reference architecture for Red Hat OpenShift on IBM Power Systems that use IBM Power Systems L922 servers.

This chapter contains the following topics:

- ▶ A typical reference architecture for Red Hat OpenShift
- ▶ Storage requirements

2.1 A typical reference architecture for Red Hat OpenShift

This section describes a typical reference architecture for a recommended installation that uses Power L922 servers.

Figure 2-1 shows a block diagram of a typical Red Hat OpenShift on Power Systems cluster that uses Power L922 servers. Power L922 is IBM industry solution server for mission-critical Linux workloads delivering optimum price-performance benefits. It is a 2U server with a large memory footprint and comes with PowerVM virtualization built-in, so you can create multiple logical partitions (LPARs) to host different types of Red Hat OpenShift nodes.

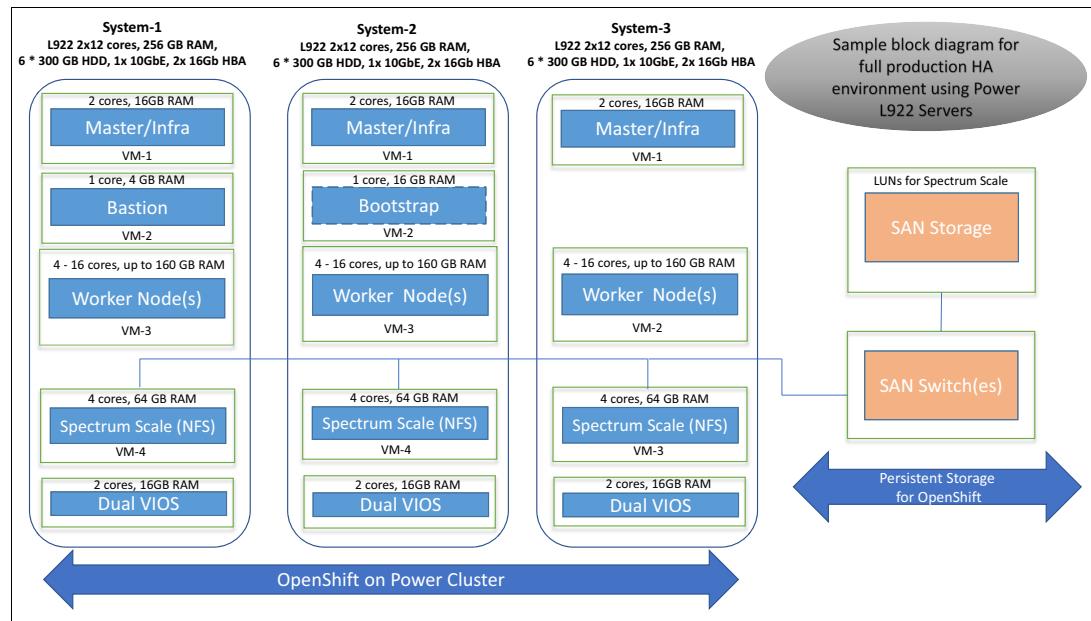


Figure 2-1 Typical Red Hat OpenShift on Power L922 servers that use SAN as persistent storage

The Red Hat OpenShift architecture builds on Kubernetes and consists of multiple types of roles for the Red Hat OpenShift nodes:

- ▶ **Bootstrap**

Red Hat OpenShift Container Platform uses a temporary bootstrap node. The bootstrap node role during initial configuration is to provide the required information to the master node (control plane).

The bootstrap node starts by using an ignition configuration file that describes how to create the cluster. The process for creating ignition file is described in Example 6-7 on page 87.

The bootstrap node sets up the master nodes, and the master nodes set up the worker nodes. You must create the LPARs for masters and workers and set them to use the correct ignition files.

The master nodes install more services in the form of a set of operators. The Red Hat OpenShift bootstrap node runs Red Hat CoreOS V4.3. At a minimum, the bootstrap node must have 1 core, 16 GB of RAM and 120 GB of storage. The complete cluster requirements are documented in Table 6-2 on page 93.

Note: The cluster requires the bootstrap machine to deploy the Red Hat OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.

► Master and infrastructure:

The Red Hat OpenShift Container Platform master is a server that performs control functions for the entire cluster environment. The master machines form the control plane that is responsible for the creation, scheduling, and management of all objects that are specific to Red Hat OpenShift. It includes application programming interface (API), controller manager, and scheduler capabilities in one Red Hat OpenShift binary file.

It is also a best practice to install an etcd key-value store on Red Hat OpenShift masters to achieve a low-latency link between etcd and Red Hat OpenShift masters. To maintain high availability (HA) of the cluster, use separate physical hosts for the masters. The Red Hat OpenShift master node runs Red Hat CoreOS V4.3.

The following Red Hat OpenShift Container Platform components are infrastructure components:

- Kubernetes and Red Hat OpenShift Container Platform control plane services that run on masters
- The default router
- The container image registry
- The cluster metrics collection, or monitoring service
- Cluster aggregated logging
- Service brokers

The Red Hat OpenShift master and infrastructure nodes run Red Hat CoreOS V4.3.

Important: Because of the consensus that is required by the RAFT algorithm, the etcd service must be deployed in odd numbers to maintain quorum. For this reason, the minimum number of etcd instances for production environments is three.

For more information about the RAFT Consensus Algorithm, see [The Raft Consensus Algorithm](#).

It is also a best practice to consolidate the master and infrastructure nodes into a single virtual machine (VM).

► Worker:

Red Hat OpenShift worker nodes run containerized applications that are created and deployed by developers. A Red Hat OpenShift worker node contains the Red Hat OpenShift node components, including the container engine CRI-O, Kubelet for running and stopping container workloads, and a service proxy for managing cross worker nodes communication for pods. A Red Hat OpenShift worker node runs Red Hat CoreOS V4.3.

► Bastion:

A Bastion node is a virtual or physical host that is required for the installation of Red Hat OpenShift, and it hosts the infrastructure services that are used to deploy Red Hat OpenShift.

The Red Hat OpenShift installation assumes that many of the external services like DNS, load balancing, HTTP Server, and DHCP are already available in an existing data center, so there is no need to duplicate them on a node in the Red Hat OpenShift cluster.

However, current experience shows that creating a Bastion node and consolidating the Red Hat OpenShift installation external services on the installation greatly simplifies installation.

After installation is complete, the Bastion node can continue to serve as a load balancer for the Red Hat OpenShift API service (running on each of the master nodes) and the application ingress controller (also running on the three master nodes). By providing this single interface to the Red Hat OpenShift cluster, the Bastion node can serve as a jump server that controls access between an external network and the Red Hat OpenShift cluster network.

The Red Hat OpenShift Bastion node runs Red Hat Enterprise Linux 8.x.

- ▶ IBM Spectrum Scale (NFS):

Red Hat OpenShift environments with multiple workers usually require a distributed shared file system. This requirement stems from the fact that most applications require some sort of persistent store. In most cases, the developer of the application does not have any control over which worker in the Red Hat OpenShift cluster to which the application pod is dispatched. Hence, regardless of which worker the application can be deployed to, the persistent storage that is required by the application must be available.

At the time of writing, the only supported ReadWriteMany (rwx) storage provider for Red Hat OpenShift V4.3 on the IBM Power Architecture® is an NFS server. You can use IBM Spectrum Scale as your NFS server, as shown in Figure 2-1 on page 4.

The Red Hat OpenShift IBM Spectrum Scale (NFS) node runs Red Hat Enterprise Linux 8.x.

2.2 Storage requirements

This section describes the storage solutions that are available for Red Hat OpenShift Container Platform on Power Systems. In most cases, the container platform demands a storage solution that can support various use cases for running applications while maintaining the platform's flexibility and agility. Red Hat OpenShift environments with multiple worker nodes require a distributed shared file system for persistent storage so that developers do not need to control the application pod deployment among the Red Hat OpenShift cluster's worker nodes because in a distributed shared file system, data can be accessed from any worker node. However, there are some applications that require block storage environments for application compatibility because of the nature of the input and output activities, or because less complexity is required to provision a shared and distributed storage environment.

In a container world, the Persistent Volume (PV) is a unit of storage in the cluster that has been provisioned by an administrator, for example, state provisioning or dynamically provisioned by using a storage driver or plug-in. A Persistent Volume Claim (PVC) is a request for storage by a user (for example, creating a pod). The PV is used by the PVC.

The Container Storage Interface¹ (CSI) is the result of a collaborative initiative across multiple storage vendors to unify the storage interface of Container Orchestrator Systems (COS), such as Kubernetes, Mesos, Docker Swarm, and Cloud Foundry, combined with storage vendors such as IBM, Ceph, Portworx, and NetApp. A single CSI implementation for a storage vendor must work with all COS. CSI defines a specification that storage vendors implement in a CSI driver. CSI is based on the general-purpose Remote Procedure Calls (gRPC) framework. It provides extended volume management functions, such as snapshots, clones, and volume expansion.

For containers, there are two basic topologies that use storage for persistent storage:

- ▶ Container-ready storage
- ▶ Container-native storage

2.2.1 Container-ready storage

Container-ready storage is a storage device that is provisioned as a stand-alone storage cluster, such as a SAN storage array, NAS device, or software-defined storage (SDS) to provide persistent storage to containerized applications. This external stand-alone storage device presents persistent storage to a container application (by using a CSI driver) through a fabric to a Kubernetes PVC. Volumes can be directly provisioned by using dynamic or statically provisioned storage with all supported volume access modes.

As a stand-alone storage cluster, container-ready storage provides full support for Red Hat OpenShift Container Platform infrastructure services such as tiering, audit logging, performance metrics, replication services, and health monitoring. Furthermore, by using a stand-alone cluster, the Red Hat OpenShift administrator to manage, scale, and upgrade the storage cluster independently.

¹ <https://kubernetes-csi.github.io/docs/introduction.html>

There are several Red Hat and IBM storage products that are container-ready with the release of their CSI drivers (see Figure 2-2):

- ▶ IBM Spectrum Virtualize (including public cloud)
- ▶ IBM Spectrum Scale and IBM Elastic Storage® Server
- ▶ IBM Cloud Object Storage
- ▶ Red Hat Ceph
- ▶ IBM Power VC

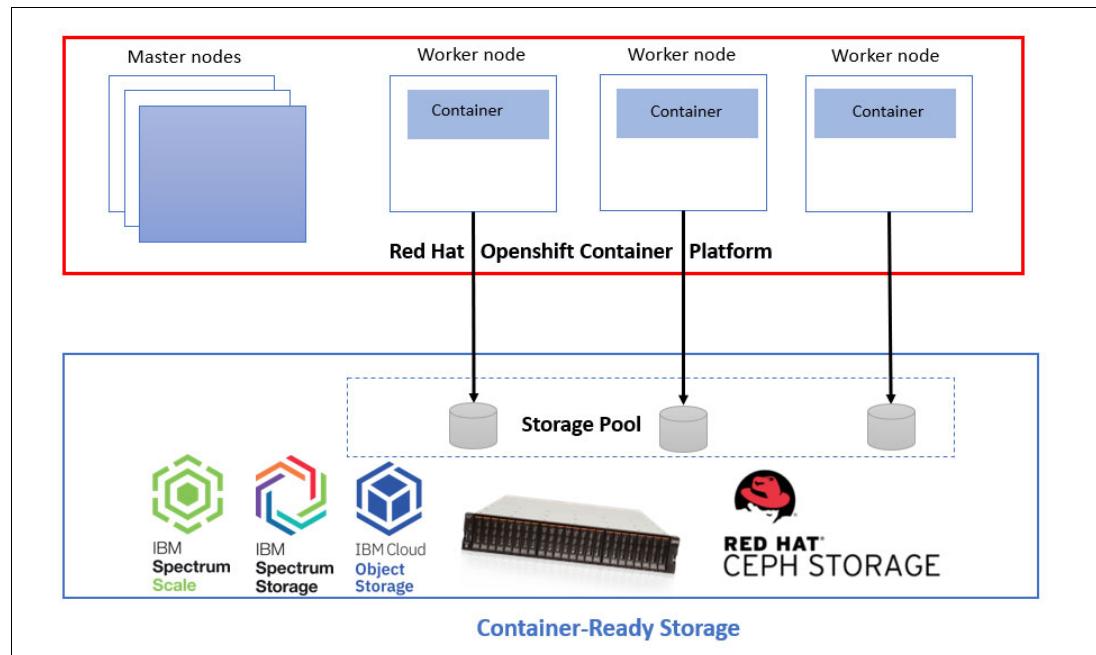


Figure 2-2 Container ready storage

There are others key factors to choose container-ready storage:

- ▶ Can be used as file, block, or object storage depending on the supported external storage.
- ▶ Can use existing investments in infrastructure, management, and monitoring tools.
- ▶ Can perform cluster maintenance activities independently while storage administrators are responsible for data integrity.

2.2.2 Container-native storage

Container-native storage is SDS that runs in a container that co-resides with the compute containers within the Kubernetes cluster. Container-native storage serves storage from local or direct-attached storage by virtualizing the underlying physical storage to create a storage pool that is used to create volumes with the properties that are required by application containers for persistent storage. Container-native storage can use either internal drives in cluster nodes or external storage devices. It is a best practice to have dedicated storage nodes to enable storage and compute nodes to scale independently of each other within the cluster.

In this topology, storage deployment and management are integrated with Red Hat OpenShift Container Platform services, which provides persistent storage to a container application (by using CSI) through PVC by using dynamic and statically provisioned storage with all supported volume access modes.

Container-native storage gives cluster users a level of automation and responsiveness that is not available with traditional storage (even augmented with a CSI driver). Cluster administrators should be aware of this value to the cluster of the worker nodes that provide storage. There are several products for container-native storage (see Figure 2-3):

- ▶ Red Hat OpenShift Container Storage: Composed of Ceph, Rook, and NooBaa.
- ▶ IBM Spectrum Scale (Container Native).

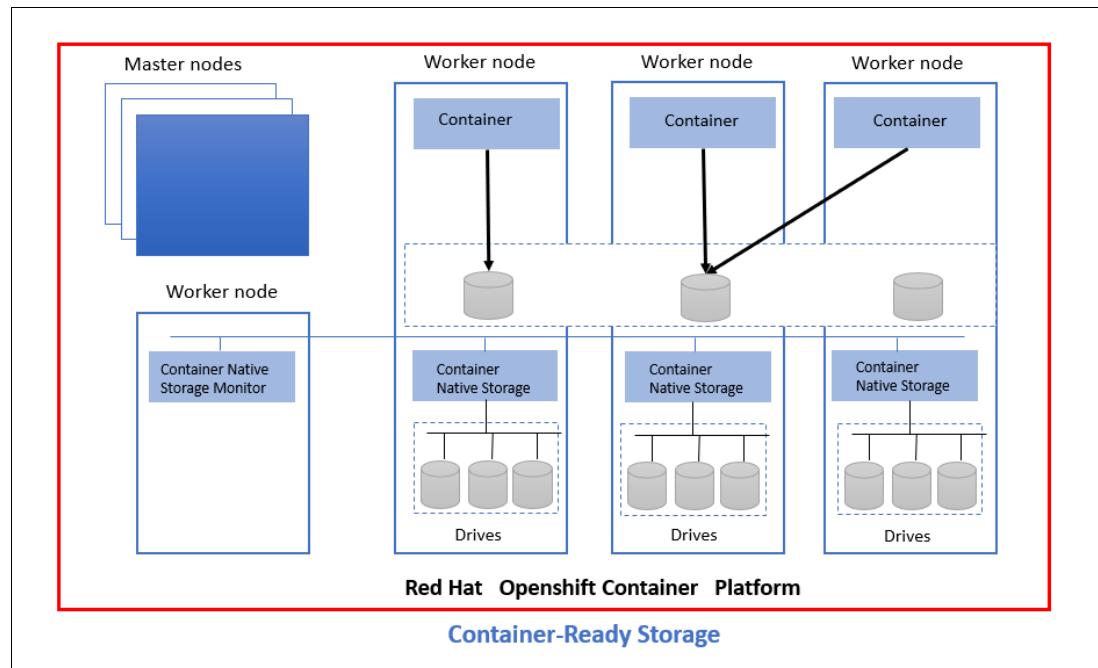


Figure 2-3 Red Hat OpenShift Container Platform: Container-native storage

There are other key factors to choose container-native storage:

- ▶ Can be deployed on a public cloud, bare metal, or virtualized hardware without any changes.
- ▶ Can be used as file, object, or block storage based on the supported storage architecture.
- ▶ Can be deployed by a user-defined Kubernetes and Red Hat OpenShift Cloud Platform controller.
- ▶ Can be managed by a single administrator who is responsible for Red Hat OpenShift and storage services.
- ▶ Can be monitored by using standard Kubernetes and Red Hat OpenShift health monitoring services.

2.2.3 Container storage use cases

A container storage solution for Kubernetes and the Red Hat OpenShift Container platform must support various use cases and still maintain the platform's flexibility and agility, as shown in Table 2-1.

Table 2-1 Platform and application storage requirements

Requirement	Persistent storage requirements	Preferred storage type
Platform storage requirements	Internal container image registries.	Object storage with rwx access mode.
	Prometheus-based platform monitoring and metrics stack.	Block storage.
	Elastic Logstash Kibana logging stack.	Local storage on the node as block storage for faster access.
	Shared application data access outside the Red Hat OpenShift and Kubernetes cluster.	Block or file storage that uses the CSI driver as container-ready storage.
	Multi-cluster or multi-site replication.	Block and file storage that supports geo-dispersed cluster replication, such as IBM Spectrum Scale or IBM Spectrum Virtualize.
	High throughput for relational or NOSQL databases.	IBM Spectrum Virtualize for block storage.
	Low latency performance requirements.	IBM Spectrum Scale is used on high-performance computing and delivers low latency and high performance.
Application requirements	Scale-out (10000) storage volume requirement.	IBM Spectrum Scale supports more storage (1000 independent file sets, 10,000 dependent file sets, and unlimited lightweight directories). IBM Spectrum Virtualize is limited to 10,000 volumes per system and 2048 volume mappings per host.
	Storage footprint greater than 36 TiB.	IBM Spectrum Scale, IBM Block Storage, or IBM Cloud Object Storage. There are no practical size limitations.
	Heterogeneous Red Hat OpenShift cluster, including IBM Z® or IBM Power Systems. A requirement for rwx.	IBM Spectrum Scale or Red Hat OpenShift Container Storage independent mode.
	Standardization to build apps that follow the cloud-native approach and align with the performance characteristics of object storage.	IBM Cloud Object Storage or Red Hat OpenShift Container Storage.
	Take advantage of and integrate into existing IBM block storage footprint. Requires only RWO (single-writer) capable storage.	IBM Spectrum Virtualize for block storage.

From a persistent storage perspective, there are three types of storage that can be used in a container application: file, block, and object. For all types of use cases, the chosen solution must support:

- ▶ Dynamic and static volume provisioning
- ▶ Red Hat OpenShift Container Platform PV access modes
- ▶ A wide range of application use cases

Static and dynamic volume provisioning

This section describes the volume provisioning options.

Static provisioning

A cluster administrator creates several PVs upfront. PVs carry the details of the real storage that is available for use by cluster users, which causes the administrator to know and set the storage requirements upfront. This type of provisioning is useful when there is existing data on a storage cluster that must be provisioned and consumed as a PV for containers.

Dynamic provisioning

Dynamic volume provisioning enables storage volumes to be created on-demand. This feature eliminates the need for cluster administrators to pre-provision storage. Instead, dynamic provisioning automatically provisions storage when it is requested by users. The implementation of dynamic volume provisioning is based on storage class objects. A cluster administrator can define as many storage class objects as needed, each specifying a volume plug-in (also known as a provisioner) that provisions a volume and the set of parameters to pass to that provisioner when provisioning. A cluster administrator can define and expose multiple types of storage (from the same or different storage systems) within a cluster, each with a custom set of parameters.

Red Hat OpenShift Container Platform Persistent Volume access modes

A PV can be mounted on a worker node in any supported way. There can be providers with different capabilities for PV access modes that are supported by that specific volume. The access modes are:

- ▶ **ReadWriteOnce (RWO)**: The volume can be mounted as read/write by a single node.
- ▶ **ReadOnlyMany (ROX)**: The volume can be mounted read-only by many nodes.
- ▶ **ReadWriteMany (rwx)**: The volume can be mounted as read/write by many nodes.

2.2.4 Container storage on Power Systems servers

This section describes the storage options that are available for Red Hat OpenShift Container Platform V4.3 on Power Systems. Red Hat OpenShift environments with multiple workers usually require a distributed shared file system for persistent storage because the developer of the application often does not have any control over which worker in the Red Hat OpenShift cluster to which the application pod is dispatched. So, regardless of which worker the application can be deployed to, the persistent storage that is required by the application must be available.

NFS storage

At the time of writing, the only supported rwx storage provider for Red Hat OpenShift V4.3 on Power Systems is an NFS server. You must install the required NFS components on each of the Red Hat OpenShift worker nodes in addition to the node that is designated as the NFS server. For each Red Hat OpenShift PV, an explicit NFS export must be created. However, you do not explicitly mount the exports on the Red Hat OpenShift workers. The mounts are done by Red Hat OpenShift as needed by the containers when they issue an associated PVC for the predefined PVs.

IBM Spectrum Scale and NFS

IBM Spectrum Scale provides extra protocol access methods. Providing these extra file and object access methods and integrating them with IBM Spectrum Scale offers several benefits, such as enabling users to consolidate various sources of data efficiently in one global namespace. Another benefit is that it provides a unified data management solution that enables efficient space utilization and avoids unnecessary data moves because access methods can be different.

Protocol access methods that are integrated with IBM Spectrum Scale are NFS, SMB, and object. Although each of these server functions (NFS, SMB, and object) uses open source technologies, this integration adds value by providing scaling and HA by using the clustering technology in IBM Spectrum Scale. The NFS support for IBM Spectrum Scale enables customers to access the IBM Spectrum Scale file system by using NFS clients with their inherent NFS semantics.

For more information about setting up IBM Spectrum Scale as an NFS server, see [IBM Knowledge Center](#).

2.2.5 IBM Power Virtualization Center Container Storage Interface driver

With CSI, Red Hat OpenShift can use storage from storage back ends that implement the CSI interface as persistent storage. The IBM Power Virtualization Center (IBM PowerVC) CSI driver is standard for providing storage RWO functions to containers.

The IBM PowerVC CSI pluggable driver interacts with IBM PowerVC storage for operations such as create volumes, delete volumes, or attach or detach volumes.

For more information about the CSI in Red Hat OpenShift, see [Configuring CSI volumes](#).

For more information about the IBM PowerVC CSI and how to configure it, see [IBM Knowledge Center](#).

2.2.6 IBM Spectrum Scale Container Native Storage

IBM Spectrum Scale is a cluster file system that provides concurrent access to a single file system or set of file systems from multiple nodes. The nodes can be SAN-attached, network-attached, a mixture of SAN-attached and network-attached, or in a shared-nothing cluster configuration. IBM Spectrum Scale enables high-performance access to a common set of data to support a scale-out solution or to provide a HA platform, as shown in Figure 2-4 on page 13.

With IBM Spectrum Scale Container Native Storage Access, an IBM Spectrum Scale client cluster can run as a container in a Red Hat OpenShift Container Platform cluster with an automatic remote mount of an external IBM Spectrum Scale storage cluster by using the OpenShift operator. Then, application developers can use a container-persistent data store for stateful applications by using the IBM Spectrum Scale CSI driver.

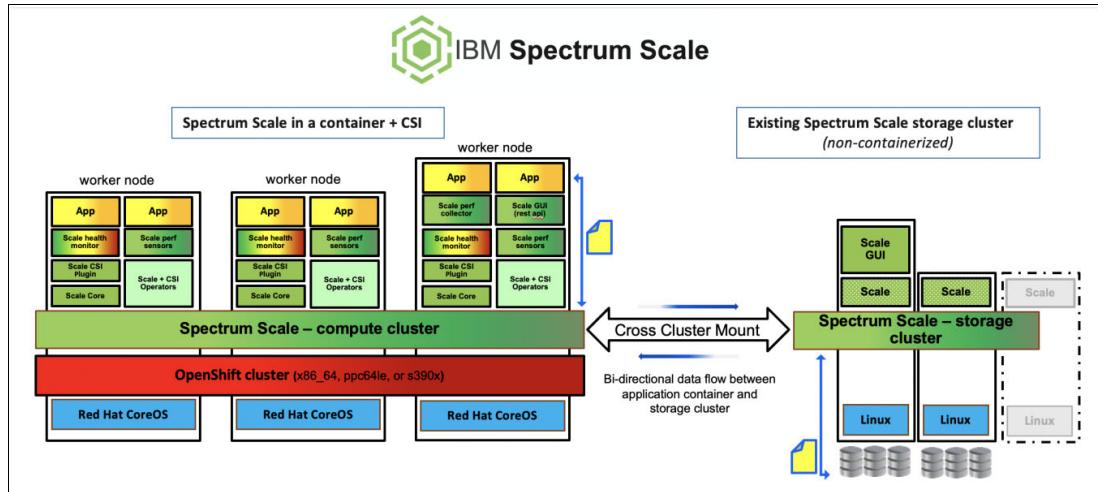


Figure 2-4 High-level architecture of IBM Spectrum Scale Container Native Storage Access

For more information about the benefits and features of IBM Spectrum Scale Container Storage, see [IBM Knowledge Center](#).

2.2.7 Red Hat OpenShift Container Storage

Red Hat OpenShift Container Storage is SDS integrated with and optimized for Red Hat OpenShift Container Platform. It is built on Red Hat Ceph Storage, Rook, and NooBaa to provide container-native storage services that can support block, file, and object storage concurrently. Red Hat OpenShift Container Storage deployment can be achieved on-premises or in the public cloud with Red Hat OpenShift Container Platform.

Note: At the time of writing, Red Hat OpenShift Container Storage for Red Hat OpenShift Container Platform is not available.

2.2.8 IBM Storage Suite for IBM Cloud Paks

IBM Storage Suite for IBM Cloud Paks is a flexible SDS solution that is based on IBM Spectrum Storage family and Red Hat storage solutions. This suite offers enterprise data and storage services to container environments with a comprehensive set of SDS and a faster, more reliable way to modernize, complement, and support the deployment for effective operations of IBM Cloud Paks solutions. IBM Storage Suite for IBM Cloud Paks is a comprehensive set of SDS solutions that includes data resources for file, object, and block data, and services for data management that include:

- ▶ IBM Spectrum Scale Data Management Edition provides high-performance, highly functional file data management for all the types of business data.
- ▶ IBM Spectrum Virtualize provides comprehensive data services and storage virtualization capabilities for block storage.

- ▶ IBM Cloud Object Storage is a highly scalable cloud object storage solution for unstructured data on-premises and cloud-based.
- ▶ IBM Spectrum Discover provides a metadata management solution that provides data insight for petabyte-scale unstructured storage.
- ▶ Red Hat OpenShift Container Storage deploys applications and storage in either container-ready or container-native mode.
- ▶ Red Hat Ceph Storage is a storage solution for modern workloads like cloud infrastructure, data analytics, media repositories, and backup and restore systems.

For pricing and more information, see [IBM Storage Suite for IBM Cloud Paks](#).



IBM Cloud Paks

IBM Cloud Paks are an integrated set of software solutions for hybrid cloud to help you fully implement intelligent workflows in your business to accelerate digital transformation. The solutions are built on the Red Hat [open hybrid cloud](#) platform so you can build once and deploy anywhere.

This chapter provides an overview of IBM Cloud Paks. This chapter contains the following topics:

- ▶ IBM Cloud Paks
- ▶ IBM Cloud Paks offering
- ▶ IBM Cloud Pak for Data
- ▶ IBM Cloud Pak for Multicloud Management
- ▶ Use cases: IBM Cloud Pak for Data

3.1 IBM Cloud Paks

The IT world is primarily divided into two major segments:

- ▶ New development.
- ▶ A “Run” operation where new features and enhancements are deployed to a production environment.

Both phases have their own requirements, constraints, and challenges. The cloud migration process for a production environment is different than one in a development environment. This process uses new terms like *cloud migration* and *cloud modernization*.

Some of the key performance indicators (KPIs) are:

- ▶ Easy to manage orchestration service
- ▶ Improved security
- ▶ Efficiency in governance
- ▶ Reduced cost to maximize return on investment (ROI)

IBM Cloud Paks are enterprise-ready, containerized middleware and common software solutions that are hosted on a Kubernetes based platform. It provides an open, faster, and more secure way to move core business applications to any cloud. It is a full stack, converged infrastructure with a virtualized cloud hosting environment that helps to extend applications to cloud.

IBM Cloud Paks are built on a Kubernetes based portable platform that uses a common container platform from Red Hat OpenShift. This enterprise-ready containerized software solution provides several key benefits to different segment of users (see Figure 3-1 on page 17):

- ▶ Build phase: The solution is packaged with open platform components to take advantage of several application programming interface (API) services that are available from different sources. The solution is easy to build and distribute. Developers can take advantage of the single application environment that is configured with all the required tools for planning the modernization process and building the cloud-native API and runtime platform to deploy the solution.
- ▶ Move phase: The solution is a run-anywhere model where the same software can be ported to an on-premises or private or public cloud based on the customer's requirements. This phase is a transition from a large monolithic application to an API-based microservice model.
- ▶ Run phase: The solution has built-in services like logging, monitoring, metering, security, identity management, and an image registry. Each business has a unique set of key business values that can determine whether the application operates from on-premises or private and public cloud. The unified Kubernetes platform provides such flexibility to deploy and run the same application in any platform.

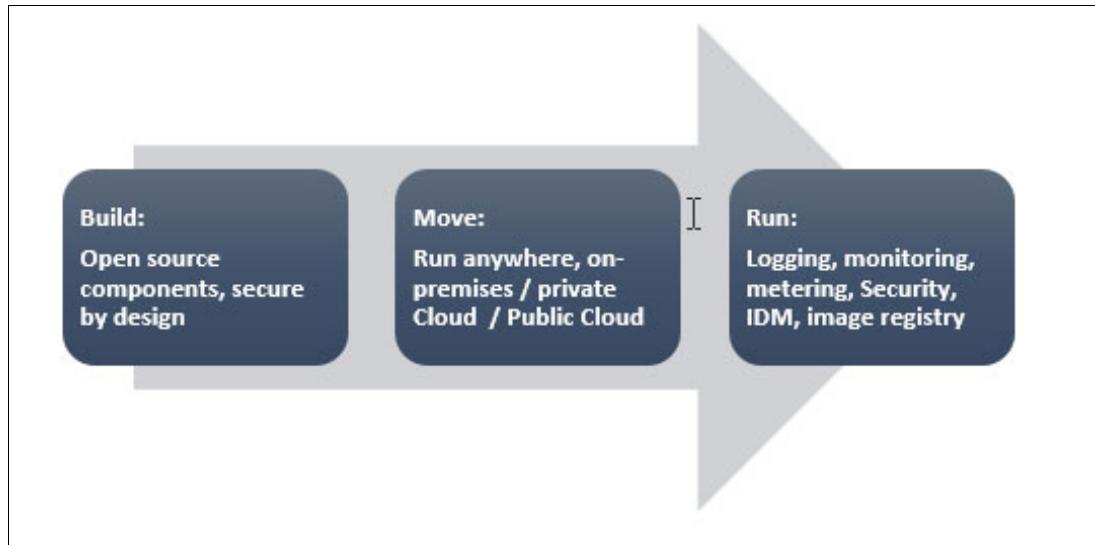


Figure 3-1 IBM Cloud Paks transition phases

3.2 IBM Cloud Paks offering

Beyond containers and Kubernetes, enterprises must orchestrate their production topology, and provide management, security, and governance for their applications.

IBM Cloud Paks are enterprise-ready, containerized software solutions that give customers an open, faster, and more secure way to move core business applications to any cloud. Each IBM Cloud Pak includes containerized IBM middleware and common software services for development and management on top of a common integration layer, which reduces development time and operational expense.

3.2.1 Benefits of IBM Cloud Paks

Here are a few benefits of IBM Cloud Paks:

- ▶ **Market ready:** IBM Cloud Paks with Red Hat OpenShift is the most flexible combination to ensure faster deployment with high scalability. API-based microservices ensure faster adoption of changes. Cloud-native applications are quickly developed by using container and microservices that can take advantage of capabilities of a middleware database by using DevOps practices.
- ▶ **Run anywhere:** IBM Cloud Paks are portable. They can run on-premises, on public clouds, or in an integrated system.
- ▶ **Open and secure:** IBM Cloud Paks are certified by IBM with up-to-date software to provide full-stack support from hardware to applications.

3.2.2 IBM Cloud Pak for Application

IBM Cloud Pak for Application accelerates the modernization and building of applications by using built-in developer tools and processes, including support for analyzing existing applications and guiding the application owner through the modernization journey. In addition, it supports cloud-native development microservices functions and serverless computing. Customers can quickly build cloud apps, and existing IBM middleware customers gain the most straightforward path to modernization.

Transformation of traditional applications is one of the key features in this scope. Development, testing, and redeployment are some of the phases where most of the effort and challenges are experienced in a traditional development model. An agile DevOps based development model is one of the potential solutions to break this traditional challenge. To complement an agile development process, IBM Cloud Pak for Application extends Kubernetes features for a consistent and faster development cycle, which helps IBM customers to build cost-optimized and smarter applications.

3.2.3 IBM Cloud Pak for Data

IBM Cloud Pak for Data unifies and simplifies the collection, organization, and analysis of data. Enterprises can turn data into insights by using an integrated cloud-native architecture. IBM Cloud Pak for Data is extendable and can be customized to a customer's unique data and artificial intelligence (AI) landscapes by using an integrated catalog of IBM, open source, and third-party microservice add-ons.

For more information about IBM Cloud Pak for Data, see 3.3, “IBM Cloud Pak for Data” on page 19 and 3.5, “Use cases: IBM Cloud Pak for Data” on page 34.

3.2.4 IBM Cloud Pak for Integration

IBM Cloud Pak for Integration integrates applications, data, cloud services, and APIs. It supports the speed, flexibility, security, and scale that is required for modern integration and digital transformation initiatives. It comes with a pre-integrated set of capabilities, which includes an API lifecycle, application and data integration, messaging and events, high-speed transfer, and integration security.

Personalizing the customer experience is the primary business focus that needs an integrated view of all data. IBM Cloud Pak for Integration facilitates rapid integration of data along with security, compliance, and version capability. Here are some of the key capabilities:

- ▶ API lifecycle management
- ▶ Application and data integration
- ▶ Enterprise messaging
- ▶ Event streaming
- ▶ High-speed data transfer
- ▶ Secure gateway

3.2.5 IBM Cloud Pak for Automation

IBM Cloud Pak for Automation transforms business processes, decisions, and content. It is a pre-integrated set of essential software that enables customers to easily design, build, and run intelligent automation applications at scale. It has three major KPIs:

- ▶ Improved efficiency and productivity
- ▶ Enhanced customer experience
- ▶ Operational insight

IBM Cloud Pak for Automation helps to automate business operations by using an integrated platform. Kubernetes makes it easier to configure, deploy, and manage containerized applications. It is compatible with all types of projects small and large to deliver better end-to-end customer journeys with improved governing of content and processes.

By using IBM Cloud Pak for Automation, you can work more effectively in the following cases:

- ▶ When you have a limited work force to manage a higher workload from new applications or services, rising customer demand, or seasonal fluctuations.
- ▶ When you create enhanced and personalized customer experiences that increase loyalty by drawing insights instantly from multiple sources of information.
- ▶ When you must scale operations to help maximize revenue and customer service.

3.2.6 IBM Cloud Pak for Multicloud Management

IBM Cloud Pak for Multicloud Management provides consistent visibility, governance, and automation across a range of multicloud management capabilities, such as infrastructure management, application management, multicluster management, edge management, and integration with existing tools and processes.

3.3 IBM Cloud Pak for Data

This section describes the features of the IBM Cloud Pak for Data solution.

3.3.1 Key features of IBM Cloud Pak for Data

Here are the key features of IBM Cloud Pak for Data:

- ▶ The source system is on-premises data and a data warehouse. The source system contains a large volume of data.
- ▶ Data management:
 - Connect to data.
 - Data governance.
 - Identify data.
 - Data transformation for analytics.
- ▶ User access and collaboration capability:
 - Single unified interface.
 - Services to data management and collaboration.
 - Data readiness for ready use in analytics.
 - Access and connection are built-in features.

- ▶ Functions: Fully integrated data and AI platform:
 - Data collection.
 - Data organization.
 - Data analysis.
 - Infuse AI into business.
 - Supports a multi-cloud architecture.
- ▶ Benefits
 - Cost saving.
 - Built-in services.
 - Tools are integrated.
 - Scope for a customized solution.

3.3.2 IBM Cloud Pak for Data features

IBM Cloud Pak for Data V2.5.0, customers can selectively install and activate required services. IBM Cloud Pak for Data has several enhancements to enhance usability and integrated service orientation. Enhancements are classified into sections, as shown in Figure 3-2.

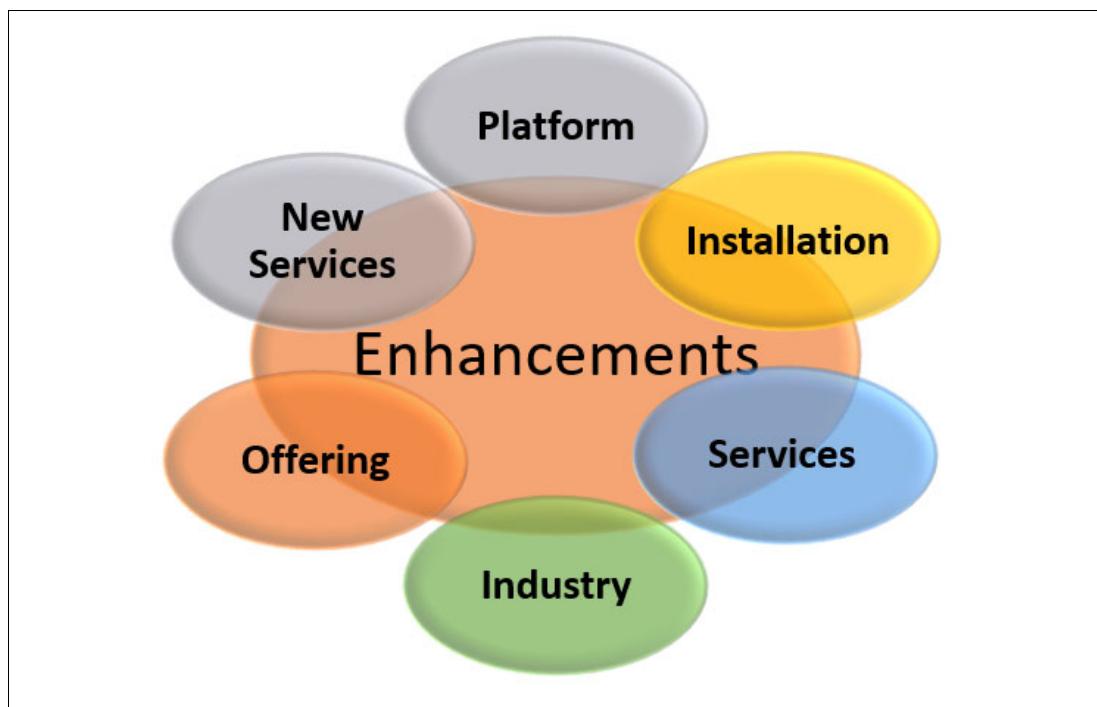


Figure 3-2 IBM Cloud Pak for Data enhancement cycle and scope of activities

- ▶ Platform enhancements:
 - Has a modular services-based platform setup for efficient and optimized utilization of resources.
 - Has a built-in dashboard with meaningful KPI for better monitoring, metering, and serviceability.
 - Offers an open extendable platform with new Platform and Service APIs.

- ▶ Installation enhancements:
 - Has a simplified installation.
 - Uses Red Hat OpenShift v4.x.
- ▶ Service enhancements:
 - Data processing and analytics are some of the key enhancements.
 - Has advanced integration with IBM DataStage® and Db2.
 - Has advanced predictive and analytical models that use IBM SPSS® and IBM Watson® Studio Streams suites.
- ▶ Offering: IBM Cloud Pak for Data enhancements:
 - DataStage Edition.
 - Watson Studio Premium.
 - IBM Db2.
- ▶ New services categories:
 - AI.
 - Integration with Spark and Hadoop.
 - Developer tools like Jupyter Notebook with Python 3.6.

IBM Cloud Pak for Data emerged as an extensible and highly scalable data and AI platform. It is built on a comprehensive foundation of unified code, data, and AI services that can take advantage of multiple cloud-native services, and it is flexible enough to adopt customization to address specific business needs through an extended service catalog, which includes the following services:

- ▶ AI: Consists of several Watson libraries, tools, and studios.
- ▶ Analytics: Services include Trusted Predictive and analytical platforms like Cognos, SPSS, Dashboards, Python, and others.
- ▶ Data governance: Consists of IBM InfoSphere® and Watson libraries.
- ▶ Data Store service.
- ▶ Industry solution.
- ▶ Storage.

3.3.3 IBM Cloud Pak for Data: Journey with data

The real power of data can be observed only after it is interpreted correctly. A series of complex transformation processes must be performed on the data until the analytical solution provides business insight to the customer.

IBM Cloud Pak for Data provides several unique solutions in each phase of this data transformation, as shown in Figure 3-3.

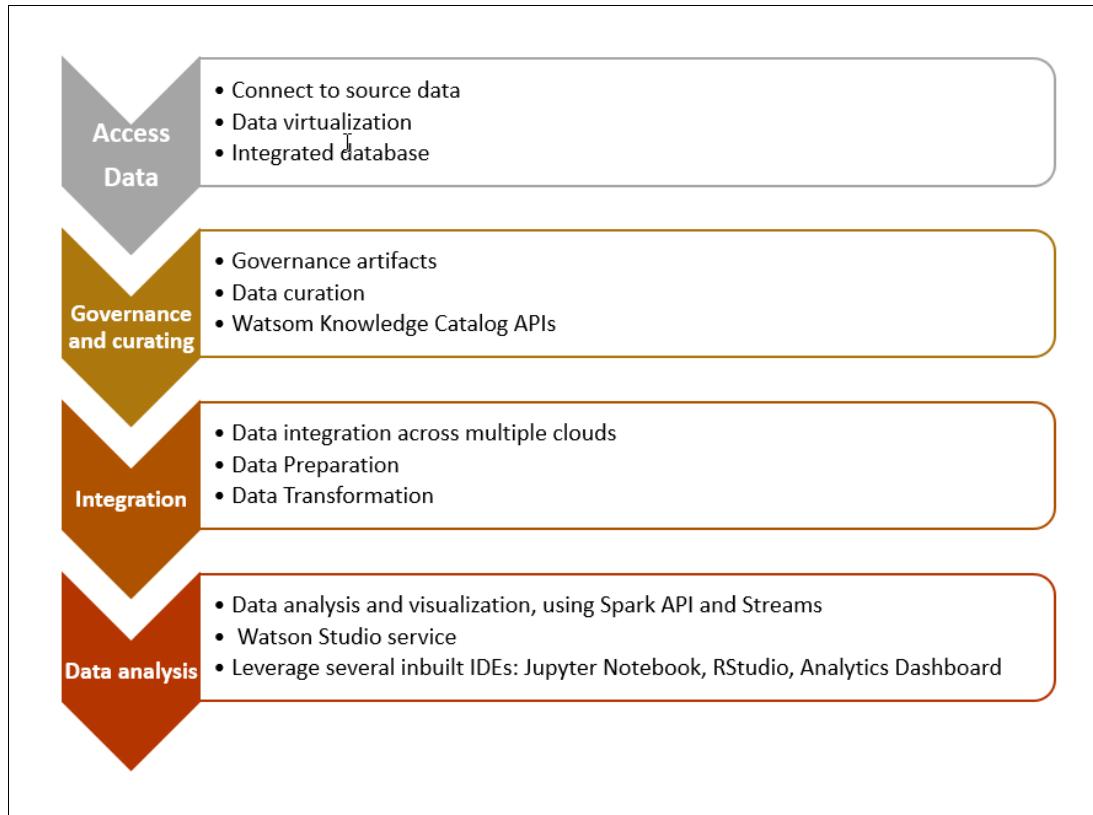


Figure 3-3 IBM Cloud Pak for Data: Data transformation phases

Accessing data

IBM Cloud Pak for Data is integrated with a wide range of data source connectors that ensure that data can be fetched from any possible sources for all possible activities. It can be integrated with third-party sources like Amazon, Apache, Google, Azure, Teradata, Salesforce, and others.

Virtualization of data is the process that combines data from one or more similar tables from multiple sources into a table view, as shown in Figure 3-4 on page 23. It creates a unified definition containing columns and data from all participating data sources. Instead of fetching data from multiple tables, the user can take advantage of this virtualized data to query information.

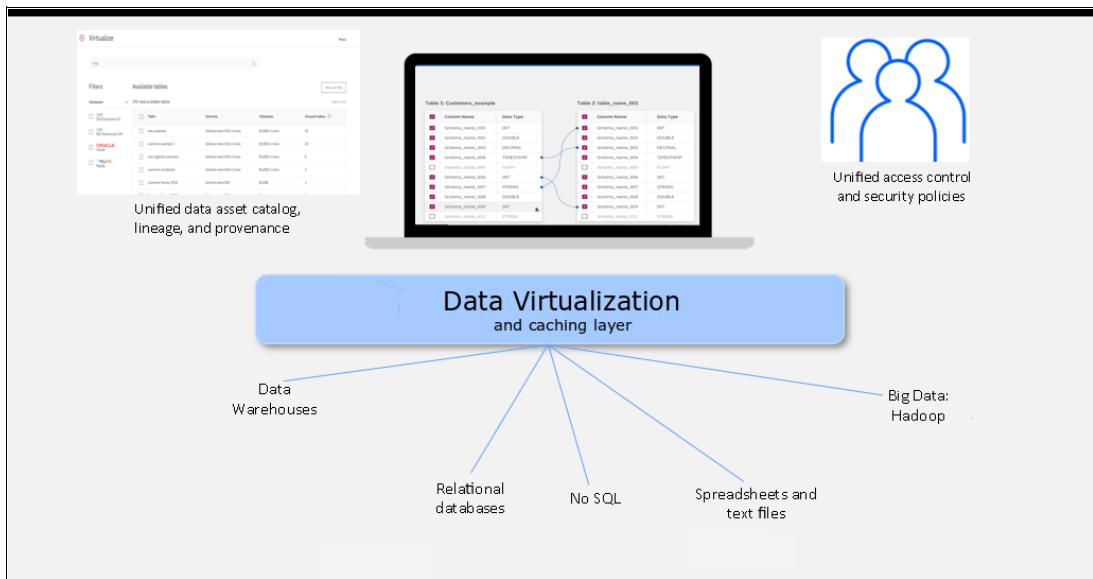


Figure 3-4 IBM Cloud Pak for Data: Data source control

IBM Watson Studio and IBM Watson Knowledge Catalog provide data access for analytical activities.

Governing and curating

This service is available under Watson Knowledge Catalog, which works as a centralized repository for an organization to maintain several assets for data science, analytics, and others. Driven by data protection rules, access control-based governance authorizes a user to access only the relevant data set.

Governance artifacts drive most processes like curating, enriching, and controlling of data. A high-level view of governance artifacts and tools is shown in Figure 3-5.

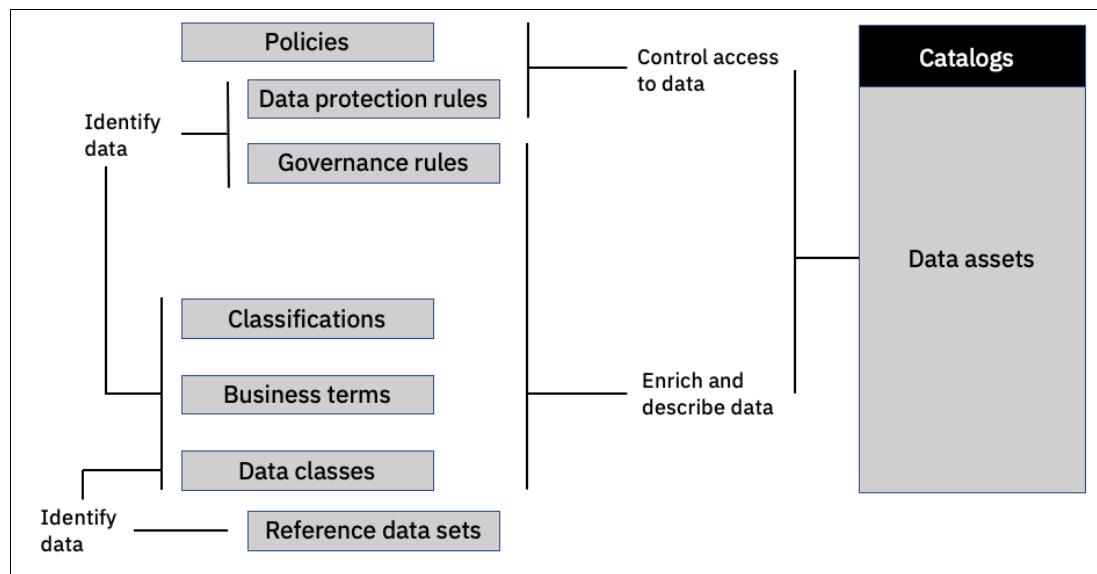


Figure 3-5 Data governance by using IBM Cloud Pak for Data

After data is fetched from a source, the next step is *curation*, where the data's quality is ensured before it is consumed for analytical activities. This process is defined and structured for deeper analysis to explore insights about data quality.

Integrating

After data is accessible and secured with its required governance, then it is time for the integration of data from multiple clouds and prepare it for transformation. Industry leading DataStage Edition services are core components in this offering that develop jobs to create, edit, load, and transform jobs. Here are some of the key built-in features from DataStage Edition Service:

- ▶ Built-in search.
- ▶ Automatic metadata propagation: Jobs are migrated to different stages and then into the job.
- ▶ Simultaneous highlighting of all compilation errors: Multiple errors can be addressed before recompilation.

The data refining flow is the process that cleanses and shapes data by using R libraries by performing ordered operations on data. The supported data formats are tables from relational data, and several unstructured sources like Avro, CSV, JSON, Parquet, and text.

Data analysis

Watson Studio is an integral component for data analysis and visualization. Its most often used services are Jupyter Notebook, RStudio, and Watson Studio Stream flows. Watson Studio is not available by default, so you must install it on the IBM Cloud Pak for Data platform.

The following services also can be used for data analysis:

- ▶ IBM Cognos Analytics: Create compelling visualizations and dashboards without needing a data science background.
- ▶ Analytics Engine powered by Apache Spark: Run analytic workloads with Spark APIs.
- ▶ Watson Studio Streams: Ingest and analyze streaming data.

Some of the integral components for IBM Cloud Pak for Data are explained in the following sections.

Jupyter Notebooks in Watson Studio

A Jupyter Notebook is a web-based environment for interactive computing to run small pieces of code that processes user data and generates visualization results. Jupyter Notebooks come with all the required plug-ins for data processing:

- ▶ Data
- ▶ Data processing services
- ▶ Visualizations service
- ▶ Text and rich media to enhance understanding

Jupyter Notebook is integrated with following Watson tools:

- ▶ Notebook editor: The Notebook editor is largely used for interactive, exploratory data analysis programming and data visualization. The Notebook editor is more convenient for people starting with this platform.
- ▶ JupyterLab: JupyterLab offers an integrated developer environment (IDE)-like development interface that includes Notebooks. The modular structure of the interface is extensible and open to developers so that they can work with several open Notebooks or files in tabs in the same window. JupyterLab is a high-performance IDE for creating and running Python Notebooks.

RStudio

R is a popular statistical analysis and machine-learning package that includes tests, models, analyses, and graphics, and enables data management. RStudio provides an IDE for working with R.

RStudio (Watson Studio) includes an extension for accessing a Git repository so that you can work in repository branches. Integration with the Git repository is recommended for better backup, collaboration, and organization of files.

Apache Spark

Apache Spark is an optional installable service to compute and run machine learning jobs. A dedicated Spark cluster is created to support each new job that is submitted. Some of the configurable objects are the size of the Spark driver, the size of the executor, or the number of executors for the job. With this service, users can achieve predictable and consistent performance.

The cluster is automatically cleaned up when the associated job is terminated to ensure that resources are available for other jobs. The service also includes interfaces to analyze the performance of Spark applications and debug problems. Jobs can be submitted to Spark clusters in two ways:

- ▶ Specifying a Spark environment definition for a job in an analytics project.
- ▶ Running Spark job APIs.

Cognos Dashboard

This service provides sophisticated visualizations of analytics results, and communicates the insights that you discover from data onto a dashboard. The supported data formats are:

- ▶ CSV files
- ▶ Data in tables in Db2 Warehouse, Db2 Warehouse on Cloud, PostgreSQL, IBM Netezza® (IBM PureData® System for Analytics), and Microsoft SQL Server
- ▶ Data Virtualization assets

The data size can be of any volume, but columns in CSV files can be only 128 characters long.

Streams

Streaming applications meet the need for real-time data processing, and they are intended to run indefinitely. For example, consider a company whose product scans traffic sensors across the world to determine traffic patterns and trends. Because there is always a car passing by, there is a perpetual need to process data. The application that processes these large volumes of data must also be able to run for an indefinite amount of time.

After the Streams service instance is provisioned in IBM Cloud Pak for Data, users get an instance of Streams that is ready to deploy to monitor the streaming applications.

Here are some best practices to develop applications to analyze streaming data:

- ▶ Streams Flows from within a IBM Cloud Pak for Data project
Streams Flows is an editor that uses dragging for constructing simple Streams applications, so it is a great way to get running quickly.
- ▶ The Streams Python API from a Jupyter Notebook in an IBM Cloud Pak for Data project
A Jupyter Notebook is useful for prototyping and experimenting with data science models.
- ▶ The Streams Python API and any text editor or Python IDE
- ▶ Streams Processing Language (SPL)
SPL applications can use the full set of Streams capabilities. You can use Microsoft Visual Studio Code or Streams Studio as your editor.

3.4 IBM Cloud Pak for Multicloud Management

IBM Cloud Pak for Multicloud Management provides consistent visibility, governance, and automation for applications from on-premises to the edge.

With IBM Cloud Pak for Multicloud Management, the user gains multicloud visibility, governance, and automation. Capabilities include multicloud management for containers, full stack multicloud provisioning, and infrastructure and application monitoring for mixed workloads.

3.4.1 Overview

IBM Cloud Pak for Multicloud Management is an enterprise-grade, multicloud, and multicloud management solution that addresses the visibility, governance, and automation challenges of deployments with multiple clouds and clusters.

IBM Cloud Pak for Multicloud Management delivers secure automated operations and intelligent insight for events, applications, infrastructure, multicloud, and virtual machine (VM) management for applications on-premises or in the cloud.

Hybrid cloud is a cloud computing environment that uses a mix of on-premises, private cloud, and third-party public cloud services with orchestration between the multiple platforms.

The challenges of a hybrid cloud are described in the following subsections.

Visibility and control

Complexity and risk grow exponentially when multiple cloud environments are integrated together. The administrator has limited access to and a limited view of the entire platform, which makes support and management a major challenge for security, compliance, patches, and configuration management. Manual monitoring is not sufficient to address a heterogeneous and distributed environment. An automated, tool-based, and centralized management and monitoring system is needed to manage such a platform.

Compliance and governance

Manual validation to meet regulatory security baselines for security compliance and auditing requirements is a tedious, complex, and error-prone process that becomes even more so when dealing with a mix of heterogeneous systems in the cloud and on-premises. When configuration changes are made manually, these changes often go undetected, so processes are not repeatable, sharable, and reproducible, which they must be if you plan to pass a security audit.

Data security

Across increasingly complex and expansive hybrid cloud environments, data might be exposed to risk in transit and at rest. No single protective implementation can prevent all possible methods of compromise because the same information can be at rest and in motion at different points.

Scaling complications

Scaling up a service changes its capacity because of adjusting the number of pods. The service is HA, and some services can take advantage of more pods to increase processing capacity. Scale up services only if IBM Cloud Pak for Data is set up for medium or large production usage. When you scale up a service, any components or services that it requires are also scaled up.

To determine the scale of a service, check the number of replicas it has. If a service has more than one replica, its scale is medium.

You can set the scale of services by specifying the corresponding assembly name in the scaling command. Any services that are affected when the specified service is scaled to medium are listed.

Handling multiple cloud providers

To meet the unique needs of business and remain competitive in today's fast-moving environment, you might adopt infrastructure and solutions from many cloud vendors.

A hybrid, multicloud world is quickly becoming the new normal, but managing cloud-based services and data across multiple providers can feel overwhelming. With each set of cloud services with its own tools, a platform owner likely faces increased complexity and cost. New management solutions and delivery methods can help optimize performance and control costs, and provide quick cloud access and a secure mix of applications, environments, and data, whether they are inside the data center or in the cloud.

IBM Cloud Pak for Multicloud Management can manage Kubernetes clusters that are deployed on any target infrastructure, such as a customer's private data center or in a public cloud. IBM Cloud Pak for Multicloud Management includes IBM Cloud App Management to simplify monitoring applications across any cloud environment.

3.4.2 Multicloud management fundamentals

Around 85% of enterprises are in a multicloud environment. However, only a few of them have a defined multicloud strategy. IBM Cloud Pak for Multicloud Management is a digital consumption and delivery platform with integration and orchestration layers that support multiple technology stacks across a multivendor platform.

Stakeholders and users can support their cloud journey by:

- ▶ Optimizing cloud spending and usage
- ▶ Managing services mapping and dependencies
- ▶ Extending DevOps processes for traditional IT and cloud natives

3.4.3 Component view of IBM Cloud Pak for Multicloud Management

IBM Cloud Pak for Multicloud Management offers consistent delivery for event management, application and infrastructure management, and multicloud management of your applications on-premises, in the cloud, at the edge, or anywhere else. Figure 3-6 shows the operational view of IBM Cloud Pak for Multicloud Management.

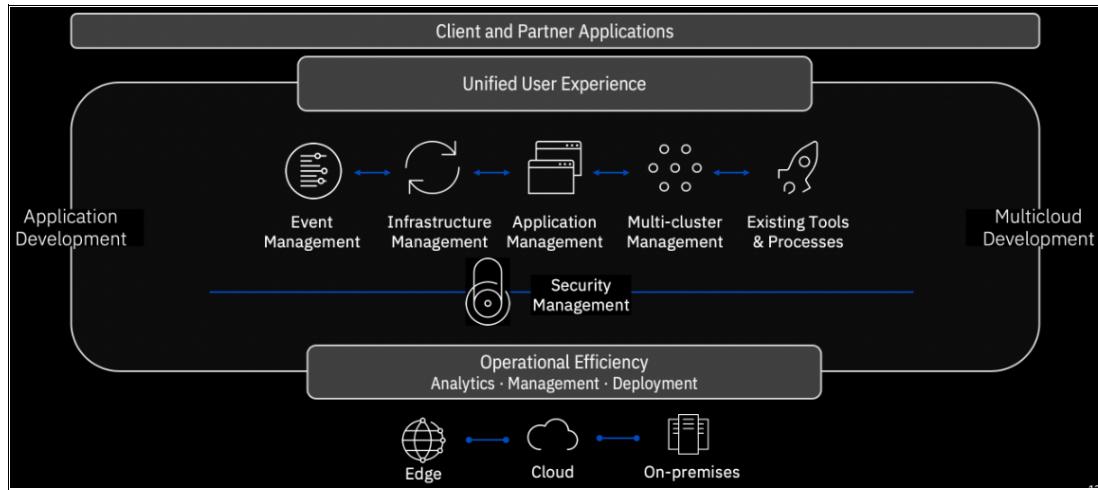


Figure 3-6 Operational view of IBM Cloud Pak for Multicloud Management

Event management

Events are generated from application, service, or a monitored object to indicate certain situations or status. In a steady state environment, the application is built over several pods and nodes. The event management capability plays a critical role in supporting operators to ensure the availability of the infrastructure. Timely error and failure reporting from multiple clusters followed by automated and guided remediation ensures high availability (HA) of the service. The event manager receives incidents from multiple sources either from on-premises or in the cloud.

Monitoring module

The performance and availability of both cloud and on-premises environments are the key features of this module. After the monitoring module is installed, then the following components must be integrated to capture the matrices from a multicloud environment:

- ▶ Agent: Placed on on-premises cluster-based application environments.
- ▶ Data Collector: Available on both local and remote platforms.
- ▶ Plug-ins for monitoring: Unified agents to monitor cloud resources. Agents collect, process, aggregate, and write matrices to monitoring modules.

This module monitors web-based resources and user experiences. This capability is known as digital experience monitoring (DEM).

Policies

Policies take more efficient action against incidents by using automation. Primarily, policies are divided into incident and event scenarios.

Incident policies perform activities such as:

- ▶ Assigning user groups.
- ▶ User notification.
- ▶ Escalation of aging tickets.

Event policies act on an event to:

- ▶ Provide more information during an event.
- ▶ Apply runbook-based resolution to implement a self-heal feature.

IBM Cloud Pak for Multicloud Management console: Govern risk

To see risk details about a policy in the IBM Cloud Pak for Multicloud Management console, select **Govern risk**, as shown in Figure 3-7.

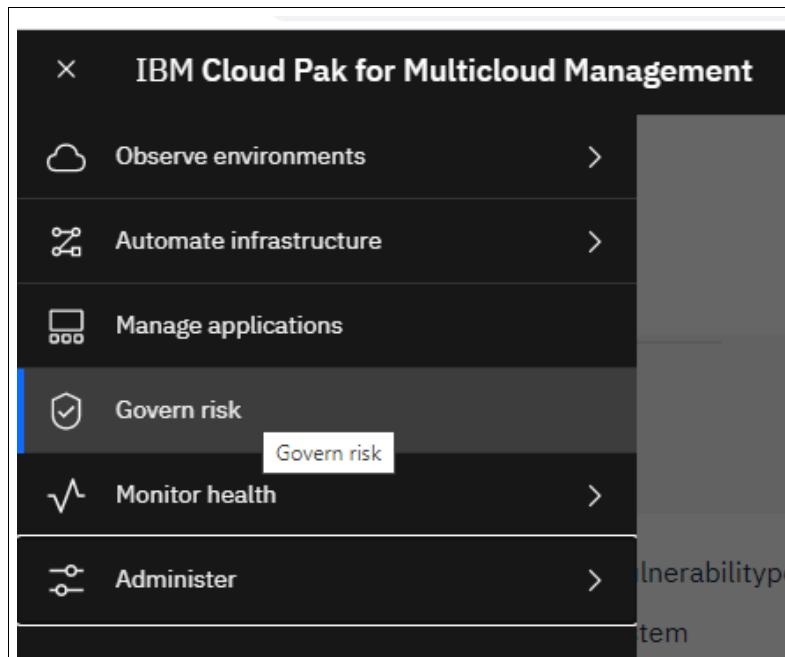


Figure 3-7 IBM Cloud Pack for Multicloud Management

Select a policy to see details about it, as shown in Figure 3-8.

The screenshot shows the 'Policy details' window for a policy named 'policy-vulnerabilitypolicy'. The window is divided into several sections:

- Policy details:** Shows basic information like Name (policy-vulnerabilitypolicy), Namespace (kube-system), Enforcement (inform), Exclude namespaces (kube-*), Include namespaces (default), Categories (DE.CM Security Continuous Monitoring), Controls (DE.CM-0 Vulnerability scans), and Standards (NIST-CSF).
- Placement:** Contains a 'Placement policy' section with fields: Name (placement-policy-vulnerabilitypolicy), Namespace (kube-system), Cluster selector (matchExpressions: [{"key": "cloud", "operator": "In", "values": ["MyDataCenter1", "MyDataCenter2"]}]), and a 'Decisions' section with a timestamp of 5 days ago. It also includes a 'Placement binding' section with fields: Name (binding-policy-vulnerabilitypolicy), Namespace (kube-system), Placement policy (placement-policy-vulnerabilitypolicy), Subjects (policy-vulnerabilitypolicy(policy.mcm.ibm.com)), and a timestamp of 5 days ago.

Figure 3-8 IBM Cloud Pak for Multicloud Management: Policy details window

Infrastructure management

The IBM Cloud Automation Manager solution automates provisioning of the infrastructure and VM applications across multiple cloud environments with optional workflow orchestration.

Application management

The application development and enhancement process is DevOps-based, unified, and simplified, and it is made more efficient by using the application management functions of IBM Cloud Pak for Multicloud Management. This capability is built on a Kubernetes resource-based application model, along with a channel- and subscription-based deployment model. The model unifies and simplifies application management across single and multicloud scenarios.

The application management capability uses a channel- and subscription-based model to optimize continuous and automated delivery in managed clusters. Application release management is automated through DevOps platform for operations like deployment, manage, and monitor.

Application model

Application development and deployment are two phases in a project's lifecycle. Application development is often restricted within fewer instances. However, in a production deployment, multiple instances are made available when scalability becomes a major factor. In a DevOps environment, roles can be defined for development and deployment. A development team must focus more on application development and defining application resources. A DevOps admin can set up a channels and subscription model for a faster, smoother, and more efficient deployment of the application to achieve high scalability in a managed cluster environment, as shown in Figure 3-9 on page 31.

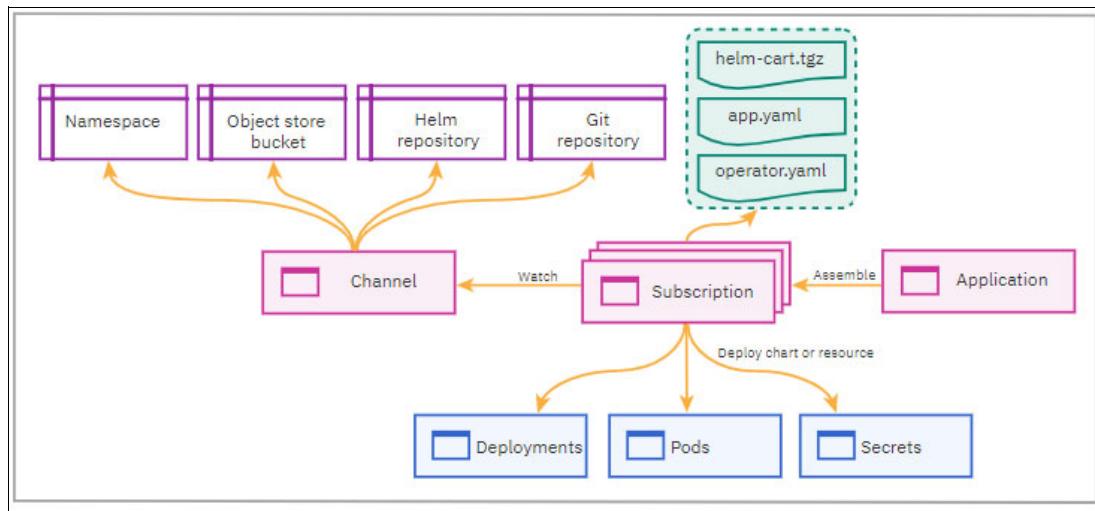


Figure 3-9 Application model for IBM Cloud Pak for Multicloud Management

Application resource

In IBM Cloud Pak for Multicloud Management, resources are classified as application resources and deployable resources. These resources are further divided into channel, subscription, and placement rule resources to facilitate deploying, updating, and managing applications that are spread across clusters. Both single and multi-cluster applications use the same Kubernetes specifications, but multi-cluster applications involve more automation of the deployment and application management lifecycle.

IBM Cloud Pak for Multicloud Management console: Manage applications

To browse application-specific resources in the IBM Cloud Pak for Multicloud Management console, see Figure 3-10.

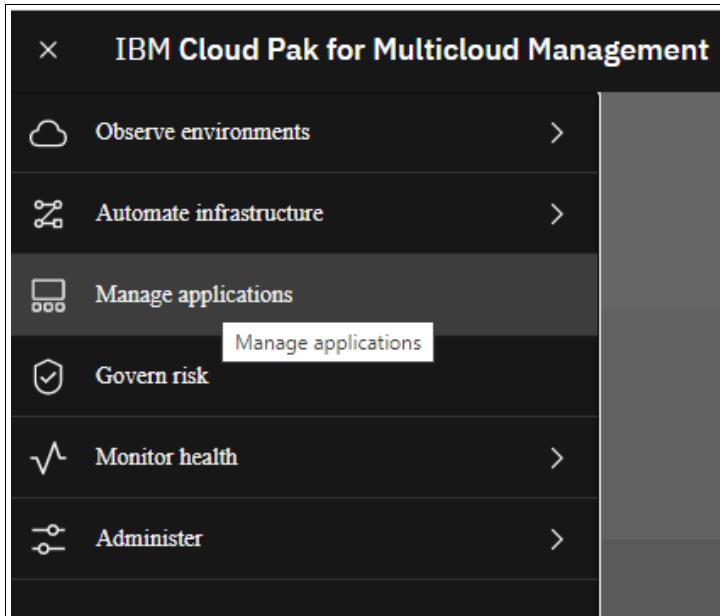


Figure 3-10 IBM Cloud Pack for Multicloud Management: Manage applications menu

A window opens that shows the **Resources** tab, which shows highlighted applications and four primary resources, for example, subscriptions, managed clusters, channels, and placement rules, as shown in Figure 3-11.

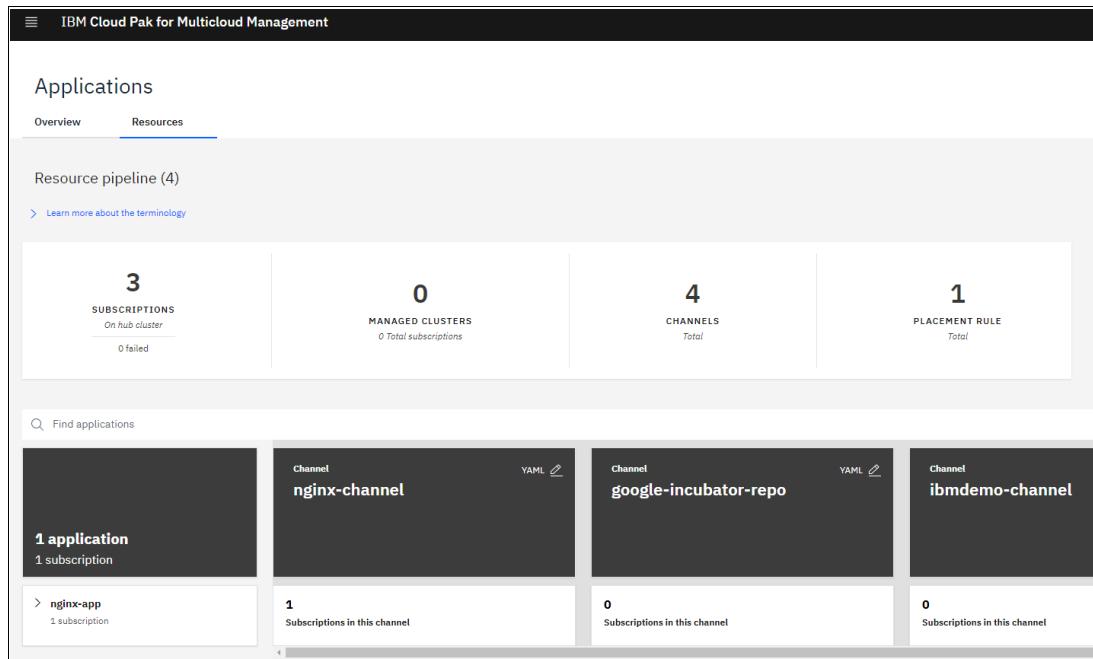


Figure 3-11 IBM Cloud Pak for Multicloud Management: Applications window

When you select a resource, you can view more detailed information. For example, you can select **Subscription**, and then **Related Cluster**, as shown in Figure 3-12.

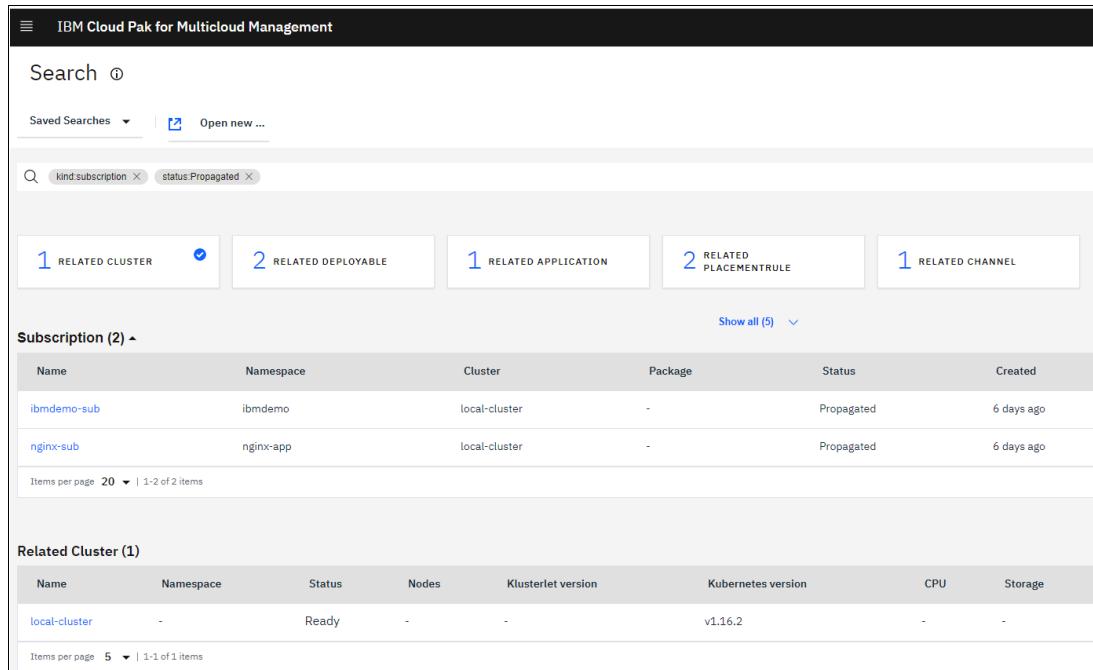


Figure 3-12 IBM Cloud Pak for Multicloud Management: Search window

Multicloud management

IBM Cloud Pak for Multicloud Management takes advantage of a Kubernetes core-based API for multicloud management. Some of the key activities are:

- ▶ Create.
- ▶ Import.
- ▶ Scale.
- ▶ Delete.

Using the IBM Cloud Pak for Multicloud Management console, you can view the clusters by selecting **Observing environments** → **Topology**, as shown in Figure 3-13.

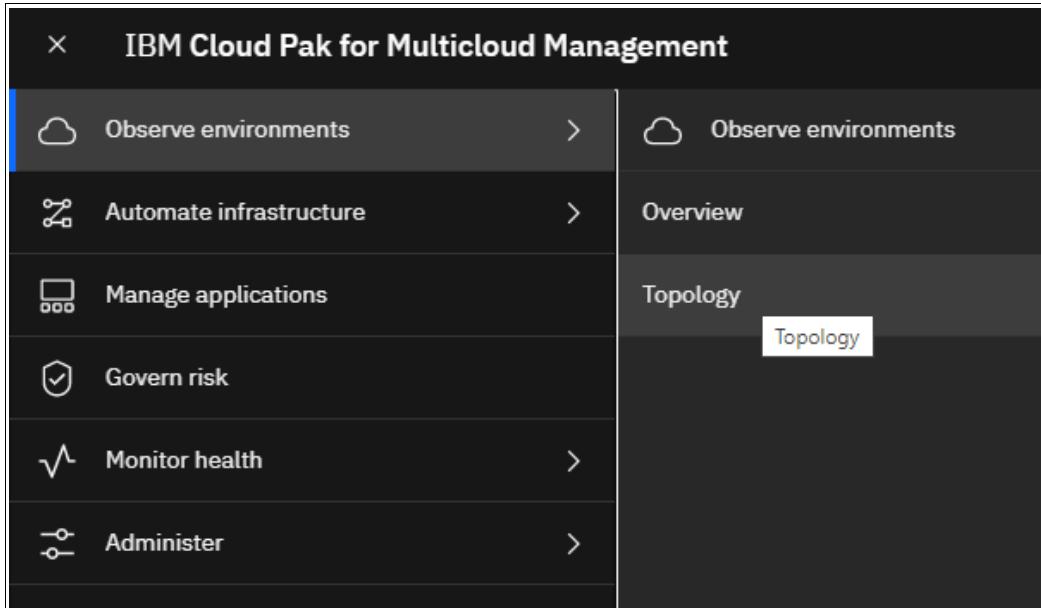


Figure 3-13 IBM Cloud Pak for Multicloud Management: Topology option

A window with information about all the available clusters opens, as shown in Figure 3-14.

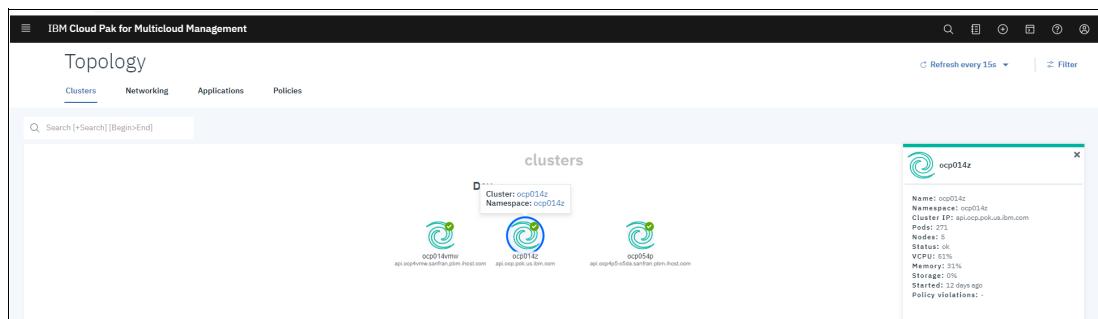


Figure 3-14 IBM Cloud Pak for Multicloud Management: Topology window

3.4.4 IBM Cloud Pak for Multicloud Management: Modules and features

An IBM Cloud Pak for Multicloud Management environment is integrated with several more modules to facilitate several critical capabilities. A high-level installation flow is shown in Figure 3-15.

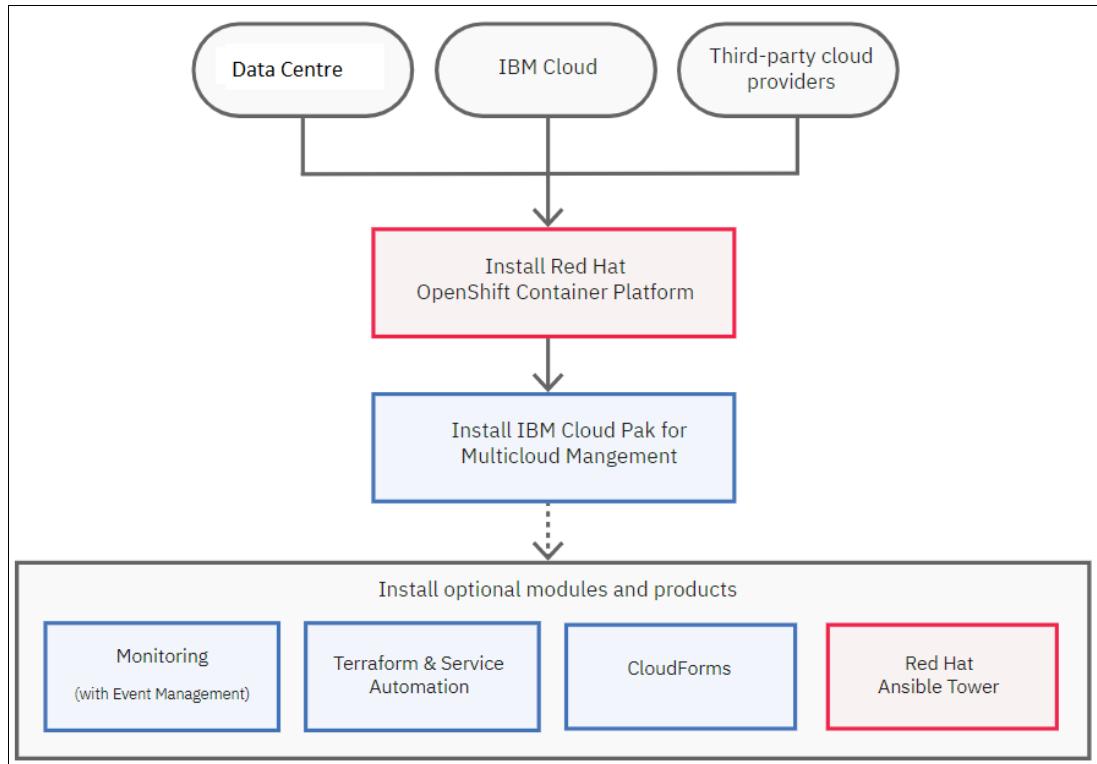


Figure 3-15 IBM Cloud Pak for Multicloud Management: Installation flow

3.5 Use cases: IBM Cloud Pak for Data

To understand IBM Cloud Pak for Data, we show a few use cases in this section.

3.5.1 Automated AI lifecycle: Organize and analyze data

Data science teams need integrated systems to manage assets across the AI lifecycle where AI models and the data that is associated with them is properly governed.

With IBM Cloud Pak for Data, a user can manage the end-to-end AI lifecycle from a single platform where all required services are installed that support the automated AI lifecycle. Some of the essential services are:

- ▶ IBM Watson Knowledge Catalog
- ▶ IBM Watson Studio
- ▶ IBM Watson Machine Learning
- ▶ IBM Watson OpenScale™

Some of the primary objectives from this integrated environment are described in the following subsections.

Data governance

Governance is the process of curating, enriching, and controlling data. Data governance is especially important in the context of the General Data Protection Regulation (GDPR). Governance artifacts must be organized into categories to manage the artifacts efficiently and effectively. Some governance artifacts are explained in the following subsections.

Classifications

Classifications are used to describe the sensitivity of the data in data assets. Each data asset has one classification. Catalog collaborators assign the classification when they add data assets to a governed catalog. Classifications are assigned to governance artifacts, such as business terms, data classes, reference data, policies, and governance rules.

Data classes

Data classes are used to categorize columns in relational data sets according to the type of the data and how the data is used. One data class is assigned to each column during profiling within a catalog. Catalog collaborators can change the data class that is assigned to a column. Users with the *Discover assets* permission can assign data classes to data set columns before adding the data to a catalog.

Reference data sets

Reference data sets are created to define values for specific types of columns. A reference data set is included in the definition of a data class as part of the data matching criteria. During data profiling, if the values in a column match the reference data set and other criteria, that data class is assigned to the column. Reference data sets are also used in data quality analysis.

Business terms

Business terms are created to define business concepts in a standard way for the enterprise. Catalog collaborators can assign one or more terms to data assets and columns within relational data sets to describe the data. Users with the *Discover assets* permission can add business terms to data sets before adding them to a catalog.

Policies

Policies are created to describe how to govern data in catalogs. Data protection rules are included in policies to control and manage data. Governance rules are included in policies to describe data.

Data protection rules

Data protection rules are created to identify the data to control and to specify the method of control. Within data protection rules, users can include classifications, data classes, business terms, or tags to identify the data to control. Users can also choose to deny access to data or to mask sensitive data values.

Governance rules

You create governance rules to provide a natural-language description of the criteria that are used to determine whether data assets are compliant with business objectives.

Figure 3-16 shows an overview of how governance artifacts and governance tools work together to describe, enrich, improve, and protect data.

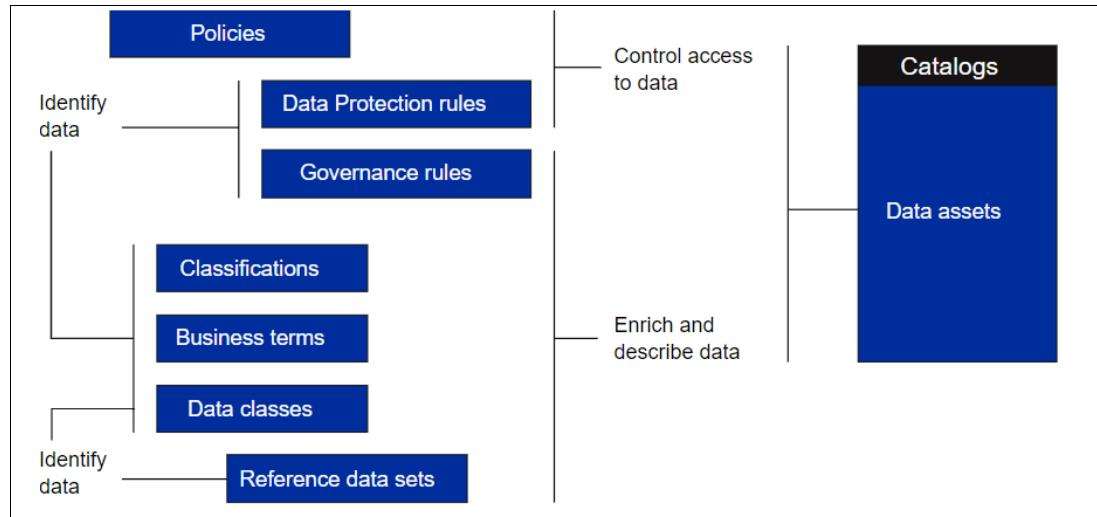


Figure 3-16 Governance artifacts and tools

Data transformation

Use the DataStage service to transform data to provide enriched and tailored information for your enterprise. DataStage is available as DataStage Enterprise and DataStage Enterprise Plus.

With DataStage, users can create, edit, load, and run transformation jobs. DataStage has features like built-in search, automatic metadata propagation, and simultaneous highlighting of all compilation errors. Developers can use these features to be more productive. DataStage Enterprise Plus comes with extra useful features for data quality:

- ▶ Cleansing data by identifying potential anomalies and metadata discrepancies.
- ▶ Identifying duplicates by using data matching and probabilistic matching of data entities between two data sets.

Building, deploying, managing, and scaling models

The AutoAI graphical tool in Watson Studio automatically analyzes user data and generates candidate model pipelines that are customized for a predictive modeling problem. These model pipelines are created iteratively as AutoAI analyzes a data set and discovers data transformations, algorithms, and parameter settings that work best for the identified problem setting, as shown in Figure 3-17 on page 37. The results are displayed on a leader board, showing the automatically generated model pipelines ranked according to the problem optimization objective.

The Watson Machine Learning Service analyzes source data that is available in comma-delimited CSV format, with a maximum of 1 GB.

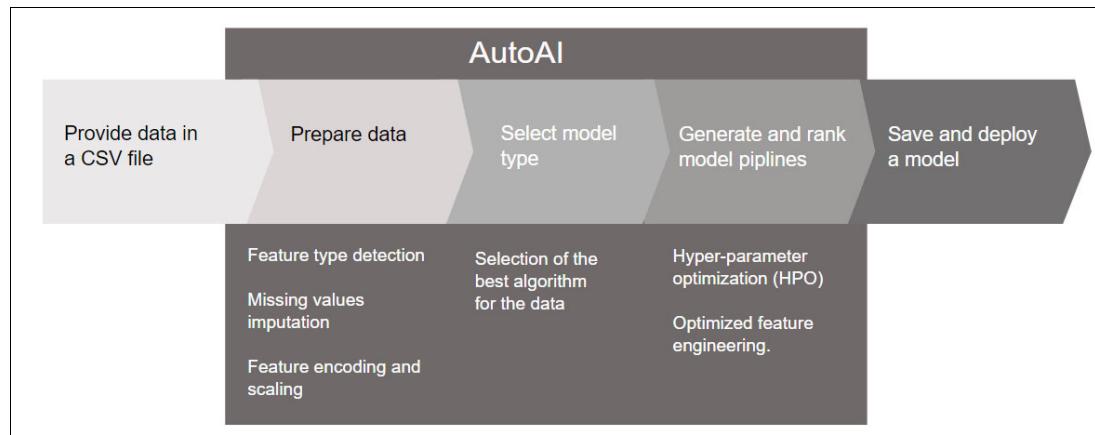


Figure 3-17 IBM Cloud Pak for Data scalability model

Data pre-processing

Most data sets contain different data formats and missing values, but standard machine learning algorithms work with numbers and no missing values. AutoAI applies various algorithms, or estimators, to analyze, clean, and prepare raw data for machine learning. AutoAI automatically detects and categorizes features based on data type, such as categorical or numerical. Depending on the categorization, it uses hyper-parameter optimization to determine the best combination of strategies for missing value imputation, feature encoding, and feature scaling for data.

Automated model selection

The next step is automated model selection that matches the provided data. AutoAI uses a novel approach that enables testing and ranking candidate algorithms against small subsets of the data, gradually increasing the size of the subset for the most promising algorithms to arrive at the best match. This approach saves time without sacrificing performance. It enables the ranking of many candidate algorithms and selecting the best match for the data.

Automated feature engineering

Feature engineering attempts to transform the raw data into the combination of features that best represents the problem to achieve the most accurate prediction. AutoAI uses a unique approach that explores various feature construction choices in a structured, non-exhaustive manner while progressively maximizing model accuracy by using reinforcement learning. This process results in an optimized sequence of transformations for the data that best matches the algorithms of the model selection step.

Hyperparameter optimization

Finally, a hyperparameter optimization step refines the best performing model pipelines. AutoAI uses a novel hyperparameter optimization algorithm that is optimized for costly function evaluations, such as model training and scoring that are typical in machine learning. This approach enables fast convergence to a good solution despite long evaluation times of each iteration.

Governing AI models

Data stewards and data quality analysts can govern data assets in catalogs by using governance artifacts in these ways:

- ▶ Protect sensitive information from unauthorized access by creating policies and data protection rules that deny access or mask data values in data assets. Policies and data protection rules apply to all catalogs with data protection that is enabled.
- ▶ Describe data to catalog users by associating classifications, data classes, business terms, and governance rules with data assets to help catalog users understand the data.
- ▶ Identify data for other governance artifacts. Classifications, data classes, and business terms can identify the data to protect in data protection rules. Reference data sets help match data classes to data columns, and manage standards for data consistency and quality.

Administrators can configure workflows for governance artifacts to require explicit approvals for new or updated artifacts, and detect and mitigate bias and drift in AI models.

Understanding how AI models make predictions

Predictive models can be quickly developed with SPSS Modeler flows in IBM Cloud Pak for Data. These models are developed by using business expertise and deploying them into business operations to improve decision making. Designed around the long-established SPSS Modeler client software and the industry-standard CRISP-DM model that the software it uses, the flow interface supports the entire data mining process, from data to better business results.

IBM Cloud Pak for Data offers many modeling methods that are taken from machine learning, AI, and statistics. Each method has certain strengths and is best suited for particular types of problems.

Using the Flow Editor, users prepare or shape data, train or deploy a model, or transform data and export it back to a database table or a file. To create an SPSS model, add the Modeler flow asset type to your project, then select **SPSS** as the flow type.



Solution scenarios

This chapter describes two solution scenarios that demonstrate how to use the technology that is available to the solutions.

This chapter contains the following topics:

- ▶ Installing Red Hat OpenShift in an OpenStack environment
- ▶ Banking architecture use case scenarios

4.1 Installing Red Hat OpenShift in an OpenStack environment

This section looks at three methods for installing Red Hat OpenShift in an OpenStack private cloud environment on Kernel-based Virtual Machine (KVM) based hosts. Deploying to an OpenStack environment is useful in a test and development environment where you can quickly stand up and tear down Red Hat OpenShift clusters.

Here are the three methods:

- ▶ Deployment by using an OpenStack Heat template.
- ▶ Using the Red Hat OpenShift installer for deployment.
- ▶ Deployment by using Terraform.

4.1.1 Deploying by using an OpenStack Heat template

OpenStack Heat Orchestration is a method for deploying resources and cloud applications by using an easy to read template format. The Heat Orchestration Template (HOT) format is the most commonly one that is used to deploy resources. For more information about HOT, see [Welcome to the Heat documentation](#).

The components of HOT are:

- ▶ Template: The definition of a stack as code.
- ▶ Stack: A collection of resources.
- ▶ Resources: A collection of objects that is created during the orchestration process. The objects can include networks, routers, subnets, instances, volumes, floating IP addresses, security groups, and Persistent Volumes (PVs).
- ▶ Parameters: They allow input to be provided during orchestration.
- ▶ Output: Information that is provided to the user during orchestration.

Deploying an Red Hat OpenShift cluster on an OpenStack cloud environment can be achieved by combining a HOT with the OCP4 Helper Node, which is used to set up an all-in-one infrastructure node that is used for the deployment of the Red Hat OpenShift cluster.

For more information about the Helper Node and what it does, see [GitHub](#).

The environment file and HOT can be downloaded from [GitHub](#) too.

Deploying a Red Hat OpenShift cluster

First, create an environment file. This file contains variables that are passed to the template. Then, define the heat template that creates all the resources for the Red Hat OpenShift cluster. Example 4-1 shows the environment file that is used.

Example 4-1 Heat environment file

```
parameters:  
  num_workers: - defaults to 3  
  cluster_name: - the name to call the cluster  
  openshift_version: defaults to 4.3.0.  
  worker_flavor: - defaults to m1.large  
  master_flavor: - defaults to m1.large  
  internal_network_cidr: - the CIDR definition for the internal subnet to be used, for example,  
    10.20.10.0/24
```

infrastructure_node_ip: - the IP address that is used for the Bastion node. This must be in the subnet above, for example, 10.20.10.5
internal_network_id: - This is the UUID of the internal project network that is used.
pullSecret: '' - A Red Hat subscription pull secret that is required for deploy OpenShift.
key_name: - The OpenStack keypair name that is used to deploy the Bastion node
public_key: '' - The public key of an OpenStack keypair used to deploy the instances.
availability_zone: - defaults to nova.

Note: The pull secret and public key must be in single quotation marks, and the default number of masters is coded into the template as 3 (Example 4-1 on page 40).

The heat template normally does not need to be changed, but you can customize it to suit your OpenStack environment. Example 4-2 shows a partial heat template.

Example 4-2 Partial heat template

```
heat_template_version: 2017-09-01

description: OCP cluster

parameters:
    num_workers:
        type: number
        description: Number of workers
        default: 3
    coreos_image:
        type: string
        default: "rhcosh-4.5.4-ppc64le"
        description: Base image for Red Hat CoreOS image to be used for the OpenShift
cluster
    cluster_name:
        type: string
        description: clustername
        default: "ocp"
    openshift_version:
        type: string
        description: Version of OpenShift V4 to install for example, 4.2.2 , 4.3.0
        default: "4.3.24"
    worker_flavor:
        type: string
        description: Flavor for work nodes
        default: "m1.large"
    master_flavor:
        type: string
        description: Flavor for master nodes
        default: "m1.large"
    centos_image:
        type: string
        default: "a18a8157-411d-476f-b703-a01152a2d135"
        description: Base image UUID for CentOS bastion image
    key_name:
        type: string
        description: SSH OpenStack keypair name to use
    public_key:
        type: string
```

```

    description: SSH OpenStack public key contents allows login to ocp nodes with
keypair.
    availability_zone:
        type: string
        description: The availability zone name to deploy to
        default: PowerKVM
    internal_network_cidr:
        type: string
        description: The CIDR of the internal network
    internal_gw_name:
        type: string
        description: The name of the internal network router
        default: ocp_router
    infrastructure_node_ip:
        type: string
        description: Address of infrastructure node on subnet
    dns_secondary:

```

Using the heat template

To deploy the heat template, you can either use the OpenStack Horizon UI or the OpenStack command-line interface (CLI).

The first example shows the UI method for deployment:

1. In the Horizon UI, select **Compute** → **Orchestration**. In the window that opens (Figure 4-1), select the **Environment File** and the heat template.

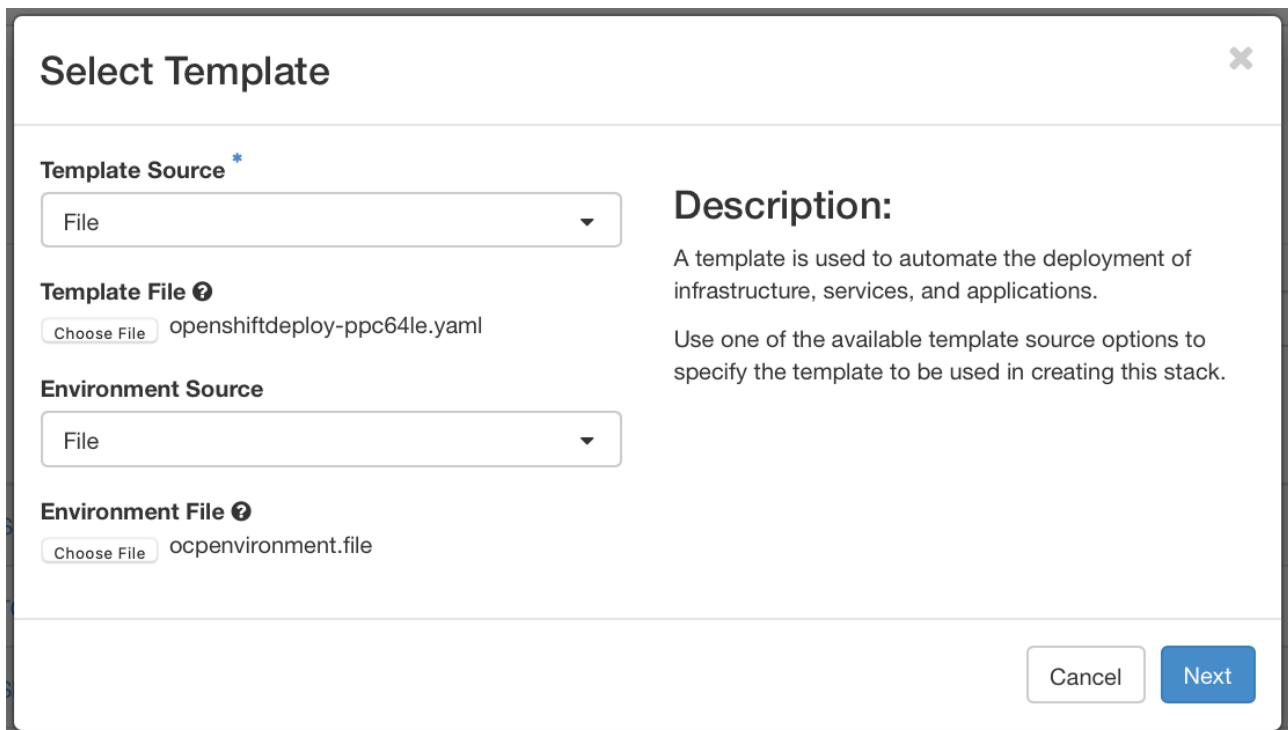


Figure 4-1 Starting the Red Hat OpenShift stack

2. Click **Next**. The stack starts, and you see its status as Create In Progress, as shown in Figure 4-2 on page 43.

<input type="button" value="Stack Name = ▾"/>	<input type="text"/>	<input type="button" value="Filter"/>	<input type="button" value="+ Launch Stack"/>	<input type="button" value="Preview Stack"/>	<input type="button" value="Delete Stacks"/>	<input type="button" value="More Actions ▾"/>
Displaying 20 items Next »						
<input type="checkbox"/> Stack Name			Created	Updated	Status	Actions
<input type="checkbox"/>	sg248486-osp-ocp45-ppc64e		3 minutes	Never	Create In Progress	<input type="button" value="Check Stack ▾"/>

Figure 4-2 Creating the Red Hat OpenShift stack

After the create process completes, you see the Red Hat OpenShift cluster in the instance list, as shown in Figure 4-3.

<input type="checkbox"/> Instance Name	Image Name	IP Address	Flavour	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/>	sg248486-osp-ocp45-ppc64e-master2	rhcoss-4.5.4-ppc64le	10.129.10.17	icp.platform	-	Active		PowerKVM	None	Running
<input type="checkbox"/>	sg248486-osp-ocp45-ppc64e-worker2	rhcoss-4.5.4-ppc64le	10.129.10.29	icp.platform.large	-	Active		PowerKVM	None	Running
<input type="checkbox"/>	sg248486-osp-ocp45-ppc64e-worker1	rhcoss-4.5.4-ppc64le	10.129.10.7	icp.platform.large	-	Active		PowerKVM	None	Running
<input type="checkbox"/>	sg248486-osp-ocp45-ppc64e-master1	rhcoss-4.5.4-ppc64le	10.129.10.18	icp.platform	-	Active		PowerKVM	None	Running
<input type="checkbox"/>	sg248486-osp-ocp45-ppc64e-bootstrap	rhcoss-4.5.4-ppc64le	10.129.10.6	m1.large	-	Active		PowerKVM	None	Running
<input type="checkbox"/>	sg248486-osp-ocp45-ppc64e-master0	rhcoss-4.5.4-ppc64le	10.129.10.5	icp.platform	-	Active		PowerKVM	None	Running
<input type="checkbox"/>	sg248486-osp-ocp45-ppc64e-worker0	rhcoss-4.5.4-ppc64le	10.129.10.15	icp.platform.large	-	Active		PowerKVM	None	Running
		10.129.10.10								
<input type="checkbox"/>	sg248486-osp-ocp45-ppc64e-infra	PKVM-CNT7.8-Srv-ppc64le	Floating IPs: 9.20.204.151	m1.large	sg248486-osp-key	Active		PowerKVM	None	Running
									10 minutes	

Figure 4-3 Red Hat OpenShift cluster in OpenStack

The heat template can also be started by using the OpenStack CLI by completing the following steps:

1. Authenticate by sourcing your openrc file, as shown in Example 4-3.

Example 4-3 CLI authentication

```
source openrc.v3
```

2. Start the heat template from the OpenStack CLI, where sg248486-osp-ocp44-ppc64e is the name of the stack, as shown in Example 4-4.

Example 4-4 Starting the heat template

```
openstack stack create -e ocpenvironment.file -t openshiftdeploy-ppc64le.yaml
sg248486-osp-ocp44-ppc64e
```

Note: When you want to remove the Red Hat OpenShift cluster, you must delete the stack to ensure that all the resources that are created during the stack execution are correctly removed.

4.1.2 Deploying by using the Red Hat OpenShift installer

The openshift-installer is supported on all major public cloud environments. By using it, you can deploy a Red Hat OpenShift cluster to your own private OpenStack environment.

There are many requirements that you must meet to deploy the installer. OpenStack Swift Object Storage is a requirement for the Red Hat OpenShift Installer. For more information, see [OpenStack Platform Support](#).

4.1.3 Deploying by using Terraform

Terraform is a cloud-neutral open source *infrastructure as code* tool that enables developers to use a high-level configuration language called Hashi Corporation Language (HCL) to describe an *end-state* cloud or on-premises infrastructure for running an application. Then, Terraform generates a plan for reaching that end-state and runs the plan to provision the infrastructure.

With Terraform, it is possible to create a Red Hat OpenShift cluster in an OpenStack environment. For more information, see this [GitHub repo](#).

4.2 Banking architecture use case scenarios

The banking industry is a network of financial institutions that offers various services. Retail banks provide critical services by accepting deposits and use these funds to make loans. In addition to retail banks, there are corporate and commercial banks, investment banks, custody banks, and various other financial institutions.

4.2.1 Requirements and challenges

Financial institutions have reliability and security at the forefront of their critical services, which might be why many of them rely on IBM Z and IBM Power Systems platforms for their core services.

Because the modern banking industry has extra challenges, its IT infrastructure and services must be more agile, flexible, and heterogeneous. Among these challenges are the constant need to stay current with new regulations and state-of-the-art business models. All these challenges require innovation and a set of new practices and processes, such as continuous integration (CI) and continuous deployment (CD) (CI/CD).

Thousands of banking customers already run their most mission-critical applications on IBM Power Systems running AIX or Red Hat Enterprise Linux. In addition to the scalability, reliability and security capabilities, an IBM Power Systems server is a flexible platform that can reduce total cost of ownership (TCO) and enable you to build an enterprise-wide hybrid cloud infrastructure.

By using Red Hat Enterprise Linux and the Red Hat OpenShift Container Platform, financial institutions can build open cloud-native applications with speed, agility, and security that can run anywhere with confidence.

While planning for an open-source hybrid cloud architecture in banking, you must consider the cost and effort that is required to migrate workloads; use proper security tools; and meet regulatory compliance, all while avoiding vendor lock-in.

Figure 4-4 on page 45 shows these considerations in more detail.



Figure 4-4 Planning for an open hybrid cloud

4.2.2 Private cloud, public cloud, or hybrid cloud for banking

In a private cloud model, the cloud infrastructure and resources are deployed on-premises and owned and managed by the organization. In a public cloud model, a company uses compute, network, storage, and application resources as services that are delivered by a cloud provider over the internet.

The hybrid cloud model represents the best of both worlds. You can run sensitive, highly regulated, and mission-critical applications and workloads or workloads with reasonably consistent performance and capacity requirements on a private cloud infrastructure. You can run less sensitive, more dynamic, or even temporary workloads (such as development and test environments for a new application) on the public cloud. Figure 4-5 shows the hybrid cloud model.

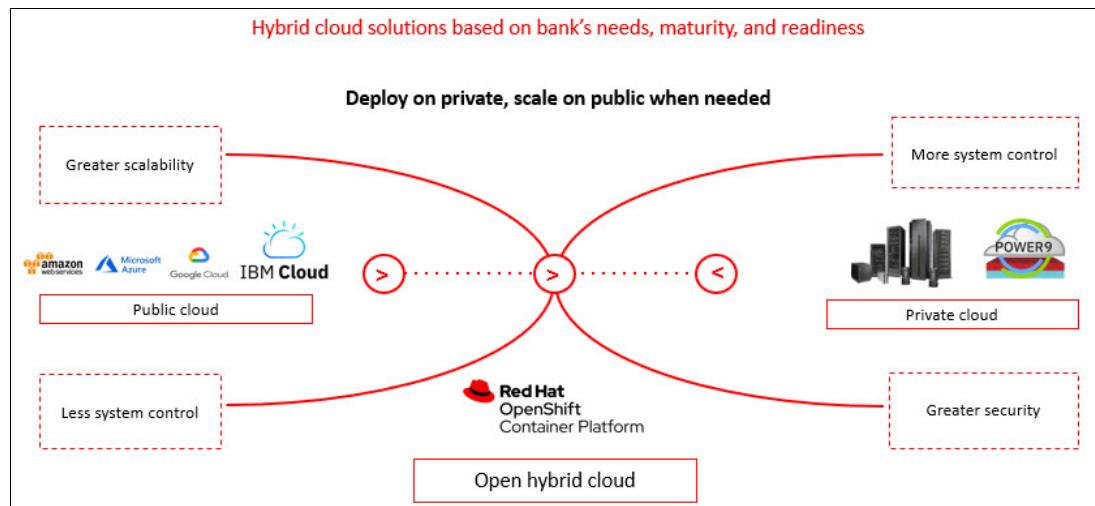


Figure 4-5 Cloud deployment models

4.2.3 Modernization and cloud journey for Power Systems

Business units and IT executives that are accustomed to an on-premises data center can find the prospect of upgrading traditional systems with an enterprise-level cloud solution to be complicated and time-consuming. Fortunately, banks can approach this transformation incrementally. They can mix and match open hybrid multi-cloud solutions based on their bank's needs, maturity, and readiness.

In general, the journey to cloud begins by managing within the traditional IT model, and then preparing for deployment of Red Hat OpenShift into the new cloud native world. The first step always starts with IBM Power Virtualization Center (IBM PowerVC) managing of your cloud infrastructure. It is the fundamental building block.

After you get onto an IBM PowerVC technology stack, a wealth of options opens up in terms of moving to Red Hat OpenShift or having an easier path to the public cloud. Opening up the possibility of migrating images between clouds is shown in Figure 4-6.

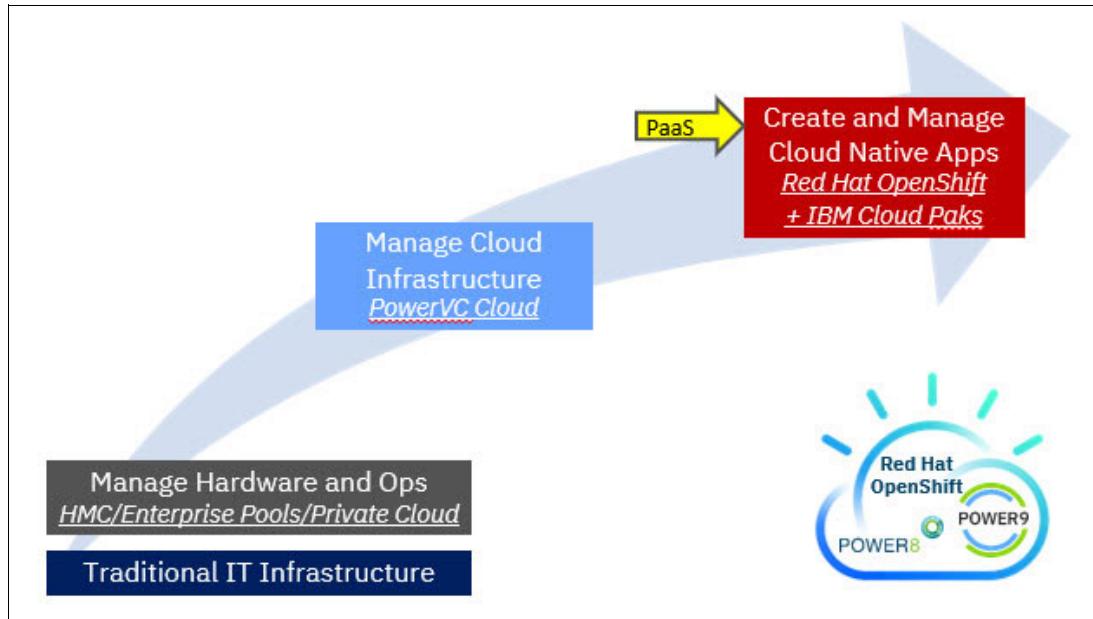


Figure 4-6 Moving to the cloud

The case scenarios that are presented here are based on the Red Hat OpenShift Container Platform running on IBM Power Systems with extra middleware software. Most of these components are provided by the following IBM Cloud Paks offerings (for more information, see Chapter 3, “IBM Cloud Paks” on page 15):

- ▶ IBM Cloud Pak for Data provides a suite of services that support your journey to artificial intelligence (AI) and machine learning.
- ▶ IBM Cloud Pak for Integration is a complete set of integration capabilities to connect efficiently your applications and data wherever they live.
- ▶ IBM Cloud Pak for Automation is a complete and flexible set of integrated automation software that can be deployed wherever you need it, on any cloud.

4.2.4 Benefits of open hybrid cloud for banking

There are many benefits to adopting an open hybrid cloud architecture for banking, as presented in Figure 4-7. The most significant of these benefits is the possibility of opening new business frontiers and optimizing the organization:

- ▶ **New business frontiers:**
Using new integration tools and capabilities and close alignment between business needs and developers help create markets and build customer experiences.
- ▶ **Optimizing the organization:**
Respond quickly to disruption and adapt computing capacity as needed, which saves time and money. Also, the state of the art tools keep the IT environments more secure and compliant.

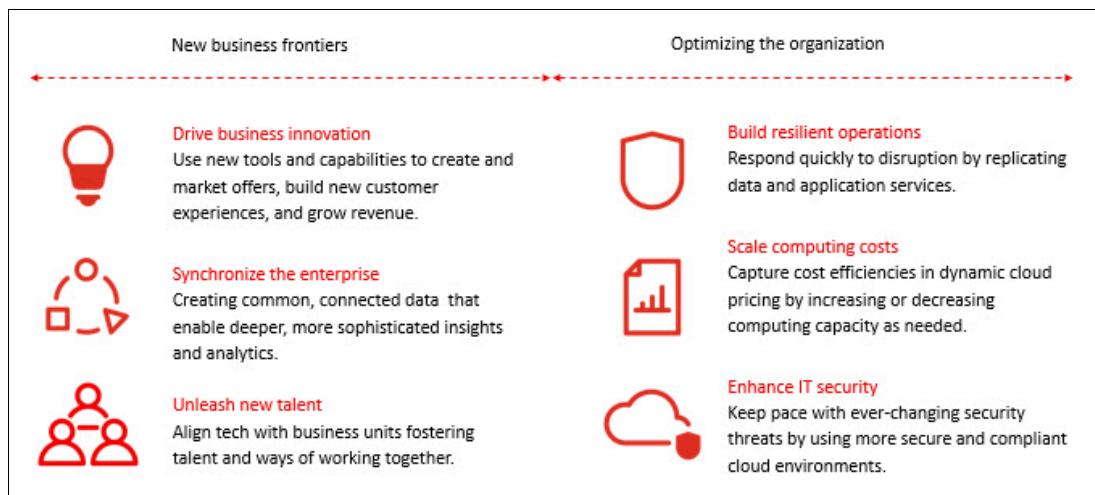


Figure 4-7 Benefits of open hybrid cloud for banking

For more information about banking technology on hybrid clouds, see [Do more for your customers with banking technology](#).

4.2.5 Use cases

The use cases in this section present architectures that are based on IBM Power Systems servers that are coupled with the Red Hat Enterprise Linux software portfolio to deliver foundation models to support the most demanding mission-critical workloads. The *business problem* that is targeted is illustrated by the following scenarios:

- ▶ Real-time payments (RTPs)
- ▶ Anti-money laundering (AML)
- ▶ Open banking
- ▶ Risk analytics

Real-time payments

Figure 4-8 presents a high-level design for an RTP configuration solution that is based on The Clearing House (TCH) organization. Known as *TCH RTP*, it represents a network that centralizes the United States' RTP system for some financial corporations.

For more information about TCH RTP, see [The Clearing House](#).

The solution is built with Red Hat OpenShift and extra middleware software that is provided by IBM Cloud Pak for Integration and IBM Cloud Pak for Automation.

Here are the components that are used to build the RTP use case:

- ▶ A - Red Hat OpenShift Platform: Cloud foundation for building and scaling containerized applications.
- ▶ B - Red Hat Fuse: Distributed, cloud-native integration bus.
- ▶ C - Red Hat AMQ streams: Apache Kafka on Red Hat OpenShift.
- ▶ D - Red Hat 3scale API Management: Application programming interfaces (APIs) for more straightforward and flexible cloud-native development.
- ▶ E - Red Hat Decision Manager: Platform for developing containerized applications that automate business rules and logic.
- ▶ F - Red Hat Data Grid: In-memory, distributed, and NoSQL data store solution.
- ▶ G - Red Hat Process Automation Manager: Platform for developing containerized applications that automate business decisions and processes.

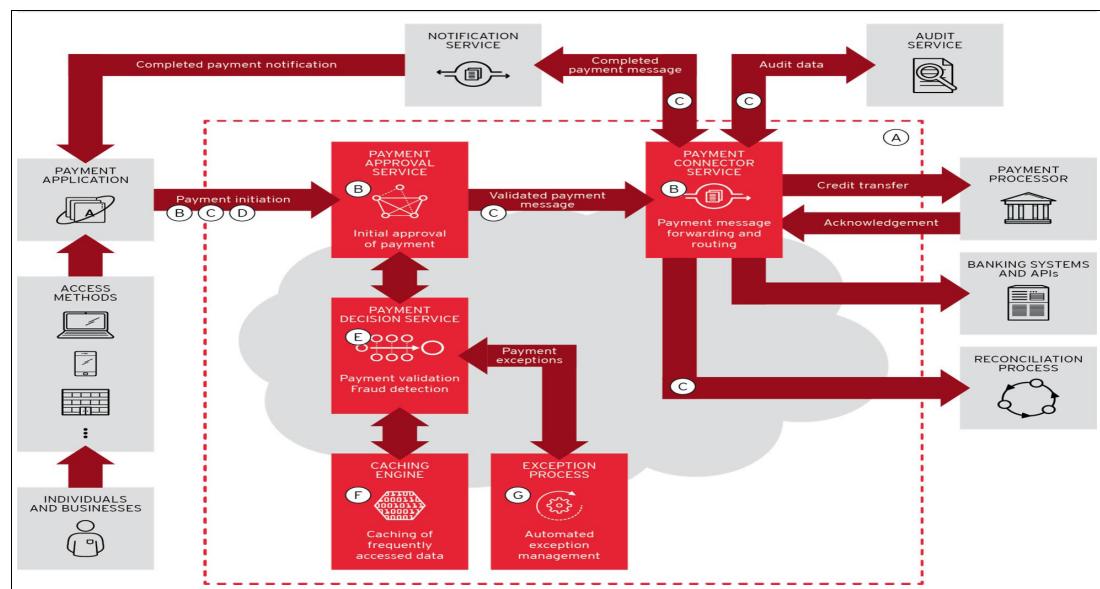


Figure 4-8 Real-time payments

For more information about RTP integration with TCH, see [IBM Knowledge Center](#).

Anti-money laundering

An AML solution helps financial institutions reduce exposure to money laundering. It is a set of tools, protocols, and compliance rules that keep the institutions from propagating money laundering transactions. Among the tools, you learn about the latest advancements in machine learning, cognitive analysis, AI, and robotic process automation.

Figure 4-9 presents a high-level design for an AML case that is based on the Red Hat OpenShift Container Platform and extra middleware software that is provided by IBM Cloud Pak for Data, IBM Cloud Pak for Integration, and IBM Cloud Pak for Automation.

Here are the components that are used to build the AML use case:

- ▶ A - Red Hat Fuse: A distributed, cloud-native integration bus (IBM Cloud Pak for Integration).
- ▶ B - Red Hat 3scale API Management: APIs for more straightforward and flexible cloud-native development (IBM Cloud Pak for Integration).
- ▶ C - Red Hat AMQ streams: Apache Kafka on Red Hat OpenShift (IBM Cloud Pak for Integration).
- ▶ D - Data Virtualization: Allows you to query data across many systems without having to copy and replicate data (IBM Cloud Pak for Data).
- ▶ E - Debezium: Open-source distributed platform that turns your existing databases into event streams so that applications can see and respond immediately to each row-level change in the databases. Debezium is built on top of Apache Kafka.
- ▶ F - Red Hat Decision Manager: Platform for developing containerized applications that automate business rules and logic (IBM Cloud Pak for Automation).
- ▶ G - Red Hat Process Automation Manager: Platform for developing containerized applications that automate business decisions and processes (IBM Cloud Pak for Automation).
- ▶ H - Red Hat Data Grid: In-memory, distributed, and NoSQL data store solution.
- ▶ I - Red Hat OpenShift Platform: Cloud foundation for building and scaling containerized applications.
- ▶ J - Red Hat OpenShift Container Storage: Software-defined storage (SDS) for containers.
- ▶ K - Open Data Hub: Open-source project that is based on Kubeflow, which provides open-source AI tools for running large and distributed AI workloads on Red Hat OpenShift Container Platform (IBM Cloud Pak for Data).

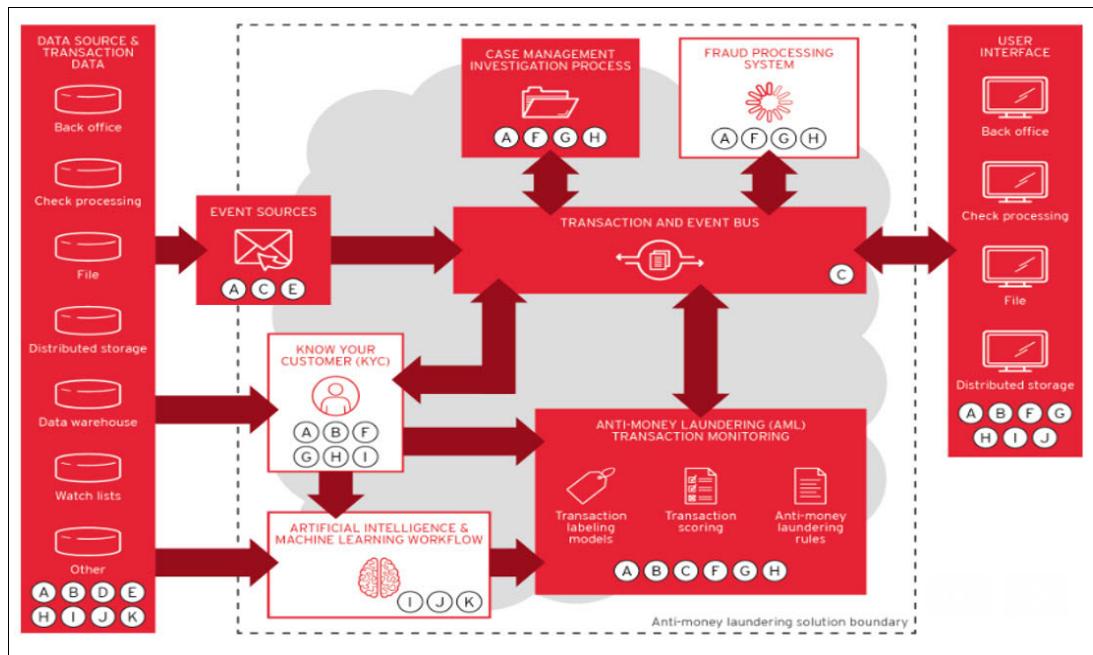


Figure 4-9 Anti-money laundering

For more information about modernizing AML compliance, see [Modernize AML compliance: How five institutions are preparing for the next decade of financial services](#).

Open banking

With open banking, consumers can access their financial data through multiple providers. Open banking is being adopted worldwide to generate new revenue streams through digital channels or to comply with regulatory mandates and directives for third-party access to bank data. Regardless of the reason, the solution architecture can be the same.

Figure 4-10 presents a high-level design for an open banking solution that is based on the Red Hat OpenShift and IBM Cloud Pak for Integration components.

Here are the components that are used to build the open banking use case:

- ▶ A - Red Hat OpenShift Platform: Cloud foundation for building and scaling containerized applications.
- ▶ B - Red Hat 3scale API Management: APIs for more straightforward and flexible cloud-native development (IBM Cloud Pak for Integration).
- ▶ C - Red Hat Fuse: Distributed, cloud-native integration bus (IBM Cloud Pak for Integration).

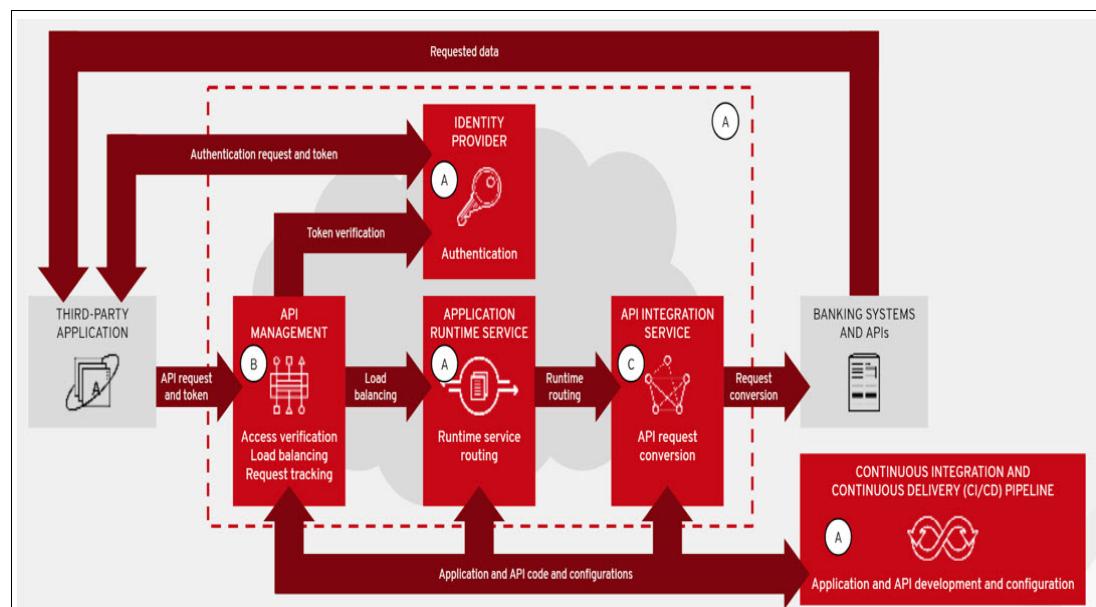


Figure 4-10 Open banking

For more information about IBM Cloud Pak for Integration, see [Automate integrations](#).

Risk analytics

In the financial services sector, complex transactions demand a considerable amount of real-time and data-intensive examination to provide elements to measure and process pricing and risk analysis. Financial sectors can face many types of risks, and the central business drivers for implementing risk analytics computing infrastructure are based on efficiency, profitability, and regulatory compliance.

Figure 4-11 on page 51 presents a high-level design for Risk Analytics solution that is based on Red Hat OpenShift, IBM Cloud Pak for Integration, and IBM Cloud Pak for Automation components.

Here are the components that are used to build the risk analytics use case:

- ▶ A - Red Hat Fuse: Distributed, cloud-native integration bus (IBM Cloud Pak for Integration).
- ▶ B - Red Hat Decision Manager: Platform for developing containerized applications that automate business rules and logic (IBM Cloud Pak for Automation).
- ▶ C - Red Hat OpenShift Container Storage: SDS for containers.
- ▶ D - Red Hat OpenShift Platform: Cloud foundation for building and scaling containerized applications.
- ▶ E - Red Hat Data Grid: In-memory, distributed, and NoSQL data store solution.

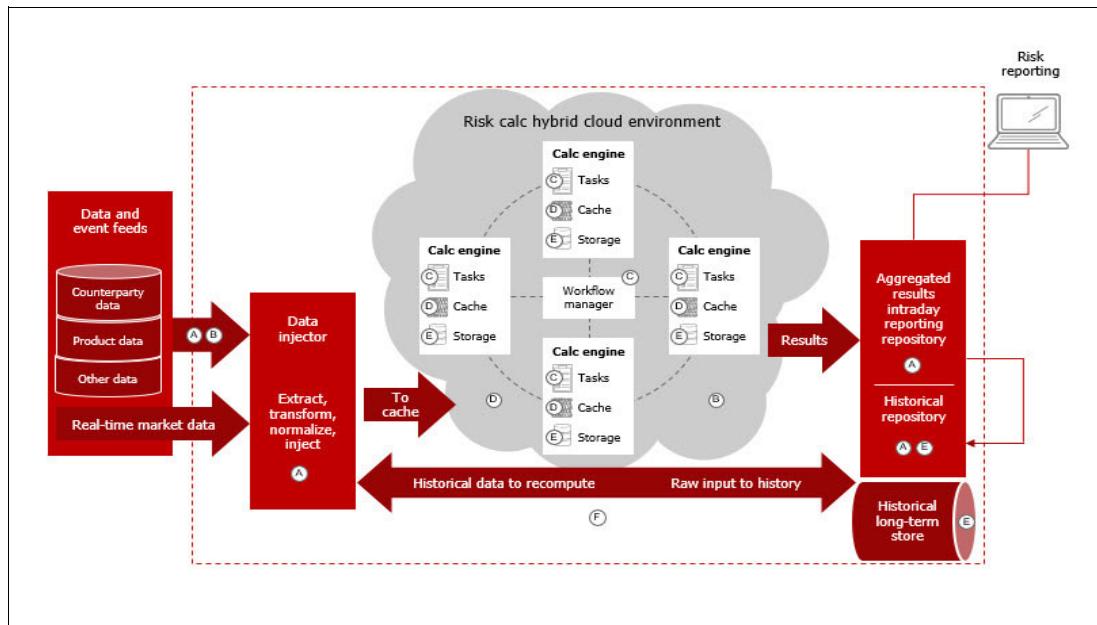


Figure 4-11 Risk analytics

For more information about risk analytics in banking services, see [Journey to AI Blog - The IBM Data and AI blog](#).



Red Hat OpenShift V4.x on IBM Power Systems cluster administration and monitoring

This chapter provides cluster administration and monitoring techniques for your cluster.

This chapter contains the following topics:

- ▶ Logging with Cluster Logging Operator
- ▶ Monitoring with Prometheus
- ▶ Monitoring the Hardware Management Console Performance Counter Monitoring data collection in Grafana dashboards

5.1 Logging with Cluster Logging Operator

This section describes the logging platform (Cluster Logging Operator (CLO)) that is used in Red Hat OpenShift Container Platform to search and analyze large data sets. This platform is used to find actionable insights from any type of data: logging or metrics from pods, nodes, services, and others.

5.1.1 Deploying cluster logging

Deploying these kind of products on Red Hat OpenShift Container Platform is easier than a deploying on a normal installation because most of the components that are involved in the cluster logging solution are cluster-based architectures. With these architectures, solutions can be maintained by the high availability (HA) component of the Red Hat OpenShift Container Platform cluster.

How it works

Red Hat OpenShift Container Platform administrators can deploy cluster logging by using operators. Operators are responsible for managing resources, and they provide interfaces for interaction with deployments, such as an application programming interfaces (APIs) to configure Elasticsearch.

Note: The procedure to deploy cluster logging can be different between minor releases. This chapter describes the architecture to understand how it works.

The cluster logging subsystem consists of the following multiple components.

Elasticsearch

Elasticsearch is a Lucene-based indexing object store into which logs are fed. Logs for node services and all containers in the cluster are fed into one deployed cluster. The Elasticsearch cluster must be deployed with redundancy and persistent storage for scaling and HA. For more information, see [Elastic](#).

Fluentd

Fluentd is responsible for gathering log entries from nodes, enriching them with metadata, and feeding them into Elasticsearch. For more information, see [Fluentd](#).

Kibana

Kibana presents a web UI for browsing and visualizing logs in Elasticsearch. It is used for creating dashboards, charts, and other data visualization features. For more information, see [Kibana](#).

To authenticate the Kibana user against Red Hat OpenShift Oauth2, a proxy is required that runs in front of Kibana.

Curator

With Curator, the admin can remove old indexes from Elasticsearch on a per-project basis.

These components are open-source products. To ease the deployment and administration of the system, you use CLO.

Cluster Logging Operator

The CLO provides a set of APIs to control collection and forwarding of logs from all pods and nodes in a cluster, including application and infrastructure logs. The operator does not collect or forward logs itself. The operator starts, configures, monitors, and manages the components that do the work. The installation must include the Elasticsearch operator subscription because it helps with the deployment and management of Elasticsearch.

Installation and documentation references

Red Hat OpenShift Container Platform V4.x represents a change in the way that Red Hat OpenShift Container Platform clusters are deployed, managed, and developed on. Therefore, the installation differs between releases. It is important to use the specific documentation for your cluster version, as shown in Table 5-1.

Table 5-1 Red Hat OpenShift Container Platform references

Version	Documentation reference
4.3	Understanding cluster logging and OpenShift Container Platform
4.5	Understanding cluster logging

5.2 Monitoring with Prometheus

Red Hat OpenShift Container Platform V4.x is shipped with a preinstalled monitoring stack that is based on the Prometheus open-source project. With this stack, you can monitor the cluster components. It includes a set of Grafana dashboards that you can use to look at metrics, and a set of pre-configured alerts to notify the cluster administrators of any issue when it occurs. The stack monitors only the Red Hat OpenShift Container Platform cluster.

The monitoring stack is deployed by the Cluster Monitoring Operator (CMO). This operator controls the deployed components, which ensure that the stack is up to date.

In addition to the CMO, the stack is composed of many other components:

- ▶ Prometheus operator: This operator is intended for creating, configuring, and managing Prometheus and Alertmanager instances.
- ▶ Prometheus: Prometheus is an open-source systems monitoring and alert toolkit. In this implementation, Prometheus is a service monitoring system on which the monitoring stack is based.
- ▶ Prometheus adapter: This component provides the cluster resource metrics API for horizontal pod autoscaling. The resource metrics are CPU and memory utilization.
- ▶ Alertmanager: This service handles alerts that are sent by Prometheus.
- ▶ Node-exporter: An agent that is deployed on every node to collect metrics about system information like resources utilization.
- ▶ Grafana: Grafana is an open source visualization and analytics software. You can use it to query, visualize, alert about, and explore metrics. The Grafana instance that is provided with the monitoring stack, along with its dashboards, display all the metrics that are collected by the Prometheus service, which are read-only.

5.2.1 Configuration

The configuration of the monitoring subsystem is explained in the Red Hat OpenShift documentation (Table 5-2) found at [Configuring the monitoring stack](#).

Table 5-2 Red Hat OpenShift Container Platform monitoring references

Version	Documentation reference
4.3	Configuring the monitoring stack for Version 4.3
4.5	Configuring the monitoring stack for Version 4.5l

5.3 Monitoring the Hardware Management Console Performance Counter Monitoring data collection in Grafana dashboards

This section explains how to obtain Hardware Management Console (HMC) metrics from the Performance Counter Monitoring (PCM) data collection API and display them in a Grafana dashboard. You use this process to monitor the cluster and obtain information directly from the Power Systems server about the virtual machines (VMs) where the cluster nodes and the load balancer are installed. To accomplish this task, you use three open-source tools: Grafana, InfluxDB, and Nextract.

5.3.1 How it works

Grafana is a tool that is used for displaying real-time monitoring dashboards. InfluxDB is used as a data source for the Grafana dashboards. InfluxDB is a time-series database that handles high-write and query loads. InfluxDB used as a backing store for any use case that involves large amounts of time stamped data, such as application metrics and real-time analytics. In this case, HMC PCM API data is used for populating the data source.

Note: To store built ImageStreams, the Red Hat OpenShift Container Platform internal registry must be already configured. For more information, see [Configuring the registry for bare metal](#).

These tools must be connected as shown in Figure 5-1 on page 57. In this case, Nextract CronJobs is used to insert the metrics that are obtained from the HMC into the InfluxDB database. This tool connects to the HMC API, collects the metrics, and inserts them into an InfluxDB database as described in [POWER8 Watts, Temp, SSP I/O, and Server/LPAR stats from HMC REST API - Version 10](#).

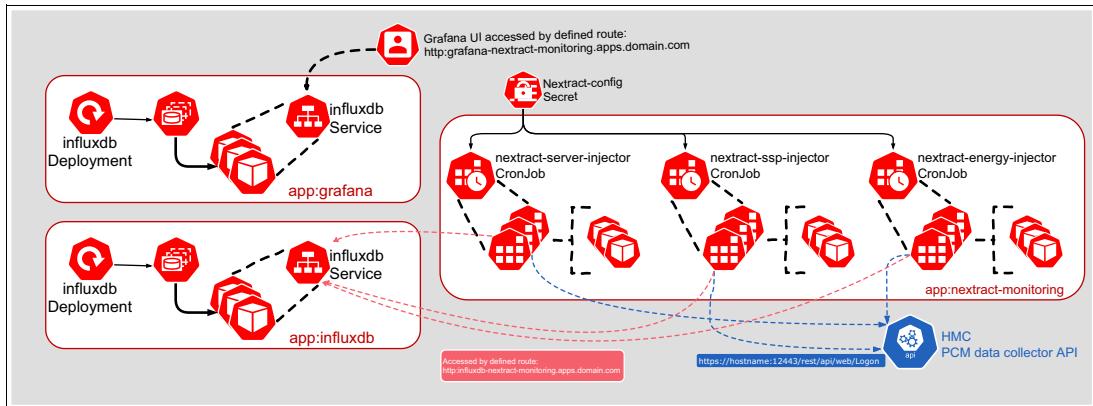


Figure 5-1 Tools that are connected for monitoring

5.3.2 Deploying components

This section describes how to deploy the components.

Building and deploying InfluxDB

To build and deploy InfluxDB, complete the following steps:

1. Create a project for the deployment by running the following command:

```
$ oc new-project nextract-monitoring
```
2. After you create the project, deploy InfluxDB. This component is directly deployed from a container image by running the **new-app** command:

```
$ oc new-app ibmcom/influxdb-ppc64le:v1.7.4 \
-e INFLUXDB_DB=nextract \
-e INFLUXDB_ADMIN_USER=admin \
-e INFLUXDB_ADMIN_PASSWORD=password \
-e INFLUXDB_USER=nextract \
-e INFLUXDB_USER_PASSWORD=password
```

Note: The **new-app** command is described in [Creating applications using the CLI](#).

The environment variables, which are assigned by using the **-e** prefix, are used to create users and initialize the database. A complete reference of the environment variables can be found in the [InfluxDB Docker repository](#). The official images of InfluxDB are not compiled for the IBM PowerPC® architecture. For that reason, an image that is compiled by IBM is used.

Tip: For this demonstration, the user credentials were defined by environment variables. In production, storing credentials that use secrets is highly recommended.

- The following command creates a basic cloud-native deployment of InfluxDB by using multiple Kubernetes objects. If some adjustments are needed, this file can be edited before applying it.

```
$ oc new-app ibmcom/influxdb-ppc64le:v1.7.4 \
-e INFLUXDB_DB=nextract \
-e INFLUXDB_ADMIN_USER=admin \
-e INFLUXDB_ADMIN_PASSWORD=password \
-e INFLUXDB_USER=nextract \
-e INFLUXDB_USER_PASSWORD=password \
-o yaml > influxdb.yaml
```

- After you run the command, a deployment, service, and ImageStream are defined in the output file, as shown in Example 5-1. This file can be modified as needed before placing it into the Red Hat OpenShift Container Platform cluster.

Example 5-1 Output file of InfluxDB cloud-native deployment

```
apiVersion: v1
items:
- apiVersion: image.openshift.io/v1
  kind: ImageStream
  metadata:
    annotations:
      openshift.io/generated-by: OpenShiftNewApp
    creationTimestamp: null
    labels:
      app: influxdb-ppc64le
      app.kubernetes.io/component: influxdb-ppc64le
      app.kubernetes.io/instance: influxdb-ppc64le
      name: influxdb-ppc64le
  spec:
    lookupPolicy:
      local: false
    tags:
      - annotations:
          openshift.io/imported-from: ibmcom/influxdb-ppc64le:v1.7.4
        from:
          kind: DockerImage
          name: ibmcom/influxdb-ppc64le:v1.7.4
        generation: null
        importPolicy: {}
        name: v1.7.4
        referencePolicy:
          type: ""
    status:
      dockerImageRepository: ""
- apiVersion: apps/v1
  kind: Deployment
  metadata:
    annotations:
      image.openshift.io/triggers:
        '[{"from":{"kind":"ImageStreamTag","name":"influxdb-ppc64le:v1.7.4"},"fieldPath":"spec.template.spec.containers[?(@.name==\"influxdb-ppc64le\").image"]}]'
        openshift.io/generated-by: OpenShiftNewApp
    creationTimestamp: null
    labels:
```

```

    app: influxdb-ppc64le
    app.kubernetes.io/component: influxdb-ppc64le
    app.kubernetes.io/instance: influxdb-ppc64le
    name: influxdb-ppc64le
  spec:
    replicas: 1
    selector:
      matchLabels:
        deployment: influxdb-ppc64le
    strategy: {}
    template:
      metadata:
        annotations:
          openshift.io/generated-by: OpenShiftNewApp
        creationTimestamp: null
        labels:
          deployment: influxdb-ppc64le
  spec:
    containers:
      - env:
          - name: INFLUXDB_ADMIN_PASSWORD
            value: password
          - name: INFLUXDB_ADMIN_USER
            value: admin
          - name: INFLUXDB_DB
            value: nextract
          - name: INFLUXDB_USER
            value: nextract
          - name: INFLUXDB_USER_PASSWORD
            value: password
        image: ''
        name: influxdb-ppc64le
      ports:
        - containerPort: 8086
          protocol: TCP
        resources: {}
      volumeMounts:
        - mountPath: /var/lib/influxdb
          name: influxdb-ppc64le-volume-1
    volumes:
      - emptyDir: {}
        name: influxdb-ppc64le-volume-1
  status: {}
- apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: influxdb-ppc64le
    app.kubernetes.io/component: influxdb-ppc64le
    app.kubernetes.io/instance: influxdb-ppc64le
    name: influxdb-ppc64le
spec:
  ports:
    - name: 8086-tcp

```

```
        port: 8086
        protocol: TCP
        targetPort: 8086
      selector:
        deployment: influxdb-ppc64le
      status:
        loadBalancer: {}
    kind: List
  metadata: {}
```

Tip: These object types and structures are used in cloud-native development to make applications cloud-ready. For more information, see [Kubernetes workloads](#).

5. Create objects by running the following command. Example 5-2 shows the output of the command.

```
$ oc create -f influxdb.yaml
```

Example 5-2 Object creation command and output

```
[user@workstation ~]$ oc create -f influxdb.yaml
imagestream.image.openshift.io/influxdb-ppc64le created
deployment.apps/influxdb-ppc64le created
service/influxdb-ppc64le created
```

6. Check that everything is correct by running the following command. Example 5-3 show the output of the command.

```
$ oc get pods -n nextract-monitoring
```

Example 5-3 Checking the configuration

```
[user@workstation ~]$ oc get pods -n nextract-monitoring
NAME                  READY   STATUS    RESTARTS   AGE
influxdb-ppc64le-69fbc667db-rrkd8   1/1     Running   0          1m16s
```

7. After the app is deployed, you must create a route to access the database when using Grafana and Nextract. You must make the InfluxDB service accessible. You can display the name by running the following command. Example 5-4 show the output of the command.

```
$ oc get services -n nextract-monitoring
```

Example 5-4 Displaying the InfluxDB service name

```
[user@workstation ~]$ oc get services -n nextract-monitoring
NAME            TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
influxdb-ppc64le   ClusterIP  172.30.189.235  <none>        8086/TCP   1m
```

To make the InfluxDB service accessible, run the following command. Example 5-5 shows the output of the command.

```
$ oc expose svc/influxdb-ppc64le
```

Example 5-5 Making the InfluxDB service accessible

```
[user@workstation ~]$ oc expose svc/influxdb-ppc64le
route.route.openshift.io/influxdb-ppc64le exposed
```

The command makes the service accessible and automatically generates a URL to access it (Example 5-5 on page 60).

8. To create objects and display the generated URL, which appears under the HOST/PORT column, run the following command. Example 5-6 shows the output of the command.

```
$ oc get routes -n nextract-monitoring
```

Example 5-6 Showing the service and generated URL

```
[user@workstation ~]$ oc get routes --label-columns=NAME,HOST/PORT
NAME          HOST/PORT
influxdb-ppc64le  influxdb-ppc64le-nextract-monitoring.apps.ocp44.ibmlab.com
```

Grafana

To deploy the Grafana server, complete the following steps:

1. Create an app for Grafana by running the following command. Example 5-7 shows the output of the command.

```
$ oc new-app ibmcom/grafana-ppc64le:5.2.0
```

Example 5-7 Creating the Grafana app

```
[user@workstation ~]$ oc new-app ibmcom/grafana-ppc64le:5.2.0
imagestream.image.openshift.io/grafana-ppc64le created
deployment.apps/grafana-ppc64le created
service/grafana-ppc64le created
```

2. If more configuration is needed, generate a definition file by running the following commands. Example 5-8 shows the output of the commands.

```
$ oc new-app ibmcom/grafana-ppc64le:5.2.0 -o yaml > grafana.yaml
$ oc create -f grafana.yaml
```

Example 5-8 Creating a definition file

```
[user@workstation ~]$ oc create -f grafana.yaml
imagestream.image.openshift.io/grafana-ppc64le created
deployment.apps/grafana-ppc64le created
service/grafana-ppc64le created
```

3. To log in to Grafana by using the GUI, extract the service and make it accessible by running the following commands. Example 5-9 and Example 5-10 show the outputs of these commands.

```
$ oc get services -n nextract-monitoring
$ oc expose svc/grafana-ppc64le
```

Example 5-9 Extracting the service

```
[user@workstation ~]$ oc get services -n nextract-monitoring
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)      AGE
grafana-ppc64le  ClusterIP  172.30.58.168  <none>        3000/TCP    101s
influxdb-ppc64le  ClusterIP  172.30.189.235 <none>        8086/TCP    34m
```

Example 5-10 Making the service accessible

```
[user@workstation ~]$ oc expose svc/grafana-ppc64le
route.route.openshift.io/grafana-ppc64le exposed
```

Nextract

To poll the metrics that are provided by the HMC, Grafana uses the Nextract Injector. Because the injector must run in short periods, it is a best practice to deploy the injector as a CronJob Kubernetes Object. The object schedules jobs to be run at defined intervals. Users can rely on jobs to ensure that the tasks (Nextract data polling) successfully complete. This way, Nextract injectors run in regular periods to obtain metrics and insert them into the database.

Note: The first thing that must be done is activating PCM data collection through the HMC, which is described in [IBM Knowledge Center](#).

Before creating CronJobs, you must create a build for the Nextract container image by using the Dockerfile that is found at [GitHub](#). This task provides a great opportunity to use the **new-build** command, which creates the ImageStream, ImageStreamTag, and BuildConfig. In this case, the command is used to clone the GitHub repository and compile the Dockerfile as an image:

```
$ oc new-build -to=nextract:1.0  
https://github.com/IBMRibooks/SG248486-Red-Hat-OpenShift-and-IBM-Cloud-Paks-on-IB  
M-Power-Systems-Volume2.git
```

Example 5-11 shows the output of the command.

Example 5-11 Cloning the GitHub repository

```
[user@workstation ~]$ oc new-build --to=nextract:1.0 \  
https://github.com/IBMRibooks/SG248486-Red-Hat-OpenShift-and-IBM-Cloud-Paks-on-IB  
M-Power-Systems-Volume2.git  
--> Found image c20dcc7 (2 months old) in image stream "openshift/python" under  
tag "3.6" for "python:3.6"  
    Python 3.6  
-----  
    Python 3.6 available as container is a base platform for building and running  
various Python 3.6 applications and frameworks. Python is an easy to learn,  
powerful programming language. It has efficient high-level data structures and a  
simple but effective approach to object-oriented programming. Python's elegant  
syntax and dynamic typing, together with its interpreted nature, make it an ideal  
language for scripting and rapid application development in many areas on most  
platforms.  
    Tags: builder, python, python36, python-36, rh-python36  
    * A Docker build using source code from  
https://github.com/IBMRibooks/SG248486-Red-Hat-OpenShift-and-IBM-Cloud-Paks-on-IB  
M-Power-Systems-Volume2.git will be created  
        * The resulting image will be pushed to image stream tag "nextract:1.0"  
        * Use 'oc start-build' to trigger a new build  
--> Creating resources with label build=nextract-openshift-monitoring ...  
    imagestream.image.openshift.io "nextract" created  
    buildconfig.build.openshift.io "nextract-openshift-monitoring" created  
--> Success
```

To stream the build's logs to stdout, specify the flag **--follow**. Example 5-12 on page 63 shows the output of the command.

```
$ oc new-build -to=nextract:1.0  
https://github.com/abgutierrez/nextract-openshift-monitoring.git --follow
```

Example 5-12 Logs to stdout

```
[user@workstation ~]$oc new-build --to=nextract:1.0 \
https://github.com/IBMRibooks/SG248486-Red-Hat-OpenShift-and-IBM-Cloud-Paks-on-IB
M-Power-Systems-Volume2.git --follow
build.build.openshift.io/nextract-openshift-monitoring-2 started
Cloning
"https://github.com/IBMRibooks/SG248486-Red-Hat-OpenShift-and-IBM-Cloud-Paks-on-I
BM-Power-Systems-Volume2.git" ...
Replaced Dockerfile FROM image python:3.6
Caching blobs under "/var/cache/blobs".
Pulling image
image-registry.openshift-image-registry.svc:5000/openshift/python@sha256:d8881affb
7666e9f1e5b718b27db89c41ab3b8ebef12510772bf221ee74895db ...
Getting image source signatures
Copying blob
sha256:e7021e0589e97471d99c4265b7c8e64da328e48f116b5f260353b2e0a2adb373
Copying blob
sha256:3e4d11cf9a3e6074fc46d8cd24d0c1149284fc505e3de95b036037e7da7a36cf
Copying blob
sha256:9e7a6dc796f0a75c560158a9f9e30fb8b5a90cb53edce9ffbd5778406e4de39
Copying blob
sha256:f659c5c779ac4373302bfe3dc7d713c59cf9ec9f179a71e9b26336a51043fad2
Copying blob
sha256:fc5b206e9329a1674dd9e8efbee45c9be28d0d0dcbabba3c6bb67a2f22cfef2a
Copying config
sha256:c20dcc77d7330c38bb14791f52837bf166dee585bddb6323988efb7079732a56
Writing manifest to image destination
Storing signatures
STEP 1: FROM
image-registry.openshift-image-registry.svc:5000/openshift/python@sha256:d8881affb
7666e9f1e5b718b27db89c41ab3b8ebef12510772bf221ee74895db
STEP 2: ENV version v10
--> c04b7f072c5
...
```

After you create the image, define a secret that contains the Nextract configuration. This configuration specifies the HMC user and password, and also the InfluxDB user, password, host, and port that is configured.

Example 5-13 shows the secret that was created for the Nextract configuration. Before you create the secret, you must specify all the information about the credentials and address. After you create the secret, mount it as a volume when defining the CronJobs.

Create the secret as shown in Example 5-13.

Example 5-13 Nextract configuration secret

```
[user@workstation ~]$oc create -f - << EOF
apiVersion: v1
kind: Secret
metadata:
  name: nextract-config
  namespace: nextract-monitoring
stringData:
  nextract_config.json: |-
```

```

{
  "hostname": "hmc.hostname",
  "user": "hmcUser",
  "password": "passwordHMC",
  "ihost": "influxdb-nextract-monitoring.apps.subdomain.domain.com",
  "iport": "80",
  "iuser": "admin",
  "ipassword": "SECRET2",
  "idbname": "nextract"
}
EOF
secret/nextract-config created

```

Note: The secret object type provides a mechanism to hold sensitive information such as the password, Red Hat OpenShift Container Platform configuration files, private source repository credentials, and other information. For more information, see [Providing sensitive data to pods](#).

Port 80 is used for accessing the database, and 8086 is used in the InfluxDB container configuration because all routed services that are defined in Red Hat OpenShift Container Platform are accessed through the load balancer. So, if the static URL is used to access InfluxDB, the configured ports in the load balancer must be used. Now, you can create the CronJobs.

The nextract_server.py injector: Performance stats for POWER8 onwards

The nextract_server.py injector extracts from the HMC the server-level CPU busy, memory allocated network, and storage I/O stats (read and write KBps, input/output per second (IOPS), and response times, and pool size/used and stats at the Virtual I/O Server (VIOS) level). In this example, only the VIOS I/O KBps is shown, which show the busiest VIOS in the shared storage pool. This injector covers managed system (server) stats and logical partition (LPAR) stats.

Example 5-14 shows the ImageStream that is created before scheduling CronJobs. This ImageStream is built and named by using the name of the cluster internal registry.

Example 5-14 Creating the ImageStream

```
[user@workstation ~]$oc create -f - << EOF
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: nextract-server-injector
  namespace: nextract-monitoring
label:
  app: nextract-monitoring
spec:
  schedule: "*/5 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          volumes:
            - name: config
              secret:
                secretName: nextract-config
EOF
```

```

        defaultMode: 420
    containers:
        - name: nextract-server
          image:
            image-registry.openshift-image-registry.svc:5000/nextract-monitoring/nextract:1.0
            volumeMounts:
                - name: config
                  mountPath: /scripts/nextract_config.json
                  subPath: nextract_config.json
                  command: ["python3", "./nextract_server.py"]
            restartPolicy: OnFailure
EOF
cronjob.batch/nextract-server-injector created

```

The nextract_ssp.py injector: For VIOS 2.2.5.10 or later

The nextract_ssp.py injector extracts from the HMC the overall shared storage pool I/O stats (read and write KBps, IOPS, and response times, and pool size/used plus VIOS level I/O KBps). Example 5-15 shows how the CronJob object uses the ImageStream to run the Python script.

Example 5-15 CronJob running the script

```

[user@workstation ~]$oc create -f - << EOF
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: nextract-ssp-injector
  namespace: nextract-monitoring
label:
  app: nextract-monitoring
spec:
  schedule: "*/5 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          volumes:
            - name: config
              secret:
                secretName: nextract-config
                defaultMode: 420
            containers:
              - name: nextract-ssp
                image:
                  image-registry.openshift-image-registry.svc:5000/nextract-monitoring/nextract:1.0
                  volumeMounts:
                      - name: config
                        mountPath: /scripts/nextract_config.json
                        subPath: nextract_config.json
                        command: ["python3", "./nextract_ssp.py"]
                restartPolicy: OnFailure
EOF
cronjob.batch/nextract-ssp-injector created

```

The nextract_energy.py injector

Example 5-16 shows how to extract from the HMC data about the watts of electricity and multiple temperatures in Celsius per server. For selected POWER8 servers or later:

- ▶ Supported: IBM Power Systems S8nn(L) and IBM Power System E850.
- ▶ Not supported: IBM Power System E870, IBM Power System E880, and IBM Power System LC models.

Example 5-16 Script extracting energy consumption and temperature data

```
[user@workstation ~]$oc create -f - << EOF
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: nextract-energy-injector
  namespace: nextract-monitoring
  labels:
    app: nextract-monitoring
spec:
  schedule: "*/5 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          volumes:
            - name: config
              secret:
                secretName: nextract-config
                defaultMode: 420
            containers:
              - name: nextract-energy
                image:
                  image-registry.openshift-image-registry.svc:5000/nextract-monitoring/nextract:1.0
                volumeMounts:
                  - name: config
                    mountPath: /scripts/nextract_config.json
                    subPath: nextract_config.json
                    command: ["python3", "./nextract_energy.py"]
                restartPolicy: OnFailure
EOF
cronjob.batch/nextract-energy-injector created
```

Note: Each CronJob works in parallel. This way, only the necessary injector scripts are used based on the kind of data that is needed. In addition, two CronJob instances, which are based on the same injector, can be used to monitor two different HMCs. To do so, you must create a secret with the new HMC credentials. Then, you configure the new CronJob to use this secret and mount the JSON file as a volume.

The secret that was previously created for the Nextract configuration file was bound. You bind a secret by mounting the secret as a volume in the deployment configuration.

5.3.3 Configuring the Grafana interface

After deploying all the components, you can configure Grafana. You must set a data source for getting data from the InfluxDB database, and then you must create some dashboards for displaying the data.

Complete the following steps:

1. After Grafana is running, you can access the web UI by using the URL that is generated by the route that is defined for Grafana. To list the routes, run the following command. Example 5-17 shows the output of the command.

```
$ oc get routes -n nextract-monitoring
```

Example 5-17 Listing the routes

NAME	HOST/PORT	PATH	SERVICES	PORT	TERMINATION	WILDCARD
grafana-ppc64le	grafana-ppc64le-nextract-monitoring.apps.ocp44.celestey.uy		grafana-ppc64le	3000-tcp	None	
influxdb-ppc64le	influxdb-ppc64le-nextract-monitoring.apps.ocp44.celestey.uy		nfluxdb-ppc64le	8086-tcp	None	

2. Define the admin password, as shown in Figure 5-2. The default user and password are admin and admin.



Figure 5-2 Entering the user ID and password

3. Configure the InfluxDB data source by creating a data source, as shown in Figure 5-3 and Figure 5-4 on page 69.

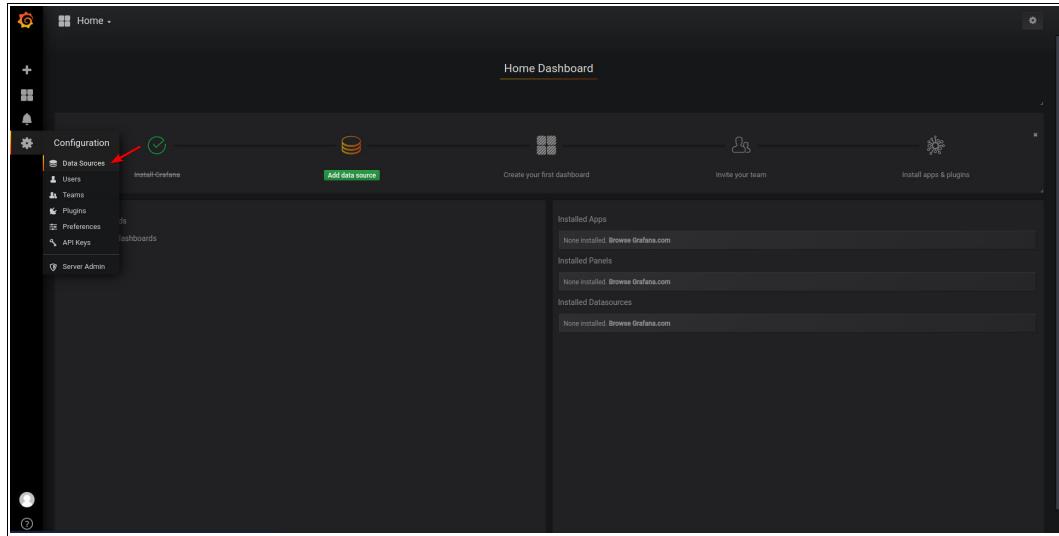


Figure 5-3 Creating a data source

The screenshot shows the 'Data Sources / nextract' configuration page in Grafana. At the top, there is a blue diamond icon and the text 'Data Sources / nextract' followed by 'Type: InfluxDB'. Below this, there are two tabs: 'Settings' (selected) and 'Metrics'. The 'Settings' tab contains the following fields:

Name	nextract	i	Default	<input checked="" type="checkbox"/>
Type	InfluxDB			

Below the settings, there are three sections: 'HTTP', 'Auth', and 'Advanced HTTP Settings'.

HTTP section:

URL	http://influxdb-ppc64le-nextract-monitoring.ap... i			
Access	Server (Default) Help ▾			

Auth section:

Basic Auth	<input type="checkbox"/>	With Credentials	i	<input type="checkbox"/>
TLS Client Auth	<input type="checkbox"/>	With CA Cert	i	<input type="checkbox"/>

Advanced HTTP Settings section:

Whitelisted Cookies	Add Name i
---------------------	----------------------------

InfluxDB Details section:

Database	nextract		
User	nextract	Password

Figure 5-4 Data sources for nextract

4. Set the name as nextract and the type as InfluxDB.

5. The URL of the data source is the one that is generated by the route that exposes the InfluxDB service. To get the URL, run the following command:

```
$ oc get routes
```

The port of the data source is 80, or 443 if you configured SSL in your cluster instance. Even though the InfluxDB service exposes port 8086, the route binds the external port of the cluster (80 or 443) to the internal service port, which in this case is 8086.

The database and the user and password credentials are the same ones that are defined for the environment variables of the InfluxDB deployment. In this case, the values are the ones that are shown in Table 5-3.

Table 5-3 Database credentials

Key	Value
Database	nextract
Username	nextract
Password	password

6. Click **Save & Test**, as shown in Figure 5-5. A message appears that states that the data source is working.

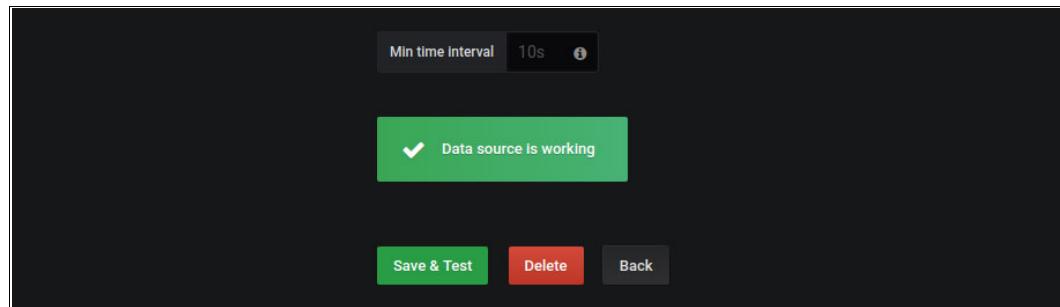


Figure 5-5 Testing the data source

Now, you import some dashboards to show the data, as shown in Figure 5-6. Some pre-configured dashboards are available to import. JSON files can be downloaded from [Power Systems HMC system view](#).

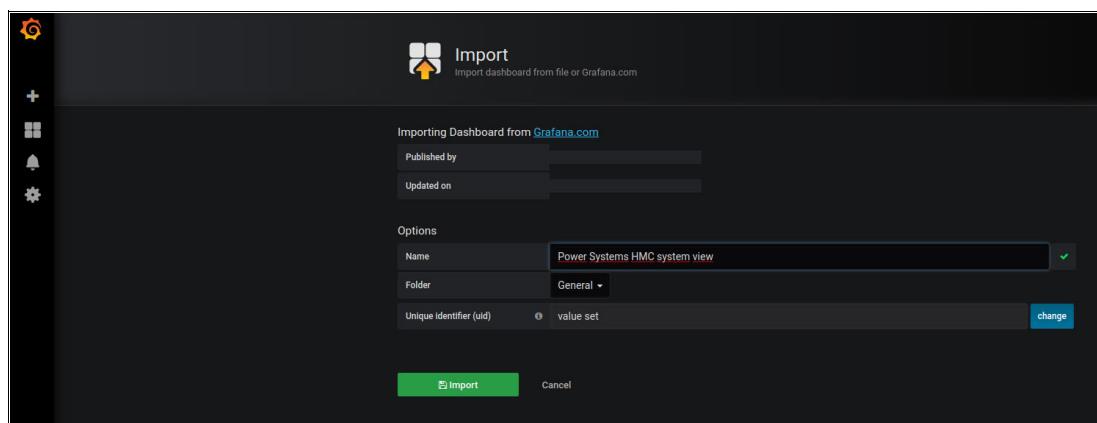


Figure 5-6 Importing dashboards

After the JSON files are downloaded, import them by using the Grafana GUI, as shown in Figure 5-7.

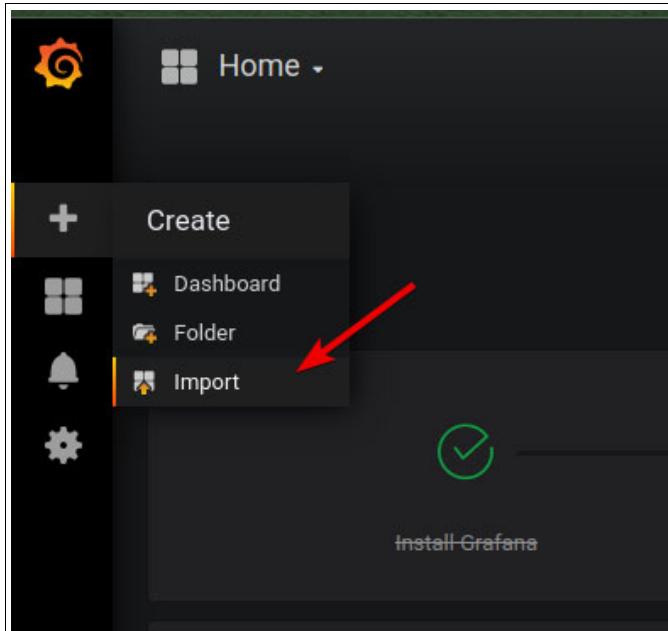


Figure 5-7 Importing the JSON files

Figure 5-8 shows the window for importing the files.

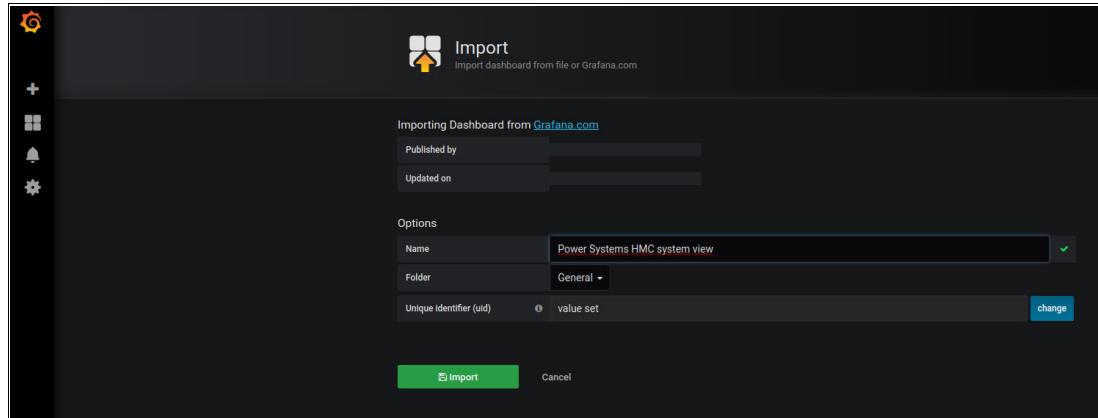


Figure 5-8 Importing the files

Figure 5-9 shows the thermal and energy metrics.



Figure 5-9 Showing the thermal and energy metrics

Figure 5-10 shows the networking metrics.



Figure 5-10 Showing the networking metrics



Deployment scenarios

This chapter describes several deployment scenarios that were tested during the development of this publication, some tips and tricks, and lessons that were learned by the team.

This chapter contains the following topics:

- ▶ Introducing deployment scenarios
- ▶ Deploying Red Hat OpenShift Container Platform V4.5 in an IBM Power Systems Virtual Server on IBM Cloud
- ▶ Bare metal deployment by using Red Hat Enterprise Linux CoreOS LPARs on a Virtual I/O Server
- ▶ Installing Red Hat OpenShift V4.3 and V4.4: Tips and tricks

6.1 Introducing deployment scenarios

This section focuses on tested scenarios that were implemented during the development of this publication, describes how flexible the solution is, and how the solution can improve your application deployment speed, resilience, and cost efficiency.

Note: For more information about using Red Hat OpenShift with IBM Cloud Pak for Applications, see Chapter 3, “IBM Cloud Paks” on page 15.

Figure 6-1 shows the different deployment setups that were used: Red Hat OpenShift V4.3, V4.4, and V4.5 using bare metal servers (a POWER9 processor-based server with logical partitions (LPARs), but in bare metal mode), A POWER8 processor-based installation with IBM Power Virtualization Center (IBM PowerVC) and Terraform automation scripts as an on-premises deployment, and an IBM Cloud installation that used IBM Power Systems Virtual Server.

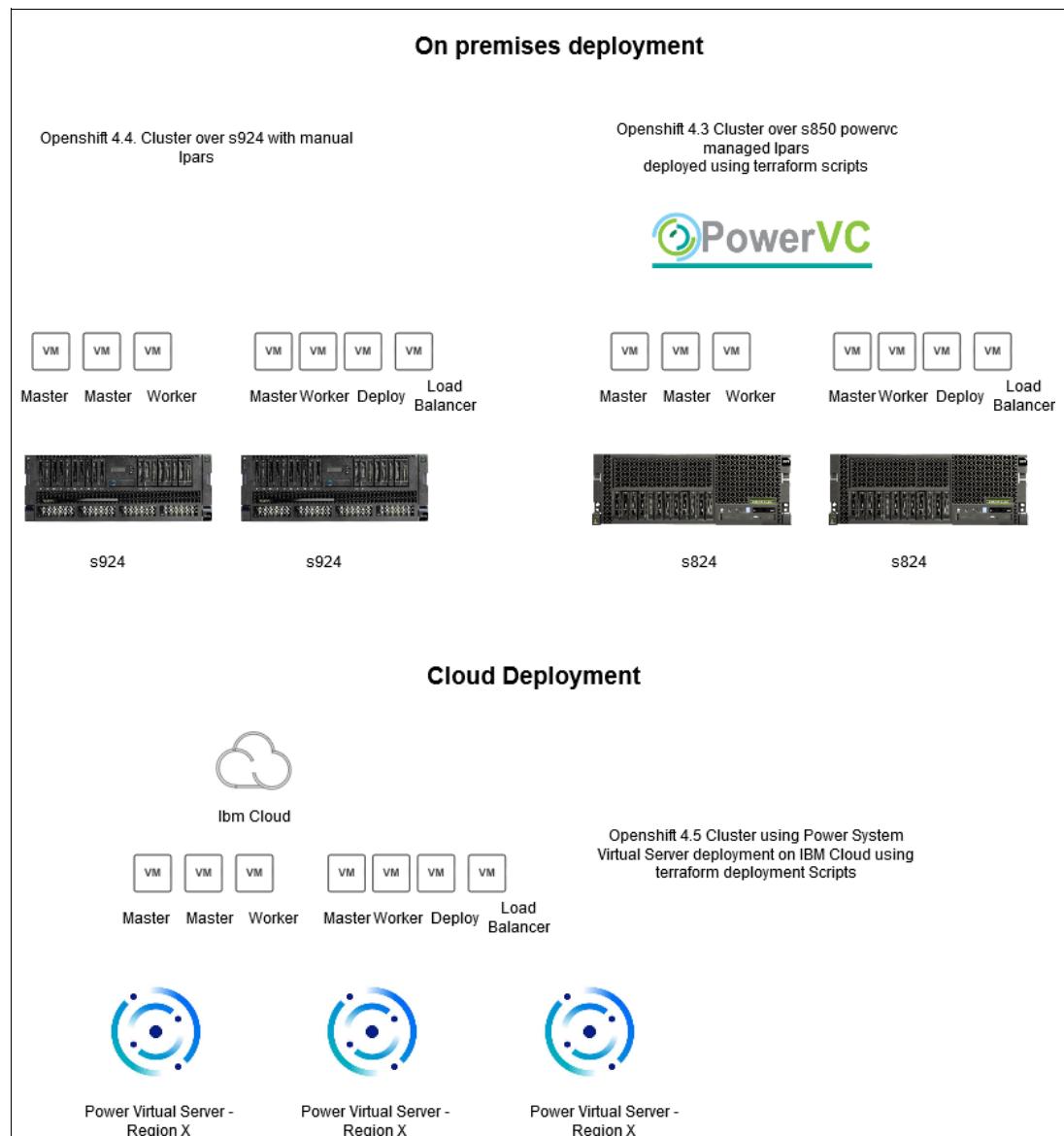


Figure 6-1 Deployment scenarios that were used in this book

6.2 Deploying Red Hat OpenShift Container Platform V4.5 in an IBM Power Systems Virtual Server on IBM Cloud

This section describes the Red Hat OpenShift Container Platform V4.5 deployment process in IBM Power Systems Virtual Server on IBM Cloud by using Red Hat Ansible and Terraform. The process uses the code that is found at the following [GitHub repository](#).

The process includes the following steps:

1. Setting up the IBM Cloud environment:
 - a. IBM Power Systems Virtual Server Service.
 - b. Private network.
 - c. Import Red Hat images.
 - d. User application programming interface (API) key.
2. Setting up the deployment host:
 - a. Install Terraform.
 - b. Install IBM Cloud Terraform Provider.
 - c. Install the IBM Power Systems Virtual Server command-line interface (CLI).
3. Deploy the Red Hat OpenShift Container Platform:
 - a. Clone the ocp4-upi-powervs Git repository.
 - b. Set up the Terraform variables.
 - c. Install the Red Hat OpenShift Container Platform.

Note: Terraform is not required to install Red Hat OpenShift Container Platform. Multiple approaches are available for deploying the infrastructure. This section demonstrates how to use an infrastructure as code with Terraform to simplify the deployment.

6.2.1 IBM Power Systems Virtual Server on IBM Cloud

IBM Power Systems Virtual Server is a Power Systems enterprise infrastructure-as-a-service (IaaS) offering on IBM Cloud. This offering provides a way for customers to purchase AIX, IBM i, or Linux on Power virtual machine (VM)-based Virtual Servers.

With IBM Power Systems Virtual Server, customers can provision the CPU, memory, storage, and networking capabilities through the IBM Cloud catalog while also self-managing from the operating system (OS) up. Customers can also use extra IBM Cloud Services to build a cost-effective, secure, and scalable hybrid multicloud architecture.

IBM Power Systems Virtual Servers on IBM Cloud are composed of the following items:

- ▶ Powered by IBM Power Systems: S-Class (scale-out) and E-Class (scale-up) systems running PowerVM.
- ▶ AIX, IBM i, and SUSE Linux Enterprise Server: Choose from a catalog of supported AIX, IBM i, and SUSE Linux Enterprise Server images, or use your own image.

IBM Power Systems Virtual Server enables provisioning of IBM Power Systems Virtual Server LPARs (VMs) in the following multi-zone regions:

- ▶ US-East (WDC04)
- ▶ US-South (DAL13 and DAL12)
- ▶ Germany-Frankfurt (FRA04 and FRA05)
- ▶ UK-London (LON06 and LON04)

- ▶ Canada-Toronto (TOR01)
- ▶ Australia-Sydney (SYD04)

The following new multi-zone regions will be available in 3Q and 4Q of 2020 and in 1Q of 2021:

- ▶ Canada-Montreal (MON01)
- ▶ Japan-Tokyo and Osaka (TOK04 and OSA02)
- ▶ Brazil-Sao Paulo (SAO01 and SAO04)
- ▶ Australia-Sydney (SYD05)

Note: For more information about IBM Power Systems Virtual Server, see [Power Systems Virtual Servers Release notes](#).

6.2.2 Getting started with deployment in IBM Power Systems Virtual Server

This section describes the minimal configuration of Red Hat OpenShift V4.5 on IBM Power Systems Virtual Server to start using the container platform. This initial sizing helps you to place a Red Hat OpenShift V4.5 instance (stand-alone) in a specific region. This option also helps you to scale to a high availability (HA) production level deployment when it is available.

To deploy a minimal configuration of Red Hat OpenShift Container Platform V4.5, you need seven IBM Power Systems Virtual Server instances:

- ▶ A Bastion (helper) system
- ▶ A temporary bootstrap machine
- ▶ Three masters or control plane machines
- ▶ Two workers or compute machines

Table 6-1 lists the minimum resources that are required for Red Hat OpenShift Container Platform V4.5 for the IBM Power Systems Virtual Server instances. eCPU refers to the number of processing units of entitled capacity.

Table 6-1 Red Hat OpenShift for IBM Power Systems Virtual Server instances roles and configurations

Instances	OS	vCPU	eCPU	Memory	Storage (GB)
Bastion (Helper)	Red Hat Enterprise Linux V8.2	1	1	16 GB	120+300 for NFS
Bootstrap	Red Hat Enterprise Linux CoreOS	1	0.5	16 GB	120
Control plane	Red Hat Enterprise Linux CoreOS	1	0.5	16 GB	120
Compute	Red Hat Enterprise Linux CoreOS	1	0.5	32 GB	120

Note: The pricing for memory is calculated based on a ratio of 64 GB per core. For example, if you use more than 16 GB for 0.25 cores, you must pay a premium high-use RAM price for the excess memory. However, as another example, if you use up to 128 GB for two cores, you do not have to pay the premium memory price.

In Red Hat OpenShift V4.x, the three master nodes are a requirement, and at least two worker nodes must be present in the cluster. Red Hat OpenShift environments with multiple workers usually require persistent storage for containers. In most cases, the developer of the application does not control to which worker in the Red Hat OpenShift cluster the application pod is dispatched. So, regardless to which worker the application can be deployed, the persistent storage that is needed by the application must be available. At the time of writing, the supported storage provider for the Red Hat OpenShift V4.5 on IBM Power Systems Virtual Server architecture (Figure 6-2) is an NFS server.

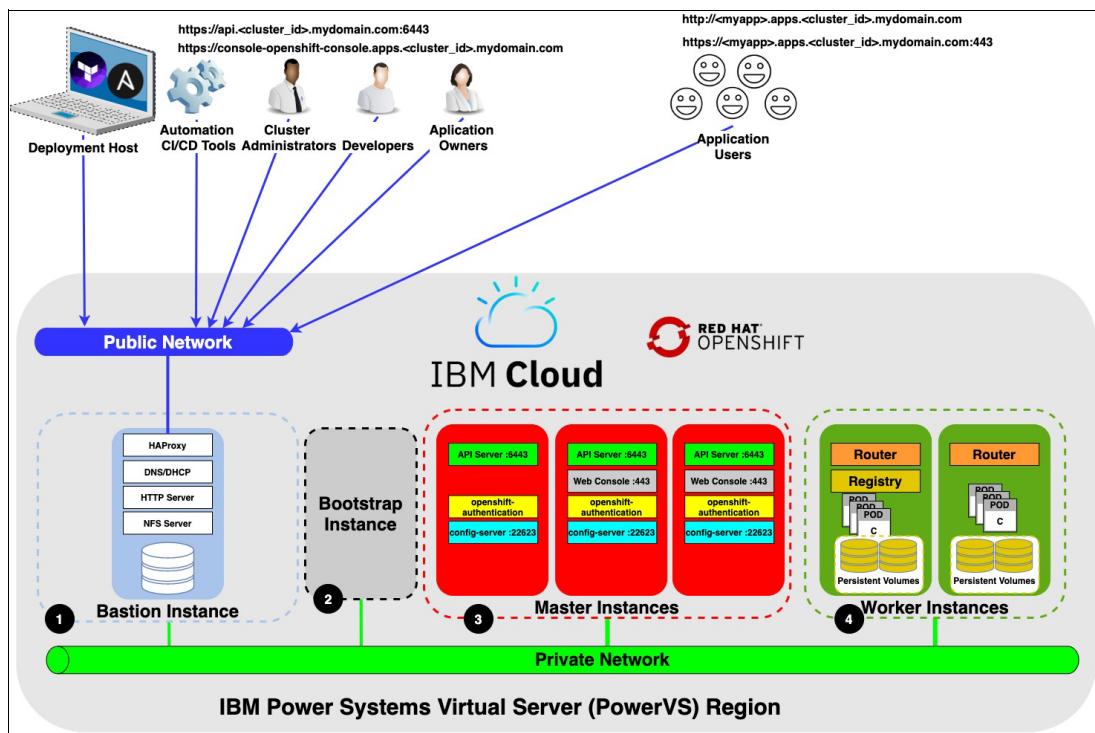


Figure 6-2 Red Hat OpenShift Container Platform V4.5 in IBM Power Systems Virtual Server

In Figure 6-3, the solid boxes represent the minimal IBM Power Systems Virtual Server instances that are required to run Red Hat OpenShift Container Platform V4.5 with NFS storage.

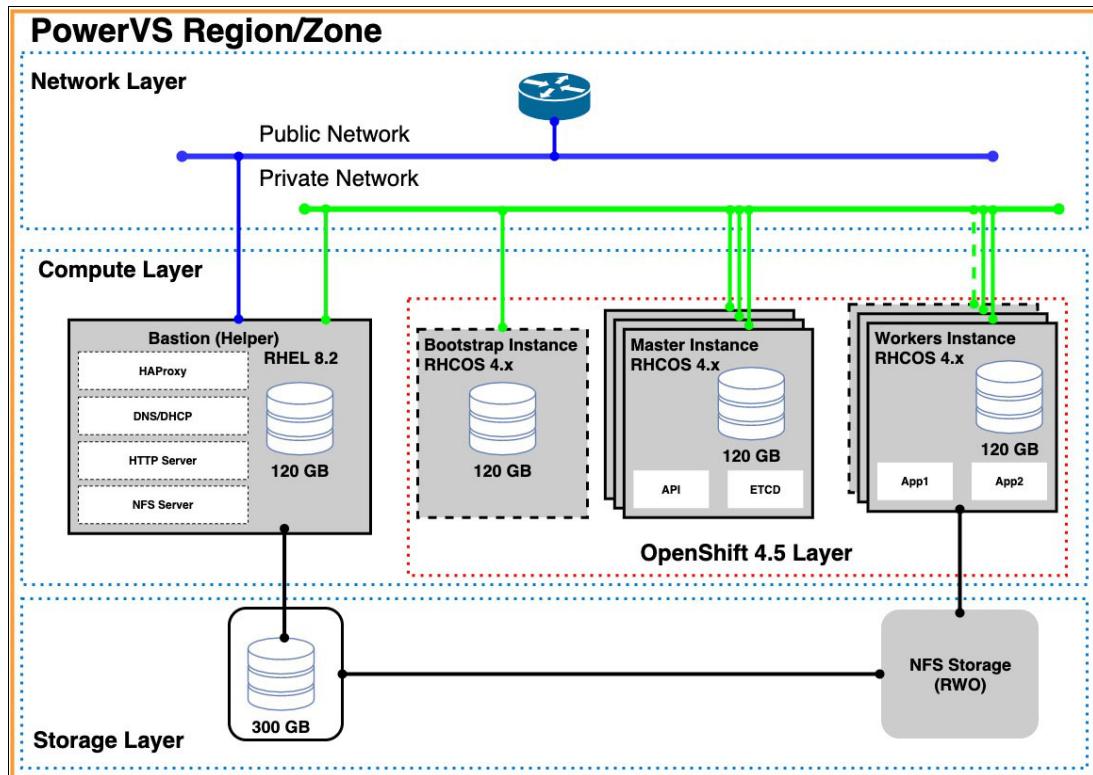


Figure 6-3 Typical Red Hat OpenShift V4.5 deployment in IBM Power Systems Virtual Server

As you can see in Figure 6-3, the Bastion (Helper) instance plays a key role because it is the interface between the external world and Red Hat OpenShift Container Platform, and it acts as the NFS storage provider. The following services are automatically configured by Red Hat Ansible on the Bastion instance during the deployment process:

- ▶ DHCP server
- ▶ DNS server
- ▶ HTTP file server to host ignition configurations
- ▶ HAproxy to load-balance traffic to Red Hat OpenShift controllers and the ingress router
- ▶ Squid proxy for Red Hat OpenShift instances to access the internet
- ▶ NFS server for persistent storage to containers

6.2.3 Prerequisites

This section shows the prerequisites to use IBM Power Systems Virtual Server.

- ▶ IBM Cloud account
You need a paid IBM Cloud account to use IBM Power Systems Virtual Server.
- ▶ Red Hat subscription
You need a valid Red Hat subscription, and the subscription ID is required during the setup process.
- ▶ Red Hat OpenShift Pull Secret
You must have this item when provisioning the Red Hat OpenShift Container Platform.

6.2.4 Setting up the IBM Cloud environment

This section describes how to set up the IBM Cloud environment.

IBM Power Systems Virtual Server service

You need an IBM Power Systems Virtual Server service on the region where you plan to deploy the Red Hat OpenShift Container Platform.

Before you create a virtual server, you must understand the difference between a IBM Power Systems Virtual Server *service* and a IBM Power Systems Virtual Server *instance*. Think of the IBM Power Systems Virtual Server service as a container for all IBM Power Systems Virtual Server instances at a specific geographic region. The IBM Power Systems Virtual Server service is available in the resource list in the IBM Power Systems Virtual Server user interface. The service can contain multiple IBM Power Systems Virtual Server instances.

If the IBM Power Systems Virtual Server Service does not exist, complete the following steps to create it:

1. Log in to the [IBM Cloud Dashboard](#) and search for Power in the Cloud Catalog, as shown in Figure 6-4.

The screenshot shows the IBM Cloud Catalog interface. On the left, there's a sidebar with categories like Catalog, Featured Services, Software, Consulting, and a detailed Category section with various sub-options. The main area has a search bar at the top with the word 'Power'. Below the search bar, the text 'Search results for "Power" 86 results' is displayed. There are six service cards listed:

- Power Systems Virtual Server**: IBM • Services • Compute. Provision flexible, secure, and scalable compute capacity for Power Systems enterprise workloads. IAM-enabled. Free.
- Powerlytics Behavior/Propensity Model API**: Third party • Services • Analytics. Improve customer behavior/propensity models. Free.
- Powerlytics Consumer Income API**: Third party • Services • Developer Tools. The income profile of consumers at the ZIP+4 level. Free.
- Powerlytics Investable Assets & Wealth API**: Third party • Services • Developer Tools. Investable assets & wealth income at Zip+4 level. Free.
- Email Delivery, powered by Sendgrid**: Third party • Services • Developer Tools. Integrate and Deliver via SMTP or API in 5 Minutes. Less Our SMTP relay setup and flexible Web and SMTP APIs provide a customizable integration... Free.
- Powering Content and data for your application.**: Third party • Services • Storage. Whether you are building a line of business app, content management software or need to display... Free.

Figure 6-4 IBM Cloud Catalog

2. Select **Power Systems Virtual Server** and select a region, as shown in Figure 6-5.

The screenshot shows the 'Power Systems Virtual Server' service creation page. At the top, it says 'Catalog / Services / Power Systems Virtual Server'. Below that, it shows the author as 'IBM' and the date last updated as '08/31/2020'. There are two tabs: 'Create' (which is active) and 'About'. A section titled 'Select a region' contains a dropdown menu with 'Sydney 04' selected.

Figure 6-5 Creating a IBM Power Systems Virtual Server service: Region selection

- Specify the **Service name**, apply any promo code, and create the IBM Power Systems Virtual Server service, as shown in Figure 6-6.

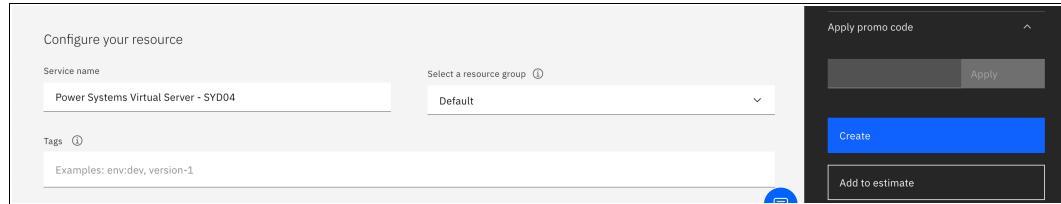


Figure 6-6 Creating a IBM Power Systems Virtual Server service

Private network

A private network is required for the Red Hat OpenShift Container Platform.

If you do not have a private network and you want to create a network for Red Hat OpenShift, complete the following steps:

- Select your service instance, and create a private subnet by selecting **Subnets** and providing the required inputs, as shown in Figure 6-7.

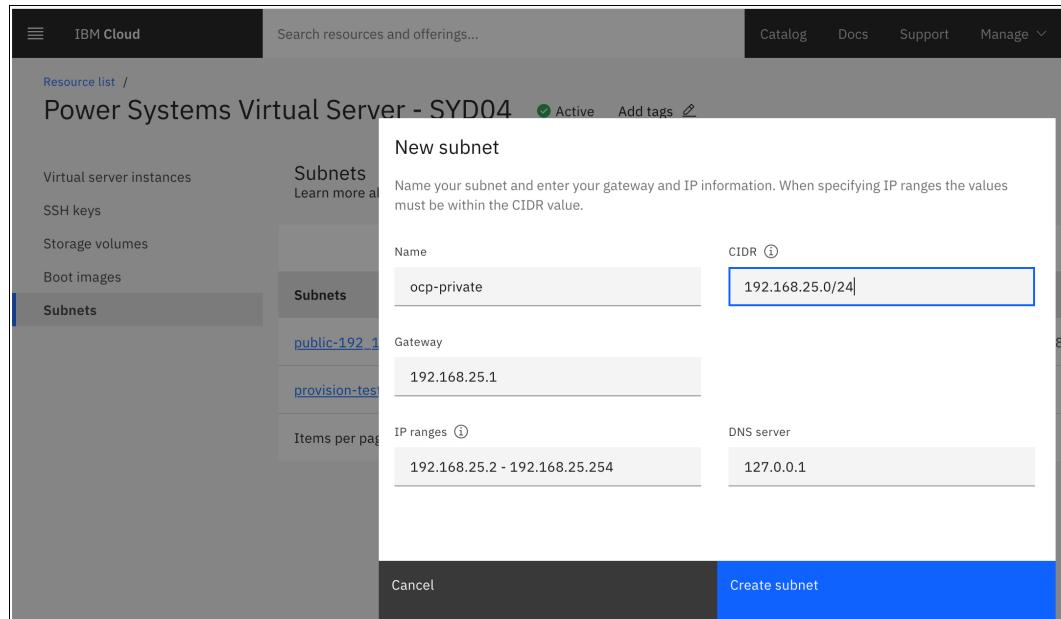


Figure 6-7 Creating a subnet in the IBM Power Systems Virtual Server

- Open a new Service Request to enable IP communication between IBM Power Systems Virtual Server instances in the private network, as shown in Figure 6-8 on page 81. Click **Support** on the top bar and scroll down to Contact Support. Select **Create a case**, and then select the IBM Power Systems Virtual Server and complete the details by using the template that is shown in Example 6-1.

Example 6-1 Support case template for a new subnet

[Subject:] Enable communication between IBM Power Systems Virtual Server instances on private network
[Body:]

Enable IP communication between IBM Power Systems Virtual Server instances for the following private network:

Name: <your-subnet-name-from-above>
Type: Private
CIDR: <your ip subnet-from-above>
VLAN ID: <your-vlan-id> (listed in your subnet details post-creation)
Location: <your-location> (listed in your subnet details post-creation)
Service Instance: <your-service-name>

The screenshot shows a support case interface on the IBM Cloud platform. At the top, there's a navigation bar with 'IBM Cloud', a search bar, and links for 'Catalog' and 'Docs'. The main area has a title 'Subject' with a character limit of '65/160' and a text input field containing 'Enable communication between PowerVS instances on private network'. Below this is a 'Description' section with a character limit of '244/4000' and a text input field containing a detailed request. A blue rectangular box highlights the request text, which includes a list of network parameters and a service instance reference.

Subject 65/160

Enable communication between PowerVS instances on private network

Description 244/4000

Please enable IP communication between PowerVS instances for the following private network:
Name: ocp-private
Type: Private
CIDR: 192.168.25.0/24
VLAN ID: 215
Location: Sydney04
Service Instance: Power Systems|Virtual Server - SYD04

Figure 6-8 Support case example for opening a private network

Importing Red Hat images

A Red Hat Enterprise Linux image is used for the Bastion (Helper) instance, and Red Hat Enterprise Linux CoreOS is used for the Red Hat OpenShift Container Platform instances.

The images must be compatible with IBM PowerVC and use the Open Virtual Application (.OVA) file format. The OS disk must be a minimum of 120 GB.

If you have qcow2 images, you can use this [script](#) to convert it to a compatible OVA image.

Note: This book does not cover image creation. For more information, see [Working with images](#).

Uploading OVA images to IBM Cloud Object Storage

This section provides directions to upload OVA images into IBM Cloud Object Storage:

- ▶ Create the IBM Cloud Object Storage service and bucket.
For more information about creating the IBM Cloud Object Storage service and the required storage bucket to upload the OVA images, see [Getting started with IBM Cloud Object Storage](#).
- ▶ Create the secret and the access keys by using Hash-based Message Authentication Code (HMAC).

For more information about creating the keys that are required for importing the images into your IBM Power Systems Virtual Server service instance, see [Using HMCA credentials](#).

- ▶ Upload the OVA image to the IBM Cloud Object Storage bucket.

For more information about uploading the OVA image to the respective bucket, see [Upload data](#).

Alternatively, you can use the Python script at [GitHub](#).

Importing the images into IBM Power Systems Virtual Server

This section describes how to import the images into IBM Power Systems Virtual Server:

Select your service instance, click **View full details**, and then select **Boot images**. Click **Importing image** and enter the details like **Catalog image name**, **Storage type**, and **Cloud storage details**, as shown in Figure 6-9.

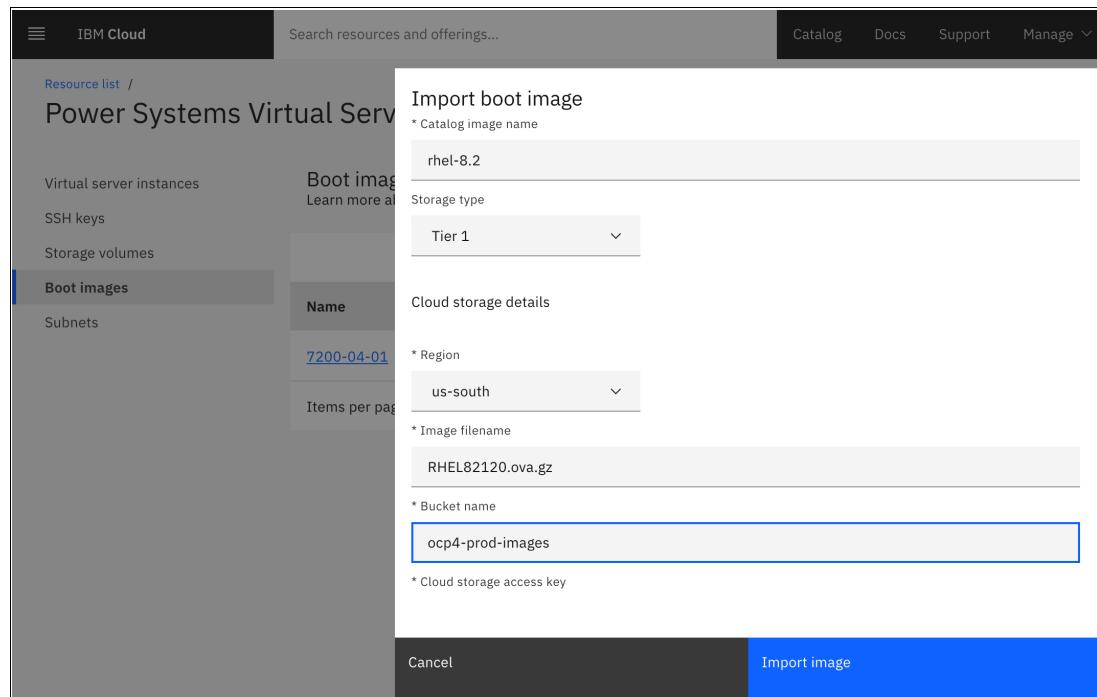


Figure 6-9 Importing the Red Hat Enterprise Linux V8.2 image

Figure 6-10 on page 83 shows how to import the boot image for Red Hat Enterprise Linux CoreOS 4.5.4.

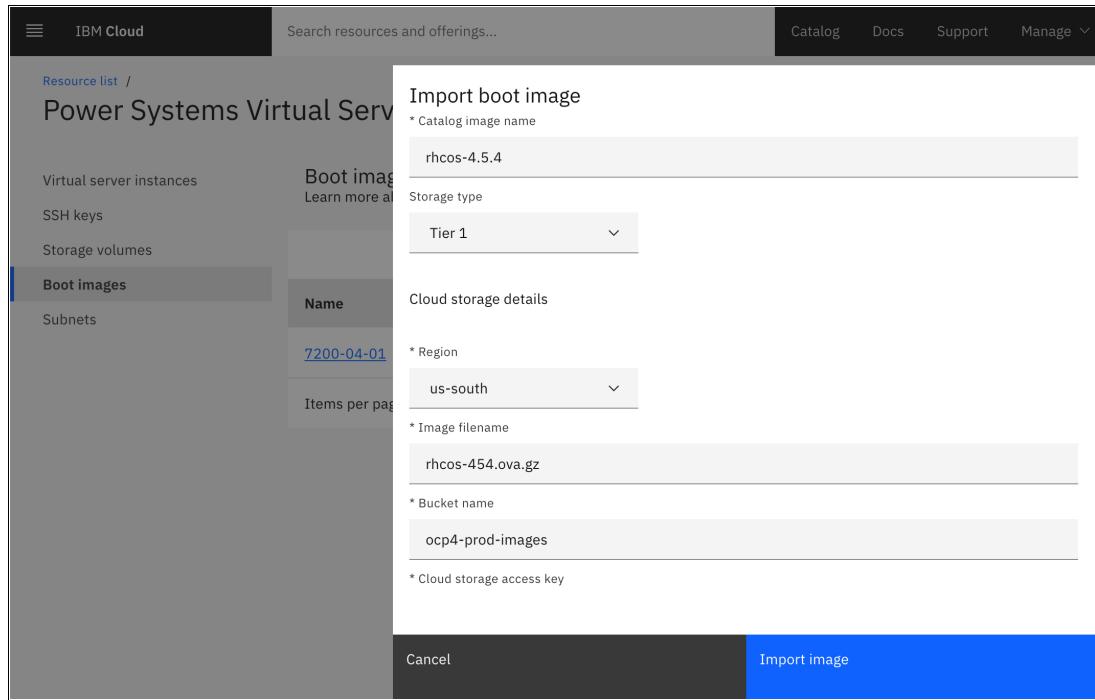


Figure 6-10 Importing the Red Hat Enterprise Linux CoreOS V4.5.4 image

User API key

The Terraform plug-in for IBM Cloud is using an API key for authentication. The key is also needed for the CLI. To generate and export a new key, you must connect to the IBM Cloud console, select **Manage** → **Access (IAM)**, and select **API Keys** from the left pane, as shown in Figure 6-11.

Status	Name	Description	Date Created
	bsavu_api_key	Bogdan's API Key	2020-06-24 19:56 GMT

Figure 6-11 User API key

6.2.5 Setting up the deployment host

The Terraform automation must run from a system with internet access, such as a Notebook or a VM with public internet connectivity. The automaton code at this [GitHub repository](#) was tested on the following OSs:

- ▶ Linux (x86_64)
- ▶ Mac OSX (Darwin)
- ▶ Windows 10

Note: For this book, the deployment host is a Red Hat 8 (x86_64) VM.

Installing Terraform 0.12.29

Important: At the time of writing, the supported Terraform versions are greater than or equal to Version 0.12.2 and less than Version 0.13.

Example 6-2 shows the installation commands for Terraform 0.12.29. For more information about installing Terraform, see [Install Terraform](#).

Example 6-2 Installing Terraform

```
wget https://releases.hashicorp.com/terraform/0.12.29/terraform_0.12.29_linux_amd64.zip
...
Output truncated
...
100%[=====] 27.11M 11.1MBps in 2.4s
2020-09-15 12:55:27 (11.1 MBps) - 'terraform_0.12.29_linux_amd64.zip' saved [28424050/28424050]

unzip terraform_0.12.29_linux_amd64.zip
Archive: terraform_0.12.29_linux_amd64.zip
  inflating: terraform

sudo mv terraform /usr/local/bin/

terraform -version
Terraform v0.12.29
```

Installing IBM Cloud Terraform Provider V1.9.0

Example 6-3 shows the installation commands for IBM Cloud Terraform Provider V1.9.0. For more information about installing the provider plug-in, see [Install the IBM Cloud Provider plug-in](#).

Example 6-3 Installing IBM Cloud Terraform Provider

```
wget
https://github.com/IBM-Cloud/terraform-provider-ibm/releases/download/v1.9.0/linux_amd64.zip
...
Output truncated
...
100%[=====] 14.31M 3.33 MBps in 6.3s
2020-09-15 12:59:33 (2.26 MBps) - 'linux_amd64.zip' saved [15006584/15006584]

mkdir $HOME/.terraform.d/plugins
mv linux_amd64.zip $HOME/.terraform.d/plugins/
```

```

cd $HOME/.terraform.d/plugins

unzip linux_amd64.zip
Archive: linux_amd64.zip
  inflating: terraform-provider-ibm_v1.9.0.pem
  inflating: terraform-provider-ibm_v1.9.0.sig
  inflating: terraform-provider-ibm_v1.9.0

./terraform-provider-ibm_v1.9.0
2020/09/15 13:03:11 IBM Cloud Provider version 1.9.0
This binary is a plug-in. These are not meant to be run directly.
Execute the program that consumes these plug-ins, which will
load any plug-ins automatically

```

IBM Power Systems Virtual Server CLI

Example 6-4 shows the installation commands for the IBM Power Systems Virtual Server CLI.
For more information about installing the IBM Power Systems Virtual Server CLI,
see [IBM Power Systems Virtual Server CLI Reference](#).

Example 6-4 Installing IBM Power Systems Virtual Server CLI

```

sudo curl -sL https://ibm.biz/itd-installer | bash
[main] ---[ IBM Cloud Developer Tools for Linux/MacOS - Installer, v1.2.3 ]---
which: no ibmcloud in (/home/bogdan/.local/bin:/home/bogdan/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin)
[install] Starting Update...
which: no ibmcloud in (/home/bogdan/.local/bin:/home/bogdan/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin)
[install] Note: You might be prompted for your 'sudo' password during installation.
[install_deps_with_yum] Checking for and updating 'yum' support on Linux
[install_deps_with_yum] Installing/updating external dependency: curl
[install_deps_with_yum] Installing/updating external dependency: git
which: no git in (/home/bogdan/.local/bin:/home/bogdan/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin)
Last metadata expiration check: 0:10:42 ago on Tue 15 Sep 2020 12:54:06 PM EEST.
Dependencies resolved.
=====
Package           Architecture      Version       Repository Size
=====
Installing:
git              x86_64          2.27.0-1.el8   Stream-AppStream 164 k
Installing dependencies:
git-core          x86_64          2.27.0-1.el8   Stream-AppStream 5.7 M
git-core-doc      noarch          2.27.0-1.el8   Stream-AppStream 2.5 M
perl-Error        noarch          1:0.17025-2.el8 AppStream 46 k
perl-Git          noarch          2.27.0-1.el8   Stream-AppStream 77 k
perl-TermReadKey x86_64          2.37-7.el8    AppStream 40 k
Transaction Summary
=====
Install 6 Packages
...
Output truncated
...
Plug-in 'container-service 1.0.157' was successfully installed into /home/bogdan/.bluemix/plugins/container-service. Use 'ibmcloud plugin show container-service' to show its details.
[install_plugins] Running 'ibmcloud plugin list'...
Listing installed plug-ins...

Plug-in Name      Version  Status
container-service/kubernetes-service 1.0.157
cloud-functions/wsk/functions/fn     1.0.46
cloud-object-storage                 1.2.0
container-registry                  0.1.494

[install_plugins] Finished installing/updating plug-ins
[env_setup] WARN: Restart your shell to enable 'ic' alias for ibmcloud!
[install] Installation finished.
[main] ---[ Total time: 666 seconds ]---

ibmcloud plugin install power-iaas
Looking up 'power-iaas' from repository 'IBM Cloud'...
Plug-in 'power-iaas/pi 0.2.9' found in repository 'IBM Cloud'
Attempting to download the binary file...
16.43 MiB / 16.43 MiB [=====] 100.00% 0s

```

```

17223680 bytes downloaded
Installing binary...
OK
Plug-in 'power-iaas 0.2.9' was successfully installed into /home/bogdan/.bluemix/plugins/power-iaas. Use 'ibmcloud plugin show power-iaas' to show its details.

ibmcloud plugin update
Checking upgrades for all installed plug-ins from repository 'IBM Cloud'...
No updates are available.

ibmcloud plugin list
Listing installed plug-ins...



| Plug-in Name                         | Version | Status |
|--------------------------------------|---------|--------|
| cloud-functions/wsk/functions/fn     | 1.0.46  |        |
| cloud-object-storage                 | 1.2.0   |        |
| container-registry                   | 0.1.494 |        |
| container-service/kubernetes-service | 1.0.157 |        |
| power-iaas                           | 0.2.9   |        |


```

6.2.6 Deploying the Red Hat OpenShift Container Platform

This section describes how to deploy the Red Hat OpenShift Container Platform.

Cloning the ocp4-upi-powervs Git repository

Example 6-5 shows how to clone the ocp4-upi-powervs Git repository.

Example 6-5 Cloning the ocp4-upi-powervs Git repository

```

git clone https://github.com/ocp-power-automation/ocp4-upi-powervs.git -b release-4.5
ocp4-upi-powervs-4.5
Cloning into 'ocp4-upi-powervs-4.5'...
remote: Enumerating objects: 74, done.
remote: Counting objects: 100% (74/74), done.
remote: Compressing objects: 100% (64/64), done.
remote: Total 495 (delta 27), reused 42 (delta 10), pack-reused 421
Receiving objects: 100% (495/495), 1.76 MiB | 1.82 MiBps, done.
Resolving deltas: 100% (255/255), done.
cd ocp4-upi-powervs-4.5

```

Setting up Terraform variables

This section shows how to set up the Terraform variables:

- ▶ Copy the Secure Shell (SSH) key pair under the data directory. The keys are used later to access the Bastion and cluster instances.
- ▶ Create a file (pull-secret.txt) under the data directory that you use to store the Red Hat OpenShift pull secret.
- ▶ Update the var.tfvars file based on your environment, as shown in Example 6-6. For descriptions of the variables, see the following [GitHub repository](#).

Example 6-6 The var.tfvars file

```

### IBM Cloud details

ibmcloud_api_key      = "xyzaaaaaaaaabcdeaaaaaa"
ibmcloud_region        = "syd"
ibmcloud_zone          = "syd04"
service_instance_id    = "abc123xyzaaaa"

### OpenShift Cluster Details

```

```

### This is default minimalistic config. For PowerVS processors are equal to entitled physical
count

bastion          = {memory      = "16",   processors = "1"}
bootstrap        = {memory      = "16",   processors = "0.5",  "count"   = 1}
master          = {memory      = "16",   processors = "0.5",  "count"   = 3}
worker          = {memory      = "32",   processors = "0.5",  "count"   = 2}

rhel_image_name    = "rhel-8.2"
rhcosh_image_name = "rhcosh-4.5.4"
processor_type     = "shared"
system_type        = "s922"
network_name       = "ocp-private"

rhel_username      = "root"
public_key_file    = "data/id_rsa.pub"
private_key_file   = "data/id_rsa"
rhel_subscription_username = "<username>"
rhel_subscription_password = "<password>"
rhel_smt           = 4

### OpenShift Installation Details

openshift_install_tarball  =
"https://mirror.openshift.com/pub/openshift-v4/ppc64le/clients/ocp/4.5.4/openshift-install-linux
.tar.gz"
openshift_client_tarball   =
"https://mirror.openshift.com/pub/openshift-v4/ppc64le/clients/ocp/4.5.4/openshift-client-linux.
tar.gz"
pull_secret_file         = "data/pull-secret.txt"

cluster_domain          = "ibm.com"
cluster_id_prefix        = "bs-ocp"
cluster_id               = ""

```

Installing the Red Hat OpenShift Container Platform

Apply the Terraform configuration as shown in Example 6-7 and wait about 60 minutes for the deployment to complete. The deployment is fully automated. The automated process uses Terraform for infrastructure creation and Red Hat Ansible for the Red Hat OpenShift deployment.

Example 6-7 Installing Red Hat OpenShift Container Platform

```

terraform init
Initializing modules...
- install in modules/5_install
- nodes in modules/4_nodes
- prepare in modules/1_prepare

Initializing the backend...

Initializing provider plug-ins...
- Checking for available provider plug-ins...

```

```
- Downloading plug-in for provider "null" (hashicorp/null) 2.1.2...
- Downloading plug-in for provider "random" (hashicorp/random) 2.3.0...
- Downloading plug-in for provider "ignition" (terraform-providers/ignition) 1.2.1...
```

```
Terraform has been successfully initialized!
```

```
...
```

```
Output truncated
```

```
...
```

```
terraform apply -var-file var.tfvars -parallelism=3
```

```
module.nodes.data.ignition_file.b_hostname: Refreshing state...
module.nodes.data.ignition_file.m_hostname[0]: Refreshing state...
module.nodes.data.ignition_file.m_hostname[2]: Refreshing state...
module.nodes.data.ignition_file.m_hostname[1]: Refreshing state...
module.nodes.data.ignition_file.w_hostname[0]: Refreshing state...
module.nodes.data.ignition_file.w_hostname[1]: Refreshing state...
module.nodes.data.ibm_pi_network.network: Refreshing state...
module.install.data.ibm_pi_network.network: Refreshing state...
module.prepare.data.ibm_pi_network.network: Refreshing state...
module.prepare.data.ibm_pi_image.bastion: Refreshing state...
module.nodes.data.ibm_pi_image.rhcos: Refreshing state...
```

```
An execution plan has been generated and is shown below.
```

```
Resource actions are indicated with the following symbols:
```

```
+ create
```

```
<= read (data resources)
```

```
Terraform will perform the following actions:
```

```
...
```

```
Output truncated
```

```
...
```

```
Do you want to perform these actions?
```

```
Terraform will perform the actions described above.
```

```
Only 'yes' will be accepted to approve.
```

```
Enter a value: yes
```

```
...
```

```
Output truncated
```

```
...
```

```
Apply complete! Resources: 17 added, 0 changed, 0 destroyed.
```

```
Outputs:
```

```
bastion_private_ip = 192.168.25.156
bastion_public_ip = 130.198.3.133
bastion_ssh_command = ssh -i data/id_rsa root@130.198.3.133
bootstrap_ip = 192.168.25.68
cluster_authentication_details = Cluster authentication details are available in 130.198.3.133
under ~/openstack-upi/auth
cluster_id = bs-ocp-e0ee
etc_hosts_entries =
130.198.3.133 api.bs-ocp-e0ee.ibm.com console-openshift-console.apps.bs-ocp-e0ee.ibm.com
integrated-oauth-server-openshift-authentication.apps.bs-ocp-e0ee.ibm.com
oauth-openshift.apps.bs-ocp-e0ee.ibm.com
```

```

prometheus-k8s Openshift Monitoring.apps.bs-ocp-e0ee.ibm.com
grafana-openshift-monitoring.apps.bs-ocp-e0ee.ibm.com example.apps.bs-ocp-e0ee.ibm.com

install_status = COMPLETED
master_ips = [
    "192.168.25.189",
    "192.168.25.231",
    "192.168.25.91",
]
oc_server_url = https://api.bs-ocp-e0ee.ibm.com:6443
storageclass_name = nfs-storage-provisioner
web_console_url = https://console-openshift-console.apps.bs-ocp-e0ee.ibm.com
worker_ips = [
    "192.168.25.10",
    "192.168.25.160",
]

```

Important: In our example, we used parallelism to restrict parallel instance creation requests through the IBM Power Systems Virtual Server client because of a known issue where the `apply` command fails for random parallel instance create requests. If you still get the error while creating the instance, delete the failed instance from the IBM Power Systems Virtual Server console and run the `apply` command again.

Note: If there are any errors, run the `apply` command again. For more information about potential issues and workarounds, see this [Git repository](#).

Postinstallation configuration

This section provides postinstallation configuration checks:

1. Delete the bootstrap instance.

After the deployment completes successfully, you can delete the bootstrap instance. This step is optional but recommended to release the resources that are used:

- a. Change the count value to 0 in the bootstrap map variable and rerun the `apply` command. For example:

```
bootstrap = {memory = "16", processors = "0.5", "count" = 0}
```

- b. Run the `terraform apply -var-file var.tfvars` command, as shown in Example 6-8.

Example 6-8 Deleting the bootstrap instance

```

terraform apply -var-file var.tfvars
module.nodes.data.ignition_file.m_hostname[2]: Refreshing state...
module.nodes.data.ignition_file.m_hostname[1]: Refreshing state...
module.nodes.data.ignition_file.w_hostname[1]: Refreshing state...
module.nodes.data.ignition_file.b_hostname: Refreshing state...
module.nodes.data.ignition_file.w_hostname[0]: Refreshing state...
module.nodes.data.ignition_file.m_hostname[0]: Refreshing state...
random_id.label[0]: Refreshing state... [id=404]
module.prepare.data.ibm_pi_network.network: Refreshing state...
module.install.data.ibm_pi_network.network: Refreshing state...
module.nodes.data.ibm_pi_image.rhcos: Refreshing state...
module.prepare.ibm_pi_key.key: Refreshing state...
[id=47d2bdf9-04dc-4d39-a3db-84ec96038d0e] /bs-ocp-e0ee-keypair]
module.prepare.data.ibm_pi_image.bastion: Refreshing state...

```

```
module.prepare.ibm_pi_volume.volume[0]: Refreshing state...
[id=47d2bdf9-04dc-4d39-a3db-84ec96038d0e/9b91a7e9-a7c8-4f57-b557-180c62273617]
module.nodes.data.ibm_pi_network.network: Refreshing state...
...
Output truncated
...
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
...
Output truncated
...
Apply complete! Resources: 0 added, 0 changed, 1 destroyed.
```

2. Update the /etc/hosts file.

Add the etc_host_entries value to your /etc/hosts file, as shown in Example 6-9. This value is printed at the end of a successful installation. Alternatively, you can retrieve it anytime by running the Terraform output from the installation directory.

Example 6-9 Updating /etc/hosts

```
sudo vi /etc/hosts
...
Output truncated
...
130.198.3.133 api.bs-ocp-e0ee.ibm.com console-openshift-console.apps.bs-ocp-e0ee.ibm.com
integrated-oauth-server-openshift-authentication.apps.bs-ocp-e0ee.ibm.com
oauth-openshift.apps.bs-ocp-e0ee.ibm.com
prometheus-k8s-openshift-monitoring.apps.bs-ocp-e0ee.ibm.com
grafana-openshift-monitoring.apps.bs-ocp-e0ee.ibm.com example.apps.bs-ocp-e0ee.ibm.com
```

3. Access the Red Hat OpenShift Container Platform.

The login credentials are stored on the Bastion instance, and the location is printed at the end of a successful installation. Alternatively, you can retrieve the location anytime by running the Terraform output from the installation directory, as shown in Example 6-10.

Example 6-10 Getting the login credentials

```
terraform output
...
Output truncated
...
bastion_public_ip = 130.198.3.133
bastion_ssh_command = ssh -i data/id_rsa root@130.198.3.133
bootstrap_ip =
cluster_authentication_details = Cluster authentication details are available in 130.198.3.133
under ~/openstack-upi/auth
...
Output truncated
...
```

There are two files under `~/openstack-upi/auth`:

- `kubeconfig`: This file can be used for CLI access.
- `kubeadm-password`: This file contains the password for the `kubeadm` user, which can be used for CLI and GUI access (Figure 6-12).

The screenshot shows the Red Hat OpenShift Container Platform console interface. The top navigation bar includes the Red Hat logo, 'OpenShift Container Platform', and a dropdown for 'kube:admin'. The left sidebar has a 'Administrator' section with links for Home, Operators, Workloads, Networking, Storage, Builds, Monitoring, Compute, User Management, and Administration. The main content area is titled 'Overview' under the 'Cluster' tab. It displays cluster details like Cluster API Address (https://api.bs-ocp-e0e.ibm.com:6443), Cluster ID (f6009fd8-b920-418d-b5fc-13f6f1365f4d), and OpenShift Cluster Manager. It also shows the status of the Cluster (green checkmark) and Control Plane (green checkmark), and Operators (green checkmark). A note indicates a cluster version update is available. The right side features an 'Activity' section with a list of recent events, such as 'Created container...', 'Started container...', and 'Container image "..."'.

Figure 6-12 Red Hat OpenShift Container Platform Console

Uninstalling Red Hat OpenShift Container Platform

If the Red Hat OpenShift cluster is no longer needed, you can run the following command from the deployment host to make sure that all resources are properly cleaned up:

```
terraform destroy -var-file var.tfvars -parallelism=3
```

6.3 Bare metal deployment by using Red Hat Enterprise Linux CoreOS LPARs on a Virtual I/O Server

Section 6.2, “Deploying Red Hat OpenShift Container Platform V4.5 in an IBM Power Systems Virtual Server on IBM Cloud” on page 75 shows how we deployed a cloud environment automatically with Red Hat Ansible and Terraform off-premises. Now, we show you how to create an on-premises installation by using POWER9 processor-based servers.

This case uses a mixed environment with two IBM Power System S924 servers running Red Hat Enterprise Linux CoreOS LPARs, Red Hat Enterprise Linux v8 LPARs for deployment and HAProxy servers, NFS storage running AIX over an IBM Power System E870 with PowerHA SystemMirror, and an IBM Spectrum Scale cluster running on an x86 RHEV powered cluster, as shown in Figure 6-13.

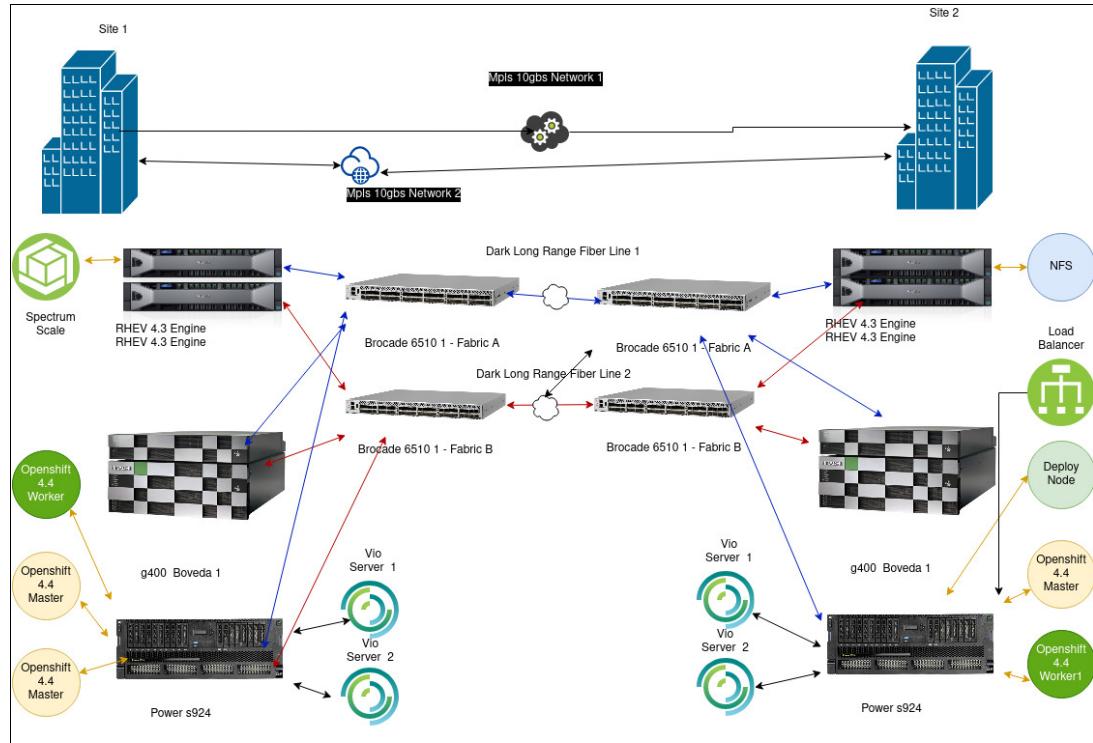


Figure 6-13 Uruguay team test environment hardware and Red Hat OpenShift cluster diagram

Figure 6-14 shows a high-level view of the Red Hat OpenShift Container Platform components for the various IBM Power Systems hardware platforms.

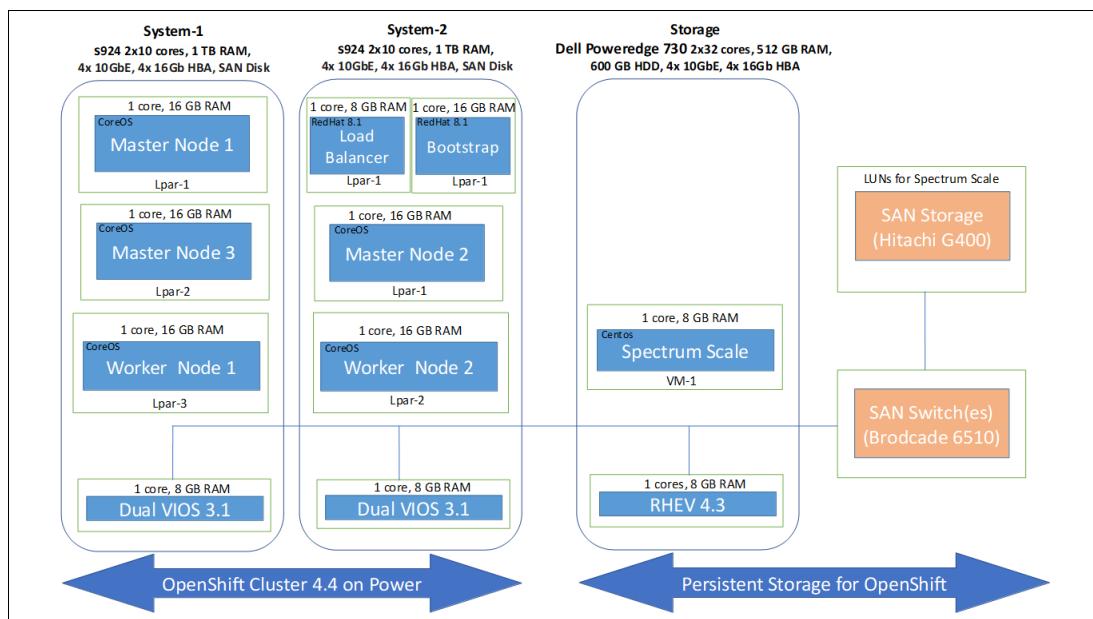


Figure 6-14 Lab setup for the Uruguay team for IBM Power Systems

6.4 Installing Red Hat OpenShift V4.3 and V4.4: Tips and tricks

This section is an update to Chapter 3, “Reference installation guide for Red Hat OpenShift V4.3 on Power Systems servers” of *Red Hat OpenShift V4.3 on IBM Power Systems Reference Guide*, REDP-5599, with some lessons learned and tips and tricks.

6.4.1 Preparing your environment

It does not matter what your base hardware is, but you need at least a minimal number of LPARs or physical machines on which to install Red Hat OpenShift V4.3, V4.4, or V4.5 on IBM Power Systems servers.

The minimum resource requirements to start a Red Hat OpenShift V4.3 cluster on Power Systems servers are described in Table 6-2.

Table 6-2 Minimum resource requirements

Machine	Amount	Operating system	vCPU	Virtual RAM	Storage
Bootstrap	1	Red Hat Enterprise Linux CoreOS	4	16 GB	120 GB
Control plane and master	3	Red Hat Enterprise Linux CoreOS	2	16 GB	120 GB
Compute and worker nodes	2	Red Hat Enterprise Linux CoreOS	2	16 GB	120 GB

Note: Even with these minimum requirements, you need other machines or appliances to get your environment working, such as a load balancer and a storage provider.

For this example, we prepare eight LPARs:

- ▶ One deployment node that works as a TFTP, HTTP, DHCP, and DNS server running Red Hat Enterprise Linux V8.1.
- ▶ One load balancer running Red Hat Enterprise Linux V8.1.
- ▶ Six running Red Hat Enterprise Linux CoreOS instances, as described in Table 6-2.

Preflight checklist

Check your environment by using the following list:

- ▶ After defining all your LPARs, write down the MAC address of each LPAR Ethernet adapter (even if it is a physical or Virtual I/O Server (VIOS) virtualized environment, you need it later).
- ▶ Check your DNS configuration. You must be able to resolve quay.io and Red Hat domains. If you are behind a proxy and using an internal DNS, you must enable DNS forwarding.
- ▶ If you are using dnsmasq or a local DNS service, set up your network on nmtui to use 127.0.0.1 as the first DNS. If you manually add this address to /etc/resolv.conf, a restart erases it.
- ▶ Enable the necessary ports in the firewall, and configure the SELinux permits.

- ▶ Go to [Red Hat Login](#), log in to your account, and in the page <https://cloud.redhat.com/openshift/install/power/user-provisioned>, download the installation files for the version that you want to install.
- ▶ Verify with your networking team that your network permits access.

Hands-on practice

Install Red Hat Enterprise Linux V7 or V8 on your deployment LPAR, register it, and set up a proxy if necessary to download the files. After Linux is installed, migrate it to the latest version (for security reasons).

For installation purposes, you must set up several services:

- ▶ The HTTPd server is needed to download several files to the nodes during installation.
- ▶ The dhcpcd server is needed for the start process to set up IP addresses for LPARs.
- ▶ The tftpd server is needed for the start process to read grub and kernel files.
- ▶ The DNS server is needed for Red Hat OpenShift to work. This scenario uses dnsmasq.
- ▶ NFS is not needed unless you need storage for your pods.

After setting up your services, enable them by running the `systemctl` service. Enable the necessary ports on your local firewall and SELinux.

If you need help or guidance installing these services, see the following references:

- ▶ [How to configure and set up a tftp server.](#)
- ▶ [Configuring a DHCP server.](#)
- ▶ For an HTTPd server, see [Web servers](#).
- ▶ [How to configure a DNS caching server with dnsmasq in Red Hat Enterprise Linux](#).

Getting your Red Hat OpenShift installation files from Red Hat

To start your Red Hat OpenShift cluster installation, you need a Red Hat account. Log in to your account by using the following link, and then create a cluster on the Red Hat Site and download the installation files.

<https://cloud.redhat.com/openshift/install/power/user-provisioned>

You need to download the following files:

- ▶ Red Hat OpenShift Installer for Linux (`openshift-install-linus.tar.gz`).
- ▶ The Pull Secret. Copy or save it as a text file because it is needed by the installer.
- ▶ The CLI to access the cluster.
- ▶ The Red Hat Enterprise Linux CoreOS images to create machines for your cluster to use during installation. Get all the referenced files in `grub.cfg`: `rhcros*metal.ppc64le.raw.gz`, `rhcros*initramfs.ppc64le.img`, and `rhcros*kernel-ppc64le`.

6.4.2 PowerVM configuration for network installation

The client process to install Red Hat Enterprise Linux CoreOS by using the network boot process is the same as in any AIX NIM installation. For example, this installation covers the installation of Red Hat OpenShift Container Platform V4.4 across two Power S924 servers with enough hardware for a starter kit.

Figure 6-15 on page 95 shows the network connections for a minimal workload with IBM Spectrum Scale as the storage back end for ReadWriteMany (rwx) Persistent Volumes (PVs).

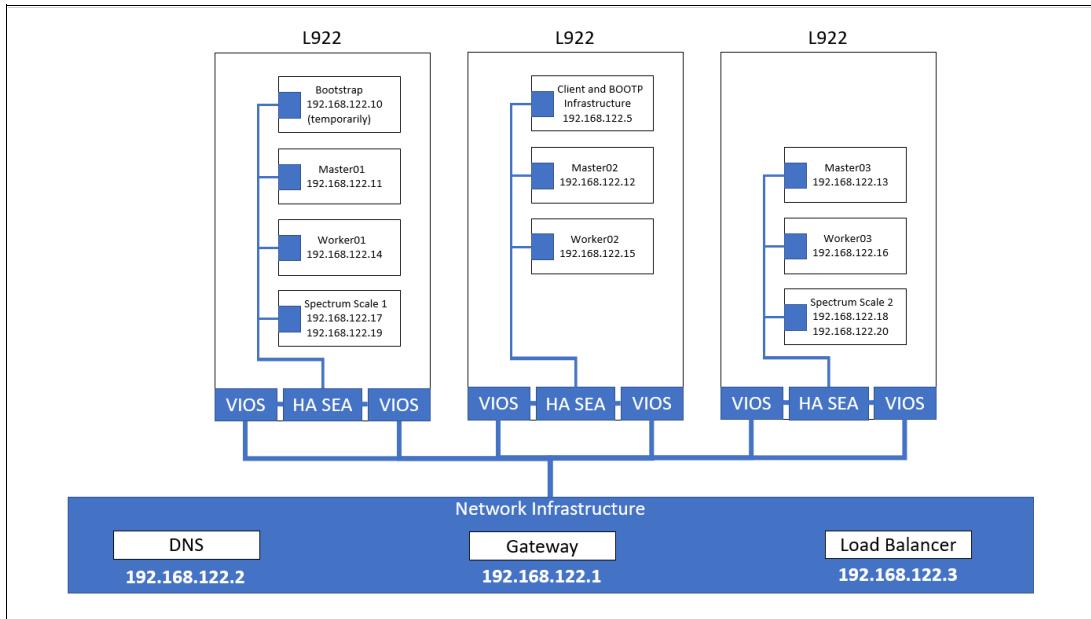


Figure 6-15 PowerVM network architecture example

This test environment uses other servers to act as the DNS and load balancer because we do not have control over the enterprise network resources. Because this configuration is a single point of failure, do not use this method on a production environment. Instead, use HA network services that are provided by the enterprise networking, infrastructure, and security teams. The DHCP/BOOTP in this case is another single server that can be used for production environments if the nodes are installed with a fixed IP address and the server is used only during installation.

The installation server can be your NIM server if you have one, as described in 3.3, “Using NIM as a BOOTP infrastructure”, in *Red Hat OpenShift V4.3 on IBM Power Systems Reference Guide*, REDP-5599.

6.4.3 Configuring the DHCP server

Now, you must configure the DHCP server on the deployment machine by editing the /etc/dhcp/dhcpd.conf file, as shown in Example 6-11.

Example 6-11 Example of the /etc/dhcp/dhcpd.conf entry

```

default-lease-time 900;
max-lease-time 7200;
subnet 192.168.197.128 netmask 255.255.255.128
{
    option routers 192.168.197.254;
    option subnet-mask 255.255.255.128;
    option domain-search "ocp44.dgi.lab";
    option domain-name-servers 192.168.197.202, 192.168.150.244;
    next-server 192.168.197.202;
    filename "boot/grub2/powerpc-ieee1275/core.elf";
    allow bootp;option conf-file code 209 = text;

host bootstrap
{
    hardware ethernet de:2c:67:d3:d6:02;
    fixed-address 192.168.197.203;
}

```

```

        option host-name "bootstrap.ocp4.dgi.lab";
        allow booting; }

host master1
{   hardware ethernet de:2c:61:e1:19:02;
    fixed-address 192.168.197.204;
    option host-name "master1.ocp4.dgi.lab";
    allow booting; }

host master2
{   hardware ethernet de:2c:60:99:81:02;
    fixed-address 192.168.197.205;
    option host-name "master2.ocp4.dgi.lab";
    allow booting; }

host master3
{   hardware ethernet de:2c:6c:05:61:02;
    fixed-address 192.168.197.206;
    option host-name "master3.ocp4.dgi.lab";
    allow booting; }

host worker1
{   hardware ethernet de:2c:68:4c:39:02;
    fixed-address 192.168.197.207;
    option host-name "worker1.ocp4.dgi.lab";
    allow booting; }

host worker2
{   hardware ethernet de:2c:60:8b:1a:02;
    fixed-address 192.168.197.208;
    option host-name "worker2.ocp4.dgi.lab";
    allow booting; }

host bootstrap44
{   hardware ethernet 7a:e5:c7:e8:fb:02;
    fixed-address 192.168.197.215;
    option host-name "bootstrap.ocp44.celeste.uy";
    allow booting; }

host master441
{   hardware ethernet 7a:e5:cc:4e:d4:02;
    fixed-address 192.168.197.216;
    option host-name "master44-1.ocp44.celeste.uy";
    allow booting; }

host master442
{   hardware ethernet 7a:e5:cf:50:2f:02;
    fixed-address 192.168.197.217;
    option host-name "master44-2.ocp44.celeste.uy";
    allow booting; }

host master443
{   hardware ethernet 7a:e5:c1:8e:a9:02;
    fixed-address 192.168.197.218;

```

```

option host-name "master44-3.ocp44.celeste.uy";
allow booting;

host worker441
{
    hardware ethernet 7a:e5:c2:4b:42:02;
    fixed-address 192.168.197.219;
    option host-name "worker44-1.ocp44.celeste.uy";
    allow booting;

host worker442
{
    hardware ethernet 7a:e5:cb:59:e4:02;
    fixed-address 192.168.197.220;
    option host-name "worker44-2.ocp44.celeste.uy";
    allow booting;

```

You notice that there are two sets of machines because we used two clusters: a cluster with Red Hat OpenShift V4.3, and a cluster with Red Hat OpenShift V4.4.

6.4.4 Starting the servers to install Red Hat Enterprise Linux CoreOS

In this example, we use the network boot method to convert the PXE boot that is present on the official document to the BOOTP installation for PowerVM LPARs. Most existing customers are using BOOTP on NIM. This section shows how to create a Linux environment to separate the NIM server from the Red Hat OpenShift installation server. You can also unify the servers and use an existing NIM server to be your network installation infrastructure (BOOTP and TFTP), as described in 3.3, “Using NIM as a BOOTP infrastructure”, in *Red Hat OpenShift V4.3 on IBM Power Systems Reference Guide*, REDP-5599.

The rhcos*metal.ppc64le.raw.gz file must be placed on the root directory of the HTTP Server, and both rhcos*initramfs.ppc64le.img and rhcos*kernel-ppc64le must be on the TFTP root directory.

Note: At the time of writing, Red Hat OpenShift V4.5 was implemented. You can select the version that you want, either [Version 4.4](#) or [Version 4.3](#).

Configure the grub load for each server on /var/lib/tftpboot/boot/grub2/grub.cfg. Example 6-12 shows entries for each kind.

Example 6-12 Example for each entry on /var/lib/tftpboot/boot/grub2/grub.cfg

```

default-lease-time 900;

if [ ${net_default_mac} == 7a:e5:c7:e8:fb:02 ];
then
    default=0
    fallback=1
    timeout=1
    menuentry "Bootstrap CoreOS (BIOS)" {
        echo "Loading kernel Bootstrap"
        linux /rhcos-4.4.9-ppc64le-installer-kernel-ppc64le rd.neednet=1
coreos.inst=yes coreos.inst.install_dev=sdb
coreos.inst.image_url=http://192.168.197.202/isos/rhcos-4.4.9-ppc64le-metal.ppc64l

```

```

e.raw.gz coreos.inst.ignition_url=http://192.168.197.202/bootstrap.ign
ip=192.168.197.215::192.168.197.254:255.255.255.128:bootstrap.ocp44.celeste.uy:env
2:none nameserver=192.168.197.202 nameserver=192.168.150.244
        echo " Loading initrd"
        initrd /rhcros-4.4.9-ppc64le-installer-initramfs.ppc64le.img
    }
fi

if [ ${net_default_mac} == 7a:e5:cc:4e:d4:02 ];
then
    default=0
    fallback=1
    timeout=1
    menuentry "Master1 CoreOS (BIOS)" {echo "Loading kernel Master1"
        linux "/rhcros-4.4.9-ppc64le-installer-kernel-ppc64le" rd.neednet=1
coreos.inst=yes coreos.inst.install_dev=sdb
coreos.inst.image_url=http://192.168.197.202/isos/rhcros-4.4.9-ppc64le-metal.ppc64l
e.raw.gz coreos.inst.ignition_url=http://192.168.197.202/master.ign
ip=192.168.197.216::192.168.197.254:255.255.255.128:master44-1.ocp44.celeste.uy:en
v2:none nameserver=192.168.197.202 nameserver=192.168.150.244
        echo "Loading initrd"
        initrd "/rhcros-4.4.9-ppc64le-installer-initramfs.ppc64le.img"}
    fi

if [ ${net_default_mac} == 7a:e5:cf:50:2f:02 ];
then
    default=0
    fallback=1
    timeout=1
    menuentry "Master2 CoreOS (BIOS)" {echo "Loading kernel Master2"
        linux /rhcros-4.4.9-ppc64le-installer-kernel-ppc64le rd.neednet=1
coreos.inst=yes coreos.inst.install_dev=sdb
coreos.inst.image_url=http://192.168.197.202/isos/rhcros-4.4.9-ppc64le-metal.ppc64l
e.raw.gz coreos.inst.ignition_url=http://192.168.197.202/master.ign
ip=192.168.197.217::192.168.197.254:255.255.255.128:master44-2.ocp44.celeste.uy:en
v2:none nameserver=192.168.197.202 nameserver=192.168.150.244
        echo "Loading initrd"
        initrd /rhcros-4.4.9-ppc64le-installer-initramfs.ppc64le.img
    }
fi

if [ ${net_default_mac} == 7a:e5:c1:8e:a9:02 ];
then
    default=0
    fallback=1
    timeout=1
    menuentry "Master3 CoreOS (BIOS)" {echo "Loading kernel Master3"
        linux /rhcros-4.4.9-ppc64le-installer-kernel-ppc64le rd.neednet=1
coreos.inst=yes coreos.inst.install_dev=sdb
coreos.inst.image_url=http://192.168.197.202/isos/rhcros-4.4.9-ppc64le-metal.ppc64l
e.raw.gz coreos.inst.ignition_url=http://192.168.197.202/master.ign
ip=192.168.197.218::192.168.197.254:255.255.255.128:master44-3.ocp44.celeste.uy:en
v2:none nameserver=192.168.197.202 nameserver=192.168.150.244
        echo "Loading initrd"
        initrd /rhcros-4.4.9-ppc64le-installer-initramfs.ppc64le.img

```

```

}

fi

if [ ${net_default_mac} == 7a:e5:c2:4:42:02 ];
then
    default=0
    fallback=1
    timeout=1
    menuentry "Worker1 CoreOS (BIOS)" { echo "Loading kernel Worker1"
        linux /rhcos-4.4.9-ppc64le-installer-kernel-ppc64le nomodeset rd.neednet=1
        coreos.inst=yes coreos.inst.install_dev=sdb
        coreos.inst.image_url=http://192.168.197.202/isos/rhcos-4.4.9-ppc64le-metal.ppc64l
        e.raw.gz coreos.inst.ignition_url=http://192.168.197.202/ign44/worker.ign
        ip=192.168.197.219::192.168.197.254:255.255.255.128:worker44-1.ocp44.celeste.uy:en
        v2:none nameserver=192.168.197.202 nameserver=192.168.150.244
        echo "Loading initrd"
        initrd /rhcos-4.4.9-ppc64le-installer-initramfs.ppc64le.img
    }
fi

if [ ${net_default_mac} == 7a:e5:cb:59:e4:02 ];
then
    default=0
    fallback=1
    timeout=1
    menuentry "Worker2 CoreOS (BIOS)" { echo "Loading kernel Worker2"
        linux /rhcos-4.4.9-ppc64le-installer-kernel-ppc64le nomodeset rd.neednet=1
        coreos.inst=yes coreos.inst.install_dev=sdb
        coreos.inst.image_url=http://192.168.197.202/isos/rhcos-4.4.9-ppc64le-metal.ppc64l
        e.raw.gz coreos.inst.ignition_url=http://192.168.197.202/worker.ign
        ip=192.168.197.220::192.168.197.254:255.255.255.128:worker44-2.ocp44.celeste.uy:en
        v2:none nameserver=192.168.197.202 nameserver=192.168.150.244
        echo "Loading initrd"
        initrd /rhcos-4.4.9-ppc64le-installer-initramfs.ppc64le.img
    }
fi

```

Note: When configuring your dhcp.conf and grub.cfg environment files, the MAC address must be in lowercase like de:2c:6c:05:61:02. Problems at start time can appear if the MAC address is in uppercase.

6.4.5 Installing from ISO instead of tftp

If you do not have a tftp or NIM server available or enough hardware to build one for the installation, you may perform the installation by using a Red Hat Enterprise Linux CoreOS ISO image. For that kind of installation, you do not need a DHCP server or TFTP server, but only the HTTPd server from which to download your images. Then, you map the ISO to the LPARs.

Complete the following steps:

1. With your ISO mapped, start the system by using a DVD and press e when the boot menu opens, as shown in Figure 6-16.

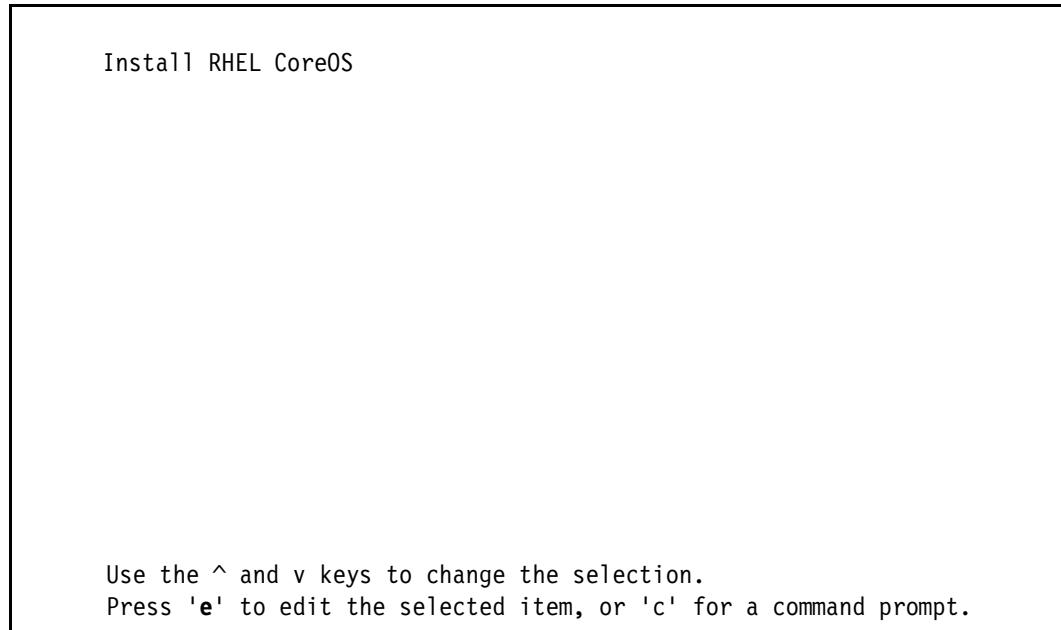


Figure 6-16 Boot from ISO panel

2. After pressing e, you are redirected to a panel where you modify the boot parameters. You can set IP addresses, a gateway, kernel, installation destination, and other settings, as shown in Figure 6-17 on page 101.

```

setparams 'Install RHEL CoreOS'

    linux /images/vmlinuz nomodeset rd.neednet=1 coreos.inst=yes
coreos.in\
st.install_dev=sdb
coreos.inst.image_url=http://192.168.197.202/isos/rhcos-4.4\
.9-ppc64le-metal.ppc64le.raw.gz
coreos.inst.ignition_url=http://192.168.197.20\
2/bootstrap.ign
ip=192.168.197.203::192.168.197.254:255.255.255.128:bootstrap\
.ocp4.ibm.lab:env2:none nameserver=192.168.197.202
nameserver=192.168.150.244 \
initrd /images/initramfs.img

```

Press **Ctrl-x** to start, **Ctrl-c** for a command prompt or **Escape** to discard edits and return to the menu. Pressing **Tab** lists possible completions.

Figure 6-17 Boot from ISO customization for machine deployment

3. After you finish editing your boot parameters, press **Ctrl+x** to start the installation. You must repeat the process for each of your nodes.

The advantage of this process is that it requires fewer services than the tftp and NIM method, but it requires more manual effort. It also is preferable if your company has security restrictions regarding the use of tftp or dhcp.

The configuration that is used for each node is shown in Example 6-13.

Example 6-13 Boot parameters for an ISO installation

For Bootstrap -

```

linux /images/vmlinuz nomodeset rd.neednet=1 coreos.inst=yes
coreos.inst.install_dev=sdb
coreos.inst.image_url=http://192.168.197.202/isos/rhcos-4.4.9-ppc64le-metal.ppc
64le.raw.gz coreos.inst.ignition_url=http://192.168.197.202/bootstrap.ign
ip=192.168.197.203::192.168.197.254:255.255.255.128:bootstrap.ocp4.ibm.lab:env2
:none nameserver=192.168.197.202 nameserver=192.168.150.244

```

For Master1 -

```

linux /images/vmlinuz nomodeset rd.neednet=1 coreos.inst=yes
coreos.inst.install_dev=sdb
coreos.inst.image_url=http://192.168.197.202/isos/rhcos-4.4.9-ppc64le-metal.ppc
64le.raw.gz coreos.inst.ignition_url=http://192.168.197.202/master.ign
ip=192.168.197.204::192.168.197.254:255.255.255.128:master1.ocp4.ibm.lab:env2:n
one nameserver=192.168.197.202 nameserver=192.168.150.244

```

For Master2 -

```

linux /images/vmlinuz nomodeset rd.neednet=1 coreos.inst=yes
coreos.inst.install_dev=sdb

```

```

coreos.inst.image_url=http://192.168.197.202/isos/rhcos-4.4.9-ppc64le-metal.ppc
64le.raw.gz coreos.inst.ignition_url=http://192.168.197.202/master.ign
ip=192.168.197.205::192.168.197.254:255.255.255.128:master2.ocp4.ibm.lab:env2:n
one nameserver=192.168.197.202 nameserver=192.168.150.244

For Master3 -
linux /images/vmlinuz nomodeset rd.neednet=1 coreos.inst=yes
coreos.inst.install_dev=sdb
coreos.inst.image_url=http://192.168.197.202/isos/rhcos-4.4.9-ppc64le-metal.ppc
64le.raw.gz coreos.inst.ignition_url=http://192.168.197.202/master.ign
ip=192.168.197.206::192.168.197.254:255.255.255.128:master3.ocp4.ibm.lab:env2:n
one nameserver=192.168.197.202 nameserver=192.168.150.244

For Worker1 -
linux /images/vmlinuz nomodeset rd.neednet=1 coreos.inst=yes
coreos.inst.install_dev=sdb
coreos.inst.image_url=http://192.168.197.202/isos/rhcos-4.4.9-ppc64le-metal.ppc
64le.raw.gz coreos.inst.ignition_url=http://192.168.197.202/worker.ign
ip=192.168.197.207::192.168.197.254:255.255.255.128:worker1.ocp4.ibm.lab:env2:n
one nameserver=192.168.197.202 nameserver=192.168.150.244

For Worker2 -
linux /images/vmlinuz nomodeset rd.neednet=1 coreos.inst=yes
coreos.inst.install_dev=sdb
coreos.inst.image_url=http://192.168.197.202/isos/rhcos-4.4.9-ppc64le-metal.ppc
64le.raw.gz coreos.inst.ignition_url=http://192.168.197.202/worker.ign
ip=192.168.197.208::192.168.197.254:255.255.255.128:worker2.ocp4.ibm.lab:env2:n
one nameserver=192.168.197.202 nameserver=192.168.150.244

```

With your LPARs installed, the installation proceeds from this point the same as with a tftpt or NIM installation.

6.4.6 Configuring the DNS server

Your Red Hat OpenShift environment must use a DNS service. You can set up binding or other similar DNS software, but this test environment uses dnsmasq as a local DNS for our environment.

dnsmasq supports the configuration in /etc/dnsmasq.conf or the included configurations in the /etc/dnsmasq.d/ files. In our example, we configured the DHCP server on the deployment machine by editing /etc/dnsmasq.d/openshift.conf, as shown in Example 6-14.

Example 6-14 Example of the edited /etc/dnsmasq.d/openshift.conf file

```

address=/client.ocp4.dgi.lab/192.168.197.202

address=/bootstrap44.ocp44.celeste.uy/192.168.197.215
ptr-record=215.197.168.192.in-addr.arpa,bootstrap.ocp44.celeste.uy

address=/master44-1.ocp44.celeste.uy/192.168.197.216
address=/etcd-0.ocp44.celeste.uy/192.168.197.216
srv-host=_etcd-server-ssl._tcp.ocp44.celeste.uy,etcd-0.ocp44.celeste.uy,2380
ptr-record=216.197.168.192.in-addr.arpa,master44-1.ocp44.celeste.uy

address=/master44-2.ocp44.celeste.uy/192.168.197.217

```

```

address=/etcd-1.ocp44.celeste.uy/192.168.197.217
srv-host=_etcd-server-ssl._tcp.ocp44.celeste.uy,etcd-1.ocp44.celeste.uy,2380
ptr-record=217.197.168.192.in-addr.arpa,master44-2.ocp44.celeste.uy

address=/master44-3.ocp44.celeste.uy/192.168.197.218
address=/etcd-2.ocp44.celeste.uy/192.168.197.218
srv-host=_etcd-server-ssl._tcp.ocp44.celeste.uy,etcd-2.ocp44.celeste.uy,2380
ptr-record=218.197.168.192.in-addr.arpa,master44-3.ocp44.celeste.uy

address=/worker44-1.ocp44.celeste.uy/192.168.197.219
ptr-record=219.197.168.192.in-addr.arpa,worker44-1.ocp44.celeste.uy

address=/worker44-2.ocp44.celeste.uy/192.168.197.220
ptr-record=220.197.168.192.in-addr.arpa,worker44-2.ocp44.celeste.uy

address=/api.ocp44.celeste.uy/192.168.197.221
address=/api-int.ocp44.celeste.uy/192.168.197.221
address=/apps.ocp44.celeste.uy/192.168.197.221

# Listen on lo and env2 only
bind-interfaces
interface=lo,env2
server=192.168.150.244
server=192.168.150.245

```

6.4.7 Configuring the load balancer

Any installation of Red Hat OpenShift requires at least one load balancer to balance the access to the master and worker nodes, and enable or disable them in case of failure or maintenance. A best practice for production environments is to have two load balancers. For this publication, our team installed a Red Hat Enterprise Linux partition running HAProxy.

You must configure the HAProxy software in the load balancer defined machine. For more information, see [How do I set up HAProxy as a load balancer](#).

Example 6-15 shows the `haproxy.cfg` file that was created for our cluster.

Example 6-15 The /etc/haproxy/haproxy.cfg file

```

address=/client.ocp4.dgi.lab/192.168.197.202

global
  log 127.0.0.1 local0 warning
  chroot      /var/lib/haproxy
  pidfile    /var/run/haproxy.pid
  maxconn   4000
  user        haproxy
  group       haproxy
  daemon
  stats socket /var/lib/haproxy/stats
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

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

backend openshift-api
  balance source
  mode tcp
  server ocp44-bootstrap 192.168.197.215:6443 check
  server ocp44-master1 192.168.197.216:6443 check
  server ocp44-master2 192.168.197.217:6443 check
  server ocp44-master3 192.168.197.218:6443 check

frontend openshift-configserver
  bind *:22623
  default_backend openshift-configserver
  mode tcp
  option tcplog

backend openshift-configserver
  balance source
  mode tcp
  server ocp44-bootstrap 192.168.197.215:22623 check
  server ocp44-master1 192.168.197.216:22623 check
  server ocp44-master2 192.168.197.217:22623 check
  server ocp44-master3 192.168.197.218:22623 check

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

backend openshift-http
  balance source
  mode tcp
  server ocp44-worker1 192.168.197.219:80 check
  server ocp44-worker2 192.168.197.220:80 check

frontend openshift-https
  bind *:443
  default_backend openshift-https
  mode tcp
  option tcplog

backend openshift-https
  balance source
  mode tcp
  server ocp44-worker1 192.168.197.219:443 check
  server ocp44-worker2 192.168.197.220:443 check

```

6.4.8 Creating the SSH key

The communication to our cluster nodes is secured by using SSH, so we create a Bastion (in this example, we used the deployment node) host to access our Red Hat OpenShift cluster machines with SSH access.

Create the SSH key with the processes that are shown in Example 6-16 and place it in the `install-config.yaml` file before installation.

Example 6-16 Creating the SSH key to connect to your nodes

```
[root@openshift-deploy-os-44 ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Arxjp1ceE95Ca76lq/H4K3maGqDTngk1DPEIjS8zyfs root@openshift-deploy-os44
The key's randomart image is:
+---[RSA 3072]---+
|oo
|o+..
|+o. o o
|B.. o o +
|.=+ + o S .
| * o + * +
|+ o o o.o .
| + E oo=.=+
| + ..=*Bo
+---[SHA256]---+
[root@openshift-deploy-os44 ~]# ls -lrt /root/.ssh/id_rsa.pub
-rw-r--r-- 1 root root 576 Aug 11 15:13 /root/.ssh/id_rsa.pub
[root@openshift-deploy-os44 ~]# cat /root/.ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQDQxVWM1BUxcATnIYpj7Q+ZpdoqrziI2+47LOGBeERr9DFAdHNZsZF
fEpe3CnwTjbGXMnLyr1zMus3VZHLabRIMn++pqgm/cwsjkT1p0RqmmXN1/rLHi6GOV3hqtvIM/FfW4dSrj
nqErQZQDV/y6jDPNDal/1H/qaLvFNI+2s8bdDK9L2+Eeq3qxnuIkemrrmL72zxDuMAEr3FeHNGtCEkYtRg
1V2j0BDq4Gb9oqXt1g0177AGckYt1FxLBypQKXqfdRSjeh7H1aMqyGU7mqBM/rphj06AprzSkVWA+k2GCI
xfuql1ZrnjZKzTmnMCauC9Tc0mVLOEGdS111Dq2Qm6LwUen0d2G8LnLGSIytMgn9a6SobABZHgP5/hKdkx
MyQ9KrtUg4yoMGwj1XbvihRquG/g12ye3pzRgJEOfc/YRxfzxYE+VFng0wYyUAUm0UCd68mkr417Kzh9dz
NuKNySO/v0iSPOESQnstg2VQc2WVQ8cH8W145w429S9aXxa8FcE= root@openshift-deploy-os44
[root@openshift-deploy-os44 ~]#
```

6.4.9 Installation process

During installation process, three sets of files are needed and must be created:

- ▶ `install-config.yaml`

For installations of Red Hat OpenShift on a user-provisioned infrastructure, the configuration file must be manually generated. This file describes your cluster, it has embedded the ssh-key that was created before, and it has the pull secret that was obtained from Red Hat.

Tip: Keep the `install-config.yaml` safe because the installation process uses it and then erases it.

- ▶ Kubernetes manifests
- ▶ Ignition config files

The installation configuration file is transformed into Kubernetes manifests, as shown in Example 6-17. Then, the manifests are wrapped into ignition config files, as shown in Example 6-19 on page 107.

Complete the following steps:

1. Create the `install-config.yaml` file for the Power Systems server, as shown in Example 6-17.

Example 6-17 Transforming the installation configuration file into Kubernetes manifests

```
apiVersion: v1
baseDomain: celeste.uy
proxy:
  httpProxy: http://xxxxxxxxs:xxxxxx@192.168.xx.xx:8080
  httpsProxy: http://xxxxxxxxs:xxxxxx@192.168.xx.xx:8080
  noProxy: celeste.uy,dgi.lab,192.168.197.202/25,192.168.197.221/25
compute:
  - hyperthreading: Enabled
    name: worker
    replicas: 0
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
metadata:
  name: ocp44
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  none: {}
fips: false
pullSecret: '{"auths": ...<<the pull secret downloaded from Redhat site >>... }'
sshKey: 'ssh-rsa ...<< The ssh key created above >>....'
```

2. Generate the Kubernetes manifests, as shown in Example 6-18.

Example 6-18 Generating the Kubernetes manifests

```
[root@openshift-deploy-os44 install]# ./openshift-install create manifests  
--dir=/ocp449/install/cluster-celeste  
INFO Consuming Install Config from target directory  
WARNING Making control-plane schedulable by setting MastersSchedulable to true  
for Scheduler cluster settings
```

3. The manifests and auth files are ready, but you must make a change before you create the ignition files. Edit the cluster-scheduler file to change it to false:

```
[root@openshift-deploy-os44 install]# vi  
/ocp449/install/cluster-celeste/manifests/cluster-scheduler-02-config.yml
```

```
apiVersion: config.openshift.io/v1  
kind: Scheduler  
metadata:  
  creationTimestamp: null  
  name: cluster  
spec:  
  mastersSchedulable: false <- Value must be changed to false  
  policy:  
    name: ""  
  status: {}
```

4. Using the edited cluster-scheduler properties, generate the ignitions files, as shown in Example 6-19.

Example 6-19 Generating ignitions files

```
[root@openshift-deploy-os44 install]# ./openshift-install create  
ignition-configs --dir=/ocp449/install/cluster-celeste  
INFO Consuming OpenShift Manifests from target directory  
INFO Consuming Worker Machines from target directory  
INFO Consuming Master Machines from target directory  
INFO Consuming OpenShift Install (Manifests) from target directory  
INFO Consuming Common Manifests from target directory
```

```
[root@openshift-deploy-os44 install]# ls -lrt cluster-celeste/  
total 308  
drwxr-x--- 2 root root      50 Aug 11 15:26 auth  
-rw-r----- 1 root root   1821 Aug 11 15:43 master.ign  
-rw-r----- 1 root root   1821 Aug 11 15:43 worker.ign  
-rw-r----- 1 root root 300307 Aug 11 15:43 bootstrap.ign  
-rw-r----- 1 root root     98 Aug 11 15:43 metadata.json  
[root@openshift-deploy-os44 install]#
```

5. Put the ignitions files into the DocumentRoot of the HTTP Server in the deployment machine. In most servers, it is /var/www/html.

Note: When you copy your ign files to DocumentRoot, set the correct permission to 755.

6. Start your servers, including the bootstrap server, and the master and worker nodes. After your servers are running, wait for the bootstrap to complete, as shown in Example 6-20.

Note: If you do not know how to boot in SMS mode, see Appendix B, “Booting from System Management Services”, in *Red Hat OpenShift V4.3 on IBM Power Systems Reference Guide*, REDP-5599.

Example 6-20 Waiting for the bootstrap to complete

```
./openshift-install wait-for bootstrap-complete --log-level debug
DEBUG OpenShift Installer 4.4.9
DEBUG Built from commit 1541bf917973186bbab6a5f895f08db4334a5d9a
INFO Waiting up to 20m0s for the Kubernetes API at
https://api.ocp44.celeste.uy:6443...
INFO API v1.17.1+3288478 up
INFO Waiting up to 40m0s for bootstrapping to complete...
DEBUG Bootstrap status: complete
INFO It is now safe to remove the bootstrap resources
```

After the bootstrap completes, it is safe to shut down the bootstrap server because it is not needed anymore. The master node takes control.

Attention: Before finishing your installation, if you are running PowerVM on an IBM POWER9 processor-based server, you modify the TCP/IP parameters for your LPARs, as shown in Example 6-21. For more information about this topic, see the following resources:

- ▶ Red Hat Bugzilla – Bug 1816254 [Red Hat Bugzilla - Bug 1816254](#)
- ▶ [PPC64 RHEL: OpenShift vxlan tx_dropped errors and connectivity problems](#)

7. Add the following line to /etc/sysctl.conf, as shown in Example 6-21.

Example 6-21 Modifying the TCPIP parameter

```
Add the following line to /etc/sysctl.conf:
net.ipv4.ip_no_pmtu_disc=1
Then run
# sysctl -p
```

8. When you need to log in to a cluster node, run the command that is shown in Example 6-22.

Example 6-22 Logging in to a cluster node

```
[root@openshift-deploy-os44 install]# ssh core@master44-1.ocp44.celeste.uy
Red Hat Enterprise Linux CoreOS 44.82.202007211340-0
Part of OpenShift 4.4, RHCOS is a Kubernetes native operating system
managed by the Machine Config Operator (`clusteroperator/machine-config`).

WARNING: Direct SSH access to machines is not recommended; instead,
make configuration changes through `machineconfig` objects:
  https://docs.openshift.com/container-platform/4.4/architecture/architecture-rhcos.html

---
Last login: Tue Aug  4 18:54:52 2020 from 192.168.197.202
```

Wait for the installation to complete, as shown in Example 6-23.

Example 6-23 Waiting for the installation to complete

```
[root@openshift-deploy-os44 install]# cd /ocp449/install/cluster-celeste
[root@openshift-deploy-os44 install]#../openshift-install wait-for
install-complete --log-level=debug
DEBUG OpenShift Installer 4.4.9
DEBUG Built from commit 1541bf917973186bbab6a5f895f08db4334a5d9a
DEBUG Fetching Install Config...
DEBUG Loading Install Config...
DEBUG Loading SSH Key...
DEBUG Loading Base Domain...
DEBUG Loading Platform...
DEBUG Loading Cluster Name...
DEBUG Loading Base Domain...
DEBUG Loading Platform...
DEBUG Loading Pull Secret...
DEBUG Loading Platform...
DEBUG Using Install Config loaded from state file
DEBUG Reusing previously-fetched Install Config
INFO Waiting up to 30m0s for the cluster at https://api.ocp44.celestey:6443 to
initialize...
DEBUG Cluster is initialized
INFO Waiting up to 10m0s for the openshift-console route to be created...
DEBUG Route found in openshift-console namespace: console
DEBUG Route found in openshift-console namespace: downloads
DEBUG OpenShift console route is created
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/ocp449/install/cluster-celeste/auth/kubeconfig'
INFO Access the OpenShift web-console here:
https://console-openshift-console.apps.ocp44.celestey
INFO Login to the console with user: kubeadmin, password: *****
```

6.4.10 Checking the installation

During the installation process, certificate signing requests are created, and you must approve them to finish your installation, as shown in Example 6-24.

Important: Set your KUBECONFIG environment in your run time, or in your .bash_profile or .bashrc. For example:

```
export KUBECONFIG=/ocp449/install/cluster-celeste/auth/kubeconfig
```

Example 6-24 Checking the certificates status

```
[root@openshift-deploy-os44 cluster-config]# oc get csr
NAME AGE REQUESTER CONDITION
csr-4bjcc 24m system:node:worker44-2.ocp44.celestey Pending
csr-7bm5r 40m system:node:master44-2.ocp44.celestey Approved,Issued
csr-7rg2t 26m
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-8qbk5 40m system:node:master44-3.ocp44.celestey Approved,Issued
```

```

csr-9cz9q 23m system:node:worker44-1.ocp44.celeste.uy Pending
csr-jgk69 39m system:node:master44-1.ocp44.celeste.uy Approved,Issued
csr-l9r5c 40m
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-lpbsj 40m
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-nwbwc 39m
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-qwlzh 26m
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued

```

Tip: Either of the following commands can help to bulk approve your certificates when your cluster first starts:

- ▶ `oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs oc adm certificate approve`
- ▶ `for i in $(oc get csr | grep Pending | awk '{ print $1 }'); do echo $i;oc adm certificate approve $i; done`

After your certificates are approved, your nodes are running, and you are ready to finish your installation. You might need to run the certificate approval process more than once.

Complete the following steps:

1. Check the status of your nodes by running the command shown in Example 6-25.

Example 6-25 Checking the nodes status

NAME	STATUS	ROLES	AGE	VERSION
master44-1.ocp44.celeste.uy	Ready	master	47h	v1.17.1+3288478
master44-2.ocp44.celeste.uy	Ready	master	47h	v1.17.1+3288478
master44-3.ocp44.celeste.uy	Ready	master	47h	v1.17.1+3288478
worker44-1.ocp44.celeste.uy	Ready	worker	34h	v1.17.1+3288478
worker44-2.ocp44.celeste.uy	Ready	worker	34h	v1.17.1+3288478

You can see the status of all the cluster operators, as shown in Example 6-26.

Example 6-26 Cluster operators status

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.4.14	True	False	False	3d12h
cloud-credential	4.4.14	True	False	False	4d1h
cluster-autoscaler	4.4.14	True	False	False	4d
console	4.4.14	True	False	False	3d6h
csi-snapshot-controller	4.4.14	True	False	False	3d12h
dns	4.4.14	True	False	False	4d
etcd	4.4.14	True	False	False	4d
image-registry	4.4.14	True	False	False	2d1h
ingress	4.4.14	True	False	False	3d8h
insights	4.4.14	True	False	False	4d
kube-apiserver	4.4.14	True	False	False	4d
kube-controller-manager	4.4.14	True	False	False	4d

kube-scheduler	4.4.14	True	False	False	4d
kube-storage-version-migrator	4.4.14	True	False	False	3d8h
machine-api	4.4.14	True	False	False	4d
machine-config	4.4.14	True	False	False	3d7h
marketplace	4.4.14	True	False	False	3d6h
monitoring	4.4.14	True	False	False	3d6h
network	4.4.14	True	False	False	4d
node-tuning	4.4.14	True	False	False	3d10h
openshift-apiserver	4.4.14	True	False	False	3d6h
openshift-controller-manager	4.4.14	True	False	False	3d3h
openshift-samples	4.4.14	True	False	False	3d10h
operator-lifecycle-manager	4.4.14	True	False	False	4d
operator-lifecycle-manager-catalog	4.4.14	True	False	False	4d
operator-lifecycle-manager-packageserver	4.4.14	True	False	False	3d6h
service-ca	4.4.14	True	False	False	4d
service-catalog-apiserver	4.4.14	True	False	False	4d
service-catalog-controller-manager	4.4.14	True	False	False	4d
storage	4.4.14	True	False	False	3d10h

If all cluster operators are true and none are progressing, your cluster is ready. You can see the status and version of the entire Red Hat OpenShift cluster as shown in Example 6-27.

Example 6-27 Cluster version

```
[root@openshift-deploy-os44 install]# oc get clusterversion
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE     STATUS
version   4.4.14    True        False         3d5h      Cluster version is 4.4.14
```

2. The cluster is up and the machines are running. Access the admin console by using the following link:

<https://console-openshift-console.apps.ocp44.celestey.uy>

Figure 6-18 show the admin console. You must log in by using the credentials that are provided at the end of you **openshift-install wait-for complete** command, as shown in Example 6-23 on page 109.

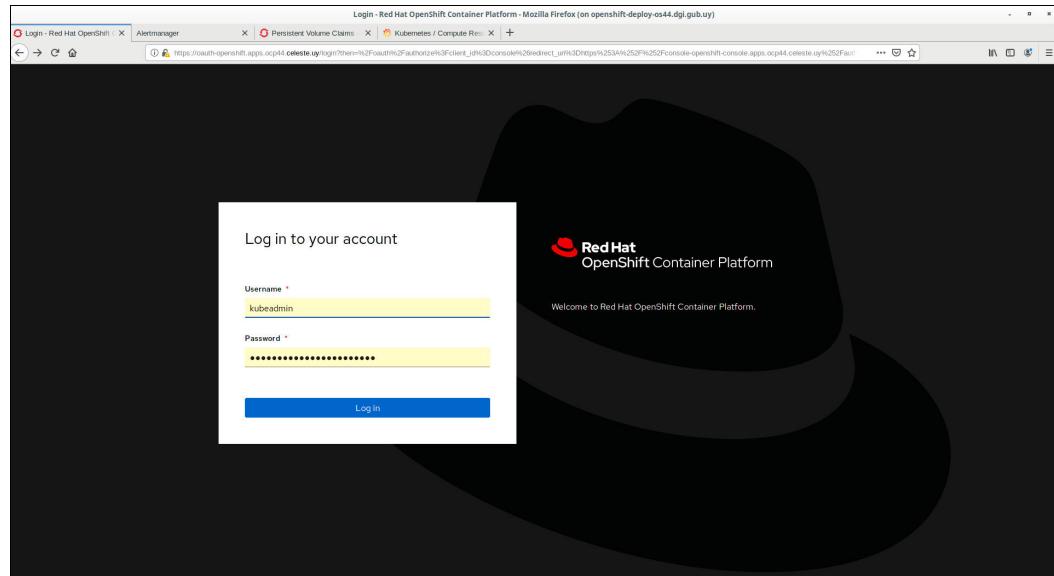


Figure 6-18 Red Hat OpenShift Console login window

The Red Hat OpenShift dashboard shows the status of your cluster, as shown in Figure 6-19.

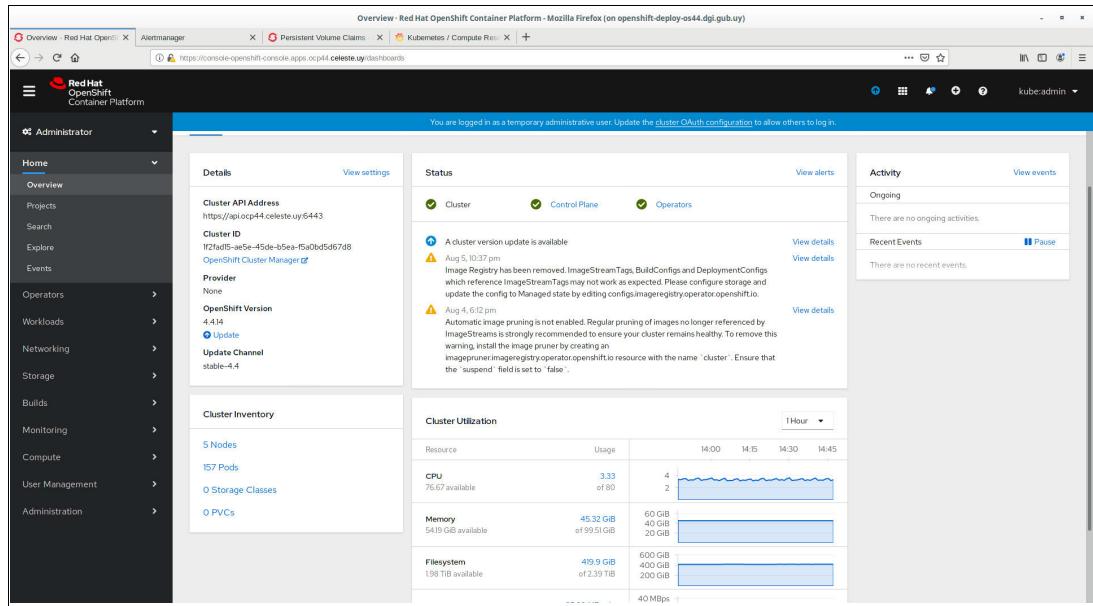


Figure 6-19 Red Hat OpenShift dashboard

6.4.11 Backing up your cluster

After your cluster is running, it is a best practice to take a backup if something fails so that you do not need to reinstall your cluster.

To take a backup in Red Hat OpenShift V4.4, complete the following steps:

1. Connect to one of the master servers. In this example, we use master44-2.
2. From your deployment node, run the following command:
`oc debug node/master44-2`
3. Connect as root by running the following command:
`chroot /host`
4. Back up your system by running the following command:
`/usr/local/bin/cluster-backup.sh /home/core/assets/backup`

The entire process is shown in Example 6-28.

Example 6-28 Creating a cluster backup

```
[root@openshift-deploy-os44 ~]# oc debug node/master44-2.ocp44.celestey.uy
Starting pod/master44-2ocp44celestey-debug ...
To use host binary files, run `chroot /host`
chroot Pod IP: 192.168.197.217
If you don't see a command prompt, try pressing enter.
sh-4.2# chroot /host
sh-4.4#/usr/local/bin/cluster-backup.sh /home/core/assets/backup
55a6ae2663667c95ceda0408d06528a57aff800e1e386e61b210ced8a9c7b5cf
etcdctl version: 3.3.22
API version: 3.3
found latest kube-apiserver-pod:
/etc/kubernetes/static-pod-resources/kube-apiserver-pod-22
```

```
found latest kube-controller-manager-pod:  
/etc/kubernetes/static-pod-resources/kube-controller-manager-pod-10  
found latest kube-scheduler-pod:  
/etc/kubernetes/static-pod-resources/kube-scheduler-pod-3  
found latest etcd-pod: /etc/kubernetes/static-pod-resources/etcd-pod-3  
2020-09-21 19:04:29.273218 I | clientv3: opened snapshot stream; downloading  
2020-09-21 19:04:30.786448 I | clientv3: completed snapshot read; closing  
Snapshot saved at /home/core/asssets/backup/snapshot_2020-09-21_190427.db  
snapshot db and kube resources are successfully saved to /home/core/asssets/backup  
sh-4.4#
```

The process is slightly different depending on your Red Hat OpenShift cluster version. The complete documentation for the process is at the following websites:

- ▶ [Red Hat OpenShift V4.3](#)
- ▶ [Red Hat OpenShift V4.4](#)



A

IBM Power Systems in the hybrid cloud world

This chapter provides a practical guide for IBM Power Systems users to gain an understanding about the Power Systems cloud portfolio, and map out a journey to a secure and reliable hybrid cloud world.

This appendix contains the following topics:

- ▶ IBM Power Systems: Journey to hybrid cloud
- ▶ Introducing Red Hat Ansible for Power Systems
- ▶ Introducing Red Hat Ansible for IBM i
- ▶ Sample labs: Deploying IBM virtual machines into a hybrid cloud
- ▶ Moving a VM to IBM Cloud Object Storage in IBM Cloud from IBM PowerVC
- ▶ Deploying an IBM virtual machine by using IBM Cloud Pak for Multicloud Management in IBM PowerVC
- ▶ Deploying IBM VM into IBM Power Systems Virtual Server in IBM Cloud by using IBM Cloud Pak for Multicloud Management

IBM Power Systems: Journey to hybrid cloud

The cloud has been present in the IT industry and marketplace for more than a decade. The *hybrid cloud* is a computing environment that combines a private cloud and a public cloud by allowing applications and data to be shared between them. The hybrid cloud came into existence because there is a need for an amalgamation of two or more various environments.

Customers have massive investments in traditional applications in the data center that they need to modernize and move to the cloud. IBM stands apart from the competition regarding the hybrid cloud because it provides customers a choice of more cloud models (private, dedicated, public, and managed) and extensive hybrid integration capabilities that provide more flexibility and options for connecting clouds to a customer's existing IT environment.

Today, enterprises must innovate to create customer experiences and meet rapidly changing customer and workforce demands. Enterprises are increasingly adopting containers to build applications faster and deliver and scale. With containers, developers can focus on building innovative applications and ensure portability across hybrid cloud environments.

Docker and Kubernetes are the leading open-source technologies for containerization, along with a growing ecosystem of open source tools. Red Hat OpenShift is the leading open-source container platform and the preferred container management platform for enterprises.

Red Hat OpenShift is targeted at both developers and IT operations teams. With Red Hat OpenShift, developers can build cloud-native applications quickly, and IT operations can manage environments securely and at scale. Enterprises are using Red Hat OpenShift to drive hybrid cloud adoption, shorten development cycles, and lower the cost of operations.

Against this background, IBM enhanced the IBM Power Systems portfolio to provide flexibility for systems that operate together as a pool of resources. The new consumption models for IBM Power Systems with hybrid cloud enhancements enable you to optimize costs and improve continuity as you build a seamless hybrid cloud. Combined with Red Hat OpenShift, IBM Cloud Paks, and Red Hat Ansible automation, you have the keys to a robust modern IT infrastructure that extends the value of your existing investments.

Figure A-1 on page 117 shows the comprehensive hybrid cloud management infrastructure for IBM Power Systems servers.

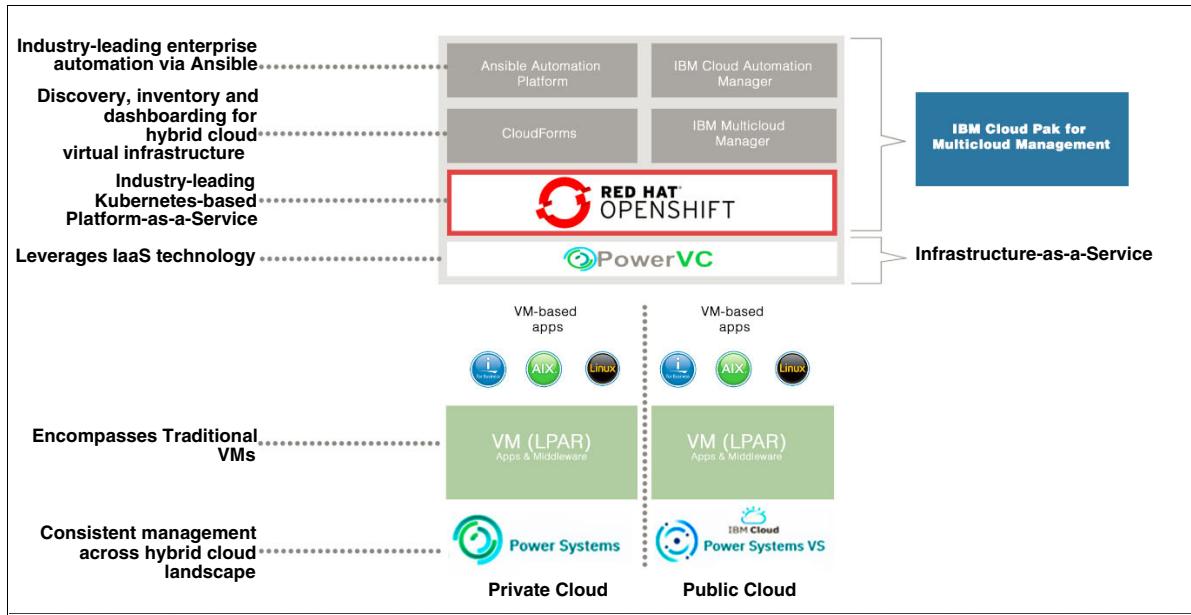


Figure A-1 Hybrid cloud management and enterprise automation for IBM Power Systems servers

Introducing Red Hat Ansible for Power Systems

This section describes Red Hat Ansible for Power Systems, and provides an overview and introduction. It describes collections, modules, and how to install Red Hat Ansible for IBM i. It also shows some examples.

Overview

The IBM Power Systems platform is enabled for Red Hat Ansible Automation Platform, which means that there is support for full automation across IBM AIX, IBM i, and Linux for Power Systems operating systems (OSs). These environments run on-premises as Private Cloud - IBM Power Virtualization Center (IBM PowerVC) and off-premises as Public Cloud - IBM Cloud or Hybrid Cloud.

Red Hat Ansible is an open-source, third-party tool that you can use for management and automation of repetitive tasks. Red Hat Ansible is agent-less and performs actions on a set of servers from a Red Hat Ansible control node. Red Hat Ansible is installed only on the system that operates as the control node.

There are extensive sets of Red Hat Ansible modules that are available for IBM Power Systems for AIX, IBM i, and Linux to automate operations.

This section introduces and describes Red Hat Ansible.

What is Red Hat Ansible

Red Hat Ansible is an IT automation tool that can automate repetitive tasks. It is an open-source automation platform. It is easy to learn, simple to set up, and easy to write. Red Hat Ansible can help users with configuration management, application deployment, cloud provisioning, ad hoc task execution, network automation, and multi-node orchestration. Red Hat Ansible makes complex changes like zero-downtime rolling updates with load balancers easy.

Red Hat Ansible does not use an agent on the remote host. Instead, it uses Secure Shell (SSH), which is assumed to be installed on the Red Hat Ansible control node and all the systems to be managed. Also, Red Hat Ansible is written in Python, which must be installed on the control node and all the systems to be managed.

Why Red Hat Ansible on Red Hat OpenShift

Red Hat Ansible is an open source technology for automation of workloads and provisioning tasks. It is embedded in Red Hat OpenShift, and it supports the automation of mult-tier workloads across hundreds (or even thousands) of nodes. By using easy to understand playbooks that are written in YAML, entire organizations can benefit from the automation process.

Popular use cases (Figure A-2 on page 119) for Red Hat Ansible in with Red Hat OpenShift include:

- ▶ Configuration management
- ▶ Application deployment
- ▶ Security and compliance
- ▶ Continuous integration (CI) and continuous deployment (CD) (CI/CD) pipelines
- ▶ Orchestration
- ▶ Provisioning

Red Hat Ansible and Red Hat OpenShift provide enterprise-grade security for the management of both infrastructure and containerized applications:

- ▶ Red Hat-exclusive Security Enhanced Linux (SELinux) kernel at the core of every Red Hat OpenShift environment isolates containers from the host kernel and containers from each other.
- ▶ Red Hat Ansible uses SSH to interface directly with automated hosts, ensuring that Red Hat Ansible-automated machines cannot see or affect how other machines are configured.
- ▶ On infrastructure or hosts that will be automated, no administrator access is required, and no dedicated users must be created specifically for Red Hat Ansible.

Red Hat Ansible architecture

Red Hat Ansible has a clear architecture with a few principle components that are important to understand:

- ▶ Red Hat Ansible Tower: The commercial form of Red Hat Ansible AWX (that is, the open-source version), Red Hat Ansible Tower provides a GUI to scale Red Hat Ansible across the entire enterprise from a highly consumable and convenient interface and runs on x86 Linux.
- ▶ Red Hat Ansible Engine: The engine (control node) is the system on which Red Hat Ansible is installed and used to run playbooks (that is, the files where Red Hat Ansible code is written and used to tell Red Hat Ansible what to run). It runs on x86 Linux.
- ▶ Red Hat Ansible Endpoints: Endpoints OS on which Red Hat Ansible modules run. From the IBM Power Systems perspective, the endpoints include AIX, IBM i, and Linux on Power Systems, but also includes other OSs (such as IBM z/OS® and Microsoft Windows) for a consistent Red Hat Ansible management experience across the entire data center.

Figure A-2 shows the architecture of Red Hat Ansible components.

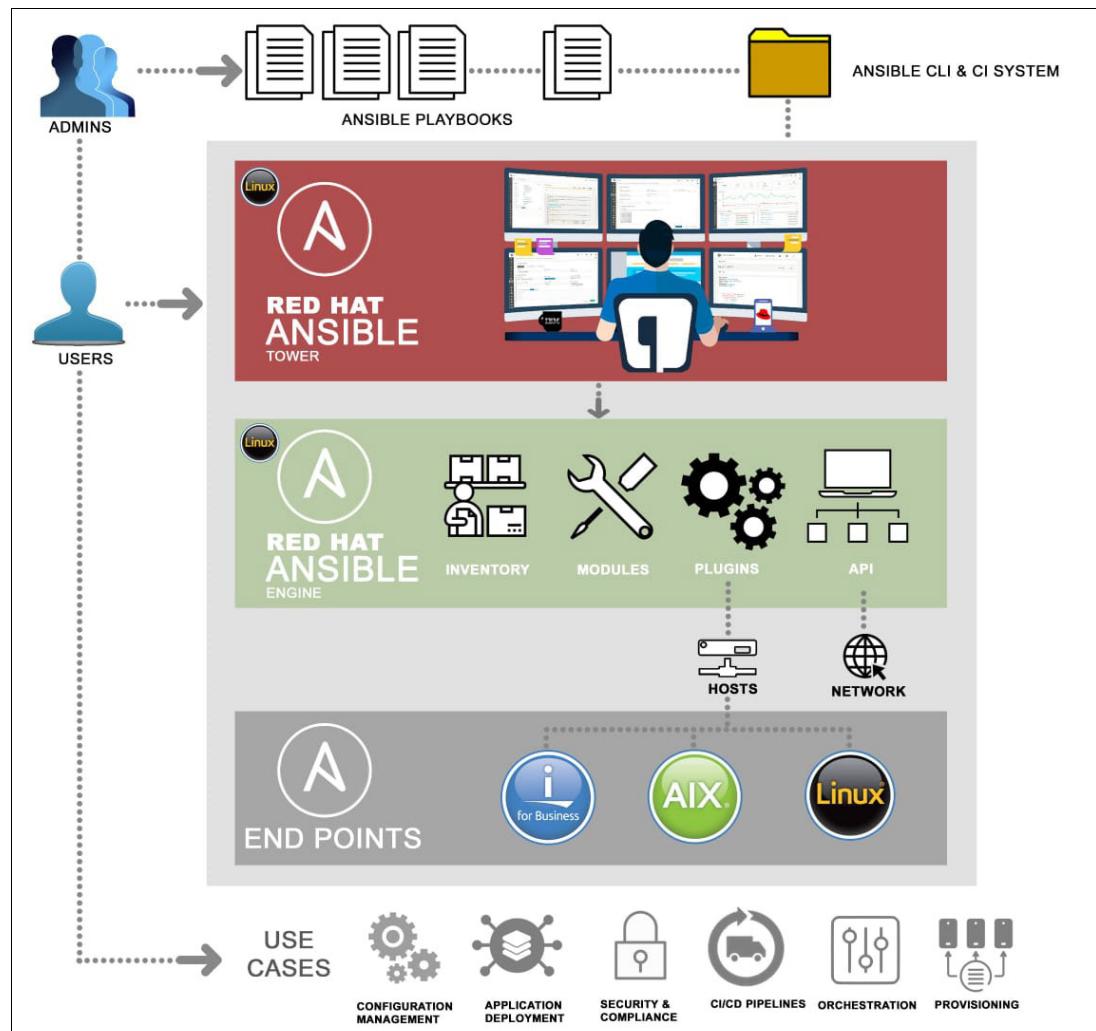


Figure A-2 Red Hat Ansible architecture

Red Hat Ansible automation with IBM Power Systems

Although OS configuration management is the heart and soul of Red Hat Ansible, users demand a rich set of modules to be available for their OSs. So, there are several Red Hat Ansible modules for AIX and IBM i to automate operations, such as patching (Service Packs and Program Temporary Fixes (PTFs)), user and group management, boot management, running commands and SQL queries, and managing object authority. From a support standpoint, AIX and IBM i have commercial Red Hat subscription options that are available, and Linux on IBM Power Systems is common.

Note: For more information about the Supported Red Hat Ansible Collection for IBM Power Systems from Automation Hub (a Red Hat Ansible subscription is required), see [Red Hat Ansible](#).

IBM Cloud Automation Manager and Red Hat Ansible

When containers and microservices were developed, production environments had a major challenge to track them. Orchestration and the management of the containers were needed. As a result, Kubernetes, Helm charts, and Tiller were created to ease and automate the deployment, orchestration, management of containers and microservices.

Now, consider a much bigger picture with a similar problem. As more on-premises applications are migrated and modernized to run in hybrid cloud environments, you might need to horizontally expand cloud environments and the application services stacks in them.

IBM Cloud Automation Manager for cloud infrastructure and services is what Kubernetes is for containers and microservices. Together, IBM Cloud Automation Manager and IBM Cloud Private provide a complete, developer-friendly, and enterprise-grade cloud-management platform that supports multiple clouds, workload architectures, and application service orchestration. This combination gives IT managers unprecedented choice and flexibility to meet the needs of the business.

Within this context, IBM Cloud Automation Manager supports pluggable formats and frameworks, such as Red Hat Ansible, Terraform, HELM for infrastructure as code, and Chef, and Puppet as automation.

Note: To use Red Hat Ansible Playbooks with IBM Cloud Automation Manager, see [IBM Developer](#).

Introducing Red Hat Ansible for IBM i

This section introduces and describes Red Hat Ansible for IBM i OS.

Overview

IBM i is an OS that has thousands of core workloads running for different industries worldwide. Red Hat Ansible for IBM i can fit most of the on-premises tasks and cloud automation requirements for IBM i customers.

There are several cases where IBM i customers or independent software vendors (ISVs) can use Red Hat Ansible:

1. Automate traditional IBM i administration tasks, such as PTF management, system and application configuration, and application deployment and installation. These tasks are common and repeatedly run in a user environment. Automating such tasks and processes can improve system management efficiency.
2. Improve application development process and efficiency to shorten the software delivery cycle. CI/CD delivery is mentioned more by IBM i customers, who need tools to bring Report Program Generator (RPG)-written PGMs and object-based applications into the CI/CD world.
3. Manage multiple IBM i systems with interactive command-line interfaces (CLIs). A single place to manage all the Linux (or other platforms) and IBM i partitions together is important for Managed Services Providers (MSPs) and cloud users. There are many cases where IT environments and solutions are built on many different systems, so the central management of these systems is important. Red Hat Ansible supports different platforms, and the workflow can be managed by playbooks that are written in YAML. By using playbooks, a complex IT environment can be managed from a single place.
4. Automate IBM i tasks with reusable playbooks. There are several cases where existing playbooks can be reused for IBM i customers. For example, for the tasks dealing with open-source software in Portable Application Solution Environment (PASE), playbooks that are written for the AIX platform may be reused with few modifications in some situations.

Figure A-3 shows the Red Hat Ansible for IBM i components.

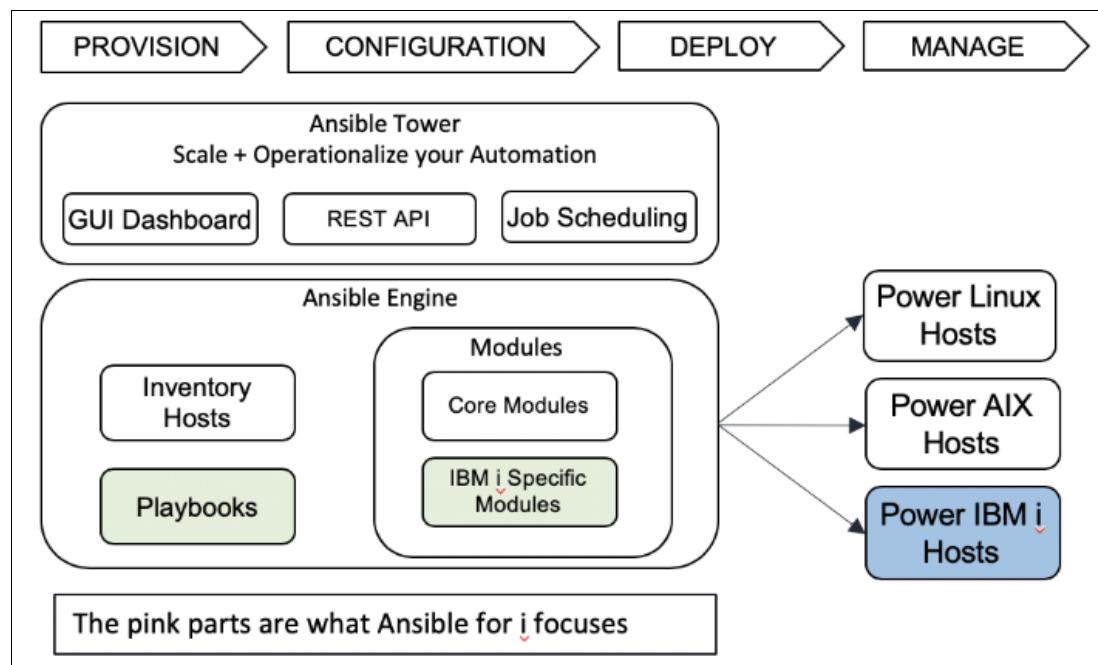


Figure A-3 Red Hat Ansible for IBM i components

Red Hat Ansible is operated by a certain group, and the operator with no IBM i skills wants to do basic management of IBM i by using Red Hat Ansible. Red Hat Ansible developers also can use IBM i modules to complete their playbook writing even if they do not have enough IBM i skills.

Red Hat Ansible Collections for IBM i

Collections are the standard way to extend and complement base Red Hat Ansible content, and now IBM i content is available in both commercial and community form through Red Hat Ansible Certified Content for IBM Power Systems and Ansible Galaxy.

IBM i support in Red Hat Ansible Certified Content

The IBM i collection is a part of Red Hat Ansible Certified Content for IBM Power Systems (requires a subscription), which helps manage IBM i workloads on the IBM Power Systems infrastructure as part of wider enterprise automation strategy through the Red Hat Ansible Automation Platform infrastructure. The IBM i collection is delivered as a fully supported enterprise-grade solution that provides easy to use modules that can accelerate the automation of OS configuration management. Customers can also use more community provided, open-source Red Hat Ansible modules (no enterprise support is available) to automate hybrid cloud operations on IBM Power Systems. With Red Hat Ansible, customers can manage IBM Power Systems easily and consistently.

Note: For more information about Red Hat Ansible Certified Content that is distributed by the Red Hat Ansible Automation Hub, see [Red Hat Certified Technology](#).

Support by using Red Hat Ansible Certified IBM i modules

For IBM i customers to get support when using Red Hat Certified IBM i modules from Automation Hub, they must contact Red Hat first. If IBM assistance is needed, the customer is directed to open a case with IBM for the product area “Ansible on IBM i”.

Figure A-4 shows the Ansible on IBM i option when requesting support.

The screenshot shows a web-based support ticket submission interface. At the top, there is a dropdown menu labeled "Preferred Method of Contact" with the option "Support community Case updates" selected. Below this is a section for "Product Area". A search bar is labeled "Search product area". A dropdown menu lists several product areas: "WM: System Reply List", "WM: Jobs", "DBWQ: DB WebQuery", "DBWQ: DB2 Web Query", "ARCAD", and "Ansible on IBM i". The "Ansible on IBM i" option is highlighted with a red rectangular box. At the bottom of the form, there is a text area with the placeholder "32,000 characters remaining".

Figure A-4 Selecting Ansible on IBM i for support

After the case is open, the case number is provided to Red Hat.

Tip: To open a case for Ansible on IBM i, see [IBM Support](#).

IBM i support at Ansible Galaxy

Ansible Galaxy is an online website that third parties can use to provide Ansible modules, plug-ins, and roles to achieve specific automations, workflows, and tasks.

Note: For more information about Ansible Galaxy, see the [Ansible Galaxy website](#).

The Ansible Content for IBM Power Systems - IBM i (IBM i Ansible collections) website provides modules, action plug-ins, roles, and sample playbooks to automate tasks on IBM i, such as command execution, system and application configuration, work management, fix management, and application deployment.

Some key information can be found at the Ansible Galaxy website:

- ▶ Collection installation information: Provides the command to install IBM i collections on to the Red Hat Ansible engine server.
- ▶ Docs Site: Goes to the documentation website of IBM i Collections. The document introduces the steps to enable IBM i as Red Hat Ansible endpoints, such as installing the required software and PTFs. Also, module usage explanations are provided.
- ▶ Repo: IBM i Ansible collections are open-sourced so that all the code and playbook scripts can be downloaded from public GitHub repository. For each release of the collections, a corresponding branch is created in the repository. In addition, different development branches are also created to integrate fixes and contain unreleased functions.

Normally, only released versions of IBM i collections can be downloaded through Ansible Galaxy. However, the collections under development can be accessed and downloaded directly from the development branch of the GitHub repository.

IBM i specific Red Hat Ansible modules

As of Red Hat Ansible V2.9, it has the modules that are shown in Table A-1 that are available for use specifically for IBM i. These modules are not the complete subset of modules that are available in Red Hat Ansible, but only a small portion that is IBM i specific.

Table A-1 IBM i specific Red Hat Ansible modules

Module	Minimum Red Hat Ansible version	Description
ibmi_at	2.9	Schedule a batch job on a remote IBM i node.
ibmi_cl_command	2.9	Runs a CL command.
ibmi_copy	2.9	Copies a save file from local to a remote IBM i node.
ibmi_display_subsystem	2.9	Displays all active subsystems or active jobs in a subsystem.
ibmi_end_subsystem	2.9	Ends a subsystem.
ibmi_start_subsystem	2.9	Starts a subsystem.
ibmi_lib_restore	2.9	Restores one library on a remote IBM i node.
ibmi_lib_save	2.9	Saves one library on a remote IBM i node.
ibmi_reboot	2.9	Restarts the IBM i machine.
ibmi_sql_execute	2.9	Runs an SQL non-DQL (Data Query Language) statement.

Module	Minimum Red Hat Ansible version	Description
ibmi_sql_query	2.9	Runs an SQL DQL (Data Query Language) statement.
ibmi_fix	2.9	Loads from save file, and applies, removes, or queries PTFs.
ibmi_fix_imgclg	2.9	Installs fixes from a virtual image.
ibmi_object_find	2.9	Finds a specific IBM i object.
ibmi_submit_job	2.9	Submits an IBM i job.
ibmi_iasp	2.9	Controls an independent auxiliary storage pool (IASP) on a target IBM i node.
ibmi_tcp_interface	2.9	Manages the IBM i TCP interface. You can add, remove, start, end, or query a TCP interface.
ibmi_tcp_server_service	2.9	Manages a TCP server on a remote IBM i node.

Note: Additionally, third-party custom modules that are created by the open-source community for IBM i are available at [GitHub](#).

Installing Red Hat Ansible for IBM i

This section describes the prerequisites before you install the IBM i collection for Red Hat Ansible in online mode or from a source. Then, this section describes how to enable the IBM i nodes of Red Hat Ansible.

IBM i collection for Red Hat Ansible server

There are two options to install IBM i collection for Red Hat Ansible:

- ▶ Installing online directly from Ansible Galaxy.
- ▶ Installing from a source.

Before using the Red Hat Ansible collection, you must install some prerequisite software and make it available in your Red Hat Ansible server:

- ▶ Python V3.6+

Python can be installed from various sources, including the package manager for your OS (such as `apt` or `yum`). If you install Python from the package manager for your OS, you must also install the development libraries (usually a package that is named `python3-devel`) because they are required when installing modules through `pip`. To download the latest version of Python, see [Python](#).

- ▶ Red Hat Ansible V2.9+

Red Hat Ansible can be installed from various sources, including the package manager for your OS (such as `apt` or `yum`). You can also install it by using `pip`, which is the package manager for Python, by running the following command:

```
pip install ansible
```

Installing online

The ansible-galaxy tool is the package manager tool for Ansible. The collection is published at [Ansible Galaxy](#). To install the IBM i collection by using ansible-galaxy, run the following command:

```
ansible-galaxy collection install ibm.power_ibmi
```

Installing from a source

Customers might want to install the IBM i collection from another source if they cannot access Ansible Galaxy or if they must install a version of the collection that is unpublished. To install from a source, complete the following steps:

1. Clone the repository by running the following command:

```
git clone https://github.com/IBM/ansible-for-i.git
```

Note: For more information about cloning a GitHub repository, see [Cloning a repository](#).

2. Build the collection artifact by running the following commands:

- `cd ansible-for-i`
- `ansible-galaxy collection build`

3. Install the collection by running the following command, but replace x.y.z with the current version:

```
ansible-galaxy collection install ibm-power_ibmi-x.y.z.tar.gz
```

Enabling IBM i nodes

To enable the managed nodes of Red Hat Ansible from IBM i, you must meet the following requirements:

- ▶ IBM Portable Utilities for i (5733-SC1 option base)
- ▶ OpenSSH, OpenSSL, and zlib functions (5733-SC1 option 1)
- ▶ IBM HTTP Server for i (5770-DG1 option base)
- ▶ Python 3
- ▶ Python packages
 - `python3-itoolkit`
 - `python3-ibm_db`

Note: Because 5733-SC1 options base and 1, and 5770-DG1 do not require a software license key for IBM i, you can download them by completing the steps that are described in [How to download 5733-SC1 from ESS Website](#).

For open source Python and its packages, there are several ways to install them on IBM i:

- ▶ Installing rpm packages manually, as described at [Bitbucket](#).
- ▶ Installing rpm packages automatically onto an IBM i system that can access the internet.
- ▶ Installing rpm packages automatically onto IBM i systems that are offline.

For more information, see [Installing IBM i collection to Ansible server](#).

Running your first Red Hat Ansible example for IBM i

With the supported IBM i modules and Red Hat Ansible core modules, common IBM i tasks can all be done by Red Hat Ansible.

After installing the Red Hat Ansible for IBM i collection, a few examples can be found in the following directory:

```
~/.ansible/collections/ansible_collections/ibm/power_ibmi/playbooks
```

To get started, prepare an inventory file. Figure A-5 shows sample file content for the IBM i inventory.

```
[ibmi]
your_ibmi_ip ansible_ssh_user=your_user ansible_ssh_pass=your_host_password

[ibmi:vars]
ansible_python_interpreter="/QOpenSys/pkgs/bin/python3"
ansible_ssh_common_args=' -o StrictHostKeyChecking=no'
```

Figure A-5 IBM i inventory file

In Figure A-5, a group that is named ibmi was created, and there is only one system under that group. Also, the variables against group ibmi are defined in the file. One important variable is *ansible_python_interpreter*, which tells Red Hat Ansible where to find Python on the endpoint IBM i system in group ibmi. If you use Red Hat Ansible Tower, you must do all inventory configurations through the GUI.

Write a simple Red Hat Ansible playbook, which is used to run a list of tasks that are run by modules. IBM i modules can be used together with other Red Hat Ansible modules to complete complex tasks. Figure A-6 shows the content of a simple playbook that demonstrates module usage for IBM i systems. It can be found in the following directory:

```
~/.ansible/collections/ansible_collections/ibm/power_ibmi/playbooks/ibmi-cl-command-sample.yml
```

```
- hosts: all
  gather_facts: no
  collections:
    - ibm.power_ibmi

  tasks:
    - name: run the CL command to create a library
      ibmi_cl_command:
        cmd: crtlib lib(ansiblei)
      register: crt_lib_result
```

Figure A-6 Playbook of module usage in IBM i

In the playbook, the inventory ibmi is used, the collection ibm.power_ibmi is included, and module ibmi_cl_command is run to create a library named ansiblei.

After you write your playbook, run the following **ansible-playbook** command to run the playbook:

```
ansible-playbook -i /yourpath/hosts_ibmi.ini ibmi-cl-command-sample.yml
```

Running from the CLI

You can run your Red Hat Ansible tasks from the CLI. When you run the Red Hat Ansible command, you need to specify which endpoints you want to select to run the task, for example:

```
ansible ibmi -m ibm.power_ibmi.ibmi_cl_command -a "cmd='crtlib lib(C1)' joblog=true"
```

In the command, `ibmi` is the group of endpoint IBM i systems. You must define which module to use to complete the task. The module in this example is the `ibmi_cl_command` module in the `power_ibmi` collections in the `ibm` namespace. This full name must be specified after the `-m` option. The parameter values to this module must be provided, and in this example, the `create library` CL command is used as the input of parameter `cmd`. By default, Ansible goes to the default inventory file location to look up what endpoint systems are defined for group `ibmi`. The content of the inventory file is shown in Figure A-7.

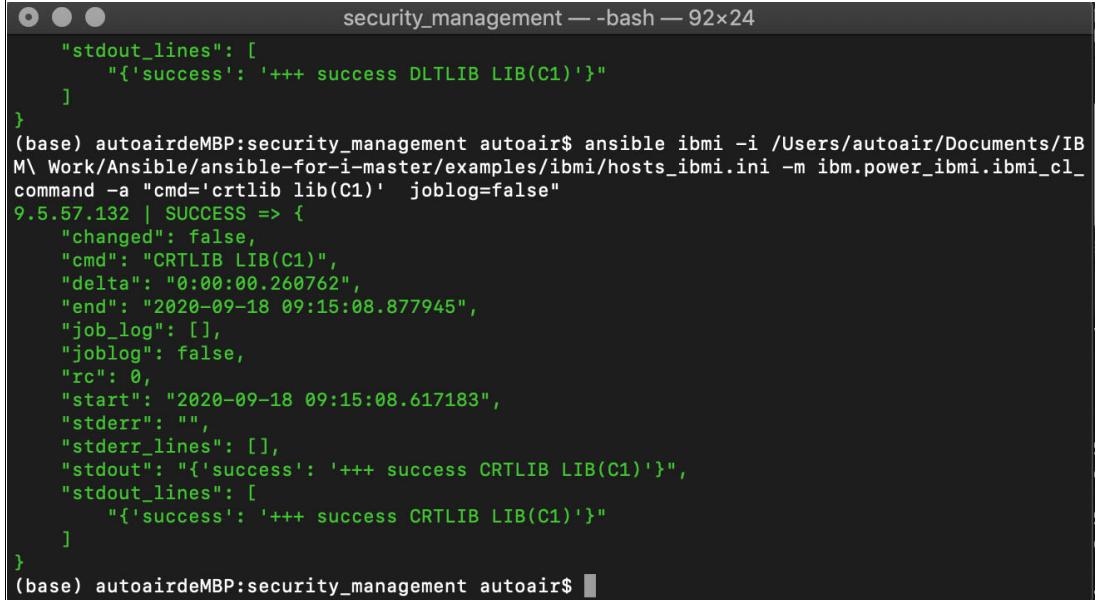
```
[ibmi]
9.5.xxx.xxx ansible_ssh_user=you_ssh_user ansible_ssh_pass=your_ssh_pwd

[ibmi:vars]
ansible_python_interpreter="/QOpenSys/pkgs/bin/python3"
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
```

Figure A-7 Sample inventory file

As shown in Figure A-7, the inventory file contains the systems list for a group, and also includes several variables that are used during IBM i task run times. The `ansible_python_interpreter` variable is important because the default installation location of Python on IBM i is not recognized by Red Hat Ansible. The variable `ansible_python_interpreter` tells Red Hat Ansible where to find Python on IBM i.

Figure A-8 shows an example of running Red Hat Ansible from the CLI.

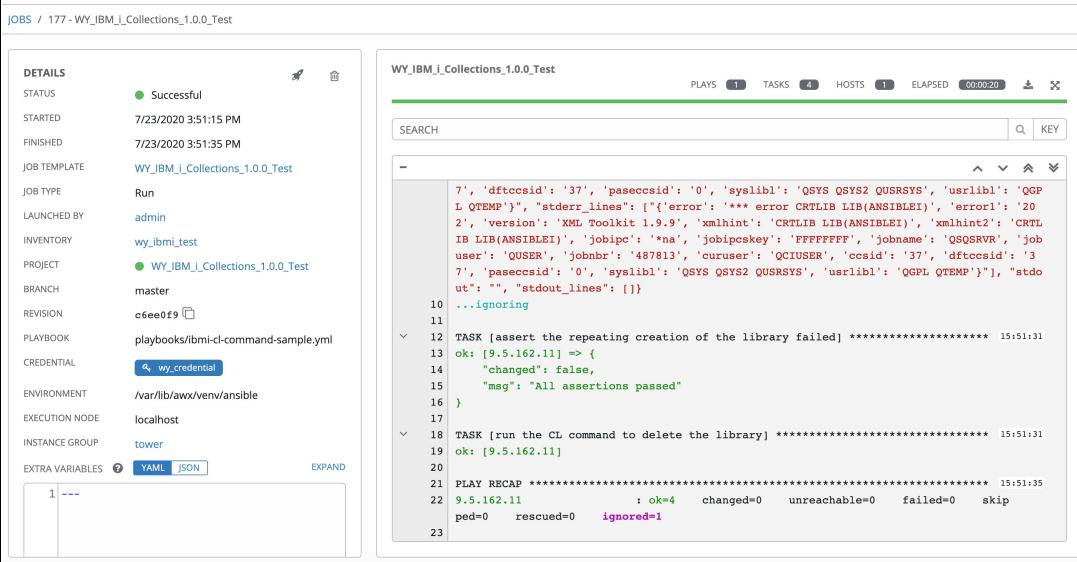


```
security_management — bash — 92x24
"stdout_lines": [
    "'success': '+++ success DTLIB LIB(C1)'"
]
}
(base) autoairdeMBP:security_management autoair$ ansible ibmi -i /Users/autoair/Documents/IB
M\ Work/Ansible/ansible-for-i-master/examples(ibmi)/hosts_ibmi.ini -m ibm.power_ibmi.ibmi_cl_
command -a "cmd='crtlib lib(C1)' joblog=false"
9.5.57.132 | SUCCESS => {
    "changed": false,
    "cmd": "CRTLIB LIB(C1)",
    "delta": "0:00:00.260762",
    "end": "2020-09-18 09:15:08.877945",
    "job_log": [],
    "joblog": false,
    "rc": 0,
    "start": "2020-09-18 09:15:08.617183",
    "stderr": "",
    "stderr_lines": [],
    "stdout": "'success': '+++ success CRTLIB LIB(C1)'",
    "stdout_lines": [
        "'success': '+++ success CRTLIB LIB(C1)'"
    ]
}
(base) autoairdeMBP:security_management autoair$
```

Figure A-8 Running Red Hat Ansible from the CLI

Running from Red Hat Ansible Tower

Red Hat Ansible Tower provides a GUI interface and REST application programming interfaces (APIs) to drive and manage Ansible tasks. It provides complete management of inventory, orchestration of the tasks and securities, and other functions. Also, a dashboard is provided so that you have an overview of the Ansible jobs and tasks, as shown in Figure A-9. You can manage your IBM i tasks and use most of the Red Hat Ansible Tower features.



The screenshot shows the Red Hat Ansible Tower interface. On the left, a sidebar displays job details: Status (Successful), Started (7/23/2020 3:51:15 PM), Finished (7/23/2020 3:51:35 PM), Job Template (WY_IBM_i_Collections_1.0.0_Test), Job Type (Run), Launched By (admin), Inventory (wy_ibmi_test), Project (WY_IBM_i_Collections_1.0.0_Test), Branch (master), Revision (c6eed09), Playbook (playbooks/ibmi-cl-command-sample.yml), Credential (wy_credential), Environment (/var/lib/awx/venv/ansible), Execution Node (localhost), Instance Group (tower), and Extra Variables (YAML, JSON). On the right, the main pane shows the log output for the job WY_IBM_i_Collections_1.0.0_Test. The log includes several tasks, notably task 12 which asserts the creation of a library failed, and task 18 which runs a CL command to delete the library. Task 21 is a play recap showing 4 successful tasks, 0 changes, 0 unreachable hosts, 0 failed tasks, and 1 skipped task.

```
7', 'dftccsid': '37', 'paseccsid': '0', 'syslibl': 'QSYS QSYS2 QUSRSYS', 'usrlibl': 'QGP L QTEMP}', "stderr_lines": [{"error": "*** error CRTL LIB(ANSIBLEI)", "error1": "20 2", "version": "XML Toolkit 1.9.9", "xmlihint": "CRTL LIB(ANSIBLEI)", "xmlihint2": "CRTL LIB(ANSIBLEI)", "jobipc": "na", "jobipcskey": "FFFFFFFFFF", "jobname": "QSQSRVR", "job user": "QUSER", "jobnbr": "487813", "curuser": "QCIUSER", "ccsid": '37', 'dftccsid': '3 7', 'paseccsid': '0', 'syslibl': 'QSYS QSYS2 QUSRSYS', 'usrlibl': 'QGPL QTEMP')", "stdo ut": "", "stdout_lines": []}]  
10 ...ignoring  
11  
12 TASK [assert the repeating creation of the library failed] **** 15:51:31  
13 ok: [9.5.162.11] => {  
14     "changed": false,  
15     "msg": "All assertions passed"  
16 }  
17  
18 TASK [run the CL command to delete the library] **** 15:51:31  
19 ok: [9.5.162.11]  
20  
21 PLAY RECAP **** 15:51:35  
22 9.5.162.11 : ok=4    changed=0    unreachable=0    failed=0    skip ped=0    rescued=0    ignored=1
```

Figure A-9 Running Red Hat Ansible from Red Hat Ansible Tower

Common use cases and integrated examples

This section describes common used cases and integrated examples.

Fix management

Fix management is a common use case for IBM i Ansible automations. Installing PTF groups and single PTFs is repetitive work that sometimes produces common errors. There are several ways that you can benefit from automation, for example, sending fixes from your local repository to multiple IBM i systems, grasping and installing dependent PTFs together with the PTF that you want to install on the target system, or periodically checking the version of groups to see whether they are far behind. Some of the users might want a more automated way to filter PTFs. These cases and areas might improve by building Ansible playbooks and solutions on top of the modules that support IBM i.

Security control management

Security compliance checking is important for both on-premises and cloud users. The common use case for Ansible is that certain security rules should be applied to the systems, and timely checking of compliance with system values, user profiles, object authorities, and networking, are critical. If there is non-compliance, you might need to change the values of the security configuration. You could use IBM i modules and playbooks to drive such security tasks.

CI/CD

CI/CD is the backbone of the modern DevOps environment. To achieve CI/CD, automating environment preparation, code delivery, testing, and deployment is key.

Ansible fits well into the CI/CD process. It can provision VMs and applications for environment preparation, choose the correct host in the inventory for specific tasks, and automatically trigger build, testing, and cleaning up work. It interacts with version control solutions such as GitHub to check and deliver the final code after a series of software lifecycle steps are done. CI/CD can apply to IBM i application lifecycle management too.

Sample labs: Deploying IBM virtual machines into a hybrid cloud

This section describes how to deploy an IBM i virtual machine (VM) into a hybrid cloud. You deploy an IBM i VM into IBM PowerVC, and an IBM Power Systems Virtual Server by using IBM Cloud Pak for Multicloud Management.

Deploying an IBM virtual machine by using IBM Cloud Pak for Multicloud Management in IBM PowerVC

This section demonstrates how to create and deploy a template by using Terraform, and how to delete an IBM i VM by using IBM Cloud Pak for Multicloud Management in IBM PowerVC.

Overview

IBM Cloud Pak for Multicloud Management runs on Red Hat OpenShift, which provides consistent visibility governance, and automation from on-premises to the edge. With it, you get more application and cluster visibility across the enterprise to public or private cloud. IBM Cloud Pak for Multicloud Management includes IBM Cloud Automation Manager that uses HashiCorp Terraform as its underlying engine. IBM Cloud Automation Manager enables connectivity to numerous cloud infrastructures, including IBM PowerVC (OpenStack), IBM Cloud, AWS, Azure, Google, and several others. IBM Cloud Automation Manager can provision VMs (including logical partitions (LPARs) by using IBM PowerVC) and containers, so users can create software catalog entries that build complex multilayer applications with a single click.

Because IBM Cloud Automation Manager is delivered as part of an IBM Cloud Pak, it runs on Red Hat OpenShift, which creates a centralized management plane from which you can deploy all your applications regardless of what platform or technologies they use.

In this present section, we describe how to deploy an IBM i VM in IBM PowerVC by using IBM Cloud Pak for Multicloud Management, which that uses Terraform templates.

Configuring a cloud connection

To enable IBM Cloud Pak for Multicloud Management communication with IBM PowerVC, you must create a cloud connection.

Table A-2 shows the parameters that are required for the configuration of a cloud connection in IBM Cloud Pak for Multicloud Management against IBM PowerVC.

Table A-2 Parameters prompted for the OpenStack cloud connection

Parameters	Description
Authentication URL	Enter the authentication URL. It is used for identification. The syntax is <code>http://IP_ADDRESS/identity/v3</code> . Here, v3 refers to Version 3 of the keystone identity API.
User name	Enter the username that is used to log in to the OpenStack server.
Password	Enter the password of the OpenStack user.
Domain name	Enter the keystone domain name.
Region	Enter the region of the cloud OpenStack.
Project name	Enter the name of the project that is supported by OpenStack and within the specified domain.

Note: The values for the URL, domain, and region can be discovered on any OpenStack implementation by examining the output of the OpenStack catalog list. For more information about these values, see [OpenStack Provider](#).

For more information about the configuration and setup of a cloud connection of IBM PowerVC to IBM Cloud Pak Multicloud Management, see *IBM AIX Enhancements and Modernization*, SG24-8453.

Note: IBM PowerVC Standard Edition V1.4.4.1 or earlier can perform any API integration with IBM Cloud Pak for Multicloud Management, AIX, or IBM i. IBM Cloud PowerVC Manager Edition can be used too.

Terraform

By default, IBM Cloud Automation Manager that is embedded in IBM Cloud Pak for Multicloud Management manages Terraform only. Terraform is the module in IBM Cloud Automation Manager that communicates with and provisions the infrastructure.

Terraform also enables DevOps capabilities such as “infrastructure as code” by using HashiCorp Terraform, that is, Terraform can provision IBM Power Systems resources through IBM PowerVC by using an OpenStack provider. IBM PowerVC provides the foundational technology on top of which the rest of the IBM Power Systems cloud stack builds.

For the templates that we created in our example and use on deployments such as on-premises and off-premises, we use Terraform V0.12 because it is a major release, and the syntax changed much from the Terraform V0.11 syntax. For more information, see [Upgrading to Terraform V0.12](#).

Tip: You can use the Terraform V.0.12.x `terraform 0.12checklist` command ([Pre-upgrade Checklist](#)) to see whether there are any pre-upgrade steps, and then use the `terraform 0.12upgrade` command ([Upgrading Terraform configuration](#)) to upgrade Terraform.

You can use these commands on your templates to upgrade the syntax, for example, from Version .0.11 to Version .0.12.

Git

Git is not bundled as part of IBM Cloud Pak for Multicloud Management, but a best practice is to store all the templates and content libraries that you create into a Git repository. This best practice is also a requirement for air-gapped environments that cannot reach [GitHub](#).

To integrate IBM Cloud Pak for Multicloud with GitHub, you must configure your GitHub account. If you do not have a GitHub account, you can create one. For more information about creating an account, generating a personal access token, and creating a Git repository, see [Install and use Cloud Automation Manager in IBM Cloud Private](#).

The sample templates that are generated for on-premises and off-premises deployments can be found at this [GitHub repository](#).

Creating a template for IBM i deployment

One of the key components of IBM Cloud Automation Manager is the template. A *template* is commonly used in reference to the Terraform configuration file with the .tf extension. A Terraform configuration is also used interchangeably with a template. In that context, you can use Terraform templates to define VMs or LPARs by using IBM PowerVC provisioning information, such as a compute template, storage template, host group, and network.

Designing templates by using the Template Designer

The Template Designer is an interface for IBM Cloud Pak for Multicloud Management that you can use to design templates. To use it, open the Library window and click **Create Template**, as shown in Figure A-10.

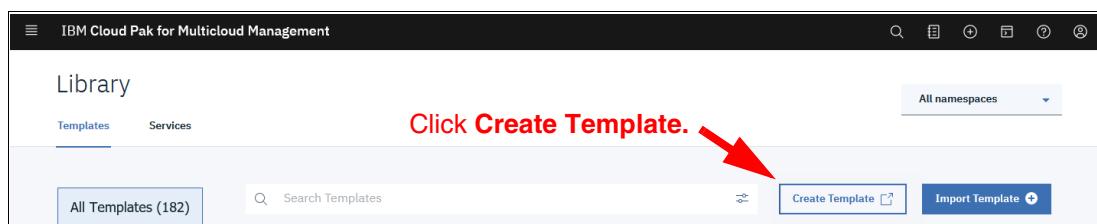


Figure A-10 IBM Cloud Pak for Multicloud Management: Create Template

A window opens that shows the Template Designer, as shown in Figure A-11.

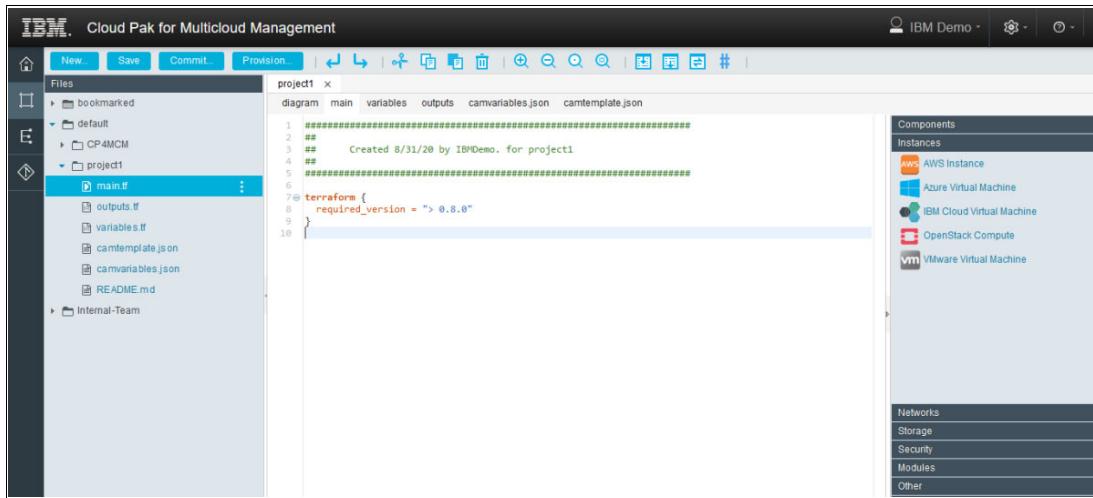


Figure A-11 IBM Cloud Pak for Multicloud Management: Template Designer

Note: For more information about creating your own template to deploy a customized instance to your environment, see [Creating a template](#).

Importing a template

In this section, we import a template source from GitHub. However, you also can import templates from the following sources:

- ▶ GitLab
- ▶ Bitbucket Server
- ▶ From Scratch
- ▶ From a URL (compressed file)
- ▶ From a folder
- ▶ From a file (compressed)

To import a template after logging in to IBM Cloud Pak for Multicloud Management, complete the following steps:

1. Select **Automate infrastructure** → **Terraform automation**, as shown in Figure A-12 on page 133.

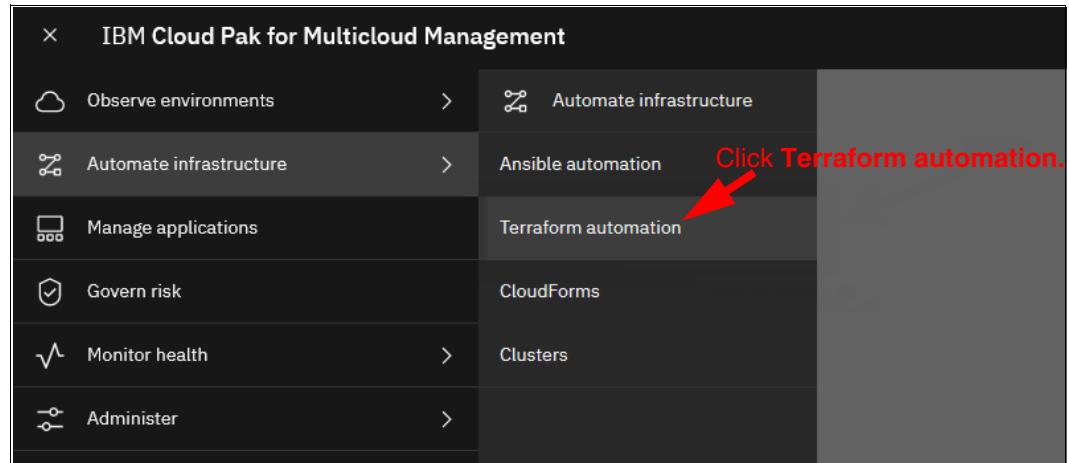


Figure A-12 IBM Cloud Pak for Multicloud Management: Terraform automation

2. The Library window opens. Click **Import Template**, as shown in Figure A-13.

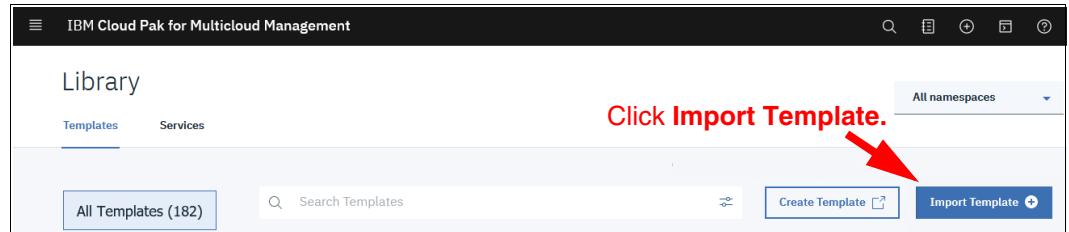


Figure A-13 IBM Cloud Pak for Multicloud Management: Import Template

3. In this example, we import a template by using a GitHub repository. The window that is shown in Figure A-14 opens. In this window, enter the following information:

- Assign Access: **Make Template Globally Accessible (available to all users)**.
- Import Template source: **GitHub**.
- GitHub Repository URL: Enter the URL for your GitHub repository.
- GitHub Access Token: Enter your token.

Click **Import**.

The screenshot shows the 'Import Template' dialog box. It has a title bar 'Import Template'. Below it, a note says '* indicates a required field'. The first step, '1. Select Assign Access.', points to the 'Assign Access' section where 'Make Template Globally Accessible (available to all users)' is selected. The second step, '2. Choose Import Template source.', points to the 'Import template source' dropdown which is set to 'GitHub'. The third step, '3. Paste GitHub Repository URL.', points to the 'GitHub Repository URL' input field containing the URL 'https://github.com/IBMRRedbooks/SG248486-Red-Hat-OpenShift'. The fourth step, '4. Paste GitHub Access Token.', points to the 'GitHub Access Token' input field which contains several dots ('.....'). The fifth step, '5. Click Import.', points to the 'Import' button at the bottom right of the dialog. There are also 'Cancel' and 'Learn how?' buttons.

Figure A-14 IBM Cloud Pak for Multicloud Management: Importing a template from GitHub

4. The Template Metadata window opens, as shown in Figure A-15 on page 135. Enter enter a template name, a short description of the template, and select a cloud provider, which in this case is OpenStack - IBM PowerVC. Click **Save**.

After your changes to the template are saved, go back to Template Library. The template **IBMi_VM** is displayed and ready to be use.

1. Enter Template name.

2. Insert short description.

3. Select Cloud Provider.

4. Click Save.

Figure A-15 IBM Cloud Pak for Multicloud Management: Template Metadata

A basic Terraform template to deploy a virtual machine

To deploy a VM instance resource within OpenStack, run `openstack_compute_instance_v2`. A sample Terraform template is shown in Example A-1, which creates an IBM i VM from an image that is available in IBM PowerVC that is named IBMi73. When the command runs, a small compute template or flavor name is selected, the template attaches to a network VLAN that is named `vlan133_vra`, and you are prompted to enter a name for the new VM.

This process uses variables in Terraform and JSON files so that you can input the information that is needed to deploy a VM into IBM PowerVC that uses IBM Cloud Pak for Multicloud Management.

Note: If you are using a self-signed certificate on your IBM PowerVC server, you must add the `insecure=true` option, as shown in Example A-1, or use the IBM PowerVC certificate when initializing the cloud connection.

To do this task, obtain the certificate file from the IBM PowerVC server and upload it to your Terraform container in a location under a Persistent Volume (PV). Next, specify the path to the certificate when configuring the cloud connection.

Example A-1 A Terraform template to create an IBM PowerVC virtual machine

```
provider "openstack" {
  insecure = true
}

#Create an IBM i partition
resource "openstack_compute_instance_v2" "PowerVC-VM" {
  name      = var.ibm_stack_name
  image_name = var.openstack_image_name
  flavor_name = var.openstack_flavor_name
```

```

network {
    name = var.openstack_network_name
}
}

#Variables for deployment
variable "openstack_image_name" {
    description = "Insert the name of the image of PowerVC."
}

variable "openstack_flavor_name" {
    description = "Insert the Compute Template or flavor to deploy the virtual machine."
}

variable "openstack_network_name" {
    description = "Insert the name of the network."
}

variable "ibm_stack_name" {
    description = "Insert a new name for the Virtual Machine"
}

output "VM_IP_Address" {
    value = openstack_compute_instance_v2.PowerVC-VM.*.network.0.fixed_ip_v4
}

```

Note: In Example A-1 on page 135, the Terraform template is represented as a single file, but this template may be split into separate files, for example, putting the variables into a file that is named variables.tf. For more information about defining the standard structure of the Terraform and Service Automation template for IBM Cloud Pak for Multicloud Management, see [Defining the structure of Terraform and Service Automation template](#).

Example A-2 shows the IBM Cloud Automation Manager variables file, which is a metadata file (camvariables.json) that describes how each variable is presented in the Terraform & Service Automation user interface.

Example A-2 Set of IBM Cloud Automation Manager variables for a specific template.

```
{
    "output_datatype": "content_template_output",
    "input_datatypes": [ ],
    "input_namespaces": [ ],
    "output_namespace": "",
    "input_groups": [ ],
    "output_groups": [ ],
    "template_input_params": [
        {
            "name": "openstack_image_name",
            "label": "Insert the name of the image of PowerVC",
            "description": "Insert the name of the image of PowerVC",
            "type": "string",
            "default": "IBMi73",
            "validation": ""
        }
    ]
}
```

```

    "required": true,
    "secured": false,
    "hidden": false,
    "immutable": false,
    "immutable_after_create": false
},
{
    "name": "openstack_flavor_name",
    "label": "Insert the Compute Template to deploy the virtual machine.",
    "description": "Insert the Compute Template to deploy the virtual machine.",
    "type": "string",
    "default": "small",
    "validation": "",
    "required": true,
    "secured": false,
    "hidden": false,
    "immutable": false,
    "immutable_after_create": false
},
{
    "name": "openstack_network_name",
    "label": "Insert the name of the network.",
    "description": "Insert the name of the network.",
    "type": "string",
    "default": "vlan133_vra",
    "validation": "",
    "required": true,
    "secured": false,
    "hidden": false,
    "immutable": false,
    "immutable_after_create": false
},
{
    "name": "ibm_stack_name",
    "label": "Insert a new name for the Virtual Machine",
    "description": "Insert a new name for the Virtual Machine",
    "type": "string",
    "default": "IBMiVM",
    "validation": "",
    "required": true,
    "secured": false,
    "hidden": false,
    "immutable": false,
    "immutable_after_create": false
}
],
"template_output_params": [ ]
}

```

Note: Example A-1 on page 135 and Example A-2 on page 136 make up the file that is provided by a Git Repository URL from which IBM Cloud Pak for Multicloud Management pulls them. One file is the main entry point for the template (`main.tf`), and the second one is the meta-structure of a IBM Cloud Automation Manager variables file (`camvariables.json`). Both files complement each other. For information about the structure of the second file, see [Defining the structure of a Cloud Automation Manager template](#).

Deploying an IBM i virtual machine by using a template

To deploy an IBM i VM by using a template, complete the following steps:

1. In the IBM Cloud Pak for Multicloud Management application, select **Automate infrastructure** → **Terraform automation**, as shown in Figure A-12 on page 133.

Search for `IBMi_VM`, select the template that you want to use, and click **Deploy**, as shown in Figure A-16.

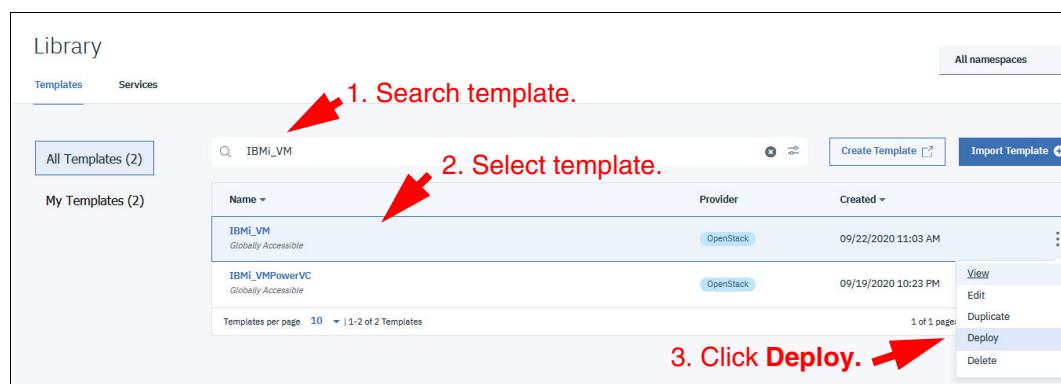


Figure A-16 IBM Cloud Pak for Multicloud Management: Selecting the template and Deploy

In the next window, click **Deploy** again.

2. Enter details of the deployment, such as the namespace that you want to deploy in IBM Cloud Pak for Multicloud Management and the Instance Name for the specific deployment of the template. In this instance, we use the default namespace and use `IBMi_VM` as the instance name. For **Select a Cloud connection**, select the connection that matches the name that you created when adding the IBM PowerVC provider, and select a Terraform version, as shown in Figure A-17 on page 139.

1. Select a Namespace
Namespace **1. Select a Namespace.**
 default

2. Enter Instance Name
Instance Name **2. Insert Instance name.**
 IBMi_VM

3. Select a Cloud Connection
OpenStack Cloud Connection **3. Select a Cloud connection.**
 Nirvana-PowerCloud

4. Select a Terraform Version
Terraform Version **4. Select a Terraform version.**
 0.12.21

5. Additional Options
Please insert the name of the image of PowerVC **5. Complete all additional options.**
 IBMi73

Please insert the Compute Template to deploy the virtual machine. **small**

Please insert the name of the network. **vlan133_vra**

Please insert a new name for the Virtual Machine **IBMiVM**

6. Click Deploy. **Deploy**

Figure A-17 IBM Cloud Pak for Multicloud Management: Enter details of the deployment

There are more options that are required by the template and must be entered, as shown in Table A-3.

Table A-3 Extra parameters that are required by template

Field	Description
image_name	The image name from IBM PowerVC for the image that you want to deploy.
flavor_name	Name of the Compute Template to deploy.
network_name	The name of the network in IBM PowerVC that you want to deploy.
ibm_stack_name	The name of the new LPAR to deploy.

IBM PowerVC provides Compute Templates by default. In this case, we select a small one for deployment. For more information about the Compute Templates that are provided with IBM PowerVC by default, see [Compute Templates](#).

After you enter all the options, click **Deploy**, as shown in Figure A-17.

Note: Instead of `image_name` and `flavor_name`, you can use `image_id` and `flavor_id`. The first parameter is a user-defined image UUID in IBM PowerVC OpenStack to which you want to deploy, and the second parameter is a unique ID (integer or UUID) that is an available hardware configuration in IBM PowerVC to which you want to deploy. The template should be adjusted at the main entry point.

For more information about `image_id`, see [Image Service API V2](#).

For more information about `flavor_id`, see [Flavors](#).

Important: For this deployment, we use Terraform V0.12.21, and Terraform Provider OpenStack V1.17.0.

3. The **Deployed Instances** window opens, where you can check the status and relevant information about your deployed instances, as shown in Figure A-18.

The screenshot shows the 'Deployed Instances' window in the IBM Cloud Pak for Multicloud Management. At the top, there's a header with a back arrow, the title 'Deployed Instances', and three tabs: 'Overview' (selected), 'Modify', and 'Log File'. Below the tabs, there's a section for a 'Template File' named 'IBMi_VM' with a 'View Git URL' button. To the right of this is an 'Instance Details' panel and an 'Activity' panel. The 'Activity' panel shows a green checkmark and the message 'Deployment Successful 09/24/2020 1:41 PM' with a 'View Log' link. The 'Instance Details' panel lists various parameters: NAMESPACE (default), TEMPLATE (Terraform), TEMPLATE VERSION (master), CLOUD (OpenStack), OPENSTACK CONNECTION (Nirvana-PowerCloud), CREATION TIME (09/24/2020 1:41 PM), and OUTPUT DATA TYPE (content_template_output). Below these panels is a 'Resource Details' table with one row for 'openstack_compute_instance_v2.Po...'. The table columns are Name, Console (with a link to https://129.40.156.2), IP Address (129.40.186.43), Created (09/24/2020 1:41PM), and State (Running). At the bottom of the window is an 'Input Variables' table with four rows, each with a 'Name' and 'Value' column. The variables are: 'Please insert the name of the image of PowerVC' (IBMi73), 'Please insert the Compute Template to deploy the virt...' (small), 'Please insert the name of the network.' (vlan133_vra), and 'Please insert a new name for the Virtual Machine' (IBMIVM).

Figure A-18 IBM Cloud Pak for Multicloud Management: State details after deployment

Example A-3 shows a sample of the log file from the deployed instance. Here you can view the latest log output and get detailed information from the Plan and Apply action.

Example A-3 Sample log file of a successful deployment into IBM PowerVC

```
Thu Sep 24 2020 16:41:16 GMT+0000 (UTC) Running terraform init ...
```

```
Initializing provider plugins...
Terraform has been successfully initialized!
```

```
You may now begin working with Terraform. Try running "terraform plan" to see
```

any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

Thu Sep 24 2020 16:41:17 GMT+0000 (UTC) Running terraform apply ...

PowerVC-VM

```
openstack_compute_instance_v2.PowerVC-VM: Creating...
  access_ip_v4:      "" => ""
  access_ip_v6:      "" => ""
  all_metadata.%:    "" => ""
  availability_zone: "" => ""
  flavor_id:         "" => ""
  flavor_name:       "" => "small"
  force_delete:      "" => "false"
  image_id:          "" => ""
  image_name:        "" => "IBMi73"
  name:              "" => "IBMiVM"
  network.#:         "" => "1"
  network.0.access_network: "" => "false"
  network.0.fixed_ip_v4: "" => ""
  network.0.fixed_ip_v6: "" => ""
  network.0.floating_ip: "" => ""
  network.0.mac:      "" => ""
  network.0.name:     "" => "vlan133_vra"
  network.0.port:     "" => ""
  network.0.uuid:     "" => ""
  power_state:        "" => "active"
  region:             "" => ""
  security_groups.#: "" => ""
  stop_before_destroy: "" => "false"
```

openstack_compute_instance_v2.PowerVC-VM: Still creating... (10s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (20s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (30s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (40s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (50s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (1m0s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (1m10s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (1m20s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (1m30s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (1m40s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (1m50s elapsed)

openstack_compute_instance_v2.PowerVC-VM: Still creating... (2m0s elapsed)

```
openstack_compute_instance_v2.PowerVC-VM: Still creating... (2m10s elapsed)
openstack_compute_instance_v2.PowerVC-VM: Still creating... (2m20s elapsed)
openstack_compute_instance_v2.PowerVC-VM: Still creating... (2m30s elapsed)
openstack_compute_instance_v2.PowerVC-VM: Still creating... (2m40s elapsed)
openstack_compute_instance_v2.PowerVC-VM: Creation complete after 2m46s (ID: 2fbc8763-5d13-4202-b96a-d6d94d68fc1d)
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

```
VM_IP_Address = [
  129.40.186.43
]
```

Note: Based on your configuration, Terraform creates a plan and describes the actions that must be done to get to the required state. You can review the plan, change it, or simply do the plan. When you change your configuration, Terraform can determine what changed, and create incremental plans that you can apply to your IBM Cloud resources. For more information, see [IBM Cloud](#)

4. Using IBM Navigator for i, which is an integrated web-based interface that consolidates IBM i system access and management tasks in one place, we establish a connection to the IP address of the VM that was deployed, as shown in Figure A-19 on page 143.

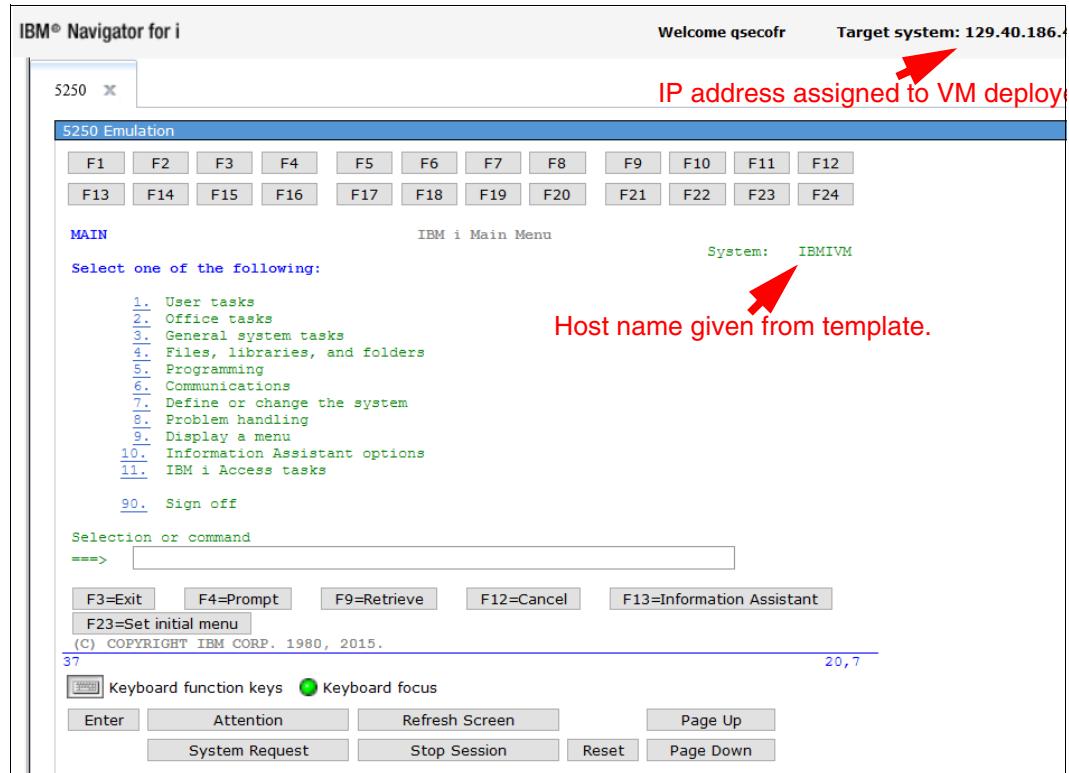


Figure A-19 Accessing IBM Navigator for i after the deployment from IBM Cloud Pak for Multicloud Management

Tip: To access the IBM Navigator for i, point your web browser at the following URL:

`http://{IBM i system name}:2001`

For more information, see [IBM Navigator for i](#).

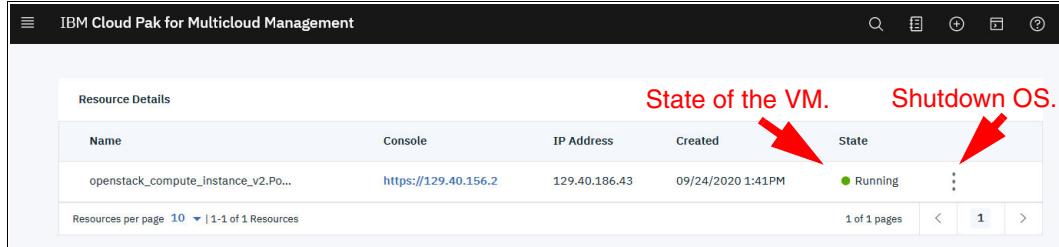
Deleting a virtual machine by using IBM Cloud Pak for Multicloud Management

Before you delete a VM by using IBM Cloud Pak for Multicloud Management, shut down the OS, and then delete the VM.

Complete the following steps:

1. Look at the state of the VM (in this case, the VM name is IBMi_VM) by opening IBM Cloud Pak for Multicloud Management, selecting **Deployed Instances** → **Templates**, searching the instances, and clicking the instance IBMi_VM.

The Resources Details window opens, as shown in Figure A-20. You can see the state of the VM, which is Running.



A screenshot of the IBM Cloud Pak for Multicloud Management interface. The title bar says "IBM Cloud Pak for Multicloud Management". Below it is a table titled "Resource Details". The table has columns: Name, Console, IP Address, Created, and State. The "State" column shows "Running" with a green dot icon. Red arrows point from the text "State of the VM." and "Shutdown OS." to the "State" column and the three-dot menu next to it respectively. At the bottom of the table, there's a note: "Resources per page 10 | 1-1 of 1 Resources".

Resource Details				
Name	Console	IP Address	Created	State
openstack_compute_instance_v2.Po...	https://129.40.156.2	129.40.186.43	09/24/2020 1:41PM	● Running
Resources per page 10 1-1 of 1 Resources				
				⋮

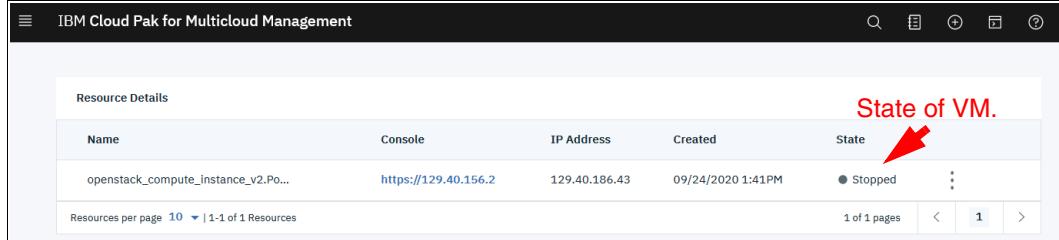
Figure A-20 IBM Cloud Pak for Multicloud Management: Checking the state of the virtual machine

2. You can shut down IBM i from IBM Navigator for i by opening a 5250 emulator and running the IBM i command **PWRDWNSYS**.

Note: The Power Down System (**PWRDWNSYS**) command prepares the IBM i system for shutdown and then starts the power-off sequence. For more information, see [Power Down System \(PWRDWNSYS\)](#).

You can also power off your IBM i VM directly from IBM Cloud Pak for Multicloud Management. In Figure A-20, click the three dots next to Running, and click **Shutdown OS**.

The state changes to Stopped, as shown in Figure A-21.

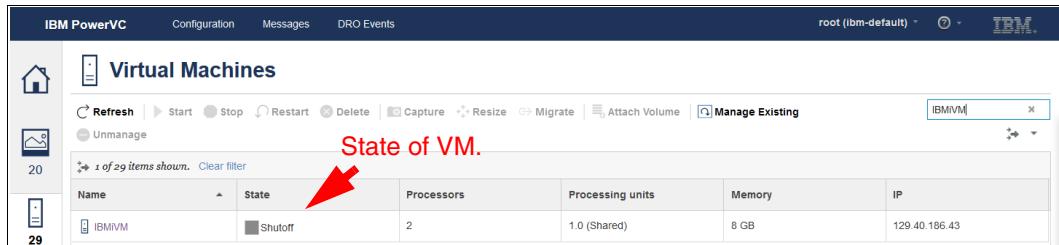


A screenshot of the IBM Cloud Pak for Multicloud Management interface, similar to Figure A-20 but showing the VM in a different state. The "State" column now shows "Stopped" with a grey dot icon. Red arrows point from the text "State of VM." to the "State" column and the three-dot menu next to it. The rest of the interface is identical to Figure A-20.

Resource Details				
Name	Console	IP Address	Created	State
openstack_compute_instance_v2.Po...	https://129.40.156.2	129.40.186.43	09/24/2020 1:41PM	● Stopped
Resources per page 10 1-1 of 1 Resources				
				⋮

Figure A-21 IBM Cloud Pak for Multicloud Management: Stopped state of virtual machine

If you open IBM PowerVC, you see that the VM is in the Shutoff state, as shown in Figure A-22.



A screenshot of the IBM PowerVC interface. The title bar says "IBM PowerVC". Below it is a table titled "Virtual Machines". The "State" column shows "Shutoff" with a grey dot icon. Red arrows point from the text "State of VM." to the "State" column and the three-dot menu next to it. The rest of the interface is identical to Figure A-20.

Virtual Machines						
<input type="button" value="Refresh"/> <input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Restart"/> <input type="button" value="Delete"/> <input type="button" value="Capture"/> <input type="button" value="Resize"/> <input type="button" value="Migrate"/> <input type="button" value="Attach Volume"/> <input type="button" value="Manage Existing"/>						
1 of 29 items shown. Clear filter						
Name	State	Processors	Processing units	Memory	IP	
IBMVIM	Shutoff	2	1.0 (Shared)	8 GB	129.40.186.43	

Figure A-22 IBM PowerVC: Shutoff state of virtual machine

3. To delete the resources, go to **Deployed Instances**, as shown in Figure A-23 on page 145.

Deployed Instances

1. Search deployed instance.

2. Click Destroy Resources.

Name	Provider	Template	Created	Status
IBMi_VM Namespace: default	OpenStack	IBMi_VM	09/24/2020 1:41 PM	Ready

Figure A-23 IBM Cloud Pak for Multicloud Management: Destroying the resources of the virtual machine

A confirmation window opens. Type **Destroy** to confirm that you want to delete the resources, and then click **Destroy**. The instance status changes from Active to In Progress, and it takes a few moments to change the status to Destroyed.

Note: Destroying removes the resources from the cloud, in this particular case from the on-premises IBM PowerVC, but it leaves the deployed instance in IBM Cloud Automation Manager.

- Now, you must remove the instance, as shown in Figure A-24. Select **View Details** → **Delete Instance**. A new confirmation window opens. Type **Delete** to confirm the deletion of the instance, and then click **Delete**. The instance is deleted, and there are no traces of the deployed instance.

Deployed Instances

State of VM.

Click Delete Instance.

Name	Provider	Template	Created	Status
IBMi_VM Namespace: default	OpenStack	IBMi_VM	09/24/2020 1:41 PM	Destroyed

Figure A-24 IBM Cloud Pak for Multicloud Management: Deleting an instance of the virtual machine

Moving a VM to IBM Cloud Object Storage in IBM Cloud from IBM PowerVC

In this section, an IBM i VM image is generated in IBM PowerVC, and then moved to IBM Cloud Object Storage in IBM Cloud.

Overview

IBM PowerVC is an advanced virtualization and cloud management offering that is built on OpenStack that provides simplified virtualization management and cloud deployments for IBM AIX, IBM i, and Linux VMs running on IBM Power Systems.

It also provides numerous operational benefits, such as one-click system evacuation for simplified server maintenance, dynamic resource optimization (DRO) to balance server usage during peak times, and automated VM restart to recover from failures. Users can easily import and export VM images (in the standard Open Virtual Application (.OVA) format) from IBM PowerVC and upload them into IBM Cloud for easy back-and-forth image mobility.

With this public cloud solution, customers can grow at their own pace, run enterprise workloads when and where they want, and choose from various flexible OS, compute, storage, and networking configurations.

This section describes how you can export VMs that are managed by IBM PowerVC into IBM Cloud. In this particular case, we select an IBM i V7.3. VM to be exported. Taking a capture of an IBM i image takes a snapshot of the volumes that the OS is running, and you can use the image to create VM clones. Also, the image can be imported and exported to different environments, whether on-premises, off-premises, or both.

Capturing a virtual machine from IBM PowerVC

To capture the VM, complete the following steps:

1. To ensure that all the data is captured permanently for creating a master image, shut down the IBM i system, as shown in Figure A-25. You may shut down your VM directly from the IBM Power VC interface, but that is not a best practice for production environments.

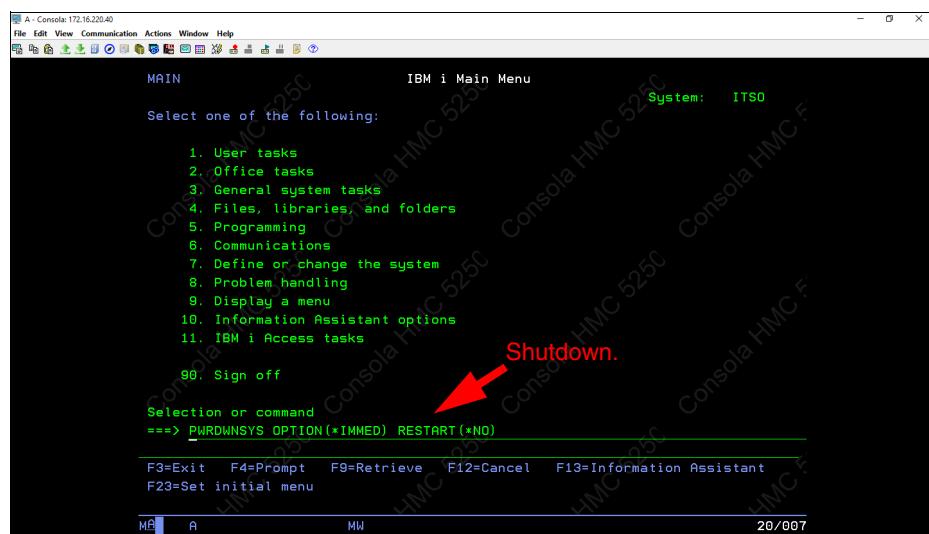


Figure A-25 IBM i Access Client Solutions: Shutting down the IBM i environment

Important: Before a VM can be captured, it must meet specific requirements. If you do not prepare the VM before you capture it, you might experience problems when you deploy the resulting image. For more information, see [Capturing a virtual machine](#).

Tip: Before you capture a VM image, install the cloud-init initialization package. For more information about cloud-init for AIX, IBM i, and Linux, see [Installing and configuring cloud-init](#).

Note: For AIX and IBM i, there are more methods to move the image into IBM Cloud. For more information, see [Migration strategies for IBM Power Systems Virtual Servers](#).

2. Log on to the IBM PowerVC GUI and go to the Virtual Machines view. Select the VM that you want to capture, which in this case is an IBM i system, and click **Capture**. A window opens, where you enter a name for a new image and select the volumes to capture. Click **Capture**, as shown in Figure A-26.

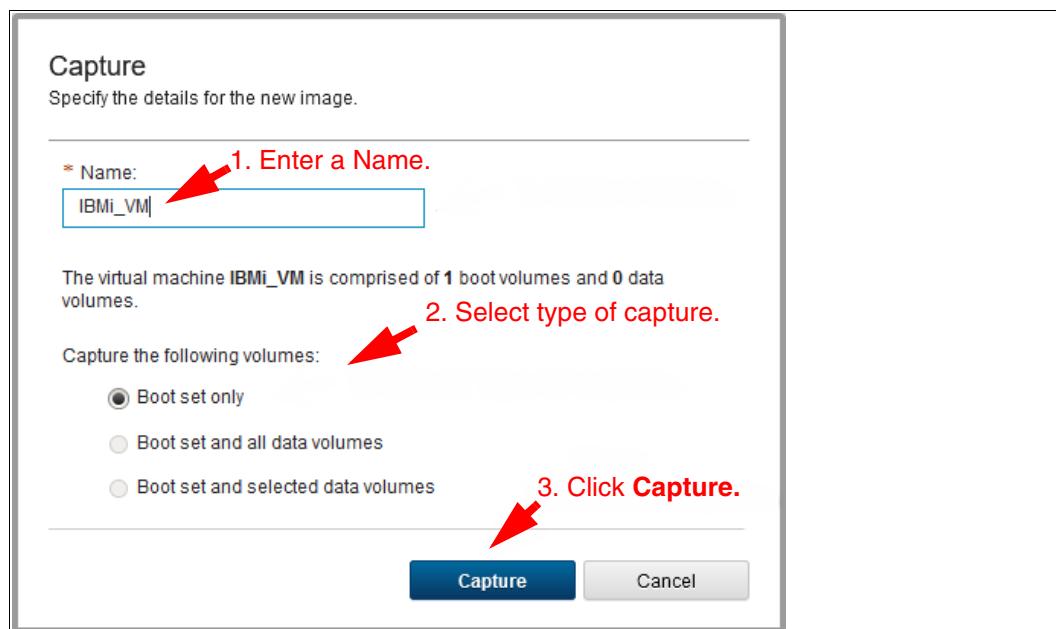
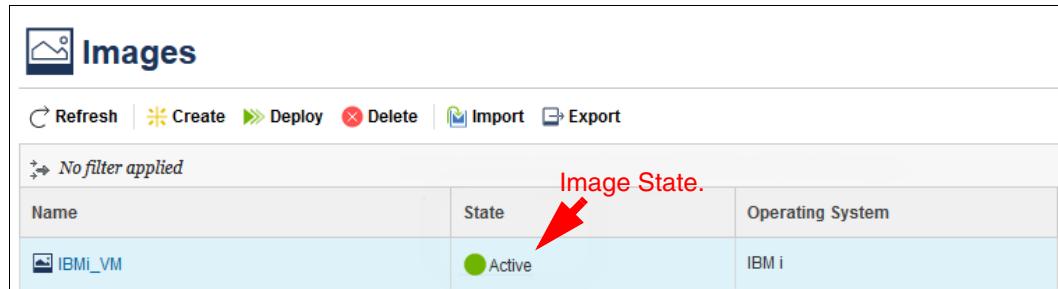


Figure A-26 IBM PowerVC: Capturing the image details

A Confirm Capture window opens that lists all the VM volumes that were chosen for capture. Click **Capture** again to start the capture process.

Note: For this example, the IBM i VM has an auxiliary storage pool (ASP) instead of an IASP, which is why we selected **Boot set only**.

After the image capture completes, the state of the image shows as Active, as shown in Figure A-27.



The screenshot shows the 'Images' section of the IBM PowerVC interface. At the top, there are buttons for Refresh, Create, Deploy, Delete, Import, and Export. Below that, a message says 'No filter applied'. A red arrow points to the 'State' column header, which is labeled 'Image State.' in red text. The main table has three columns: Name, State, and Operating System. One row is visible, showing 'IBMi_VM' in the Name column, 'Active' in the State column (with a green circle icon), and 'IBM i' in the Operating System column.

Name	State	Operating System
IBMi_VM	Active	IBM i

Figure A-27 IBM PowerVC: Image pane - Active

3. To export the captured image to an OVA package, open a terminal to IBM PowerVC and run the `ssh` command, or use PUTTY from your local computer. Example A-4 shows the command to list the available images in IBM Power VC.

Example A-4 IBM PowerVC: List command to view available images.

```
# powervc-image list
```

Example A-5 shows an example of how to export and compress the image from IBM PowerVC by using a CLI. You can use the IBM PowerVC GUI to export the image, but the CLI has more options for exporting. In this example, we use the IBM i 7.3 image that we created (IBMi_VM) and an Open Virtualization Format (OVF) that is generated from the image. As the root user, we add the OVF to an OVA, and compress them by running the `gz` command. The compression of the OVA depends on the size of the image.

Example A-5 IBM PowerVC: Example of generating the image by using the CLI

```
# powervc-image export --image IBMi_VM -p /home/ibmi.ova --compress
[Set the OS_PASSWORD environment variable to avoid this prompt] Enter password
for 'root':
Created temporary staging directory /var/opt/ibm/powervc/imgstaging/tmpokbjI5
Found image with ID 'f83e3622-ba9b-4221-a806-ff00695510f7' and name 'IBMi_VM'.
The export directory and the staging directory are on different file systems.
Register temporary file-copy volume driver.
Registered temporary driver PVC-Lite-File_tmpokbjI5 servicing location
/var/opt/ibm/powervc/imgstaging/tmpokbjI5
Cloning 'Image IBMi_VM volume 1' into temporary volume
'Image_IBMi_VM_volume_1_tmpokbjI5'.
The size to clone is 40 GiBs..Done cloning.
Migrate volume data for 'Image_IBMi_VM_volume_1_tmpokbjI5' from 'dc2_v7000_02' to
the target storage template 'PVC-Lite-File_tmpokbjI5 base template'.
Attaching volume..

Copying Image_IBMi_VM_volume_1 [100%] Rate: 26.02 MiB-per-S, ETA: 0:00:00 [H:MM:SS]
Detaching volume and finalizing metadata...
Copy complete after waiting 0:26:14 [H:MM:SS]

GiBs remaining to copy for image: 0
Creating image package with 1 volumes.
Creating OVF: /var/opt/ibm/powervc/imgstaging/tmpokbjI5/IBMi_VM.ovf
Creation of OVF completed.
Adding OVF to OVA /home/ibmi.ova
```

```
Adding volume 'Image_IBMi_VM_volume_1' to OVA.  
Cleaning up Lite-Volume export resources...  
Unregister the temporary file driver 'PVC-Lite-File_tmpokbjI5'.  
Cleaning up the temporary staging directory...  
Compressing OVA.  
Exported OVA /home/ibmi.ova.gz size: 7.09 GiB  
Time spent: 0:54:25 [H:MM:SS]  
Successfully finished creating image package /home/ibmi.ova.gz
```

Note: By default, the resulting OVA package is in the /var/opt/ibm/powervc/ova folder. However, you can choose the location by using **-p PATH** and specifying the name and path to receive the OVA package. In this example, we use the path /home because of the free space on it.

Tip: It is a best practice to compress the OVA package by running the **gzip --compress** command so that the package can be transferred faster from on-premises to off-premises.

After the images are successfully exported, a file that is called `ibmi.ova.gz` is created. As you can see, the size of the OVA is 7.09 GiB, which is much smaller than the image capture of the volumes, which was 40 GiB.

In this specific case, we copy the OVA package from `/home/ibmi.ova.gz` in IBM PowerVC to a Microsoft Windows PC by using the PuTTY Secure Copy (**pscp**) utility.

Configuring IBM Cloud Object Storage in IBM Cloud

An IBM Cloud Object Storage in IBM Cloud system is a breakthrough cloud platform that helps solve petabyte and larger storage challenges. It uses an innovative and cost-effective approach for storing large volumes of unstructured data while still ensuring scalability, security, availability, reliability, manageability, and flexibility.

Here are the key features of IBM Cloud Object Storage:

- ▶ New efficiencies that lower cost and increase performance.
- ▶ Use IBM Spectrum Discover for an artificial intelligence (AI) workflow.
- ▶ Store objects and files at a lower cost.
- ▶ Reduce the complexity of managing large-scale storage.
- ▶ Data protection without replication or triple copies.
- ▶ A leader in object storage for over 5 years.
- ▶ Built-in industry compliance for safe data protection.
- ▶ Flexible local or geo-dispersed data protection.
- ▶ Engineered for massive scalability.
- ▶ Pre-certified designs for easy configuration.
- ▶ Autodelete with policy enabled object expiration.

In this section, we import the OVA file that is generated in IBM PowerVC into IBM Cloud Object Storage. Because the OVA file is copied to a Windows PC, you can import it from there to IBM Cloud Object Storage.

To import the OVA file, complete the following steps:

1. Open your web browser, go the login page for [IBM Cloud](#), provide your account credentials, and click **Log in**.
2. Go to **Resources list**, and then click **Create resource**, as shown in Figure A-28.

The screenshot shows the IBM Cloud dashboard. On the left, there's a sidebar with various service icons and a tree view of resources: Devices (0), VPC infrastructure (0), Clusters (0), Cloud Foundry apps (0), Cloud Foundry services (0), Services (0), and Storage (0). The main area is titled 'Resource list'. At the top right, there are several navigation tabs: Catalog, Docs, Support, Manage, and a user profile. Below these tabs is a search bar with placeholder text 'Search resources and offerings...'. To the right of the search bar is a blue button labeled 'Create resource' with a '+' sign. Red arrows and labels are overlaid on the image: '1. Click Resource list.' points to the sidebar link, and '2. Click Create resource.' points to the blue button.

Figure A-28 IBM Cloud: Create resources

3. Click **Catalog**, and then select **Object Storage**, as shown in Figure A-29. If you do not see the Object Storage pane, use the **Search the catalog** field to find it.

The screenshot shows the IBM Cloud Catalog page. At the top, there's a search bar with placeholder text 'Search resources and offerings...' and a 'Catalog' tab which is highlighted. Below the search bar, the title 'IBM Cloud products' is displayed. A large blue banner on the right side says 'Over 200 products available for you to customize and build the solutions that you need for your business'. In the center, there's a 'Search the catalog...' input field. On the left, under 'Recommended for you', there's a box for 'Object Storage' which is highlighted with a blue border. The 'Object Storage' box contains the text: 'Provides flexible, cost-effective, and scalable cloud storage for unstructured data.' and 'Lite • Free • IAM-enabled'. To the right of this box is another product card for 'Red Hat OpenShift on IBM Cloud'. Red arrows and labels are overlaid on the image: '1. Select Catalog.' points to the 'Catalog' tab, and '2. Select Object Storage.' points to the 'Object Storage' box.

Figure A-29 IBM Cloud: Cloud Object Storage option

4. Select a plan. You can choose Lite or Standard. In this example, select **Lite**. Enter a **Service Name**, and insert a tag, as shown in Figure A-30 on page 151. Click **Create**.

The screenshot shows the 'Cloud Object Storage' creation interface on the IBM Cloud. It highlights three steps:

- 1. Select Plan.** A red arrow points to the 'Lite' plan row, which is highlighted with a red border. The 'Features' and 'Pricing' columns are visible.
- 2. Enter Service Name.** A red arrow points to the 'Service name' input field, which contains 'Cloud Object Storage-IBM i'.
- 3. Add tag.** A red arrow points to the 'Tags' section, which contains a single tag 'ibmi'.

Figure A-30 IBM Cloud: Creating an IBM Cloud Object Storage plan

Note: The difference between a Lite plan and a Standard plan is that the Lite plan is designed for development and evaluation purposes. It has a restricted number of ops and amount of storage, and it should not be used for production scenarios because the Lite plan does not have sufficient capacity. The API is the same for both plans. For more information about the plans, see [Plans and provisioning](#).

5. One of the crucial steps in designing your IBM Cloud Object Storage instance is organizing your data into buckets to meet your needs. To do so, go to **Buckets**, and click **Create bucket**.

In this specific instance, we select **Custom bucket**, and click it.

Another window opens, where we enter a unique *bucket name*, select a **Resiliency**, and select a **Location**, in this case, us-south, as shown in Figure A-31.

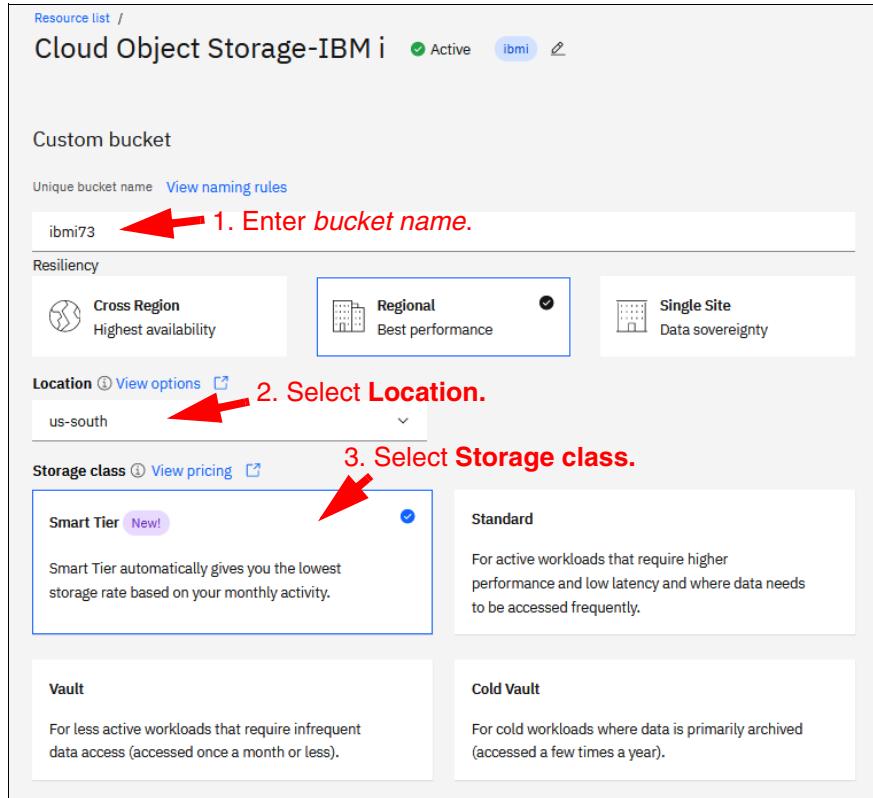


Figure A-31 IBM Cloud: Bucket customized

At the bottom of the window in Figure A-31, click **Create** (not shown here), and the bucket is created.

Note: In Figure A-31, we mention all the options for this example, but there are more options in the Custom bucket window. If you want, you can customize your bucket as needed. For more information about customizing buckets, see [Getting started with IBM Cloud Object Storage](#).

6. Transfer the image capture to the object storage. It is a best practice to use IBM Aspera® transfers to move the OVA package to improve data transfer performance under most conditions, especially with high latency and packet loss. Go to **Bucket**, and select the bucket that you created, as shown in Figure A-32 on page 153.

The screenshot shows the 'Cloud Object Storage-IBM i' resource list. The 'Buckets' tab is selected. A table displays the bucket information. The 'Name' column shows 'ibmi73'. To the right, under 'Bucket location.', it says 'us-south'. Red arrows point to the 'Bucket name.' and 'Bucket location.' labels.

Figure A-32 IBM Cloud: Bucket created

The Objects window opens. Drag the `ibmi.ova.gz` to this window to transfer the OVA package. You can monitor the status of the upload.

The screenshot shows the 'Objects' section of the storage console. Step 1, 'Select Objects.', has an arrow pointing to the 'Objects' tab in the left sidebar. Step 2, 'Drag and drop OVA here.', has an arrow pointing to the 'Drag and drop files (objects) or folders to upload' area. Step 3, 'Transfer starts.', has an arrow pointing to the 'Aspera transfers' sidebar where a transfer for 'ibmi.ova.gz' is listed as 'Initiating'.

Figure A-33 IBM Cloud: Uploading an object

Note: For more information about Aspera, see [Using the console](#).

Figure A-34 shows the uploaded object.

The screenshot shows the 'Objects' section after the upload. An arrow points to the 'ibmi.ova.gz' file in the list with the label 'An OVA uploaded.'. The file details are shown: Size 7.1 GB and Last modified 09/02/2020 5:12:46 PM.

Figure A-34 IBM Cloud: Uploaded object

7. Create a credential in IBM Cloud Object Storage by clicking **Service credentials**, selecting **New credential**, and providing the necessary information. Enter a name and select a **Role**, which in this case is **Writer** so that we can create and destroy buckets and objects.

To generate Hash-based Message Authentication Code (HMAC) credentials, click **Advanced Options**, and then select **Include HMAC Credential**. Verify that the option is selected before continuing. Figure A-35 shows the creation of a service credential with an HMAC credential. Click **Add** to generate a service credential.

Note: The bucket permissions assign access roles to users and Service IDs for buckets. For more information, see [Bucket permissions](#).

Note: HMAC credentials consist of an Access Key and Secret Key that are paired for use with S3-compatible tools and libraries that require authentication. For more information, see [Using HMAC credentials](#).

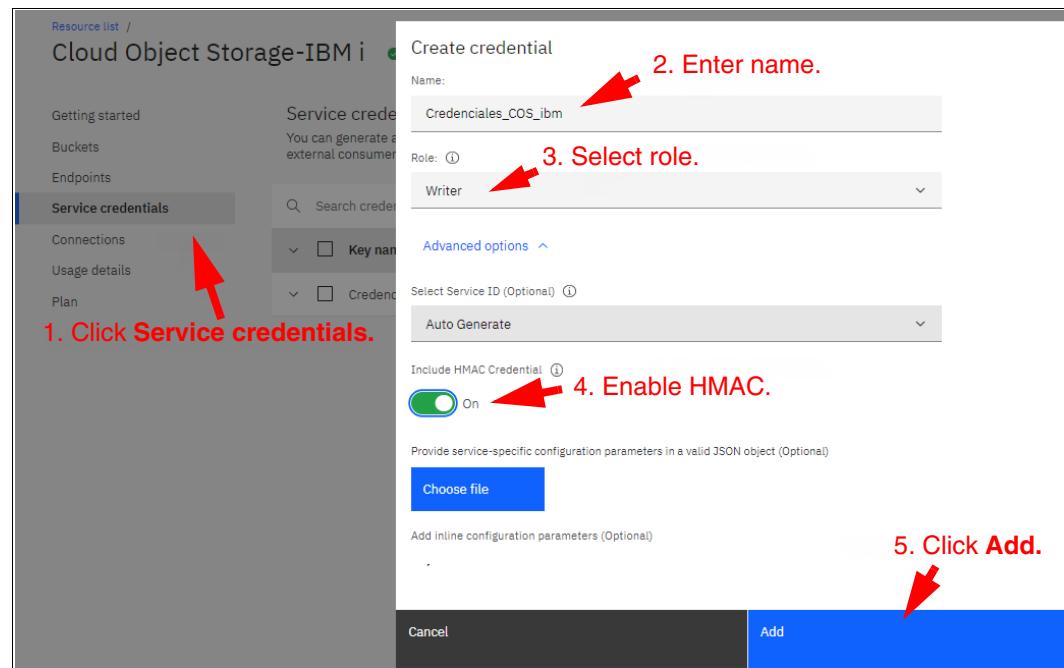


Figure A-35 IBM Cloud: Creating a credential for IBM Cloud Object Storage

8. In the Service credentials window that is shown in Figure A-36 on page 155, copy and save the cloud storage access and cloud storage secret keys, which you need later to import your image in to IBM Power Systems Virtual Server.

Cloud Object Storage-IBM i Active IBM

Getting started Buckets Endpoints Service credentials Connections Usage details Plan

Service credentials
You can generate a new set of credentials for cases where you want to manually connect an app or external consumer to an IBM Cloud™ service. [Learn more](#)

1 item selected

Key name	Date created
Credenciales_COS_ibmi	SEP 2, 2020 - 11:42:33 PM

```
{
  "apikey": "Hm19za-zA0SZAUJzyxGfgqE9pwmDn8AFBt6ZHYdu",
  "cos_bmc_kave": "a67393b851934d969383c560f2f78645",
  "access_key_id": "a67393b851934d969383c560f2f78645",
  "secret_access_key": "ac1ed4023647e7a454c2621096e0d69c68bc48498226550"
}
```

Figure A-36 IBM Cloud: Service credentials access key

Summary

Importing an OVA image to an IBM Power Systems Virtual Server requires that you provide some cloud storage details before you import an image:

- ▶ You must provide the bucket name and the location of the IBM Cloud Storage instance, as shown in Figure A-32 on page 153.
- ▶ At the same level in IBM Cloud Object Storage, you must review the service credential and copy the cloud storage access key and the cloud storage secret key, as shown in Figure A-36, and you must provide the image file name or object name, as shown in Figure A-34 on page 153.

Deploying IBM VM into IBM Power Systems Virtual Server in IBM Cloud by using IBM Cloud Pak for Multicloud Management

This section demonstrates how to create, and deploy a template by using Terraform for an IBM i VM by using IBM Cloud Pak for Multicloud Management in IBM Power Systems Virtual Server in IBM Cloud.

Overview

IBM Power Systems customers who rely on an on-premises private cloud solution include IBM PowerVC in that solution to provide the infrastructure-as-a-service (IaaS) layer. Now, these customers can quickly and economically extend their IT resources onto IBM Power Systems Virtual Servers.

IBM Power Systems Virtual Servers on IBM Cloud integrates AIX and IBM i capabilities into the IBM Cloud experience. Users receive fast self-service provisioning, flexible management, and access to a stack of enterprise IBM Cloud services, all with easy pay-as-you-use billing.

IBM Cloud Pak for Multicloud Management provides advanced multicloud management capabilities through IBM Cloud Automation Manager, which enables connectivity to IBM Power Systems Virtual Server in IBM Cloud to streamline VM deployments.

From that perspective, we describe how to deploy IBM i VM by using IBM Cloud Pak for MultiCloud Management into IBM IBM Power Systems Virtual Server in IBM Cloud.

Before you create your first IBM Power Systems Virtual Server instance, review the following prerequisites and complete them if necessary:

1. Create an IBM Cloud account.
2. Review the Identity and Access Management (IAM) information.
3. Create a public and private SSH key that you can use to securely connect to your IBM Power Systems Virtual Server.
4. If you want to use a custom AIX or IBM i image, you must create an IBM Cloud Object Storage instance and upload it there.
5. If you want to use a private network to connect to an IBM Power Systems Virtual Server instance, you must order the Direct Link Connect service. You cannot create a private network during the VM provisioning process. You must first use the IBM Power Systems Virtual Server user interface, CLI, or API to create one (optional).

Creating an IBM Power Systems Virtual Server service

With IBM Power Systems Virtual Server, you cannot create a service instance if you do not have a Pay-As-You-Go account. You can convert your IBM Cloud account to a Pay-As-You-Go account by adding a credit card. Your credit card is billed monthly based on your usage.

Important: When creating or upgrading a Lite account to a Pay-As-You-Go account, IBM Cloud offers a USD 200 credit that can go toward creating, testing, and so on.

To create and set up an IBM Power System Virtual Server, complete the following steps:

1. Log in to the [IBM Cloud Catalog](#) by using your IBM Cloud account credentials.
2. In the Search Catalog box, enter Power Systems Virtual Server and click the **Power Systems Virtual Server** tile, as shown in Figure A-37.

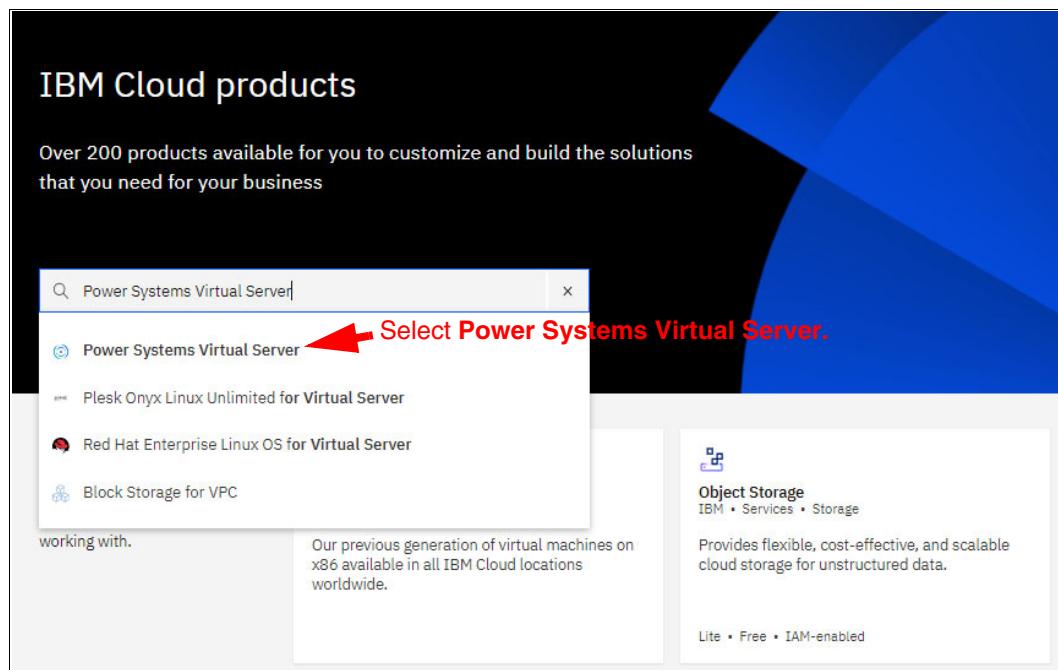


Figure A-37 IBM Cloud: Searching IBM Power Systems Virtual Server

3. In Figure A-38, specify a name for your service and choose where you would like to deploy your Power System Virtual Server instance. Table A-4 shows the locations where you can place your service.

Table A-4 IBM Power Systems Virtual Server data centers

Location	Region	Data center
Dallas, Texas	us-south	DAL13
Washington, D.C.	us-east	WDC04
Toronto, Canada	eu-east	TOR01
Frankfurt, Germany	eu-de	FRA04/ FRA05
London, United Kingdom	eu-gb	LON04/ LON06
Sydney, Australia	au-syd	SYD04

In this example, we use Washington, D.C., and then select a pricing plan. You can estimate the costs before you create the instance by clicking **Estimate costs**, which shows a box with a red border at the upper right. Enter **Service Name**, and add a tag to identify this service. Click **Create**.

1. Select Region.

2. Select a pricing plan.

3. Type Service name.

4. Insert tag.

5. Click Create.

Estimate costs.

Figure A-38 IBM Cloud: Creating an IBM Power Systems Virtual Server instance

The status of the IBM Power Systems Virtual Server instance changes to Provision in Progress, which takes a few minutes to change to Active, as shown in Figure A-39.

The screenshot shows the 'Resource list /' page for 'Power Systems Virtual Server-ibmipowervs'. On the left, there's a sidebar with 'Virtual server instances' selected, and a list of resources: SSH keys, Storage volumes, Boot images, and Subnets. The main area is titled 'Virtual server instances' with a sub-instruction 'Learn more about [Getting started](#) and [Creating a virtual server instance](#)'. Below this is a table header with columns 'Name', 'IPs', and 'Image'. A note at the bottom says 'The virtual server instances list is empty. To create virtual server instances, click ...'. In the top right corner, there's a status indicator 'Active' with a green checkmark, and a small red arrow points to it. Next to it are buttons for 'ibmipowervs' and a refresh icon.

Figure A-39 IBM Cloud: View of IBM Power Systems Virtual Server after creation

Note: For more information about creating an IBM Power Systems Virtual Server instance and its terminology, see [Getting started with IBM Power Systems Virtual Servers](#).

Generating an IBM Cloud IAM token by using an API key

Generate an IBM Cloud IAM token by using either your IAM API key or the service ID of an API key. IBM Cloud APIs can be accessed only by users who are authorized by an assigned IAM role. Each user who calls the API must pass credentials for the API to authenticate.

The API key is a permanent credential that can be reused if you do not lose the API key value or delete the API key in the account. This process is also used if you are developing an application that must work with other IBM Cloud services. You must use a service ID API key to get an access token to be passed to each of the IBM Cloud services. To create the API key, complete the following step:

1. In Figure A-40 on page 159, click **Manage** → **Access (IAM)**. Then, in the left pane, click **API keys**.
2. Click **Create an IBM Cloud API key**, and the Create API key window opens. Enter a name and description, and click **Create**.
3. In the next window that opens, copy the API key and keep it safe. You also can download the key.

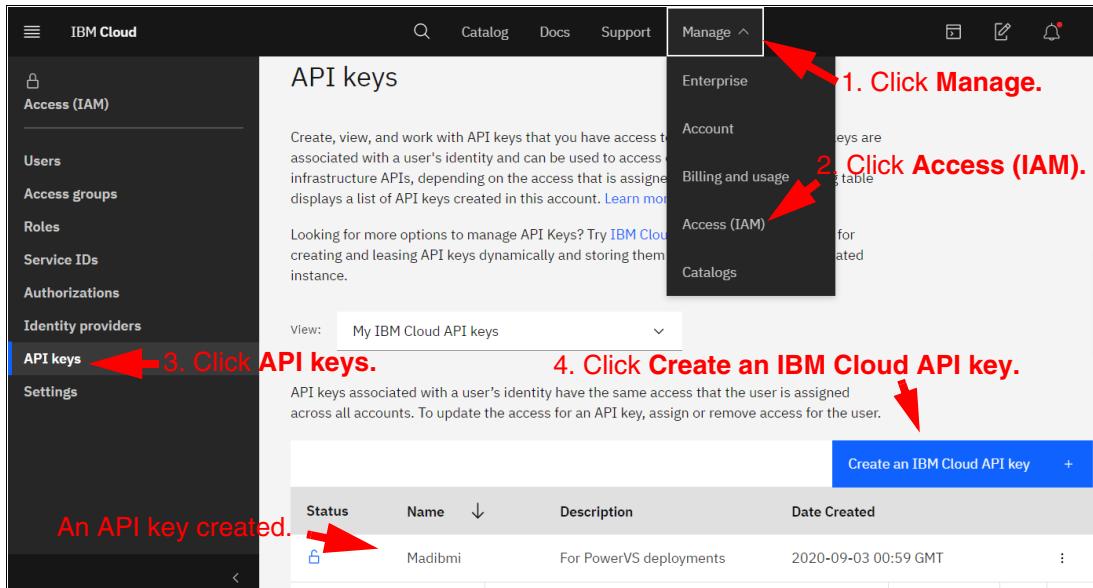


Figure A-40 IBM Cloud: API key creation

Creating a public and private key to connect to an IBM Power Systems Virtual Server instance

To create a public and private SSH key, complete the following steps:

1. In the IBM Cloud interface, open the IBM Cloud CLI from any web browser.

Note: For more information about how to start a session of the IBM Cloud Shell, see [Getting started with IBM Cloud Shell](#).

2. The Dallas (us-south) region is the default region. In this example, you must switch the region. To do so, first list of all your services by using your IBM Cloud account to run the command that is shown in Example A-6.

Example A-6 Listing all the services under your IBM Cloud account

```
$ibmcloud pi service-list
ID                                     Name
crn:v1:bluemix:public:power-iaas:us-east:a/5500e8b400534694884c8efaa020cc7b:30295a9a-9ffa-4b5b-8b7d-efa06f3d38a7::: Power Systems Virtual
Server-ibmi
```

3. To target the Cloud Instance ID of your service, copy the generated ID (shown in Example A-6) by running the command that is shown in Example A-7.

Example A-7 Targeting your service in IBM Power Systems Virtual Server

```
$ibmcloud pi service-target
crn:v1:bluemix:public:power-iaas:us-east:a/5500e8b400534694884c8efaa020cc7b:30295a9a-9ffa-4b5b-8b7d-efa06f3d38a7:::

Targeting service crn:v1:bluemix:public:power-iaas:us-east:a/5500e8b400534694884c8efaa020cc7b:30295a9a-9ffa-4b5b-8b7d-efa06f3d38a7:::....
```

4. Target the name of region to use, which in this example is us-east, as shown in Example A-8.

Example A-8 Targeting the name of the region for the service

```
$ ibmcloud target -r us-east
```

5. In Example A-9, you create a public key on IBM Cloud Shell by using the **ssh-keygen** tool. After the tool runs, enter a file in which to save the key, which in this example is **clou dibmi**. Enter the same passphrase twice, and press Enter. A key fingerprint and a key that is made of random images are displayed.

Example A-9 Generating a public/private RSA key pair

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/m_avalos/.ssh/id_rsa): clou dibmi
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in clou dibmi.
Your public key has been saved in clou dibmi.pub.
The key fingerprint is:
SHA256:QgK+dD8Nv+DIATxovP5r50E6A0GED3pJbh977K6uDH4
m_avalos@cloudshell-c8aa8507-e994-41a2-ba16-d519bd25ad11-1-584985cf czhk5
The key's randomart image is:
+---[RSA 2048]----+
|o...
|*...
+0*.o o
+.B=.+ +
=...== S
. .+=+0+
.. oo+.
.o.E..o
.=*=+.
+---[SHA256]----+
```

6. To display the public/private RSA key pair that was generated in Example A-10, run the **cat** command, as shown in Example A-10.

Example A-10 Displaying the public/private RSA key pair that is generated

```
cat clou dibmi.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQCoSaFdNPG2CQ5dFX9b1w0y+ihDE6F7t0yaHeg6s7auLONkR0
hGb3h2p/K7pUfQ8J/UHEnTsR6jPp9QSLgpzqzTqDK7fihmv51HiZ7kyfprwFwHv3QwomcJ6L6xAsUTC
PYE+aWTIxRm2iwE96YSKnqFip3J0+TxtI6EBxcYt2n0DKdn/i/gUPiGg1WW9bFbXdxU7AZdFTgtCh2D
27sHOZA/RuJmtUD3rVTZD6hqjUoG/Eg0CT9ukhameVjcX/i3Kj05x1hSa1Iyd816aRhP6BnIeMrwdye
v08RIeWV1XvbHOyopfWe161QKfSuZSBqVpm035MXeeN3KJhdvh4dJJ4B
m_avalos@cloudshell-c8aa8507-e994-41a2-ba16-d519bd25ad11-1-584985cf czhk5
```

7. Example A-11 shows the import of the RSA public key that was generated in Example A-10 on page 160. The key is named `ibmikey`. Copy the output of `cloudimbi.pub`.

Example A-11 Importing the RSA public key

```
$ ibmcloud pi key-create ibmikey --key "ssh-rsa"
AAAAB3NzaC1yc2EAAAQABAAQCo...F9b1w0y+ihDE6F7tOyaHeg6s7auLONkR0
hGb3h2p/K7pUfQ8J/UHEnTsR6jPp9QSLgpzqzTqDK7fihmv51HiZ7kyfprwFwHv3QwomcJ6L6xAsUTc
PYE+aWTIxRm2iwE96YSKnqFip3J0+TxtI6EBxcYt2n0DKdn/i/gUPiGg1WW9bFbXdxU7AzdFTgtCh2D
27sHOZA/RuJmtUD3rVTZD6hqqjUoG/Eg0CT9ukhameVjcX/i3Kj05x1hSa1Iyd816aRhP6BnIeMrwdye
v08RIeWV1XvbH0yopfWe161QKfSuZSbqVpm035MXeeN3KJhdvh4dJJ4B
m_avalos@cloudshell-c8aa8507-e994-41a2-ba16-d519bd25ad11-1-584985fczhk5"
Creating key ibmikey under account Marcelo Avalos's Account as user
marcelo.avalos@ibm.com...
Key ibmikey created.
Name      Key
ibmikey   ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQCo...F9b1w0y+ihDE6F7tOyaHeg6s7auLONkR0
hGb3h2p/K7pUfQ8J/UHEnTsR6jPp9QSLgpzqzTqDK7fihmv51HiZ7kyfprwFwHv3QwomcJ6L6xAsUTc
PYE+aWTIxRm2iwE96YSKnqFip3J0+TxtI6EBxcYt2n0DKdn/i/gUPiGg1WW9bFbXdxU7AzdFTgtCh2D
27sHOZA/RuJmtUD3rVTZD6hqqjUoG/Eg0CT9ukhameVjcX/i3Kj05x1hSa1Iyd816aRhP6BnIeMrwdye
v08RIeWV1XvbH0yopfWe161QKfSuZSbqVpm035MXeeN3KJhdvh4dJJ4B
m_avalos@cloudshell-c8aa8507-e994-41a2-ba16-d519bd25ad11-1-584985fczhk5
```

8. To confirm that the key was successfully added, run the `ibmcloud pi key` command, as shown in Example A-12.

Example A-12 Verifying that addition of the public key

```
$ ibmcloud pi key ibmikey
Name      Key
ibmikey   ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQCo...F9b1w0y+ihDE6F7tOyaHeg6s7auLONkR0
hGb3h2p/K7pUfQ8J/UHEnTsR6jPp9QSLgpzqzTqDK7fihmv51HiZ7kyfprwFwHv3QwomcJ6L6xAsUTc
PYE+aWTIxRm2iwE96YSKnqFip3J0+TxtI6EBxcYt2n0DKdn/i/gUPiGg1WW9bFbXdxU7AzdFTgtCh2D
27sHOZA/RuJmtUD3rVTZD6hqqjUoG/Eg0CT9ukhameVjcX/i3Kj05x1hSa1Iyd816aRhP6BnIeMrwdye
v08RIeWV1XvbH0yopfWe161QKfSuZSbqVpm035MXeeN3KJhdvh4dJJ4B
m_avalos@cloudshell-c8aa8507-e994-41a2-ba16-d519bd25ad11-1-584985fczhk5
```

Importing a boot image into IBM Power Systems Virtual Server from IBM Cloud Object Storage

You can import an AIX or IBM i boot image by using the IBM Power Systems Virtual Server CLI or the console. In this example, we use the user interface to import a boot image.

In this example, all the data centers use Tier 1 (NVMe-based flash storage) or Tier 3 (SSD flash storage) storage types. The Tier 1 storage type is best for customers who require higher throughput. Customers who do not require exceptionally high throughput and are looking to minimize costs want to select Tier 3. The storage types cannot be changed after the volume is created. A VM cannot have disks from both storage types. Large boot images take time to successfully import. You might experience a delay before receiving a confirmation message.

If you previously deployed a VM on an old storage type (SSD or standard), no action is required. Your VM continues to run by using the old storage type. You can also add new disks from those legacy tiers.

Note: The **Image file name** field supports the following formats: .ova, .ova.gz, .tar, .tar.gz, and .tgz.

Table A-5 describes the necessary fields that you must complete before you import a boot image.

Table A-5 Fields to complete before you import a boot image

Field	Description
Catalog image name	Enter the name that you want displayed in your catalog.
Storage type	Select whether you want Tier 1 (NVMe-based flash storage) or Tier 3 (SSD flash storage) for the storage type. A VM cannot have disks from both Tier 1 and Tier 3 storage types.
Region	The IBM Cloud Object Storage region. Select either us-east , us-south , or eu-de for the region.
Image file name	Enter the file name of the image. The image file name must not contain spaces. The supported file formats are .tar and .ova. You can compress image files by using gzip . The supported file name extensions are .ova, .ova.gz, .tar, .tar.gz, and .tgz. You must use the private endpoint domain.
Bucket name	The IBM Cloud Object Storage bucket name. Subfolders can be used and specified as bucketName/optional/folders. Optional folders are created automatically if they do not exist. Optional folders can be added during an export image operation to IBM Cloud Object Storage. To identify your bucket name, select the Menu icon, and then select Resource list → Storage → Cloud Object Storage name → Buckets .
Cloud Object Storage access key	To identify your access key, select the Menu icon, and then select Resource list → Storage → Cloud Object Storage name → Service credentials → View credentials . Copy the access_key_id value and paste it into this field.
Cloud Object Storage secret key	To identify your secret key, select the Menu icon, and then select Resource list → Storage → Cloud Object Storage name → Service credentials → View credentials . Copy the secret_access_key value and paste it into this field.

To import a boot image by using the IBM Power Systems Virtual Server user interface, complete the following steps:

1. In IBM Cloud, go to the navigation menu and select **Resource list** → **Services**, and then select the resources of the IBM Power Systems Virtual Server, in this case Power Systems Virtual Server-ibmi.
2. Inside the Power Systems Virtual Server-ibmi resource list, on the left side, click **Boot images**.

3. A window opens where you specify the details for importing an OVA from IBM Cloud Object Storage to IBM Power Systems Virtual Server. The OVA is a new boot image. For the necessary details, see Figure A-34 on page 153 (Image Filename), Figure A-32 on page 153 (Bucket name and Region (the bucket is at us-south)), and Figure A-36 on page 155 (Cloud storage access key and Cloud storage secret key). All these details are shown in Figure A-41. After the details are entered, click **Import image**.

Import boot image
Import a new boot image and specify its details

* Catalog image name: ibmi73vm

Storage type: Tier 3

Cloud storage details

* Region: us-south

* Image filename: ibmi.ova.gz

* Bucket name: ibmi73

* Cloud storage access key: a67393b851934d909353c560f2f78646

* Cloud storage secret key: ac10d4b23847d7a454c2821896ecd694c68bc48498226550

Import image

Figure A-41 IBM Power Systems Virtual Server: Importing image

Note: After the image is imported, the status of the boot image might be Queued. Depending on the size of the image, it can take several minutes for the status to change to Active. When the state is active, it is ready to be selected when you create a virtual server instance.

Ensure that you verify and obtain the boot image ID or name because that value is needed to deploy a VM. The **ibmcloud pi images** command shows three things: An ID, Name, and Address of the image, as shown in Example A-13.

Example A-13 Listing the images that are available in IBM Power Systems Virtual Server by using IBM Cloud Shell

\$ **ibmcloud pi images**

ID	Name	Address
b5b250ae-d7a5-40cc-bec6-24121a9a6aa7	ibmi73vm	/pcloud/v1/cloud-instances/f2140984c9ee4a1faa96a227d65d59df/images/b5b250ae-d7a5-40cc-bec6-24121a9a6aa7

Now that the image is imported, you can deploy a VM directly from the virtual server instance. In this example, you use an IBM i V7.3 system. When you create virtual server instances in IBM Power Systems Virtual Server, you can boot the selected image.

There are other available images in IBM Power Systems Virtual Server with different releases of AIX, IBM i, and Linux, and they appear when you create them by using the IBM Cloud GUI.

Note: The IBM Power Systems Virtual Server offering supports IBM i V7.2 or later. For more information about the supported AIX, IBM i, or Linux OS versions, see [Deploying a custom image within a Power Systems Virtual Server](#).

Creating a template to deploy a virtual machine into an IBM Power Systems Virtual Server instance

A basic Terraform template to deploy a VM into IBM Power Systems Virtual Server uses the `ibm_pi_instance` resource. In this particular case, a sample Terraform block depicts the structure of a template with six built-in blocks (modules) to create a single VM IBM Power Systems Virtual Server in IBM Cloud from a boot image that is uploaded, as described in “Moving a VM to IBM Cloud Object Storage in IBM Cloud from IBM PowerVC” on page 146.

Here are the blocks or modules for this template:

- ▶ provider.tf
- ▶ main.tf
- ▶ variables.tf
- ▶ output.tf
- ▶ camvariables.json
- ▶ version.tf

IBM Cloud Provider is one of the blocks that is used to manage the provisioning and orchestration of IBM Cloud resources with the Terraform open source provisioning engine. The provider must be configured with the correct credentials before it can be used. To configure the IBM Cloud Provider, you must use the same details that are described in Table A-6.

Table A-6 Provider configuration with different provider parameters

Input parameter	Description
version	The version of the IBM Cloud Terraform provider.
ibmcloud_api_key	The IBM Cloud API key to authenticate with the IBM Cloud platform.
region	The IBM Cloud region where you want to create the Power Systems resources.
zone	The zone of an IBM Cloud region where you want to create the Power Systems resources. This value is required if you want to work with resources in a multizone-capable region. For example, if you want to work in the eu-de region, you must enter eu-de-1 or eu-de-2. You can specify the zone in the provider block or retrieve the value from the IC_ZONE or IBMCLOUD_ZONE environment variables. If both environment variables are specified, IC_ZONE takes precedence.

Note: For more information about the Terraform provider block configuration that is related to Power Systems instances, see [Terraform provider block configuration](#).

Example A-14 shows a sample of the IBM Cloud Terraform provider.

Example A-14 Sample provider.tf for Terraform deployment to an IBM Cloud Terraform provider

```
provider "ibm" {
    version = "~> 1.12"
    ibmcloud_api_key = var.ibmcloud_api_key
    region           = var.ibmcloud_region
}
```

Important: In this particular deployment, we are not using zone, which means that if you want to create, update, or delete Power Systems resources in a multizone-capable region, you must specify the zone in the provider block of your Terraform configuration file, for example:

```
zone          = var.ibmcloud_zone
```

For more information about a provider block and multizone-capable region, see [Terraform provider block configuration](#).

Table A-7 shows information about the network that your IBM Power Systems Virtual Server instance is connected to.

Table A-7 Information about the network for your IBM Power Systems Virtual Server instance

Input parameter	Description
ibm_pi_network	Provides a network resource so that a network can be created, updated, and deleted.
count	Creates multiple instances according to a count (count_index, which is the distinct index number starting with 0 that corresponds to this instance).
pi_network_name	The subnet that the instance belongs to or the name of the network.
pi_cloud_instance_id	The cloud instance ID for this account.
pi_network_type	The type of network what you want to create, such as pub-vlan or vlan.

Note: For more information networking for IBM Power Systems Virtual Server, see [ibm_pi_network](#).

Table A-8 shows details about a public network that is used for an IBM Power Systems Virtual Server instance.

Table A-8 Information about a public network that is used for an IBM Power Systems Virtual Server instance

Input parameter	Description
ibm_cloud_instance_id	The service instance that is associated with the account.

Table A-9 shows the input parameters that are used by a Power Systems resource to create, update, or delete a Power System Virtual Server image.

Table A-9 Information about creating, updating, or deleting an IBM Power Systems Virtual Server image

Input parameter	Description
ibm_pi_image	Import the details of an existing IBM Power Systems Virtual Server Cloud image as a read-only data source. Then, you can reference the fields of the data source in other sources within the same configuration by using interpolation syntax.
pi_image_name	The name of the image.
pi_cloud_instance_id	The <code>cloud_instance_id</code> for this account.

Table A-10 shows the details to create or update a IBM Power Systems Virtual Server instance.

Table A-10 Information to create or update an IBM Power Systems Virtual Server instance

Input parameter	Description
ibm_pi_instance	Create or update an IBM Power Systems Virtual Server instance.
pi_memory	The amount of memory that you want to assign to your instance in gigabytes.
pi_processors	The number of vCPUs to assign to the VM (as visible within the guest OS).
pi_instance_name	The name of the IBM Power Systems Virtual Server instance.
pi_proc_type	The type of processor mode in which the VM runs (shared/dedicated).
pi_image_id	The ID of the image that you want to use for your IBM Power Systems Virtual Server instance. The image determines the OS that is installed in your instance. To list the available images, run the <code>ibmcloud pi images</code> command.
pi_volume_ids	The list of volume IDs that you want to attach to the instance during creation.
pi_network_ids	The list of network IDs that you want to assign to the instance.
pi_key_pair_name	The name of the SSH key that you want to use to access your IBM Power Systems Virtual Server instance. The <code>ssh</code> key must be uploaded to IBM Cloud.
pi_sys_type	The type of system on which to create the VM (s922, e880, or any).
pi_replication_policy	The replication policy that you want to use. If this parameter is not set, none is used by default.
pi_replication_scheme	The replication scheme that you want to set (pre-fix/suffix).
pi_replicants	The number of instances that you want to provision with the same configuration. If this parameter is not set, 1 is used by default.
pi_cloud_instance_id	The <code>cloud_instance_id</code> for this account.

Regarding the main entry point for the template, the `main.tf` sample is shown in Example A-15 on page 167. It includes the routine for IBM Power Systems Virtual Server in context with images, networking, public networking, and instance. The main template also describes all the variables and resources that are deployed on the cloud.

Example A-15 Sample of the main.tf template to be used by the IBM Cloud Provider for Power Systems

```
data "ibm_pi_image" "PowerVS_images" {
    pi_image_name      = var.image_name
    pi_cloud_instance_id = var.power_instance_id
}

resource "ibm_pi_network" "power_networks" {
    count              = 1
    pi_network_name    = var.networkname
    pi_cloud_instance_id = var.power_instance_id
    pi_network_type    = "pub-vlan"
}

data "ibm_pi_public_network" "dsnetwork" {
    depends_on          = [ibm_pi_network.power_networks]
    pi_cloud_instance_id = var.power_instance_id
}

resource "ibm_pi_instance" "PowerVS_instance" {
    pi_memory           = var.memory
    pi_processors        = var.processors
    pi_instance_name     = var.vm_name
    pi_proc_type         = var.proc_type
    pi_image_id          = data.ibm_pi_image.PowerVS_images.id
    pi_network_ids       = [data.ibm_pi_public_network.dsnetwork.id]
    pi_key_pair_name     = var.ssh_key_name
    pi_sys_type          = var.system_type
    pi_replicants        = var.replicants
    pi_cloud_instance_id = var.power_instance_id
}
```

In the main.tf sample, we do not use some ibm_pi_instance inputs parameters, such as pi_volume_ids, pi_replication_policy, and pi_replication_scheme because they are not required in this particular deployment.

Example A-16 provides information about the default values of the variables that are needed to run the template. The variables in Terraform provide for user input within IBM Cloud Pak for Multicloud Management.

Example A-16 Sample variables.tf template to be used by IBM Cloud Provider for Power Systems

```
# IBM Cloud configuration
variable "ibmcloud_api_key" {
    description = "Indicate the IBM Cloud API key to use"
}

variable "ibmcloud_region" {
    description = "Indicate which IBM Cloud region to connect to"
    default     = "us-east"
}

variable "power_instance_id" {
    description = "IBM Power Systems Virtual Server instance ID associated with your IBM Cloud account (note that this is NOT the API key)"
```

```

        default      = "30295a9a-9ffa-4b5b-8b7d-efa06f3d38a7"
    }

variable "ssh_key_name" {
    description = "SSH key name in IBM Cloud to be used for SSH logins"
    default      = "ibmikey"
}

# Boot Image
variable "image_name" {
    description = "Name of the image from which the virtual machine should be
deployed"
    default      = "ibmi73vm"
}

# Virtual Machine configuration
variable "vm_name" {
    description = "Name of the virtual machine"
    default      = "ITSOVS"
}

variable "memory" {
    description = "Amount of memory (GB) to be allocated to the virtual machine"
    default      = "16"
}

variable "processors" {
    description = "Number of virtual processors to allocate to the virtual machine"
    default      = "0.25"
}

variable "proc_type" {
    description = "Processor type for the LPAR - shared/dedicated"
    default      = "shared"
}

variable "system_type" {
    description = "Type of system on which the VM should be created - s922/e880"
    default      = "s922"
}

variable "replicants" {
    description = "Number of virtual machine instances to deploy"
    default      = "1"
}

# Network configuration
variable "networkname" {
    description = "Name of the Network"
    default      = "CMN"
}

```

Important: If you use a multizone-capable region, you must specify the zone. However, you should declare variables in the variables block, as shown in the following lines:

```
variable "ibmcloud_zone" {
    description = "Indicate the zone within the region to connect to (only needed for multi-zone regions--e.g., eu-de-2)"
    default      = "us-east"
}
```

Example A-17 provides information about important variables that can be queried after being deployed through Terraform and the Terraform and Service Automation user interface.

Example A-17 Sample of the output.tf template that is used by the IBM Cloud Provider for Power Systems

```
#Displays the IP address assigned
output "vm-ip" {
    value = ibm_pi_instance.PowerVS_instance.addresses
}
```

The camvariables.json file contains metadata that is used in IBM Cloud Automation Manager to augment the support for variables in Terraform. Metadata adds more parameters for existing variables that are understood and handled by IBM Cloud Automation Manager. Here, you define template_input_params for all the variables in the main.tf file.

When you deploy the template in the IBM Cloud Automation Manager user interface, all these defined variables are displayed. After you enter the values in the user interface, they are passed on to a JSON file that is sent for execution to Terraform. Terraform matches these parameters with the variables in the variables.tf file.

Tip: You can create camvariables.json and camtemplate.json files for your template, although it is not mandatory to have them in the template. However, their presence simplifies the deployment process, as shown in the camvariables.json in Example A-18.

Example A-18 Sample of camvariables.json used by IBM Cloud for Power Systems

```
{
    "output_datatype": "content_template_output",
    "input_datatypes": [ ],
    "input_namespaces": [ ],
    "output_namespace": "",
    "input_groups": [
        {
            "name": "cloud",
            "label": "IBM Cloud Configuration"
        },
        {
            "name": "image",
            "label": "Boot image"
        },
        {
            "name": "system",
            "label": "System Configuration"
        }
    ]
}
```

```

        "name": "vm",
        "label": "Virtual Machine Configuration"
    },
    {
        "name": "net",
        "label": "Network Configuration"
    }
],
"output_groups": [
    {
        "name": "content_template_output",
        "label": "Outputs"
    }
],
"template_input_params": [
    {
        "name": "ibmcloud_api_key",
        "label": "IBM Cloud API Key",
        "description": "Indicate the IBM Cloud API key to use",
        "type": "string",
        "default": "",
        "regex": "",
        "group_name": "cloud",
        "required": true,
        "secured": true,
        "hidden": false,
        "immutable": false,
        "immutable_after_create": false
    },
    {
        "name": "ibmcloud_region",
        "label": "IBM Cloud Region (e.g., us-south)",
        "description": "Indicate which IBM Cloud region to connect to",
        "type": "string",
        "default": "us-east",
        "regex": "",
        "group_name": "cloud",
        "required": true,
        "secured": false,
        "hidden": false,
        "immutable": false,
        "immutable_after_create": false
    },
    {
        "name": "power_instance_id",
        "label": "IBM Power Systems Virtual Server Instance ID",
        "description": "IBM Power Systems Virtual Server instance ID associated with your IBM Cloud account (note that this is NOT the API key)",
        "type": "string",
        "default": "30295a9a-9ffa-4b5b-8b7d-efa06f3d38a7",
        "regex": "",
        "group_name": "cloud",
        "required": true,
        "secured": false,
        "hidden": false,
        "immutable": false,
        "immutable_after_create": false
    }
]

```

```

},
{
  "name": "ssh_key_name",
  "label": "Name of SSH Key in IBM Cloud for Login",
  "description": "SSH key name in IBM Cloud to be used for SSH logins",
  "type": "string",
  "default": "ibmikey",
  "regex": "",
  "group_name": "cloud",
  "required": true,
  "secured": false,
  "hidden": false,
  "immutable": false,
  "immutable_after_create": false
},
{
  "name": "image_name",
  "label": "Name of Image to Deploy",
  "description": "Name of the image from which the virtual machine should be deployed",
  "type": "string",
  "default": "ibmi73vm",
  "regex": "",
  "group_name": "image",
  "required": true,
  "secured": false,
  "hidden": false,
  "immutable": false,
  "immutable_after_create": false
},
{
  "name": "vm_name",
  "label": "Name of the virtual machine",
  "description": "Name of the virtual machine",
  "type": "string",
  "default": "ITSOVS",
  "regex": "",
  "group_name": "vm",
  "required": true,
  "secured": false,
  "hidden": false,
  "immutable": false,
  "immutable_after_create": false
},
{
  "name": "memory",
  "label": "Amount of Memory (GB)",
  "description": "Amount of memory (GB) to be allocated to the virtual machine",
  "type": "string",
  "default": "16",
  "regex": "",
  "group_name": "vm",
  "required": true,
  "secured": false,

```

```

        "hidden": false,
        "immutable": false,
        "immutable_after_create": false
    },
    {
        "name": "processors",
        "label": "Number of Virtual Processors",
        "description": "Number of virtual processors to allocate to the virtual machine",
        "type": "string",
        "default": "0.25",
        "regex": "",
        "group_name": "vm",
        "required": true,
        "secured": false,
        "hidden": false,
        "immutable": false,
        "immutable_after_create": false
    },
    {
        "name": "proc_type",
        "label": "Processor Type",
        "description": "Processor type for the LPAR - shared/dedicated",
        "type": "string",
        "default": "shared",
        "regex": "",
        "group_name": "vm",
        "required": true,
        "secured": false,
        "hidden": false,
        "immutable": false,
        "immutable_after_create": false,
        "options": [
            {
                "value": "shared",
                "label": "Shared",
                "default": "true"
            },
            {
                "value": "dedicated",
                "label": "Dedicated"
            }
        ]
    },
    {
        "name": "system_type",
        "label": "System on which to Deploy",
        "description": "Type of system on which the VM should be created - s922/e880",
        "type": "string",
        "default": "s922",
        "regex": "",
        "group_name": "system",
        "required": true,

```

```

    "secured": false,
    "hidden": false,
    "immutable": false,
    "immutable_after_create": false
},
{
    "name": "replicants",
    "label": "Number of virtual machines to Deploy",
    "description": "Number of virtual machine instances to deploy",
    "type": "string",
    "default": "1",
    "regex": "",
    "group_name": "vm",
    "required": true,
    "secured": false,
    "hidden": false,
    "immutable": false,
    "immutable_after_create": false
},
{
    "name": "networkname",
    "label": "Name of the Network",
    "description": "Name of the Network",
    "type": "string",
    "default": "CMN",
    "regex": "",
    "group_name": "net",
    "required": true,
    "secured": false,
    "hidden": false,
    "immutable": false,
    "immutable_after_create": false
}
],
"template_output_params": [
    {
        "name": "vm-ip",
        "label": "Virtual machine IP Address",
        "description": "null",
        "group_name": "",
        "secured": false,
        "hidden": false,
        "shortname": "",
        "type": "string"
    }
]
}

```

Important: In Example A-18 on page 169 at the variables block, if you want to enter a value from the user interface at IBM Cloud Pak for Multicloud Management that was declared before the template was deployed in context with the cloud zone, you must add lines to camvariables.json that match the ones in variables.tf, for example:

```
{  
    "name": "ibmcloud_zone",  
    "label": "IBM Cloud Zone (e.g., us-south or eu-de-2)",  
    "description": "Indicate the zone within the region to connect to (only  
needed for multi-zone regions--e.g., eu-de-2)",  
    "type": "string",  
    "default": "us-east",  
    "regex": "",  
    "group_name": "cloud",  
    "required": true,  
    "secured": false,  
    "hidden": false,  
    "immutable": false,  
    "immutable_after_create": false  
},
```

Note: The sixth module or block is version.tf, and has the following structure:

```
terraform { required_version = ">= 0.12" }
```

This line implies that for this template and syntax, you must use Terraform V12 or later. We use Terraform V0.12.21 and IBM Cloud Terraform Provider V1.12.0. If you do not use the correct versions, the deployment of the template can fail. For more information about IBM Cloud Terraform Provider, see this [GitHub repository](#).

Deploying a virtual machine into IBM Power Systems Virtual Server by using IBM Cloud Pak for Multicloud Management

This section is part of an overall scenario to create an IBM i VM in the IBM Power Systems Virtual Server by using IBM Cloud Pak for Multicloud Management.

Cloud connection

To enable IBM Cloud Pak for Multicloud Management to communicate with IBM Cloud, first you must create a cloud connection to IBM Cloud. The cloud connections must be configured only once per IBM Cloud account to which you deploy. Hence, for cloud connection, specify the credentials and configuration to deploy workloads to a cloud provider.

To create a cloud connection to IBM Cloud, complete the following steps:

1. Log in to IBM Cloud Pak for Multicloud Management, and select **Automate infrastructure** → **Terraform automation**.
2. In the Terraform & Service Automation window, select **Manage** → **Cloud Connections**, as shown in Figure A-42 on page 175. Then, click **Create Connection**.

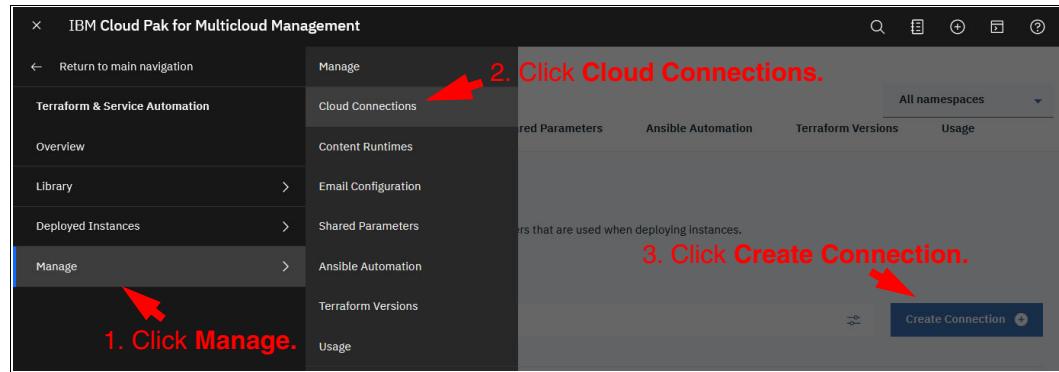


Figure A-42 IBM Cloud Pak for Multicloud Management: Creating a cloud connection for IBM Cloud

3. You are prompted about the cloud connection parameters to establish communication with the instance of IBM Power Systems Virtual Server in IBM Cloud.

Underpinned by your IBM Cloud setups, you must complete the configuration window to suit your environment. In our example, we generate the IBM Cloud API key that is shown in Figure A-40 on page 159.

Enter and select the following Cloud Connections parameters, as shown in Figure A-43:

- Assign Access: **Make Connection Globally Accessible (accessible to all users)**.
- Cloud Provider: IBM.
- Connection Name: ITSO-IBMCLOUD.
- Connection description: Enter a description if you want.
- IBM Cloud API Key: Paste in the key.

1. Select a Namespace
Assign Access
 Make Connection Globally Accessible (available to all users)
 Make Connection part of a namespace

2. Select a Cloud Provider
Cloud Provider
IBM

3. Enter Connection Name
Connection Name
ITSO-IBMCLOUD

4. Connection Description
Connection Description
IBM Cloud connections against IBM Cloud Pak for Multicloud Management

5. Configure Connection
How to configure an IBM cloud
Provide the IBM Cloud API Key to provision and manage resources.
IBM Cloud API Key

6. Paste IBM Cloud API Key
.....

Create

Figure A-43 IBM Cloud Pak for Multicloud Management: Enter details for the cloud connection

Click **Create**, and a confirmation window opens, where you save the connection. Then, click **Save**. The cloud connection is created and ready to be used.

Importing a template

Another component that is need to deploy an image within IBM Cloud Pak for Multicloud Management is a template. For this example, we use Terraform templates to define how we want new VMs deployed through IBM Power Systems Virtual Server in IBM Cloud. To import a template source from GitHub, complete the following steps:

1. In IBM Cloud Pak for Multicloud Management, select **Library** → **Templates**, and click **Import Template**.
2. The Import Template dialog box opens (Figure A-44 on page 177). Complete the entries in the window as follows:
 - Assign Access: Select **Make Template Globally Accessible (available to all users)**.
 - Import Template source: **GitHub**.
 - GitHub Repository URL: Paste in the correct URL.
 - GitHub Access token: Paste in the correct token.

Click **Import**.

Import Template

Assign Access **1. Select Assign Access.**

Make Template Globally Accessible (available to all users)
 Make Template part of a namespace

* Select a namespace

Import template source **2. Choose Import template source.**

GitHub

* GitHub Repository URL **3. Paste a GitHub Repository URL.**

https://github.com/IBMRibooks/SG248486-Red-Hat-OpenShift

GitHub Access Token **4. Paste a GitHub Access Token.** [Learn how?](#)

.....

5. Click Import.

Cancel Import

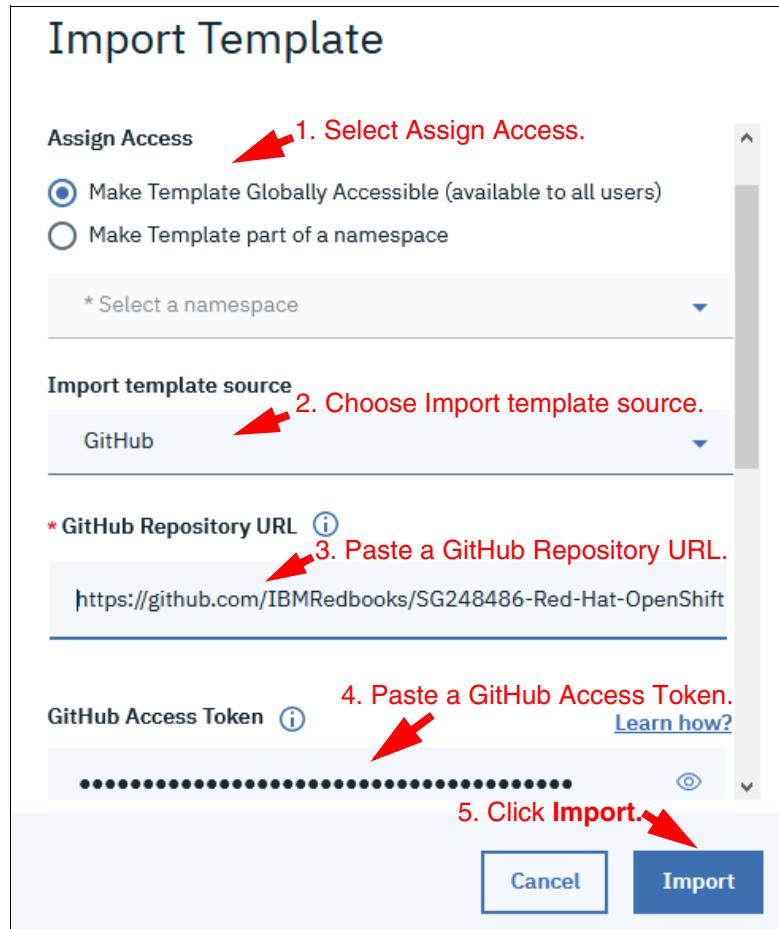


Figure A-44 IBM Cloud Pak for Multicloud Management: Enter details for import

- The template loads, and a new template window opens (Figure A-45). Complete the entries in the window:
 - Name.
 - Short Description.
 - Long Description.
 - Cloud Provider: **IBM Cloud**.

Click **Save**.

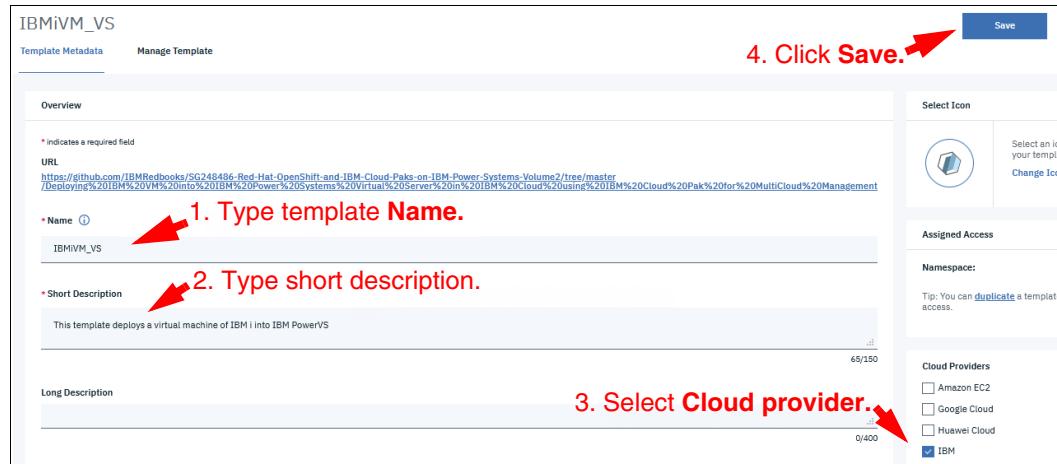


Figure A-45 IBM Cloud Pak for Multicloud Management: Template metadata for IBM Cloud

Deploying a virtual machine

To deploy the virtual machine, complete the following steps:

- In IBM Cloud Pak for Multicloud Management, select **Library** → **Templates**. The window that is shown in Figure A-46 opens. To deploy an IBM i VM by using a template, search for the template that you want to use, which in this example is IBMiVM_VS, and select it. Then, click **Deploy** at the lower right of the window.

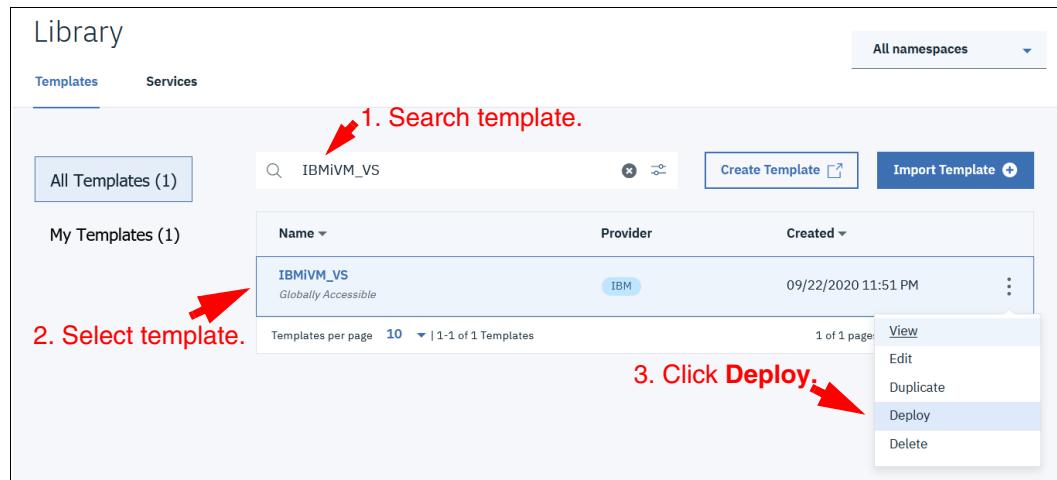


Figure A-46 IBM Cloud Pak for Multicloud Management: Select a template to deploy

- Enter the details of the deployment. All the details are in one window, but for illustrative purposes, we divide the window into three parts (Figure A-47 on page 179, Figure A-48 on page 179, and Figure A-49 on page 180).

Figure A-47 shows the following settings:

- Namespace
- Instance Name
- IBM Cloud Connection
- Terraform Version

The screenshot shows a configuration interface with four numbered steps:

1. Select a Namespace: A dropdown menu with 'default' selected. A red arrow points to the input field with the text '1. Select Namespace.'
2. Enter Instance Name: An input field containing 'IBMiVM_VS'. A red arrow points to the input field with the text '2. Enter Instance Name.'
3. Select a Cloud Connection: A dropdown menu with 'ITSO-IBMCLOUD' selected. A red arrow points to the input field with the text '3. Choose IBM Cloud Connection to use.'
4. Select a Terraform Version: A dropdown menu with '0.12.21' selected. A red arrow points to the input field with the text '4. Choose Terraform Version to use.'

Figure A-47 IBM Cloud Pak for Multicloud Management: Details part 1

Figure A-48 shows the following settings:

- IBM Cloud API Key
- IBM Cloud Region
- Power Systems Virtual Server Instance ID
- Name of SSH Key in IBM Cloud for login

The screenshot shows a configuration interface with four numbered steps:

5. IBM Cloud Configuration: An input field containing a masked API key. A red arrow points to the input field with the text '5. Paste API Key'.
6. Type IBM Cloud Region: An input field containing 'us-east'. A red arrow points to the input field with the text '6. Type IBM Cloud Region'.
7. Paste IBM Power Systems Virtual Server instance ID: An input field containing a long UUID. A red arrow points to the input field with the text '7. Paste IBM Power Systems Virtual Server instance ID'.
8. Enter a name of ssh key: An input field containing 'ibmikey'. A red arrow points to the input field with the text '8. Enter a name of ssh key'.

Figure A-48 IBM Cloud Pak for Multicloud Management: Details part 2

Figure A-49 shows the following settings:

- Name of Image to Deploy
- System on which to deploy (You can select between **s922** or **e880**.)
- Name of the virtual machine
- Amount of memory (GB)
- Number of Virtual Processors
- Processor Type
- Number of virtual machines to Deploy
- Name of the Network

Click **Deploy**.

Alert: Due to limited capacity on the IBM Power System E880 on IBM Cloud, the maximum availability for cores and memory is limited. Typically, the maximum capacity is 143 cores and 8,029 GB of memory.

Alert: Due to limited capacity on IBM Power System S922 on IBM Cloud, the maximum availability for cores and memory is limited. Typically, the maximum capacity is 15 cores and 956 GB of memory.

6. Boot image
* Name of Image to Deploy ⓘ **9. Enter Name of Image to Deploy.**
ibm73vm

7. System Configuration
* System on which to Deploy ⓘ **10. Type system on which to deploy.**
s922

8. Virtual Machine Configuration
* Name of the virtual machine ⓘ **11. Type name of the VM.**
ITSOVS

* Amount of Memory (GB) ⓘ **12. Insert amount of memory.**
16

* Number of Virtual Processors ⓘ **13. Insert number of virtual processors to use.**
0.25

* Processor Type ⓘ **14. Select processor type.**
Shared

* Number of virtual machines to Deploy ⓘ **15. Insert number of VMs to deploy.**
1

9. Network Configuration
* Name of the Network ⓘ **16. Enter name for the network to use.**
CMN

17. Click Deploy.

Cancel Deploy

Figure A-49 IBM Cloud Pak for Multicloud Management: Details Part 3

After the deployment starts, it takes a few minutes for the status to change from In Progress to Active. If during the procedure errors are encountered, they are displayed in the log file window, as shown in Example A-19 on page 181. Figure A-50 on page 181 shows a successful deployment and other relevant details about the deployment.

The screenshot shows the 'Deployed Instances' page in the IBM Cloud Pak for Multicloud Management. At the top, an instance named 'IBMiVM_VS' is listed with a green dot indicating it is 'Ready'. Below the instance name, there are three tabs: 'Overview' (selected), 'Modify', and 'Log File'. A red arrow points from the text 'VM status is ready to be use.' to the 'Ready' status. Another red arrow points from the text 'Deployment log file.' to the 'Log File' tab. In the 'Template File' section, a blue button labeled 'View Git URL' is highlighted with a red arrow and the text 'View Git URL used.'. In the 'Instance Details' section, a table shows various parameters: NAMESPACE (default), TEMPLATE (Terraform), TEMPLATE VERSION (master), CLOUD (IBM), IBM CONNECTION (ITSO-IBMCLOUD), CREATION TIME (09/23/2020 9:54 AM), and OUTPUT DATA TYPE (content_template_output). A red arrow points from the text 'Cloud connection.' to the 'CLOUD' field. In the 'Activity' section, there are three entries: 'Apply Complete' (09/23/2020 9:57 AM) with a 'View Log' link, 'Plan Complete' (09/23/2020 9:56 AM) with a 'View Log' link, and a third entry that is partially obscured by a red arrow. A red arrow also points from the text 'Deployment status of the VM.' to the 'Activity' section. In the 'Resource Details' section, a table lists four resources: 'data.ibm_pi_image.PowerVS_images', 'ibm_pi_network.power_networks[0]', 'data.ibm_pi_public_network.dsnetw...', and 'ibm_pi_instance.PowerVS_instance'. Each resource has a 'Console' URL, an IP Address (all show '--'), a Created date (09/23/2020 9:54AM), and a State (all show three dots). A red arrow points from the text 'IBM Power Systems Virtual Server resource details created.' to the first resource in the table.

Figure A-50 IBM Cloud Pak for Multicloud Management: State details after deploying the VM into IBM Power Systems Virtual Server

Example A-19 shows a sample log file of a deployed instance. You can view the latest log output and get detailed information by using the plan and apply actions.

Example A-19 Sample log file of a successful VM deployment into IBM Power Systems Virtual Server

```
Wed Sep 23 2020 12:57:38 GMT+0000 (UTC) Running terraform init ...
```

```
Initializing provider plugins...
```

```
Terraform has been successfully initialized!
```

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

```
Wed Sep 23 2020 12:57:41 GMT+0000 (UTC) Running terraform apply ...
```

```
ibm_pi_network.power_networks[0]: Creating...
```

```
ibm_pi_network.power_networks[0]: Still creating... [10s elapsed]
```

```
ibm_pi_network.power_networks[0]: Creation complete after 16s
[id=30295a9a-9ffa-4b5b-8b7d-efa06f3d38a7/a547a262-19d7-42f6-9933-a80a61527d82]
```

```
data.ibm_pi_public_network.dsnetwork: Refreshing state...
```

```
ibm_pi_instance.PowerVS_instance: Creating...
ibm_pi_instance.PowerVS_instance: Still creating... [10s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [20s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [30s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [40s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [50s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [1m0s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [1m10s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [1m20s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [1m30s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [1m40s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [1m50s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [2m0s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [2m10s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [2m20s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [2m30s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [2m40s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [2m50s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [3m0s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [3m10s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [3m20s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [3m30s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [3m40s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [3m50s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [4m0s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [4m10s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [4m20s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [4m30s elapsed]
```

```
ibm_pi_instance.PowerVS_instance: Still creating... [4m40s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [4m50s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [5m0s elapsed]
ibm_pi_instance.PowerVS_instance: Still creating... [5m10s elapsed]

ibm_pi_instance.PowerVS_instance: Creation complete after 5m14s
[id=30295a9a-9ffa-4b5b-8b7d-efa06f3d38a7/fe5e96c1-95a8-4e0c-bf05-47c310c0a802]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

The state of your infrastructure has been saved to the path below. This state is required to modify and destroy your infrastructure, so keep it safe. To inspect the complete state use the `terraform show` command.

State path: `terraform.tfstate`

Outputs:

```
vm-ip = [
  {
    "external_ip" = "169.47.177.24"
    "ip" = "192.168.4.24"
    "macaddress" = "fa:6e:15:b1:57:20"
    "network_id" = "a547a262-19d7-42f6-9933-a80a61527d82"
    "network_name" = "CMN"
    "type" = "fixed"
  },
]
```

Figure A-51 shows the state of the VM after a successful deployment in the IBM Power System Virtual Server in the IBM Cloud. All the details of your deployment are shown too.

VM status is ready to be use.

Open Console.

Server details.

IPL status progress.

Load source disk.

IP for 5250 emulator.

Name	IP address	External IP	Gateway	MAC address	VLAN ID	CIDR
CMN	192.168.4.24	169.47.177.24	192.168.4.17	fa:6e:15:b1:57:20	2033	192.168.4.16/28

Figure A-51 IBM Cloud: A view of the state of the deployed virtual machine and details

The IBM i VM that is named ITSOVS can be accessed by using the external IP address to establish a 5250 emulation. However, you can also open a console in IBM Power Systems Virtual Server, as shown in Figure A-52 on page 185.

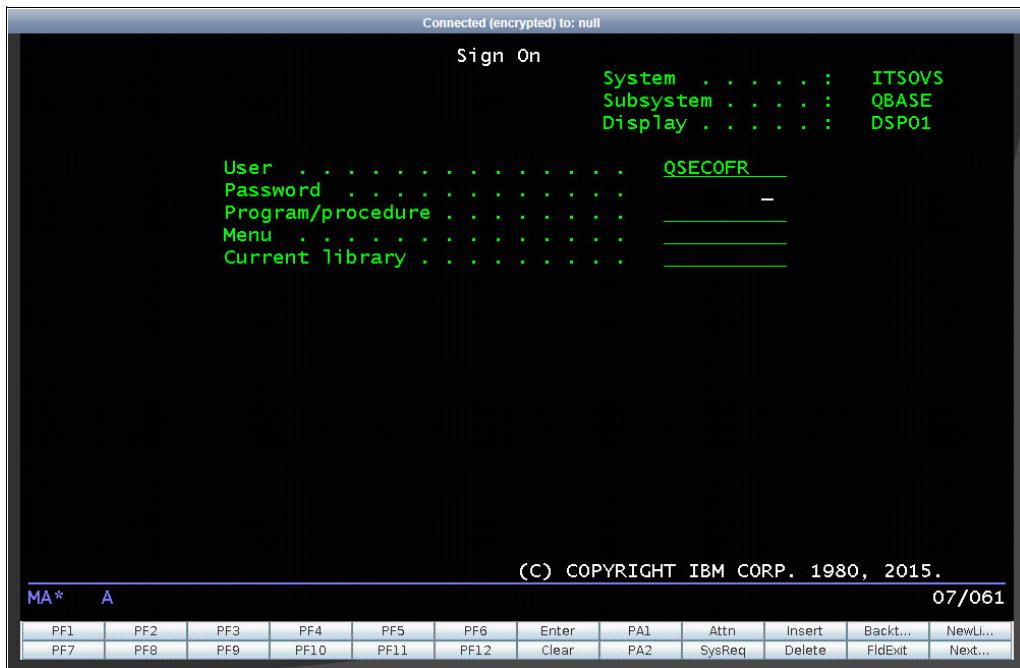


Figure A-52 IBM Cloud: IBM i virtual machine console

After deploying the VM, you can delete the VM when it is no longer required.

Note: The monthly billing cycle ends when you delete the VM in IBM Power Systems Virtual Server in IBM Cloud. If you scale up or down your infrastructure in response to workload requirements, your billing follows the timing of the VM provision change. If you stop the VM, the billing process is not stopped. You must delete the VM to stop the billing cycle. For more information, see [Pricing for Power Systems Virtual Servers](#).

You can delete the VM by using IBM Cloud Pak for Multicloud Management.

Summary

Figure A-53 summarizes the examples in this appendix that demonstrated the journey for IBM Power Systems in the hybrid cloud world.

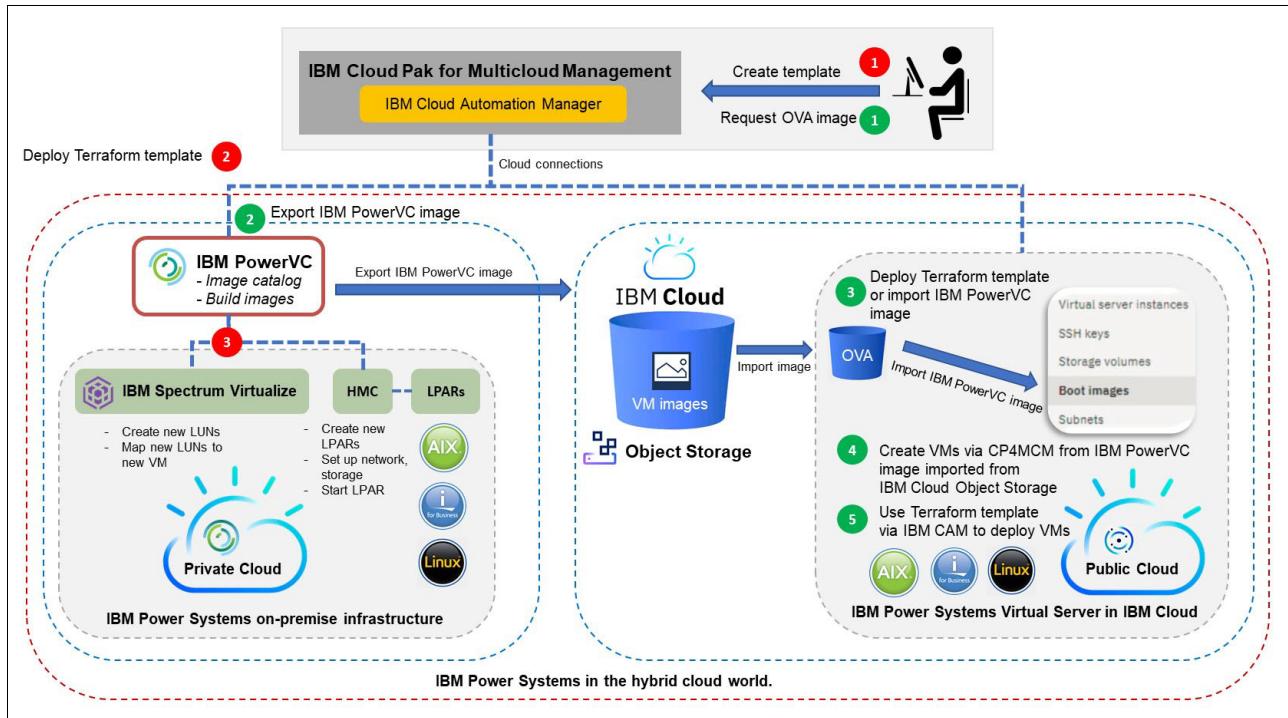


Figure A-53 Summary of deployed samples of IBM Power Systems in the hybrid cloud



B

Red Hat OpenShift and Kubernetes for the developer

This appendix describes how a development team can interact with Red Hat OpenShift to quickly deploy, modify, and manage applications.

Special focus is given to the Red Hat OpenShift Do command-line interface (CLI), also known as **odo**.

This appendix contains the following topics:

- ▶ Red Hat OpenShift from the developer standpoint
- ▶ The odo CLI

Red Hat OpenShift from the developer standpoint

The main objective of DevOps as a practice is to integrate development and operations in a single efficient entity. Modern development and operations tools reflect this ideal by integrating capabilities that are useful for the entire development lifecycle, including development, deployment, and maintenance.

Red Hat OpenShift is designed to reduce the inherent difficulties in the deployment and management of running applications. However, this task is not the only one in which Red Hat OpenShift can help the teams behind the applications.

The `odo` CLI

The Red Hat OpenShift Do (`odo`) CLI tool is an easy way to deploy and manage applications, which does not require advanced knowledge of the underlying mechanisms of the Red Hat OpenShift cluster itself.

The `odo` tool is focused on the development part of DevOps, so it is regarded as a *developer CLI*. The proper use of the tool provides a fast way to test changes in the application code and see them running in Red Hat OpenShift, so it is perfect for developers who want to make small changes to the code and see them running in a real Red Hat OpenShift instance immediately. This tool is useful in the early stages of development while troubleshooting hard problems in the application.

The Red Hat OpenShift Client (`oc`) tool is much more focused on the operations side of the project and it is generally used by infrastructure engineers who monitor, manage, and improve internal aspects of their Red Hat OpenShift instances. Meanwhile, the `odo` tool assumes that the infrastructure is already configured and working correctly although its goal is to provide only direct interaction with the applications running on it.

The terminology of the `odo` tool is tailored to developers. Concepts like project, application, and service are easily understood. This simplified syntax reduces the learning curve for developers who need to run their code in Red Hat OpenShift without having to learn how it all works internally.

Installing `odo`

The `odo` tool is not meant to be installed in the Red Hat OpenShift cluster. Instead, the tool is installed on the workstation of the developer who wants to interact with the Red Hat OpenShift cluster. At the time of writing, there are versions of `odo` that are compatible with Windows, Linux, and MacOS.

Just like its usage, the `odo` tool installation process is easy and straightforward. On Linux and MacOS, only a couple of commands are necessary to set up the tool.

For more information about installation instructions for all supported platforms, see [Installing odo](#).

Creating an application

In this section, a new application is created by using **odo**. There are multiple programming languages and frameworks that are supported by **odo**, including Node.js, Java, Ruby, Perl, PHP, and Python.

First, a *hello world* Python web application is created. Then, the Node.js runtime environment is used to run a similar application, but some changes are applied to it and uploaded to the Red Hat OpenShift instance.

Note: The examples run in a Linux workstation terminal by using **odo** V1.2.5. It is assumed that **odo** and the Git CLI are installed and configured.

The **odo** commands remain the same across operating systems (OSs), but other commands, for example, for creating directories or moving to other paths, might differ in a non Linux workstation.

Creating a Python application

To set up a Python web application that is written by using the Django web framework, complete the following steps:

1. Log in to the cluster by running the following command. Its hostname is `api.crc.testing`, and it is running on port 6443.

```
odo login -u developer -p developer https://api.crc.testing:6443
Connecting to the OpenShift cluster
The server uses a certificate signed by an unknown authority.
You can bypass the certificate check, but any data you send to the server could
be intercepted by others.
Use insecure connections? (y/n): y
Login successful.
You don't have any projects. You can try to create a new project, by running
odo project create <project-name>
```

2. Create a project by running the following command:

```
odo project create django
Project 'django' is ready for use
New project created and now using project: django
```

3. Create a directory for the required files and move them there by running the following command:

```
mkdir django && cd django
```

4. Clone the code from the GitHub OpenShift sample repository by running the following command:

```
git clone https://github.com/openshift/django-ex.git
```

5. Create a component by running the following command:

```
odo create python
Validation
Warning: python is not fully supported by odo, and it is not guaranteed to work
Validating component [12ms]
Use `odo push` command to create the component with source deployed
```

- Push the code to Red Hat OpenShift by running the following command:

```
odo push
Validation
Checking component [19ms]

Configuration changes
Retrieving component data [30ms]
Applying configuration [30ms]

Applying URL changes
URLs are synced with the cluster, no changes are required.
```

```
Pushing to component python-django-ex-kioc of type local
Checking file changes for pushing [3ms]
Waiting for component to start [3m]
Syncing files to the component [308ms]
Building component [32s]
Changes successfully pushed to component
```

- Create a URL to access the application by running the following application:

```
odo url create --port 8080
URL python-django-ex-kioc-8080 created for component: python-django-ex-kioc
To apply the URL configuration changes, use `odo push`
```

- List the available URLs by running the following command. This list includes the newly created address.

```
odo url list
```

The URL for the python-django-ex-kioc component is shown in Table B-1.

Table B-1 Components for the application

Item	Value
Name	python-django-ex-kioc-8080
State	Pushed
URL	http://python-django-ex-kioc-8080-app-django.apps-crc.testing
Port	8080
Secure	False

Opening a browser and accessing the URL (<http://python-django-ex-kioc-8080-app-django.apps-crc.testing>) displays the Welcome page that is shown in Figure B-1 on page 191.

Welcome to your Django application on OpenShift

How to use this example application

For instructions on how to use this application with OpenShift, start by reading the [Developer Guide](#).

Deploying code changes

The source code for this application is available to be forked from the [OpenShift GitHub repository](#). You can configure a webhook in your repository to make OpenShift automatically start a build whenever you push your code:

1. From the Web Console homepage, navigate to your project
2. Click on Browse > Builds
3. Click the link with your BuildConfig name
4. Click the Configuration tab
5. Click the "Copy to clipboard" icon to the right of the "GitHub webhook URL" field
6. Navigate to your repository on GitHub and click on repository settings > webhooks > Add webhook
7. Paste your webhook URL provided by OpenShift
8. From the "Content Type" dropdown, select "application/json"
9. Leave the defaults for the remaining fields — that's it!

After you save your webhook, if you refresh your settings page you can see the status of the ping that GitHub sent to OpenShift to verify it can reach the server.

Note: adding a webhook requires your OpenShift server to be reachable from GitHub.

Working in your local Git repository

If you forked the application from the OpenShift GitHub example, you'll need to manually clone the repository to your local system. Copy the application's source code Git URL and then run:

Managing your application

Documentation on how to manage your application from the Web Console or Command Line is available at the [Developer Guide](#).

Web Console

You can use the Web Console to view the state of your application components and launch new builds.

Command Line

With the [OpenShift command line interface \(CLI\)](#), you can create applications and manage projects from a terminal.

Development Resources

- [OpenShift Documentation](#)
- [Openshift Origin GitHub](#)
- [Source To Image GitHub](#)
- [Getting Started with Python on OpenShift](#)
- [Stack Overflow questions for OpenShift](#)
- [Git documentation](#)

Request information

```
Server hostname: python-django-ex-kioc-app-1-bq6rs
Database server: SQLite (/opt/app-root/src/db.sqlite3)
Data persistence warning: You are currently using SQLite. This is fine for development, but your data won't be persisted across application deployments.
Page views: 1
```

Figure B-1 Welcome page for the sample Django application

Creating a Node.js application

Create a project that is dedicated to running a Node.js application by completing the following steps:

1. Create a local directory for the project by running the following command:

```
mkdir nodejs && cd nodejs
```
2. Create the project by running the following command:

```
odo project create project2
Project 'project2' is ready for use
New project created and now using project: project2
```
3. Clone the sample code by running the following command:

```
git clone https://github.com/openshift/nodejs-ex.git
Cloning into 'nodejs-ex'...
remote: Enumerating objects: 653, done.
remote: Total 653 (delta 0), reused 0 (delta 0), pack-reused 653
Receiving objects: 100% (653/653), 260.96 KiB | 0 bytes/s, done.
Resolving deltas: 100% (253/253), done.
```
4. Change to the new directory that contains the code by running the following command:

```
cd nodejs-ex
```
5. Create a Node.js component by running the following command:

```
odo create nodejs
Validation
Validating component [7ms]
Use `odo push` command to create the component with source deployed
```

6. Push the code to Red Hat OpenShift by running the following command:

```
odo push
Validation
Checking component [17ms]

Configuration changes
Initializing component
Creating component [71ms]

Applying URL changes
URLs are synced with the cluster, no changes are required.
```

```
Pushing to component nodejs-nodejs-ex-lzbm of type local
Checking files for pushing [895735ns]
Waiting for component to start [10s]
Syncing files to the component [172ms]
Building component [10s]
Changes successfully pushed to component
```

7. Create a URL for the web application by running the following command:

```
odo url create --port 8080
URL nodejs-nodejs-ex-lzbm-8080 created for component: nodejs-nodejs-ex-lzbm
To apply the URL configuration changes, use `odo push`
```

8. Push the code again by running the following command:

```
odo push
Validation
Checking component [24ms]
```

```
Configuration changes
Retrieving component data [24ms]
Applying configuration [24ms]
```

```
Applying URL changes
URL nodejs-nodejs-ex-lzbm-8080:
http://nodejs-nodejs-ex-lzbm-8080-app-project2.apps-crc.testing created
```

```
Pushing to component nodejs-nodejs-ex-lzbm of type local
Checking file changes for pushing [1ms]
No file changes detected, skipping build. Use the '-f' flag to force the build.
```

9. Using a browser, go to the URL

http://nodejs-nodejs-ex-lzbm-8080-app-project2.apps-crc.testing to see a page similar to Figure B-2 on page 193.

Welcome to your Node.js application on OpenShift

<p>How to use this example application</p> <p>For instructions on how to use this application with OpenShift, start by reading the Developer Guide.</p> <p>Deploying code changes</p> <p>The source code for this application is available to be forked from the OpenShift GitHub repository. You can configure a webhook in your repository to make OpenShift automatically start a build whenever you push your code:</p> <ol style="list-style-type: none"> 1. From the Web Console homepage, navigate to your project 2. Click on Browse > Builds 3. Click the link with your BuildConfig name 4. Click the Configuration tab 5. Click the "Copy to clipboard" icon to the right of the "GitHub webhook URL" field 6. Navigate to your repository on GitHub and click on repository settings > webhooks > Add webhook 7. Paste your webhook URL provided by OpenShift in the "Payload URL" field 8. Change the "Content type" to "application/json" 9. Leave the defaults for the remaining fields — that's it! 	<p>Managing your application</p> <p>Documentation on how to manage your application from the Web Console or Command Line is available at the Developer Guide.</p> <p>Web Console</p> <p>You can use the Web Console to view the state of your application components and launch new builds.</p> <p>Command Line</p> <p>With the OpenShift command line interface (CLI), you can create applications and manage projects from a terminal.</p> <p>Development Resources</p> <ul style="list-style-type: none"> • OpenShift Documentation • Openshift Origin GitHub • Source To Image GitHub • Getting Started with Node.js on OpenShift • Stack Overflow questions for OpenShift • Git documentation
---	--

Figure B-2 Welcome page for the sample Node.js application

Modifying an application

Changing the code in the application demonstrates how easy is to deploy updates to the applications by using **odo**.

In this example, you replace most of the text that is shown in the welcome page with the Red Hat logo by completing the following steps:

1. Using any text editor, edit the `index.html` file that is in the `views` directory.
2. In the `index.html` file, search for `<div class="row">` inside the container section.
3. Replace the entire `div` section with the following HTML code:

```

```

4. Save the file.
5. Push the changes by running **odo push**:

```
odo push
Validation
Checking component [28ms]
```

```
Configuration changes
Retrieving component data [31ms]
Applying configuration [35ms]
```

```
Applying URL changes
URLs are synced with the cluster, no changes are required.
```

```
Pushing to component nodejs-nodejs-ex-lzbm of type local
Checking file changes for pushing [1ms]
Waiting for component to start [10ms]
Syncing files to the component [139ms]
Building component [2s]
Changes successfully pushed to component
```

6. If you refresh the page, you should see the Red Hat logo instead of the boilerplate text from the welcome page, as shown in Figure B-3.

Tip: If you lost or closed the page, you can see the URL again by running `odo url list`.

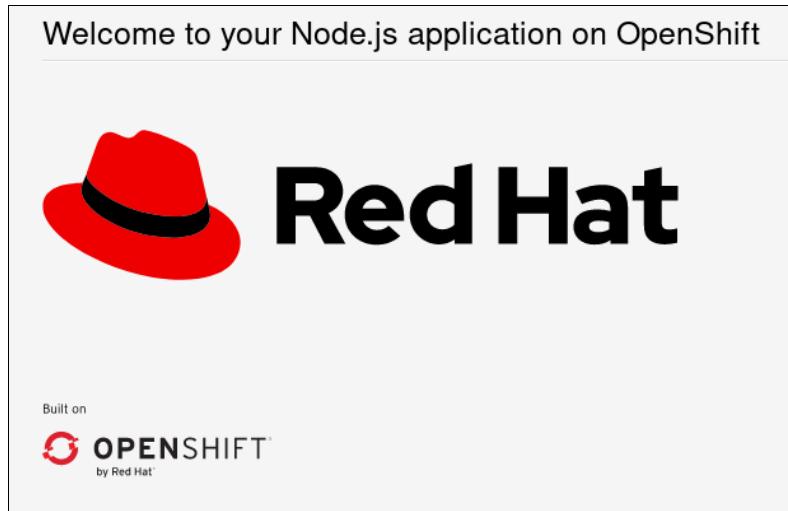


Figure B-3 Modified welcome page

After you see the change in the page, try to upload new updates to the code. For example, you can try to alter the size of the image, use a different image, or create an application.

Cleaning up

After an application is no longer needed, use `odo` to delete it. Entire projects can be removed by using `odo`. In this example, you delete one of the applications that you created, including its corresponding project, by completing the following steps:

1. List the applications in the project by running the following command:

```
odo app list  
The project 'project2' has the following applications:  
NAME  
app
```

2. List the components by running the following command:

```
odo component list  
OpenShift Components:  
APP      NAME          PROJECT      TYPE      SOURCETYPE      STATE  
app      nodejs-nodejs-ex-1zbm    project2    nodejs      local        Pushed  
Delete the application.  
odo app delete app
```

This application has following components that will be deleted

component named nodejs-nodejs-ex-1zbm

This component has following urls that will be deleted with component

```
URL named nodejs-nodejs-ex-lzbm-8080 with host  
nodejs-nodejs-ex-lzbm-8080-app-project2.apps-crc.testing having protocol http  
at port 8080
```

```
No services / could not get services
```

```
Are you sure you want to delete the application: app from project: project2 Yes
```

```
Deleted application: app from project: project2
```

3. List the applications again to see the results by running the following command:

```
odo app list
```

```
There are no applications deployed in the project 'project2'
```

4. Delete the project by running the following command:

```
odo project delete project2
```

```
Are you sure you want to delete project project2 Yes
```

```
Deleted project : project2
```

```
Warning! Projects are deleted from the cluster asynchronously. Odo does its  
best to delete the project. Due to multi-tenant clusters, the project may still  
exist on a different node.
```

Conclusion

The **odo** tool offers a straightforward way to interact with the Red Hat OpenShift and Kubernetes ecosystem. Its simplified approach can be appreciated by developers or by anybody who wants to build or support applications and iterate quickly on code changes.

This appendix shows only a subset of the **odo** capabilities. There is much more that can be accomplished with Red Hat OpenShift and **odo**, including deploying applications that are written in other programming languages. New features are periodically added to **odo**, and it seems that the trend continues, so its usefulness and adoption might increase in the future.



C

Red Hat OpenShift and IBM Cloud Paks entitlements

This appendix describes what you get when you purchase a Red Hat OpenShift subscription and an IBM Cloud Paks license.

This appendix contains the following topics:

- ▶ What you get with a Red Hat OpenShift V4.x subscription
- ▶ What you get with an IBM Cloud Paks license

What you get with a Red Hat OpenShift V4.x subscription

One key difference between Red Hat OpenShift V3.x and V4.x is the change in the base operating system (OS) because it moves from Red Hat Enterprise Linux V7 to Red Hat Enterprise Linux CoreOS. Red Hat OpenShift V4.x comes bundled with Red Hat Enterprise Linux CoreOS, which is a stripped-down version of Red Hat Enterprise Linux that is optimized for container environments. Because Red Hat CoreOS is bundled with and part of the Red Hat OpenShift V4.x subscription, you do not have to account for a Red Hat Enterprise Linux subscription when installing Red Hat OpenShift V4.x on bare metal.

The Red Hat OpenShift V4.x subscription carries its own *core-pair* entitlements to run any Red Hat OpenShift based container, and it includes an entitlement for Red Hat CoreOS too. You still must account for the Red Hat Enterprise Linux host OS subscription if you plan to install or run Red Hat OpenShift V4.x on Kernel-based Virtual Machines (KVMs).

Each Red Hat OpenShift Container Platform subscription provides extra entitlements for Red Hat OpenShift, Red Hat Enterprise Linux, and other Red Hat OpenShift related components. These extra entitlements are included for running either Red Hat OpenShift Container Platform master or infrastructure nodes.

So, a Red Hat OpenShift subscription must be purchased only for the worker nodes.

This section describes two scenarios and what to account for:

- ▶ Running Red Hat OpenShift 4.x on bare metal with no virtual machines (VMs), as shown in Figure C-1, does not account for a Red Hat Enterprise Linux subscription.

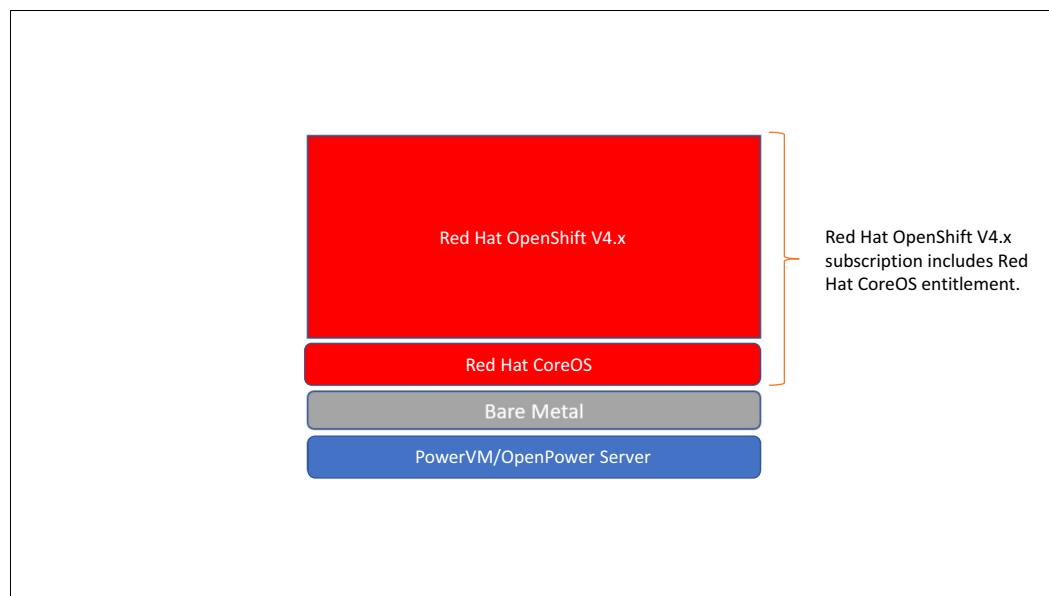


Figure C-1 Red Hat OpenShift V4.x: Bare metal scenario

- ▶ When running Red Hat OpenShift 4.x on KVMs, as shown in Figure C-2 on page 199, you must account for an extra Red Hat Enterprise Linux subscription for the host OS.

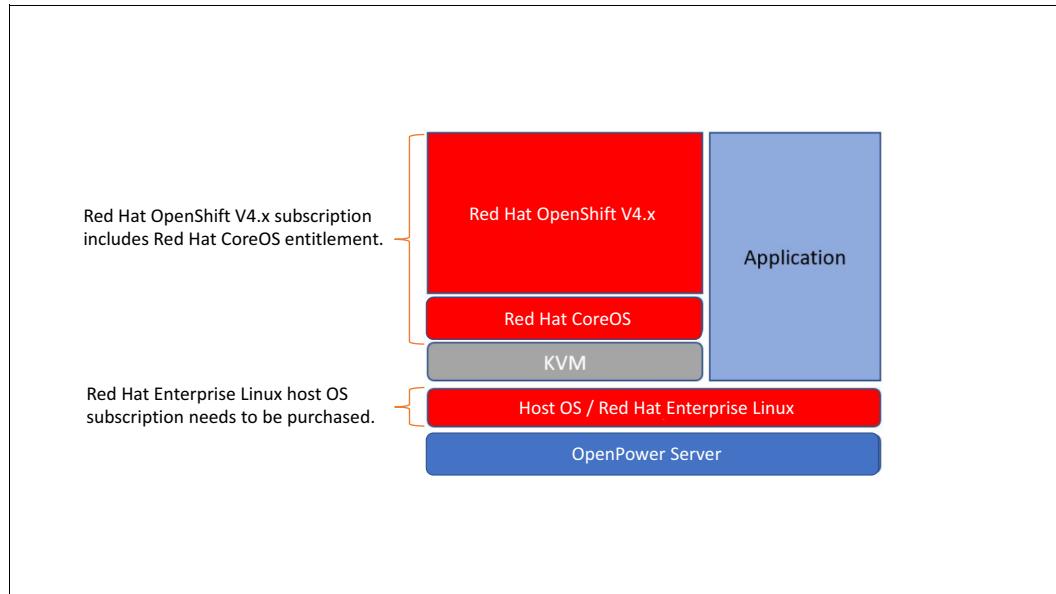


Figure C-2 Red Hat OpenShift V4.x: KVM scenario

Note: Installing Red Hat OpenShift on a KVM hypervisor is supported only for test and development purposes and *not* for a production environment. For more information, see this [GitHub repository](#).

What you get with an IBM Cloud Paks license

IBM Cloud Paks are enterprise-ready, containerized software solutions that give customers an open, faster, and more secure way to move core business applications to any cloud. Each IBM Cloud Pak includes a container platform, containerized IBM middleware and open-source components, and common software services for development and management, on top of a common integration layer that is designed to reduce development time by up to 84% and operational expenses by up to 75%.

IBM Cloud Paks are licensed by Virtual Processor Cores (VPCs). Each IBM Cloud Pak includes a *limited use* license of Red Hat OpenShift V4.x that entitles customers to run the entitled IBM software that is included in the IBM Cloud Pak on the number of VPCs for which the IBM software is entitled. The customer does not have to purchase Red Hat OpenShift V4.x licenses for the nodes on which IBM Cloud Paks are deployed, but they must ensure that these nodes are running only IBM Cloud Paks workloads.

Note: An IBM Cloud Paks license covers the Red Hat OpenShift 4V.x subscription for the master nodes.

Because IBM Cloud Paks includes a Red Hat OpenShift V4.x entitlement, you do not have to account for a Red Hat OpenShift V4.x subscription or Red Hat Enterprise Linux subscription when running bare metal, but you must account for a Red Hat Enterprise Linux host OS subscription if you plan to install and run IBM Cloud Paks on KVMs.

For clarity, this section describes two scenarios and what to account for:

- ▶ Running IBM Cloud Paks on bare metal with no VMs, as shown in Figure C-3, does not account for the Red Hat Enterprise Linux subscription.

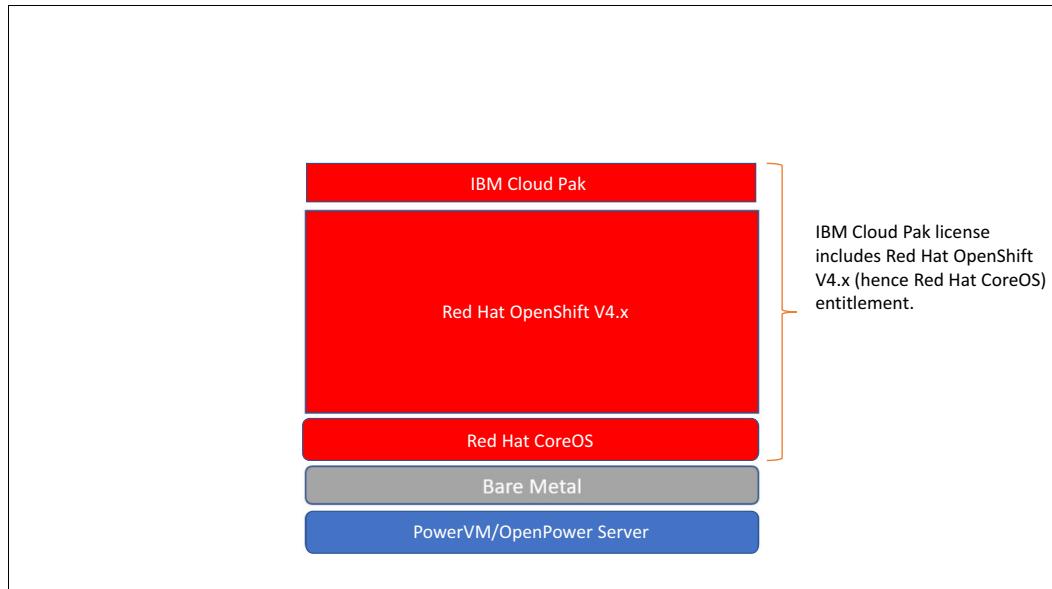


Figure C-3 Red Hat OpenShift V4.x and Cloud Pak: Bare metal scenario

- ▶ While running IBM Cloud Paks on KVMs, as shown in Figure C-4, you must account for more Red Hat Enterprise Linux subscriptions for the host OS.

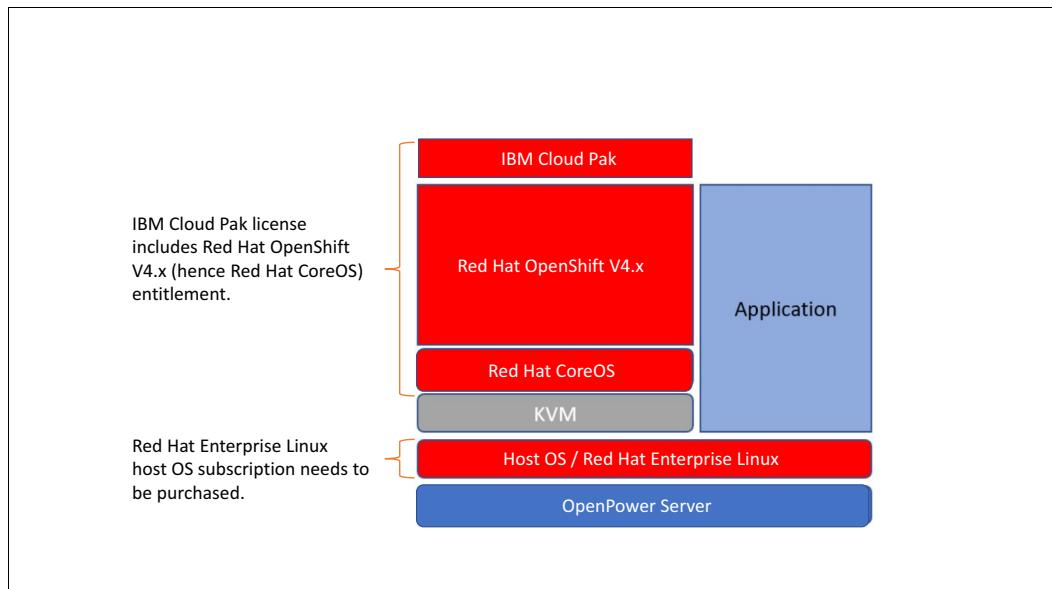


Figure C-4 Red Hat OpenShift V4.x and Cloud Pak: KVM scenarios



D

Virtual I/O Server and Shared Ethernet Adapter configuration

This appendix provides steps to configure the Virtual I/O Server (VIOS) and Shared Ethernet Adapter (SEA).

This appendix contains the following topic:

- ▶ Configuration steps

Configuration steps

When you are installing your environment by using VIOS and SEA, some parameters are required to achieve the necessary network speed between the IBM Power Systems servers. To accomplish this task, complete the following steps:

1. List the Ethernet network devices or other network devices by running the following command:

```
$ lsdev |grep ent
ent0      Available  4-Port Gigabit Ethernet PCI-Express Adapter (e414571614102004)
ent1      Available  4-Port Gigabit Ethernet PCI-Express Adapter (e414571614102004)
ent2      Available  4-Port Gigabit Ethernet PCI-Express Adapter (e414571614102004)
ent3      Available  4-Port Gigabit Ethernet PCI-Express Adapter (e414571614102004)
ent4      Available  PCIe3 2 PORT 10 Gb NIC&ROCE SR/Cu ADAPTER (b315151014101f06)
ent5      Available  PCIe3 2 PORT 10 Gb NIC&ROCE SR/Cu ADAPTER (b315151014101f06)
ent6      Available  Virtual I/O Ethernet Adapter (1-lan)
ent7      Available  Shared Ethernet Adapter
```

2. List the attributes for the SEA to be modified by running the following command:

```
$ lsdev -dev ent7 -attr |grep -i adapter
adapter_reset  no      Reset real adapter on HA takeover          True
ctl_chan       Control Channel adapter for SEA failover        True
pvid_adapter   ent6    Default virtual adapter to use for non-VLAN-tagged packets True
qos_mode       disabled Adapters to use when the primary channel fails True
real_adapter   ent3    Physical adapter associated with the SEA  True
virt_adapters  ent6    List of virtual adapters associated with the SEA (comma separated) True
```

3. List the network interfaces to configure by running the following command:

```
$ ifconfig -a
en7:
flags=1e084863,200080<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT
,64BIT,CHECKSUM_OFFLOAD(ACTIVE),CHAIN>
        inet 9.108.102.146 netmask 0xfffffff800 broadcast 9.108.103.255
                tcp_sendspace 131072 tcp_recvspace 65536 rfc1323 0
lo0:
flags=e08084b,c0<UP,BROADCAST,LOOPBACK,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,
LARGESEND,CHAIN>
        inet 127.0.0.1 netmask 0xff000000 broadcast 127.255.255.255
        inet6 ::1%1/64
                tcp_sendspace 131072 tcp_recvspace 131072 rfc1323 1
```

4. Define the interfaces (IP address, SEA, and physical) to modify their attributes by running the following command:

```
(0) padmin @ rb04vio1: /home/padmin
$ rmdev -dev en7 -ucfg
en7 Defined
```

```
(0) padmin @ rb04vio1: /home/padmin
$ rmdev -dev ent7 -ucfg
ent7 Defined
```

```
(0) padmin @ rb04vio1: /home/padmin
$ rmdev -dev ent3 -ucfg
ent3 Defined
```

5. Change the attributes for the SEA by running the following command:

```
$ chdev -dev ent7 -attr largesend=1 large_receive=yes
ent7 changed
```

6. Change the attributes for the physical interface by running the following command:

```
(130) padmin @ rb04vio1: /home/padmin  
$ chdev -dev ent4 -attr large_receive=yes large_send=yes  
ent4 changed
```

7. Remove the device to define the virtual interface by running the following command:

```
(0) padmin @ rb04vio1: /home/padmin  
$ rmdev -dev ent6 -ucfg  
ent6 Defined
```

8. Change the attributes for the virtual device by running the following command:

```
(0) padmin @ rb04vio1: /home/padmin  
$ chdev -dev ent6 -attr min_buf_tiny=2048 max_buf_tiny=4096 min_buf_small=2048  
max_buf_small=4096 min_buf_medium=512 max_buf_medium=1024 min_buf_large=96  
max_buf_large=256 min_buf_huge=96 max_buf_huge=128  
ent6 changed
```

9. Configure the detected devices by running the following command:

```
(0) padmin @ rb04vio1: /home/padmin  
$ cfgdev
```

10. Put the devices into the Available or Up state by running the following command:

```
(0) padmin @ rb04vio1: /home/padmin  
$ chdev -dev en7 -attr state=up -restoreroute  
en7 changed
```

For more information about SEA attributes, see [Network attributes](#).



E

Where to start with Red Hat OpenShift on IBM Power Systems

This appendix provides guidance about where to find information about planning your deployment of Red Hat OpenShift on Power Systems servers.

This appendix contains the following topics:

- ▶ Red Hat OpenShift on IBM Power Systems support
- ▶ Red Hat support information

Red Hat OpenShift on IBM Power Systems support

Red Hat OpenShift V4.3 is available on IBM Power Systems servers. For more information about the announcement about the availability of Red Hat OpenShift V4.3 for IBM Power Systems, see [OpenShift 4.3 now available on IBM Power Systems](#).

Red Hat support information

The Red Hat holistic approach to services and support puts you and your team in a position to address challenges and find the most success with Red Hat products.

For more information about Red Hat support, see [Red Hat Services & Support](#).

Glossary

application programming interface (API) An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system (OS) or another program.

bare metal machine A dedicated, fully customizable physical server that can be used for virtualization or web hosting.

CI/CD Continuous Delivery/Continuous Integration.

Cloud Native Computing Foundation (CNCF) A consortium to promote cloud-native technologies and principles and provide guidance to the cloud community. Although CNCF is not intended to be a standards body, they do have criteria and a review board before a project can join.

cloud computing A computing platform where users can access applications or computing resources as services from anywhere through their connected devices.

cloud provider An organization that provides cloud computing resources.

cloud-native How an application is built and deployed. A cloud-native application consists of discrete, reusable components that are known as microservices that integrate into any cloud environment.

command-line interface (CLI) A computer interface in which the input and output are text-based.

container orchestration Manages the deployment, placement, and lifecycle of containers. See *Kubernetes*.

container A system construct that allows users to simultaneously run separate logical OS instances. Containers use layers of file systems to minimize image sizes and promote reuse.

containerization A practice of encapsulating or packaging software code (containers) and all its dependencies so that it can run uniformly and consistently on any infrastructure.

continuous delivery (CD) A practice by which you build and deploy your software so that it can be released into production at any time.

continuous integration (CI) A process where developers integrate their code more frequently to identify integration issues earlier when they are easier to fix.

CRI-O An integration point between Kubernetes and container run times that makes pods (groups of containers) work in Kubernetes clusters.

dashboard A user interface component that provides a comprehensive summary of pertinent information from various sources to the user.

DevOps A software methodology that integrates application development and IT operations so that teams can deliver code faster to production and iterate continuously based on market feedback.

Dockerfile A text file that contains instructions to build a Docker image.

hybrid cloud A cloud computing environment that consists of multiple public and private resources.

Identity and Access Management (IAM) The process of controlling access of authorized users to data and applications while helping companies comply with various regulatory requirements.

image A file and its execution parameters that are used within a container run time to create a container. The file consists of a series of layers, which are combined at run time, that are created as the image is built by successive updates.

infrastructure-as-a-service (IaaS) The delivery of a computer infrastructure, including server functions, networking functions, data center functions, and storage functions, as an outsourced service.

instance An entity that consists of resources that are reserved for a particular application or a service.

Internet of Things (IoT) A global network of endpoints that can capture or generate data. For example, a smartphone, smart watch, and back-end server might all communicate with each other, sending data back and forth, or even to other devices within the network.

Kubernetes (k8s or kube) A container orchestration platform for scheduling and automating the deployment, management, and scaling of containerized applications.

load balancer as a service (LBaaS) A service that can distribute traffic among instances in a virtual private cloud.

local cloud A cloud computing environment within the customer's data center. The local cloud is on-premises, which provides improved latency and security.

microservices An application architectural style in which an application is composed of many discrete, network-connected components that are called microservices.

mobile backend as a service (MBaaS) A computing model that connects mobile applications to cloud computing services and provides features, such as user management, push notifications, and integration with social networks.

mobile cloud An infrastructure in which the storage and processing of data for applications is offloaded from a mobile device into the cloud.

multicloud A cloud adoption strategy that embraces a mix of cloud models (public, dedicated, private, and managed) to best meet unique business, application, and workload requirements.

OCI container image A container image that is compliant with the OCI Image Format Specification.

on-premises Software that is installed and run on the local computers of a user or organization.

platform as a service (PaaS) The delivery of a computing platform, including applications, optimized middleware, development tools, and runtime environments, in a cloud-based environment.

pod A group of containers that are running on a Kubernetes cluster. A pod is a unit of work that can be run, which can be a stand-alone application or a set of microservices.

private cloud A cloud computing environment on which access is limited to members of an enterprise and partner networks.

public cloud A cloud computing environment on which access to standardized resources, such as infrastructure, multi-tenant hardware, and services, is available to subscribers on a pay-per-use basis.

registry A public or private repository that contains images that are used to create containers.

software as a service (SaaS) A model of software deployment where software, including business processes, enterprise applications, and collaboration tools, is provided as a service to customers through the cloud.

Related publications

The publications that are listed in this section are considered suitable for a more detailed description of the topics that are covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topics in this document. Some publications that are referenced in this list might be available in softcopy only.

- ▶ *Red Hat OpenShift and IBM Cloud Paks on IBM Power Systems: Volume 1*, SG24-8459
- ▶ *Red Hat OpenShift V4.3 on IBM Power Systems Reference Guide*, REDP-5599

You can search for, view, download, or order these documents and other Redbooks, Redpapers, web docs, drafts, and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ Getting started with IBM Cloud Pak for Multicloud Management
<https://cloud.ibm.com/docs/cloud-pak-multicloud-management?topic=cloud-pak-multicloud-management-getting-started>
- ▶ IBM Redbooks GitHub repository for *Red Hat OpenShift V4.X and IBM Cloud Pak on IBM Power Systems Volume 2*, SG24-8486
<https://github.com/IBMRedbooks/SG248486-Red-Hat-OpenShift-and-IBM-Cloud-Paks-on-IBM-Power-Systems-Volume2.git>
- ▶ IBM Cloud Pak for Multicloud Management V1.3.0
https://www.ibm.com/support/knowledgecenter/en/SSFC4F_1.3.0/kc_welcome_cloud_pak.html
- ▶ IBM Cloud App Management 2019.4.0
https://www.ibm.com/support/knowledgecenter/SS8G7U_19.4.0/com.ibm.app.mgmt.doc/welcome.html
- ▶ IBM Cloud Pak for Data: Common use cases
https://www.ibm.com/support/producthub/icpdata/docs/content/SSQNUZ_current/usecase/use-cases.html
- ▶ Red Hat Customer Portal: Chapter 3., “Disaster Recovery”, of *OpenShift Container Platform*
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.2/html/backup_and_restore/disaster-recovery

- ▶ Understanding cluster logging and OpenShift Container Platform V4.3
<https://docs.openshift.com/container-platform/4.3/logging/cluster-logging.html>
- ▶ Understanding cluster logging and OpenShift Container Platform V4.5
<https://docs.openshift.com/container-platform/4.5/logging/cluster-logging.html>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

Red Hat OpenShift v4.X and IBM Cloud Pak on IBM Power Systems Volume 2

(0.2"spine)
0.17" <-> 0.473"
90<->249 pages



SG24-8486-00

ISBN 0738459569

Printed in U.S.A.

Get connected

