

Red Hat OpenShift Container Platform on IBM Z and LinuxONE

Lydia Parziale

Rakesh Krishnakumar

Shrirang Kulkarni

Li Liyong

Anilkumar Patil

Richard Young



 **Cloud**

IBM Z



IBM Redbooks

Red Hat OpenShift Container Platform on IBM Z and LinuxONE

July 2024

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

First Edition (July 2024)

This edition applies to IBM z/VM 7.3, Red Hat Enterprise Linux 9.1, and Red Hat OpenShift Container Platform 4.13.

This document was created or updated on July 3, 2024.

© Copyright International Business Machines Corporation 2024. 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!	xi
Comments welcome	xi
Stay connected to IBM Redbooks	xi
Chapter 1. Introduction to Red Hat OpenShift Container Platform on IBM Z and LinuxONE	1
1.1 Red Hat OpenShift overview	2
1.2 Red Hat OpenShift on IBM Z and LinuxONE	3
1.2.1 Red Hat OpenShift capabilities on IBM Z and LinuxONE	3
1.2.2 Red Hat OpenShift benefits on IBM Z or LinuxONE	6
1.2.3 Red Hat OpenShift deployment options on IBM Z and LinuxONE	8
Chapter 2. Red Hat OpenShift Container Platform architecture	13
2.1 Red Hat OpenShift components overview	14
2.2 Red Hat OpenShift components and their roles	17
2.2.1 Bootstrap node	17
2.2.2 Control plane	17
2.2.3 Compute Node	18
2.2.4 Bastion node	18
Chapter 3. Implementation architectural considerations	21
3.1 Red Hat OpenShift product deployment requirements	22
3.1.1 Load balancer	22
3.1.2 Network Configuration Management	24
3.1.3 Bastion host	24
3.1.4 Hosting hypervisor environment	25
3.2 Number of hosting logical partitions for a cluster	25
3.3 Deployment architectures used in this paper	26
3.3.1 Single-LPAR deployment configuration	26
3.3.2 Three-LPAR deployment configuration	27
3.4 Storage architecture	28
3.4.1 FCP-attached SCSI and FICON-attached ECKD storage	28
3.4.2 CoreOS node storage	28
3.4.3 RHEL virtual server disk storage	28
3.4.4 Red Hat OpenShift Persistent Storage	29
3.5 Multitenancy with other workloads	31
3.5.1 LPAR level controls in IBM Z and LinuxONE	31
3.5.2 Guest controls in KVM	31
3.5.3 Guest controls in IBM z/VM	32
3.6 Recovery site considerations	32
3.7 IBM Secure Execution	33
3.8 IBM CEX high-performance hardware security modules requirements	33
3.9 FIPS requirements	33
3.10 Multus for a second network interface	34

3.11	Authenticating	34
3.12	Monitoring	35
3.13	Logging	35
Chapter 4. Resource considerations for Red Hat OpenShift		37
4.1	LPAR adjustments and weights	38
4.1.1	General LPAR adjustments	38
4.1.2	IBM z/VM weights	40
4.1.3	Adjustments for Red Hat OpenShift	40
Chapter 5. Red Hat OpenShift deployment topologies on IBM Z		41
5.1	Deployment topology criteria	42
5.1.1	Data gravity	42
5.1.2	Consolidation and TCO Reduction	42
5.1.3	Business continuity	42
5.1.4	Vertical Solutions	43
5.2	IBM z/VM One LPAR cluster implementation	43
5.2.1	Resource planning	43
5.2.2	DNS configuration	44
5.2.3	HAPROXY configuration	45
5.2.4	Ignition files and the HTTP server	47
5.2.5	USER DIRECT and PARM files for OCP nodes	51
5.2.6	Building the Red Hat OpenShift Container Platform cluster	54
5.2.7	Using Ansible playbooks	55
5.3	IBM z/VM three-LPAR cluster implementation	64
5.3.1	Architecture	64
5.3.2	Planning resources	65
5.3.3	HAPROXY configuration	66
5.3.4	USER DIRECT and PARM files for OCP nodes	71
5.3.5	Building the OCP cluster	74
5.4	KVM single hypervisor cluster implementation	75
5.4.1	Architecture	75
5.4.2	Planning resources	76
5.4.3	DNS configuration	79
5.4.4	Ansible controller configuration	82
5.4.5	Load balancer configuration	83
5.4.6	File server for Ansible Playbook	84
5.4.7	Building the OCP cluster through Ansible	85
5.4.8	Validating the deployment	93
5.5	KVM three-LPAR cluster implementation	95
5.5.1	Architecture	95
5.5.2	Hypervisor preparation	99
5.5.3	DNS server configuration	103
5.5.4	DHCP server configuration	106
5.5.5	Highly available load balancer configuration	108
5.5.6	Rapidly creating the guests that provide the support infrastructure	113
5.5.7	Creating the Red Hat OpenShift cluster	121
5.5.8	Completing the initial installation of Red Hat OpenShift 4.12	130
Chapter 6. Best practices and moving forward		137
6.1	Applying best practices	138
6.1.1	CPU entitlement and vCPU number	138
6.1.2	Disabling transparent huge pages	138
6.1.3	Enabling RFS	139

6.1.4 Infrastructure nodes	139
6.1.5 HyperPAV	139
6.1.6 Specific for KVM	140
6.2 Post-installation configurations	141
6.2.1 Defining an identity provider	141
6.2.2 Configuring persistent storage	143
6.2.3 Configuring the NTP Server	145
6.2.4 Disabling kubeadmin	145
6.2.5 Backing up the etcd database	146
6.3 Sample application deployment	146
6.3.1 Application architecture	147
6.3.2 Deployment	147
Appendix A. Additional material	153
Locating the web material	153
Using the web material	153
System requirements for downloading the web material	153
Downloading and extracting the web material	153
Related publications	155
IBM Redbooks	155
Online resources	155
Help from IBM	155

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](https://www.ibm.com/legal/copytrade.shtml) 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 <https://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.

FICON®	IBM Spectrum®	Redbooks (logo)  ®
GDPS®	IBM Z®	z Systems®
IBM®	IBM z Systems®	z/OS®
IBM Cloud®	Parallel Sysplex®	z/VM®
IBM Consulting™	Redbooks®	

The following terms are trademarks of other companies:

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

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

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

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

Preface

Whether you need to run mission-critical workloads, handle massive data volumes or support high-demand applications, Red Hat OpenShift Container Platform on IBM® Z or IBM LinuxONE can give you the tools you need to build, deploy and manage your containerized applications.

Red Hat OpenShift provides enterprise-level support and additional features for large-scale container deployments. It includes advanced security and compliance features, as well as management tools for monitoring and scaling containerized applications.

With advanced security features and flexible deployment options, Red Hat OpenShift on IBM Z or IBM LinuxONE provides scalability, performance, reliability and security.

This IBM Redpaper publication provides a basic understanding of Red Hat OpenShift on IBM Z and LinuxONE, discusses architectural considerations for implementation as well as resource considerations, basic deployment examples and some best practices to move forwards in your own implementations. It has been written for IT architects and IT specialists.

Authors

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

Lydia Parziale is a Project Leader for the IBM Redbooks® team in Poughkeepsie, New York, with domestic and international experience in technology management including software development, project leadership, and strategic planning. Her areas of expertise include business development and database management technologies. Lydia is a PMI certified PMP and an IBM Certified IT Specialist with an MBA in Technology Management and has been employed by IBM for over 30 years in various technology areas.

Rakesh Krishnakumar is working as an IBM zStack Principal zArchitect for the United Kingdom/Ireland region. He has been working with IBM for 15 years with total industry experience spanning 26 years, predominantly in the IBM Z brand. Rakesh has been helping clients to embrace, modernize, and adopt new technology solutions around IBM Z. His primary skills center around customer advocacy, leading development and testing practices involving Linux on IBM Z, as well as providing technical support services for IBM Z clients. Rakesh has been part of various IBM Redbooks residency programs in the past and has co-authored multiple IBM Redbooks publications involving Linux on IBM Z.

Shrirang Kulkarni is a LinuxONE and Cloud Architect who has been with IBM over 18 years working with IBM System Labs as a LinuxONE and Cloud Architect supporting IBM Z® Global System Integrators. He has worked with various clients in over 25 countries worldwide from IBM Dubai as a Lab services consultant for IBM Z in the MEA region. He has achieved “IBM Expert Level IT specialist” and “The Open Group Certified Master IT Specialist” certifications. He co-authored *IBM Redbooks Security for Linux on System z, SG24-7728* and *Implementing, Tuning, and Optimizing Workloads with Red Hat OpenShift on IBM Power, SG24-8537*. He also authored “Bringing Security to Container Environments, Performance Toolkit and Streamline Fintech Data Management With IBM Hyper Protect Services”, which was published in IBM System Magazine. His areas of expertise include Linux on IBM Z, IBM z/VM®, cloud solutions, IBM z/OS® Container Extensions (zCX), Red Hat OpenShift, architecture design and solutions for z/VM and Linux on IBM Z, performance tuning Linux on IBM Z, IBM z/VM, Oracle, IBM Power, and IBM System x.

Li Liyong is a Certified Consulting IT Specialist and open source enthusiast. Liyong is the technical lead within IBM System Expert Labs organization in ASEAN, helping customers in adopting new technologies, such as hybrid cloud solutions on IBM Z and LinuxONE, Red Hat OpenShift Container Platform and Ansible Automation Platform. He has been with IBM for 17 years, and holds a degree in Computer Science. He has written and contributed to several IBM Redbooks publications on Cloud, IBM z/VM, and Linux.

Anilkumar Patil is an Executive Cloud Architect and Solution Thought Leader in Hybrid Cloud Services within IBM Consulting, US. He is a Certified Thought leader in Architect and Solution Consultant with 23 years of IT experience in design, development, architecture and Cloud migration for large and complex projects and deal solutions. His core experience is in IBM Cloud®, Red Hat OpenShift, Amazon Web Services (AWS), Cloud Application Engineering and Migration services. He is Chief Architect and Solution Consultant Leader for various clients in North America within cross industries. Anil is an IBM Redbooks publication author for different Redbooks and technical contributor for various IBM materials and blogs. Anil is employed with IBM for more than 10 years and holds a BE degree in Electronics and Executive MBA in finance and strategy from Rutgers Business School, New Jersey.

Richard Young is an Executive I.T. Specialist in IBM US. He has extensive experience and is a leader within IBM for Linux and Virtualization technologies. He holds degrees in Computer Science and Business from the University of Wisconsin. His most recent focus has been helping clients with large scale consolidation of Linux on to IBM Z and LinuxONE platforms.

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

Robert Haimowitz and Wade Wallace
IBM Redbooks, Poughkeepsie Center

Tom Ambrosio and Bill Lamastro
IBM CPO

Gerald Hosch and Holger Wolf
IBM

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 papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:
ibm.com/redbooks
- ▶ Send your comments in an email to:
redbooks@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, IBM Redbooks
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on LinkedIn:
<https://www.linkedin.com/groups/2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/subscribe>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<https://www.redbooks.ibm.com/rss.html>



Introduction to Red Hat OpenShift Container Platform on IBM Z and LinuxONE

This chapter provides an overview of Red Hat OpenShift Container Platform on IBM Z and LinuxONE. It highlights the unique capabilities and benefits that Red Hat OpenShift brings to IBM Z and LinuxONE, as well as the potential synergies between the two platforms.

1.1 Red Hat OpenShift overview

Red Hat OpenShift is a cloud-native application platform powered by containers and Kubernetes. It offers standardized workflows, support for multiple environments, continuous integrations, and release management, allowing developers to consistently deploy applications across multiple environments.

Enterprises are using microservices and containers to build applications faster by adopting greenfield projects, modernization initiatives, or cloud-native application development. Red Hat OpenShift empowers developers, DevOps, and Site Reliability Engineers (SREs) to quickly build, deploy, run and manage applications anywhere, securely and at scale.

Red Hat OpenShift is a self-managed application platform that includes Red Hat Enterprise Linux® CoreOS, Kubernetes, over-the-air updates, container runtime, networking, ingress, monitoring, logging, container registry, authentication, and authorization solutions. The platform provides supplementary tooling around the complete life cycle of applications, from building and continuous integration/continuous delivery (CI/CD) to monitoring and logging. These components are tested together for unified operations on a complete Kubernetes platform

Figure 1-1 presents an overview of the Red Hat OpenShift architecture and capabilities.

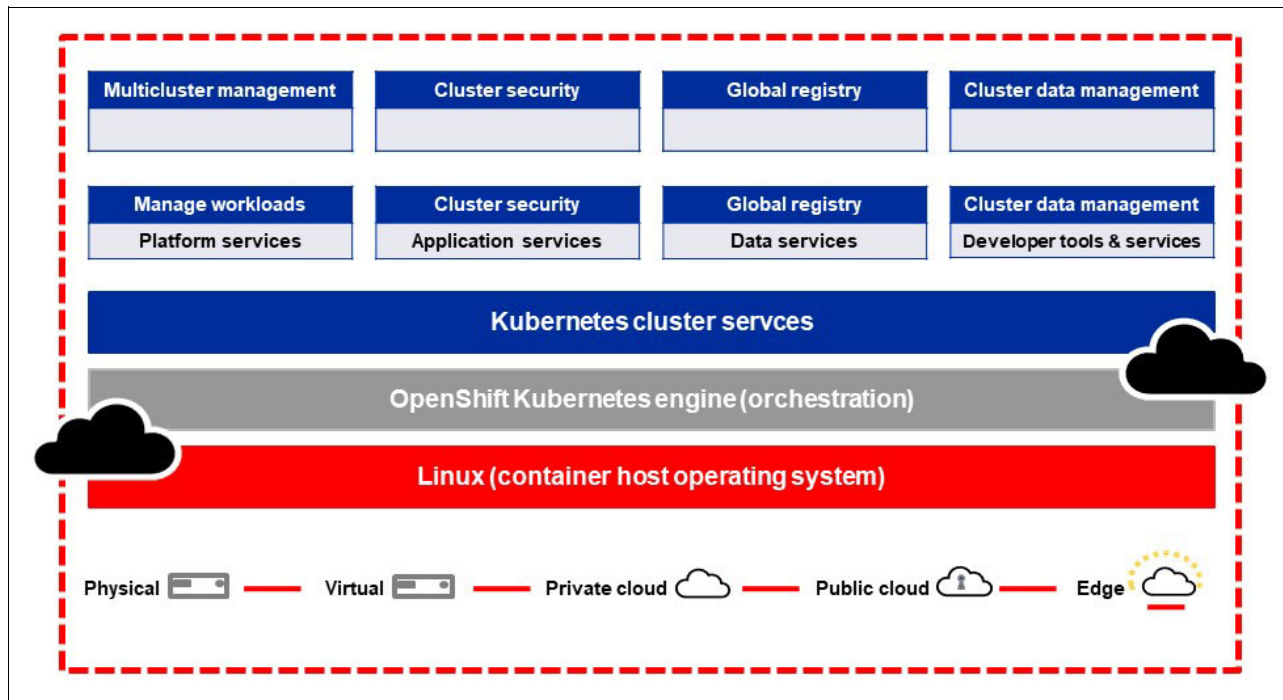


Figure 1-1 Red Hat OpenShift architectural and capabilities overview

1.2 Red Hat OpenShift on IBM Z and LinuxONE

Organizations face the challenge of delivering extraordinary customer experiences by developing new applications, while modernizing existing applications to speed up their cloud-native journey. For developers and IT operations teams, this requires flexibility and agility to develop and deploy applications across multiple infrastructures, from on-premises to public cloud. Red Hat OpenShift on IBM Z and LinuxONE empowers organizations to accelerate transformation with greater flexibility and agility through integrated tooling and a security-focused and resilient foundation for cloud-native development.

Red Hat OpenShift is a trusted Kubernetes enterprise platform that supports modern, hybrid-cloud application development and provides a consistent foundation for applications anywhere, across physical, virtual, private, and public cloud environments. Red Hat OpenShift and IBM Cloud Paks helps teams to develop, deploy and orchestrate cloud-native applications consistently, while taking advantage of the security, reliability, scalability, and reduced carbon footprint of the IBM Z and LinuxONE infrastructure.

1.2.1 Red Hat OpenShift capabilities on IBM Z and LinuxONE

The Red Hat OpenShift Container Platform subscription includes a set of developer and operations services and tools enabled for IBM Z and LinuxONE. The proven platform includes services that empower developers to code with speed and agility, as well as services providing flexibility and efficiency for SREs and operations teams.

Some of these capabilities include the following:

- ▶ Red Hat OpenShift Service Mesh

Red Hat OpenShift Service Mesh offers a unified approach to interconnecting, governing, and monitoring microservices-based applications, making it easier to manage and secure communication between services as their complexity increases. Based on the open source Istio project, Red Hat OpenShift Service Mesh helps developers increase productivity by integrating communications policies without changing application code or integrating language-specific libraries. For more information, see [Red Hat OpenShift Service Mesh](#).

- ▶ Red Hat OpenShift Pipelines

Red Hat OpenShift Pipelines is a cloud-native CI/CD solution based on the open source Tekton project. It provides a Kubernetes-native CI/CD framework to design and run pipelines and is designed to run each step of the CI/CD pipeline in its own container, allowing each step to scale independently to meet the demands of the pipeline. For more information, see [Cloud-native CI/CD on Red Hat OpenShift](#).

- ▶ Red Hat OpenShift Serverless

A serverless cloud computing model providing developers with a modern, cloud-native app dev stack for hybrid clouds. Serverless lets developers focus on their code without worrying about the infrastructure.

Red Hat OpenShift Serverless is a serverless platform based on the open source Knative project, offering an enterprise-grade solution for developing and deploying cloud-native, source-centric applications across hybrid and multi-cloud environments. By utilizing Red Hat OpenShift Serverless, organizations can leverage the portability and consistency provided by Knative, enabling developers to create and manage a series of Custom Resource Definitions (CRDs) and associated controllers in Kubernetes for their applications.

For more information, see [“Why Choose Red Hat OpenShift Serverless?”](#).

- ▶ Red Hat OpenShift Do

Red Hat OpenShift Do (odo) is an open source command-line interface (CLI) tool designed to facilitate the writing and deployment of applications on Red Hat OpenShift and Kubernetes. By offering a user-friendly and intuitive interface, odo enables developers to focus on their code and application logic, without needing to navigate through complex configuration files or manage infrastructure.

For more information, see [OpenShift Do developer CLI \(odo\)](#).

- ▶ Red Hat OpenShift GitOps

Red Hat OpenShift GitOps is built from the open source Argo CD project and lets IT teams implement GitOps workflows for cluster configuration and application delivery for more speed, security, and scalability software development. It provides a declarative way of continuous deployment (CD) workflows integrating into an application development platform.

For more information, see [Red Hat OpenShift GitOps](#).

- ▶ Red Hat OpenShift Dev Spaces

Red Hat Dev Spaces is a cloud-native, container-based in-browser integrated development environment (IDE) for rapid application development that uses Kubernetes and containers to provide developers and other IT team members with a consistent, protected, and zero-configuration development environment on Red Hat OpenShift.

For more information, see [Red Hat OpenShift Dev Spaces](#).

Additional Red Hat products are available to provide benefits for Red Hat OpenShift clusters on IBM Z and LinuxONE that are not included in the Red Hat OpenShift subscription. These products include the following:

- ▶ Red Hat Advanced Cluster Management for Kubernetes

Red Hat Advanced Cluster Management for Kubernetes provides comprehensive management, visibility, and control over your entire Kubernetes cluster and application lifecycle. This solution enhances security and compliance across various data centers and public cloud environments, ensuring a seamless experience for your organization. Red Hat Advanced Cluster Management offers a hybrid cloud management platform, addressing common challenges faced by administrators and SREs in managing various environments, including multiple data centers, private clouds, and public clouds, all running Kubernetes clusters.

For more information, see [Red Hat Advanced Cluster Management for Kubernetes](#).

- ▶ Red Hat Advanced Cluster Security for Kubernetes Support

Red Hat Advanced Cluster Security for Kubernetes Support (ACS) offers visibility into the security of your cluster, vulnerability management, and security compliance through auditing, network segmentation awareness and configuration, security risk profiling, security-related configuration management, threat detection, and incident response. In addition, Red Hat ACS grants an ability to pull the actions from that tooling deep into the application code development process through APIs. These security features represent the primary work any developer or administrator faces as they work across a range of environments, including multiple data centers, private clouds, or public clouds that run Kubernetes clusters.

For more information on the Red Hat Advanced Cluster Security, see [Red Hat Advanced Cluster Security for Kubernetes](#).

- ▶ Red Hat Runtimes

Red Hat Runtimes is a collection of enterprise-grade, production-ready runtimes and frameworks that offers developers access to established and emerging technologies. By incorporating industry standards such as OpenJDK, JBoss Enterprise Application Platform, JBoss Web Server, Red Hat Data Grid, and Quarkus, Runtimes enables developers to boost their productivity and reduce time to market. Also, the suite includes popular cloud-native runtimes like vert.x and Spring Boot, fostering the development of modern, cloud-based applications.

For more information, see [Accelerate app dev and delivery with Red Hat Runtimes](#).

- ▶ Red Hat AMQ Streams

Red Hat AMQ Streams is a massively scalable, distributed, and high-performance data streaming platform based on the Apache Kafka project that offers a distributed backbone for microservices and other applications to share data with high throughput and low latency.

For more information, see [Streams for Apache Kafka](#).

- ▶ Red Hat 3scale API Management

Red Hat 3scale API Management is an infrastructure platform that allows users to share, secure, distribute, control, and monetize APIs.

For more information, see [Red Hat 3scale API Management](#).

- ▶ Red Hat Fuse

Red Hat Fuse is an integration platform that provides agile integrations solutions. The API-centric, container-based architecture decouples services so they can be created, extended, and deployed independently.

For more information on the Red Hat Fuse, see [“What is Red Hat Application Foundations?”](#).

1.2.2 Red Hat OpenShift benefits on IBM Z or LinuxONE

In this section we describe some of the benefits of using Red Hat OpenShift on IBM Z or LinuxONE.

Security-rich and resilient foundation

IBM Z and LinuxONE deliver robust data protection through IBM Crypto Express (CEX) adapters, which meet stringent Federal Information Processing Standard (FIPS) 140-2 Level 4 security standards and incorporate quantum-safe cryptography. These systems offer Confidential Computing within a hardware-based attested Trusted Execution Environment (TEE), ensuring maximum data privacy and regulatory compliance. The integrated HSMS enable efficient and productive management of cryptographic operations while maintaining the highest level of security.

Red Hat OpenShift applications can benefit from the IBM CEX adapters using the Kubernetes device plug-in for IBM CEX adapters, which can be downloaded from the Red Hat certified container catalog.

A TEE that can isolate and protect machine state and memory information is provided through IBM Secure Execution. Red Hat provides a unique image for Secure Execution since Red Hat OpenShift 4.13.

The IBM Z and LinuxONE servers help to avoid or recover from failures to minimize business disruptions, realized through component reliability, redundancy and features that assist in providing fault avoidance and tolerance, as well as permitting concurrent maintenance and repair.

The unified OpenShift container security and IBM Z and LinuxONE cryptographic hardware form a distinguished, security-focused solution. This powerful combination delivers comprehensive protection for your applications and data, ensuring a secure and compliant environment.

Flexibility and scalability

As organizations modernize existing applications to cloud-native architectures, it is essential to have the flexibility to manage and deploy the entire application portfolio across different infrastructures to scale. IBM Z and LinuxONE provide organizations the possibility to scale on a single system with the ability to nondisruptively add capacity on demand and grow processing with minimal impact on energy usage, floor space, and staffing. Alongside the IBM z/VM® hypervisor, Kernel-based Virtual Machine (KVM) as included with RHEL, are the supported virtualization options on IBM Z and LinuxONE for Red Hat OpenShift.

Together with IBM Z and LinuxONE, organizations can scale on a single system with the ability to add capacity on demand and grow processing with minimal impact on energy usage, floor space, and staffing. Teams can take advantage of the high flexibility through dynamic resource sharing and reconfiguration and continue to deliver excellent customer experiences with ultra low latency and large volume data serving and transaction processing.

Sustainability

Running Red Hat OpenShift on a centralized infrastructure such as IBM Z and LinuxONE can contribute to fewer greenhouse gas emissions and a more environmentally sustainable IT environment. IBM Z and LinuxONE are designed to make a powerful improvement in sustainability by decreasing electricity consumption, reducing the number of standing servers, and enabling high compute and resource utilization.

IBM Z and LinuxONE are in a long line of machines that are designed for system and data center energy efficiency with differentiated architectural advantages, including on-chip compression, high per-core performance, and encryption that is designed to sustain 90% utilization along with new embedded on-chip AI acceleration to seamlessly integrate real-time AI insights into business-critical transactions.

The biggest opportunity for energy savings with IBM Z and LinuxONE come through workload modernization and consolidation of distributed x86 systems. Many enterprises cannot easily grow their data centers, but the vertical scalability of IBM Z and LinuxONE can address this problem while reducing your power usage.

Colocation efficiency and economic advantages

The colocation of Red Hat OpenShift apps side by side with workloads running on IBM z/OS or Linux provide the unique opportunity to integrate and modernize without disruption on one system, thereby benefiting in throughput, latency, and operational efficiency. With the immense capacity of IBM Z and LinuxONE servers, all environments and workloads can be expanded at the same time without the need to add an additional server.

By deploying Red Hat OpenShift on IBM Z and LinuxONE, organizations can indeed unlock significant economic advantages. The platform's security-rich nature, coupled with IBM Z and LinuxONE's unmatched resiliency, scalability, flexibility, and low environmental impact, contributes to a more cost-effective solution. Furthermore, the colocation benefits and superior quality of service offered by IBM Z and LinuxONE can lead to further cost savings and improved operational efficiency. Therefore, running Red Hat OpenShift on IBM Z and LinuxONE can provide a strong economic case for businesses.

1.2.3 Red Hat OpenShift deployment options on IBM Z and LinuxONE

IBM Z provides two options to run Red Hat OpenShift:

- ▶ Red Hat OpenShift can be run inside an IBM z/OS address space called IBM zCX Foundation for Red Hat OpenShift (IBM zCX for Red Hat OpenShift).
- ▶ Red Hat OpenShift can be run in virtual machines (VMs) based on IBM z/VM or KVM.

Both Red Hat OpenShift environments can run in parallel, as in parallel with z/OS and Linux environments on a single IBM Z server, while Red Hat OpenShift in virtual machines can run in parallel to Linux environments on LinuxONE servers.

Figure 1-2 provides an example overview on the deployment options.

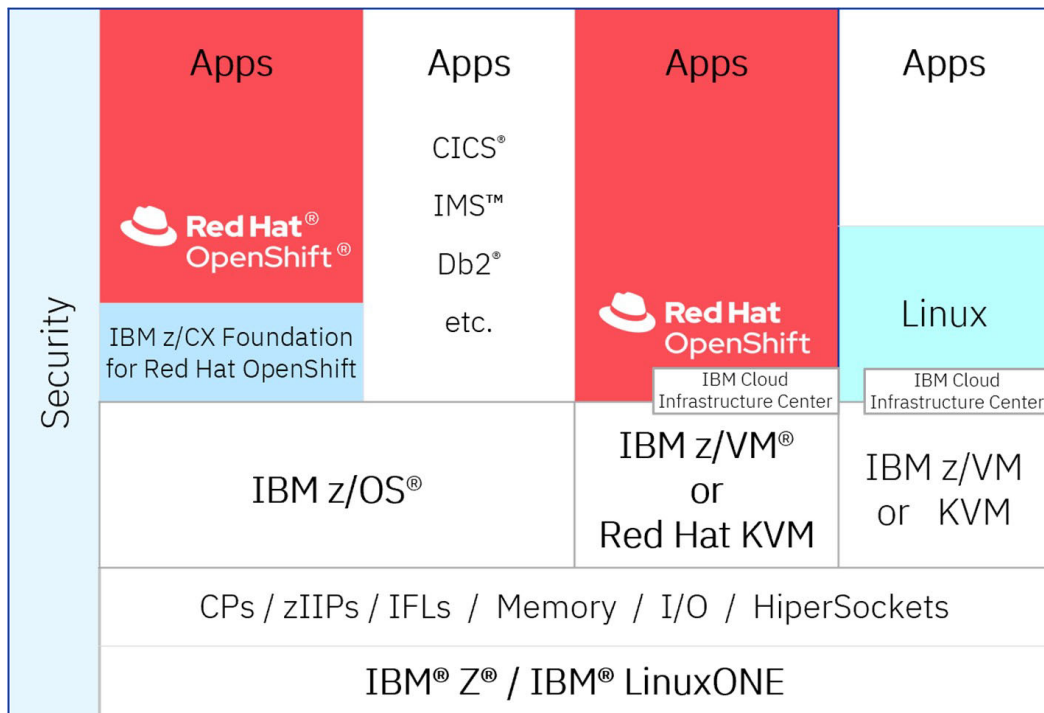


Figure 1-2 Red Hat OpenShift deployment options on IBM Z and LinuxONE

IBM zCX for Red Hat OpenShift

IBM zCX for Red Hat OpenShift provides customers with an option to deploy containerized Linux for IBM Z applications on Red Hat OpenShift on z/OS. This extends the capabilities clients have with the IBM zCX for Red Hat OpenShift running Docker containers with the addition of a Red Hat OpenShift orchestrated containerized environment guaranteeing the same quality of services of traditional z/OS applications.

IBM zCX for Red Hat OpenShift architecture

IBM zCX for Red Hat OpenShift supports a user provisioned infrastructure deployment where the system administrator is responsible for managing firewall and DNS configurations. This architecture is illustrated in Figure 1-3.

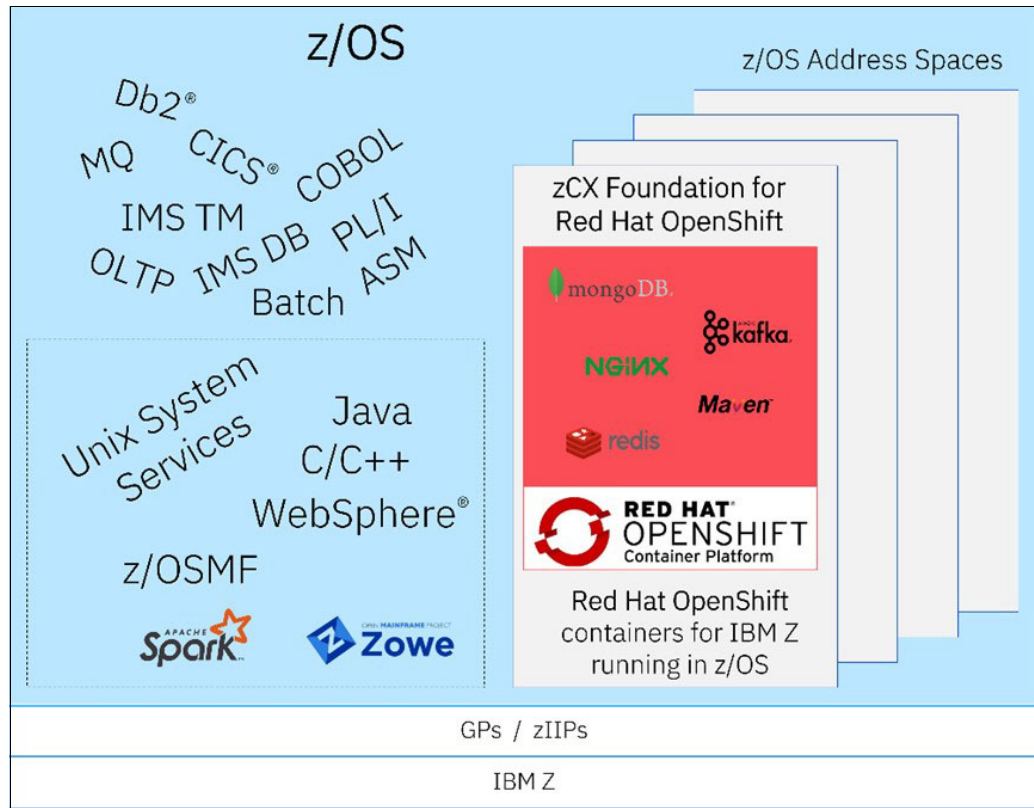


Figure 1-3 IBM z/OS environment with IBM zCX Foundation for Red Hat OpenShift

IBM zCX for Red Hat OpenShift benefits

Some of the benefits of IBM zCX for Red Hat OpenShift include the following.

- ▶ Workload modernization:
 - Enable existing or new IBM z/OS applications to use services that were previously unavailable.
 - Access a large ecosystem of open source and container for Linux on IBM Z workloads, colocated on the IBM z/OS platform with no porting required.
- ▶ IBM Z Quality of Service:
 - Easily scale resources up or down as needed, ensuring optimal performance and efficient use of resources.
 - Achieve fully automated fail-over capability, ensuring high availability and data integrity, with IBM GDPS® offerings.
 - Leverage FIPS 140-2 Level 4 IBM CEX cards to encrypt data at rest and in transit, enhancing security and regulatory compliance.
 - Integration with IBM z/OS Workload Manager (WLM) to optimize, which manages various resource allocation policies based on business priorities.

- ▶ Operational efficiency:
 - Get more out of existing hardware investments by enabling optimal utilization.
 - By employing IBM Z and LinuxONE certified Red Hat OpenShift, businesses can significantly reduce time to value and minimize effort in their Red Hat OpenShift deployments.
 - IBM zCX for OpenShift empowers businesses to overcome cross-platform cultural and operational challenges, fostering resource efficiency and innovation.

IBM zCX for Red Hat OpenShift is also provisioned and managed by IBM z/OS Management Facility (z/OSMF) workflows in a similar fashion as it is performed for IBM zCX.

For more information, see [zCX Foundation for Red Hat OpenShift 1.1.0](#).

Red Hat OpenShift in virtual machines

Red Hat OpenShift in VMs based on IBM z/VM or KVM provides clients the option to run the Red Hat supported Red Hat OpenShift environment with all Red Hat OpenShift services and tools on IBM Z and LinuxONE. Figure 1-4 illustrates that Red Hat OpenShift environment.

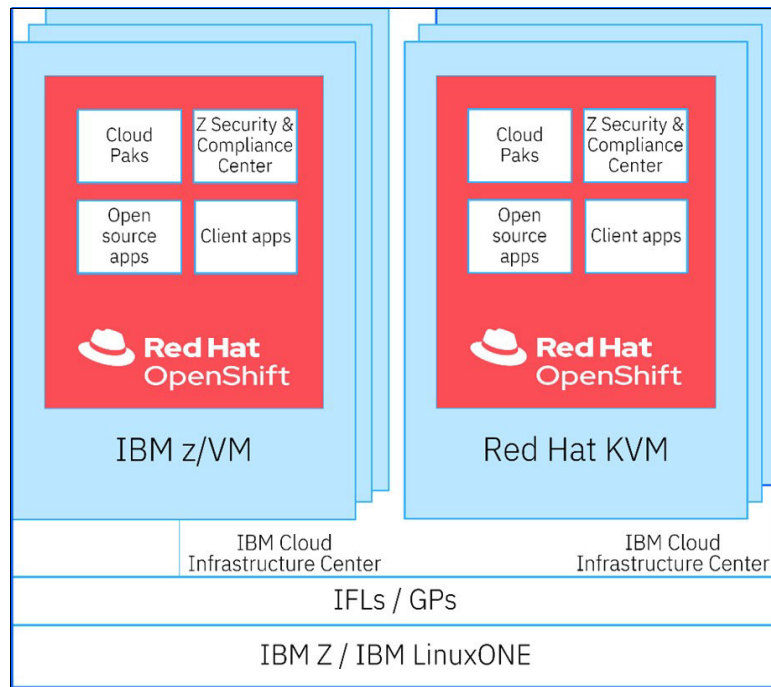


Figure 1-4 Red Hat OpenShift in virtual machines based on IBM z/VM or KVM

Benefits of Red Hat OpenShift in VMs

Some of the benefits of Red Hat OpenShift in VMs include the following:

- ▶ Workload modernization:
 - Enable existing and new containerized applications for IBM Z to benefit from the capabilities of the Red Hat OpenShift environment, including the related services and tools, on IBM Z and LinuxONE.
 - Modernize existing workloads and integrate them with digital services across the hybrid cloud - while keeping the data safe, encrypted and resilient.
- ▶ IBM Z Quality of Service:
 - Benefit from data protection and privacy at scale through confidential computing.
 - Generate up to 100,000 certificates per second using protected keys exploiting IBM CEX adapters.
 - IBM Z, in conjunction with Red Hat OpenShift, IBM GDPS®, IBM DS8000® series storage, and IBM HyperSwap®, offers an unmatched high availability solution, guaranteeing 99.99999% uptime.
- ▶ Operational Efficiency:
 - Scale containers on a single IBM z16 for nondisruptive vertical and horizontal growth to accommodate increases of workloads on demand.
 - By colocating existing and new Linux, IBM z/OS, and Red Hat OpenShift workloads on a single IBM Z server, you can take advantage of the platform's low latency and high throughput.
 - The IBM z16, powered by IBM Z, can seamlessly scale out to support 192 Red Hat OpenShift Container Platform Compute Nodes, allowing you to deploy and manage up to 40,000 NGINX pods.
 - Establish a cloud operation model across the enterprise, including IBM Z and IBM LinuxONE.

For more information, see [Red Hat OpenShift Container Platform 4.14. release notes](#).



Red Hat OpenShift Container Platform architecture

In this chapter, we delve into Red Hat OpenShift features and tools, offering a comprehensive understanding of the OpenShift Container Platform components and their relevance to operations. We also discuss essential considerations for managing and operating Red Hat OpenShift effectively.

2.1 Red Hat OpenShift components overview

Figure 2-1 provides an overview of the components for Red Hat OpenShift.

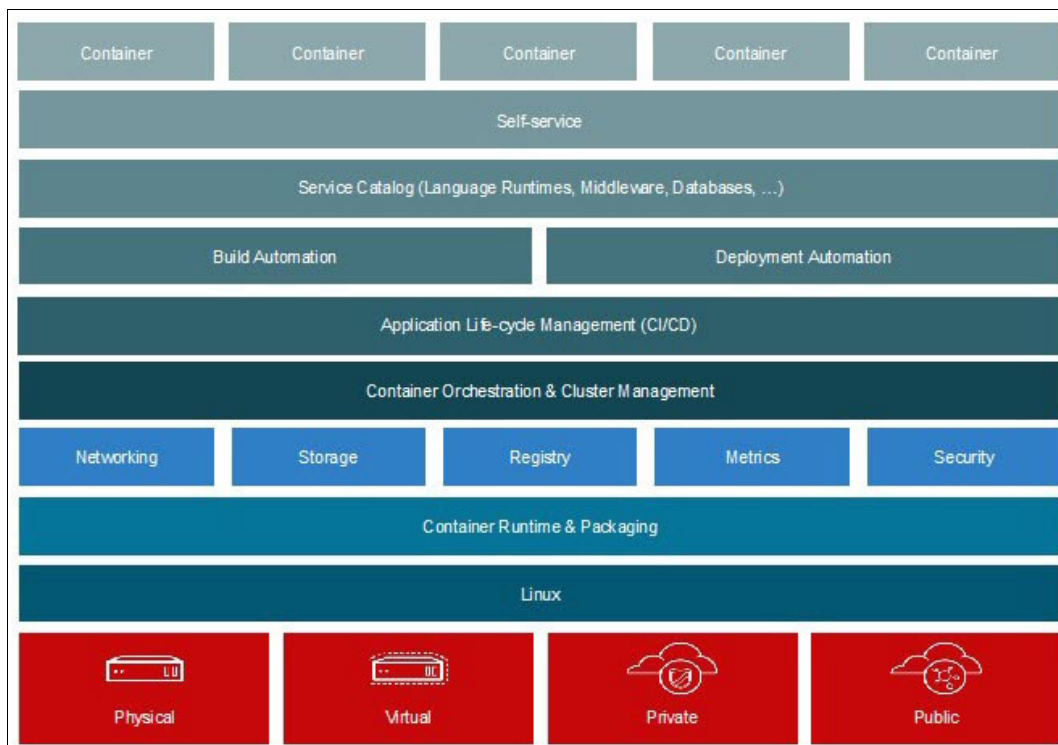


Figure 2-1 Overview of components

Figure 2-2 shows the components of Red Hat Open Container Platform and their relevance to operations.

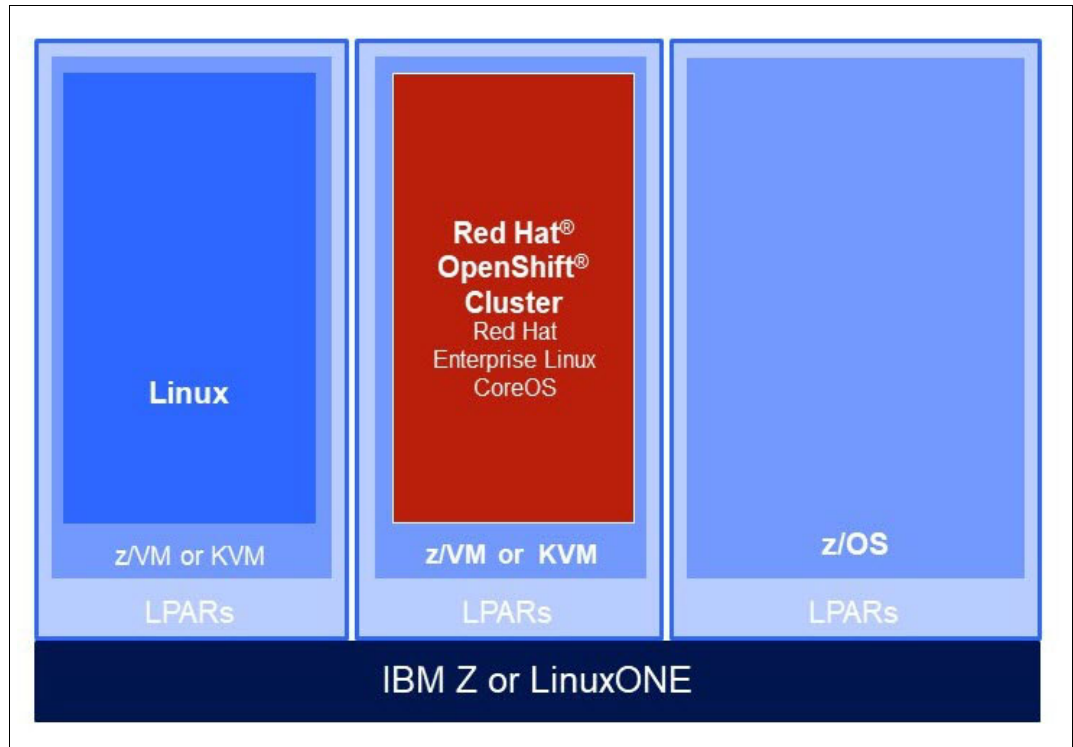


Figure 2-2 Components of Red Hat OpenShift Container Platform

Kubernetes

Kubernetes is a cluster management system for managing containerized applications across multiple hosts, providing mechanisms for deployment, maintenance, and scaling of applications. Kubernetes, a key component of the Red Hat OpenShift Container Platform, plays a pivotal role in streamlining container management. By automating the deployment, networking, scaling, and availability of containerized workloads and services, Kubernetes enables businesses to efficiently manage and orchestrate their container environments, ensuring optimal performance and resource utilization.

Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform is an open source, enterprise-grade Kubernetes platform for building, deploying, and managing containerized applications. It extends Kubernetes with additional features and tooling to simplify deployment, management, and scalability of containerized applications.

Red Hat Enterprise Linux CoreOS

Red Hat Enterprise Linux CoreOS is a lightweight, secure operating system based on RHEL. It serves as the foundation for Red Hat OpenShift Container Platform, providing a stable and secure environment for running containerized workloads. Specifically, Red Hat Enterprise Linux CoreOS is the only supported operating system for Red Hat OpenShift Container Platform control plane nodes and compute nodes.

This unique relationship between Red Hat Enterprise Linux CoreOS and Red Hat OpenShift ensures a cohesive and integrated solution for managing and deploying containerized applications, enabling organizations to streamline their application development and deployment processes.

Docker

Docker is a platform that allows developers to easily deploy their applications in containers to run on Linux. Docker allows users to package an application with all of its dependencies into a standardized unit for software development. Containers do not have high overhead and hence enable more effective usage of the underlying system and resources. Containers also provide full control over resources, giving your infrastructure improved efficiency, which can result in better utilization of your computer resources.

Security

Red Hat OpenShift Container Platform provides various security features and best practices, including Role-Based Access Control (RBAC), container image scanning, and vulnerability management. Security is a top concern in containerized environments. Red Hat OpenShift provides strong encryption controls to protect sensitive data, including platform secrets and application configuration data. Red Hat OpenShift optionally uses FIPS 140-2 Level 1 compliant encryption modules. For more information, see [How Red Hat OpenShift enables container security](#).

Networking

Red Hat OpenShift Container Platform provides advanced networking features. Understanding networking concepts such as service discovery, load balancing, and network policies is essential for managing and securing container communications. Red Hat OpenShift Container Platform uses software-defined networking (SDN). It provides a unified cluster network that enables communication between pods across the cluster.

Operators

Red Hat OpenShift Operators automate the creation, configuration, and management of instances of Kubernetes-native applications. Understanding how to develop, install, and maintain Operators is important for efficient operations. For more information, see [“What are Red Hat OpenShift Operators?”](#).

Registry

Red Hat OpenShift Container Platform includes a container registry, such as Red Hat Quay, where container images are stored and distributed. The registry plays a crucial role in the container life cycle, and operations personnel need to be familiar with image management, security, and distribution processes. For more information, see [Red Hat Quay](#).

Monitoring and Logging

Red Hat OpenShift Container Platform offers built-in monitoring and logging capabilities that leverage popular tools like Prometheus and Elasticsearch. Red Hat OpenShift Logging components include a collector deployed to each node in the Red Hat OpenShift Container Platform cluster that collects all node and container logs and writes them to a log store. Familiarity with these tools and their integration with Red Hat OpenShift Container Platform is important for identifying issues, troubleshooting, and optimizing the platform.

2.2 Red Hat OpenShift components and their roles

In this section, we delve into the specifics of Red Hat OpenShift components and their functions.

2.2.1 Bootstrap node

The bootstrap node is a dedicated node that uses the Red Hat Enterprise Linux CoreOS. The bootstrap node is the main deployment and management server for the Red Hat OpenShift Container Platform cluster. Bootstrap is used as the logon node for the cluster administrators to perform system deployment and management operations. The bootstrap node is a temporary node that is used to create the controller nodes that make up the control plane. The control plane nodes then create the compute nodes. To remotely access an instance, you need to access the bastion instance. Then, through another SSH connection, you are able to access the intended Red Hat OpenShift instance. After the cluster initializes, the bootstrap node can be released if you need to use the resources to perform another installation or increase cluster capacity.

2.2.2 Control plane

The Red Hat OpenShift control plane performs control functions for the whole cluster environment. The control plane is responsible for the creation, scheduling, and management of all objects specific to Red Hat OpenShift. These nodes use Kubernetes services in the background and they are responsible for managing and controlling the entire Red Hat OpenShift cluster while taking advantage of the operating system technology to deploy container workloads on IBM Z.

The control plane components of the system are organized into groups based on the types of resources that they manage. These groups of machines are called machine configuration pools (MCPs). Each MCP manages a set of nodes and its corresponding machine configurations. An example of this is shown in Figure 2-3.

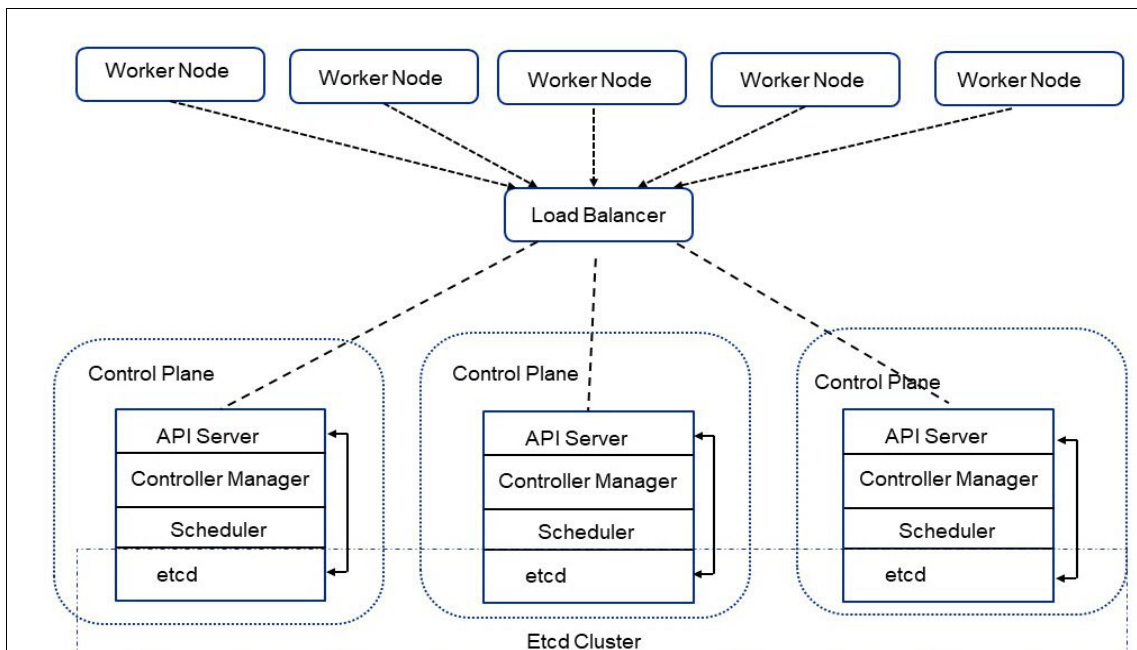


Figure 2-3 Red Hat OpenShift Control Plane

2.2.3 Compute Node

The Red Hat OpenShift compute nodes run containerized applications created and deployed by developers. The compute node hosts use Red Hat Enterprise Linux CoreOS. The Red Hat OpenShift compute node houses the node components, which include the container engine, an open source, community-driven container engine (CRI-O), the node agent Kubelet, and a service proxy (kube-proxy). These components work together to manage and orchestrate containerized workloads and services on the Red Hat OpenShift platform. Compute nodes are responsible for running workloads for the cluster users.

2.2.4 Bastion node

The bastion node is a dedicated node that serves as the key deployment and management server for the Red Hat OpenShift cluster. The bastion node can be any Linux distribution (RHEL, SUSE Linux Enterprise Server, or Ubuntu) and it can be hosted on any hardware platform, including IBM Z, as shown in Figure 2-4.

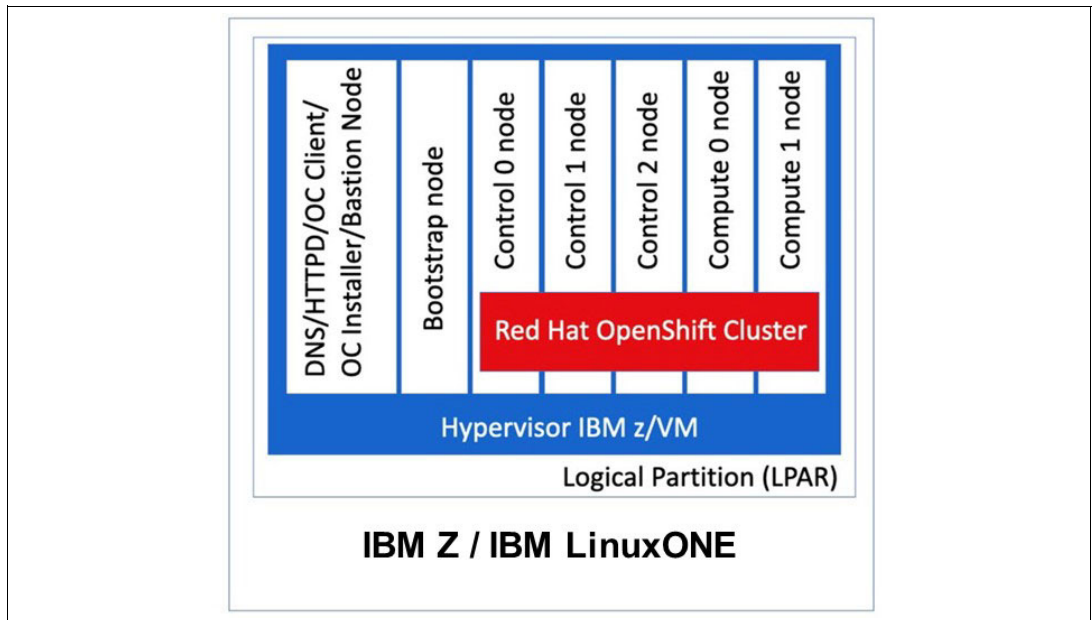


Figure 2-4 Architecture on IBM Z or LinuxONE

Red Hat OpenShift Container Platform is comprised of numerous key components that work together to deliver a robust container application platform. Figure 2-5 shows some of the main components.

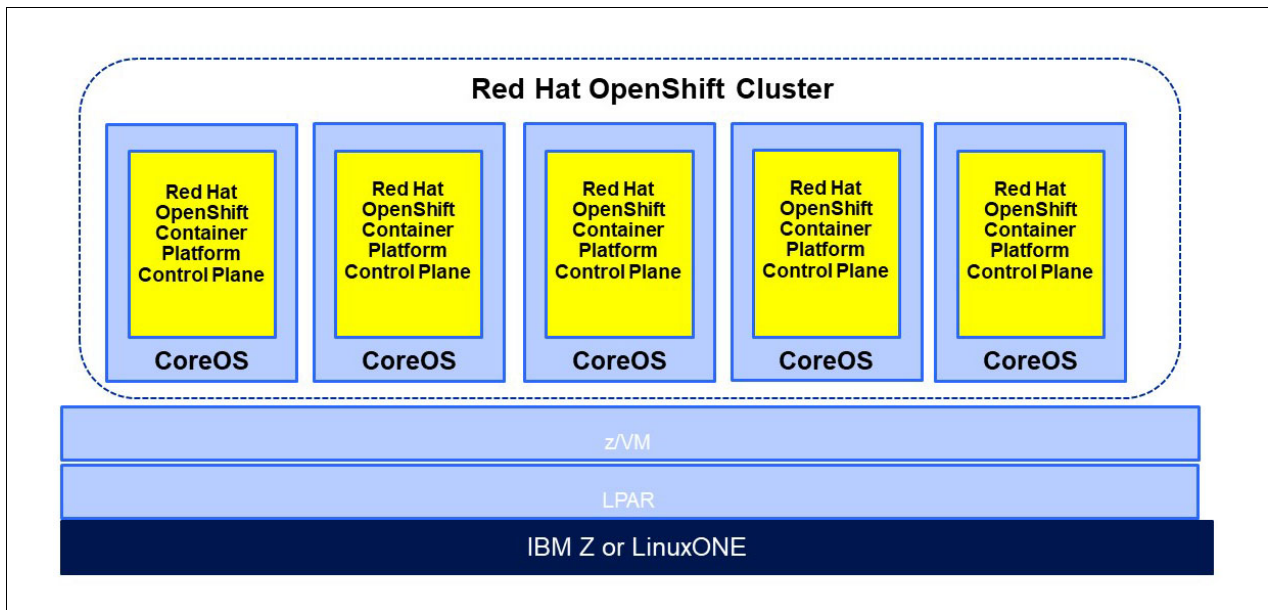


Figure 2-5 Sample of main components

The main components include:

- ▶ *Etc*d is used by Red Hat OpenShift to store cluster state and configuration information. Etc d is a distributed key-value store used for storing and retrieving cluster configuration data.
- ▶ The *control plane node* is responsible for managing and orchestrating the cluster. It includes components such as the API server, controller manager, and scheduler.
- ▶ The *compute node* is a compute machine in the Red Hat OpenShift cluster. It runs containerized applications using the Kubernetes runtime and communicates.
- ▶ *Kubernetes* is an open source container orchestration platform that Red Hat OpenShift is built on. It leverages Kubernetes features for container management, scaling, networking, and scheduling.
- ▶ *Red Hat OpenShift Operators* automate the creation, configuration, and management of instances of Kubernetes-native applications. Understanding how to develop, install, and maintain Operators is important for efficient operations.
- ▶ The *Red Hat OpenShift Image Registry* stores and distribute container images, allowing developers to build, push, and pull container images as part of their application development and deployment process.
- ▶ *Source-to-Image* is a framework within Red Hat OpenShift that enables developers to build reproducible container images from source code.
- ▶ *Red Hat OpenShift CLI* is a command-line interface (CLI) tool that allows users to interact with and manage Red Hat OpenShift clusters. It provides a set of commands for deploying applications, scaling resources, managing users and permissions, and more.



Implementation architectural considerations

This chapter describes some of the key requirements and options when deploying Red Hat OpenShift on IBM Z or LinuxONE. Selecting a particular architecture does not necessarily mean locking it in as the final state. In practical terms, you can begin by deploying a Red Hat OpenShift cluster onto a single LPAR within an IBM Z or LinuxONE central processor complex (CPC). Once the cluster is up and running, you may then decide to distribute the cluster nodes across multiple LPARs for better resource utilization, maintenance, or performance reasons. Similarly, individual nodes within the cluster can be moved to different LPARs or even other IBM Z or LinuxONE CPCs, allowing for flexible and scalable cluster management.

It is also important not to over architect or over engineer the deployment. This can cause delays in deployment and increase costs of the overall solution.

The ability to endure an individual component outage is built into the design of Red Hat OpenShift. For example, the loss of a single control plane, infrastructure, or application compute node should not cause a loss of function. This removes any hard requirement for live relocation or live migration capabilities in the hosting hypervisors.

Initially, deploying a cluster with Dynamic Host Configuration Protocol (DHCP) can simplify the process of modifying certain aspects of the cluster in the future. For instance, if you need to replicate the cluster to another location for site failure recovery and require the CoreOS nodes to boot onto new IP addresses, DHCP facilitates this change by providing a supported method for updating IP addresses. Without DHCP, you may face limitations in making such modifications, potentially requiring more complex workarounds.

3.1 Red Hat OpenShift product deployment requirements

Red Hat has published the requirements for an Red Hat OpenShift deployment. For information on Kernel-based Virtual Machine (KVM), see [Preparing to install RHEL KVM on IBM Z and IBM LinuxONE](#). For information on IBM z/VM, see [Preparing to install with z/VM on IBM Z and IBM LinuxONE](#).

For a basic cluster, the resource requirements are roughly 100 GB of memory, 1 TB of disk, and 6 IBM Integrated Facilities for Linux (IFLs). The documented preferred starting resource requirement is 3 LPARs each with 6 IFLs.

Note: When planning for a Red Hat OpenShift cluster deployment, consider the initial resource requirements for the cluster itself, such as the number of nodes, CPU cores, and memory. However, these figures should not account for application workload resource demands or optional features that may be enabled later. To accommodate these factors and ensure the cluster's long-term stability and performance, allocate additional resources as needed. The product documentation usually offers guidance on the resources required to support each additional function, allowing you to make informed decisions about how to scale and optimize your Red Hat OpenShift cluster over time.

Enabling additional features can more than double the baseline CPU consumption compared to having no additional features enabled.

Failure to properly deploy a domain name server (DNS) and load balancer configuration in a product-required manner is one of the most common deploy failure reasons we have observed.

3.1.1 Load balancer

The load balancer infrastructure handles two separate aspects for a Red Hat OpenShift cluster: API traffic and Application Ingress traffic. Both of these are critical functions for a viable cluster. The loss of either of these functions severely impacts the cluster's ability to function. Per the production documentation, “[i]f you want to deploy the API and application ingress load balancers with a Red Hat Enterprise Linux (RHEL) instance, you must purchase the RHEL subscription separately.”¹

The load balancer function can be met by using HAProxy on a traditional Linux server. For a nonproduction environment, a single instance of HAProxy may be sufficient. For environments with higher availability requirements, you may need a load balancer with multiple instances that can stand in when the activated instance is no longer viable. In the case of multiple load balancer instances, you can use technologies such as Keepalived to provide the floating IP address that represents the logical composite of all three HAProxy instances.

You may also meet this requirement by using commercial solutions such as F5 or similar.

¹ https://docs.openshift.com/container-platform/4.12/installing/installing_ibm_z/installing-ibm-z-kvm.html#installation-load-balancing-user-infra_installing-ibm-z-kvm

Do not confuse these load balancers that run outside of the Red Hat OpenShift cluster with the Ingress Routers (HAProxy) that are a part of the Red Hat OpenShift cluster and run on CoreOS nodes. You will have both types of load balancing.

Additionally, you may even introduce a third layer of load balancing between Red Hat OpenShift clusters in different data centers, which is referred to as a Global Traffic Manager (GTM). The load balancer within a given data center for a cluster is typically referred to as a Local Traffic Manager (LTM).

If you use HAProxy, remember to open the load balancer ports in the Linux firewall and also to tell SELinux to allow HAProxy to bind to the ports by issuing the following command:

```
setsebool -P haproxy_connect_any=1
```

Red Hat OpenShift 4.12 API load balancer requirements

The conditions are as follows:

- ▶ Implement Layer 4 load balancing, which can be referred to as Raw TCP, SSL pass-through, or SSL Bridge mode. When using SSL Bridge mode, ensure that Server Name Indication (SNI) is enabled for the API routes.
- ▶ Utilize a stateless load-balancing algorithm.

Ports 6443 and 22623 are used on the front-end and back-end for the bootstrap node and the control plane nodes. The bootstrap node is removed from the load balancer configuration after the cluster is established early in the deployment.

For more information, see [Load-balancing requirements for user-provisioned infrastructure](#).

Red Hat OpenShift 4.12 application load balancer requirements

The conditions for application ingress are similar:

- ▶ Layer 4 load balancing.
- ▶ Connection or session persistence is recommended.

In the context of a Red Hat OpenShift cluster, ports 80, 443, and 1936 are designated for application ingress. To configure the load balancer's back end, direct it to the Red Hat OpenShift Ingress Routers that run on the compute nodes initially. However, you can later modify this configuration to have the Ingress Routers run only on Infrastructure compute nodes, after which the load balancer back end can be adjusted to point solely to the Infrastructure compute nodes. This process enables more efficient management and scaling of the OpenShift cluster.

For more information, see [Load-balancing requirements for user-provisioned infrastructure](#).

3.1.2 Network Configuration Management

Failure to properly deploy any of these components can lead to installation or operational failures. It is crucial to have a highly available DNS and DHCP environment for production use, with no single point of failure. Using multiple DNS and Network Time Protocol (NTP) servers is always recommended. Before deploying the OpenShift cluster, thoroughly test DNS entries for forward and reverse lookups.

DNS entries should be tested for forward and reverse lookups before attempting to deploy the cluster.

DNS

DNS is indeed a hard requirement for deploying and using Red Hat OpenShift. While it is possible to use a local bind DNS server during initial deployment to get the server up and running, it is generally recommended to use a corporate or enterprise DNS server after deployment for production use. This is because users of the cluster must use DNS names to access the cluster and its applications.

While it is theoretically possible to have entries in two different DNS servers, it is generally not recommended due to the added complexity and potential for conflicts. Instead, it is better to use a single, centralized DNS server for the Red Hat OpenShift cluster to simplify management and ensure consistency.

DHCP

Red Hat OpenShift supports both Static and DHCP-based IP address assignment. When using DHCP, the IP address assignment is expected to be fixed and persist to the given Media Access Control (MAC) address of the CoreOS node, in relation to the DNS entries. Changing static IP assignments after the initial deployment can be problematic. If you need to change IP addresses and corresponding DNS entries for the CoreOS nodes, using a DHCP server would be ideal. Such a change would need to be coordinated across components and may require downtime. DHCP offers additional benefits, such as allowing you to set DNS servers to use and Maximum Transmission Unit (MTU) size to use in the CoreOS nodes.

NTP

Public NTP servers are configured by default. If you do not allow access to the public time servers you will need to configure the Red Hat OpenShift cluster to use locally provided time servers with its [Chrony](#) services, after the cluster is deployed. This should be done as soon as possible, as time may drift on the CoreOS guest operating systems. Common time is critical to the operation of a cluster.

3.1.3 Bastion host

The bastion host is a critical component in deploying and administering the Red Hat OpenShift Cluster. It is designed to provide secure and direct access to the cluster for administrators and users. While it is expected that the bastion host remains available and operational, there is a small risk that it could experience a momentary outage. However, this is not expected to impact the operation of the cluster itself.

The standard Red Hat OpenShift architecture includes only a single Bastion host, but the `oc` command interface and content can be replicated on to other hosts for redundancy and improved performance. This allows for a more resilient and scalable architecture, as the load of the `oc` command interface can be distributed across multiple hosts.

3.1.4 Hosting hypervisor environment

Red Hat OpenShift requires deployment under either KVM or IBM z/VM as a software-based hypervisor. At the time of writing, Bare Metal installs of Red Hat OpenShift on IBM Z or LinuxONE are not currently supported. Both KVM and IBM z/VM are fully supported by Red Hat OpenShift and you may select to use either or both.

According to the [Red Hat OpenShift Container Platform on IBM Z and IBM LinuxONE reference architecture](#), it is recommended that you spread your Red Hat OpenShift cluster across three LPARs because any one LPAR can be taken out of service at any time for planned or unplanned events. If the recommended reference architecture is followed, the cluster can survive the loss of any one LPAR at a time.

While both hypervisors support Live Migration or Live Guest Relocation, we generally do not plan to use these features due to the nature of Red Hat OpenShift already having the built-in ability to respond automatically to both planned and unplanned outages when deployed on the proper architecture. Live migrations or relocations do have the potential for adverse reactions such as node etcd leader elections or loss of quorum from a cluster, due to delayed responses.

In this IBM Redbooks publication, we demonstrate deployments that use both hypervisors.

3.2 Number of hosting logical partitions for a cluster

Red Hat OpenShift can be deployed into a single LPAR or multiple LPARs. A single LPAR may meet all of your requirements for nonproduction environments. Production environments typically require enhanced resiliency and availability. They need to be serviceable for maintenance updates, without causing full application outages.

Single or multiple LPAR deployments in an IBM Z CPC or LinuxONE system can be modified over time to accommodate changing business needs. The different components that make up the Red Hat OpenShift cluster, such as the Red Hat OpenShift Container Platform, etcd database, and Operators, can be moved to different partitions, either on the same or different LinuxONE or IBM Z CPCs. Additionally, you can add more CoreOS nodes that can be hosted on different LPARs as well.

With the right planning and preparation, there is no need to delay deploying a Red Hat OpenShift cluster because the second, third, or nth partition environment was not ready. These partitions can be made part of the cluster after the initial installation, allowing for a more flexible and scalable architecture.

The use of at least three LPARs is recommended for production environments for enhanced resiliency and serviceability. In a three-LPAR environment you would place one control plane node in each LPAR. The same can be done with infrastructure and application compute nodes. While they do not have a requirement for three nodes because of quorum, like the control plane, it becomes logical to take advantage of the three LPARs in that way. Also, when deploying Red Hat OpenShift Data Foundation, you need three Red Hat OpenShift Data Foundation nodes for quorum and replication purposes. You would place one node in each of the three LPARs.

3.3 Deployment architectures used in this paper

Both a single LPAR and a three LPAR configuration are deployed in this paper as references. Details on these deployments can be found in Chapter 5, “Red Hat OpenShift deployment topologies on IBM Z” on page 41.

3.3.1 Single-LPAR deployment configuration

In this Red Hat OpenShift cluster, the single LPAR architecture consists of three RHEL servers and seven CoreOS servers. The hypervisor uses two network bonds: bond0 and bond1. Bond0 consists of the pair of Open Systems Adapters (OSA) that are used for hypervisor administration, while bond1 consists of the pair of Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) adapters that are used for the Red Hat OpenShift cluster and supported RHEL guests.

The OSA adapters in bond0 provide high-speed network connectivity for the hypervisor, while the RoCE adapters in bond1 provide low-latency, high-bandwidth network connectivity for the Red Hat OpenShift cluster and RHEL guests.

In the single LPAR configuration, we use Ansible to deploy the Red Hat OpenShift cluster. We manually deploy the DNS and load balancer services as we do not desire to host them on the bastion node.

Figure 3-1 shows our configuration for the single LPAR architecture.

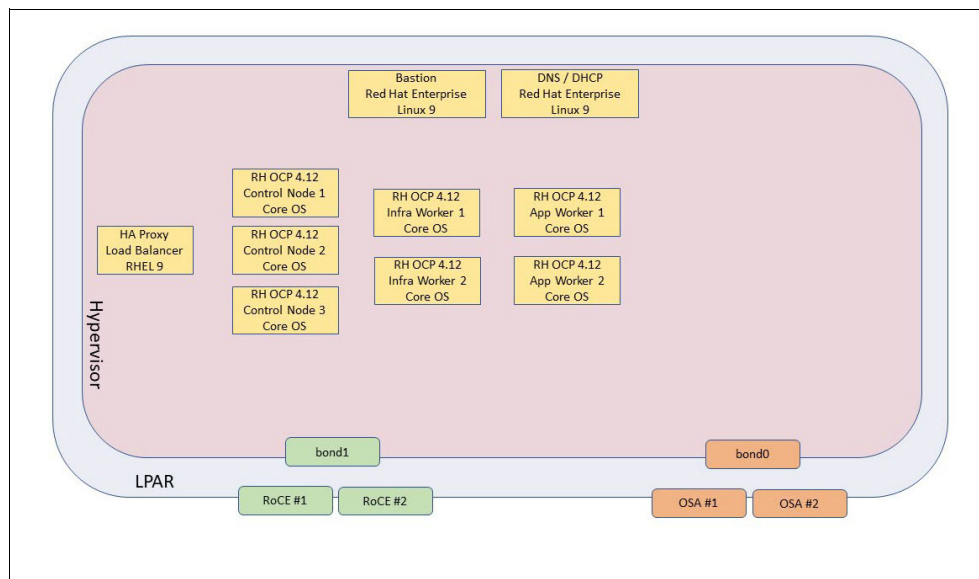


Figure 3-1 Single LPAR

For more information, see 5.2, “IBM z/VM One LPAR cluster implementation” on page 43.

3.3.2 Three-LPAR deployment configuration

In this paper, the second deployment utilizes three LPARs for enhanced availability and potentially to bring additional compute, network, or storage capacity. The three LPAR configuration does allow for maintenance of an LPAR/hypervisor without reducing or removing cluster function.

For the three LPAR configuration we do not use Ansible so we can show you the simple steps needed to deploy the cluster. For more complex deployments, Ansible automation may not always meet your needs without modification.

We implement the three LPAR configuration according to [Fast-track installation by using a prepackaged QCOW2 disk image](#). The initial storage requirements with this method is dramatically reduced.

Figure 3-2 shows our configuration for the three LPAR architecture.

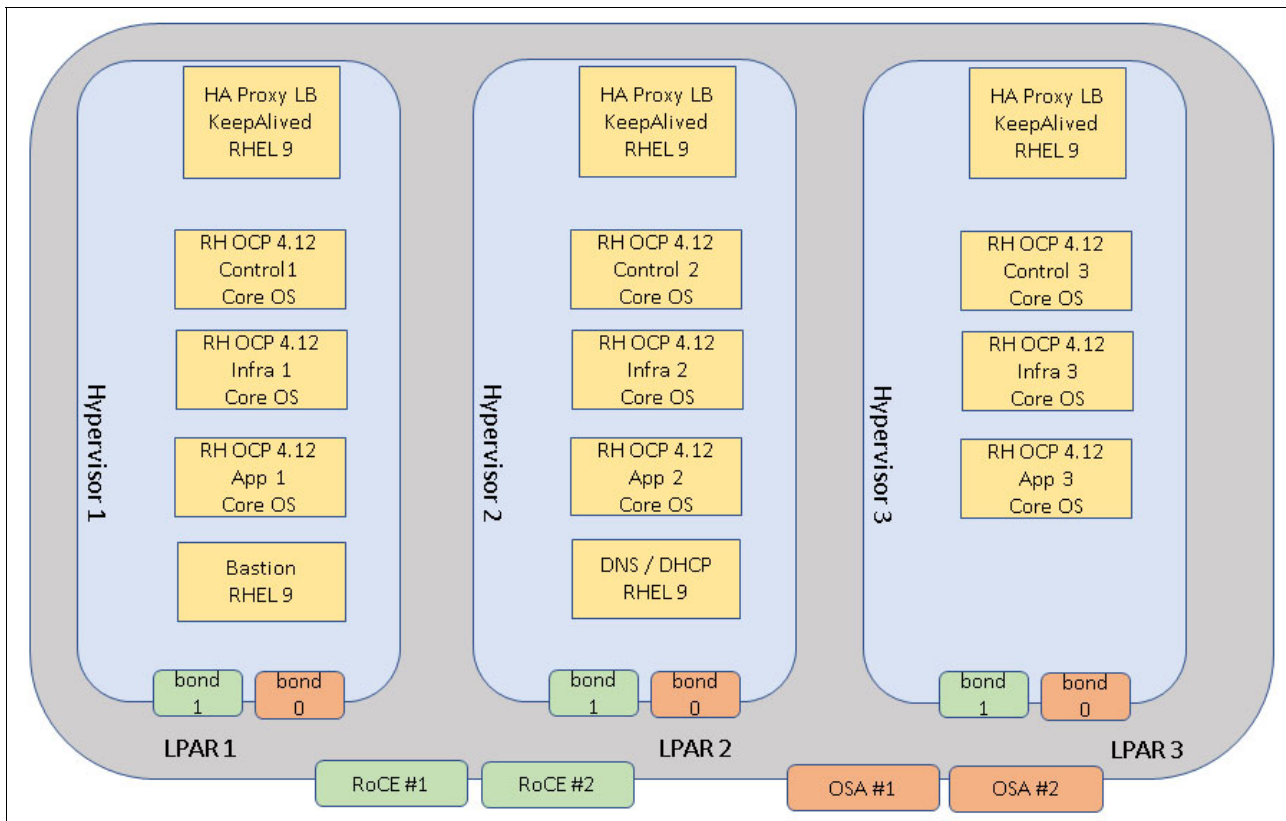


Figure 3-2 Three LPAR Configuration

For more information, see 5.3, “IBM z/VM three-LPAR cluster implementation” on page 64.

3.4 Storage architecture

Disk storage must be viewed and understood from multiple different perspectives. IBM Z and LinuxONE support Fibre Channel Protocol (FCP) attached Small Computer System Interface (SCSI) Logical Unit Numbers (LUNs), IBM FICON® attached Enterprise Computing Control Model Disk (ECKD), as well as Internal NVMe storage. CoreOS nodes are each deployed with a single boot disk that is recommended to be at least 120 GB. Red Hat OpenShift has Ephemeral and Persistent storage. Ephemeral storage is a transient workspace for the pods that lacks some of the manageability of persistent storage. The data in ephemeral storage does not live beyond the life of the pod.

3.4.1 FCP-attached SCSI and FICON-attached ECKD storage

When working with FICON-attached ECKD, plan for use of Parallel Access Volumes (PAVs) because the size of the ECKD devices being used. In the case of IBM z/VM, the PAVs should be directly available to the guest containers the CoreOS is operating in. In the case of KVM, the PAVs only need to be in the hypervisor itself, as the Virtio interface does not have any need or concept for PAVs.

Multipathing is highly recommended for FCP-attached SCSI LUNs. When using IBM z/VM and dedicating the FCP devices to a guest, multipathing is needs to be enabled within the CoreOS guest. When using KVM the multipathing is performed in the KVM host. The Virtio device, when passed to a guest, does not require multipathing.

3.4.2 CoreOS node storage

Each CoreOS node has the need for a single disk that it boots from. This disk is recommended to be 120 GB. When backing this with ECKD storage, plan for a loss of 20% of usable space due to the interblock gaps, assuming a maximum 4 K block size.

In the KVM use case, the QCOW2 images used are typically hosted in `/var/lib/libvirt/images`. This might be a logical volume composed of one to many physical volumes. QCOW2 images can be sparse and each guest can share base QCOW2 images where there is a unique delta file per guest. This will greatly reduce the initial storage required, but you will still need to plan to support the total required values over time. You can also use `discard=unmap` in this hypervisor to keep the QCOW2 image actual disk usage lower.

If you deploy Red Hat OpenShift Data Foundation, you will add additional disk to these nodes after they have been deployed.

3.4.3 RHEL virtual server disk storage

You may need to support one or more RHEL virtual servers to support your cluster. There is always a bastion host, which is not a CoreOS system. You may choose to host other functions on additional RHEL servers. DNS, DHCP, Network File System (NFS), and HAProxy are common examples. The installation instructions for Red Hat OpenShift document the use of a 100 GB disk for NFS to host the image repository function. This may be fine initially but you may want to consider hosting your persistent storage on a more resilient solution such as Red Hat OpenShift Data Foundation or IBM Spectrum® Scale Container Native Storage Access (CNSA). These may not only provide enhanced performance but enhanced resiliency as well.

3.4.4 Red Hat OpenShift Persistent Storage

Persistent storage is a key aspect to plan for when deploying your cluster. The requirement for persistent storage can exist even when you do not plan to deploy middleware such as databases or messaging systems. Applications can utilize persistent storage as well as Red Hat OpenShift infrastructure such as monitoring and logging. In our example environment in this paper will not show deploying Red Hat OpenShift Data Foundation or IBM Spectrum Scale CNSA.

Red Hat OpenShift Data Foundation

Red Hat OpenShift Data Foundation is one way to provide persistent storage for your containers within the Red Hat OpenShift Container Platform. It can be used to deliver block, file, or object storage on the platform. Much of the Red Hat OpenShift Data Foundation technology is based in [Ceph](#). Ceph clusters replicate their data across multiple nodes to guarantee high availability and prevent data loss if there is hardware failure. While commodity storage devices can be used for basic setups, enterprise-grade storage servers often include redundancy features that eliminate the need for additional commodity storage. These systems are designed to handle high availability and data replication requirements efficiently, making them suitable for mission-critical applications.

When planning to deploy Red Hat OpenShift Data Foundation, pay special attention to the required number of processors and the amount of memory that must be allocated to your compute nodes. Red Hat OpenShift Data Foundation may fail to deploy, or provide reduced function, if the required processor and memory resources are not provided. Actual processor consumption observed after deployment is typically well below the number of virtual processors allocated during deployment.

It is important to also understand that Red Hat OpenShift Data Foundation replicates storage between its nodes over the network to ensure availability of the persistent storage it is providing. For example, if your application requires 2 TB of persistent storage, you would provide a 2 TB disk in each of your nodes. When providing 2 TB of persistent storage to a container, 6 TB of disk is consumed. Red Hat OpenShift Data Foundation does not include a built-in Logical Volume Manager (LVM) that merges multiple physical disks into a single logical volume. To create a 4 TB persistent disk, you must provision a minimum of 4 TB of disk space on each node in the OpenShift cluster. This ensures that each node has sufficient resources to handle the required disk size. Since the size of these disks is significant, when using ECKD, plan for PAVs and the 20% loss of usable disk space when presented to Linux.

In Red Hat OpenShift Container Platform 4.12, Red Hat OpenShift Data Foundation clusters are only supported as internal clusters and cannot be deployed externally for IBM Z and LinuxONE.

IBM Spectrum Scale Container Native Storage Access

IBM Spectrum Scale CNSA is another way to provide persistent storage to your Red Hat OpenShift containers. IBM Spectrum Scale is deployed in a cluster made up of an odd number of nodes, typically a minimum of three. When used with Red Hat OpenShift, there are two main parts:

- ▶ The IBM Spectrum Scale cluster that is external to Red Hat OpenShift that functions the same as any IBM Spectrum Scale deployment without Red Hat OpenShift.
- ▶ The second part is the deployment of the IBM Spectrum Scale CNSA portion within the Red Hat OpenShift cluster that provides high-performance, scalable storage for applications running within the Red Hat OpenShift environment.

IBM Spectrum Scale CNSA provides shared file access to containers in Red Hat OpenShift as persistent storage.

IBM Spectrum Scale CNSA differs from Red Hat OpenShift Data Foundation in that it does not require a separate set of Red Hat OpenShift Container Platform compute nodes. You designate which Red Hat OpenShift nodes you want the IBM Spectrum Scale CNSA containers to operate in. The IBM Spectrum Scale CNSA containers build their own IBM Spectrum Scale cluster within the Red Hat OpenShift Container Platform nodes.

IBM Spectrum Scale CNSA can be configured to provide storage over a Internet Protocol network connection or by direct Fibre Channel or FICON attachment. IBM Spectrum Scale can also combine multiple physical disks together in the IBM® General Parallel File System (GPFS™) and present them as one logical disk to the Red Hat OpenShift cluster. In its default configuration, IBM Spectrum Scale would not replicate disk storage to other nodes; however, it does have the means to perform replication if needed.

Local Storage Operator

The Local Storage Operator can be used to provide statically created persistent volumes in your Red Hat OpenShift cluster. With the Local Storage Operator, the storage is not shared across the nodes of your cluster; it only exists locally to the node it is attached to. At first this may seem problematic; however, there are several use cases where it may be an ideal fit. For example, if deployed with EnterpriseDB (EDB) Postgres where the database exists on a three node cluster in your Red Hat OpenShift Container Platform cluster, the database handles all the replication to each node and its local storage.

If your hypothetical EDB Postgres database needs to serve 4 TB of data, you would need three 4 TB disks for the local storage operator. Alternatively, if you use Red Hat OpenShift Data Foundation with the standard three replica configuration, each EDB Postgres node needs a unique 4 TB persistent disk, backed up in Red Hat OpenShift Data Foundation by three 4 TB disks, for a total of nine 4 TB disks.

The Local Storage Operator may lack some of the advanced features you would enjoy with Red Hat OpenShift Data Foundation or IBM Spectrum Scale CNSA, such as snapshotting, but you may not have those needs. You may be able to perform your backups live with the middleware you plan to use and not need those snapshots.

NFS persistent storage

NFS is another way to provide persistent storage to an NFS cluster as shared file storage. It might be a quick way to deploy initial persistent storage, but it is often viewed to be less enterprise-ready. You might find that it lacks certain integrated features such as snapshotting, encryption, replication, and the uninterrupted resiliency provided by Red Hat OpenShift Data Foundation or IBM Spectrum Scale CNSA single node restarts versus NFS server restarts.

3.5 Multitenancy with other workloads

Multitenancy is a concept that manifests in various ways across different contexts. In the realm of IBM Z and LinuxONE systems, multitenancy can be observed in several scenarios:

- ▶ **Workloads in Different LPARs:** An LPAR is a virtualized environment that runs its own operating system and applications. When multiple LPARs are present within the same IBM Z or LinuxONE system, each LPAR represents a separate tenant, ensuring isolation and security for each workload.
- ▶ **Workloads in Different Guests:** Within a single LPAR, multiple virtual machines or containers (guests) can be created to host different workloads. Each guest can be considered a separate tenant, with its own resources and isolation mechanisms.
- ▶ **Workloads in the Same Red Hat OpenShift Cluster:** Red Hat OpenShift is an open source containerization and orchestration platform that allows for the deployment and management of multiple applications and services across a cluster. In a multitenant Red Hat OpenShift cluster, each application or service represents a separate tenant, with its own set of resources and configurations.

During our discussion, we focus on the unique controls and considerations associated with multitenancy in IBM Z and LinuxONE systems, as these environments offer robust security and isolation features tailored to the needs of highly regulated industries. The Red Hat OpenShift production documentation provides comprehensive coverage of multitenancy in the context of that platform.

3.5.1 LPAR level controls in IBM Z and LinuxONE

LPARs have a high degree of ability to isolate workloads. They have been evaluated under Common Criteria at Evaluation Assurance Level 5+. For more information on the BSI issued certificates, see [BSI-DSZ-CC-1133-2020](#) and [BSI-DSZ-CC-1160-2021](#).

When managing a system with multiple LPARs, the primary concern for a systems administrator often involves striking an optimal balance between resource allocation and isolation for each tenant or workload.

3.5.2 Guest controls in KVM

RHEL (the product in which the KVM we used is shipped) was Common Criteria certified by National Information Assurance Partnership (NIAP). It was tested and certified for ISO/IEC 15408 to follow 4.2.1 of the NIAP General Purpose Operating System Protection Profile including Extended Package for Secure Shell (SSH). Previously, RHEL operating systems were certified at EAL4+. The treaty the countries use to recognize Common Criteria certifications now only recognizes up to EAL2.

RHEL has also received FIPS 140-2 and 140-3 certifications for its key cryptographic modules.

For a list of all Red Hat product certifications, see [Compliance Activities and Government Standards](#).

The US Defense Information Systems Agency (DISA) publishes a Secure Technical Implementation Guide for RHEL that can be found here [STIGS Document Library](#).

Also, KVM has several fine controls for resource allocation and priorities between different domains hosting different workloads. Virtual process and memory allocation can be configured in the domain XML for a guest. A shares value can be specified to indicate the processor priority of one domain over another on a relative basis. Period and Quota can be specified to control process consumption as more an absolute limit. Numerous other cgroup controls exist as well.

3.5.3 Guest controls in IBM z/VM

IBM z/VM has completed its Common Criteria certification, which certifies the product is in accordance with NIAP Virtualization Protection Profile (VPP) and with Server Virtualization Extended Package. IBM z/VM has been FIPS 140-2 validated, and also Common Criteria EAL 4+ certified. The certificates can be found at the following websites:

- ▶ [OCSI](#)
- ▶ [Cryptographic Module Validation Program](#)
- ▶ [IBM® z/VM® Version 7 Release 2 System SSL Cryptographic Module](#)

As a systems administrator, managing multitenancy in a system with multiple guests involves several key tasks related to resource allocation and prioritization:

- ▶ **Guest Configuration:** Properly configuring each guest with relative or absolute share values, the number of virtual processors, and the amount of virtual memory is crucial for ensuring fair resource distribution and preventing resource contention. These settings can be adjusted at run time using the IBM z/VM **SET SHARE** command.
- ▶ **Dynamic Resource Allocation:** Adjusting resource allocation during runtime based on changing workload demands is essential for maintaining system performance and stability. The IBM z/VM **SET SHARE** command allows for dynamic modification of relative and absolute shares for each guest.
- ▶ **Monitoring and Reporting:** Regularly monitoring and reporting on system performance and resource utilization helps identify potential bottlenecks, resource contention, or security vulnerabilities. This information can be used to fine-tune resource allocation and isolation strategies for each guest.
- ▶ **Isolation Mechanisms:** Implementing robust isolation mechanisms between guests is vital for maintaining security and data privacy. This includes configuring access controls, network segmentation, and virtualization technologies to prevent unauthorized access and data leakage.

3.6 Recovery site considerations

When providing mission-critical applications, organizations endeavor to plan for failures from the smallest components, such as an individual network card, to a full site or region failure, where processing must be resumed as soon as possible at a distant location.

Decades ago, recovering at a remote location meant restoring from tape backups. This is a slow and very manual process. Testing this took weeks of planning and often a week to run. As technology has evolved, disk replication has replaced tape technologies. Migrating an existing system to a new data center can be time-consuming and complex, with potential recovery time taking hours and requiring changes to IP addresses, gateways, and other aspects, potentially causing disruptions at inconvenient times. To activate your Red Hat OpenShift cluster at a remote location, consider using a DHCP server to help automate some of the change. Unlike RHEL, Red Hat OpenShift CoreOS nodes do not have a published supported way to changes IP address when you do not use DHCP.

One modern method that enjoys near zero downtime is to build Red Hat OpenShift clusters at the primary and secondary or recovery sites. For stateless workloads, this is easily managed with Global Traffic Managers (GTMs) that route traffic to the proper data center. In this case, you can run both sites active simultaneously.

3.7 IBM Secure Execution

IBM Z and LinuxONE have a unique technology called IBM Secure Execution. It can isolate and protect machine state and memory information to provide a Trusted Execution Environment. The design is to isolate this from everything outside of the virtual server. The KVM Hypervisor, the HMC, and the Support Element have no way to access the memory or state information of the guest. The guest is built to be accessible from only specific IBM Z or LinuxONE machines you designate and the disk storage the guest uses is encrypted.

Secure Execution for Linux isolates and protects KVM guests from hypervisor access. The hypervisor administrator can still manage and deploy workloads, but cannot view data on a guest. Multiple tenants (applications) running in an LPAR as second-level guests under KVM have fully isolated environments, which help protect intellectual property and proprietary secrets.

IBM Secure Execution is available since Red Hat OpenShift 4.13 and Red Hat provides a unique QCOW2 image for Secure Execution. Each Red Hat OpenShift node gets encrypted with a unique Linux Unified Key System (LUKS) key.

For more information on deploying IBM Secure Execution, see [Installing RHCOS using IBM Secure Execution](#) and [IBM Secure Execution for Linux](#).

3.8 IBM CEX high-performance hardware security modules requirements

The IBM CEX adapter can be configured to act as a tamper proof hardware security module or HSM. For more information, see [IBM Crypto Express meets RHOCP on IBM Z/LinuxONE](#).

Red Hat OpenShift supports the IBM CEX adapter through the Kubernetes device plug-in for IBM CEX cards, which provides access to IBM CEX adapters for IBM Z and LinuxONE Kubernetes container loads. The IBM CEX adapter provides fast hardware cryptography, as well as protecting keys in an HSM, providing the strengths of IBM Z/IBM® LinuxONE regarding advanced security. For more information, see the [Kubernetes device plug-in for IBM Crypto \(CEX\) cards - Installation and User Guide](#).

3.9 FIPS requirements

FIPS mode can be enabled for a cluster, but must be done so in the `install-config.yaml` before deployment. This cannot be changed after deployment. When enabling FIPS, expect to encounter some additional processor consumption.

3.10 Multus for a second network interface

If you require HiperSockets with Red Hat OpenShift, using Multus as a method for Red Hat OpenShift nodes to communicate externally through a RoCE or OSA adapter, and communicate internally to each other or an adjacent z/OS LPAR through HiperSockets, can be a good solution. The IBM z/OS would need to use IBM z/OS HiperSockets Converged Interface (HSCI) for Layer 2 support. On the Linux LPARs, the IBM z/VM LPAR has HiperSockets Bridge and KVM LPARs have HSCI support as well. Multus does not use the Red Hat OpenShift software-defined networking (SDN), which can introduce some additional latency in communications.

If you are considering using Multus, you may want to start with a MacVTap deployment using the `ipam-tool` command and whereabouts for IP address assignments in your Red Hat OpenShift containers. For more information, see [Configuration for a MACVLAN additional network](#).

Also, pay attention to the MTU sizes allocated in your nodes for your network connections and your cluster maximum MTU size. For more information, see [Changing the MTU for the cluster network](#).

3.11 Authenticating

Initially, credentials for the kubeadmin user ID are generated as an output of the installation process. For more information, see [Supported identity providers](#).

It is one of the first configuration changes typically made after a cluster ID is deployed. The simplest one may be the `htpasswd` method, as there is no need to integrate with other systems and you can eliminate that shared user ID of kubeadmin quickly. One of the other most common methods is to use LDAP. The 389 LDAP server in Red Hat Identity Manager is known to work well as an LDAP source and there are multiple Red Hat Knowledge Base articles on the topic. In general, there are no IBM Z or LinuxONE specific aspects to this topic.

3.12 Monitoring

There are multiple aspects or layers to monitoring that can be considered because the architecture includes more than Red Hat OpenShift itself. For the IBM Z and LinuxONE HMC you can monitor the partitions, and physical resources and adapters. The performance data is kept for 36 hours on the HMC. For more information, see the [IBM Z HMC Prometheus Exporter](#).

It uses Prometheus to store the data and Grafana Dashboards to visualize the data. You can find a full set of s390x Prometheus binaries at [Prometheus DOWNLOAD](#), including `node_exporter` and `alertmanager`. `node_exporter` can also be installed on your RHEL systems and used to send your performance/capacity data to Prometheus and visualized in Grafana. This would apply to your KVM hosts, bastion hosts, load balancers, and DNS/DHCP servers. HAProxy can also directly feed its performance data to Prometheus. For IBM z/VM environments, Performance Toolkit and ITM / Omegamon are typically used to record and analyze performance.

Within the Red Hat OpenShift Cluster, Prometheus and Grafana are also used. For more information, see [Configuring the monitoring stack](#).

Typical initial customization is to move the monitoring stack to the Infrastructure Nodes and that have [taints](#) implemented on them. One of the next customizations is to implement persistent storage for the monitoring stack. Without doing so, the performance data is lost when a node is rebooted. Finally, there are a series of steps to enable the monitoring of user projects (your applications), in addition to the default Red Hat OpenShift infrastructure.

3.13 Logging

There are several logging options to consider in Red Hat OpenShift. None of the considerations are specific to the IBM Z or LinuxONE platforms, although knowledge of the options can be helpful.

One option is to not even store the logs on the platform in the cluster at all. Red Hat OpenShift comes with a log forwarding capability. So, if you have an Enterprise Logging solution, such as Splunk, you may choose to forward all the logs there. You can send logs or portions of logs to multiple locations. You may also have need to send the logs to Security devices that perform security information and event management (SIEM) functions.

A long-time option now has been to use the Elasticsearch provided in Red Hat OpenShift. If your logging requirements are modest, you may find this sufficient. The one observation we have had is that Elasticsearch can require significant additional processor and memory requirements well beyond the minimal number of IFLs required to install a cluster.

A third option is the new Loki-based logging stack. For more information, see [Logging and using LokiStack](#).



Resource considerations for Red Hat OpenShift

As businesses continue to adopt containerization technologies and leverage the benefits of virtualization, it is critical to understand the intricacies of managing resources effectively.

In this chapter, we provide an overview of LPAR adjustments and weights, IBM z/VM weights and their assignment, KVM scheduling and management, and the prioritization of resources between production and nonproduction environments.

Running Red Hat OpenShift on IBM z/VM or KVM brings the power and reliability of IBM Z and IBM LinuxONE together with a robust and flexible infrastructure for containerized workloads. To ensure optimal performance and resource utilization, administrators must consider various factors and make informed decisions.

4.1 LPAR adjustments and weights

LPAR adjustments and weights play a vital role in managing resources on IBM Z. LPAR technology allows the subdivision of a physical server into multiple virtual machines, providing isolation and resource allocation. By adjusting LPAR weights, administrators can prioritize the allocation of resources among different partitions, ensuring that critical workloads receive the necessary computing power.

By assigning weights to individual virtual machines, IBM z/VM ensures that resources are allocated proportionally based on their assigned weights. Understanding how to assign these weights effectively is crucial to optimizing resource utilization and balancing the workload across multiple virtual machines.

4.1.1 General LPAR adjustments

On IBM Z and LinuxONE, the physical resources of a system can be divided into numerous LPARs. Each LPAR looks and functions like its own physical machine, independent of and without knowledge of, other partitions with diverse configurations, processors, storage, I/O, and operating system.

LPARs are used for hardware consolidation and workload balancing. Partitions are managed by a Processor Resource/Systems Manager (PR/SM) hypervisor that runs natively on the machine. PR/SM hardware is utilized by two IBM products: a microcode feature called Logical Partitioning Mode and VM/XA Multiple Preferred Guest. Work is dispatched on a logical processor basis, not on the partition as a whole. Tasks from different partitions can be operating in parallel on different physical processors. Up to 85 partitions can be defined and active at any given time. The activation profile contains configuration details such as the number and type of processors, amount of storage etc. The activation is done manually through the Hardware Management Console (HMC) or automatically at Power on Reset (POR).

When an LPAR is defined, the amount of central and expanded storage allocated to it is specified in 1 MB increments. This storage is dedicated and is not shared with or accessible in any way by other LPARs. Individual processors, such as IFLs or Central Processors (CPs), on the other hand, can either be dedicated to or shared among LPARs. Dedicated IFLs are used exclusively by the LPAR and are not controlled in any way by the PR/SM feature. LPARs can have multiple logical IFLs defined, but the number cannot exceed the number of IFLs under PR/SM control.

When you divide your central processing complex into separate logical partitions, each partition is assigned its own LPAR weight, which corresponds to the percentage of overall processing power that is guaranteed to the work in that partition.

PR/SM shares IFL processor resources among LPARs through weighted prioritization. The weighting for each LPAR can be dynamically modified. PR/SM will not take control of a CP when the logical processor goes into a wait state but leaves the CP under the control of the LPAR for the entire duration of the time slice.

Processing weights

As an example, consider a system with six IFLs and three LPARs defined in Table 4-1.

Table 4-1 LPAR weight table

LPAR Name	Number of IFLs	Weight
LNxVM1	1	300
LNxVM2	6	100
LNxVM3	2	900

Processing weights can range from 1 - 999. The processing weights for all active, sharing Logical Partitions are added together. This total is considered to be 100% of the processing resource available to shared IFL. The share of processing resources for each LPAR is calculated by dividing the processing weight for each sharing LPAR by the total processing weight, as shown in Example 4-1.

Example 4-1 Share of processing resources

LNxVM1 $300/1300 = 23.1\%$
LNxVM2 $100/1300 = 7.7\%$
LNxVM3 $900/1300 = 69.2\%$

The share of processing resources for each online logical core with Hyper Dispatch disabled in the logical partition is calculated by dividing the share for each LP by the number of online logical cores. The percentage is used to determine the priority for I/O interruptions is shown in Example 4-2.

Example 4-2 Priority for I/O interruptions

LNxVM1 $23.1/1$ IFL = 23.1%
LNxVM2 $7.7/6$ IFL = 1.3%
LNxVM3 $69.2/2$ IFL= 34.6%

The PR/SM capping function provides the capability of limiting CPU resource usage for one or more processor types for one or more logical partitions. The relative processing weight of a processor type for a logical partition is its capping value for that processor type.

For more information, see 4.15, *ABCs of z/OS System Programming Volume 10*, SG24-6990 as well as the *PRSM Planning Guide*, SB10-7178.

4.1.2 IBM z/VM weights

IBM z/VM weights are a mechanism that allows administrators to allocate resources based on the relative priority of virtual machines (VMs) within an LPAR. By assigning weight values to VMs, IBM z/VM dynamically adjusts resource allocations to optimize performance and resource utilization in IBM Z systems.

4.1.3 Adjustments for Red Hat OpenShift

When setting up an IBM z/VM or KVM environment for Red Hat OpenShift, several LPAR adjustments can be made to optimize the performance and resource utilization:

- ▶ **CPU Allocation:** Ensure that an adequate number of CPUs is allocated to the LPAR hosting the Red Hat OpenShift environment. The number of CPUs should be based on the expected workload and resource requirements of the containerized applications. Consider factors such as the number of concurrent pods, the CPU-intensive nature of the applications, and any specific requirements of the workloads.
- ▶ **Memory Allocation:** Allocate sufficient memory to the LPAR to accommodate the Red Hat OpenShift environment and the containerized applications. The memory allocation should consider the size and number of containers running concurrently, along with any additional memory requirements for other system processes and services. Monitor the memory usage of the environment and adjust the allocation as needed.
- ▶ **Disk Space:** Ensure that an appropriate amount of disk space is allocated to the IBM z/VM LPAR to accommodate the Red Hat OpenShift environment, VM images, and storage requirements of the containerized applications. Consider the expected workload, the size of the container images, and any persistent storage requirements. Regularly monitor disk space utilization to prevent storage constraints and ensure smooth operation.
- ▶ **Networking:** Configure the networking settings of the LPAR to support the networking requirements of Red Hat OpenShift. This includes assigning appropriate IP addresses, setting up network interfaces, ensuring appropriate bandwidth, and configuring network routing.
- ▶ **I/O Configuration:** Optimize the I/O configuration of the LPAR to provide efficient disk and network I/O for the Red Hat OpenShift environment. This involves configuring I/O adapters, disk subsystems, and network interfaces for optimal performance. Consider enabling features such as virtual SCSI (VSCSI) or virtual Ethernet (VETH) to enhance I/O performance.
- ▶ **Monitoring and Resource Management:** Implement a monitoring and resource management solution to track the performance and resource utilization of the LPAR and the Red Hat OpenShift environment. Utilize tools such as IBM z/VM Performance Toolkit (IBM z/VM Perfkit) for or other monitoring solutions to gain insights into CPU usage, memory utilization, disk I/O, and network traffic.

The specific LPAR adjustments may vary based on the scale, workload, and performance requirements of your Red Hat OpenShift deployment. It is essential to monitor the system regularly and make adjustments as needed to ensure efficient resource allocation and optimal performance for containerized applications. For more information, see [Recommended host practices for IBM Z & IBM® LinuxONE environments](#).



Red Hat OpenShift deployment topologies on IBM Z

The need to have a standard DevOps pipeline which would help in rapidly building, deploying, and maintaining applications across various architectures and hardware platforms has become a basis for clients to adopt a hybrid cloud deployment topology for their workloads.

This chapter describes the various deployment topologies to follow while planning to deploy Red Hat OpenShift on IBM Z.

Red Hat OpenShift Container Platform on IBM Z and LinuxONE, which is based on Kubernetes framework, provides IT Organizations and business leaders with a secure platform to build containerized applications as part of their hybrid cloud strategy.

5.1 Deployment topology criteria

The deployment topology of an Red Hat OpenShift Container Platform cluster is driven by several factors, which we describe in this section.

5.1.1 Data gravity

Red Hat OpenShift Container Platform exploits the key platform capabilities of IBM Z and LinuxONE by moving processing to the data when colocating containerized applications with traditional workloads (for example data lakes, databases, transactional systems, or other traditional workloads running in Linux on IBM Z or IBM z/OS). In this case, the applications have close proximity to the data, resulting in reduced latency, response time, deployment, security, service and cost.

The resulting synergy can dramatically improve performance and total cost of ownership (TCO).

5.1.2 Consolidation and TCO Reduction

By consolidating an Red Hat OpenShift Container Platform environment onto IBM Z or IBM LinuxONE, you can achieve many economic and operational advantages. Three-dimensional scalability (vertical, horizontal and combined) results in high flexibility without the need for a new hardware footprint. This is a key advantage for dynamic workloads and unpredicted growth. The other key benefit you may achieve from a consolidation perspective is contributing towards sustainability goals by cutting down on the requirements around energy usage and floor space.

5.1.3 Business continuity

For business continuity, consider these two aspects:

- ▶ High availability
- ▶ Disaster recovery

For high availability (HA), IBM Z and IBM LinuxONE provide an internal network with significantly more reliability based on a higher degree of redundancy in the hardware. Because virtualization happens within a single hardware environment, the networking traffic is more predictable with less latency.

In an environment with IBM z/VM, you can take advantage of IBM z/VM Single System Image (SSI) and Live Guest Relocation (LGR), which can help reduce operational costs (though this option requires shared ECKD Direct access storage devices (DASDs)). As mentioned in 3.1.4, “Hosting hypervisor environment” on page 25, this is not the recommended approach.

Building a disaster recovery (DR) setup with IBM Z and IBM LinuxONE is much simpler in such an environment, because you have to deal with fewer hardware units. Most of all, you can take advantage of current Extended Disaster Recovery (xDR) solutions for IBM Z and IBM LinuxONE - based on IBM Geographically Dispersed Parallel Sysplex® (GDPS).

In IBM Z, hardware consolidation means that capacity, operational analytics, and analysis are simplified because all results are consolidated on the single hardware machine for the entire Red Hat OpenShift Container Platform environment versus having a distributed environment with many physical servers. As a result, capacity planning is more predictable.

5.1.4 Vertical Solutions

Red Hat OpenShift Container Platform is intended for generic orchestration and life-cycle management of containerized workloads. While it is used widely among industries, the most relevant background of Red Hat OpenShift Container Platform for IBM Z and IBM LinuxONE customers is in industries like:

- ▶ Banking and insurance: With a strong focus on high availability, transactions, and security
- ▶ Government: Strong focus on high availability and security
- ▶ Retail: Strong focus on scalability and coping with peak loads
- ▶ Cloud and computing services: Strong focus on high availability and variable load / scalability

5.2 IBM z/VM One LPAR cluster implementation

In this section, we describe the procedure to deploy a Red Hat OpenShift cluster on an IBM z/VM LPAR. The architecture was briefly described in 3.3.1, “Single-LPAR deployment configuration” on page 26.

5.2.1 Resource planning

As mentioned in Chapter 3, “Implementation architectural considerations” on page 21, we deploy the Load Balancer services and DNS separately as we choose not to host them on the bastion node. Our LPAR cluster resource planning is shown in Table 5-1.

Table 5-1 One LPAR cluster resources planning

Purpose	OS	VM Name	CPU	Memory (Min/Max)	Size (Cyl) - 40G	DASD Grp
Bastion	RHEL9.1	BASTION2	4	24G/256G	50150	DASD400
DNS	RHEL9.1	DNS2	4	24G/256G	50150	DASD400
HAPROXY	RHEL9.1	HAPROXY2	4	24G/256G	50150	DASD400
Bootstrap	RHCOS4.12.10	OCP2B0	4	24G/256G	50150	DASD400
Controller1	RHCOS4.12.10	OCP2M1	4	24G/256G	50150	DASD400
Controller2	RHCOS4.12.10	OCP2M2	4	24G/256G	50150	DASD400
Controller3	RHCOS4.12.10	OCP2M3	4	24G/256G	50150	DASD400
Worker1	RHCOS4.12.10	OCP2W1	4	24G/256G	50150	DASD400
Worker2	RHCOS4.12.10	OCP2W2	4	24G/256G	50150	DASD400
Infra1	RHCOS4.12.10	OCP2F1	4	24G/256G	50150	DASD400
Infra2	RHCOS4.12.10	OCP2F2	4	24G/256G	50150	DASD400

Our network resource planning is shown in Table 5-2. In our lab environment, we use an IBM z/VM VSWITCH for the network connectivity. To configure RoCE cards, see 5.3, “IBM z/VM three-LPAR cluster implementation” on page 64.

Table 5-2 Network resources

Purpose	VM Name	VSWITCH	IP Address	OCP Domain Name
Bastion	BASTION2	VSWITCH1	9.76.62.150	bastion2.zvm2.rdbk.com
DNS	DNS2	VSWITCH1	9.76.62.146	dns2.zvm2.rdbk.com
HAPROXY	HAPROXY2	VSWITCH1	9.76.62.145	haproxy2.zvm2.rdbk.com
Bootstrap	OCP2B0	VSWITCH1	9.76.62.160	ocp2b0.zvm2.rdbk.com
Controller1	OCP2M1	VSWITCH1	9.76.62.151	ocp2m1.zvm2.rdbk.com
Controller2	OCP2M2	VSWITCH1	9.76.62.152	ocp2m2.zvm2.rdbk.com
Controller3	OCP2M3	VSWITCH1	9.76.62.153	ocp2m3.zvm2.rdbk.com
Worker1	OCP2W1	VSWITCH1	9.76.62.154	ocp2w1.zvm2.rdbk.com
Worker2	OCP2W2	VSWITCH1	9.76.62.155	ocp2w2.zvm2.rdbk.com
Infra1	OCP2F1	VSWITCH1	9.76.62.156	ocp2f1.zvm2.rdbk.com
Infra2	OCP2F2	VSWITCH1	9.76.62.157	ocp2f2.zvm2.rdbk.com

5.2.2 DNS configuration

There are challenges in deploying DNS in our Red Hat OpenShift environment. In a production environment, you typically use your enterprise DNS servers. In our lab environment, we configured the `named.conf` file with our DNS server. Example 5-1 shows our lab environment’s content for our `62.76.9.in-addr.arpa.zone`, which is a part of the `named.conf` file.

Example 5-1 Sample content for `62.76.9.in-addr.arpa.zone`

```

151 IN PTR ocp2m1.zvm2.rdbk.com.
152 IN PTR ocp2m2.zvm2.rdbk.com.
153 IN PTR ocp2m3.zvm2.rdbk.com.
154 IN PTR ocp2w1.zvm2.rdbk.com.
155 IN PTR ocp2w2.zvm2.rdbk.com.
156 IN PTR ocp2f1.zvm2.rdbk.com.
157 IN PTR ocp2f2.zvm2.rdbk.com.
160 IN PTR ocp2b0.zvm2.rdbk.com.

```

Example 5-2 shows our sample content for the `zvm2.rdbk.com.zone`.

Example 5-2 Sample content for `zvm2.rdbk.com.zone`

```
dns2 IN A 9.76.62.146
bastion          IN A 9.76.62.150
api              IN A 9.76.62.145
api-int         IN A 9.76.62.145
apps            IN A 9.76.62.145
*.apps          IN A 9.76.62.145
ocp2b0          IN A 9.76.62.160
ocp2m1          IN A 9.76.62.151
ocp2m2          IN A 9.76.62.152
ocp2m3          IN A 9.76.62.153
ocp2w1          IN A 9.76.62.154
ocp2w2          IN A 9.76.62.155
ocp2f1          IN A 9.76.62.156
ocp2f2          IN A 9.76.62.157
```

5.2.3 HAPROXY configuration

For the load balancer in our lab environment, we configure the `HAProxy.conf` file and enable port 1936 for HAProxy statistics monitoring. An example of your load balancer link would have the following pattern, where `HAProxy_ip_address` is your HAProxy IP address:

```
http://haproxy_ip_address/haproxy?stats
```

Example 5-3 shows our lab environment's `HAProxy.cfg` content.

Example 5-3 Sample content for `HAProxy.cfg`

```
listen  stats
        bind 0.0.0.0:1936
        mode          http
        log           global
        maxconn 10
        timeout queue 100s
        stats enable
        stats hide-version
        stats refresh 30s
        stats show-node
        stats auth admin:password
        stats uri /haproxy?stats

global
  log          127.0.0.1 local2
  chroot      /var/lib/haproxy
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  user        HAProxy
  group       HAProxy
  daemon
  #stats socket /var/lib/haproxy/stats
  ssl-default-bind-ciphers PROFILE=SYSTEM
  ssl-default-server-ciphers PROFILE=SYSTEM

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 30m
timeout server 30m
timeout http-keep-alive 10s
timeout check 10s
maxconn 3000
frontend ocp4-kubernetes-api-server
mode tcp
option tcplog
bind api.zvm2.rdbk.com:6443
bind api-int.zvm2.rdbk.com:6443
default_backend ocp4-kubernetes-api-server
frontend ocp4-machine-config-server
mode tcp
option tcplog
bind api.zvm2.rdbk.com:22623
bind api-int.zvm2.rdbk.com:22623
default_backend ocp4-machine-config-server
frontend ocp4-router-http
mode tcp
option tcplog
bind apps.zvm2.rdbk.com:80
default_backend ocp4-router-http
frontend ocp4-router-https
mode tcp
option tcplog
bind apps.zvm2.rdbk.com:443
default_backend ocp4-router-https
backend ocp4-kubernetes-api-server
mode tcp
balance source
server ocp2b0 ocp2b0.zvm2.rdbk.com:6443 check
server ocp2m1 ocp2m1.zvm2.rdbk.com:6443 check
server ocp2m2 ocp2m2.zvm2.rdbk.com:6443 check
server ocp2m3 ocp2m3.zvm2.rdbk.com:6443 check
backend ocp4-machine-config-server
mode tcp
balance source
server ocp2b0 ocp2b0.zvm2.rdbk.com:22623 check
server ocp2m1 ocp2m1.zvm2.rdbk.com:22623 check
server ocp2m2 ocp2m2.zvm2.rdbk.com:22623 check
server ocp2m3 ocp2m3.zvm2.rdbk.com:22623 check
backend ocp4-router-http
mode tcp
server ocp2w1 ocp2w1.zvm2.rdbk.com:80 check
server ocp2w2 ocp2w2.zvm2.rdbk.com:80 check

```

```
server ocp2f1 ocp2f1.zvm2.rdbk.com:80 check
server ocp2f2 ocp2f2.zvm2.rdbk.com:80 check
backend ocp4-router-https
mode tcp
server ocp2w1 ocp2w1.zvm2.rdbk.com:443 check
server ocp2w2 ocp2w2.zvm2.rdbk.com:443 check
server ocp2f1 ocp2f1.zvm2.rdbk.com:443 check
server ocp2f2 ocp2f2.zvm2.rdbk.com:443 check
```

5.2.4 Ignition files and the HTTP server

In this section, we describe how to create Red Hat OpenShift Container Platform ignition files and setup an HTTP server for holding the rootfs image and ignition files.

Downloading RHOCF rootfs image to an HTTP server

The following steps assume that the HTTP server is up and running on the bastion node on port 8080.

1. Create a directory by using the following command:

```
/var/www/html/rootfs
```

2. Navigate to the newly created directory:

```
cd /var/www/html/rootfs
```

3. Perform the following command to download the file:

```
wget -O 4.12.10.rootfs.img
https://mirror.openshift.com/pub/openshift-v4/s390x/dependencies/rhcos/4.12/4.1
2.10/rhcos-4.12.10-s390x-live-rootfs.s390x.img
```

4. Verify the location of the file with the following command:

```
[root@bastion2 html]# tree rootfs
rootfs
|__ 4.12.10.rootfs.img
```

Downloading the client to the bastion node

The following steps enable you to download the Red Hat OpenShift Container Platform client to the bastion node.

1. Change the directory by using the following command:

```
cd /usr/local/bin/
```

2. Download the Red Hat OpenShift Container Platform client and installation utilities specific for your bastion operating system by using the following command:

```
curl -L
https://mirror.openshift.com/pub/openshift-v4/s390x/clients/ocp/4.12.10/
openshift-install-linux-4.12.10.tar.gz | tar xvz openshift-install
```

3. Download the Red Hat OpenShift command-line interface (oc) with the following command.

```
curl -L
https://mirror.openshift.com/pub/openshift-v4/s390x/clients/ocp/4.12.10/openshi
ft-client-linux-4.12.10.tar.gz | tar xvz oc
```

4. Change the file permission by using the following commands:

```
chmod +x /usr/local/bin/oc
chmod +x /usr/local/bin/openshift-install
```

5. Verify the location of the files by using the following command:

```
[root@bastion2 html]# tree /usr/local/bin/
/usr/local/bin/
|--oc
|--openshift-install
```

Create the ignition files

The Red Hat OpenShift Container Platform installation program creates the ignition configuration files that you need to deploy your cluster. The following steps provide instructions on how to create these ignition files.

1. Create an SSH key pair with the following command:

```
ssh-keygen -t rsa -b 2048 -N '' -C 'OCP-4-Admin' -f /root/.ssh/id_rsa
```

2. Get the pull secret file by logging into the following web site and download the pull secret, as shown in Figure 5-1.

<https://cloud.redhat.com/openshift/install/pull-secret>

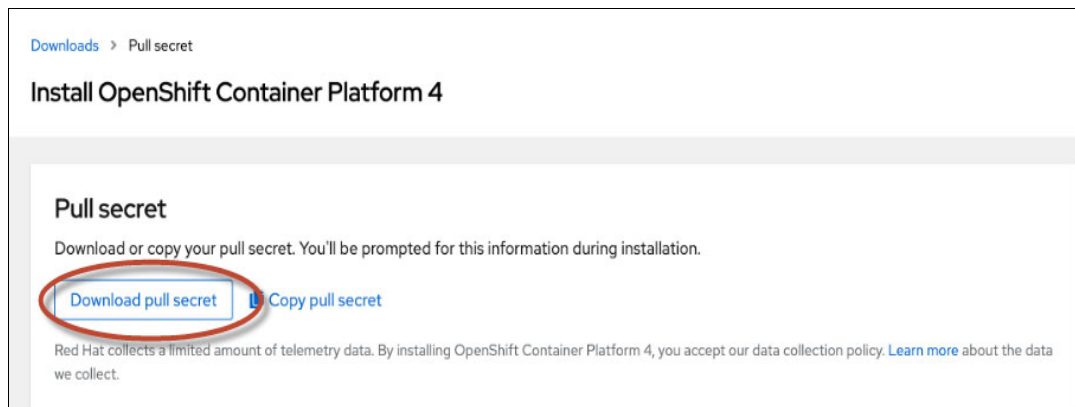


Figure 5-1 Get the pull secret

3. Save it to bastion node, for example, /root/ocp/pull-secret.

4. Generate install-config.yaml.

Refer to the sample shown in Example 5-4 as a template for the Ansible deployment. Change the variables sshKey and pullSecret, according to your own environment.

Example 5-4 Template for install-config.yaml

```
apiVersion: v1
baseDomain: "{{ cluster_base_domain }}"
{% if install_proxy is defined %}
proxy:
  httpProxy: http://{{ install_proxy }}:3128
  httpsProxy: http://{{ install_proxy }}:3128
  noProxy: .{{ cluster_domain_name }},169.254.169.254,{{ subnet_cidr }}
{% endif %}
compute:
- hyperthreading: Enabled
```

```
name: worker
replicas: 0
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: {{ cluster_nodes.masters | length }}
metadata:
  name: "{{ cluster_name }}"
networking:
  clusterNetworks:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  none: {}
pullSecret: '{"auths":{ your ocp4_pull_secret | to_json }}'
sshKey: '{{ bastion_pubkey.content | b64decode }}'
```

Example 5-5 shows the yaml file that we use in our lab environment. Insert your own pullSecret and sshKey, as ours have been masked in this example.

Example 5-5 Our lab environment install-config.rdbk1.yaml file

```
apiVersion: v1
baseDomain: "rdbk.com"
compute:
- hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
metadata:
  name: "zvm"
networking:
  clusterNetworks:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
pullSecret: '{"auths": {"cloud.openshift.com": {"auth":
"b3B1bnNoaWZ0LXJ1bGVhc2UtZGV2K2Ric19pdG8xYjZucGdudTVpNnhwZDVsbT1rdDV5OHVhZXE6Sj
lKWE9SQ0Q1WjczOFBOVkJVM1Y3MEs3U1ZBNjE2UTZFWU9CMEgxU0tSVExUV1hWSVZVW1NBOTYzNUwyU
FI3WQ=="}, "email": "nobody@ibm.com"}, "zxyz.io": {"auth":
"b3B1bnNoaWZ0LXJ1bGVhc2UtZGV2K2Ric19pdG8xYjZucGdudTVpNnhwZDVsbT1rdDV5OHVhZXE6Sj
lKWE9SQ0Q1WjczOFBOVkJVM1Y3MEs3U1ZBNjE2UTZFWU9CMEgxU0tSVExUV1hWSVZVW1NBOTYzNUwyU
FI3WQ=="}, "email": "nobody@ibm.com"}, "registry.connect.redhat.com": {"auth":
"NTMyNDMONj18dWhjLTFiNm5wR25VNUk2eFBkNUxNOwtONXk4VUFFcTpleUpoYkdjaU9pS1Nve1V4TW
lKOS5leUp6ZFdJaU9pSTBaak5tTjJZ", "email": "nobody@ibm.com"},
"registry.redhat.io": {"auth":
"NTMyNDMONj18dWhjLTFiNm5wR25VNUk2eFBkNUxNOwtONXk4VUFFcTpleUpoYkdjaU9pS1Nve1V4TW
lKOS5leUp6ZFdJaU9pSTBaak5tT", "email": "nobody@ibm.com"}, "registry:5000":
{"auth": "bX11c2Vy0m15cGFzc3dvcnQ=", "email": "me@working.me"}}}'
sshKey: 'ssh-rsa
AAAAB3NzaC1yc2EAAAAv/V3Sn1khLqguUY4W6xsBRe+akr/Gv4+sCrI6ABxfyl1S/OCP-4-Admin
'
```

5. Create manifests by using the following command:


```
/usr/local/bin/openshift-install --dir=~/.root/ocp` create manifests
```
6. Make the controller node unschedulable with the following command:


```
sed -i 's/true/false/g' /root/ocp/manifests/cluster-scheduler-02-config.yml
```
7. Create the ignition configuration files:


```
/usr/local/bin/openshift-install --dir=~/.root/ocp` create ignition-configs
```
8. Copy the ignition configuration files to the HTTP server by using the following commands:


```
cp /root/ocp/*.ign /var/www/html/ignition
chmod o+r /var/www/html/ignition/*.ign
```
9. Make the Red Hat OpenShift Container Platform environment ready on the bastion node with the following commands:


```
mkdir ~/.kube
cp /root/ocp/auth/kubeconfig ~/.kube/config
```

5.2.5 USER DIRECT and PARM files for OCP nodes

This step is only applicable to the IBM z/VM environment. In this section, we define the VM definitions and prepare the PARM files for OCP nodes, which contain settings and options that are used to customize the behavior of the VM or the IBM z/VM system.

USER DIRECT

The IBM z/VM user directory specifies the configuration and operating characteristics of VMs.

Example 5-6 shows our USER DIRECT file for the bootstrap node.

Example 5-6 Bootstrap node

```

USER OCP2B0 LBYONLY 24G 256G G
  INCLUDE INSTALL
  COMMAND TERM MORE 0 0
  COMMAND TERM HOLD OFF
  CPU 00 BASE
  CPU 01
  CPU 02
  CPU 03
  IPL CMS PARM AUTOCR
  LOGONBY MAINT IBMVM1
  MACHINE ESA 64
  NICDEF 1000 TYPE QDIO DEVICES 3 LAN SYSTEM VSWITCH1
  MDISK 0100 3390 330551 50150 RRLX01 MR

```

Example 5-7 shows our USER DIRECT file for the controller node.

Example 5-7 Controller node

```
USER OCP2M1 LBYONLY 24G 256G G
  INCLUDE INSTALL
  COMMAND TERM MORE 0 0
  COMMAND TERM HOLD OFF
  CPU 00 BASE
  CPU 01
  CPU 02
  CPU 03
  IPL CMS PARM AUTOOCR
  LOGONBY MAINT IBMVM1
  MACHINE ESA 64
  NICDEF 1000 TYPE QDIO DEVICES 3 LAN SYSTEM VSWITCH1
  MDISK 0100 3390 180301 50150 RRLX02 MR
```

Example 5-8 shows our USER DIRECT file for the compute node.

Example 5-8 Compute node

```
USER OCP2W1 LBYONLY 24G 256G G
  INCLUDE INSTALL
  COMMAND TERM MORE 0 0
  COMMAND TERM HOLD OFF
  CPU 00 BASE
  CPU 01
  CPU 02
  CPU 03
  IPL CMS PARM AUTOOCR
  LOGONBY MAINT IBMVM1
  MACHINE ESA 64
  NICDEF 1000 TYPE QDIO DEVICES 3 LAN SYSTEM VSWITCH1
  MDISK 0100 3390 380701 50150 RRLX01 MR
```

USER PARM

For each node, an ignition file has to be created. These files contain necessary configuration data for the Red Hat OpenShift Core OS and are typically created in a temporary directory, such as `/root/rhcos-bootfiles`. Example 5-9 shows our sample configuration data to boot the bootstrap.

Example 5-9 Bootstrap node

```
OCP2B0 PRM Z1 V 80 Trunc=80 Size=10 Line=0 Col=1 Alt=0
rd.neednet=1 dflttcc=off
console=ttysclp0
searchdomain=zvm2.rdbk.com
coreos.inst.install_dev=/dev/dasda
coreos.live.rootfs_url=http://9.76.62.150:8080/rootfs/4.12.10.rootfs.img
coreos.inst.ignition_url=http://9.76.62.150:8080/ignition/bootstrap.ign
ip=9.76.62.160::9.76.62.1:255.255.255.0::none nameserver=9.76.62.146
rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1,portno=0
rd.dasd=0.0.0100
```

Example 5-10 shows our sample configuration data to boot the control plane node.

Example 5-10 Controller node

```
OCP2M1 PRM Z1 V 80 Trunc=80 Size=10 Line=0 Col=1 Alt=0
rd.neednet=1 dfltcc=off
console=ttysclp0
searchdomain=zvm2.rdbk.com
coreos.inst.install_dev=/dev/dasda
coreos.live.rootfs_url=http://9.76.62.150:8080/rootfs/4.12.10.rootfs.img
coreos.inst.ignition_url=http://9.76.62.150:8080/ignition/master.ign
ip=9.76.62.151::9.76.62.1:255.255.255.0:::none nameserver=9.76.62.146
rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1,portno=0
rd.dasd=0.0.0100
```

Example 5-11 shows our sample configuration data to boot the compute node.

Example 5-11 Compute node

```
OCP2W1 PRM Z1 V 80 Trunc=80 Size=10 Line=0 Col=1 Alt=0
rd.neednet=1 dfltcc=off
console=ttysclp0
searchdomain=zvm2.rdbk.com
coreos.inst.install_dev=/dev/dasda
coreos.live.rootfs_url=http://9.76.62.150:8080/rootfs/4.12.10.rootfs.img
coreos.inst.ignition_url=http://9.76.62.150:8080/ignition/worker.ign
ip=9.76.62.154::9.76.62.1:255.255.255.0:::none nameserver=9.76.62.146
rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1,portno=0
rd.dasd=0.0.0100
```

USER DIRECT OCP kernel and initrd files

Example 5-12 is the sample REXX EXEC script for booting up the VM.

Example 5-12 REXX EXEC script to boot the VM

```
OCP41210 EXEC Z1 V 130 Trunc=130 Size=9 Line=0 Col=1 Alt=0
/* */
'CL RDR'
'PURGE RDR ALL'
'SPOOL PUNCH * RDR'
'PUNCH OCP41210 KERNEL A (NOH'
'PUNCH ' userid() ' PRM A (NOH'
'PUNCH OCP41210 INITRD A (NOH'
'CH RDR ALL KEEP NOHOLD'
'I OOC'
```

Upload the following two files to IBM z/VM. They should be transferred in binary format with a fixed record length of 80.

OCP41210	INITRD	Z1	F	80	877284	17135	4/28/23	12:41:02
OCP41210	KERNEL	Z1	F	80	94163	1818	4/28/23	12:41:25

5.2.6 Building the Red Hat OpenShift Container Platform cluster

In this section, we describe the next step: to build the Red Hat OpenShift Container Platform cluster. When using a single LPAR cluster, you only need to log into one IBM z/VM system to start up the virtual machines:

1. Start up the bootstrap node by using the following command, where OCP2B0 is your VM name. For more information, see Table 5-1 on page 43.

```
XAUTOLOG OCP2B0
```

2. Check if Bootstrap node is ready by using the following command:

```
openshift-install --dir <installation_directory> wait-for bootstrap-complete  
--log-level=info
```

3. Check if port 22623 is ready:

```
curl https://<haproxy_ip>:22623/config/master
```

4. Start up the Controller and compute nodes, as shown in Example 5-13.

Example 5-13 Start up the nodes

```
XAUTOLOG OCP2M1  
XAUTOLOG OCP2M2  
XAUTOLOG OCP2M3  
XAUTOLOG OCP2W1  
XAUTOLOG OCP2W2  
XAUTOLOG OCP2F1  
XAUTOLOG OCP2F2
```

5. During the OCP building process, approve the Certificate Signing Requests. Use the following command:

```
oc get csr -o name | xargs /usr/local/bin/oc adm certificate approve
```

Common ways to check the build process

In this section, we provide commands and the URL for checking the progress of your build.

Commands:

```
oc get nodes  
oc get co  
openshift-install --dir <installation_directory> wait-for install-complete
```

Browser:

```
http://<haproxy_ip>:1936/haproxy?stats
```

Note: The user id and password for the website is admin and password.

5.2.7 Using Ansible playbooks

In this section, we describe how to leverage Ansible playbooks to automatically deploy a Red Hat OpenShift Cluster in an IBM z/VM environment.

Prerequisites

The following are prerequisites for enabling Ansible automated deployment:

1. Ansible controller: The Ansible controller could be on any platform (LinuxONE, x86, power, Mac), as long as the Ansible controller node can connect to the target controlled environment. In our case, we use the following commands to install Ansible:

```
yum -y install python39 python39-pip
yum -y install gcc libffi-devel python-devel OpenSSL-devel cargo
pip3 install cryptography
pip3 install netaddr paramiko
pip3 install ansible
```

There are many ways to set up authentication between the Ansible controller node and target controlled systems. The following commands are an example:

```
ssh-keygen
ssh-copy-id 9.76.62.149 (zvmagent)
ssh-copy-id 9.76.62.146 (dns)
ssh-copy-id 9.76.62.145 (haproxy)
```

2. IBM z/VM configuration: For each controlled IBM z/VM system, we need to create some small Linux virtual machines. In our case, we create the following profiles and virtual machines on IBM z/VM.

The INSTALL profile (Example 5-14) is used for installing Red Hat OpenShift nodes. After successfully installing the nodes, we configure the profiles to include the LNXDFLT profile.

Example 5-14 Install profile

```
PROFILE INSTALL
COMMAND SET RUN ON
COMMAND SET VSWITCH VSWITCH1 GRANT &USERID
CPU 00 BASE
CPU 01
IPL CMS PARM AUTOOCR
LOGONBY IBMVM1
MACHINE ESA 64
OPTION APPLMON
CONSOLE 0009 3215 T MAINT
NICDEF 1000 TYPE QDIO LAN SYSTEM VSWITCH1
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK LNXMAINT 0192 0191 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
```

The LNXDFLT profile (Example 5-15) is used for regular Linux use, not for installation.

Example 5-15 LNXDFLT profile

```
PROFILE LNXDFLT
  COMMAND SET RUN ON
  COMMAND SET VSWITCH VSWITCH1 GRANT &USERID
  CPU 00 BASE
  CPU 01
  IPL CMS
  LOGONBY IBMVM1
  MACHINE ESA 64
  OPTION CHPIDV ONE APPLMON
  CONSOLE 0009 3215 T
  NICDEF 1000 TYPE QDIO LAN SYSTEM VSWITCH1
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
```

The ZVMAGENT (Example 5-16) links the LNXMAINT 192 minidisk, where we put the kernel, the initrd disk, and all the PARM files for installing the Linux Red Hat Core OS.

Example 5-16 ZVMAGENT profile

```
USER ZVMAGENT LBYONLY 4G 64G      G
  INCLUDE LNXDFLT
  COMMAND SET VCONFIG MODE LINUX
  IPL 0100
  IUCV ANY
  LOGONBY IBMVM1
  MACHINE ESA 32
  LINK LNXMAINT 0192 0192 MR
  MDISK 0100 3390 30051 10016 RRLX01 MR
```

We granted the System Management API (SMAPI) authorization to ZVMAGENT by editing the file VSMWORK1 AUTHLIST and adding ZVMAGENT at the end, as the following shows:

```
VSMWORK1 AUTHLIST P1 F 195 Trunc=195 Size=4 Line=0 Col=1 Alt=0
DO.NOT.REMOVE
DO.NOT.REMOVE
MAINT                                     ALL
IBMVM1                                    ALL
ZVMAGENT                                  ALL
```

3. IBM z/VM Agent: ZVMAGENT is a virtual machine that is used for executing `smcli` commands (an IBM z/VM SMAPI command line utility) to define USERID, and the XAUTOLOG virtual machine on IBM z/VM. Very limited resources are required for this IBM z/VM agent. You generally only need one virtual Central Processing Unit (vCPU), 4 GB memory, and a 10 GB volume.
 - a. Set up read-write to the LNXMAINT 192 disk from ZVMAGENT.

Note: We use the `cmsfs` utility (the CMS file system package that allows access to CMS files on CMS minidisks that are owned or linked by a Linux on IBM Z guest on IBM z/VM) to mount the IBM z/VM PARM disk to the Linux Operating System, where Ansible creates PARM files. There are other options available to achieve the same purpose such as using a VMNFS server on IBM z/VM, and exported to Linux. For more details on how to setup NFS server on IBM z/VM, please refer to [TCP/IP Planning and Customization](#).

Upload the following two files to the LNXMAINT 192 disk. They should be transferred in BINARY format and FIX record length 80:

```
OCP41210 INITRD    Z1 F          80      877284      17135  4/28/23 12:41:02
OCP41210 KERNEL   Z1 F          80      94163       1818  4/28/23 12:41:25
```

Filename OCP41210 means that we are installing OCP 4.12.10.

There are some ways to avoid uploading these two files to IBM z/VM beforehand, but this activity is required one time only. In our case, we upload them to the LNXMAINT 192 disk. Other options are, but not limited to:

- ▶ Leverage the `vmur` utility (from the `s390-tools` package) to punch the above two files into the IBM z/VM reader. For more information, see [Preparing the z/VM reader as an IPL device for Linux](#).
- ▶ `znetboot` makes it easy to bring up Linux on IBM z/VM by issuing the command `znetboot`. For more information, see [znetboot](#).

The commands shown in Example 5-17 make the LNXMAINT 192 minidisk accessible from Linux.

Example 5-17 Make LNXMAINT accessible from Linux-commands and output

```
[root@zvmagent ~]# cat /etc/rc.local
#!/bin/bash
...
cmsfs-fuse -a -o noauto_cache -o rw /dev/dasdb /mnt/lnxmaint

[root@zvmagent ~]# systemctl status rc-local
? rc-local.service - /etc/rc.d/rc.local Compatibility
   Loaded: loaded
   (;;file:///zvmagent/usr/lib/systemd/system/rc-local.service/usr/lib/systemd/s
system/rc-local.service;;; enabled-runtime; vendor preset: d>
   Active: active (running) since Sat 2023-05-13 22:02:58 CDT; 1h 8min ago
   Docs:
   ;;man:systemd-rc-local-generator(8)man:systemd-rc-local-generator(8);;
   Process: 1228 ExecStart=/etc/rc.d/rc.local start (code=exited,
status=0/SUCCESS)
   Tasks: 1 (limit: 23651)
```

```
Memory: 832.0K
CPU: 9ms
CGroup: /system.slice/rc-local.service
??1269 cmsfs-fuse -a -o noauto_cache -o rw /dev/dasdb
/mnt/lnxmaint
```

```
May 13 22:02:58 zvmagent systemd[1]: Starting /etc/rc.d/rc.local
Compatibility...
```

```
May 13 22:02:58 zvmagent systemd[1]: Started /etc/rc.d/rc.local
Compatibility.
```

```
[root@zvmagent ~]# df -h
```

```
Filesystem                Size  Used Avail Use% Mounted on
...
/dev/dasdb                  704M  128M  576M  19% /mnt/lnxmaint
...
```

b. Install smcli with the following commands:

```
yum install git make gcc
```

```
git clone https://github.com/openmainframeproject/feilong.git
```

In our lab environment, we used the following command for RHEL 9.1:

```
ln -s /usr/lib64/libtirpc.so.3 /usr/lib64/libtirpc.so
```

Issue the following command to install smcli:

```
cd feilong/zthin-parts; OS_IS_RHEL8=1 make; make install; make post
```

Verify smcli is working:

```
/opt/zthin/bin/smcli Image_Query_DM -T MAINT730
```


Deploying a cluster with Ansible playbook

In this section, we provide step-by-step instructions to deploy a cluster with an Ansible playbook.

1. Download the Ansible Playbooks and unzip using the following commands:

```
git clone
https://github.com/IBMRedbooks/REDP5711-Red-Hat-OpenShift-on-IBM-zSystems-and-L
inuxONE.git
cd REDP5711-Red-Hat-OpenShift-on-IBM-zSystems-and-LinuxONE
tar -zxvf chapter_5_ansible_on_linuxone_rdbk-main.tar.gz
```

2. Modify the variables with the following commands:

```
cd ansible_on_linuxone_rdbk-main
vi groups_vars/all.yml
```

In this sample, we prepare Linux OS for Bastion, DNS, and HAProxy. To put all these services onto a single system, please specify the same IP address for following variables:

```
dns_nameserver:
haproxy_server:
bastion_public_ip_address:
bastion_private_ip_address:
```

Create your cluster using the configuration file show in Example 5-18.

Example 5-18 Create the configuration file for your cluster

```
---
### Environment #####
out_path: /root/zvm
workdir: /root/ocp
bootfile_dest: "{{workdir}}"

##### z/VM Settings #####
load_address: 100
install_mode: "zvm"
guest_profile: "LNXDFLT"
zvm_vsw_name: "VSWITCH1"

### For NFS persistent Storage
nfs_server: "9.76.62.150"
nfs_share_folder: "/nfs_share"

##### OCP #####

### Pull-secret
path_ps:
"/REDP5711-Red-Hat-OpenShift-on-IBM-zSystems-and-LinuxONE/ansible_on_linuxone_r
dbk-main/test1_pull_secret"
ocp4_pull_secret: "{{ lookup('file', path_ps) | from_json }}"

### OCP parameters
controller_schedulable: false
cluster_name: "zvm2"
cluster_base_domain: "rdbk.com"
cluster_domain_name: "{{cluster_name}}.{{cluster_base_domain}}"
dns_nameserver: "9.76.62.146"
```

```
dns_hostname: "dns1"
haproxy_server: "9.76.62.145"
subnet_gateway: "9.76.62.1"
subnet_netmask: "255.255.255.0"
subnet_cidr: "9.76.62.0/24"
bastion_public_ip_address: "9.76.62.150"
bastion_private_ip_address: "9.76.62.150"
ocp_repo_type: "Online"
ocp_idp: "RDBK2_HTPASSWD"
ocp_idp_users_secret: "users-secret"
ocp_version: "4.12"
ocp_minor_version: "4.12.10"
rhcos_version: "4.12.10-s390x"
```

```
cluster_nodes: {
  bootstrap: [
    { hostname: "ocp2b0",
      ip: "9.76.62.160",
      ign_profile: "bootstrap.ign",
      vm_ubuntu: "OCP2B0",
      eckd_disk: ["rd.dasd=0.0.0100"],
      vol_type: "3390",
      vol_size: "50150",
      vol_grp: "DASD400",
      vcpu: 4,
      min_mem: "24G",
      max_mem: "256G",
      install_mode: "zvm",
      kvm_network: "",
      vm_node: "RDBKZVMD",
      vm_agent: "9.76.62.149"
    }
  ],
  masters: [
    { hostname: "ocp2m1",
      ip: "9.76.62.151",
      ign_profile: "master.ign",
      vm_ubuntu: "OCP2M1",
      eckd_disk: ["rd.dasd=0.0.0100"],
      install_mode: "zvm",
      vol_type: "3390",
      vol_size: "50150",
      vol_grp: "DASD400",
      vcpu: 4,
      min_mem: "24G",
      max_mem: "256G",
      kvm_network: "",
      vm_node: "RDBKZVMD",
      vm_agent: "9.76.62.149"
    },
    { hostname: "ocp2m2",
      ip: "9.76.62.152",
      ign_profile: "master.ign",
      vm_ubuntu: "OCP2M2",
      eckd_disk: ["rd.dasd=0.0.0100"],
      install_mode: "zvm",
```

```

    vol_type: "3390",
    vol_size: "50150",
    vol_grp: "DASD400",
    vcpu: 4,
    min_mem: "24G",
    max_mem: "256G",
    kvm_network: "",
    vm_node: "RDBKZVMD",
    vm_agent: "9.76.62.149"
  },
  { hostname: "ocp2m3",
    ip: "9.76.62.153",
    ign_profile: "master.ign",
    vm_undef: "OCP2M3",
    eckd_disk: ["rd.dasd=0.0.0100"],
    install_mode: "zvm",
    vol_type: "3390",
    vol_size: "50150",
    vol_grp: "DASD400",
    vcpu: 4,
    min_mem: "24G",
    max_mem: "256G",
    kvm_network: "",
    vm_node: "RDBKZVMD",
    vm_agent: "9.76.62.149"
  }
}],
workers: [
  {hostname: "ocp2w1",
    ip: "9.76.62.154",
    ign_profile: "worker.ign",
    vm_undef: "OCP2W1",
    eckd_disk: ["rd.dasd=0.0.0100"],
    install_mode: "zvm",
    vol_type: "3390",
    vol_size: "50150",
    vol_grp: "DASD400",
    vcpu: 4,
    min_mem: "24G",
    max_mem: "256G",
    kvm_network: "",
    vm_node: "RDBKZVMD",
    vm_agent: "9.76.62.149"
  },
  {hostname: "ocp2w2",
    ip: "9.76.62.155",
    ign_profile: "worker.ign",
    vm_undef: "OCP2W2",
    eckd_disk: ["rd.dasd=0.0.0100"],
    install_mode: "zvm",
    vol_type: "3390",
    vol_size: "50150",
    vol_grp: "DASD400",
    vcpu: 4,
    min_mem: "24G",
    max_mem: "256G",
  }
]

```

```

kvm_network: "",
vm_node: "RDBKZVMD",
vm_agent: "9.76.62.149"
},
{hostname: "ocp2f1",
ip: "9.76.62.156",
ign_profile: "worker.ign",
vm_uname: "OCP2F1",
eckd_disk: ["rd.dasd=0.0.0100"],
install_mode: "zvm",
vol_type: "3390",
vol_size: "50150",
vol_grp: "DASD400",
vcpu: 4,
min_mem: "24G",
max_mem: "256G",
kvm_network: "",
vm_node: "RDBKZVMD",
vm_agent: "9.76.62.149"
},
{hostname: "ocp2f2",
ip: "9.76.62.157",
ign_profile: "worker.ign",
vm_uname: "OCP2F2",
eckd_disk: ["rd.dasd=0.0.0100"],
install_mode: "zvm",
vol_type: "3390",
vol_size: "50150",
vol_grp: "DASD400",
vcpu: 4,
min_mem: "24G",
max_mem: "256G",
kvm_network: "",
vm_node: "RDBKZVMD",
vm_agent: "9.76.62.149"
},
]}

```

-
3. Prepare the pull secret, depending on the variable that you specify in Step 2. In our case, we change the following variable:

```

path_ps:
"/REDP5711-Red-Hat-OpenShift-on-IBM-zSystems-and-LinuxONE/ansible_on_linuxone_r
dbk-main/test1_pull_secret"

```

We upload the pull secret file to the directory on the Ansible Controller:

```

/REDP5711-Red-Hat-OpenShift-on-IBM-zSystems-and-LinuxONE/ansible_on_linuxone_r
dbk-main/test1_pull_secret

```

4. Run the Ansible playbooks to deploy Red Hat OpenShift Container Platform:

```
cd ansible_on_linuxone_rdbk-main
ansible-playbook -i inventory -c paramiko -e @secrets_file.enc
--vault-password-file password_file playbooks/rdbk/module_ocp_deploy_zadmin.yml
```

Note: If you do not have sensitive information (such as a password) in the `secrets_file.enc`, the command could be shorter, for example:

```
ansible-playbook -i inventory -c paramiko
playbooks/rdbk/module_ocp_deploy_zadmin.yml
```

- Approve the Certificate Signing Request:

```
ansible-playbook -i inventory -c paramiko playbooks/rdbk/configure-csr.yml
```

Important: After you verify the Red Hat OpenShift Cluster is running, execute the following Ansible playbook to update USER DIRECT and installation status and log in to IBM z/VM to make sure that the USER DIRECT for the cluster node is using IPL 100 (load address), instead of IPL CMS PARM AUTOOCR.

```
ansible-playbook -i inventory -c paramiko
playbooks/rdbk/module_ocp_update_user_direct_zadmin.yml
```

5. Run the Ansible playbook For Day2 Operations:

```
cd ansible_on_linuxone_rdbk-main
```

- a. Disable Transparent Huge Pages (THP):

```
ansible-playbook -i inventory -c paramiko
playbooks/rdbk/ocp-day2-disable-thp.yml
```

- b. Enable Network Receive Flow Steering (RFS):

```
ansible-playbook -i inventory -c paramiko
playbooks/rdbk/ocp-day2-network-rfs-control.yml
ansible-playbook -i inventory -c paramiko
playbooks/rdbk/ocp-day2-network-rfs-compute.yml
```

- c. Set up the NFS-Client for the automated Persistent Volume Claim (PVC):

```
ansible-playbook -i inventory -c paramiko
playbooks/rdbk/ocp-day2-nfs-client.yml
```

- d. Enable NFS as the default Storage Class:

```
ansible-playbook -i inventory -c paramiko
playbooks/rdbk/ocp-day2-default-sc.yml
```

- e. Move Image Registry to NFS storage:

```
ansible-playbook -i inventory -c paramiko
playbooks/rdbk/ocp-day2-image-registry.yml
```

- f. Create HTPASSWD user credential:

This step creates the user id `admin001` as cluster-admin and the password is specified in the encrypted `secrets_file.enc`. We can issue the **`ansible-vault edit secrets_file.enc`** command to change the password.

```
ansible-playbook -i inventory -c paramiko -e @secrets_file.enc
--vault-password-file password_file playbooks/rdbk/ocp-day2-user.yml
```

- g. Disable the kubeadmin user:


```
ansible-playbook -i inventory -c paramiko
playbooks/rdbk/ocp-day2-disable-kubeadmin.yml
```
- h. Back up Etcd database:


```
ansible-playbook -i inventory -c paramiko
playbooks/rdbk/ocp-day2-backup-etcd.yml -e
'controller_node=ocp1m1.zvm.rdbk.com'
```
- i. Finally, copy to the bastion node:


```
scp -rp core@ocp1m1.zvm.rdbk.com:/home/core/ocpbkp /tmp/
```

5.3 IBM z/VM three-LPAR cluster implementation

In this section, we describe the procedure to deploy a Red Hat OpenShift cluster on three IBM z/VM LPARs. Most of the steps are the same as one LPAR cluster; therefore, we only describe the steps that differ when we deploy to a production environment.

5.3.1 Architecture

The three-LPAR cluster architecture is shown in Figure 5-2 and was briefly described in 3.3.2, “Three-LPAR deployment configuration” on page 27.

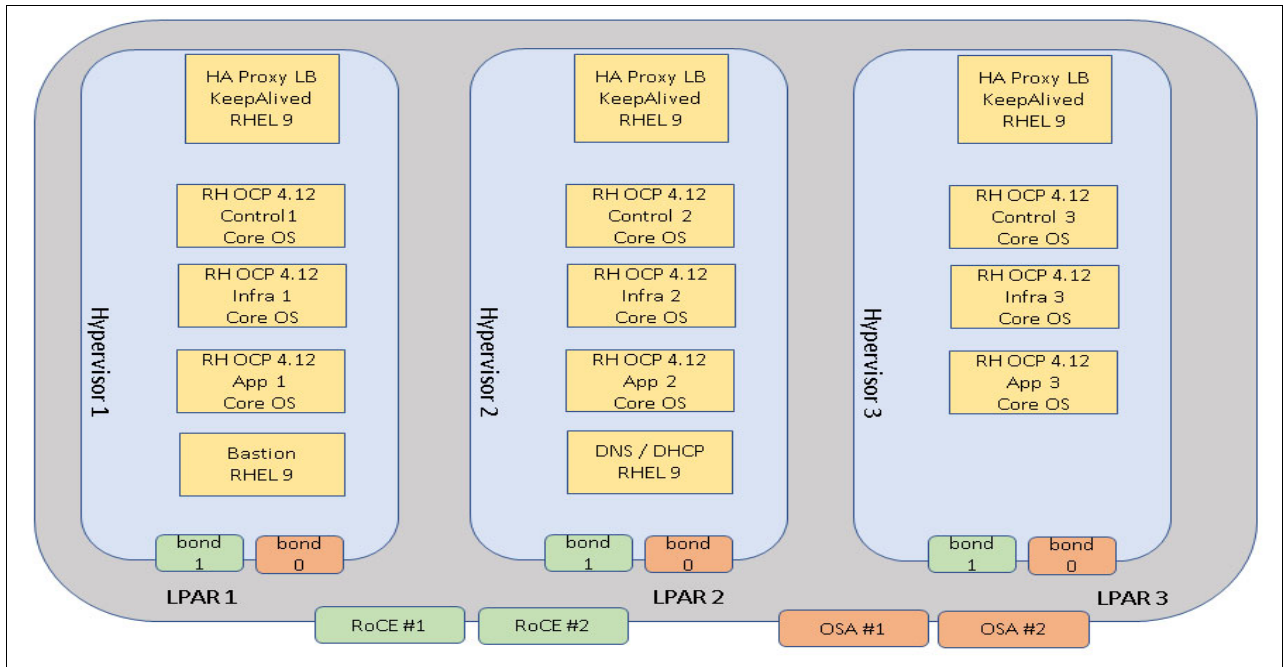


Figure 5-2 Three LPAR cluster architecture

5.3.2 Planning resources

As mentioned in Chapter 3, “Implementation architectural considerations” on page 21, we deploy three HAProxy servers for high availability, and all the nodes are distributed on three LPARs.

The three-LPAR cluster resources planning is shown in Table 5-3.

Table 5-3 Three-LPAR cluster resources planning

Purpose	VM Name	Node	CPU	Memory (Min/Max)	Size (Cyl) - 48G	DASD Grp
Bastion	BASTION2	RDBKZVMA	4	24G/256G	60101	DASD54
DNS	DNS1	RDBKZVMB	4	24G/256G	60101	DASD54
Haproxy1	HAPROXY1	RDBKZVMA	4	24G/256G	60101	DASD54
Haproxy2	HAPROXY2	RDBKZVMB	4	24G/256G	60101	DASD54
Haproxy3	HAPROXY3	RDBKZVMC	4	24G/256G	60101	DASD54
Bootstrap	OCP2B0	RDBKZVMA	4	24G/256G	60101	DASD54
Controller1	OCP2M1	RDBKZVMA	4	24G/256G	60101	DASD54
Controller2	OCP2M2	RDBKZVMB	4	24G/256G	60101	DASD54
Controller3	OCP2M3	RDBKZVMC	4	24G/256G	60101	DASD54
Worker1	OCP2W1	RDBKZVMA	4	24G/256G	60101	DASD54
Worker2	OCP2W2	RDBKZVMB	4	24G/256G	60101	DASD54
Worker3	OCP2W3	RDBKZVMC	4	24G/256G	60101	DASD54
Infra1	OCP2F1	RDBKZVMA	4	24G/256G	60101	DASD54
Infra2	OCP2F2	RDBKZVMB	4	24G/256G	60101	DASD54
Infra3	OCP2F3	RDBKZVMC	4	24G/256G	60101	DASD54

Network resource planning is shown in Table 5-4. In our lab environment, we use RoCE.

Table 5-4 Network resources

Purpose	OS	VM Name	PCI Functions	IP Address	OCP Domain Name
Bastion	RHEL 9.1	BASTION2		9.76.62.150	bastion.zvm.rdbk.com
DNS	RHEL 9.1	DNS2		9.76.62.141	dns1.zvm.rdbk.com
HAPROXY1	RHEL 9.1	HAPROXY1		9.76.62.145	haproxy1.zvm.rdbk.com
HAPROXY2	RHEL 9.1	HAPROXY2		9.76.62.142	haproxy2.zvm.rdbk.com
HAPROXY3	RHEL 9.1	HAPROXY3		9.76.62.143	haproxy3.zvm.rdbk.com
Bootstrap	RHCOS4.12.10	OCP2B0	00003425	9.76.62.140	ocp2b0.zvm.rdbk.com
Controller1	RHCOS4.12.10	OCP2M1	00003125	9.76.62.131	ocp2m1.zvm.rdbk.com
Controller2	RHCOS4.12.10	OCP2M2	00003126	9.76.62.132	ocp2m2.zvm.rdbk.com
Controller3	RHCOS4.12.10	OCP2M3	00003127	9.76.62.133	ocp2m3.zvm.rdbk.com
Worker1	RHCOS4.12.10	OCP2W1	00003525	9.76.62.134	ocp2w1.zvm.rdbk.com
Worker2	RHCOS4.12.10	OCP2W2	00003526	9.76.62.135	ocp2w2.zvm.rdbk.com
Worker3	RHCOS4.12.10	OCP2W3	00003527	9.76.62.136	ocp2w2.zvm.rdbk.com
Infra1	RHCOS4.12.10	OCP2F1	00003725	9.76.62.137	ocp2f1.zvm.rdbk.com
Infra2	RHCOS4.12.10	OCP2F2	00003726	9.76.62.138	ocp2f2.zvm.rdbk.com
Infra3	RHCOS4.12.10	OCP2F3	00003727	9.76.62.139	ocp2f2.zvm.rdbk.com

5.3.3 HAPROXY configuration

For the load balancer, in our lab environment, we leverage the Keepalived routing software to archive high availability of the HAProxy services.

Sample high availability architecture is shown in Figure 5-3.

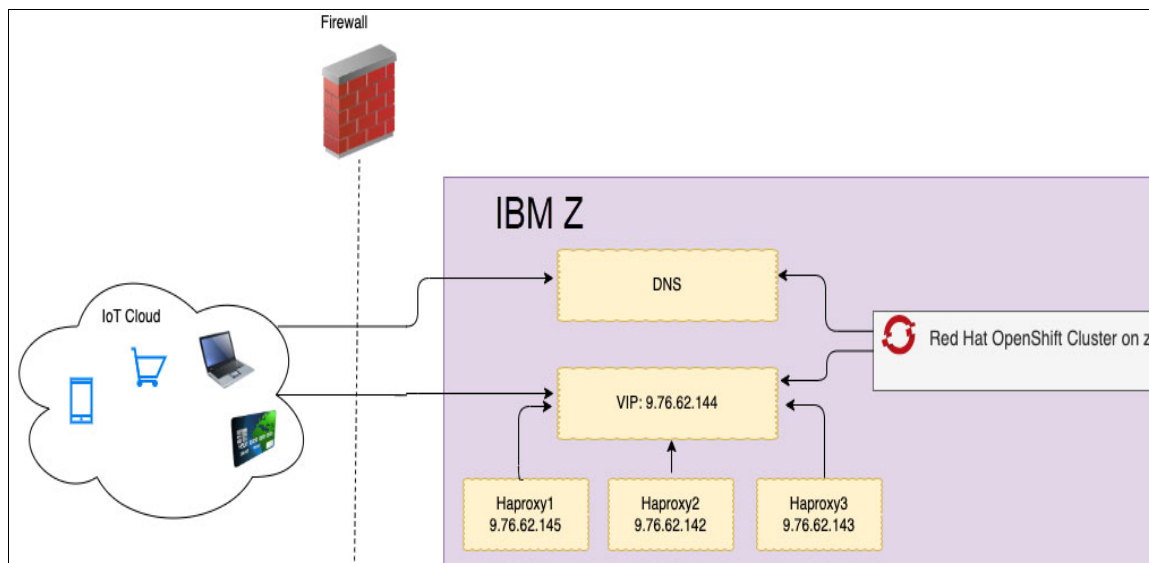


Figure 5-3 HAProxy service high availability with Keepalived

Keepalived installation

To install the Keepalived software package, follow these instructions:

1. Install packages:

```
yum install keepalived psmisc
```

2. Enable IP forwarding:

```
sysctl -w net.ipv4.ip_forward="1"
sysctl -w net.ipv4.ip_nonlocal_bind="1"
```

3. Update `/etc/keepalived/keepalived.conf` configuration file with the values shown in the following examples:

Here is an example on HAProxy1: (Example 5-19)

Example 5-19 Primary

```
global_defs {
    notification_email {
        sysadmin@firewall.loc
    }
    vrrp_version 3
    notification_email_from sysadmin@firewall.loc
    smtp_server 192.168.200.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
    vrrp_skip_check_adv_addr
    #vrrp_strict
    vrrp_garp_interval 0
    vrrp_gna_interval 0
    process_names
    enable_script_security
    script_user root
}
vrrp_script chk_haproxy {
    script "/usr/bin/systemctl is-active --quiet HAProxy"
    fall 2 # 2 fails required for failure
    rise 2 # 2 OKs required to consider the
          # process up after failure
    interval 5 # check every 5 seconds
    weight 51 # add 50 points rc=0
}
vrrp_instance VI_1 {
    state MASTER
    interface enc1000
    virtual_router_id 1
    priority 100
    advert_int 3
    # check that 10er network is up
    track_interface {
        enc1000 weight 50
    }
    # check that HAProxy is up
    track_script {
        chk_haproxy
    }
    virtual_ipaddress {
```

```
    9.76.62.144/32 dev enc1000 label enc1000:0
  }
}
```

Here is an example on HAProxy2: (Example 5-20).

Example 5-20 BACKUP - PRIORITY 90

```
global_defs {
    notification_email {
        sysadmin@firewall.loc
    }
    vrrp_version 3
    notification_email_from sysadmin@firewall.loc
    smtp_server 192.168.200.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
    vrrp_skip_check_adv_addr
    #vrrp_strict
    vrrp_garp_interval 0
    vrrp_gna_interval 0
    process_names
    enable_script_security
    script_user root
}
vrrp_script chk_haproxy {
    script "/usr/bin/systemctl is-active --quiet HAProxy"
    fall 2 # 2 fails required for failure
    rise 2 # 2 OKs required to consider the
          # process up after failure
    interval 5 # check every 5 seconds
    weight 51 # add 50 points rc=0
}
vrrp_instance VI_1 {
    state BACKUP
    interface enc1000
    virtual_router_id 2
    priority 90
    advert_int 3
    # check network is up
    track_interface {
        enc1000 weight 50
    }
    # check that HAProxy is up
    track_script {
        chk_haproxy
    }
    virtual_ipaddress {
        9.76.62.144/32 dev enc1000 label enc1000:0
    }
}
```

Here is an example on HAProxy3: (Example 5-21).

Example 5-21 BACKUP - PRIORITY 80

```
global_defs {
    notification_email {
        sysadmin@firewall.loc
    }
    vrrp_version 3
    notification_email_from sysadmin@firewall.loc
    smtp_server 192.168.200.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
    vrrp_skip_check_adv_addr
    #vrrp_strict
    vrrp_garp_interval 0
    vrrp_gna_interval 0
    process_names
    enable_script_security
    script_user root
}
vrrp_script chk_haproxy {
    script "/usr/bin/systemctl is-active --quiet HAProxy"
    fall 2 # 2 fails required for failure
    rise 2 # 2 OKs required to consider the
          # process up after failure
    interval 5 # check every 5 seconds
    weight 51 # add 50 points rc=0
}
vrrp_instance VI_1 {
    state BACKUP
    interface enc1000
    virtual_router_id 3
    priority 80
    advert_int 3
    # check network is up
    track_interface {
        enc1000 weight 50
    }
    # check that HAProxy is up
    track_script {
        chk_haproxy
    }
    virtual_ipaddress {
        9.76.62.144/32 dev enc1000 label enc1000:0
    }
}
}
```

4. Configure the firewall to allow the VRRP protocol:

```
firewall-cmd --permanent --new-service=VRRP
firewall-cmd --permanent --service=VRRP --set-description="Virtual Router
Redundancy Protocol"
firewall-cmd --permanent --service=VRRP --set-short=VRRP
firewall-cmd --permanent --service=VRRP --add-protocol=vrrp
firewall-cmd --permanent --service=VRRP --set-destination=ipv4:224.0.0.18
firewall-cmd --add-service=VRRP --permanent
```

5. Configure SELinux:

```
grep keepalived_t /var/log/audit/audit.log|audit2allow -M keepalived_custom
semodule -i keepalived_custom.pp
```

6. Enable and start Keepalived service:

```
systemctl enable keepalived --now
```

HAPROXY configuration

Example 5-22 shows some minor changes needed for HAProxy configuration. These changes are optional but allow use of a virtual IP address or the host IP address.

Example 5-22 HAProxy optional configuration changes

```
frontend ocp4-kubernetes-api-server
...
    bind *:6443
...

frontend ocp4-machine-config-server
...
    bind *:22623
...

frontend ocp4-router-http
...
    bind *:80
...

frontend ocp4-router-https
...
    bind *:443
...

```

DNS configuration

Example 5-23 shows minor changes to the DNS configuration to allow use of virtual IP addresses.

Example 5-23 Configurations changes for virtual IP addresses

```
[root@dns1 ~]# cat /var/named/zvm.rdbk.com.zone
api                IN A 9.76.62.144

api-int           IN A 9.76.62.144

apps              IN A 9.76.62.144

*.apps            IN A 9.76.62.144
...

```

At this point, you should be able to connect to the Red Hat OpenShift Cluster Console using the same domain name, but with virtual IP addresses for the HAProxy services.

5.3.4 USER DIRECT and PARM files for OCP nodes

This step is only applicable to the IBM z/VM environment. Here we define the VM definitions and prepare the PARM files for OCP nodes. In this scenario, we use RoCE card instead of an OSA card.

USER DIRECT

The IBM z/VM user directory specifies the configuration and operating characteristics of VMs.

Example 5-24 shows our USER DIRECT file for the bootstrap node.

Example 5-24 Bootstrap node

```
USER OCP1B0 LBYONLY 24G 256G G
  INCLUDE INSTALL
  COMMAND TERM MORE 0 0
  COMMAND TERM HOLD OFF
  COMMAND ATT PCIF 00003425 OCP1B0 00003425
  CPU 00 BASE
  CPU 01
  CPU 02
  CPU 03
  IPL CMS PARM AUTOOCR
  LOGONBY MAINT IBMVM1
  MACHINE ESA 64
  MDISK 0100 3390 1 60101 RRLX0A MR

```

Example 5-25 shows our USER DIRECT file for the controller node.

Example 5-25 Controller node

```
USER OCP1M1 LBYONLY 24G 256G G
  INCLUDE LNXDFLT
  COMMAND TERM MORE 0 0
  COMMAND TERM HOLD OFF
  COMMAND ATT PCIF 00003125 OCP1M1
  CPU 00 BASE
  CPU 01
  CPU 02
  CPU 03
  IPL 100
  LOGONBY MAINT IBMVM1
  MACHINE ESA 64
  MDISK 0100 3390 1 60101 RRLX0B MR
```

Example 5-26 shows our USER DIRECT file for the compute node.

Example 5-26 Compute node

```
USER OCP1W1 LBYONLY 24G 256G G
  INCLUDE LNXDFLT
  COMMAND TERM MORE 0 0
  COMMAND TERM HOLD OFF
  COMMAND ATT PCIF 00003525 OCP1W1
  CPU 00 BASE
  CPU 01
  CPU 02
  CPU 03
  IPL 100
  LOGONBY MAINT IBMVM1
  MACHINE ESA 64
  MDISK 0100 3390 1 60101 RRLX0I MR
```

USER PARM

Example 5-27 shows our sample configuration data to boot the bootstrap.

Example 5-27 Bootstrap node

```
OCP1B0 PRM      Z1 V 80 Trunc=80 Size=10 Line=0 Col=1 Alt=0
rd.neednet=1 dfltcc=off
console=ttysclp0
searchdomain=zvm.rdbk.com
coreos.inst.install_dev=/dev/dasda nameserver=9.76.62.141
coreos.live.rootfs_url=http://9.76.62.130:8080/rootfs/4.12.10.rootfs.img
coreos.inst.ignition_url=http://9.76.62.130:8080/ignition/bootstrap.ign
ip=9.76.62.140::9.76.62.1:255.255.255.0::bond0:none
bond=bond0:ens1877:mode=active-backup,fail_over_mac=1
rd.dasd=0.0.0100
```

Example 5-28 shows our sample configuration data to boot the control plane node.

Example 5-28 Controller node

```
OCP1M1 PRM      Z1 V 80 Trunc=80 Size=10 Line=0 Col=1 Alt=0
rd.neednet=1 dfltcc=off
console=ttysclp0
searchdomain=zvm.rdbk.com nameserver=9.76.62.141
coreos.inst.install_dev=/dev/dasda
coreos.live.rootfs_url=http://9.76.62.130:8080/rootfs/4.12.10.rootfs.img
coreos.inst.ignition_url=http://9.76.62.130:8080/ignition/master.ign
ip=9.76.62.131::9.76.62.1:255.255.255.0::bond0:none
bond=bond0:ens1621:mode=active-backup,fail_over_mac=1
rd.dasd=0.0.0100
```

Example 5-29 shows our sample configuration data to boot the compute node.

Example 5-29 Compute node

```
OCP1W1 PRM      Z1 V 80 Trunc=80 Size=10 Line=0 Col=1 Alt=0
rd.neednet=1 dfltcc=off
console=ttysclp0
searchdomain=zvm.rdbk.com nameserver=9.76.62.141
coreos.inst.install_dev=/dev/dasda
coreos.live.rootfs_url=http://9.76.62.130:8080/rootfs/4.12.10.rootfs.img
coreos.inst.ignition_url=http://9.76.62.130:8080/ignition/worker.ign
ip=9.76.62.134::9.76.62.1:255.255.255.0::bond0:none
bond=bond0:ens1877:mode=active-backup,fail_over_mac=1
rd.dasd=0.0.0100
```

5.3.5 Building the OCP cluster

In this step, you build the OCP cluster for a three-LPAR cluster case. Log in to the three IBM z/VM systems to start up the VMs. In an IBM z/VM SSI cluster environment, you could issue the command **RDBKZVMx CMD Xauto1og nodename** to start a specific system from a single IBM z/VM node.

1. To start up the bootstrap node, issue the following commands:

```
Logon to RDBKZVMA
XAUTOLOG OCP2B0
```

Check if the bootstrap node is ready with the following command:

```
openshift-install --dir <installation_directory> wait-for bootstrap-complete
--log-level=info
```

Check if port 22623 is ready or not with the following command.

```
curl https://<haproxy_ip>:22623/config/master
```

2. Start up the Controller and compute nodes. In our lab environment, we use the following commands:

```
RDBKZVMA: XAUTOLOG OCP2M1
RDBKZVMB: XAUTOLOG OCP2M2
RDBKZVMC: XAUTOLOG OCP2M3
RDBKZVMA: XAUTOLOG OCP2W1
RDBKZVMB: XAUTOLOG OCP2W2
RDBKZVMC: XAUTOLOG OCP2W3
RDBKZVMA: XAUTOLOG OCP2F1
RDBKZVMB: XAUTOLOG OCP2F2
RDBKZVMC: XAUTOLOG OCP2F3
```


5.4 KVM single hypervisor cluster implementation

In our single hypervisor example environment we perform an Ansible-based installation using a user-provided installer (UPI) approach. For more information, see [Ansible-Automated OpenShift Provisioning on KVM on IBM zSystems / LinuxONE](#).

The playbook can install a KVM host into an LPAR, but in our environment this was already installed. Additionally, the playbook requires a Dynamic Partition Manager (DPM) enabled host to achieve the automated installation and our lab is not using a DPM enabled system.

The document for the UPI playbook indicates we need a file server and an Ansible Controller node. We will not need the jump host mentioned for a NAT network. As a best practice, we are not going to add file server or controller functions to the KVM host, as they both reside in a single RHEL guest system. We will call this guest domain `controller`. In our example deployment, we also place the DNS server in its own virtual server. The load balancer, along with the bastion host, and the OCP cluster itself will be deployed by the Ansible Playbook automation.

5.4.1 Architecture

Figure 5-4 shows our Single LPAR Ansible-automated architecture. There are eight RoCE ports on four adapter cards that are configured to four Linux network bonds in active Stand-By mode. The bastion, DNS, load balancer, and controller are RHEL virtual servers. The remainder are Red Hat CoreOS nodes that compose our Red Hat OpenShift 4.12 cluster. The Red Hat CoreOS nodes could be moved to other LPARs, where you could easily convert the Single LPAR deployment in to a three-LPAR deployment. The Ansible code does support a multi-LPAR deployment. We only use Ansible in this single LPAR deployment so you have the opportunity see deployments through both methods.

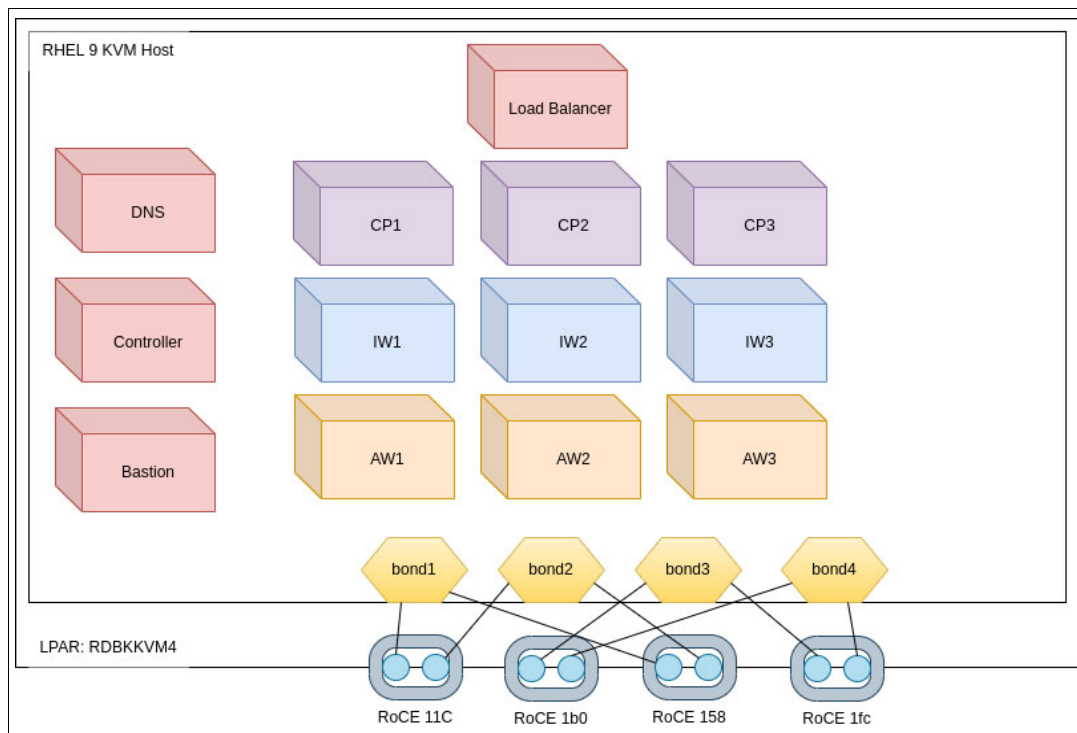


Figure 5-4 Single LPAR Ansible Automated Architecture

5.4.2 Planning resources

We map all of our RHEL and Red Hat CoreOS servers with virtual CPU, memory, and disk allocations as required, as well as network adapter usage. We create four different bonds across the different RoCE adapters. We can utilize all of them across the different Red Hat OpenShift nodes as needed. While we start with use a single bond for the Red Hat CoreOS guests, we could easily add others as needed. Since we use active/standby mode, we could potentially have up to 100 GB (4 x 25) of bandwidth if needed.

We also map out the DNS entries. DNS entries and load balancer configuration are two of the most common problem areas during an Red Hat OpenShift deployment.

If you add up the QCOW disk sizes they total almost 1.3 TB. However, because the QCOW2 files are sparse, even weeks after the deployment we see less than 200 GB of disk usage. The Ansible script does deploy the nodes with `discard=unmap`, so any files deleted within the virtual server frees up the space in the file system. Additionally, because the QCOW2 are hosted on a logical volume manager (LVM), the requirement to use extended address volumes (EAVs) is eliminated. However, you still may use them if you wish.

Since we use ECKD EAVs as well as Parallel Access Volumes (PAVs) for parallelism of I/O. The single LPAR cluster resource plan is shown in Table 5-5.

Table 5-5 Resource planning by function

Function	KVM Domain	Hypervisor	Virtual CPU	Initial Domain Memory in GB	Max QCOW2 Size GB
Ansible Controller	controller	rdbkkvm4	4	16	20
Bastion/Load Balancer	bastion	rdbkkvm4	4	8	20
DNS	dnshcp	rdbkkvm4	4	16	20
Bootstrap	bootstrap	rdbkkvm4	4	16	120
Control Plane	cp1	rdbkkvm4	4	16	120
Control Plane	cp2	rdbkkvm4	4	16	120
Control Plane	cp3	rdbkkvm4	4	16	120
Application Worker	aw1	rdbkkvm4	4	16	120
Application Worker	aw2	rdbkkvm4	4	16	120
Application Worker	aw3	rdbkkvm4	4	16	120
Infra Worker	iw1	rdbkkvm4	4	16	120
Infra Worker	iw2	rdbkkvm4	4	16	120
Infra Worker	iw3	rdbkkvm4	4	16	120

Table 5-6 provides our resource plan for the RoCE adapters and bond mappings.

Table 5-6 RoCE adapter and port to bond mappings

RoCE PCHID	RoCE Port	LPAR 4Net-Dev	Bond
11c	0	ens12616	bond1
11c	1	ens12872	bond2
1b0	0	ens13128	bond3
1b0	1	ens13384	bond4
158	0	ens13640	bond1
158	1	ens13896	bond2
1fc	0	ens14152	bond3
1fc	1	ens14408	bond4

Table 5-7 on page 78 provides our guest domain, bond, and IP address mappings.

Table 5-7 Guest domain, bond, IP address mappings

KVM Domain	Host Name (Short)	Bond	IP Address
bastion	bastion	bond4	9.76.61.82
controller	controller	bond4	9.76.61.95
dnshcp	dnshcp	bond4	9.76.61.94
bootstrap	bootstrap	bond4	9.76.61.84
cp1	cp1	bond4	9.76.61.85
cp2	cp2	bond4	9.76.61.86
cp3	cp3	bond4	9.76.61.87
aw1	aw1	bond4	9.76.61.91
aw2	aw2	bond4	9.76.61.92
aw3	aw3	bond4	9.76.61.93
iw1	iw1	bond4	9.76.61.88
iw2	iw2	bond4	9.76.61.89
iw3	iw3	bond4	9.76.61.90

Table 5-7 shows our forward and reverse DNS with their domain name and IP addresses.

Table 5-7 Forward and Reverse DNS

Host name	domain	IP Address
controller	ocp1.ibm.com	9.76.61.95
bastion	ocp1.ibm.com	9.76.61.82
bootstrap	ocp1.ibm.com	9.76.61.84
cp1	ocp1.ibm.com	9.76.61.85
cp2	ocp1.ibm.com	9.76.61.86
cp3	ocp1.ibm.com	9.76.61.87
aw1	ocp1.ibm.com	9.76.61.91
aw2	ocp1.ibm.com	9.76.61.92
aw3	ocp1.ibm.com	9.76.61.93
iw1	ocp1.ibm.com	9.76.61.88
iw2	ocp1.ibm.com	9.76.61.89
iw3	ocp1.ibm.com	9.76.61.90
api	ocp1.ibm.com	9.76.61.82
api-int	ocp1.ibm.com	9.76.61.82
apps	ocp1.ibm.com	9.76.61.82
*.apps	ocp1.ibm.com	9.76.61.82
dnshcp	ocp1.ibm.com	9.76.61.94

5.4.3 DNS configuration

DNS is a requirement to deploy and use Red Hat OpenShift. In a production environment, you usually use the enterprise DNS servers. We configured named for the DNS server. Our example DNS configuration consists of the following three files:

- ▶ File: `/var/named/named.61.76.9.in-addr.arpa.zone` shown in Example 5-30
- ▶ File: `/var/named/named.ocp1.ibm.com` shown in Example 5-31 on page 80
- ▶ File: `/etc/named.conf` shown in Example 5-32 on page 81

Example 5-30 File: `/var/named/named.61.76.9.in-addr.arpa.zone`

```
$TTL 900
@ IN SOA dnsdhcp.ocp1.ibm.com. admin.ocp1.ibm.com. (
    2022020202 ; serial
    3600      ; refresh
    1800      ; retry
    604800    ; expire
    86400     ; negative cache ttl
)

; NameServer

@ IN NS dnsdhcp.ocp1.ibm.com.

;reverse for name server

80    IN PTR 1b0.ocp1.ibm.com.
80    IN PTR 1b1.ocp1.ibm.com.

82    IN PTR bastion.ocp1.ibm.com.
84    IN PTR bootstrap.ocp1.ibm.com.
85    IN PTR cp1.ocp1.ibm.com.
86    IN PTR cp2.ocp1.ibm.com.
87    IN PTR cp3.ocp1.ibm.com.
91    IN PTR aw1.ocp1.ibm.com.
92    IN PTR aw2.ocp1.ibm.com.
93    IN PTR aw3.ocp1.ibm.com.
88    IN PTR iw1.ocp1.ibm.com.
89    IN PTR iw2.ocp1.ibm.com.
90    IN PTR iw3.ocp1.ibm.com.
83    IN PTR nfs.ocp1.ibm.com.
94    IN PTR dnsdhcp.ocp1.ibm.com.
```

Example 5-31 File: /var/named/named.ocp1.ibm.com

```
# cat named.ocp1.ibm.com
$TTL 3H
;base domain name
$ORIGIN ocp1.ibm.com.

@      IN SOA @ ocp1.ibm.com. (
                                2022020202    ; serial
                                1D             ; refresh
                                1H             ; retry
                                1W             ; expire
                                3H             ; minimum
                                )
;Names servers for this domain
      IN NS dnsdhcp.ocp1.ibm.com.
;Mail servers
; none
;A Records
lb0      IN A 9.76.61.80
lb1      IN A 9.76.61.81
bastion  IN A 9.76.61.82
bootstrap IN A 9.76.61.84
cp1      IN A 9.76.61.85
cp2      IN A 9.76.61.86
cp3      IN A 9.76.61.87
aw1      IN A 9.76.61.91
aw2      IN A 9.76.61.92
aw3      IN A 9.76.61.93
iw1      IN A 9.76.61.88
iw2      IN A 9.76.61.89
iw3      IN A 9.76.61.90
nfs      IN A 9.76.61.83
api      IN A 9.76.61.82
api-int  IN A 9.76.61.82
apps     IN A 9.76.61.82
*.apps   IN A 9.76.61.82
dnsdhcp  IN A 9.76.61.94
```

```
acl internal_nets { 9.76.61/24; };

options {
    listen-on port 53 { any; };
    listen-on-v6 port 53 { ::1; };
    listen-on-v6 port 53 { none; };
    directory      "/var/named";
    dump-file       "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    secroots-file  "/var/named/data/named.secroots";
    recursing-file  "/var/named/data/named.recursing";
    allow-query    { localhost; internal_nets; };
    forwarders     { 9.0.0.2; };
    allow-recursion { localhost; internal_nets; };
    allow-query-cache { localhost; internal_nets; };

    recursion yes;
    dnssec-validation no;

    managed-keys-directory "/var/named/dynamic";
    geoip-directory "/usr/share/GeoIP";
    pid-file "/run/named/named.pid";
    session-keyfile "/run/named/session.key";

    querylog yes;
    allow-transfer { none; };

    include "/etc/crypto-policies/back-ends/bind.config";
};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";

zone "ocp1.ibm.com" {
    type master;
    file "/var/named/named.ocp1.ibm.com";
    allow-query { any; };
    allow-transfer { none; };
    allow-update { none; };
};
```

```

zone "61.76.9.in-addr.arpa" {
    type master;
    file "/var/named/named.61.76.9.in-addr.arpa.zone";
    allow-query {any;};
    allow-transfer {none;};
    allow-update {none;};
};

```

5.4.4 Ansible controller configuration

We used `virt-install` (a CLI tool for creating a new KVM) and `kickstart` (a method for automated installation of RHEL systems) to automate the installation of the controller. The resource requirements of the controller node are modest. We just need a bit of disk space to contain the playbooks and also serve files, since it is also acting as our file server for the Ansible playbooks. Our controller's kickstart file is shown in Example 5-33.

Example 5-33 controller.ks

```

authselect --enableshadow --passalgo=sha512
cdrom
text
firstboot --enable
ignoredisk --only-use=vda
keyboard --vckeymap=us --xlayouts='us'
lang en_US.UTF-8
# Network information
firewall --enabled --ssh
network --device=encl --bootproto=dhcp --noipv6
network --device=enc9 --bootproto=static --noipv6 --noipv4
network --hostname=controller.ocpl.ibm.com
rootpw --plaintext its0
# System services
services --enabled="chronyd"
# System timezone
timezone America/New_York
user --groups=wheel --name=admin1 --password=its0 --plaintext --gecos="admin1"
# System bootloader configuration
bootloader --append="crashkernel=auto dfltcc=always transparent_hugepages=never
" --location=mbr --boot-drive=vda
#Partitioning
clearpart --all --initlabel --disklabel=gpt --drives=vda
part / --fstype="xfs" --ondisk=vda --grow
#Packages
%packages --ignoremissing --instLangs=en_US
@base
@core
nginx
ansible-core
chrony
sysstat
tmux
iotop

```



```
cockpit
kexec-tools
%end
%addon com_redhat_kdump --enable --reserve-mb='auto'
%end
%post --log=/root/ks-post.log
ip addr
df -h
%end
eula --agreed
reboot
```

The command to install the Ansible controller virtual server is shown in Example 5-34.

Example 5-34 Install the Ansible controller virtual server

```
virt-install --input keyboard,bus=virtio --input mouse,bus=virtio --graphics
vnc,listen=0.0.0.0 --video virtio --name controller --memory 16384 --vcpus=4
--iothreads=2 --disk
size=20,format=qcow2,cache=none,sparse=yes,discard=unmap,driver.iothread=1
--network network=default --network
type=direct,source=bond4,source_mode=bridge,address.cssid=0xfe,address.ssid=0,a
ddress.devno=0x0009,address.type=ccw --location
/var/lib/libvirt/images/RHEL-9.1.0-20221027.3-s390x-dvd1.iso
--initrd-inject=/root/controller.ks --extra-args "inst.ks=file:/controller.ks"
--console pty,target_type=serial --autoconsole text
```

5.4.5 Load balancer configuration

The load balancer function on this Ansible based deployment was placed onto the bastion host and used the bastion host single IP address. Deploying a load balancer with a unique IP address can provide the benefit of being able to more easily move that IP address/load balancing function to another server. While the Ansible script did fully configure the HAProxy based load balancer function, it did not open the firewall for port 1936, which provides HTTP based statistics on the load balancer operation. The other firewall ports are automatically configured by the Ansible playbook.

5.4.6 File server for Ansible Playbook

We use Nginx to provide the file-serving function of the Ansible playbook. Our Nginx file serving configuration is shown in Example 5-35. This not only serves the RHEL 8.7 ISO image required by the playbook, but also the configuration files and Red Hat OpenShift artifacts generated or obtained by the playbook automation. The playbook assumes the content is under /home/<<userid>>. Our content is placed in /home/admin1. In that directory, we have a directory called ocpauto (containing the playbooks), ocp-config, and a RHEL 8.7 upon which the RHEL ISO file is loop mounted.

We add a location directive for /home/admin1 to our nginx.conf configuration file (see Example 5-35) and restart it.

Example 5-35 Nginx file serving configuration

```
server {
    listen      80;
    listen      [::]:80;
    server_name _;
    root        /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location /home/admin1 {
        alias /home/admin1 ;
        autoindex on;
    }

    location / {

```

We also open the relevant firewall ports for http and https by using the commands shown in Example 5-36.

Example 5-36 Commands to open ports

```
[root@controller admin1]# firewall-cmd --add-port=80/tcp --perm
success
[root@controller admin1]# firewall-cmd --add-port=443/tcp --perm
success
[root@controller admin1]# firewall-cmd --add-port=443/tcp
success
[root@controller admin1]# firewall-cmd --add-port=80/tcp
success
```

5.4.7 Building the OCP cluster through Ansible

There are eight playbooks for deploying Red Hat OpenShift on Linux on IBM Z and LinuxONE:

1. Set up Playbook
2. Create LPAR Playbook
3. Create KVM Host Playbook
4. Setup KVM Host Playbook
5. Create Bastion Playbook
6. Setup Bastion Playbook
7. Create Nodes Playbook
8. OCP Verification Playbook

For more information, see [Ansible-Automated OpenShift Provisioning on KVM on IBM zSystems / LinuxONE](#).

In our environment we run the Setup Playbook, and skip both the Create LPAR and Create KVM Host playbooks as the LPAR and KVM host already exist. We run the Setup KVM Host playbook through the OCP Verification Playbook. We run the process from our non-root user id admin1.

1. We begin by installing pip on our server using the following command:

```
[root@controller admin1]# yum install python3-pip
```

2. Next, we create and change to the ocpauto subdirectory from the admin1 home directory, using the following commands:

```
[admin1@controller ~]$ mkdir ocpauto  
[admin1@controller ~]$ cd ocpauto/
```

3. Next, we clone the playbook from GitHub:

```
[admin1@controller ocpauto]$ git clone  
https://github.com/IBM/Ansible-OpenShift-Provisioning.git
```

4. In addition to installing the playbook, there are a few prerequisite galaxy collections. We use the following command to do this:

```
[admin1@controller Ansible-OpenShift-Provisioning]$ ansible-galaxy collection  
install community.general community.crypto ansible.posix community.libvirt
```

- In the `host_vars` Ansible subdirectory, `/home/admin1/ocpauto/Ansible-OpenShift-Provisioning/inventories/default/host_vars`, we create a file named `rdbkkvm4.yaml` from the `KVMhostname1-here.yaml.template`. The contents of `rdbkkvm4.yaml` are shown in Example 5-37.

Example 5-37 Our Ansible inventory YAML file

```
# Section 1 - KVM Host
networking:
  hostname: rdbkkvm4
  ip: 9.76.61.184
  subnetmask: 255.255.255.0
  gateway: 9.76.61.1
  nameserver1: 9.0.0.2
# nameserver2:
  device1: bond3
# device2:

storage:
  pool_path: /var/lib/libvirt/images

#####
# Variables below this point only need to be filled out if #
# env.z.lpar1.create is True. Meaning, if the LPARs you will #
# be using as KVM host(s) already exist and have RHEL #
# installed, the variables below are not required. #
#####

# Section 2 - CPC & HMC
cpc_name: #X
hmc:
  host: #X
  auth:
    user: #X
    pass: #X

# Section 3 - LPAR
lpar:
  name: #X
  description: #X
  access:
    user: #X
    pass: #X
    root_pass: #X

# Section 4 - IFL & Memory
ifl:
  count: #X
  initial_memory: #X
  max_memory: #X
  initial_weight: #X
  min_weight: #X
  max_weight: #X

# Section 5 - Networking
networking:
```

```

    subnet_cidr: #X
    nic:
      card1:
        name: #X
        adapter: #X
        port: #X
        dev_num: #X
#     card2:
#       name:
#       adapter:
#       port:
#       dev_num:

# Section 6 - Storage
storage_group_1:
  name: #X
  type: fcp
  storage_wwpn:
    - #X
    - #X
    - #X
    - #X
  dev_num: #X
  lun_name: #X
# storage_group_2:
#   auto_config: True
#   name:
#   type: fcp
#   storage_wwpn:
#     -
#     -
#     -
#     -
#   dev_num:
#   lun_name:

```

-
6. The other file we need to tailor is `all.yaml` in the `group_vars` subdirectory of our Ansible code. We remove our “Pull Secret” content and replace it with “<<your pull secret here>>” for readability. We did a similar change for the Red Hat credentials. You will need to supply your own Red Hat credentials. Our file is shown in Example 5-38.

Example 5-38 Our all.yaml file

```

---
#
# Section 1 - Ansible Controller
env:
  controller:
    sudo_pass: its0

# Section 2 - LPAR(s)
z:
  high_availability: False
  ip_forward: True
  lpar1:

```

```

        create: False
        hostname: rdbkvm4
        ip: 9.76.61.184
        user: lnxadmin
        pass: lnx4rdbk
    lpar2:
        create: False
#     hostname:
#     ip:
#     user:
#     pass:
    lpar3:
        create: False
#     hostname:
#     ip:
#     user:
#     pass:

# Section 3 - File Server
file_server:
    ip: 9.76.61.95
    user: admin1
    pass: its0
    protocol: http
    iso_mount_dir: /home/admin1/RHEL/8.7
    cfigs_dir: ocp-config

# Section 4 - Red Hat
redhat:
    username: <<your Red Hat credentials>>
    password: <<<your Red Hat credentials>>
    # Make sure to enclose pull_secret in 'single quotes'
    pull_secret: '<<<your pull secret here>>''

# Section 5 - Bastion
bastion:
    create: True
    vm_name: bastion
    resources:
        disk_size: 30
        ram: 8192
        swap: 4096
        vcpu: 4
    networking:
        ip: 9.76.61.82
        hostname: bastion
        base_domain: ocp1.ibm.com
        subnetmask: 255.255.255.0
        gateway: 9.76.61.1
        nameserver1: 9.76.61.94
#     nameserver2:
        forwarder: 9.76.61.94
        interface: encl
    access:
        user: admin1

```

```

    pass: its0
    root_pass: its0
  options:
    dns: False
    loadbalancer:
      on_bastion: True
      public_ip: 9.76.61.82
      private_ip: 9.76.71.82

# Section 6 - Cluster Networking
cluster:
  networking:
    metadata_name: ocp1
    base_domain: ibm.com
    subnetmask: 255.255.255.0
    gateway: 9.76.61.1
    nameserver1: 9.76.61.94
#   nameserver2:
    forwarder: 9.0.0.2

# Section 7 - Bootstrap Node
nodes:
  bootstrap:
    disk_size: 120
    ram: 16384
    vcpu: 4
    vm_name: bootstrap
    ip: 9.76.61.84
    hostname: bootstrap

# Section 8 - Control Nodes
control:
  disk_size: 120
  ram: 16384
  vcpu: 4
  vm_name:
    - cp1
    - cp2
    - cp3
  ip:
    - 9.76.61.85
    - 9.76.61.86
    - 9.76.61.87
  hostname:
    - cp1
    - cp2
    - cp3

# Section 9 - Compute Nodes
compute:
  disk_size: 120
  ram: 16384
  vcpu: 4
  vm_name:
    - aw1

```

```

        - aw2
        - aw3
    ip:
        - 9.76.61.91
        - 9.76.61.92
        - 9.76.61.93
    hostname:
        - aw1
        - aw2
        - aw3

# Section 10 - Infra Nodes
infra:
    disk_size: 120
    ram: 16384
    vcpu: 4
    vm_name:
        - iw1
        - iw2
        - iw3
    ip:
        - 9.76.61.88
        - 9.76.61.89
        - 9.76.61.90
    hostname:
        - iw1
        - iw2
        - iw3

#####
#####
# All variables below this point do not need to be changed for a default
installation #
#####
#####

# Section 11 - (Optional) Packages
pkgs:
    galaxy: [ ibm.ibm_zhmc, community.general, community.crypto, ansible.posix,
community.libvirt ]
    controller: [ openssh, expect ]
    kvm: [ libguestfs, libvirt-client, libvirt-daemon-config-network,
libvirt-daemon-kvm, cockpit-machines, virt-top, qemu-kvm, python3-lxml,
cockpit, lvm2 ]
    bastion: [ HAProxy, httpd, bind, bind-utils, expect, firewallld, mod_ssl,
python3-policycoreutils, rsync ]
    hypershift: [ make, jq, git, virt-install ]

# Section 12 - OpenShift Settings
openshift:
    version: 4.12.0
    install_config:
        api_version: v1
    compute:
        architecture: s390x

```



```

        hyperthreading: Enabled
control:
  architecture: s390x
  hyperthreading: Enabled
cluster_network:
  cidr: 10.128.0.0/14
  host_prefix: 23
  type: OVNKubernetes
service_network: 172.30.0.0/16
fips: 'false'

# Section 13 - (Optional) Proxy
# proxy:
#   http:
#   https:
#   no:

# Section 14 - (Optional) Misc
language: en_US.UTF-8
timezone: America/New_York
keyboard: us
root_access: false
ansible_key_name: ansible-ocpz
ocp_ssh_key_comment: OpenShift key
bridge_name: bond4
network_mode:

#jumphost if network mode is NAT
jumphost:
  name:
  ip:
  user:
  pass:
  path_to_keypair:

# Section 15 - RHCOS (CoreOS)

# rhcos_download_url with '/' at the end !
rhcos_download_url:
"https://mirror.openshift.com/pub/openshift-v4/s390x/dependencies/rhcos/4.12/4.12.3/"

# For rhcos_os_variant use the OS string as defined in 'osinfo-query os -f short-id'
rhcos_os_variant: rhel8.6

# RHCOS live image filenames
rhcos_live_kernel: "rhcos-4.12.3-s390x-live-kernel-s390x"
rhcos_live_initrd: "rhcos-4.12.3-s390x-live-initramfs.s390x.img"
rhcos_live_rootfs: "rhcos-4.12.3-s390x-live-rootfs.s390x.img"

# Section 16 - Hypershift

hypershift:

```

```

kvm_host:
kvm_host_user:
bastion_hypershift:
bastion_hypershift_user:
mgmt_cluster_nameserver:

go_version: "1.19.5" # Change this if you want to install any other version
of go
oc_url:

#Hosted Control Plane Parameters

hcp:
  clusters_namespace:
  hosted_cluster_name:
  basedomain:
  pull_secret_file: /root/ansible_workdir/auth_file
  ocp_release:
  machine_cidr:
  arch:
  # Make sure to enclose pull_secret in 'single quotes'
  pull_secret:

# AgentServiceConfig Parameters

asc:
  url_for_ocp_release_file:
  db_volume_size:
  fs_volume_size:
  ocp_version:
  iso_url:
  root_fs_url:
  mce_namespace: "multicluster-engine" # This is the Recommended Namespace
for Multicluster Engine operator

path_to_key_pair: /home/admin1/.ssh/ansible-ocpz.pub

```

- Next, we ran each of the playbooks by using the commands shown in Example 5-39. The output of each command has been omitted due to its length for readability.

Example 5-39 Commands to run playbooks

```
[admin1@controller Ansible-OpenShift-Provisioning]$ ansible-playbook
playbooks/0_setup.yaml
[admin1@controller Ansible-OpenShift-Provisioning]$ ansible-playbook
playbooks/3_setup_kvm_host.yaml
[admin1@controller Ansible-OpenShift-Provisioning]$ ansible-playbook
playbooks/4_create_bastion.yaml
[admin1@controller Ansible-OpenShift-Provisioning]$ ansible-playbook
playbooks/5_setup_bastion.yaml
[admin1@controller Ansible-OpenShift-Provisioning]$ ansible-playbook
playbooks/6_create_nodes.yaml
[admin1@controller Ansible-OpenShift-Provisioning]$ ansible-playbook
playbooks/7_ocp_verification.yaml
```

After deploying the cluster we did notice that the playbooks did not remove the bootstrap node from our load balancer configuration. This is a recommended step in the Red Hat OpenShift product documentation.

5.4.8 Validating the deployment

There are a number of ways to validate the initial deployment. One way is to simply log in through the command line interface (CLI) and list the cluster operators from the CLI with the command `oc get co` (Example 5-40 on page 94). A second way is through the WebUI. Figure 5-5 shows the Web UI. You can also use the list playbook as a validation step (Example 5-41 on page 95).

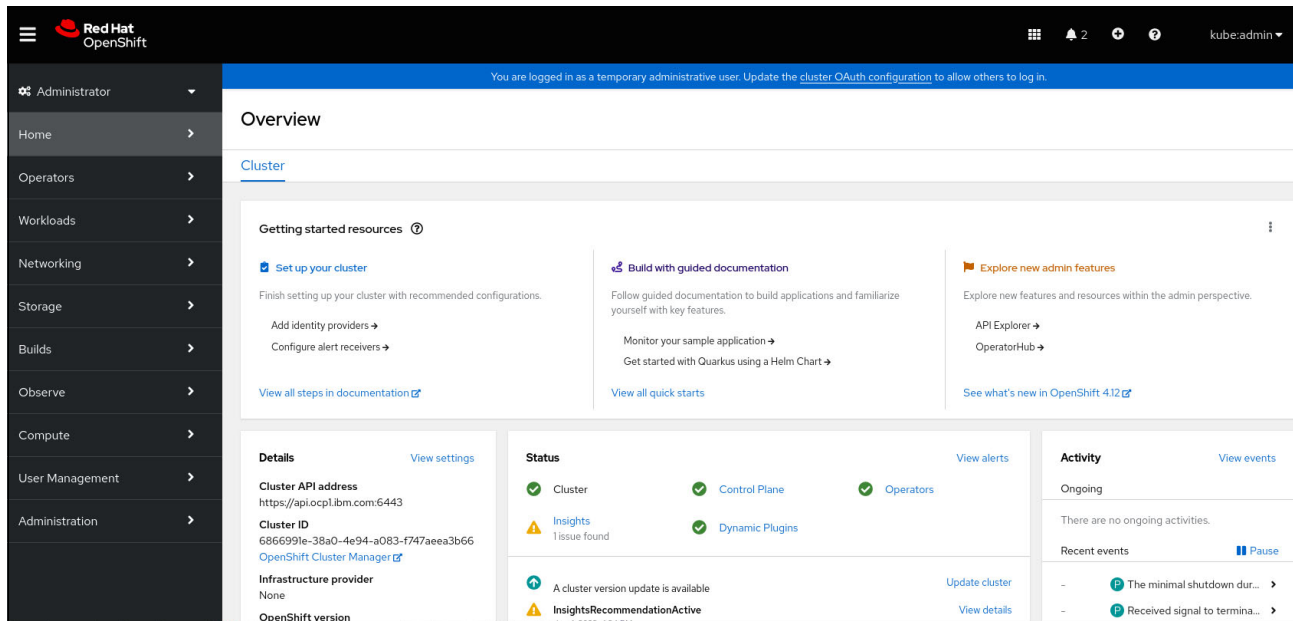


Figure 5-5 Red Hat OpenShift Web UI Overview

Example 5-40 List cluster operators from the CLI

```
[root@bastion ocpinst]# oc get co
NAME                                     VERSION  AVAILABLE  PROGRESSING
DEGRADED  SINCE  MESSAGE
authentication                             4.12.0   True       False
False    158m
baremetal                                   4.12.0   True       False
False    4h49m
cloud-controller-manager                   4.12.0   True       False
False    4h53m
cloud-credential                           4.12.0   True       False
False    4h52m
cluster-autoscaler                         4.12.0   True       False
False    4h49m
config-operator                            4.12.0   True       False
False    4h50m
console                                    4.12.0   True       False
False    164m
control-plane-machine-set                  4.12.0   True       False
False    4h49m
csi-snapshot-controller                    4.12.0   True       False
False    4h49m
dns                                          4.12.0   True       False
False    4h49m
etcd                                        4.12.0   True       False
False    4h47m
image-registry                             4.12.0   True       False
False    4h39m
ingress                                    4.12.0   True       False
False    167m
insights                                   4.12.0   True       False
False    4h43m
kube-apiserver                             4.12.0   True       False
False    4h44m
kube-controller-manager                    4.12.0   True       False
False    4h47m
kube-scheduler                             4.12.0   True       False
False    4h46m
kube-storage-version-migrator              4.12.0   True       False
False    4h50m
machine-api                                4.12.0   True       False
False    4h49m
machine-approver                           4.12.0   True       False
False    4h50m
machine-config                             4.12.0   True       False
False    4h48m
marketplace                                4.12.0   True       False
False    4h49m
monitoring                                  4.12.0   True       False
False    165m
network                                     4.12.0   True       False
False    4h49m
node-tuning                                4.12.0   True       False
False    166m
```

openshift-apiserver	4.12.0	True	False
False 4h42m			
openshift-controller-manager	4.12.0	True	False
False 4h45m			
openshift-samples	4.12.0	True	False
False 4h41m			
operator-lifecycle-manager	4.12.0	True	False
False 4h49m			
operator-lifecycle-manager-catalog	4.12.0	True	False
False 4h49m			
operator-lifecycle-manager-packageserver	4.12.0	True	False
False 4h42m			
service-ca	4.12.0	True	False
False 4h50m			
storage	4.12.0	True	False
False 4h50m			

Example 5-41 List playbook

```
[root@bastion ocpinst]#
[admin1@controller Ansible-OpenShift-Provisioning]$ ansible-playbook
playbooks/7_ocp_verification.yaml
```

5.5 KVM three-LPAR cluster implementation

For the three-LPAR installation we did not use Ansible to provide a different perspective in case Ansible did not fit your needs. You could still use Ansible whether it is a one or three LPAR install if you wish. For this installation we used shell scripts for most of the necessary commands.

5.5.1 Architecture

The goal of the three LPAR approach is not just to potentially have access to more capacity but to provide resiliency where one of the three LPARs can be taken out of service for maintenance and the Red Hat OpenShift cluster is still fully functional.

In Figure 5-6 you can see our DNS also serves as a DHCP server in this deployment. Our three load balancers use keepalived and a floating IP address to provide a highly available load balancing layer. In this configuration, we did not use Ansible so we could better illustrate all of the steps that would be performed if you chose not to use Ansible. The Ansible playbook can deploy a three-LPAR configuration as we discuss in this section.

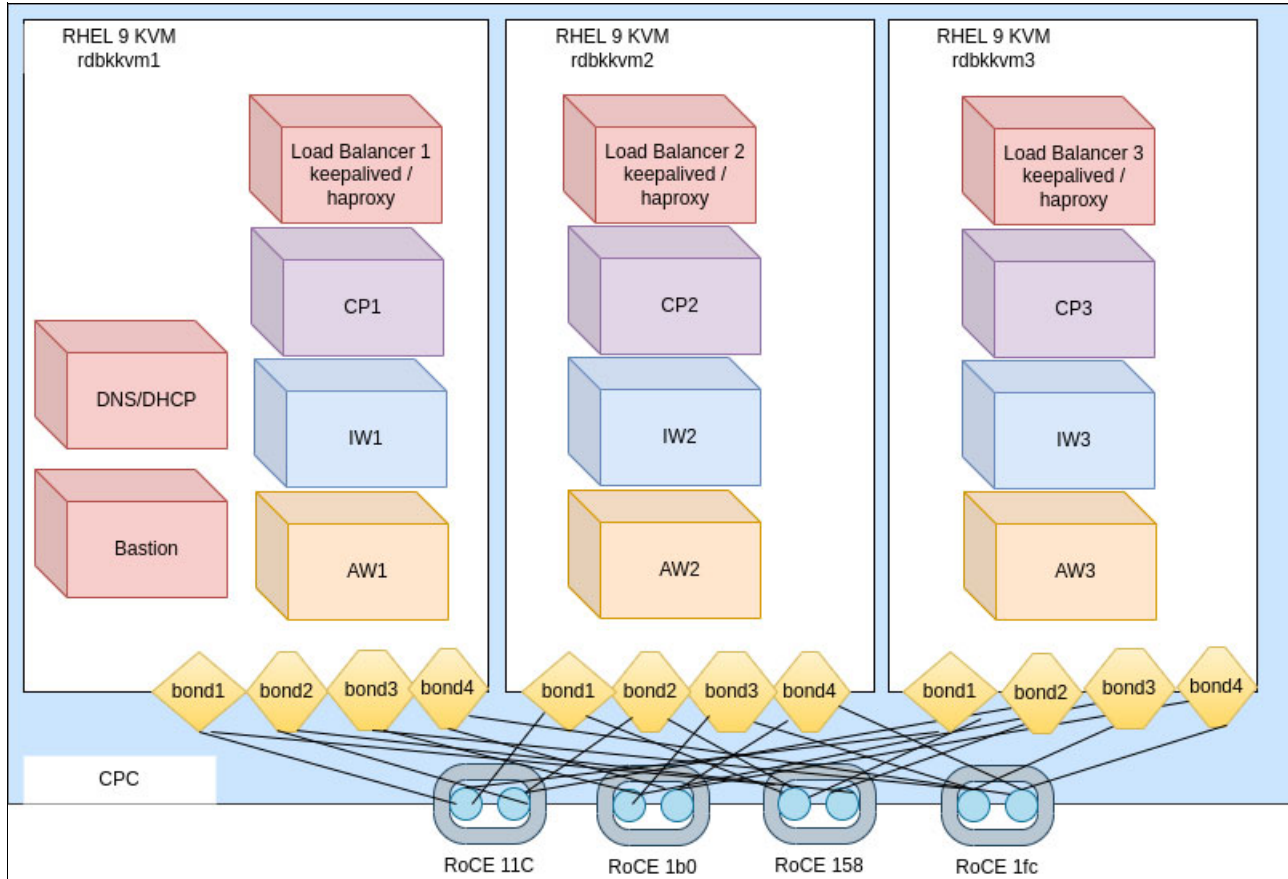


Figure 5-6 Our three LPAR architecture

The three-LPAR cluster resources plan is shown in Table 5-8 with initial resource allocations. These do not include any resources requirements for enabling additional Red Hat OpenShift features or actual application workload.

Table 5-8 Resource planning by function

Function	KVM Domain	Hypervisor	Virtual CPU	Initial Domain Memory in GB	Max QCOW2 Size GB
Bastion	bastion	rdbkkvm1	4	4096	20
DNS/DHCP	dnshcp	rdbkkvm1	4	4096	20
Load Balancer	lb1	rdbkkvm1	4	4096	20
Load Balancer	lb2	rdbkkvm2	4	4096	20
Load Balancer	lb3	rdbkkvm3	4	4096	20
Bootstrap	bootstrap	rdbkkvm1	4	16384	120
Control Plane	cp1	rdbkkvm1	4	16384	120
Control Plane	cp2	rdbkkvm2	4	16384	120
Control Plane	cp3	rdbkkvm3	4	16384	120
Application Worker	aw1	rdbkkvm1	8	16384	120
Application Worker	aw2	rdbkkvm2	8	16384	120
Application Worker	aw3	rdbkkvm3	8	16384	120
Infra Worker	iw1	rdbkkvm1	6	16384	120
Infra Worker	iw2	rdbkkvm2	6	16384	120
Infra Worker	iw3	rdbkkvm3	6	16384	120

Table 5-9 RoCE adapter and port to bond mappings

RoCE PCHID	RoCE Port	LPAR 1 Net-Dev	LPAR 2 Net-Dev	LPAR 3 Net-Dev	Bond
11c	0	ens12613	ens12614	ens12615	bond1
11c	1	ens12869	ens12870	ens12871	bond2
1b0	0	ens13125	ens13126	ens13127	bond3
1b0	1	ens13381	ens13382	ens13383	bond4
158	0	ens13637	ens13638	ens13639	bond1
158	1	ens13893	ens13894	ens13895	bond2
1fc	0	ens14149	ens14150	ens14151	bond3
1fc	1	ens14405	ens14406	ens14407	bond4

Table 5-10 Guest domain, bond, MAC, and IP address mappings

KVM Domain	Host Name (Short)	Bond	Mac	IP Address
bastion	bastion	bond1	Auto-Generate	9.76.61.23x
dnshcp	dnshcp	bond1	Auto-Generate	9.76.61.141
lb1	lb1	bond1	Auto-Generate	9.76.61.145
lb2	lb1	bond1	Auto-Generate	9.76.61.142
lb3	lb3	bond2	Auto-Generate	9.76.61.143
bootstrap	bootstrap	bond1	52:54:23:11:45:01	9.76.61.236
cp1	cp1	bond1	52:54:23:11:45:02	9.76.61.237
cp2	cp2	bond1	52:54:23:11:45:03	9.76.61.238
cp3	cp3	bond1	52:54:23:11:45:04	9.76.61.239
aw1	aw1	bond1	52:54:23:11:45:05	9.76.61.243
aw2	aw2	bond1	52:54:23:11:45:06	9.76.61.244
aw3	aw3	bond1	52:54:23:11:45:07	9.76.61.245
iw1	iw1	bond1	52:54:23:11:45:0b	9.76.61.240
iw2	iw2	bond1	52:54:23:11:45:0c	9.76.61.241
iw3	iw3	bond1	52:54:23:11:45:0d	9.76.61.242

Table 5-11 Forward and Reverse DNS

Host name	Domain	IP Address
lb0	ocp3.ibm.com	9.76.61.230
lb1	ocp3.ibm.com	9.76.61.231
lb2	ocp3.ibm.com	9.76.61.232
lb3	ocp3.ibm.com	9.76.61.233
bastion	ocp3.ibm.com	9.76.61.234
bootstrap	ocp3.ibm.com	9.76.61.236
cp1	ocp3.ibm.com	9.76.61.237
cp2	ocp3.ibm.com	9.76.61.238
cp3	ocp3.ibm.com	9.76.61.239
aw1	ocp3.ibm.com	9.76.61.243
aw2	ocp3.ibm.com	9.76.61.244
aw3	ocp3.ibm.com	9.76.61.245
iw1	ocp3.ibm.com	9.76.61.240

Host name	Domain	IP Address
iw2	ocp3.ibm.com	9.76.61.241
iw3	ocp3.ibm.com	9.76.61.242
api	ocp3.ibm.com	9.76.61.230
api-int	ocp3.ibm.com	9.76.61.230
apps	ocp3.ibm.com	9.76.61.230
*.apps	ocp3.ibm.com	9.76.61.230
dnshcp	ocp3.ibm.com	9.76.61.249

5.5.2 Hypervisor preparation

There are only a few steps we needed to take at the hypervisor level. Enable PAVs, obtain a copy of the RHEL 9 ISO image, and setup network bonds on your RoCE adapter.

Enabling the PAVs persistently can be performed with a single `chzdev` command on each of our three LPARs as shown in Example 5-42.

Example 5-42 PAV alias activation

```
[root@rdbkvm1 ~]# chzdev -e dasd-eckd 0.0.90f0-0.0.90ff
ECKD DASD 0.0.90f0 configured
ECKD DASD 0.0.90f1 configured
ECKD DASD 0.0.90f2 configured
ECKD DASD 0.0.90f3 configured
ECKD DASD 0.0.90f4 configured
ECKD DASD 0.0.90f5 configured
ECKD DASD 0.0.90f6 configured
ECKD DASD 0.0.90f7 configured
ECKD DASD 0.0.90f8 configured
ECKD DASD 0.0.90f9 configured
ECKD DASD 0.0.90fa configured
ECKD DASD 0.0.90fb configured
ECKD DASD 0.0.90fc configured
ECKD DASD 0.0.90fd configured
ECKD DASD 0.0.90fe configured
ECKD DASD 0.0.90ff configured
```

The `lsdasd` command will indicate which DASD devices are alias, as shown on Example 5-43.

Example 5-43 lsdasd with alias devices activated

```
[root@rdbkvm1 ~]# lsdasd
Bus-ID   Status   Name      Device  Type          BlkSz  Size      Blocks
=====
=
0.0.90f0 alias
0.0.90f1 alias
0.0.90f2 alias
0.0.90f3 alias
0.0.90f4 alias
0.0.90f5 alias
0.0.90f6 alias
0.0.90f7 alias
0.0.90f8 alias
0.0.90f9 alias
0.0.90fa alias
0.0.90fb alias
0.0.90fc alias
0.0.90fd alias
0.0.90fe alias
0.0.90ff alias
0.0.90dd active   dasda    94:0    ECKD (ESE)   4096   313031MB 80136000
0.0.90ed active   dasdb    94:4    ECKD (ESE)   4096   313031MB 80136000
```

To create network bonds of pairs of RoCE adapters on a LinuxONE system, we developed a small script. This script configures the bonds with ports from different RoCE cards, ensuring that they connect to separate network switches. Additionally, the script sets the Maximum Transmission Unit (MTU) size on the RoCE cards, optimizing network performance. Neglecting to set the MTU size can lead to significant performance degradation, potentially reducing it by nearly half. A sample of our script is shown in Example 5-44.

Example 5-44 Network Manager CLI (nmcli) bonds

```
#!/bin/bash

nmcli conn
nmcli dev

nmcli conn del bond1
nmcli conn del bond1-ens12613
nmcli conn del bond1-ens13637

nmcli connection add type bond con-name bond1 ifname bond1 bond.options
"mode=active-backup,fail_over_mac=1"
nmcli connection modify bond1 ipv6.method "disabled"
nmcli connection modify bond1 ipv4.method "disabled"
nmcli connection add type ethernet slave-type bond con-name bond1-ens12613
ifname ens12613 master bond1
```

```
nmcli connection add type ethernet slave-type bond con-name bond1-ens13637
ifname ens13637 master bond1
nmcli connection modify bond1 connection.autoconnect-slaves 1
nmcli connection modify bond1-eno1 802-3-ethernet.mtu 8992
nmcli connection modify bond1-eno9 802-3-ethernet.mtu 8992
nmcli connection modify bond1 802-3-ethernet.mtu 8992
nmcli connection down bond1
nmcli connection up bond1
```

```
nmcli conn del bond2
nmcli conn del bond2-ens12869
nmcli conn del bond2-ens13869
```

```
nmcli connection add type bond con-name bond2 ifname bond2 bond.options
"mode=active-backup,fail_over_mac=1"
nmcli connection modify bond2 ipv6.method "disabled"
nmcli connection modify bond2 ipv4.method "disabled"
nmcli connection add type ethernet slave-type bond con-name bond2-ens12869
ifname ens12869 master bond2
nmcli connection add type ethernet slave-type bond con-name bond2-ens13869
ifname ens13869 master bond2
nmcli connection modify bond2 connection.autoconnect-slaves 1
nmcli connection modify bond2-ens12869 802-3-ethernet.mtu 8992
nmcli connection modify bond2-ens13869 802-3-ethernet.mtu 8992
nmcli connection modify bond2 802-3-ethernet.mtu 8992
nmcli connection down bond2
nmcli connection up bond2
```

```
nmcli conn del bond3
nmcli conn del bond3-ens13125
nmcli conn del bond3-ens14149
```

```
nmcli connection add type bond con-name bond3 ifname bond3 bond.options
"mode=active-backup,fail_over_mac=1"
nmcli connection modify bond3 ipv6.method "disabled"
nmcli connection modify bond3 ipv4.method "disabled"
nmcli connection add type ethernet slave-type bond con-name bond3-ens13125
ifname ens13125 master bond3
nmcli connection add type ethernet slave-type bond con-name bond3-ens14149
ifname eno14149 master bond3
nmcli connection modify bond3 connection.autoconnect-slaves 1
nmcli connection modify bond3-ens13125 802-3-ethernet.mtu 8992
nmcli connection modify bond3-ens14149 802-3-ethernet.mtu 8992
nmcli connection modify bond3 802-3-ethernet.mtu 8992
nmcli connection down bond3
nmcli connection up bond3
```

```
nmcli conn del bond4
nmcli conn del bond4-ens13381
nmcli conn del bond4-ens14405
```

```
nmcli connection add type bond con-name bond4 ifname bond4 bond.options
"mode=active-backup,fail_over_mac=1"
nmcli connection modify bond3 ipv6.method "disabled"
nmcli connection modify bond3 ipv4.method "disabled"
nmcli connection add type ethernet slave-type bond con-name bond4-ens13381
ifname ens13381 master bond4
nmcli connection add type ethernet slave-type bond con-name bond4-ens14405
ifname eno14405 master bond4
nmcli connection modify bond4 connection.autoconnect-slaves 1
nmcli connection modify bond4-ens13381 802-3-ethernet.mtu 8992
nmcli connection modify bond4-ens14405 802-3-ethernet.mtu 8992
nmcli connection modify bond4 802-3-ethernet.mtu 8992
nmcli connection down bond4
nmcli connection up bond4

nmcli conn
nmcli dev

ip l
ip a
```

We obtained a copy of the RHEL 9 ISO by a transfer through SFTP to the /var/lib/libvirt/images directory.

5.5.3 DNS server configuration

DNS is requirement to deploy and use Red Hat OpenShift. In production environment, usually it will be the enterprise DNS servers. In our lab environment, we configured named for our DNS server. Our example DNS configuration consists of the following three files:

- ▶ Example 5-45
- ▶ Example 5-46 on page 104
- ▶ Example 5-47 on page 105

Example 5-45 File: /var/named/named.61.76.9.in-addr.arpa.zone

```
$TTL 900
@ IN SOA dnsdhcp.ocp3.ibm.com. admin.ocp3.ibm.com. (
                2022020202    ; serial
                3600    ; refresh
                1800    ; retry
                604800    ; expire
                86400    ; negative cache ttl
        )

; NameServer

@ IN NS dnsdhcp.ocp3.ibm.com.

;reverse for name server

230 IN PTR 1b0.ocp3.ibm.com.
231 IN PTR 1b1.ocp3.ibm.com.
232 IN PTR 1b2.ocp3.ibm.com.
233 IN PTR 1b3.ocp3.ibm.com.

234 IN PTR bastion.ocp3.ibm.com.
236 IN PTR bootstrap.ocp3.ibm.com.
237 IN PTR cp1.ocp3.ibm.com.
238 IN PTR cp2.ocp3.ibm.com.
239 IN PTR cp3.ocp3.ibm.com.
243 IN PTR aw1.ocp3.ibm.com.
244 IN PTR aw2.ocp3.ibm.com.
245 IN PTR aw3.ocp3.ibm.com.
240 IN PTR iw1.ocp3.ibm.com.
241 IN PTR iw2.ocp3.ibm.com.
242 IN PTR iw3.ocp3.ibm.com.
235 IN PTR nfs.ocp3.ibm.com.
249 IN PTR dnsdhcp.ocp3.ibm.com.
```

```
$TTL 3H
;base domain name
$ORIGIN ocp3.ibm.com.

@      IN SOA @ ocp3.ibm.com. (
                                2022020202    ; serial
                                1D             ; refresh
                                1H             ; retry
                                1W             ; expire
                                3H             ; minimum
                                )
;Names servers for this domain
      IN NS dnsdhcp.ocp3.ibm.com.
;Mail servers
; none
;A Records
lb0      IN A 9.76.61.230
lb1      IN A 9.76.61.231
lb2      IN A 9.76.61.232
lb3      IN A 9.76.61.233
bastion  IN A 9.76.61.234
bootstrap IN A 9.76.61.236
cp1      IN A 9.76.61.237
cp2      IN A 9.76.61.238
cp3      IN A 9.76.61.239
aw1      IN A 9.76.61.243
aw2      IN A 9.76.61.244
aw3      IN A 9.76.61.245
iw1      IN A 9.76.61.240
iw2      IN A 9.76.61.241
iw3      IN A 9.76.61.242
nfs      IN A 9.76.61.235
api      IN A 9.76.61.230
api-int  IN A 9.76.61.230
apps     IN A 9.76.61.230
*.apps   IN A 9.76.61.230
dnsdhcp  IN A 9.76.61.249
```

```
acl internal_nets { 9.76.61/24; };

options {
    listen-on port 53 { any; };
    listen-on-v6 port 53 { ::1; };
    listen-on-v6 port 53 { none; };
    directory      "/var/named";
    dump-file      "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    secroots-file  "/var/named/data/named.secroots";
    recursing-file "/var/named/data/named.recursing";
    allow-query    { localhost; internal_nets; };
    forwarders     { 9.0.0.2; };
    allow-recursion { localhost; internal_nets; };
    allow-query-cache { localhost; internal_nets; };

    recursion yes;
    dnssec-validation no;

    managed-keys-directory "/var/named/dynamic";
    geoip-directory "/usr/share/GeoIP";
    pid-file "/run/named/named.pid";
    session-keyfile "/run/named/session.key";

    querylog yes;
    allow-transfer { none; };

    include "/etc/crypto-policies/back-ends/bind.config";
};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";

zone "ocp3.ibm.com" {
    type master;
    file "/var/named/named.ocp3.ibm.com";
    allow-query { any; };
    allow-transfer { none; };
    allow-update { none; };
};
```

```
};

zone "61.76.9.in-addr.arpa" {
    type master;
    file "/var/named/named.61.76.9.in-addr.arpa.zone";
    allow-query {any;};
    allow-transfer {none;};
    allow-update {none;};
};
```

5.5.4 DHCP server configuration

DHCP deployments are valuable for a number of reasons. It allows you to centrally set and change all the IP address, MTU sizes, and DNS servers utilized by each client. In our examples, in addition to setting the IP address, we set the MTU and DNS servers that were utilized. This is very handy if you later want to switch from your own DNS server to a enterprise DNS. Example 5-48 provides an example of our configuration file to do this.

Example 5-48 File: /etc/dhcp/dhcpd.conf

```
#
# DHCP Server Configuration file.
# see /usr/share/doc/dhcp-server/dhcpd.conf.example
# see dhcpd.conf(5) man page
#

option domain-name "ocp3.ibm.com";
option domain-name-servers 9.76.61.249;

default-lease-time 86400;
authoritative;

# Host definitions

host bootstrap.ocp3.ibm.com {
    hardware ethernet 52:54:23:11:45:01;
    fixed-address 9.76.61.236;
    option domain-name-servers 9.76.61.249;
    option interface-mtu 8992;
}

host cp1.ocp3.ibm.com {
    hardware ethernet 52:54:23:11:45:02;
    fixed-address 9.76.61.237;
    option domain-name-servers 9.76.61.249;
    option interface-mtu 8992;
}

host cp2.ocp3.ibm.com {
    hardware ethernet 52:54:23:11:45:03;
    fixed-address 9.76.61.238;
```



```

    option domain-name-servers 9.76.61.249;
    option interface-mtu 8992;
}

host cp3.ocp3.ibm.com {
    hardware ethernet 52:54:23:11:45:04;
    fixed-address 9.76.61.239;
    option domain-name-servers 9.76.61.249 ;
    option interface-mtu 8992;
}

host aw1.ocp3.ibm.com {
    hardware ethernet 52:54:23:11:45:05;
    fixed-address 9.76.61.243;
    option domain-name-servers 9.76.61.249 ;
    option interface-mtu 8992;
}

host aw2.ocp3.ibm.com {
    hardware ethernet 52:54:23:11:45:06;
    fixed-address 9.76.61.244;
    option domain-name-servers 9.76.61.249;
    option interface-mtu 8992;
}

host aw3.ocp3.ibm.com {
    hardware ethernet 52:54:23:11:45:07;
    fixed-address 9.76.61.245;
    option domain-name-servers 9.76.61.249;
    option interface-mtu 8992;
}

host iw1.ocp3.ibm.com {
    hardware ethernet 52:54:23:11:45:0b;
    fixed-address 9.76.61.240;
    option domain-name-servers 9.76.61.249;
    option interface-mtu 8992;
}

host iw2.ocp3.ibm.com {
    hardware ethernet 52:54:23:11:45:0c;
    fixed-address 9.76.61.241;
    option domain-name-servers 9.76.61.249;
    option interface-mtu 8992;
}

host iw3.ocp3.ibm.com {
    hardware ethernet 52:54:23:11:45:0d;
    fixed-address 9.76.61.242;
    option domain-name-servers 9.76.61.249;
    option interface-mtu 8992;
}

subnet 9.76.61.0 netmask 255.255.255.0 {
    option subnet-mask 255.255.255.0;
}

```

```
option domain-name-servers 9.76.61.249;
option routers 9.76.61.1;
option broadcast-address 9.76.61.255;
max-lease-time 172800;

}
```

5.5.5 Highly available load balancer configuration

For the load balancer service, in this scenario, we utilized three HAProxy servers. Keepalived is used to transparently manage a floating IP address to an active HAProxy. The `haproxy.conf` configuration is common on all three load balancer virtual servers. The Keepalived configuration is unique on each of the three virtual servers

Port 1936 was enabled for HAProxy statistics monitoring. Our sample link looked as follows:

```
http://haproxy_ip_address:1936/stats
```

Example 5-49 contains a sample configuration in `/etc/haproxy/haproxy.cfg`

Example 5-49 Sample configuration in `/etc/haproxy/haproxy.cfg`

```
#-----
global
#
log          127.0.0.1 local2

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

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

# utilize system-wide crypto-policies
ssl-default-bind-ciphers PROFILE=SYSTEM
ssl-default-server-ciphers PROFILE=SYSTEM

#-----
defaults
mode                http
log                 global
option              dontlognull
option http-server-close
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 stats
  bind *:1936
  mode          http
  log           global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp3 cluster
  stats auth admin:ocp3
  stats uri /stats

listen api-server-6443
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp3.ibm.com:6443 check inter 1s backup
  server cp1      cp1.ocp3.ibm.com:6443 check inter 1s
  server cp2      cp2.ocp3.ibm.com:6443 check inter 1s
  server cp3      cp3.ocp3.ibm.com:6443 check inter 1s

listen machine-config-server-22623
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp3.ibm.com:22623 check inter 1s backup
  server cp1      cp1.ocp3.ibm.com:22623 check inter 1s
  server cp2      cp2.ocp3.ibm.com:22623 check inter 1s
  server cp3      cp3.ocp3.ibm.com:22623 check inter 1s

listen ingress-router-80
  bind *:80
  mode tcp
  balance source
    server aw1      aw1.ocp3.ibm.com:80 check inter 1s
    server aw2      aw2.ocp3.ibm.com:80 check inter 1s
    server aw3      aw3.ocp3.ibm.com:80 check inter 1s
    server iw1      iw1.ocp3.ibm.com:80 check inter 1s
    server iw2      iw2.ocp3.ibm.com:80 check inter 1s
    server iw3      iw3.ocp3.ibm.com:80 check inter 1s

listen ingress-router-443
  bind *:443
  mode tcp
  balance source
    server aw1      aw1.ocp3.ibm.com:443 check inter 1s
    server aw2      aw2.ocp3.ibm.com:443 check inter 1s
    server aw3      aw3.ocp3.ibm.com:443 check inter 1s
    server iw1      iw1.ocp3.ibm.com:443 check inter 1s
    server iw2      iw2.ocp3.ibm.com:443 check inter 1s
    server iw3      iw3.ocp3.ibm.com:443 check inter 1s
#-----

```

Our example Keepalived configuration on lb1 file /etc/keepalived/keepalived.conf is shown in Example 5-50.

Example 5-50 Sample keepalived configuration file for load balancer 1

```
global_defs {
    notification_email {
        admin1@ibm.com
    }

    notification_email_from admin1@ibm.com
    smtp_server mail.ibm.com
    smtp_connect_timeout 30
    router_id ocp3ibm
    vrrp_skip_check_adv_addr
    ! vrrp_strict
    ! vrrp_garp_interval 0
    ! vrrp_gna_interval 0
}

vrrp_instance HAProxy-vip {
    state MASTER
    priority 100
    interface enc9                # Network card
    virtual_router_id 60
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }

    unicast_src_ip 9.76.61.231    # The IP address of this machine
    unicast_peer {
        9.76.61.232                # lb2
        9.76.61.233                # lb3
    }

    virtual_ipaddress {
        9.76.61.230/23            # The VIP address
    }
}
```

Our example Keepalived configuration on lb2 file /etc/keepalived/keepalived.conf is shown in Example 5-51.

Example 5-51 Sample Keepalived configuration file for load balancer 2

```
global_defs {
    notification_email {
        admin1@ibm.com
    }

    notification_email_from admin1@ibm.com
    smtp_server mail.ibm.com
    smtp_connect_timeout 30
    router_id ocp3ibm
    vrrp_skip_check_adv_addr
!   vrrp_strict
!   vrrp_garp_interval 0
!   vrrp_gna_interval 0

}

vrrp_instance HAProxy-vip {
    state MASTER
    priority 100
    interface enc9                # Network card
    virtual_router_id 60
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }

    unicast_src_ip 9.76.61.232    # The IP address of this machine
    unicast_peer {
        9.76.61.231                # lb1
        9.76.61.233                # lb3
    }

    virtual_ipaddress {
        9.76.61.230/23            # The VIP address
    }

}
```

Our example Keepalived configuration on lb3 file /etc/keepalived/keepalived.conf is shown in Example 5-51 on page 111.

Example 5-52 Sample Keepalived configuration file for load balancer 3

```
global_defs {
    notification_email {
        admin1@ibm.com
    }

    notification_email_from admin1@ibm.com
    smtp_server mail.ibm.com
    smtp_connect_timeout 30
    router_id ocp3ibm
    vrrp_skip_check_adv_addr
!   vrrp_strict
!   vrrp_garp_interval 0
!   vrrp_gna_interval 0
}

vrrp_instance HAProxy-vip {
    state MASTER
    priority 100
    interface enc9                # Network card
    virtual_router_id 60
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }

    unicast_src_ip 9.76.61.233    # The IP address of this machine
    unicast_peer {
        9.76.61.231                # lb1
        9.76.61.232                # lb2
    }

    virtual_ipaddress {
        9.76.61.230/23            # The VIP address
    }
}
}
```

5.5.6 Rapidly creating the guests that provide the support infrastructure

We can very quickly create the guests with supporting infrastructure by using `virt-install` and Red Hat `kickstart` for a fully unattended installation. Each RHEL 9.1 server is installed in an automated fashion. On our virtual machine, `rdbkvm1`, we install the bastion, `dnshcp`, load balancer 1(`lb1`), and NFS. On `rdbkvm2`, we install `lb2`, and on `rdbkvm3` we install `lb3`. The `virt-install` commands are slightly longer than they need to be as we incorporated some of the best practices up front such as `iothreads`, `caching`, and `multiple queues`.

On `rdbkvm1`, we issue the following `virt-install` commands, which could be placed in single or multiple script files:

- For the guest domain `bastion`:

```
virt-install --input keyboard,bus=virtio --input mouse,bus=virtio --graphics vnc,listen=0.0.0.0 --video virtio --name bastion --memory 16384 --vcpus=4 --disk size=20,cache=none,sparse=yes,driver.iothread=1 --network network=default --network type=direct,source=bond1,source_mode=bridge,address.cs sid=0xfe,address.ssid=0,address.devno=0x0009,address.type=ccw --location /var/lib/libvirt/images/RHEL-9.1.0-20221027.3-s390x-dvd1.iso --initrd-inject=/root/bastion.ks --extra-args "inst.ks=file:/bastion.ks" --console pty,target_type=serial --autoconsole text
```

- For the guest domain `dnshcp`:

```
virt-install --input keyboard,bus=virtio --input mouse,bus=virtio --graphics vnc,listen=0.0.0.0 --video virtio --name dnshcp --memory 16384 --vcpus=4 --iothreads=2 --disk size=20,format=qcow2,cache=none,sparse=yes,driver.iothread=1 --network network=default --network type=direct,source=bond1,source_mode=bridge,address.cssid=0xfe,address.ssid=0, address.devno=0x0009,address.type=ccw,driver.queues=4 --location /var/lib/libvirt/images/RHEL-9.1.0-20221027.3-s390x-dvd1.iso --initrd-inject=/root/dnshcp.ks --extra-args "inst.ks=file:/dnshcp.ks" --console pty,target_type=serial --autoconsole text
```

- For the guest domain `lb1`:

```
virt-install --input keyboard,bus=virtio --input mouse,bus=virtio --graphics vnc,listen=0.0.0.0 --video virtio --name lb1 --memory 16384 --vcpus=4 --iothreads=2 --disk size=20,format=qcow2,cache=none,sparse=yes,driver.iothread=1 --network network=default --network type=direct,source=bond1,source_mode=bridge,address.cssid=0xfe,address.ssid=0, address.devno=0x0009,address.type=ccw,driver.queues=4 --location /var/lib/libvirt/images/RHEL-9.1.0-20221027.3-s390x-dvd1.iso --initrd-inject=/root/lb1.ks --extra-args "inst.ks=file:/lb1.ks" --console pty,target_type=serial --autoconsole text
```

- For the guest domain NFS:

```
virt-install --input keyboard,bus=virtio --input mouse,bus=virtio --graphics
vnc,listen=0.0.0.0 --video virtio --name nfs --memory 16384 --vcpus=4
--iothreads=2 --disk
size=20,format=qcow2,cache=none,sparse=yes,driver.iothread=1 --network
network=default --network
type=direct,source=bond1,source_mode=bridge,address.cssid=0xfe,address.ssid=0,a
ddress.devno=0x0009,address.type=ccw,driver.queues=4 -
--location /var/lib/libvirt/images/RHEL-9.1.0-20221027.3-s390x-dvd1.iso
--initrd-inject=/root/nfs.ks --extra-args "inst.ks=file:/nfs.ks"
--console pty,target_type=serial --autoconsole text
```

On rdbkvm2, we issue the following `virt-install` commands, which could be placed in a script file:

```
virt-install --input keyboard,bus=virtio --input mouse,bus=virtio --graphics
vnc,listen=0.0.0.0 --video virtio --name lb2 --memory 16384 --vcpus=4
--iothreads=2 --disk
size=20,format=qcow2,cache=none,sparse=yes,driver.iothread=1 --network
network=default --network
type=direct,source=bond1,source_mode=bridge,address.cssid=0xfe,address.ssid=0,a
ddress.devno=0x0009,address.type=ccw,driver.queues=4 --location
/var/lib/libvirt/images/RHEL-9.1.0-20221027.3-s390x-dvd1.iso
--initrd-inject=/root/lb2.ks --extra-args "inst.ks=file:/lb2.ks" --console
pty,target_type=serial --autoconsole text
```

On rdbkvm3, we issue the following `virt-install` commands, which could be placed in a script file. For guest domain lb3:

```
virt-install --input keyboard,bus=virtio --input mouse,bus=virtio --graphics
vnc,listen=0.0.0.0 --video virtio --name lb3 --memory 16384 --vcpus=4
--iothreads=2 --disk
size=20,format=qcow2,cache=none,sparse=yes,driver.iothread=1 --network
network=default --network
type=direct,source=bond1,source_mode=bridge,address.cssid=0xfe,address.ssid=0,a
ddress.devno=0x0009,address.type=ccw,driver.queues=4 --location
/var/lib/libvirt/images/RHEL-9.1.0-20221027.3-s390x-dvd1.iso
--initrd-inject=/root/lb3.ks --extra-args "inst.ks=file:/lb3.ks" --console
pty,target_type=serial --autoconsole text
```


For each of the guest domains and virt-install commands there is a unique Kickstart file that we used. They are mostly similar except for package content that is unique per the function of the server. We do also use the full domain name as the hostname. We included the qualifier that has the cluster name, ocp3. This can help add clarity when you are managing multiple clusters.

Here is our Kickstart file for the bastion node, which is named bastion.ks:

```
authselect --enablshadow --passalgo=sha512
cdrom
text
firstboot --enable
ignoredisk --only-use=vda
keyboard --vckeymap=us --xlayouts='us'
lang en_US.UTF-8
# Network information
firewall --enabled --ssh
network --device=encl --bootproto=dhcp --noipv6
network --device=enc9 --bootproto=static --noipv6 --noipv4
network --hostname=bastion.ocp3.ibm.com
rootpw --plaintext its0
# System services
services --enabled="chronyd"
# System timezone
timezone America/New_York
user --groups=wheel --name=admin1 --password=its0 --plaintext --gecos="admin1"
# System bootloader configuration
bootloader --append="crashkernel=auto dfltcc=always transparent_hugepages=never
" --location=mbr --boot-drive=vda
#Partitioning
clearpart --all --initlabel --disklabel=gpt --drives=vda
part / --fstype="xfs" --ondisk=vda --grow
#Packages
%packages --ignoremissing --instLangs=en_US
@base
@core
chrony
sysstat
tmux
iotop
cockpit
kexec-tools
%end
%addon com_redhat_kdump --enable --reserve-mb='auto'
%end
%post --log=/root/ks-post.log
ip addr
df -h
%end
eula --agreed
reboot
```

Here is the Kickstart file for dnsdhcp, which is named dnsdhcp.ks:

```
authselect --enableshadow --passalgo=sha512
cdrom
text
firstboot --enable
ignoredisk --only-use=vda
keyboard --vckeymap=us --xlayouts='us'
lang en_US.UTF-8
# Network information
firewall --enabled --ssh
network --device=encl --bootproto=dhcp --noipv6
network --device=enc9 --bootproto=static --noipv6 --noipv4
network --hostname=dnsdhcp.ocp3.ibm.com
rootpw --plaintext its0
# System services
services --enabled="chronyd"
# System timezone
timezone America/New_York
user --groups=wheel --name=admin1 --password=its0 --plaintext --gecos="admin1"
# System bootloader configuration
bootloader --append="crashkernel=auto dfltcc=always transparent_hugepages=never
" --location=mbr --boot-drive=vda
#Partitioning
clearpart --all --initlabel --disklabel=gpt --drives=vda
part / --fstype="xfs" --ondisk=vda --grow
#Packages
%packages --ignoremissing --instLangs=en_US
@base
@core
bind
dhcp-server
chrony
sysstat
tmux
iotop
cockpit
kexec-tools
%end
%addon com_redhat_kdump --enable --reserve-mb='auto'
%end
%post --log=/root/ks-post.log
ip addr
df -h
%end
eula --agreed
reboot
```

Here is the Kickstart file for lb1, which is named lb1.ks:

```
authselect --enablshadow --passalgo=sha512
cdrom
text
firstboot --enable
ignoredisk --only-use=vda
keyboard --vckeymap=us --xlayouts='us'
lang en_US.UTF-8
# Network information
firewall --enabled --ssh
network --device=encl --bootproto=dhcp --noipv6
network --device=enc9 --bootproto=static --noipv6 --noipv4
network --hostname=lb1.ocp3.ibm.com
rootpw --plaintext its0
# System services
services --enabled="chronyd"
# System timezone
timezone America/New_York
user --groups=wheel --name=admin1 --password=its0 --plaintext --gecos="admin1"
# System bootloader configuration
bootloader --append="crashkernel=auto dfltcc=always transparent_hugepages=never
" --location=mbr --boot-drive=vda
#Partitioning
clearpart --all --initlabel --disklabel=gpt --drives=vda
part / --fstype="xfs" --ondisk=vda --grow
#Packages
%packages --ignoremissing --instLangs=en_US
@base
@core
haproxy
keepalived
chrony
sysstat
tmux
iotop
cockpit
kexec-tools
%end
%addon com_redhat_kdump --enable --reserve-mb='auto'
%end
%post --log=/root/ks-post.log
ip addr
df -h
%end
eula --agreed
reboot
```

Here is the Kickstart file for lb2, which is named lb2.ks:

```
authselect --enablshadow --passalgo=sha512
cdrom
text
firstboot --enable
ignoredisk --only-use=vda
keyboard --vckeymap=us --xlayouts='us'
lang en_US.UTF-8
# Network information
firewall --enabled --ssh
network --device=encl --bootproto=dhcp --noipv6
network --device=enc9 --bootproto=static --noipv6 --noipv4
network --hostname=lb1.ocp3.ibm.com
rootpw --plaintext its0
# System services
services --enabled="chronyd"
# System timezone
timezone America/New_York
user --groups=wheel --name=admin1 --password=its0 --plaintext --gecos="admin1"
# System bootloader configuration
bootloader --append="crashkernel=auto dfltcc=always transparent_hugepages=never
" --location=mbr --boot-drive=vda
#Partitioning
clearpart --all --initlabel --disklabel=gpt --drives=vda
part / --fstype="xfs" --ondisk=vda --grow
#Packages
%packages --ignoremissing --instLangs=en_US
@base
@core
haproxy
keepalived
chrony
sysstat
tmux
iotop
cockpit
kexec-tools
%end
%addon com_redhat_kdump --enable --reserve-mb='auto'
%end
%post --log=/root/ks-post.log
ip addr
df -h
%end
eula --agreed
reboot
```

Here is the Kickstart file for lb3, which is named lb3.ks:

```
authselect --enablshadow --passalgo=sha512
cdrom
text
firstboot --enable
ignoredisk --only-use=vda
keyboard --vckeymap=us --xlayouts='us'
lang en_US.UTF-8
# Network information
firewall --enabled --ssh
network --device=encl --bootproto=dhcp --noipv6
network --device=enc9 --bootproto=static --noipv6 --noipv4
network --hostname=lb3.ocp3.ibm.com
rootpw --plaintext its0
# System services
services --enabled="chronyd"
# System timezone
timezone America/New_York
user --groups=wheel --name=admin1 --password=its0 --plaintext --gecos="admin1"
# System bootloader configuration
bootloader --append="crashkernel=auto dfltcc=always transparent_hugepages=never
" --location=mbr --boot-drive=vda
#Partitioning
clearpart --all --initlabel --disklabel=gpt --drives=vda
part / --fstype="xfs" --ondisk=vda --grow
#Packages
%packages --ignoremissing --instLangs=en_US
@base
@core
haproxy
keepalived
chrony
sysstat
tmux
iotop
cockpit
kexec-tools
%end
%addon com_redhat_kdump --enable --reserve-mb='auto'
%end
%post --log=/root/ks-post.log
ip addr
df -h
%end
eula --agreed
reboot
```

Here is the Kickstart file for nfs, which is named nfs.ks:

```
authselect --enablesshadow --passalgo=sha512
cdrom
text
firstboot --enable
ignoredisk --only-use=vda
keyboard --vckeymap=us --xlayouts='us'
lang en_US.UTF-8
# Network information
firewall --enabled --ssh
network --device=encl --bootproto=dhcp --noipv6
network --device=enc9 --bootproto=static --noipv6 --noipv4
network --hostname=nfs.ocp3.ibm.com
rootpw --plaintext its0
# System services
services --enabled="chronyd"
# System timezone
timezone America/New_York
user --groups=wheel --name=admin1 --password=its0 --plaintext --gecos="admin1"
# System bootloader configuration
bootloader --append="crashkernel=auto dfltcc=always transparent_hugepages=never
" --location=mbr --boot-drive=vda
#Partitioning
clearpart --all --initlabel --disklabel=gpt --drives=vda
part / --fstype="xfs" --ondisk=vda --grow
#Packages
%packages --ignoremissing --instLangs=en_US
@base
@core
nfs-utils
chrony
sysstat
tmux
iotop
cockpit
kexec-tools
%end
%addon com_redhat_kdump --enable --reserve-mb='auto'
%end
%post --log=/root/ks-post.log
ip addr
df -h
%end
eula --agreed
reboot
```

5.5.7 Creating the Red Hat OpenShift cluster

Creating the Red Hat OpenShift Cluster after the all the required resources are in place involves obtaining the CLI and installation code for Red Hat OpenShift, creating the `install-config.yaml` file and `install-config` directory, building the ignition files, creating the CoreOS guests on the KVM hosts, and signing the certificate requests. Except for creating the CoreOS guests, all steps are performed from the bastion host.

We place all of our artifacts in `/root/ocp412` on our bastion host.

We start with a short script to obtain the Red Hat OpenShift CLI and installation code, shown in Example 5-53.

Example 5-53 Example `getTheCode.sh`

```
#!/bin/bash

wget
https://mirror.openshift.com/pub/openshift-v4/s390x/clients/ocp/stable/openshift-client-linux.tar.gz
wget
https://mirror.openshift.com/pub/openshift-v4/s390x/clients/ocp/stable/openshift-install-linux.tar.gz

tar -xzvf openshift-client-linux.tar.gz
tar -xzvf openshift-install-linux.tar.gz
```

An example of our `install-config.yaml` is provided in Example 5-54.

Example 5-54 Example `install-config.yaml`

```
apiVersion: v1
baseDomain: ibm.com
compute:
- hyperthreading: Enabled
  name: worker
  replicas: 0
  architecture: s390x
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
  architecture: s390x
metadata:
  name: ocp3
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OVNKubernetes
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
```

```
fips: false
pullSecret: '<<<Your Pull Secret Here>>>'
sshKey: 'ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIIj28PrpMZ0khsnoIN1+UzaDv0DsHnqDRkr+sZZKVZ2U
root@bastion.ocp3.ibm.com'
```

We created a simple script called `buildit.sh` to combine the steps we need to execute. It is shown in Example 5-55.

Example 5-55 Sample script

```
#!/bin/bash

cd /root/ocp412
mkdir /root/ocp412/install-config
cp install-config.yaml /root/ocp412/install-config
./openshift-install create manifests --dir /root/ocp412/install-config
sed /master/s/true/false/ -i
/root/ocp412/install-config/manifests/cluster-scheduler-02-config.yml
./openshift-install create ignition-configs --dir /root/ocp412/install-config
```

Next, we create and run a script called `sendit.sh` (Example 5-56). This uploads the ignition files to the respective KVM hosts. SSH keys were exchanged in advance.

Example 5-56 Example `sendit.sh` script

```
#!/bin/bash

sftp root@192.168.122.1 <<DELIM
cd /var/lib/libvirt/images
lcd /root/ocp412/install-config
mput *ign
ls
DELIM

sftp root@9.76.61.186 <<DELIM
cd /var/lib/libvirt/images
lcd /root/ocp412/install-config
mput *ign
ls
DELIM

sftp root@9.76.61.182 <<DELIM
cd /var/lib/libvirt/images
lcd /root/ocp412/install-config
mput *ign
ls
DELIM
```

We created a simple two-line script to automate the necessary environment entries called `setupbashrc.sh` and is shown in Example 5-57.

Example 5-57 Example setupBashrc.sh

```
#!/bin/bash
echo "export KUBECONFIG=/root/ocp412/install-config/auth/kubeconfig" >>
~/.bashrc
echo "export PATH=$PATH:/root/ocp412" >> ~/.bashrc
```

At this point we need to create the Red Hat OpenShift CoreOS nodes from each of the KVM hosts. Example 5-58 to Example 5-60 on page 127 show the scripts that are used.

Example 5-58 Example rebuild412.sh for hypervisor rdbkvm1

```
#!/bin/bash

cp /var/lib/libvirt/images/bootstrap.ign /
cp /var/lib/libvirt/images/worker.ign /
cp /var/lib/libvirt/images/master.ign /
chown qemu /*.ign

chmod 755 /*.ign
ausearch -c 'qemu-kvm' --raw | audit2allow -M my-qemukvm ; semodule -X 300 -i
my-qemukvm.pp
semanage fcontext -a -t virt_image_t '/worker.ign' ; restorecon -v
'/worker.ign'
semanage fcontext -a -t virt_image_t '/master.ign' ; restorecon -v
'/master.ign'
semanage fcontext -a -t virt_image_t '/bootstrap.ign' ; restorecon -v
'/bootstrap.ign'

virsh list

virsh destroy bootstrap
virsh destroy cp1
virsh destroy aw1
virsh destroy iw1

virsh list

virsh undefine bootstrap
virsh undefine cp1
virsh undefine aw1
virsh undefine iw1

virsh list --all

rm -rf /var/lib/libvirt/images/bootstrap.qcow2
rm -rf /var/lib/libvirt/images/cp1.qcow2
rm -rf /var/lib/libvirt/images/aw1.qcow2
rm -rf /var/lib/libvirt/images/iw1.qcow2

cd /var/lib/libvirt/images
```

```
qemu-img create -f qcow2 -F qcow2 -b
/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2 bootstrap.qcow2
100g
qemu-img create -f qcow2 -F qcow2 -b
/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2 cp1.qcow2 100g
qemu-img create -f qcow2 -F qcow2 -b
/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2 aw1.qcow2 100g
qemu-img create -f qcow2 -F qcow2 -b
/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2 iw1.qcow2 100g
```

```
virt-install --noautoconsole --connect qemu:///system --name bootstrap
--memory 16384 --vcpus 4 --disk /var/lib/libvirt/images/bootstrap.qcow2
--accelerate --import --network
type=direct,model=virtio,source_mode=bridge,source=bond1,mac=52:54:23:11:45:01
--osinfo rhel8.6 --qemu-commandline="-drive
if=none,id=ignition,format=raw,file=/bootstrap.ign,readonly=on -device
virtio-blk,serial=ignition,drive=ignition,devno=fe.0.1234"
```

```
virt-install --noautoconsole --connect qemu:///system --name cp1
--memory 16384 --vcpus 4 --disk /var/lib/libvirt/images/cp1.qcow2
--accelerate --import --network
type=direct,model=virtio,source_mode=bridge,source=bond1,mac=52:54:23:11:45:02
--osinfo rhel8.6 --qemu-commandline="-drive
if=none,id=ignition,format=raw,file=/master.ign,readonly=on -device
virtio-blk,serial=ignition,drive=ignition,devno=fe.0.1234"
```

```
virt-install --noautoconsole --connect qemu:///system --name aw1
--memory 16384 --vcpus 8 --disk /var/lib/libvirt/images/aw1.qcow2
--accelerate --import --network
type=direct,model=virtio,source_mode=bridge,source=bond1,mac=52:54:23:11:45:05
--osinfo rhel8.6 --qemu-commandline="-drive
if=none,id=ignition,format=raw,file=/worker.ign,readonly=on -device
virtio-blk,serial=ignition,drive=ignition,devno=fe.0.1234"
```

```
virt-install --noautoconsole --connect qemu:///system --name iw1
--memory 16384 --vcpus 6 --disk /var/lib/libvirt/images/iw1.qcow2
--accelerate --import --network
type=direct,model=virtio,source_mode=bridge,source=bond1,mac=52:54:23:11:45:0b
--osinfo rhel8.6 --qemu-commandline="-drive
if=none,id=ignition,format=raw,file=/worker.ign,readonly=on -device
virtio-blk,serial=ignition,drive=ignition,devno=fe.0.1234"
```

```
#!/bin/bash

cp /var/lib/libvirt/images/bootstrap.ign /
cp /var/lib/libvirt/images/worker.ign /
cp /var/lib/libvirt/images/master.ign /
chown qemu /*.ign

chmod 755 /*.ign
#ausearch -c 'qemu-kvm' --raw | audit2allow -M my-qemukvm ; semodule -X 300 -i
my-qemukvm.pp

semanage fcontext -a -t virt_image_t '/worker.ign' ; restorecon -v
'/worker.ign'
semanage fcontext -a -t virt_image_t '/master.ign' ; restorecon -v
'/master.ign'
semanage fcontext -a -t virt_image_t '/bootstrap.ign' ; restorecon -v
'/bootstrap.ign'

virsh list

virsh destroy cp2
virsh destroy aw2
virsh destroy iw2

virsh list

virsh undefine cp2
virsh undefine aw2
virsh undefine iw2

virsh list --all

rm -rf /var/lib/libvirt/images/cp2.qcow2
rm -rf /var/lib/libvirt/images/aw2.qcow2
rm -rf /var/lib/libvirt/images/iw2.qcow2

cd /var/lib/libvirt/images

qemu-img create -f qcow2 -F qcow2 -b
/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2 cp2.qcow2 120g
qemu-img create -f qcow2 -F qcow2 -b
/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2 aw2.qcow2 120g
qemu-img create -f qcow2 -F qcow2 -b
/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2 iw2.qcow2 120g

virt-install --noautoconsole --connect qemu:///system --name cp2
--memory 16384 --vcpus 4 --disk /var/lib/libvirt/images/cp2.qcow2
--accelerate --import --network
type=direct,model=virtio,source_mode=bridge,source=bond1,mac=52:54:23:11:45:03
--osinfo rhel8.6 --qemu-commandline="-drive
```

```
if=none,id=ignition,format=raw,file=/master.ign,readonly=on -device  
virtio-blk,serial=ignition,drive=ignition,devno=fe.0.1234"
```

```
virt-install --noautoconsole --connect qemu:///system --name aw2  
--memory 16384 --vcpus 8 --disk /var/lib/libvirt/images/aw2.qcow2  
--accelerate --import --network  
type=direct,model=virtio,source_mode=bridge,source=bond1,mac=52:54:23:11:45:06  
--osinfo rhel8.6 --qemu-commandline="-drive  
if=none,id=ignition,format=raw,file=/worker.ign,readonly=on -device  
virtio-blk,serial=ignition,drive=ignition,devno=fe.0.1234"
```

```
virt-install --noautoconsole --connect qemu:///system --name iw2  
--memory 16384 --vcpus 6 --disk /var/lib/libvirt/images/iw2.qcow2  
--accelerate --import --network  
type=direct,model=virtio,source_mode=bridge,source=bond1,mac=52:54:23:11:45:0c  
--osinfo rhel8.6 --qemu-commandline="-drive  
if=none,id=ignition,format=raw,file=/worker.ign,readonly=on -device  
virtio-blk,serial=ignition,drive=ignition,devno=fe.0.1234"
```

```
#!/bin/bash

cp /var/lib/libvirt/images/bootstrap.ign /
cp /var/lib/libvirt/images/worker.ign /
cp /var/lib/libvirt/images/master.ign /
chown qemu /*.ign

chmod 755 /*.ign

semanage fcontext -a -t virt_image_t '/worker.ign' ; restorecon -v
'/worker.ign'
semanage fcontext -a -t virt_image_t '/master.ign' ; restorecon -v
'/master.ign'
semanage fcontext -a -t virt_image_t '/bootstrap.ign' ; restorecon -v
'/bootstrap.ign'

virsh list

virsh destroy cp3
virsh destroy aw3
virsh destroy iw3

virsh list

virsh undefine cp3
virsh undefine aw3
virsh undefine iw3

virsh list --all

rm -rf /var/lib/libvirt/images/cp3.qcow2
rm -rf /var/lib/libvirt/images/aw3.qcow2
rm -rf /var/lib/libvirt/images/iw3.qcow2

cd /var/lib/libvirt/images

qemu-img create -f qcow2 -F qcow2 -b
/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2 cp3.qcow2 120g
qemu-img create -f qcow2 -F qcow2 -b
/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2 aw3.qcow2 120g
qemu-img create -f qcow2 -F qcow2 -b
/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2 iw3.qcow2 120g

virt-install --noautoconsole --connect qemu:///system --name cp3
--memory 16384 --vcpus 4 --disk /var/lib/libvirt/images/cp3.qcow2
--accelerate --import --network
type=direct,model=virtio,source_mode=bridge,source=bond1,mac=52:54:23:11:45:04
--osinfo rhel8.6 --qemu-commandline="-drive
if=none,id=ignition,format=raw,file=/master.ign,readonly=on -device
virtio-blk,serial=ignition,drive=ignition,devno=fe.0.1234"
```

```

virt-install --noautoconsole --connect qemu:///system --name aw3
--memory 16384 --vcpus 8 --disk /var/lib/libvirt/images/aw3.qcow2
--accelerate --import --network
type=direct,model=virtio,source_mode=bridge,source=bond1,mac=52:54:23:11:45:07
--osinfo rhel8.6 --qemu-commandline="-drive
if=none,id=ignition,format=raw,file=/worker.ign,readonly=on -device
virtio-blk,serial=ignition,drive=ignition,devno=fe.0.1234"

virt-install --noautoconsole --connect qemu:///system --name iw3
--memory 16384 --vcpus 6 --disk /var/lib/libvirt/images/iw3.qcow2
--accelerate --import --network
type=direct,model=virtio,source_mode=bridge,source=bond1,mac=52:54:23:11:45:0d
--osinfo rhel8.6 --qemu-commandline="-drive
if=none,id=ignition,format=raw,file=/worker.ign,readonly=on -device
virtio-blk,serial=ignition,drive=ignition,devno=fe.0.1234"

```

Here is an example output from running `rebuild412.sh` on the first LPAR is shown in Example 5-61. Run the script on all three LPARs as each of the scripts is unique.

Example 5-61 Example output from running `rebuild412.sh` on the first LPAR

```

[root@rdbkvm1 ~]# ./rebuild412.sh
***** IMPORTANT *****
To make this policy package active, execute:

semodule -i my-qemukvm.pp

Relabeled /worker.ign from unconfined_u:object_r:admin_home_t:s0 to
unconfined_u:object_r:virt_image_t:s0
Relabeled /master.ign from unconfined_u:object_r:admin_home_t:s0 to
unconfined_u:object_r:virt_image_t:s0
Relabeled /bootstrap.ign from unconfined_u:object_r:admin_home_t:s0 to
unconfined_u:object_r:virt_image_t:s0
  Id   Name      State
-----
  1    nfs        running
  2    lb1        running
  3    dnshcp     running
  4    bastion    running

error: failed to get domain 'bootstrap'

error: failed to get domain 'cp1'

error: failed to get domain 'aw1'

error: failed to get domain 'iw1'

  Id   Name      State
-----
  1    nfs        running
  2    lb1        running
  3    dnshcp     running
  4    bastion    running

```

error: failed to get domain 'bootstrap'

error: failed to get domain 'cp1'

error: failed to get domain 'aw1'

error: failed to get domain 'iw1'

```
Id   Name      State
-----
1    nfs       running
2    lb1       running
3    dnshcp    running
4    bastion   running
```

```
Formatting 'bootstrap.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off
compression_type=zlib size=128849018880
backing_file=/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2
backing_fmt=qcow2 lazy_refcounts=off refcount_bits=16
Formatting 'cp1.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off
compression_type=zlib size=128849018880
backing_file=/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2
backing_fmt=qcow2 lazy_refcounts=off refcount_bits=16
Formatting 'aw1.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off
compression_type=zlib size=128849018880
backing_file=/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2
backing_fmt=qcow2 lazy_refcounts=off refcount_bits=16
Formatting 'iw1.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off
compression_type=zlib size=128849018880
backing_file=/var/lib/libvirt/images/rhcos-4.12.17-s390x-qemu.s390x.qcow2
backing_fmt=qcow2 lazy_refcounts=off refcount_bits=16
```

```
Starting install...
Creating domain...
| 0 B 00:00:00
Domain creation completed.
```

```
Starting install...
Creating domain...
| 0 B 00:00:00
Domain creation completed.
```

```
Starting install...
Creating domain...
| 0 B 00:00:00
Domain creation completed.
```

```
Starting install...
Creating domain...
| 0 B 00:00:00
Domain creation completed.
[root@rdbkvm1 ~]# virsh list
Id   Name      State
-----
1    nfs       running
```

2	lb1	running
3	dnshcp	running
4	bastion	running
23	bootstrap	running
24	cp1	running
25	aw1	running
26	iw1	running

5.5.8 Completing the initial installation of Red Hat OpenShift 4.12

After creating CoreOS nodes, the initial installation of Red Hat OpenShift requires several steps to be completed. Common issues stem from misconfigured DNS or Load Balancer settings. To ensure proper configuration, follow these steps:

- ▶ Validate that the CoreOS nodes have correctly propagated their DNS records.
- ▶ Configure the DNS server to forward queries for the OpenShift cluster domain to the appropriate IP addresses.
- ▶ Set up a Load Balancer to distribute traffic across the OpenShift cluster nodes.
- ▶ Verify that the Load Balancer is correctly routing traffic to the OpenShift cluster nodes.
- ▶ Check that the OpenShift cluster is accessible from the internet.

The first thing you need to do is wait for the bootstrap to complete, as shown in Example 5-62

Example 5-62 wait-for bootstrap-complete

```
[root@bastion ocp412]# ./openshift-install --dir /root/ocp412/install-config
wait-for bootstrap-complete --log-level=info
INFO Waiting up to 20m0s (until 9:44AM) for the Kubernetes API at
https://api.ocp3.ibm.com:6443...
INFO API v1.25.8+37a9a08 up
INFO Waiting up to 30m0s (until 9:54AM) for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
INFO Time elapsed: 0s
[root@bastion ocp412]#
```

When you see the INFO It is now safe to remove the bootstrap resources message, remove the bootstrap entries from the Load Balancer configuration. At this point the bootstrap node is no longer required. You may shut it down and remove it. Figure 5-7 shows the HAProxy Statistics Report console. You can see our bootstrap entry shutting down and the three control plane nodes are up and green. After you remove the bootstrap entry from the configuration, the red lines for bootstrap will no longer show in the statistics display.

HAProxy
Statistics Report for pid 1815 on lb3.ocp3.ibm.com: Stats for ocp3 cluster

> **General process information**

pid = 1815 (process #1, nproc = 1, nthread = 4)
 uptime = 0s 0h25m24s
 system limits: memmax = unlimited; ulimit_n = 8059
 maxsock = 8059; maxconn = 4000; maxpipes = 0
 current conn = 64; current pipes = 0/0; conn rate = 1/sec; bit rate = 4.055 Mbps
 Running tasks: 0/102; idle = 100 %

active UP backup UP
 active UP, going down backup UP, going down
 active DOWN, going up backup DOWN, going up
 active or backup DOWN not checked
 active or backup DOWN for maintenance (MAINT)
 active or backup SOFT STOPPED for maintenance
 Note: "NOLE" / "DRAIN" = UP with load-balancing disabled.

Display option:
 • Scope: []
 • Hide DOWN servers
 • Disable refresh
 • Refresh now
 • CSV export
 • JSON export (schema)

External resources:
 • Primary site
 • Updates (v2.4)
 • Online manual

Queue			Session rate			Sessions			Bytes			Denied			Errors			Warnings			Status			Server															
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis		LastChk	Wght	Act	Bck	Chk	Dwn	Dwntime	Thrtle										
stats																																							
Frontend			0	2	-	2	4	10	82			20 283	2 223 709	0	0	0	0	0	0	0	0	0	0																
api-server-6442																																							
Frontend			0	0	-	61	253	3 000	703			19 617 668	359 798 699	0	0	0	0	0	0	0	0	0	0																
bootstrap			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m24s	DOWN	L4CON in 0ms	1/1	-	Y	1	1	25m24s	-						
cp1			0	0	-	79	4	69	-	281	281	23s	12 025 887	219 774 091	0	0	0	0	0	0	0	0	0	90	9m6s	UP	L4OK in 0ms	1/1	Y	-	3	1	5s	-					
cp2			0	0	-	99	31	121	-	292	292	23s	4 771 577	47 956 492	0	0	0	1	0	0	0	1	0	40	14m54s	UP	L4OK in 0ms	1/1	Y	-	3	1	5s	-					
cp3			0	0	-	99	26	110	-	294	294	22s	2 820 204	92 068 116	0	0	0	1	0	0	0	1	0	34	11m53s	UP	L4OK in 0ms	1/1	Y	-	3	1	18s	-					
Backend			0	0	-	0	201	61	252	300	703	867	22s	19 617 668	359 798 699	0	0	0	0	2	0	0	0	164	25m24s	UP		3/3	3	0	0	0	0s						
machine-config-server-22823																																							
Frontend			0	1	-	0	2	3 000	3			1 638	1 048 884	0	0	0	0	0	0	0	0	0	0																
bootstrap			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m55s	DOWN	L4CON in 0ms	1/1	-	Y	1	1	25m23s	-						
cp1			0	0	-	1	0	1	-	1	1	9m48s	546	349 628	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-	
cp2			0	0	-	1	0	1	-	1	1	7m58s	546	349 628	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-
cp3			0	0	-	1	0	1	-	1	1	6m8s	546	349 628	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-
Backend			0	0	-	0	1	300	3	3	3	6m8s	1 638	1 048 884	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-
ingress-router-80																																							

Figure 5-7 HAProxy statistics display first page

Figure 5-8 show the bottom half of the load balancer statistics display. For ports 80 and 443, notice that only iw1 and iw2 are green. This is the expected initial state because by default there are two replicas of the ingress router. They can exist on any of our compute nodes during the initial installation configuration. It was just by chance that they were placed on iw1 and iw2. Use label infrastructure nodes and taints to ensure the ingress router only runs on the infrastructure nodes.

Queue			Session rate			Sessions			Bytes			Denied			Errors			Warnings			Status			Server																	
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis		LastChk	Wght	Act	Bck	Chk	Dwn	Dwntime	Thrtle												
api-server-6443																																									
Frontend			0	0	-	0	0	3 000	187			94 915	705 648	0	0	0	0	0	0	0	0	0	0																		
bootstrap			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m54s	DOWN	L4CON in 0ms	1/1	Y	-	1	1	25m54s	-								
cp1			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-	
cp2			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-
cp3			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-
Backend			0	0	-	0	0	300	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-	
machine-config-server-22823																																									
Frontend			0	1	-	0	2	3 000	3			1 638	1 048 884	0	0	0	0	0	0	0	0	0	0																		
bootstrap			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m54s	DOWN	L4CON in 0ms	1/1	-	Y	1	1	25m54s	-								
cp1			0	0	-	1	0	1	-	1	1	9m19s	546	349 628	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-
cp2			0	0	-	1	0	1	-	1	1	8m29s	546	349 628	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-
cp3			0	0	-	1	0	1	-	1	1	6m39s	546	349 628	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-
Backend			0	0	-	0	1	300	3	3	3	6m39s	1 638	1 048 884	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0s	-	
ingress-router-80																																									
Frontend			0	0	-	0	0	3 000	187			94 915	705 648	0	0	0	0	0	0	0	0	0	0																		
aw1			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m54s	DOWN	L4CON in 0ms	1/1	Y	-	1	1	25m54s	-								
aw2			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m54s	DOWN	L4CON in 0ms	1/1	Y	-	1	1	25m54s	-								
aw3			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m54s	DOWN	L4CON in 0ms	1/1	Y	-	1	1	25m54s	-								
iw1			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	18m13s	UP	Layer4 connection problem. Connection refused	1	1	7m41s	-											
iw2			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	18m15s	UP	L4OK in 0ms	1/1	Y	-	1	1	7m39s	-								
iw3			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m54s	DOWN	L4CON in 0ms	1/1	Y	-	1	1	25m54s	-								
Backend			0	0	-	0	0	300	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	18m15s	UP		2/2	2	0	0	0	0	0	0	0	0	0	0s	-		
ingress-router-443																																									
Frontend			0	6	-	2	5	3 000	187			94 915	705 648	0	0	0	0	0	0	0	0	0	0																		
aw1			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m54s	DOWN	L4CON in 0ms	1/1	Y	-	1	1	25m54s	-								
aw2			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m54s	DOWN	L4CON in 0ms	1/1	Y	-	1	1	25m54s	-								
aw3			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m54s	DOWN	L4CON in 0ms	1/1	Y	-	1	1	25m54s	-								
iw1			0	0	-	5	1	3	-	128	128	10s	66 687	374 890	0	0	0	0	0	0	0	0	0	18m12s	UP	L4OK in 0ms	1/1	Y	-	1	1	7m42s	-								
iw2			0	0	-	3	1	2	-	53	53	19s	28 228	330 758	0	0	0	0	0	0	0	0	0	18m14s	UP	L4OK in 0ms	1/1	Y	-	1	1	7m40s	-								
iw3			0	0	-	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	25m54s	DOWN	L4CON in 0ms	1/1	Y	-	1	1	25m54s	-								
Backend			0	0	-	0	6	2	4	300	187	181	10s	94 915	705 648	0	0	0	6	0	0	0	0	18m14s	UP		2/2	2	0	0	0	0	0	0	0	0	0	0	0	0s	-

Figure 5-8 HAProxy statistics display second page

You will want to verify you can run `oc` commands successfully but before executing the `oc whoami` command. Ensure the `KUBECONFIG` environment variable is set and the `PATH` environment variable includes a directory with the `oc` command. The expected results from the `oc whoami` command should identify your user name, as shown in Example 5-63.

Example 5-63 Output of whoami command

```
[root@bastion ocp412]# oc whoami
system:admin
```

Validate your control plane nodes are in a ready state by executing the `oc get nodes` command, as shown in Example 5-64.

Example 5-64 Output of oc get nodes

```
[root@bastion ocp412]# oc get nodes
NAME                STATUS    ROLES                AGE   VERSION
cp1.ocp3.ibm.com    Ready    control-plane,master 46m   v1.25.8+37a9a08
cp2.ocp3.ibm.com    Ready    control-plane,master 46m   v1.25.8+37a9a08
cp3.ocp3.ibm.com    Ready    control-plane,master 46m   v1.25.8+37a9a08
```

Check the certificate signing requests:

```
[root@bastion ocp412]# oc get csr
NAME                AGE   SIGNERNAME
REQUESTOR
REQUESTEDDURATION  CONDITION
csr-d4kmf          7m57s
kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
<none>             Pending
csr-d9dvw          46m
kubernetes.io/kubelet-serving      system:node:cp3.ocp3.ibm.com
<none>             Approved,Issued
csr-gtbhb          38m
kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
<none>             Pending
csr-jqddf          47m
kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
<none>             Approved,Issued
csr-kzvpd          38m
kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
<none>             Pending
csr-lfmzc          38m
kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
<none>             Pending
csr-mpkgx          46m
kubernetes.io/kubelet-serving      system:node:cp1.ocp3.ibm.com
<none>             Approved,Issued
csr-r2msq          47m
kubernetes.io/kube-apiserver-client-kubelet
```

```

system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
<none>           Approved,Issued
csr-r6l5m                7m56s
kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
<none>           Pending
csr-s2x5n                46m
kubernetes.io/kubelet-serving      system:node:cp2.ocp3.ibm.com
<none>           Approved,Issued
csr-spwdm                7m48s
kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
<none>           Pending
csr-zgdvn                47m
kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
<none>           Approved,Issued
system:openshift:openshift-authenticator-87wph  44m
kubernetes.io/kube-apiserver-client
system:serviceaccount:openshift-authentication-operator:authentication-operator
<none>           Approved,Issued
system:openshift:openshift-monitoring-mbqpb  43m
kubernetes.io/kube-apiserver-client
system:serviceaccount:openshift-monitoring:cluster-monitoring-operator
<none>           Approved,Issued
[root@bastion ocp412]#

```

Approve pending CSR requests. All requests can be signed. In our example, we show all outstanding requests approved.

```

[root@bastion ocp412]# oc get csr -o go-template='{{range .items}}{{if not
.status}}{{.metadata.name}}{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc
adm certificate approve
certificatesigningrequest.certificates.k8s.io/csr-d4kmf approved
certificatesigningrequest.certificates.k8s.io/csr-gtbhb approved
certificatesigningrequest.certificates.k8s.io/csr-kzvpd approved
certificatesigningrequest.certificates.k8s.io/csr-lfmzc approved
certificatesigningrequest.certificates.k8s.io/csr-r6l5m approved
certificatesigningrequest.certificates.k8s.io/csr-spwdm approved
[root@bastion ocp412]#

```

You may need to repeat the process of approving certificate requests multiple times. When all of your certificate signing requests (CSRs) requests have been approved, **oc get nodes** should show all of your nodes in a ready state.

```

[root@bastion ocp412]# oc get nodes
NAME                STATUS  ROLES                AGE   VERSION
aw1.ocp3.ibm.com    Ready   worker               69s   v1.25.8+37a9a08
aw2.ocp3.ibm.com    Ready   worker               68s   v1.25.8+37a9a08
aw3.ocp3.ibm.com    Ready   worker               72s   v1.25.8+37a9a08
cp1.ocp3.ibm.com    Ready   control-plane,master 63m   v1.25.8+37a9a08
cp2.ocp3.ibm.com    Ready   control-plane,master 63m   v1.25.8+37a9a08
cp3.ocp3.ibm.com    Ready   control-plane,master 63m   v1.25.8+37a9a08
iw1.ocp3.ibm.com    Ready   worker               15m   v1.25.8+37a9a08
iw2.ocp3.ibm.com    Ready   worker               15m   v1.25.8+37a9a08
iw3.ocp3.ibm.com    Ready   worker               15m   v1.25.8+37a9a08

```

The next step is to validate all of the cluster operators are available and not degraded or progressing. The process for them all to become available can vary but it may take 15 - 20 minutes.

```
[root@bastion ocp412]# oc get co
NAME                                VERSION  AVAILABLE  PROGRESSING
DEGRADED  SINCE  MESSAGE
authentication                        4.12.15  True       False
False    7m8s
baremetal                             4.12.15  True       False
False    60m
cloud-controller-manager              4.12.15  True       False
False    64m
cloud-credential                     4.12.15  True       False
False    174m
cluster-autoscaler                   4.12.15  True       False
False    61m
config-operator                       4.12.15  True       False
False    61m
console                              4.12.15  True       False
False    12m
control-plane-machine-set             4.12.15  True       False
False    60m
csi-snapshot-controller               4.12.15  True       False
False    61m
dns                                   4.12.15  True       False
False    60m
etcd                                  4.12.15  True       False
False    59m
image-registry                        4.12.15  True       False
False    53m
ingress                               4.12.15  True       False
False    15m
insights                              4.12.15  True       False
False    54m
kube-apiserver                       4.12.15  True       False
False    51m
kube-controller-manager               4.12.15  True       False
False    58m
kube-scheduler                       4.12.15  True       False
False    57m
kube-storage-version-migrator         4.12.15  True       False
False    61m
machine-api                           4.12.15  True       False
False    60m
machine-approver                     4.12.15  True       False
False    60m
machine-config                       4.12.15  True       False
False    60m
marketplace                           4.12.15  True       False
False    60m
monitoring                            4.12.15  True       False
False    13m
network                               4.12.15  True       False
False    61m
```

```

node-tuning                                4.12.15  True    False
False      57s
openshift-apiserver                        4.12.15  True    False
False      51m
openshift-controller-manager              4.12.15  True    False
False      57m
openshift-samples                         4.12.15  True    False
False      54m
operator-lifecycle-manager                 4.12.15  True    False
False      61m
operator-lifecycle-manager-catalog        4.12.15  True    False
False      61m
operator-lifecycle-manager-packageserver  4.12.15  True    False
False      54m
service-ca                                4.12.15  True    False
False      61m
storage                                    4.12.15  True    False
False      61m
[root@bastion ocp412]#

```

When all of the cluster operators are Available, run the **wait-for-install-complete** command. Below is the example output from our cluster's **wait-for-install-complete**.

```

[root@bastion ocp412]# ./openshift-install --dir /root/ocp412/install-config
wait-for install-complete
INFO Waiting up to 40m0s (until 11:01AM) for the cluster at
https://api.ocp3.ibm.com:6443 to initialize...
INFO Checking to see if there is a route at openshift-console/console...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run
'export KUBECONFIG=/root/ocp412/install-config/auth/kubeconfig'
INFO Access the OpenShift web-console here:
https://console-openshift-console.apps.ocp3.ibm.com
INFO Login to the console with user: "kubeadmin", and password:
"zsQND-JE8q3-Eq722-QxSYF"
INFO Time elapsed: 0s
[root@bastion ocp412]#

```

Next we validate access from the web UI, based on the URL and credentials supplied above. Using the UI does require DNS entries or host file entries to be in place and used by the system running the web browser to access the console application. Figure 5-9 shows the Web UI login window.

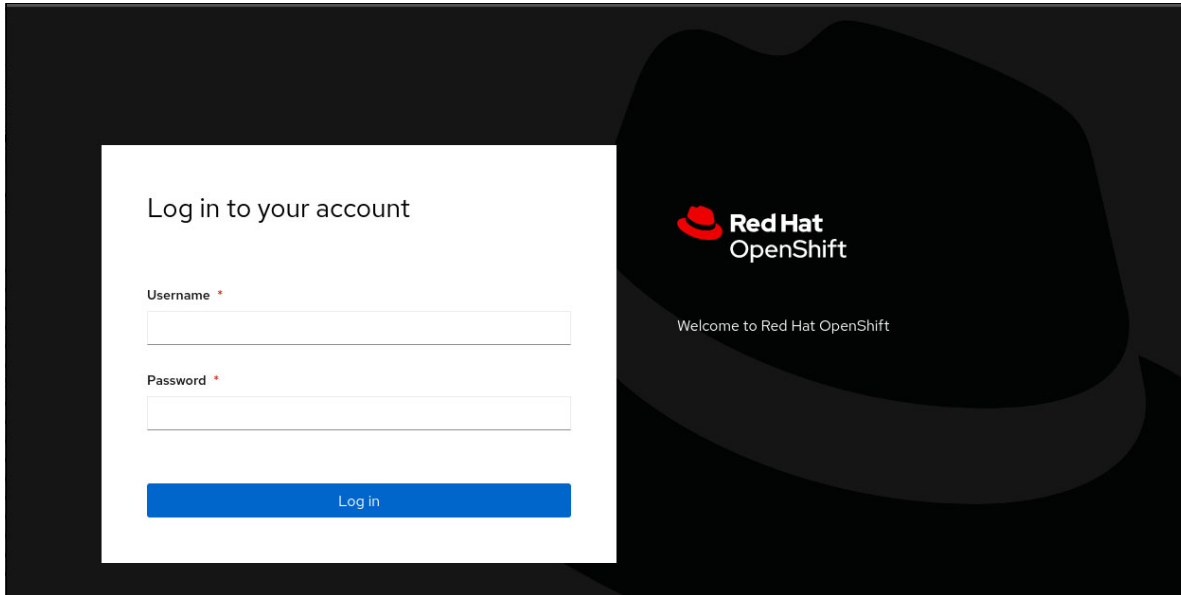


Figure 5-9 Red Hat OpenShift Container Platform initial login window

To complete the login process, supply the credential given in the **wait-for-bootstrap-complete** output. Figure 5-10 shows the initial Overview screen of our Red Hat OpenShift cluster.

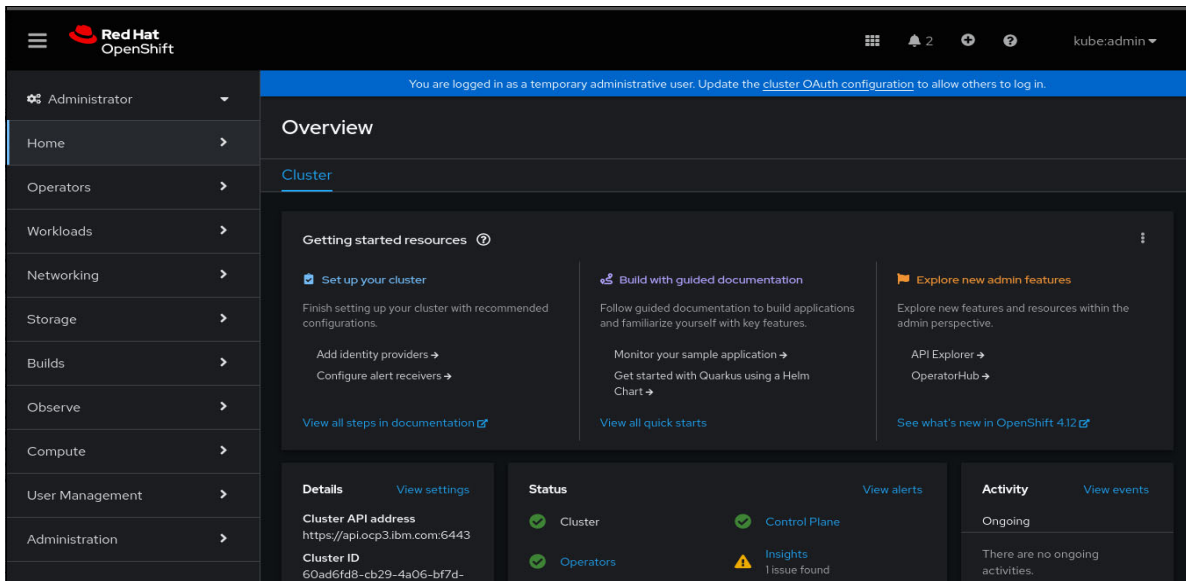


Figure 5-10 Initial Red Hat OpenShift console initial screen after login

Your cluster is now ready to be tailored to your specific sites needs.



Best practices and moving forward

In this chapter, we discuss what activities should be considered after Red Hat OpenShift Container Platform is installed.

6.1 Applying best practices

This section describes some of the recommended practices for running Red Hat OpenShift Container Platform on LinuxONE.

For more information, see [Recommended host practices for IBM Z & IBM® LinuxONE environments](#).

6.1.1 CPU entitlement and vCPU number

CPU entitlement and vCPU number should be examined first to check if the performance is reported as an issue in the LinuxONE environment.

See 3.5.1, “LPAR level controls in IBM Z and LinuxONE” on page 31, 3.5.2, “Guest controls in KVM” on page 31 and 3.5.3, “Guest controls in IBM z/VM” on page 32 to adjust the LPAR CPU weight for the LinuxONE machine. The general rules are as follows:

- ▶ Avoid using a larger IFL ratio than 1:5 (physical to vCPU). Instead, share the IFLs and dedicate fewer vCPUs.
- ▶ Reduce the vCPU count as close as possible to the actual CPU consumption of the guest.
- ▶ Avoid dedicating IFLs to a single LPAR unless there is a specific business requirement.

6.1.2 Disabling transparent huge pages

Transparent Huge Pages (THPs) attempt to handle huge page allocation automatically, which might impact the performance for some workloads. Consider disabling Transparent Huge Pages. Perform the following steps to accomplish this task:

1. Copy the tuned sample shown in Example 6-1 into a YAML file. For example, `thp-s390-tuned.yaml`.

Example 6-1 YAML file

```
apiVersion: tuned.openshift.io/v1
kind: Tuned
metadata:
  name: thp-workers-profile
  namespace: openshift-cluster-node-tuning-operator
spec:
  profile:
    - data: |
        [main]
        summary=Custom tuned profile for OpenShift on IBM Z to turn off THP on
worker nodes
        include=openshift-node
        [vm]
        transparent_hugepages=never
        name: openshift-thp-never-worker
  recommend:
    - match:
      - label: node-role.kubernetes.io/worker
        priority: 35
        profile: openshift-thp-never-worker
```

2. Create the resource definition by using the following command:


```
$ oc apply -f thp-s390-tuned.yaml
```
3. Verify that the status of the huge pages with the following command:


```
$ ssh core@workernode1 sudo cat /sys/kernel/mm/transparent_hugepage/enabled
```

6.1.3 Enabling RFS

Enabling Receive Flow Steering (RFS) can generally help to reduce network latency and improve network performance for application workloads.

To verify if RFS is enabled on the platform, issue the following commands:

```
$ ssh core@workernode1 sudo cat /proc/sys/net/core/rps_sock_flow_entries
$ ssh core@workernode1 sudo cat /sys/class/net/enc1000/queues/rx-0/rps_flow_cnt
```

For more information, see [Recommended host practices for IBM Z & IBM® LinuxONE environments](#).

6.1.4 Infrastructure nodes

The concept of an infrastructure node, or infra node, is to enable compute nodes to concentrate on handling application workloads without the responsibility of managing infrastructure services like monitoring, ingress routers, and logging. By doing so, the infrastructure node can optimize resources and improve overall system efficiency.

To move infrastructure services from compute nodes to infra nodes, perform the following steps:

1. Add the following label to the compute nodes that you want to act as an infrastructure node.


```
$ oc label node <node-name> node-role.kubernetes.io/infra=""
```
2. Move services to the infrastructure nodes. For more information, see [Moving Resources to infrastructure machine sets](#).

6.1.5 HyperPAV

This practice only applies to the environment that uses ECKD volumes. ECKD volumes have different model numbers such as mod-9, mod-27, mode-54, and so on. Usually if you are using mod-54 volumes and above, the HyperPAV feature is recommended to increase I/O access paths, and achieve better I/O performance. This would be helpful in I/O intensive environments, for example, if you are running ODF storage nodes.

1. Update the USER DIRECT, as shown in Example 6-2.

Example 6-2 USER DIRECT

```
USER ODFNODE1 LNX4VM 24G 256G G
INCLUDE LNXPDFLT
COMMAND DEFINE HYPERPAV A000 FOR BASE 120
COMMAND DEFINE HYPERPAV A001 FOR BASE 120
COMMAND DEFINE HYPERPAV A002 FOR BASE 120
COMMAND DEFINE HYPERPAV A003 FOR BASE 120
MDISK 120 3390 DEVNO 9120 MR
```

2. Copy the HyperPAV sample shown in Example 6-3 into a YAML file named, for example, 05-worker-kernelarg-hpav.yaml.

Example 6-3 HyperPAV sample file

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 05-worker-kernelarg-hpav
spec:
  config:
    ignition:
      version: 3.1.0
  kernelArguments:
    - rd.dasd=120,A000-A003
```

3. Create the resource definition by using the following command.

```
$ oc apply -f 05-worker-kernelarg-hpav.yaml
```

6.1.6 Specific for KVM

In our lab environment, we applied the following recommended configurations during the deployment phase:

- ▶ We utilize HyperPAV for our storage setup because we were previously employing ECKD volumes.
- ▶ We disable the use of transparent huge pages to enhance system performance.
- ▶ We configure multiple queues for our VirtIO network interfaces to improve network throughput and reduce latency.
- ▶ We employ I/O threads for our virtual block devices to accelerate I/O operations and enhance overall performance.
- ▶ We eschew the use of virtual SCSI devices in favor of native storage solutions.
- ▶ We enable guest caching for disk to optimize disk I/O operations and improve system responsiveness.

See sections 5.4, “KVM single hypervisor cluster implementation” on page 75 and 5.5, “KVM three-LPAR cluster implementation” on page 95 for more details.

For more information, see [Recommended host practices for IBM Z & IBM® LinuxONE environments](#).

6.2 Post-installation configurations

This section describes some common post-installation configurations that should be done before trying to deploy an application onto the cluster.

6.2.1 Defining an identity provider

By default, only the kubeadmin user id is able to log onto the Red Hat OpenShift Container Platform. To add an identity provider, create a custom resource (CR) and then add it to the cluster.

We mention two commonly used types of identity providers. For more information, see [Understanding identity provider configuration](#).

htpasswd

Configure the htpasswd identity provider to allow users to log in to the Red Hat OpenShift Container Platform with credentials from an htpasswd file. To do this, perform the following steps:

1. Create an htpasswd file to store the user id and password information by using the following command:

```
htpasswd -c -B -b users.htpasswd admin001 passw0rd
```

To add an additional user id to the htpasswd file, use the following command:

```
htpasswd -B -b users.htpasswd app001 passw0rd
```

2. To use the htpasswd identity provider, you must define a secret object that contains the htpasswd user file. To create a secret object, issue the following command:

```
oc create secret generic users-secret --from-file=htpasswd=users.htpasswd -n openshift-config
```

3. Create a CR for the identity provider by applying a YAML file. We create the YAML file users-cr.yaml shown in Example 6-4.

Example 6-4 Custom resource YAML file

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: RDBK_HTPASSWD
    mappingMethod: claim
    type: HTPasswd
    htpasswd:
      fileData:
        name: users-secret
```

4. We then apply the defined custom resource YAML file using the following command:

```
oc apply -f users-cr.yaml
```

5. Create the user with the following command:

```
oc create user admin001
```

6. Create the Red Hat OpenShift Container Platform identity by issuing the following command.

```
oc create identity RDBK_HTPASSWD:admin001
```

RDBK_HTPASSWD is the identity-provider you created when you applied the custom resource file from Example 6-4 on page 141 for the htpasswd file.

7. Bind the cluster-admin to the specific user that you created using the following command:

```
oc adm policy add-cluster-role-to-user cluster-admin admin001
```

8. Create the mapping for the user and identity:

```
oc create useridentitymapping RDBK_HTPASSWD:admin001 admin001
```

LDAP

For details on creating the LDAP identity provider, see [Configuring an LDAP identity provider](#).

6.2.2 Configuring persistent storage

In this section, we describe the steps to configure an NFS server as persistent storage.

1. Create a directory called `nfs-client` that contains each of the YAML files shown in this section for the `nfs-client-provider`. The YAML files that should be included are:

Example 6-5 deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nfs-client-provisioner
  labels:
    app: nfs-client-provisioner
  namespace: nfs-client-provisioner
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: nfs-client-provisioner
  template:
    metadata:
      labels:
        app: nfs-client-provisioner
    spec:
      serviceAccountName: nfs-client-provisioner
      containers:
        - name: nfs-client-provisioner
          image:
gcr.io/k8s-staging-sig-storage/nfs-subdir-external-provisioner:s390x-linux-cana
ry
          #image: quay.io/external_storage/nfs-client-provisioner:latest
          volumeMounts:
            - name: nfs-client-root
              mountPath: /persistentvolumes
          env:
            - name: PROVISIONER_NAME
              value: nfs-client-provisioner/nfs
            - name: NFS_SERVER
              value: {{ nfs_server }}
            - name: NFS_PATH
              value: /home/data/nfs_share
      volumes:
        - name: nfs-client-root
          nfs:
            server: {{ nfs_server }}
            path: /home/data/nfs_share
```

Example 6-6 sc.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: managed-nfs-storage
provisioner: nfs-client-provisioner/nfs # or choose another name, must match
deployment's env PROVISIONER_NAME'
parameters:
  archiveOnDelete: "false"
```

Example 6-7 rbac.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: nfs-client-provisioner
  namespace: nfs-client-provisioner
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: nfs-client-provisioner-runner
rules:
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["create", "update", "patch"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: run-nfs-client-provisioner
subjects:
  - kind: ServiceAccount
    name: nfs-client-provisioner
    namespace: nfs-client-provisioner
roleRef:
  kind: ClusterRole
  name: nfs-client-provisioner-runner
  apiGroup: rbac.authorization.k8s.io
---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: leader-locking-nfs-client-provisioner
  namespace: nfs-client-provisioner
```

```

rules:
  - apiGroups: [""]
    resources: ["endpoints"]
    verbs: ["get", "list", "watch", "create", "update", "patch"]
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: leader-locking-nfs-client-provisioner
  namespace: nfs-client-provisioner
subjects:
  - kind: ServiceAccount
    name: nfs-client-provisioner
    # replace with namespace where provisioner is deployed
    namespace: nfs-client-provisioner
roleRef:
  kind: Role
  name: leader-locking-nfs-client-provisioner
  apiGroup: rbac.authorization.k8s.io

```

2. Create a namespace with the following command.

```
oc new-project nfs-client-provisioner
```

3. Apply the YAML files that are in the directory you created in Step 1 by using the following command. In our example here, we created a directory named `nfs-client`.

```
oc apply -f nfs-client
```

4. Grant authority with the following command:

```
oc adm policy add-scc-to-user hostmount-anyuid
system:serviceaccount:nfs-client-provisioner:nfs-client-provisioner
```

6.2.3 Configuring the NTP Server

Time synchronization is always crucial for running a cluster and that would include Red Hat OpenShift Container Platform. You can either configure the NTP server during the installation phase or post-installation.

For more details, see [Installer-provisioned postinstallation configuration](#).

6.2.4 Disabling kubeadmin

User `kubeadmin` is a temporary administrative user. After defining an identity provider and creating a new cluster-admin user, consider removing the `kubeadmin` to improve cluster security.

IMPORTANT: Do not remove the `kubeadmin` user unless you are sure that you have a working cluster-admin user defined.

Delete the `kubeadmin` secrets with the following command:

```
oc delete secrets kubeadmin -n kube-system
```

6.2.5 Backing up the etcd database

The etcd database plays a very important role in a Red Hat OpenShift Cluster. The following are the instructions to backup the etcd database.

1. Use the following command to back up etcd

```
oc debug node/{{controller_node}} -- chroot /host  
/usr/local/bin/cluster-backup.sh /home/core/ocp412/
```

2. Change the file permissions if needed with the following command:

```
oc debug node/{{controller_node}} -- chroot /host chmod -R o+r  
/home/core/ocp412
```

3. To retrieve the backup data, use the following command:

```
scp -rp core@{{controller_node}}:/home/core/ocp412 /root/ocp412_bkp/
```

Important: Back up your cluster's etcd data regularly and store in a secure location, ideally outside of the Red Hat OpenShift Container Platform environment.

For more information, see [Backing up etcd](#).

6.3 Sample application deployment

In our sample application architecture, we deploy an open-source, lightweight, microservices based application called the Voting App. It is cross-platform and can be deployed on any architecture. We use this application as an example to demonstrate how to deploy applications on a Red Hat OpenShift Container Platform, which runs on IBM LinuxONE.

For more information, see the [GitHub repository](#).

6.3.1 Application architecture

Our application architecture is shown in Figure 6-1.

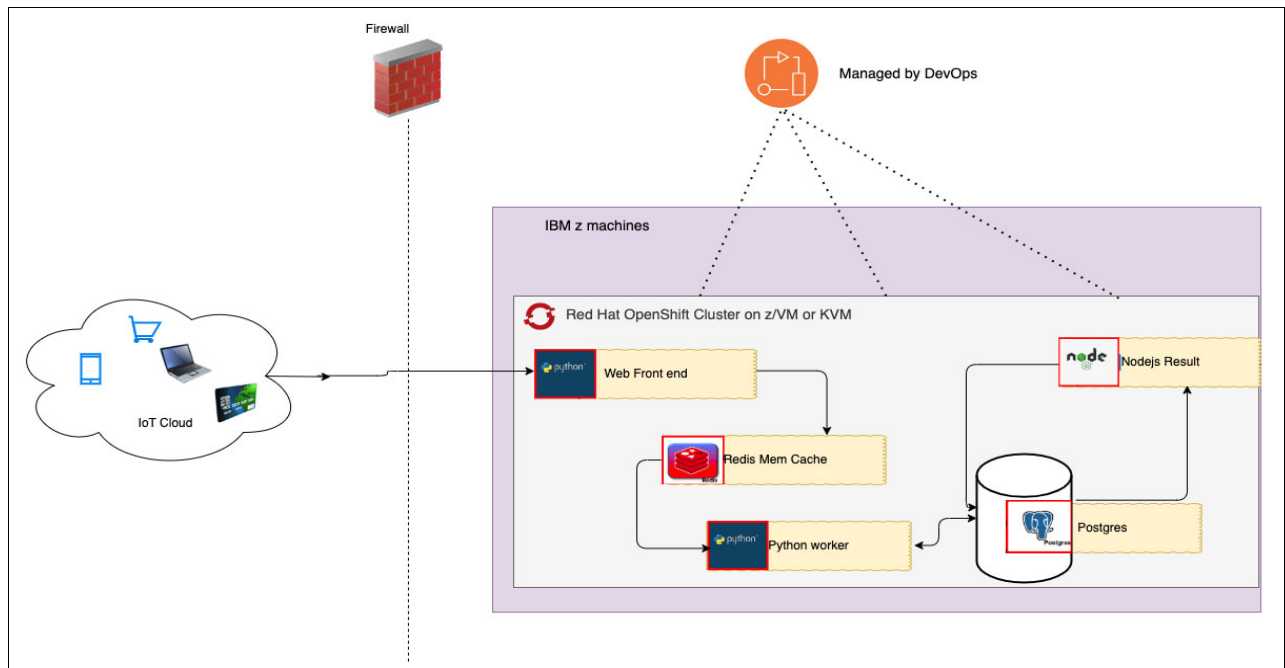


Figure 6-1 Sample Voting application architecture

The application provides the user with a choice to vote for any of two given options, such as coffee versus tea.

On Red Hat OpenShift Cluster on LinuxONE

In a Red Hat OpenShift Cluster on LinuxONE environment, the application architecture consists of several microservices working together. The Python web front end allows users to interact with the application and view available options. When users make selections, their choices are sent to the Redis microservice for temporary storage.

Redis acts as an in-memory cache, temporarily storing the received votes until they are processed by another Python microservice. This microservice then persists the votes in a PostgreSQL database for long-term storage.

A final Node.js microservice displays the current voting results as they accumulate in the PostgreSQL database. The microservices-based design enables individual scaling of each component, allowing for seamless replacement of components with alternative solutions without affecting the overall application.

6.3.2 Deployment

In this section, we discuss our deployment of the Voting application.

Environment

Our environment consisted of a Red Hat OpenShift Cluster on IBM LinuxONE. To deploy the application, get the GitHub personal access token that will be used for generating a secret.

Procedures

We performed the following steps to start the deployment.

1. Set the environment variables \$PROJECT, \$GIT_REPO, and \$GIT_TOKEN by using the following commands.

```
PROJECT=voting
GIT_REPO=https://github.com/liyong-li/voting-app-rdbk.git
GIT_TOKEN=/root/git_token
```

2. Create the new project by using the following command:

```
oc new-project ${PROJECT}
```

3. Create the secret by using the following command:

```
oc create secret generic git-token --from-file=password=${GIT_TOKEN}
--type=kubernetes.io/basic-auth
```

4. Import images with the following commands:

```
oc import-image rhel8/nodejs-12 --from=registry.redhat.io/rhel8/nodejs-12
--confirm
oc import-image ubi8/python-38 --from=registry.redhat.io/ubi8/python-38
--confirm
```

5. Deploy the Postgres database service with the following commands. Figure 6-2 provides our results of these commands:

```
oc new-app --name new-postgresql --template=postgresql-persistent \
--param=DATABASE_SERVICE_NAME=new-postgresql \
--param=POSTGRESQL_USER=admin \
--param=POSTGRESQL_DATABASE=db \
--param=POSTGRESQL_PASSWORD=admin \
--param=POSTGRESQL_VERSION=latest
```





Name ↑	Status ↓	Ready ↓	Restarts ↓	Owner ↓	Memory ↓	CPU ↓	Created ↓
 new-postgresql-1-pjrvj	 Running	1/1	0	 new-postgresql-1	31.2 MiB	0.007 cores	 11 May 2023, 14:27

Figure 6-2 Sample output for Postgres Pod

6. Deploy the Redis service with the following commands. Figure 6-3 provides our results of these commands.

```
oc new-app --name new-redis --template=redis-persistent \
--param=DATABASE_SERVICE_NAME=new-redis \
--param=REDIS_PASSWORD=admin \
--param=REDIS_VERSION=latest
```

Name ↑	Status ↓	Ready ↓	Restarts ↓	Owner ↓	Memory ↓	CPU ↓	Created ↓
new-redis-1-7f125	Running	1/1	0	new-redis-1	9.1 MiB	0.003 cores	11 May 2023, 14:27

Figure 6-3 Sample output for Redis Pod

7. Deploy the Python compute by using the following commands. Figure 6-4 provides our results of these commands.

```
oc new-app python-38:latest~${GIT_REPO} \
--source-secret=git-token \
--context-dir=worker-python \
--name=voting-app-worker-py \
-e DB_NAME="db" \
-e DB_USER="admin" \
-e DB_PASS="admin" \
-e REDIS_PASSWORD="admin"
```

Name ↑	Status ↓	Ready ↓	Restarts ↓	Owner ↓	Memory ↓	CPU ↓	Created ↓
voting-app-worker-py-1-build	Completed	0/1	0	voting-app-worker-py-1	-	-	11 May 2023, 14:27

Figure 6-4 Sample output for Python compute Pod

8. Deploy the voting application front-end with the following commands. Figure 6-5 provides our results of these commands.

```
oc new-app python-38~${GIT_REPO} \
--source-secret=git-token \
--context-dir=/vote \
--name=voting-app-py \
-e REDIS_PASSWORD="admin"
```

Name ↑	Status ↓	Ready ↓	Restarts ↓	Owner ↓	Memory ↓	CPU ↓	Created ↓
voting-app-py-696fd9c6f4	Running	1/1	0	voting-app-py-696fd9c6f4	43.8 MiB	0.002 cores	11 May 2023, 14:28

Figure 6-5 Sample output for Voting App frontend pod

9. Create the voting application front-end route by using the following commands. Figure 6-6 provides our results of these commands.

```
oc create route edge demo-py --service=voting-app-py --port=8080
```

Name	Status	Location	Service
demo-py	Accepted	https://demo-py-voting.apps.zvm2.rdbk.com	voting-app-py

Figure 6-6 Sample output for Voting App front-end route

10. Deploy the Node.js result front-end application by using the following commands. Figure 6-7 provides our results of these commands.

```
oc new-app nodejs-12:latest~${GIT_REPO} \
--source-secret=git-token \
--context-dir=result \
--name=voting-app-nodejs \
-e POSTGRES_CONNECT_STRING="postgres://admin:admin@new-postgresq1/db"
```

Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
voting-app-nodejs-dcff6dbc6-jbftb	Running	1/1	0	voting-app-nodejs-dcff6dbc6	32.1 MiB	0.001 cores	11 May 2023, 14:28

Figure 6-7 Sample output for Node.js result front-end pod

11. Create the Node.js result front-end route by using the following commands. Figure 6-8 provides our results of these commands.

```
oc create route edge demo-nodejs --service=voting-app-nodejs --port=8080
```

Name	Status	Location	Service
demo-nodejs	Accepted	https://demo-nodejs-voting.apps.zvm2.rdbk.com	voting-app-nodejs

Figure 6-8 Sample output for Node.js result front-end route

Accessing the application through a web browser

In our lab environment, we accessed the Voting App through a web browser and URL: <https://demo-py-voting.apps.zvm2.rdbk.com>. The resulting web page is shown in Figure 6-9.

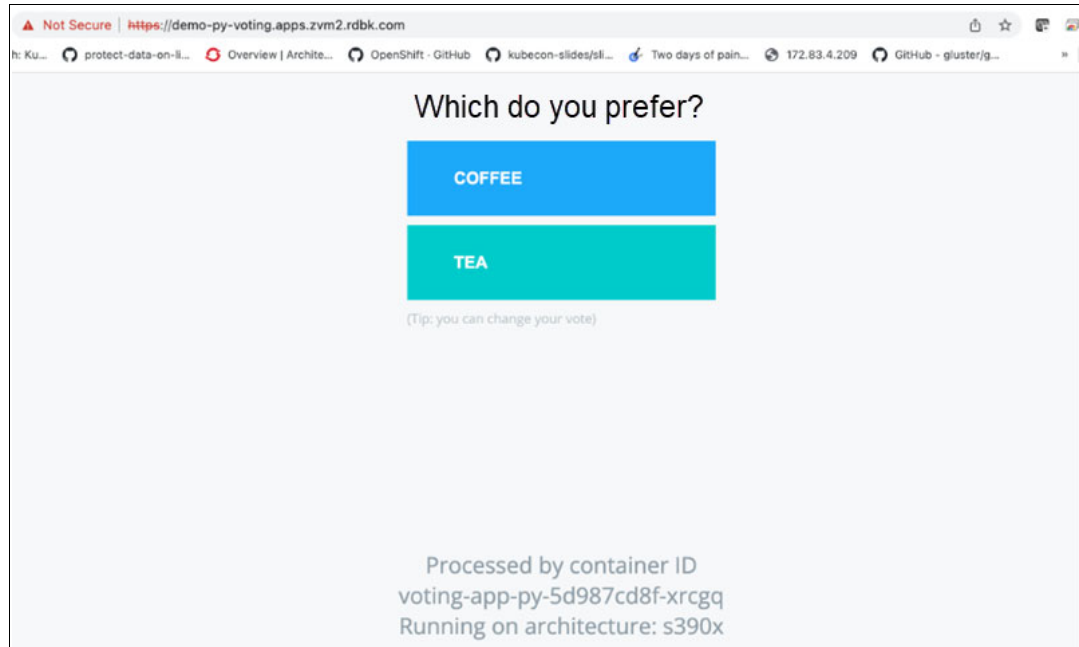


Figure 6-9 Sample Voting App frontend

The Voting application results were shown using the URL: <https://demo-nodejs-voting.apps.zvm2.rdbk.com>. Our results are shown in Figure 6-10.

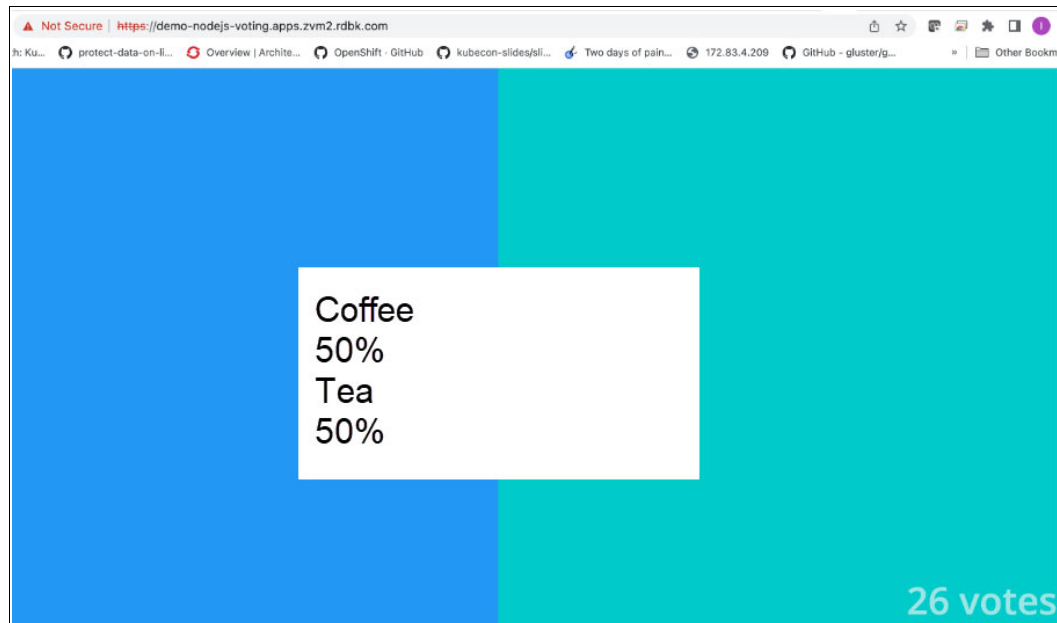


Figure 6-10 Sample output for Voting App Result



Additional material

This paper refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material associated with this paper is available at [REDP5711-Red-Hat-OpenShift-on-IBM-zSystems-and-LinuxONE](#).

The Voting application used in this IBM Redbooks publication can be found at the [GitHub repository](#).

Using the web material

The additional web material that accompanies this paper includes the following files:

- ▶ `chapter_5_ansible_on_linuxone_rdbk-main.tar.gz`
- ▶ `chapter_6_voting-app-rdbk-main.tar.gz`

System requirements for downloading the web material

The web material requires the following system configuration:

Hard disk space: 253 MB
Operating System: Windows/Linux

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation and extract the contents of the folder into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

The following IBM Redbooks publication provides additional information about the topic in this document. Note that this publication might be available in softcopy only.

- ▶ *The Virtualization Cookbook for IBM Z Volume 2: Red Hat Enterprise Linux 8.2*, SG24-8303

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:

- ▶ Install Red Hat OpenShift Container Platform 4 on IBM zSystems and IBM LinuxONE
<https://developer.ibm.com/learningpaths/get-started-ibmz/red-hat-openshift-on-ibmz/install-openshift-container-platform-on-ibmz/>
- ▶ Red Hat OpenShift for IBM zSystems and IBM LinuxONE
<https://www.redhat.com/en/resources/openshift-ibm-z-linuxone-datasheet>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



REDP-5711-00

ISBN 0738461512

Printed in U.S.A.

Get connected

