Redbooks

ibm.com/redbooks

Creating OpenShift Multiple Architecture Clusters with IBM Power

Tim Simon

Sambasiva Andaluri

Ivaylo Bozhinov

Carlos Jorge Cabanas Aguero

Carlo Castillo

Paul Chapman

Mandar Dixit

Pete Dragovich

Gayathri Gopalakrishnan

Vikash Gupta

Munshi Hafizul Haque

Youssef Largou

Mikhail Nikitin

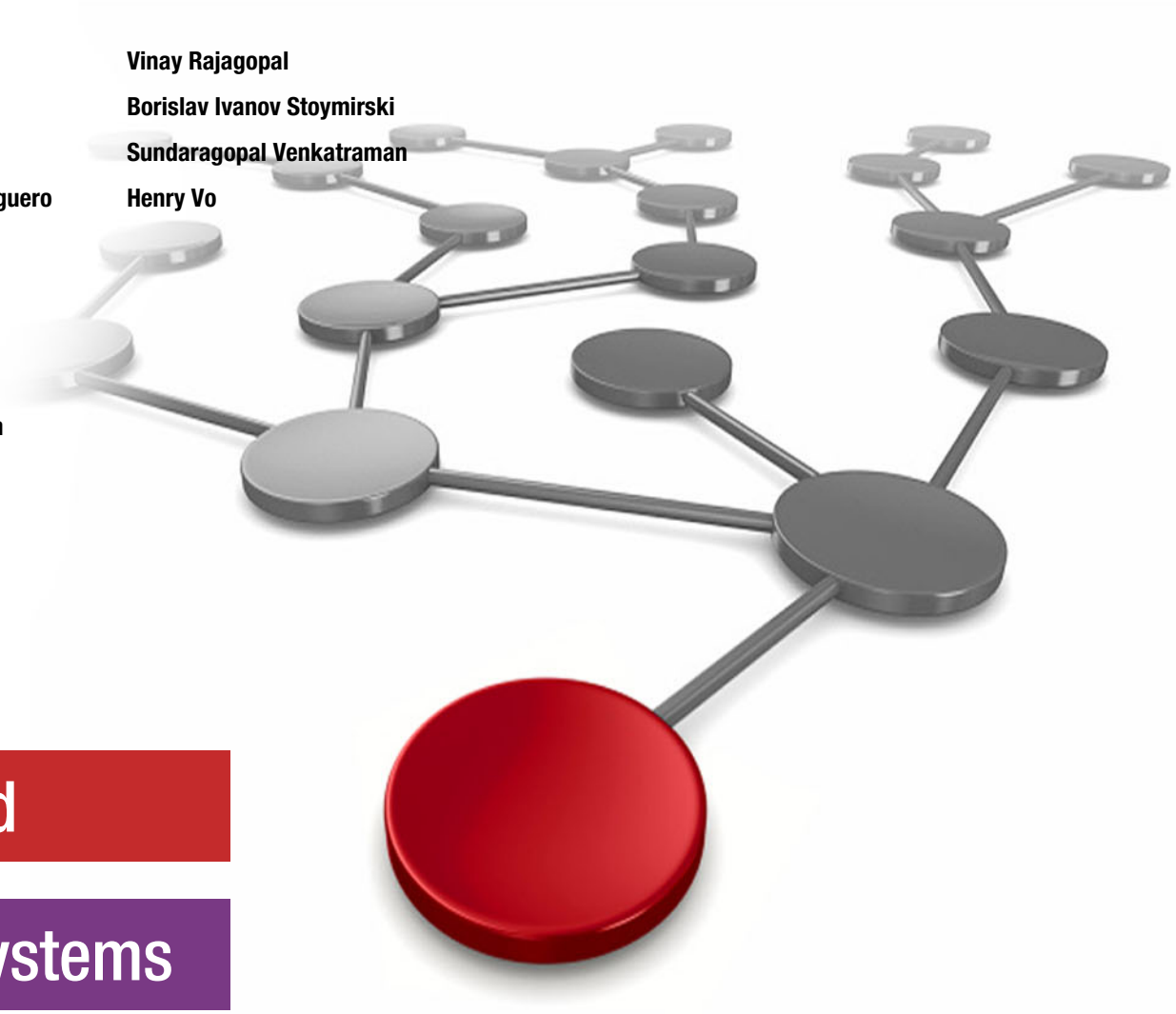
Gabriel Padilla

Vinay Rajagopal

Borislav Ivanov Stoymirski

Sundaragopal Venkatraman

Henry Vo



Cloud

Power Systems

IBM

Redbooks

IBM Redbooks

Creating OpenShift Multiple Architecture Clusters with IBM Power

November 2024

SG24-8565-00

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

First Edition (November 2024)

This edition applies to:
Red Hat OpenShift Version 4.14
Red Hat OpenShift Version 4.15
Red Hat Advanced Cluster Management Version 2.10

Contents

Notices

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

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

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

AIX®
Cognos®
Db2®
IBM®
IBM Cloud®
IBM Cloud Pak®
IBM FlashSystem®
IBM Instana™
IBM Spectrum®
IBM Watson®
IBM Z®
Instana®
POWER®
Power Architecture®
PowerVM®
Rational®
Redbooks®
Redbooks (logo)  ®
System z®
Think®
Turbonomic®
WebSphere®

The following terms are trademarks of other companies:

Intel, Intel Xeon, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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.

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

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

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

Preface

The release of Red Hat OpenShift 4.14 brought the OpenShift Container Platform Multiple-Architecture Compute feature to IBM Power. Multi-Arch Compute provides a single heterogeneous cluster, enabling fit-for-purpose computing so clients can align tasks and applications to CPU strengths and software availability rather than one architecture. This support was expanded in Red Hat OpenShift 4.15 which enabled a Red Hat OpenShift cluster to support an IBM Power control plane and add x86 architecture worker nodes.

Multi-Arch Compute for OpenShift Container Platform lets you use a pair of compute architectures, such as ppc64le and amd64, within a single cluster. This exciting feature opens new possibilities for versatility and optimization for composite solutions that span multiple architectures.

This Redbook provides a high-level overview of Red Hat OpenShift which can be run on many different architectures. We then present how IBM Power servers provide an extremely resilient and secure platform and how they can be an excellent platform for your cloud implementation using OpenShift. We then describe the benefits of using a multi-architecture cluster and provide implementation guidelines and advice to assist the reader in implementing a multi-architecture cluster using IBM Power control plane nodes and x86 or AMD-based worker nodes.

This Redbook is suitable for many levels including managers that need to understand how OpenShift on Power can help meet their business needs as well as supporting technical staff that are responsible for implementing those clusters.

Authors

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

Tim Simon is an IBM® Redbooks® Project Leader in Tulsa, Oklahoma, USA. He has over 40 years of experience with IBM – primarily in a technical sales role working with customers to help them create IBM solutions to solve their business problems. He holds a BS degree in Math from Towson University in Maryland. He has worked with many IBM products and has extensive experience creating customer solutions using IBM Power, IBM Storage, and IBM System z® throughout his career.

Sambasiva Andaluri (Sam) is an experienced Developer turned Solution Architect Leader with over 30 years of experience. For the past decade, he has been a pre-sales and post-sales solution architect for trading systems at Fidessa, presales solution architect at AWS and as an SRE and onboarding ISVs for Google marketplace at a partner. He brings multifaceted experience to the table, a continuous learner, and a strong supporter of STEM. In his free time, he coaches K-12 students for FIRST LEGO league competitions and inspires the young minds to take up STEM careers.

Ivaylo Bozhinov is a Technical Support Professional SME (FSP/Hypervisor/eBMC) for the Power System hardware division in Sofia, Bulgaria. He has been with IBM since 2015 and participated in numerous educational and client-related workshops and presentations.

He holds a bachelor's degree in Information Technology from the State University of Library and Information Technology and master's degree in Cybersecurity from New Bulgarian University. He supports a large number of clients from banking industry, telecom, and retail sector.

Carlos Jorge Cabanas Aguero has been a Consultant with IBM Technology Lifecycle Services in Argentina for the last 13 years. Prior to that he worked in various roles in the IT industry supporting specializing in IBM AIX® and other Unix solutions. He has extensive experience supporting IBM AIX and Linux on IBM Power including HA/DR solutions and performance tuning.

Carlo Castillo is a Client Services Manager for Right Computer Systems (RCS), an IBM Business Partner and Red Hat partner in the Philippines. He has over thirty years of experience in pre-sales and post-sales support, designing full IBM infrastructure solutions, creating pre-sales configurations, performing IBM Power installation, implementation and integration services, and providing post-sales services and technical support for customers, as well as conducting presentations at customer engagements and corporate events. He was the very first IBM-certified AIX Technical Support engineer in the Philippines in 1999. As training coordinator during RCS' tenure as an IBM Authorized Training Provider from 2007 to 2014, he also administered the IBM Power Systems curriculum, and conducted IBM training classes covering AIX, PureSystems, IBM PowerVM®, and IBM i. He holds a degree in Computer Data Processing Management from the Polytechnic University of the Philippines.

Paul Chapman is a Global Modernization Technical Leader for IBM Power Technology, based in the UK. With 28 years of technical and management experience working for IBM Business Partners and customers, he works closely with Offering Management and Development Leaders to deliver successful first-of-a-kind projects and early adoption programs. Paul spearheaded the .NET launch on Power, created the OpenShift Multi-Arch Compute Early Adoption Program, and collaborated with the Development and Research Team's Co-Creation project, delivering the first .NET on Power and OpenShift Multi-Arch Compute Public References. He also regularly presents at conferences, shares knowledge and skills using social media, and has co-authored the "Red Hat OpenShift V4.X and IBM Cloud® Pak on IBM Power Systems Volume 2" Redbook. In addition, he has received two Outstanding Technical Achievement Awards.

Mandar Dixit is responsible for Presales across West & North India for Corporate Customers. With 20+ years of Presales & Consulting experience, in varied segments like Cloud, Data Center, Telco & System Integration, he brings an excellent blend of Business & Technical skills. At Red Hat he has made remarkable contributions towards business growth in the hybrid cloud market based on Enterprise open-source technologies. Mandar has a proven record of accomplishment in complex solution building across Industry verticals like BFSI, Manufacturing & ITES and has been a Trusted Advisor to Large Business Conglomerates, in their Application Modernization & Hybrid Cloud journey.

Pete Dragovich is an IBM Power Senior Brand Technical Specialist in the United States. He has 35 years of experience working in a variety of roles in IBM Support, IBM GTS, and IBM Technical Sales. His experience spans not only IBM Power, but IBM Storage, IBM Cloud, AIX, Linux, and many client-facing roles in helping clients solve their most complex problems. His interests lie in helping clients realize the benefits of continuing to run their mission-critical workloads on IBM Power and setting them on the path of modernization. He holds an A.A.S. degree in Electronics Technology.

Gayathri Gopalakrishnan works for IBM India and has over 22 years of experience as a technical solution and IT architect, working primarily in consulting. She is a results-driven IT Architect with extensive working experience in spearheading the management, design, development, implementation, and testing of solutions.

A recognized leader, applying high-impact technical solutions to major business objectives with capabilities transcending boundaries. She is adept at working with management to prioritize activities and achieve defined project objectives with an ability to translate business requirements into technical solutions.

Vikash Gupta is a Solution Architect at Red Hat India, responsible for South & West India Corporate Customers. He has diversified skills on Platform and Application Modernization, Hybrid Cloud and IT Automation. He is highly customer-centric and embodies technical prowess in Cybersecurity, Cloud, AI and Analytics. Vikash has over 18 years of experience working extensively with customers from multiple industry verticals such as BFSI, Energy and Utilities, Manufacturing and Public Sector. He is a trusted advisor for customers in resolving challenges through adoption of emerging technologies and building new business growth. He is an active speaker and contributor at Red Hat Skills Academy, previously at IBM University Relations and author for IBM Redbooks.

Munshi Hafizul Haque is a Senior Platform Consultant at Red Hat in Kuala Lumpur, Malaysia. Munshi is an experienced technologist in engineering, design, and architecture of PaaS and cloud infrastructures. He has over 16 years of experience in post-delivery technical support, designing full infrastructure solutions, performing installation, implementation and integration services of systems, storages and software solutions, and conducting post implementation assessment, presentations at customer engagements and corporate events. He is currently part of the Red Hat Consulting Services team where he helps organizations adopt automation, container technology and DevOps practices. Before that, he worked for IBM as a senior consultant with IBM Systems Lab Services in Petaling Jaya, Malaysia, where he took part in various projects with different people in different ASEAN countries, and as a specialist in IBM Power Systems and associated enterprise edition technology.

Youssef Largou is the founding director of PowerM, a platinum IBM Business Partner in Morocco. He has 22 years of experience in systems, HPC, middleware, and hybrid cloud, including IBM Power, IBM Storage, IBM Spectrum®, IBM WebSphere®, IBM Db2®, IBM Cognos®, IBM WebSphere Portal, IBM MQ, ESB, IBM Cloud Pak®, SAP HANA and Red Hat OpenShift. He has worked within numerous industries with many technologies. Youssef is an IBM Champion 2020, 2021,2022, 2023 and 2024, an IBM Redbooks Platinum Author and has designed many reference architectures. His company has been recognized as an IBM Beacon Award Finalist in Storage, Software-Defined Storage, and LinuxONE five times. He is a regular speaker at IBM Think®, IBM TechXchange and Common Europe Congress. He holds an engineering degree in Computer Science from the Ecole Nationale Supérieure des Mines de Rabat and Executive MBA from EMLyon.

Mikhail Nikitin is an IBM Client Engineering Solution Architect in Canberra, Australia. He is a Linux nerd since 1998 and has more than 20 years of experience in enterprise Linux design, security and implementation. In the past he was an architect, team/practice lead, pre-sales, systems engineer, systems administrator, software developer and nuclear science student. Graduated from the National Research Nuclear University in Moscow. He is also a Red Hat Certified Architect (RHCA) in Infrastructure.

Gabriel Padilla is the Linux Test Architect for IBM Power Systems focused on Hardware Assurance. He has a Bachelor's Degree as an Electronic Engineer and also a master's degree in Information Technology. He has been in IBM for more than 10 years and his experience ranges from Test Development (Design) to Supply Chain process. Gabriel is considered as a technical leader for IBM Systems on subject matters including Linux, Red Hat OpenShift and Cloud.

Vinay Rajagopal is a Principal Technologist & ISV Lead Architect for the Partner Ecosystem at Red Hat. As a Technology lead, he is responsible for helping Independent Software Vendors (ISVs) including IOT, AI/ML and Emerging technologies-based partners to build products on Red Hat stack, accelerate containerization and adoption of cloud-native technologies thus enabling them on go-to-market strategy. He brings over 20+ years of combined application development and platform experience helping GSIs, IHVs, ISVs & MSPs in building offerings and products. He is a BlackBolt certified product management professional, has completed Design Thinking from MIT Emeritus, Business Management from LSE (London School of Economics and Political Science), Leadership and Strategy from JWMI (Jack Welch Management Institute). He holds several granted patents and published files. He is an open-source enthusiast who is currently pursuing a Masters in AI from Reva University.

Borislav Ivanov Stoymirski is an IBM Power Servers Hardware Product Engineer at IBM Bulgaria, specializing in solving complex hardware and software issues on IBM Power Servers, including IBM AIX, VIOS, HMC, IBM i, PowerVC, PowerVM NovaLink, PowerVM, Linux on IBM Power Servers, and Red Hat OpenShift. Since joining IBM in 2015, he has provided reactive break-fix, proactive, preventative, and cognitive support to several clients worldwide. Borislav earned a master's degree in Computer and Software Engineering from the Technical University of Sofia, Bulgaria, a master's degree in Transport Machinery and Technologies from the Technical University of Sofia, Bulgaria, and another Master's degree in Ecology and Environmental Protection from the University of Chemical Technology and Metallurgy, Bulgaria. He has authored several publications and has led technical training programs both inside and outside IBM. His interests lie in Green Blockchain, Artificial Intelligence, Deep Learning, Machine Learning, Cybersecurity, Internet of Things (IoT), Edge Computing, Cloud and Quantum Computing.

Sundaragopal Venkatraman (Sundar) is a Red Hat Industry Specialist. He has diversified skills on Hybrid Cloud Automation, Application Migration, and Modernization of Red Hat and IBM portfolios. Sundar has over 23 years of experience working closely with customers to overcome business challenges by leveraging technologies. A prolific author, he has been recognized as “Platinum Author” for IBM Redbooks publications. He holds multiple patents and is an Invention Plateau holder. He has delivered key notes on WW conferences on Technology Transformation & Modernization. He is a co-chair for the IT specialist board in the Asia-Pacific region.

Henry Vo is an IBM Redbooks Project Leader with more than 10 years of experience in IBM. He has technical expertise in business problem solving, risk/root-cause analyze, and writing technical plans for business. He has had multiple roles at IBM, such as project management, ST/FT/ETE Test, back end Developer, DOL agent for NY, and is certified in IBM zOS Mainframe Practices, IBM Z® System programming, Agile, and Telecommunication Development Jumpstart. Henry holds a Master’s of MIS (Management Information System) from the University of Texas at Dallas.

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

Paul Bastide, Senior Software Engineer
IBM Infrastructure, Cambridge, Massachusetts, USA

Ajay Pratap, Senior Software Engineer,
Red Hat, India

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

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

Stay connected to IBM Redbooks

- Find us on 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

Part 1

Foundations

Part 1 provides a basic introduction to Red Hat OpenShift on IBM Power. For those that are new to Red Hat OpenShift this section provides the foundation for understanding the additional topics in this Redbook. For those that are familiar with Red Hat OpenShift but only on platforms other than IBM Power, this section will help you understand why you should consider running your cloud workload on the IBM Power platform when possible. For those that are very familiar with Red Hat OpenShift running on IBM Power, this section will be a review.

The following topics are covered in this section:

1

Red Hat OpenShift Fundamentals

This chapter describes the Red Hat OpenShift platform by giving a basic overview of its history, its key features, the role it plays in the modern IT landscape, and how it compares to similar platforms. This chapter also discusses Kubernetes as the foundation of Red Hat OpenShift, and how Red Hat OpenShift improves on the fundamental features of Kubernetes. We also show the advantages of using Red Hat OpenShift for orchestrating containers in your environment. The chapter concludes by providing a brief description of Red Hat OpenShift when implemented on IBM Power infrastructure.

This chapter contains the following topics:

1.1 Introduction to Red Hat OpenShift

Red Hat OpenShift is a leading enterprise Kubernetes platform that provides a robust foundation for developing, deploying, and scaling cloud-native applications. It extends Kubernetes with additional features and tools to enhance productivity and security, making it an ideal choice for businesses looking to leverage container technology at scale.

Red Hat OpenShift is a unified platform to build, modernize, and deploy applications at scale. Work smarter and faster with a complete set of services for bringing apps to market on your choice of infrastructure. Red Hat OpenShift delivers a consistent experience across public cloud, on-premise, hybrid cloud, or edge architecture.

Red Hat OpenShift offers you a unified, flexible platform to address a variety of business needs spanning from an enterprise-ready Kubernetes orchestrator to a comprehensive cloud-native application development platform that can be self-managed or used as a fully managed cloud service.

Figure 1-1 shows how Kubernetes is only one component (albeit a critical one) in Red Hat OpenShift.

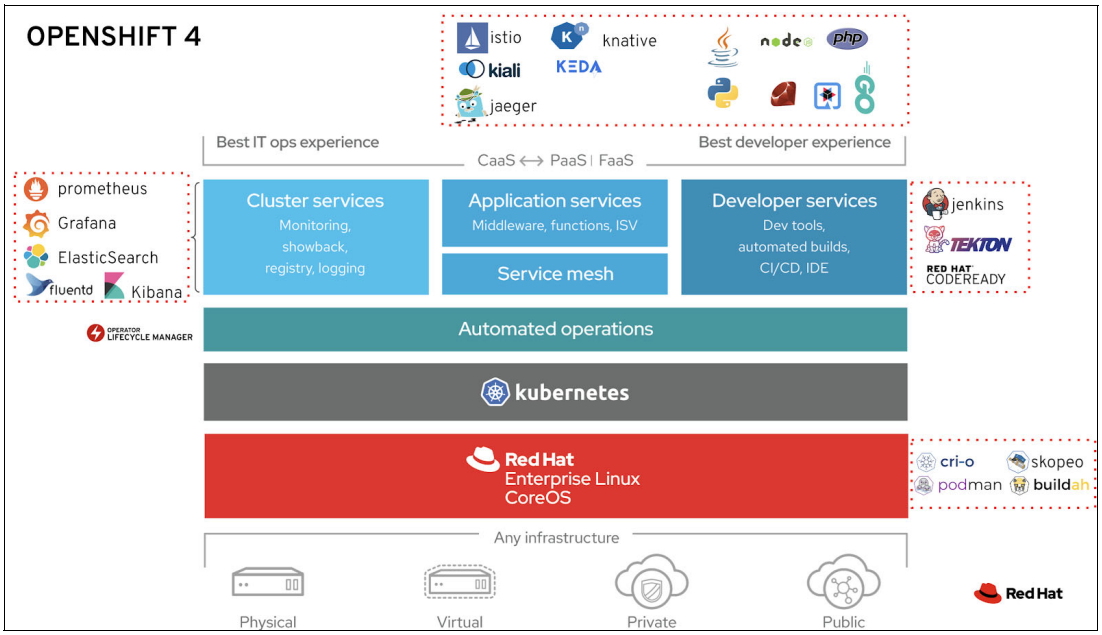


Figure 1-1 Red Hat OpenShift components

Red Hat OpenShift is a single platform that provides:

- The ability to:
  - Deploy and run in any environment.
  - The flexibility to build new applications.
  - Modernize existing applications.
  - Run third-party ISV applications.
  - Use public cloud services.
- The tools necessary to help customers integrate data analytics, artificial intelligence and machine learning (AI/ML) capabilities into cloud-native applications to deliver insight and value.

- Consistency and portability to deploy and manage containerized workloads, make infrastructure and investments future-ready, and deliver speed and flexibility. Either on-premise, across cloud environments, or to the edge of the network.
- Advanced security and compliance capabilities that allow end-to-end management and observability across the entire architecture.

Built by open source leaders, Red Hat OpenShift includes an enterprise-ready Kubernetes solution with a choice of deployment and usage options to meet the needs of your organization. From self-managed to fully managed cloud services, you can deploy the platform in the data center, in cloud environments, and at the edge of the network. With Red Hat OpenShift, you have the option to get advanced security and compliance capability, end-to-end management and observability, and cluster data management and cloud-native data services. Red Hat Advanced Cluster Security for Kubernetes modernizes container and Kubernetes security, letting developers add security controls early in the software life cycle. Red Hat Advanced Cluster Management for Kubernetes lets you manage your entire application life cycle and deploy applications on specific clusters based on labels, and Red Hat OpenShift Data Foundation supports performance at scale for data-intensive workloads.

1.1.1 Origin and evolution

Red Hat OpenShift is a continually evolving platform as Red Hat continues to integrate new features and functions to meet client needs. Here are some important evolutions in Red Hat OpenShift:

- Red Hat OpenShift originated as a Platform-as-a-Service (PaaS) from Red Hat, based on earlier technologies such as GearD and other cloud technologies. It was designed to simplify application development and hosting in cloud environments.
- With the rise of Kubernetes as the de facto standard for container orchestration, Red Hat OpenShift shifted its architecture to be based entirely on Kubernetes starting with Red Hat OpenShift 3.x. This transition allowed Red Hat OpenShift to harness the power of Kubernetes while adding additional enterprise features.
- The latest iterations, particularly Red Hat OpenShift 4.x, have introduced automated installation, updates, and lifecycle management of clusters. It integrates deeply with the hardware and software ecosystem, providing a fully managed container orchestration environment.

1.1.2 Key milestones

As Red Hat OpenShift has evolved, it has worked to meet different requirements within the enterprise cloud environment. This list shows how Red Hat OpenShift has broadened its focus to continue to provide additional benefits to the growing cloud environment:

- The initial releases focused on developer experience with easy application deployment, scaling, and management.
- The Integration of Kubernetes marked a significant pivot in Red Hat OpenShift's approach, aligning it with the Kubernetes API and ecosystem.
- Recent versions have emphasized automation with features like Operator Hub for managing Kubernetes applications and automated cluster operations.

1.1.3 Distinguishing features

There are many cloud and container management options available. Red Hat OpenShift has integrated these features to enhance the cloud and container management experience:

- With a focus on Developer-Centric Tools, Red Hat OpenShift enhances Kubernetes with developer-friendly tools, including Red Hat OpenShift Console – a developer-centric view for application management – and Source-to-Image (S2I) technology. This simplifies the process of building reproducible container images from source code.
- Designed with advanced security in mind, Red Hat OpenShift provides built-in security at every layer of the application stack – from the operating system (Red Hat Enterprise Linux CoreOS) to the application services. This ensures that compliance features and security best practices are built in from the start.
- With its focus on Hybrid Cloud Capabilities, Red Hat OpenShift is designed to operate across environments:
  - On-premise.
  - In a public cloud.
  - In a hybrid cloud environment.

  This provides consistent application portability and flexibility in deployment options.

Red Hat OpenShift is an enterprise-level production product that provides enterprise-level support for Kubernetes and Kubernetes management. Red Hat OpenShift provides the following benefits:

- Red Hat OpenShift offers automated installation, upgrades, and lifecycle management throughout the container stack – the operating system (OS), Kubernetes, cluster services, and applications – on any cloud.
- Red Hat OpenShift helps teams build with speed, agility, confidence, and choice. Get back to doing work that matters.
- Red Hat OpenShift is focused on security at every level of the container stack and throughout the application lifecycle. It includes long-term, enterprise support from one of the leading Kubernetes contributors and open source software companies.

1.1.4 The role of Red Hat OpenShift in modern IT

Red Hat OpenShift plays a crucial role in modern IT by facilitating the DevOps approach, improving software delivery speed, and enabling a more agile development environment. It offers a scalable platform that supports both microservices and traditional application models, accommodating a wide range of programming languages and frameworks. Through its comprehensive tool-set, Red Hat OpenShift addresses the needs of developers, system administrators, and IT managers, making it a pivotal tool in enterprise digital transformation strategies.

1.1.5 Comparison with other platforms

When evaluating Red Hat OpenShift, it's useful to compare it to other prominent Kubernetes platforms. This comparison can help highlight the unique features of Red Hat OpenShift and its suitability for various use cases.

AWS Elastic Kubernetes Service (EKS)

AWS EKS is provided as part of Amazon's cloud service and provides these features:

- Managed Service: Like Red Hat OpenShift, AWS EKS is a managed Kubernetes service, which simplifies cluster setup and maintenance. EKS handles the scalability and reliability of the infrastructure.
- Integration with AWS Services: EKS is deeply integrated with AWS services such as AWS Identity and Access Management (IAM), Amazon CloudWatch, and Elastic Load Balancing (ELB), making it an excellent option for users already embedded in the AWS ecosystem.

While EKS benefits from integration with AWS services it does not lend itself to hybrid cloud implementations across multiple cloud vendors. To get a more hybrid implementation, Red Hat OpenShift can be deployed on AWS, not only benefiting from native AWS integrations, but also providing additional tools and services not available in EKS, like Red Hat OpenShift's advanced CI/CD tools and developer-focused interfaces.

Google Kubernetes Engine (GKE)

Google Kubernetes Engine is an offering of Google's cloud services. GKS offers these features:

- Google Cloud Integration: GKE offers tight integration with Google Cloud services, including state-of-the-art networking and a robust security model. It leverages Google's expertise in container management.
- Auto-scaling and Site Reliability: GKE is known for its robust scaling capabilities and the backing of Google's site reliability engineering. It provides a reliable environment with a strong emphasis on performance and stability.

As GKE focuses on deep Google Cloud integration, it also lacks the true open hybrid cloud support that is provided by Red Hat OpenShift. Red Hat OpenShift provides a consistent and unified platform across various cloud and on-premise environments. The flexibility of Red Hat OpenShift can be particularly advantageous for organizations requiring multi-cloud or hybrid cloud approaches.

Microsoft Azure Kubernetes Service (AKS)

Microsoft AKS provides excellent integration with other Microsoft services in Azure. It offers these features:

- Microsoft Integration: AKS integrates seamlessly with other Microsoft services like Azure Active Directory and Azure Monitor. It is an ideal choice for enterprises heavily invested in Microsoft technologies.
- Cost-Efficiency and Scalability: AKS offers pay-as-you-go pricing, making it cost-effective for a variety of business sizes. It also supports auto-scaling and multi-node pools to efficiently manage resource usage.

As seen with the AWS and Google Kubernetes implementations, AKS does not provide a full multi vendor option as is provided by Red Hat OpenShift. Red Hat OpenShift can also be installed on Azure, allowing users to leverage Azure's cloud capabilities while adding the unique features of Red Hat OpenShift, such as enhanced security controls and a robust ecosystem of operational tools.

VMware Tanzu Kubernetes Grid

VMware provides Tanzu Kubernetes Grid (TKG) as a cloud implementation service which provides these features:

- VMware Ecosystem Integration: TKG is designed to integrate seamlessly with VMware's existing infrastructure and management tools, offering a smooth transition path for VMware users to adopt Kubernetes.
- Consistent Operation Across Environments: TKG focuses on providing a consistent Kubernetes experience across cloud and on-premise, similar to Red Hat OpenShift. However, Red Hat OpenShift extends this with more comprehensive developer tools and support for advanced deployment scenarios.

While both TKG and Red Hat OpenShift offer strong support for hybrid cloud environments, Red Hat OpenShift provides more extensive developer tools and automation features.

1.1.6 Key takeaways

When considering which cloud management platform to use, consider this:

- Red Hat OpenShift provides extensive enterprise features out-of-the-box, including advanced security features, integrated developer tools, and extensive automation capabilities that may not be as comprehensive in other Kubernetes services.
- Red Hat OpenShift's ability to deploy across multiple environments (cloud, on-premise, and hybrid) with consistency makes it a strong choice for organizations with complex infrastructure needs.
- Red Hat OpenShift distinctly benefits developers with features like Source-to-Image (S2I), a comprehensive web console, and application templates, which facilitate a smoother and more productive development experience compared to basic Kubernetes services.

Red Hat OpenShift is a strong leader in the cloud landscape of Kubernetes platforms, and is chosen for its strengths in enterprise environments, multi-environment consistency, and developer-centric features.

1.2 Core concepts of Kubernetes in Red Hat OpenShift

This section describes the core concepts of Kubernetes. Understanding the basics of a Red Hat OpenShift Kubernetes cluster is required to understand what a Multiple Architecture Cluster is and what benefits are provided by a Multi-Arch Compute implementation.

1.2.1 Kubernetes fundamentals

Kubernetes serves as the backbone of Red Hat OpenShift, providing the essential framework for orchestrating containerized applications. Understanding these core concepts is crucial for leveraging Red Hat OpenShift effectively. This section provides a detailed exploration of the fundamental components and mechanisms of Kubernetes as implemented in Red Hat OpenShift.

Basic components

The basic components of Kubernetes are:

Pods A pod is the smallest deployable unit which can be created and managed by Kubernetes.

	A pod is a group of one or more containers that share storage, network, and specifications on how to run the containers. Pods are ephemeral by nature; they are created and destroyed to match the state specified by users.
Nodes	A nodes is a physical or virtual machine where Kubernetes runs pods. A node can be a worker node or a master node, although with the latest Kubernetes (and by extension Red Hat OpenShift) practices, the distinction is often abstracted away, especially in managed environments.
Clusters	A cluster consists of at least one worker node and at least one master node. The master node manages the state of the cluster, including scheduling workloads and handling scaling and health monitoring while the worker node is where applications are run.

Figure 1-2 shows a basic cluster architecture. While a cluster can technically be created with one master node and two worker nodes. Best practices generally recommend at least three master nodes, which can share functions and provide failover, and three or more worker nodes to provide failover and scalability for your application workloads.

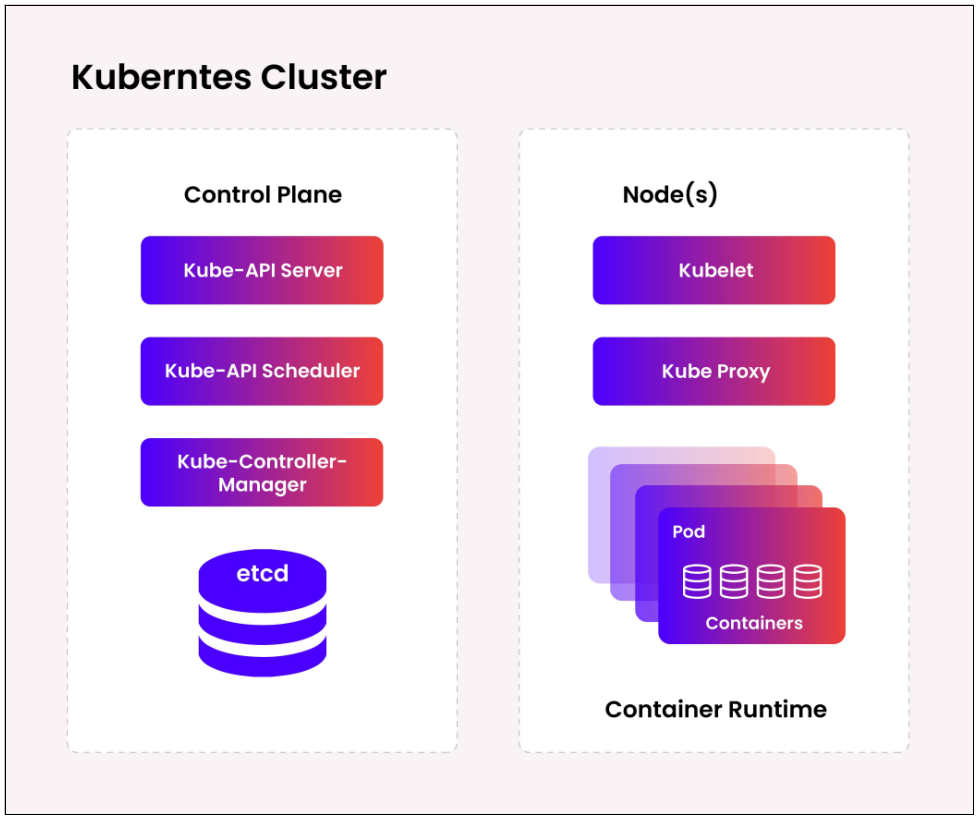


Figure 1-2 Base Kubernetes cluster

Control plane components

As outlined in the previous section, a Kubernetes cluster consists of two primary types of nodes: master nodes and worker nodes.

- Worker Nodes: These nodes are responsible for running the containers that host your applications and perform the tasks essential to your business operations.

- Master Nodes: These nodes manage the cluster by running the control plane services. They oversee and control the worker nodes and manage the pods that run the application code.

Together, the master nodes form what is known as the control plane, which is crucial for the orchestration and management of the entire Kubernetes cluster.

The major services that are running in the control plane are:

API Server The API server acts as the front end for Kubernetes. The API server is the component that clients and external tools interact with.

etcd The etcd service is a highly-available key-value store used as Kubernetes' backing store for all cluster data. It is used to maintain the state of the cluster.

Scheduler The scheduler watches for newly created pods with no assigned node, and selects a node for them to run on based on resource availability, policies, and specifications.

Controller Manager The controller manager runs controller processes, which are background tasks in Kubernetes that handle routine tasks such as ensuring that the correct number of pods are running for replicated applications.

Workload resources

The control plane is in charge of setting up and managing the worker nodes which are running the application code. Workload components can be described as:

Deployments A deployment specifies a desired state for a group of pods. You describe a desired state in a deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate. You can define deployments to create new ReplicaSets, or to remove existing deployments and adopt all their resources into new deployments.

ReplicaSet A ReplicaSet's purpose is to maintain a stable set of replica pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical pods. This maintains a stable set of replica pods running at any given time.

StatefulSets StatefulSets are used for applications that require persistent storage and a unique identity for each pod, making them ideal for databases and other stateful applications.

DaemonSets DaemonSets ensure that each node in the cluster runs a copy of a pod. They are useful for deploying system services that need to run on all or certain nodes.

Networking

Connectivity between pods and between pods and outside services is managed within a Kubernetes cluster. The following functions are maintained by the cluster:

Service A service is an abstraction that defines a logical set of pods and a policy by which to access them. Services enable communication between different pods and external traffic routing into the cluster.

Ingress Ingress controllers manage external access to the services in a cluster, typically utilizing HTTP. Ingress controllers can provide load balancing, SSL termination, and name-based virtual hosting.

Storage

Containers are by definition ethereal as is any data stored in the container. To enable persistent storage, Kubernetes uses the following concepts:

Persistent Volumes Persistent volumes (PVs) are resources in the cluster which can be connected to containers to provide persistent storage.

Persistent Volume Claims Persistence volume claims (PVCs) are requests for storage by users. These requests are satisfied by allocating PVs.

Configurations, secrets and security

ConfigMaps and Secrets provide configuration flexibility and authentication capabilities, while additional security is provided through the use of Role-Based Access Control (RBAC).

ConfigMaps Allows you to decouple configuration artifacts from image content to keep containerized applications portable.

Secrets Used to store and manage sensitive information such as passwords, OAuth tokens, and SSH keys.

RBAC RBAC Controls authorization – determining what operations a user can perform on cluster resources. It is crucial for to maintain the security of the cluster.

1.2.2 Red Hat OpenShift enhancements to Kubernetes

While Kubernetes provides a robust container orchestration platform, Red Hat OpenShift builds on this foundation with additional features and tools tailored for enterprise environments and developer work flows. These enhancements boost usability, security, and operational efficiency, including:

- A user-friendly web console for streamlined management and monitoring.
- Enhanced security features designed for rigorous compliance requirements.
- Built-in CI/CD tools to facilitate the development process.

This section explores how Red Hat OpenShift extends the core Kubernetes architecture.

Enhanced Developer Productivity

Red Hat OpenShift offers a sophisticated web-based console that surpasses the standard Kubernetes dashboard in usability. This console enables developers to efficiently manage projects, visualize application states, and access a wide range of development tools directly.

- CodeReady Containers simplifies setting up local Red Hat OpenShift clusters for development. It provides a minimal, pre-configured environment that runs on a developer's workstation, streamlining the initial setup experience.
- Source-to-Image (S2I) is a key feature for creating reproducible container images from source code. It automates the process of fetching code, embedding it into a container image, and assembling a new image with the necessary runtime artifacts, thus optimizing the work flow from source code to deployment.

Advanced Security Features

Red Hat OpenShift enhances Kubernetes security with:

- Security Context Constraints (SCCs): These offer more granular control than Pod Security Policies (PSPs), allowing administrators to set specific conditions that pods must meet to be deployed, such as prohibiting containers from running as root.

- OAuth server integration: Red Hat OpenShift integrates with external identity providers for a streamlined authentication and authorization process, enabling users to log in with corporate credentials and enhancing overall security.
- Extended network policies: While Kubernetes supports network policies, Red Hat OpenShift adds egress firewall capabilities, giving administrators control over outbound traffic from pods to external networks.

Operational Efficiency

Red Hat OpenShift improves operational efficiency through:

- Operator Pattern: By adopting the Kubernetes Operator pattern, OpenShift automates the deployment, scaling, and management of complex applications. The Operator Hub provides a marketplace for deploying Operators for various software stacks.
- Streamlined Installation and Updates: OpenShift offers an automated installation process for production-grade clusters and supports automatic updates, reducing downtime and manual intervention.
- Built-in Monitoring and Telemetry: Pre-configured monitoring capabilities collect metrics from all parts of the cluster, offering insights into application and infrastructure performance and enabling proactive management.

Enterprise Integration and Support

Red Hat OpenShift integrates advanced features for enterprise needs:

- Istio-based Service Mesh: Integrated directly into the platform, Istio supports secure, connected, and manageable microservices architectures. It provides service discovery, load balancing, failure recovery, metrics, monitoring, and supports complex operations such as A/B testing and canary releases.
- CI/CD Tool Integration: OpenShift offers built-in Jenkins support and integrates with other major Continuous Integration / Continuous Delivery (CI/CD) platforms, automating the build, test, and deployment lifecycle within the same platform.

1.2.3 Key features of Red Hat OpenShift

Red Hat OpenShift is strongly focused on the developer's experience and has integrated many features that are designed to make development of applications more efficient and productive. This section provides an overview of some of those enhancements.

Developer productivity

Red Hat OpenShift is designed to enhance developer productivity by streamlining processes and reducing the complexities typically associated with deploying and managing applications. Here is a detailed look at how Red Hat OpenShift achieves this through its key features:

- Developer-Focused User Interface
  - The Red Hat OpenShift Console is a powerful, user-friendly interface that provides developers with an overview of all projects and resources within the cluster. It offers a perspective tailored to developers' needs, allowing them to create, configure, and manage applications directly from the browser. Features like the Topology view let developers visualize their applications and services in a graphical interface, making it easier to understand and manage the relationships between components.
  - Red Hat OpenShift includes a Developer Catalog that offers a wide array of build and deploy solutions, such as databases, middleware, and frameworks, which can be deployed on the cluster with just a few clicks.

This self-service portal accelerates the setup process for developers, allowing them to focus more on coding and less on configuration.

- Code-Ready workspaces
  - Red Hat OpenShift integrates with Code-Ready Workspaces, a Kubernetes-native IDE that developers can use within their browser. This IDE provides a fully featured development environment, complete with source code management, runtimes, and dependencies that are all managed and kept consistent across the development team. This ensures that the entire team works within a controlled and replicable environment, reducing “works on my machine” problems.
- Application templates and source-to-image
  - Red Hat OpenShift application templates are predefined configurations for creating applications based on specific languages, frameworks, or technologies. These templates include everything needed to build and deploy an application quickly, such as build configurations, deployment strategies, and required services.
  - Source-to-image (S2I) is a tool for building reproducible Docker images from source code. S2I lets developers build containerized applications without needing to write Dockerfiles or become experts in Docker. It combines source code with a base Docker image that contains the appropriate runtime environment for the application. The result is a ready-to-run Docker image built according to best practices.
- Automated build and deployment pipelines
  - Red Hat OpenShift has robust support for CI/CD processes, integrating tools like Jenkins, GitLab CI, and others directly into the platform. It automates the build, test, and deployment pipeline, enabling developers to commit code changes frequently without the overhead of manual steps.
  - Red Hat OpenShift can automatically trigger builds and deployments when code changes are pushed to a source code repository or when other specified events occur. This feature ensures that applications are always up-to-date with the latest code changes.
- Live application development
  - Red Hat OpenShift supports hot deployments, where changes to application code can be made active without restarting the entire application. This capability is crucial for environments where uptime is critical, and it allows developers to see changes instantly.
  - Developers can access real-time logs and debugging tools directly through the Red Hat OpenShift console, making it easier to diagnose and resolve issues in development and production environments.

By focusing on these aspects of developer productivity, Red Hat OpenShift significantly lowers the barrier to entry for deploying applications in a Kubernetes environment, simplifies the management of these applications, and accelerates the development cycle. This enables developers to spend more time coding and less time dealing with deployment complexities, leading to faster innovation and deployment cycles in a cloud-native landscape.

1.2.4 Enterprise-grade security

Red Hat OpenShift is designed with security as a foundational aspect, integrating robust security features that support the demanding requirements of enterprise environments. This includes everything from strict access controls to ensuring container and platform integrity. Here is a deeper look into how Red Hat OpenShift delivers enterprise-grade security:

- Security Context Constraints
  - Red Hat OpenShift enhances the security of container environments by using Security Context Constraints (SCCs) to define a set of conditions that a container must comply with to run on the platform. These role-based constraints can limit the actions that a pod can perform and the resources it can access, significantly reducing the risk of unauthorized actions.
  - Using SCCs provides a fine-grained permission structure where administrators can use SCCs to manage permissions at a granular level – controlling whether pods can run as privileged containers, access sensitive volumes, or use host networking and ports along with other security settings.
- Integrated authentication and authorization
  - Red Hat OpenShift integrates with existing enterprise authentication systems – such as LDAP, Active Directory, and public OAuth providers – to provide a robust user authentication process seamlessly across the organization.
  - Role Based Access Control (RBAC) in Red Hat OpenShift allows administrators to regulate access to resources based on the roles of individual users within the enterprise. This ensures that only authorized users have access to control critical operations, thereby securing the environment against internal and external threats.
- Network policies and encryption
  - Red Hat OpenShift allows administrators to define network policies that govern how pods communicate with each other and with other network endpoints. This ensures that applications are isolated and protected from network-based attacks.
  - Data in transit and at rest can be encrypted – providing an additional layer of security. Red Hat OpenShift supports TLS for all data in transit and can integrate with enterprise key management solutions to manage encryption keys for data at rest.
- Security enhancements and compliance
  - Red Hat OpenShift provides automated mechanisms to apply security patches and updates to the container host, runtime, and the application containers themselves. This helps in maintaining security compliance and reducing the vulnerability window.
  - Red Hat OpenShift includes features to support compliance with various regulatory requirements such as PCI DSS, HIPAA, and GDPR. It provides extensive logging and auditing capabilities that help in tracking all user actions and system changes, crucial for forensic analysis and compliance reporting.
- Container security and image assurance
  - Red Hat OpenShift integrates with tools like Quay.io to provide automated container image scanning. This scans images for vulnerabilities before they are deployed, and image signing ensures that only approved and verified images are used in the environment.
  - Red Hat OpenShift runs on Red Hat Enterprise Linux and leverages Security-Enhanced Linux (SELinux) to enforce mandatory access control policies that isolate containers from each other and from the host system. This prevents a compromised container from affecting others or gaining undue access to host resources.
- Secure default settings and practices
  - Red Hat OpenShift encourages the use of minimal base images that contain only the essential packages needed to run applications, reducing the potential attack surface.

– Red Hat OpenShift is preconfigured with security best practices and regularly updated security benchmarks that guide users in setting up and maintaining a secure environment.

By providing these comprehensive security features, Red Hat OpenShift addresses the complex security challenges faced by enterprises today, ensuring that their deployments are secure by design, compliant with industry standards, and capable of withstanding modern cybersecurity threats. This security-first approach is integral to maintaining trust and integrity in enterprise applications and data.

1.3 Benefits of Using Red Hat OpenShift for Container Orchestration

There are a number of benefits to using Red Hat OpenShift for container orchestration:

Security	Built-in security ensures inherent security of your applications, data, and infrastructure, as it functions as a highly secure container orchestration platform.
Flexibility	Red Hat OpenShift enables developers the capability of deploying and managing applications on-premise, in the cloud, across multiple cloud providers, or even in a hybrid cloud setup.
Easy deployment	By automating the whole process of deploying and managing applications that reside in containers, Red Hat OpenShift greatly simplifies the entire deployment process.
Productivity	Red Hat OpenShift enables developers to focus on writing quality code while delivering applications faster by providing a platform that simplifies the entire development process.
Integrated tools	Red Hat OpenShift enables integration of a wide variety of tools, such as Docker, Jenkins, and Git, so developers can use their preferred tools for developing and deploying applications.
Customization	Red Hat OpenShift allows you the ability to customize the platform however you need, enabling you to have a highly-tailored environment for your applications.

1.4 Red Hat OpenShift on IBM Power

There are numerous options for running cloud workloads, whether in private or public cloud environments. This section focuses on utilizing Red Hat OpenShift on IBM Power servers, whether deployed within your enterprise or through a public cloud service like IBM Power Virtual Server (PowerVS). PowerVS is an Infrastructure-as-a-Service (IaaS) offering that operates on IBM Power servers located in IBM data centers around the world.

1.4.1 Red Hat OpenShift architecture

As previously mentioned, Red Hat OpenShift is an open-source container application platform built on Kubernetes and running on Red Hat Enterprise Linux CoreOS (RHCOS). It offers integrated features for scaling, monitoring, logging, and metering.

Red Hat OpenShift provides a comprehensive solution for hybrid cloud environments, encompassing essential components such as a container runtime, networking, monitoring, a container registry, authentication, and authorization.

Note: For more information about Red Hat OpenShift architecture and design, and Red Hat OpenShift on IBM Power in general, see the IBM Redbooks publication *Implementing, Tuning, and Optimizing Workloads with Red Hat OpenShift on IBM Power*, SG24-8537.

Red Hat OpenShift components in IBM Power

We have already described Red Hat OpenShift architecture and its components in section 1.2, “Core concepts of Kubernetes in Red Hat OpenShift” on page 8. This section describes how these components can be implemented on IBM Power.

Infrastructure layer

In the infrastructure layer, applications can be hosted on physical servers, virtual servers, or in the cloud (private or public). In Red Hat OpenShift, these servers are referred to as nodes. Nodes are the fundamental units of computer hardware, responsible for storing and processing data.

This book focuses on utilizing IBM Power servers, which come with built-in virtualization capabilities. These servers can run both traditional and cloud workloads within different logical partitions (LPARs) on the same physical hardware. By leveraging the strengths of IBM Power in virtualization, you can effectively manage and integrate new cloud applications with your existing traditional workloads.

Using your current IBM Power infrastructure, you can seamlessly integrate new cloud applications with legacy systems, maximizing the value from existing applications while rapidly developing new workflows to capitalize on emerging business opportunities.

Service Layer

The service layer manages the definition of pods and access policies. It assigns permanent IP addresses and hostnames to pods, facilitates connectivity between applications, and enables internal load balancing by distributing tasks across application components. The control nodes, which are part of the control plane, oversee the cluster and manage the worker nodes. They are crucial for the cluster's operation and should be hosted on infrastructure that ensures the highest levels of reliability and availability.

1.4.2 Benefits of using IBM Power for a Red Hat OpenShift environment

In Chapter 2, “IBM Power” on page 19, we discuss IBM Power and why it proves to be a superior platform for running Red Hat OpenShift compared to architectures such as x86. IBM Power servers are designed for reliability, availability, and serviceability. Some of the largest companies in the world run their businesses on IBM Power systems, including 80 of the Fortune 100. They trust IBM Power to run their business in a secure environment with minimal unplanned downtime so that they can implement the best hybrid cloud strategy to manage effectively large amounts of data and better serve their customers.

IBM Power is designed for security and for performance. They have one of the smallest number of known security issues in the industry. They are built with high performance with industry-leading connectivity and scalability to handle many concurrent users and work with large data sources effectively.

IBM Power provides a flexible platform with cloud-like scalability and pricing and is also available as a hybrid cloud offering using IBM PowerVS.

An advantage of choosing IBM Power servers for your cloud infrastructure is the ease of migration of your current workload and data into your new cloud environment without having to re-platform them.

IBM Power servers can be the right solution for your cloud requirements. We discuss the IBM Power architecture in more detail in Chapter 2, “IBM Power” on page 19.

2

IBM Power

In this chapter, we discuss the IBM family of reliable, high-performance systems: IBM Power. We provide an overview of IBM Power as one of the leading enterprise server architectures in the market and we enumerate the various advantages of the Power architecture over x86. We also describe various reasons why the IBM Power10 processor serves as an ideal platform for modernizing your applications through Red Hat OpenShift Container Platform.

In this chapter we discuss the following:

2.1 Introduction to IBM Power

IBM Power is a family of systems that are capable of running mission-critical workloads utilizing hybrid multicloud technologies. IBM Power servers are high-performance, secure, and reliable servers built on the IBM Power processor architecture. Figure 2-1 is a view of the IBM Power E1080, the high-end enterprise model of the IBM Power product portfolio.

Figure 2-1 View of Power E1080 node

IBM Power is built to be scalable and powerful, while also providing flexible virtualization and management features. IBM Power supports a wide range of open-source platforms, including Red Hat OpenShift.

Many of the world's most mission-critical enterprise workloads are run on IBM Power. The core of the global IT infrastructure, encompassing the financial, retail, government, health care, and every other sector in between, is comprised of IBM Power systems, which are renowned for their industry-leading security, reliability, and performance attributes. For enterprise applications, including AI, ERP, databases, application and web servers, many clients use IBM Power.

Enterprise IT delivery is changing as a result of digital transformation – cloud computing is playing a key role. When it comes to consuming infrastructure, you need options and flexibility, and IBM Power is designed to run in your datacenter utilizing flexible cloud capabilities or in the cloud. Whether you are using Red Hat OpenShift and Kubernetes to modernize enterprise applications, creating a private cloud environment within your data center using adaptable pay-as-you-go services, using IBM Cloud to launch applications as needed, or creating a seamless hybrid management experience across your multicloud landscape – IBM Power is fully capable of delivering whatever hybrid multicloud approach you choose.

The modern data center consists of a combination of on-premise and off-premise, multiple platforms, such as IBM Power, IBM Z, and x86. Applications range from monolithic to cloud-native – which is inherently some combination of bare metal, virtual machines, and containers. An effective hybrid cloud management solution must account for all of these factors. IBM and Red Hat are uniquely positioned to best accommodate the applications that you are running today and the modernized applications of tomorrow, wherever they reside.

IBM Power delivers one of the highest availability ratings among servers

According to the ITIC *2023 Global Server Hardware, Server OS Reliability* survey polling nearly 1,900 corporations worldwide across over 30 vertical market segments, an 88% majority of the newest IBM Power10 server users (shipping since September 2021) say their organization achieved *eight nines*--99.999999% of uptime. This is *315 milliseconds of unplanned outage time, per server, per year* due to underlying system flaws or component failures (second only to IBM Z with 31.56 milliseconds of per server annual downtime). So, Power10 corporate enterprises spend just $7.18 per server/per year performing remediation due to unplanned server outages that occurred due to inherent flaws in the server hardware or component parts.

This marks 2023 as the 15th consecutive year that the IBM Z and IBM Power Systems have dominated with the best across-the-board uptime reliability ratings among 18 mainstream distributions.

This level of availability is largely due to the inherent availability features of Power10, allowing for less downtime than comparable offerings, due to built-in recovery and self-healing functions for redundant components. Organizations are also able to switch from an earlier Power server to the current generation while applications continue to run, giving you high-availability and minimal downtime when migrating.

IBM Power is consistently rated as one of the most secure systems in the market

For the fourth straight year, IBM Power has been rated as one of the most secure systems in 2022, with only 2.7 minutes or less of unplanned outages due to security issues**.** This means that IBM Power is:

- 2x more secure than comparable HPE Superdome servers,
- 6x more secure than Cisco UCS servers,
- 16x more secure than Dell PowerEdge servers,
- 20x more secure than Oracle x86 servers
- up to more than 60x compared to unbranded white box servers.

Security breaches were also detected immediately or within the first 10 minutes in 95% of the IBM Power systems that were surveyed. This results in better chances that a business will suffer little to no downtime, nor will they be susceptible to damaged, compromised, or stolen data.

IBM Power allows businesses to boost operational efficiency to meet sustainability goals

A recent user case study illustrated how IBM Power enabled a customer to increase end-user application performance by 20%, ending up with them meeting their sustainability goals**.** By helping move the customer to IBM Power and IBM FlashSystem® storage, they were fully able to leverage their SAP S/4HANA operations enabling them to meet their climate objectives.

IBM Power streamlines AI operations with advanced on-chip technologies

IBM Power systems delivers 5X faster AI inferencing per socket for high precision math over the previous generation**.** This is accomplished through multiple Matrix Math Accelerator (MMA) units in each Power processor core. MMAs allow IBM Power systems to forgo external accelerators, such as GPUs and related device management, when running machine learning and inferencing workloads.

Current IBM Power systems can range from scale-out servers that start with 1 core and 32 GB of memory on the IBM Power S1012 to enterprise systems with up to 240 cores and 64 TB of memory on the IBM Power E1080.

Note: The full lineup of IBM server models based on the latest Power processors can be found at IBM Power.

2.1.1 Supported operating systems

As of the time of this publication, Power10 processor-based systems support the platforms/operating system versions shown in Table 2-1.

Table 2-1 Power10 platform / operating system support matrix

Operating System	Supported versions
Red Hat OpenShift Container Platform	4.9 or later
PowerVM Virtual I/O Server	4.1.0.0 or later 3.1.4.10 or later 3.1.3.10 or later 3.1.2.30 or later 3.1.1.50 or later
AIX	7.3 TL0 or later 7.2 TL4 or later (with any I/O configuration) 7.1 TL5 or later (through VIOS only)
IBM i	7.5 or later 7.4 TR5 or later 7.3 TR11 or later
Red Hat Enterprise Linux	8.4 or later 9.0 or later
SUSE Linux Enterprise Server	15.3 or later 12.5 or later
Ubuntu	22.04 or later

Note: The reference system used in the table above is the Power E1080. Software maps detailing which versions are supported on which specific IBM Power server models (including previous generations of IBM Power) can be found at IBM Support.

2.2 IBM Power processor architecture overview

The Power processor is a family of 64-bit superscalar, simultaneous multithreading, multicore microprocessors designed and sold by IBM.

Power processors are mainly used for the IBM Power line of servers, the Hardware Management Console (HMC), and also in IBM storage solutions such as the DS8900F, ESS (Elastic Storage System) and the IBM Converged Archive Solution.

As an architecture, Power-based processors have been used in various applications such as network routers, workstations, game consoles, and even on both the Curiosity and the Perseverance rovers on Mars.

As AI infrastructure challenges increase due to more AI models being deployed in production, IBM Power addresses these challenges with in-core AI inferencing and machine learning through its built-in Matrix Math Accelerator (MMA). This allows you the capability of “AI at the point data” where you can perform AI inferencing without needing to ingest your data from an external source. This gives AI operations a significant performance boost.

The Power processor also provides security within the system itself through Transparent Memory Encryption, where data is encrypted by cryptography engines located in the processor core, right where memory is located. This gives you four times the speed compared with average encryption accelerators.

Power also features reliability, availability, and serviceability (RAS) capabilities such as advanced recovery, diagnostics, and Open Memory Interface (OMI) attached advanced memory DIMMs that deliver 2X better reliability and availability than industry standard DIMMs.

The latest version of Power processors is the Power10, built on a 7nm design that is 50% faster than its predecessor and 33% more energy efficient.

Important: While this performance gain is impressive, we recommend performing a performance/capacity study with an authorized business partner or IBM Expert Labs team to properly reap those gains.

Power10 chip benefits are the result of important evolutions of many of the components that were in previous IBM POWER® chips. Several of these important Power10 processor improvements are listed in Table 2-2.

Table 2-2 Power10 processor chip technology

Technology	IBM Power10 processor chip
Processors die size	602 mm2
Fabrication technology	CMOSa 7-nm lithography 18 layers of metal
Maximum processor cores per chip	15
Maximum execution threads per core / chip	8 / 120
Maximum L2 cache core	2MB
Maximum On-chip L3 cache per core / chip	8MB / 120MB
Number of transistors	18 billion
Processor compatibility modes	Support for Power Instruction Set Architecture (ISA) of POWER8 and POWER9

Figure 2-2 on page 24 shows the Power10 processor die with several functional units that are labeled. Sixteen SMT8 processor cores are shown, but the dual-chip module (DCM) with two Power10 processors provide 12, 18, or 24 core options for Power E1050 server configurations.

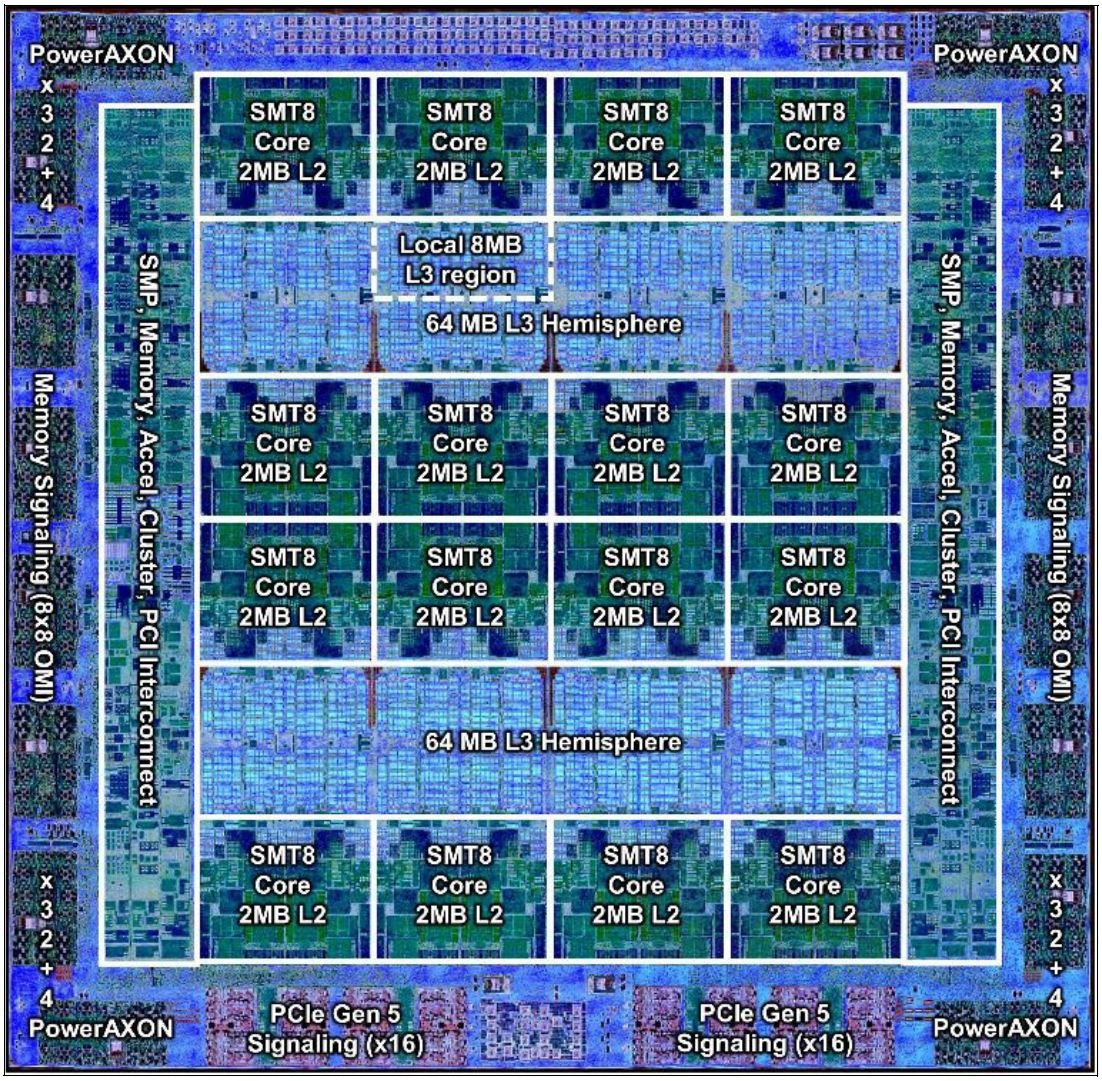


Figure 2-2 The Power10 processor chip (die photo courtesy of Samsung Foundry)

2.3 Advantages of modernizing your applications with IBM Power10

Application modernization comes in many shapes and sizes, and it is not always easy to know where to start. Here we describe how IBM Power brings strengths and benefits to your modernization efforts. There are many more benefits than those enumerated here.
IBM Power is built for core enterprise applications and the next wave of digital transformation fueled by application modernization. Here are a few advantages of modernizing with IBM Power10.

Pervasive security and resiliency

To meet today's security challenges, it is essential that every layer of your company's IT hardware and software stack remains secured. IBM Power customers utilize the most reliable mainstream server platform to innovate and get to market faster without compromising security.

IBM Power's multi-layered approach to security gives you full visibility of your hardware and software. With IBM Power10's hardware-accelerated transparent memory encryption, quantum-safe cryptography and fully homomorphic encryption, your data is protected with comprehensive end-to-end security at every layer of the stack – for both today's and tomorrow's threats.

More performance from software with fewer servers

You can buy fewer IBM Power servers to run an equivalent set of applications at comparable throughput levels than on competing platforms. That is because it provides 55% lower 3-year total cost of ownership (TCO) to run modern cloud-native applications – achieving 4.4X better per-core throughput. This allows the collocation of cloud-native apps with existing AIX, IBM i and Linux virtual machine-based applications enabling access to low-latency API connections to business-critical data. Plus, you can leverage sub-capacity licensing to greatly reduce containerized software license costs (IBM Cloud Paks, for example) using PowerVM shared processor pools, allowing CPU cores to be autonomously shared across Red Hat OpenShift worker nodes without sacrificing application performance.

Superior performance for your enterprise data

Running Red Hat OpenShift in a virtual machine adjacent to your AIX, IBM i or Linux virtual machines provides low-latency reliable communication to your enterprise data with PowerVM Virtual I/O Server. This provides improved performance due to fewer network hops. It also allows for security-enhanced communication between the new cloud-native apps and your enterprise data stores as the network traffic never has to leave the physical server.

Flexible, efficient utilization

You can manage spikes in demand and support more cloud workloads per server with IBM PowerVM hypervisor on-demand CPU capacity for IBM Power compute and memory. Power virtualization technology manages demand by sharing pools of CPU cores across nodes. These differentiating hypervisor and consumption constructs – such as uncapped processors and shared processor pools – provide the ability to guarantee performance SLAs while donating unused processor cycles to worker nodes in need of additional capacity. Additionally, on-premise pay-as-you-go consumption is available for Red Hat OpenShift running on IBM Power.

Trusted and proven foundation

Kubernetes provides the core foundation for modernizing your enterprise applications. As the industry's leading enterprise Kubernetes platform, Red Hat OpenShift provides a consistent foundation for application development and containerized workloads to support hybrid cloud, multicloud and edge deployments. This benefits both developers and IT administrators. Your developers have access to the latest software innovations within Red Hat OpenShift to build solutions faster while your IT administrators can easily observe, operate and manage the platform and infrastructure. This helps you deliver high-value, high-quality software faster to end users. All of this is enabled through Red Hat OpenShift.

Red Hat OpenShift Container Platform

Red Hat OpenShift is the industry's leading enterprise-ready Kubernetes platform that can run anywhere – either on-premise in your data center, on IBM Cloud, or on third-party cloud providers like AWS, Azure or Google.

Red Hat OpenShift is optimized to improve developer productivity and promote innovation; it is fully supported on all IBM Power servers starting with IBM POWER9 processors when running RHOS 4.14 or later.

For an even more flexible solution Red Hat OpenShift can be paired with Red Hat OpenShift Data Foundation or IBM Storage Fusion to provide a flexible software-defined storage solution to simplify cloud transformation projects.

Incremental application modernization

With IBM Power10, teams can incrementally modernize their existing AIX, IBM i, and Linux applications by extending them with new cloud-native services in a safe and methodical manner. This means you can capitalize on existing investments in applications and skills and drive incremental transformation – saving money, expediting time-to-value and minimizing risk.

For IBM i clients, this is made even easier with Merlin (IBM i Modernization Engine for Lifecycle Integration). Merlin is a set of tools that run in Red Hat OpenShift containers that guide and assist developers in the modernization of IBM i apps.

IBM Cloud Paks and Red Hat software

IBM Power provides superior performance and economics for containerized workloads like IBM Cloud Paks and an extensive set of Red Hat open-source software solutions for modernizing existing applications and developing new cloud-native apps that run on Red Hat OpenShift.

There are three main benefits of IBM Cloud Paks:

- They are comprehensive and easy to use.
- They are supported by IBM.
- They run anywhere Red Hat OpenShift runs.

IBM Cloud Paks take a bundled approach that allows you to accelerate your modernization journey by packaging everything you need to get started. The IBM Cloud Paks available on IBM Power include IBM WebSphere Hybrid Edition (formerly IBM Cloud Pak for Applications), IBM Cloud Pak for Data, IBM Cloud Pak for Watson AIOps (Infrastructure Automation), IBM Cloud Pak for Integration and IBM Cloud Pak for Business Automation. With the addition of multiple architecture clusters it is now possible to integrate additional Cloud Pak capabilities into your IBM Power based Red Hat OpenShift clusters utilizing x86 worker nodes.

From a Red Hat software perspective, there is also a comprehensive set of software solutions to accelerate your modernization efforts, including Red Hat Runtimes, Red Hat 3scale API Management, Red Hat Fuse and Red Hat AMQ.

Innovate with an extensive container software ecosystem

At the heart of any application modernization effort is a strong software ecosystem that allows teams to innovate using the latest technologies. Now, more than ever, open-source communities are playing a significant role in an organization’s modernization journeys.

IBM Power not only runs your core business applications, but also a wide range of popular open-source and commercial container software running on Red Hat OpenShift. When you choose IBM Power to modernize, you choose industry-leading reliability, performance and security, as well as superior compute performance for data-intensive and mission-critical applications. It is a foundation for modern container-based applications.

Comprehensive hybrid cloud management and automation

As teams increasingly shift to a hybrid cloud IT model, the need for consistent management, observability and automation approaches is paramount. Consistency across hardware platforms, clouds and operating systems is crucial for IT administrators and developers.

IBM and Red Hat check these boxes with IBM Cloud Pak for Watson AIOps (Infrastructure Automation), IBM Instana™ Observability, IBM Turbonomic® Application Resource Management, Red Hat Advanced Cluster Management for Kubernetes and Red Hat Ansible Automation Platform — all of which extend the value of the IBM Power platform. IBM Power makes operating and automating your hybrid cloud much easier.

2.4 Key benefits of IBM Power compared to x86 servers

It is often thought that x86 servers provide the best platform for implementing cloud-computing. This is primarily based on the assumption that the x86 based servers are the lowest cost. While it is sometimes true that x86 servers have the lowest acquisition costs, this often does not take into account performance, scalability, reliability, and manageability. The return on investment and total cost of ownership of x86 servers generally pales in comparison to those of the IBM Power systems. IBM delivers the better overall platform compared to x86 when you take into account the following benefits of IBM Power:

- World record SAP SD-two tier benchmark results with 8 sockets (120 cores), beating the best 16 socket (448 cores) result from the x86 platform.
- Power delivers per-core performance that is 2.5X faster than Intel Xeon Platinum, setting a world record 8-socket single server result on the SPEC CPU 2017 benchmark.
- When running containerized applications and databases on an IBM Power E1080 compared to running the same workloads on an x86 server, IBM Power delivers 48% lower 3-year TCO, 4.3X more throughput per core, and 4.1X better price-performance. This means you can run the same amount of workloads with fewer servers, four times less footprint, four times fewer software licenses, and four times the energy savings.
- For the 15th straight year, IBM Power delivers the top reliability results, better than any Intel x86 platform, and only exceeded by the IBM Z. IBM Power also reported less number of data breaches (one) in the same period compared to x86 platforms.
- As shown by the list of supported operating systems in Table 2-1 on page 22, IBM Power delivers the ability to run a wide variety of AIX, IBM i, or Linux workloads simultaneously, giving you flexibility in virtualization that is unmatched by any x86 offering.

Both IBM Power and x86 architectures are established, mature foundations for modern workloads, but Power stands out for its efficiency, deeply integrated virtualization, highly dependable availability and reliability, and its unparalleled scalability. This provides the ability to support enterprise-class workloads with significantly less infrastructure compared with what is required for running on x86 hardware.

2.5 PowerVM and virtualization

IBM PowerVM is the virtualization hypervisor that comes standard on IBM Power, IBM PowerVM provides support for virtual machines, enabling the creation of logical partitions (LPARs) and supports sharing of resources across multiple partitions. PowerVM is tightly integrated to the IBM Power hardware providing enterprise grade virtualization with minimal overhead compared to other virtualization technologies. PowerVM allows you to consolidate VMs running multiple workloads onto fewer systems, resulting in reduced costs, increased efficiency, better return on investment, faster deployment, workload security, and better server utilization.

PowerVM enables a Power server to have up to 1000 virtual machines on a single server running a mix of various operating systems and environments simultaneously. PowerVM also provides IBM Power other advanced features to help you manage and control your virtualized environment.

Micro-partitioning

Allows a partition/VM to initially occupy as small as 0.05 processing units, or 1/20 of a single processor core, and allows adjustments as small as a hundredth (0.001) of a processor core. This allows tremendous flexibility in the ability to adjust your resources according to the exact needs of your workload.

Shared Processor Pools

Allows for effective overall utilization of system resources by automatically applying only the required amount of processor resource needed by each partition. The hypervisor can automatically and continually adjust the amount of processing capacity allocated to each partition/VM based on system demand. You can set a shared processor partition so that, if a VM requires more processing capacity than its assigned number of processing units, the VM can use unused processing units from the shared processor pool.

Virtual I/O Server (VIOS)

VIOS provides the ability to share storage and network resources across several VMs simultaneously, thereby avoiding excessive costs by configuring the precise amount of hardware resources needed by the system.

Live Partition Mobility (LPM)

LPM brings the ability to move running VMs across different physical systems without disrupting the operating system and applications running within them.

Shared Storage Pools

The Shared Storage Pool (SSP) provides the ability to provide distributed storage resources to VIO servers in a cluster.

Dynamic LPAR operations (DLPAR)

Dynamic LPAR operations (DLPAR) Introduce the ability to dynamically allocate additional resources, such as available cores and memory, to a VM without stopping the application.

Performance and Capacity Monitoring

Supports gathering of important statistics to provide an administrator information regarding physical resource distribution among VMs and continuous monitoring of resource utilization levels, ensuring they are evenly distributed and optimally used.

Remote Restart

Allows for quick recovery in your environment by allowing you to restart a VM on a different physical server when an error causes an outage.

Note: For a more comprehensive description of PowerVM and its capabilities, see the IBM Redbooks publication *Introduction to IBM PowerVM*, SG24-8535.

2.6 PowerVC and virtual resource management

A powerful tool that you can use in managing the virtual resources on IBM Power is *IBM Power Virtualization Center*, or **PowerVC**. PowerVC is designed to simplify the management of virtual resources in your Power Systems environment.

With PowerVC you can register physical hosts, storage providers, and network resources which are then used to create a virtual environment. After the resources are registered, you can do the following tasks and more:

- Create virtual machines by deploying images and then resize and attach volumes to them.
- Import existing virtual machines and volumes so they can be managed by PowerVC.
- Create a project, add resources to the project, and assign users unique roles within each project.
- Monitor the utilization of the resources that are in your environment.
- Enable the IBM PowerVC Dynamic Resource Optimizer (DRO) to automatically rebalance host group resources when they are being overused.
- Migrate virtual machines while they are running (Live Partition Mobility).
- Capture a running virtual machine that is configured just the way you want it to be. Then, deploy that image elsewhere in your environment.
- Deploy images quickly to create new virtual machines that meet the demands of your ever-changing business needs.
- Enable PowerVC to automatically place virtual machines when you deploy or migrate them based on the criteria that you specify.

PowerVC provides support for larger enterprises with more complex system configurations. It enables you to maximize the virtualization capabilities of your IBM Power hardware with a powerful yet easy-to-use interface. Its key features include the following:

- Supports IBM Power Systems hosts that are managed by a Hardware Management Console.
- Supports storage area networks.
- Supports multiple Virtual I/O Server virtual machines on each host.
- Supports shared storage pools
- Supports host and storage templates to help you consistently and reliably deploy virtual machines across your environment.
- Supports storage connectivity groups, which enable you to deploy images so they have access to storage that is dedicated to a particular purpose. They also help to ensure that virtual machines have access to the same storage before and after they are migrated.

Note: For a more comprehensive description of PowerVC and its capabilities, you may refer to the IBM Redbooks publication *IBM PowerVC Version 2.0 Introduction and Configuration*, SG24-8477.

2.6.1 PowerVM Novalink

PowerVC can manage the virtual resources on IBM Power systems running PowerVM, whether they are managed by an HMC, or through PowerVM Novalink.

PowerVM NovaLink is a software interface that is used for virtualization management. You can install PowerVM NovaLink on a PowerVM server. PowerVM NovaLink enables highly scalable modern cloud management and deployment of critical enterprise workloads. You can use PowerVM NovaLink to provision large numbers of virtual machines on PowerVM servers quickly and at a reduced cost.

PowerVM NovaLink runs on a Linux logical partition on a POWER8, POWER9, or Power10 processor-based system that is virtualized by PowerVM. You can manage the server through a representational state transfer application programming interface (REST API) or through a command-line interface (CLI). You can also manage the server using PowerVC or other OpenStack solutions. Power VM NovaLink is available at no additional charge for servers that are virtualized by PowerVM. PowerVM NovaLink can be installed only on POWER8, POWER9, or Power10 processor-based systems.

Benefits of PowerVM NovaLink

PowerVM NovaLink provides the following benefits:

- Rapidly provisions large numbers of virtual machines on PowerVM servers.
- Simplifies the deployment of new systems. The PowerVM NovaLink installer creates a PowerVM NovaLink partition and Virtual I/O Server (VIOS) partitions on the server and installs operating systems and the PowerVM NovaLink software. The PowerVM NovaLink installer reduces the installation time and facilitates repeatable deployments.
- Reduces the complexity and increases the security of your server management infrastructure. PowerVM NovaLink provides a server management interface on the server. The server management network between PowerVM NovaLink and its virtual machines (VMs) is secure by design and is configured with minimal user intervention.
- Operates with PowerVC or other OpenStack solutions to manage your servers.

PowerVM NovaLink architecture

The PowerVM NovaLink software 2.0.2, or later is required to manage Power10 systems and the software runs only on Red Hat Enterprise Linux version 8.2 (or later) logical partitions on Power10 systems. The PowerVM NovaLink partition uses I/O resources that are virtualized by the Virtual I/O Server. The PowerVM NovaLink software is delivered by using the standard RPM Packet Manager (RPM) for Red Hat versions of PowerVM NovaLink, similar to any other software in the Linux operating system.

PowerVM NovaLink includes an installer that configures the PowerVM NovaLink environment in one action. The PowerVM NovaLink installer creates the Linux and Virtual I/O Server logical partitions and installs the operating systems and PowerVM NovaLink software.

The PowerVM NovaLink stack consists of the following services:

- PowerVM NovaLink Core Services provide direct interfaces to the managed system.
  - REST API that is similar to that on the Hardware Management Console (HMC) and a python-based software development kit.
  - Command-line interface (CLI) for shell interaction with PowerVM. This CLI differs from CLI of the HMC to provide a complete PowerVM CLI that encompasses both hypervisor and VIOS configurations.
- OpenStack Services provide drivers and plug-ins for use by OpenStack-based management solutions, including PowerVC:
- PowerVM virtualization driver for OpenStack Nova
- PowerVM Shared Ethernet Adapter agent for OpenStack Neutron
- PowerVM compute agent plug-ins for OpenStack Ceilometer

Note: For more information, see PowerVC NovaLink Overview and the IBM Redbooks publication *IBM PowerVC Version 2.0 Introduction and Configuration*, SG24-8477.

2.7 Power in the Cloud – Power Virtual Server

IBM Power Virtual Server (PowerVS) is an Infrastructure-as-a-Service (IaaS) IBM Power offering hosted on IBM data centers across the globe. PowerVS is colocated on IBM Cloud and delivers enterprise-class compute resources with the flexibility of hybrid cloud deployment. PowerVS can be used to deploy a virtual server, also called a logical partition (LPAR) or virtual machine (VM), in a matter of minutes. Now, your Power workloads that used to rely on your on-premises infrastructure can be deployed in the cloud quickly and economically.

Power servers can be isolated from other servers by using separate networks and direct-attached storage in the data centers. The internal networks are fenced but can be connected to IBM Cloud infrastructure or on-premises environments. This infrastructure design enables essential enterprise software certification and support as the PowerVS architecture is identical to certified on-premises infrastructure.

Power customers who are interested in modernization can benefit from deploying the workloads to PowerVS instead of moving their applications to a new platform that can be expensive and high risk. You can access many enterprise services from IBM with pay-as-you-use billing with which you can quickly scale up and out. PowerVS enables clients to take full advantage of this trend with the ability to provision AIX, IBM i, Linux or Red Hat OpenShift instances connected to the cloud.

3

Understanding Multi-Architecture Compute

Multi-Architecture Containerization is essential to modern software deployment strategies, particularly in environments with diverse hardware architectures. Multi-Arch Compute enables developers to create software that can be deployed seamlessly across different system architectures, such as x86_64, ARM64, and IBM Power Systems. This capability is crucial for ensuring that applications are portable, scalable, and efficient, regardless of the underlying hardware.

This chapter covers the following topics:

3.1 Benefits of multi-architecture containerization

Multi-Architecture Containerization is a software development and deployment approach that ensures applications run seamlessly across different hardware architectures. In today's globalized and diverse computing environment, this strategy is becoming increasingly important as systems operate on various platforms. Without multi-architecture support, each cluster supports only a single architecture, leading to a larger number of nodes, as control plane nodes have to be present on each architecture, and increased complexity as managing multiple clusters becomes mandatory. Let us explore why adopting multi-architecture containerization is crucial.

Hardware diversification

Enterprises expanding their operations often deploy applications in heterogeneous environments utilizing different types of hardware. A prime example includes data centers using x86_64 servers and Power servers, while edge locations might feature ARM-based devices due to their power efficiency. We discussed why IBM Power is an ideal choice for running Red Hat OpenShift clusters in Chapter 2, “IBM Power” on page 17. However, it is possible that some applications within the environment aren't compatible with IBM Power architecture nodes. You can retain the benefits of IBM Power while retaining simpler cluster management and efficient resource utilization by incorporating both IBM Power and x86_64 architecture nodes in the same cluster.

Multi-architecture capability offers several benefits:

- Platform independence: Multi-Arch Compute enables applications to function flawlessly across various hardware platforms, including Intel servers in data centers, ARM-based Raspberry Pis in remote locations, and IBM Power systems in corporate settings.
- Reduced complexity: Standardizing application deployment across architectures streamlines operations and eliminates the need to maintain separate stacks for different hardware.
- Cost efficiency: Differing hardware architectures provide varying cost-to-performance ratios, which can be exploited to minimize overall infrastructure expenses.

Multi-architecture support facilitates:

- Optimal resource usage: Organizations can select the most cost-effective architecture for each specific workload. For instance, ARM servers could be more affordable for lightweight services, whereas x86_64 or IBM Power servers might excel at handling heavy computational tasks.
- Energy efficiency: Specific architectures, like ARM, are renowned for their low power consumption, which can substantially decrease energy costs, notably in scale-out scenarios like IoT and mobile services.
- Scalability and flexibility: Implementing applications on multiple architectures enhances scalability and operational agility, which is vital for businesses experiencing fluctuating loads.

Multi-architecture support allows companies to:

  - Scale elastically across platforms: Multi-Arch Compute enables businesses to dynamically allocate resources among different types of hardware to handle surges in demand without being restricted to a single architecture.
  - Escape vendor lock-in: With Multi-Arch Compute, corporations are unshackled from a single supplier or type of hardware, thus avoiding vendor lock-in and empowering more bargaining power during procurement decisions.

- Optimize performance: Each architecture boasts unique strengths and weaknesses contingent upon the application or workload. Multi-Arch Compute maximizes performance by aligning application requirements with the architectural advantages.
- Create tailored solutions: Select applications might gain from the high I/O throughput of IBM Power systems, while others could perform optimally on the high-throughput, multi-core processors of x86_64 architectures.
- Meet specialized computing needs: Certain tasks may necessitate specialized hardware, such as GPUs for machine learning work flows, which might be more accessible or better supported on particular architectures.

Future proofing and innovation

Technological and business landscapes evolve rapidly, requiring organizations to prepare for upcoming shifts in hardware and computing paradigms. As novel hardware architectures emerge, Multi-Arch Compute guarantees that applications can be swiftly adopted to harness the newest technologies without substantial rework.

As companies navigate transitions between technologies or merge on-premises with cloud environments, Multi-Arch Compute accommodates seamless migration and hybrid deployment strategies.

Multi-architecture implementation can also aid in meeting regulatory and security requirements by offering the adaptability to situate processing and data storage in distinct geographic and jurisdictional environments. Applications can be deployed on localized architectures in specific countries to adhere to data residency laws.

In addition, diversifying the hardware landscape diminishes the risk of system-wide vulnerabilities impacting a single architecture type.

3.1.1 Multiple architecture use cases with IBM Power

There are many possible use cases for multiple architecture clusters. This publication will focus on the two use cases that involve IBM Power nodes:

- Adding Power worker nodes to an x86-based cluster. The control plane and some worker nodes are x86 architecture, while additional worker nodes are on Power servers.
- Adding x86 worker nodes to an IBM Power based cluster. In this case the control plane nodes are based on IBM servers with Power architecture worker nodes as well. A number of x86 architecture worker nodes are added to the cluster to support workloads that may not support Power architecture nodes.

An x86 cluster with Power worker nodes

Starting with Red Hat OpenShift 4.14 it was possible to add Power control nodes to an existing Intel-based cluster. This is shown in Figure 3-1 on page 36.

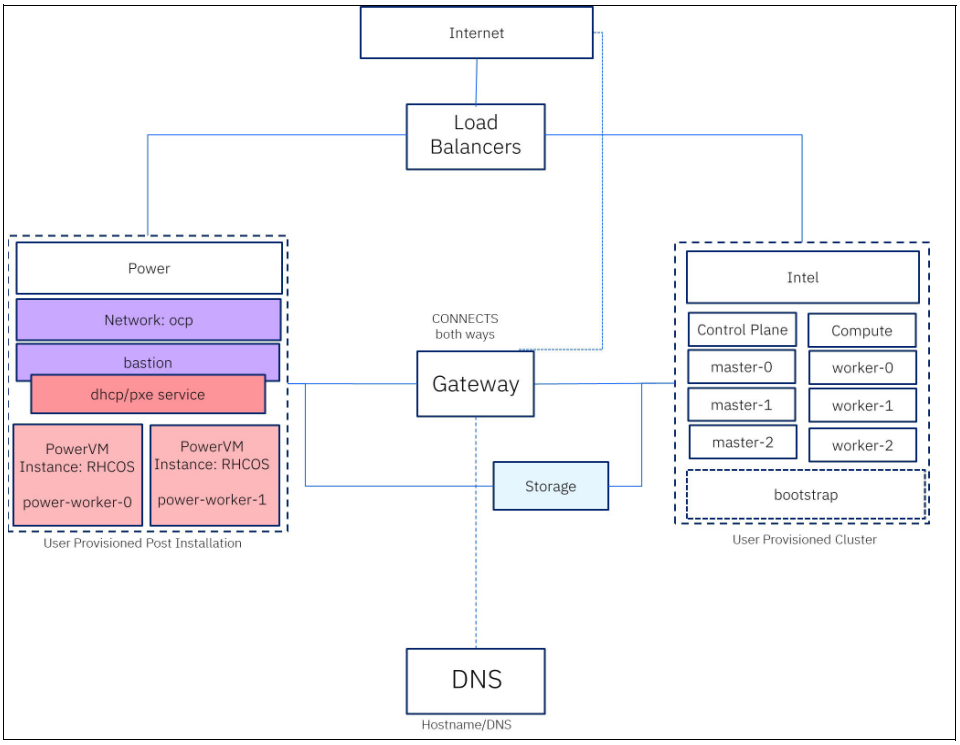


Figure 3-1 Intel cluster with Power worker nodes

Power cluster with x86 worker nodes

With the introduction of Red Hat OpenShift 4.15 it was possible to include x86 architecture worker nodes in a Power based cluster. This is shown in Figure 3-2.

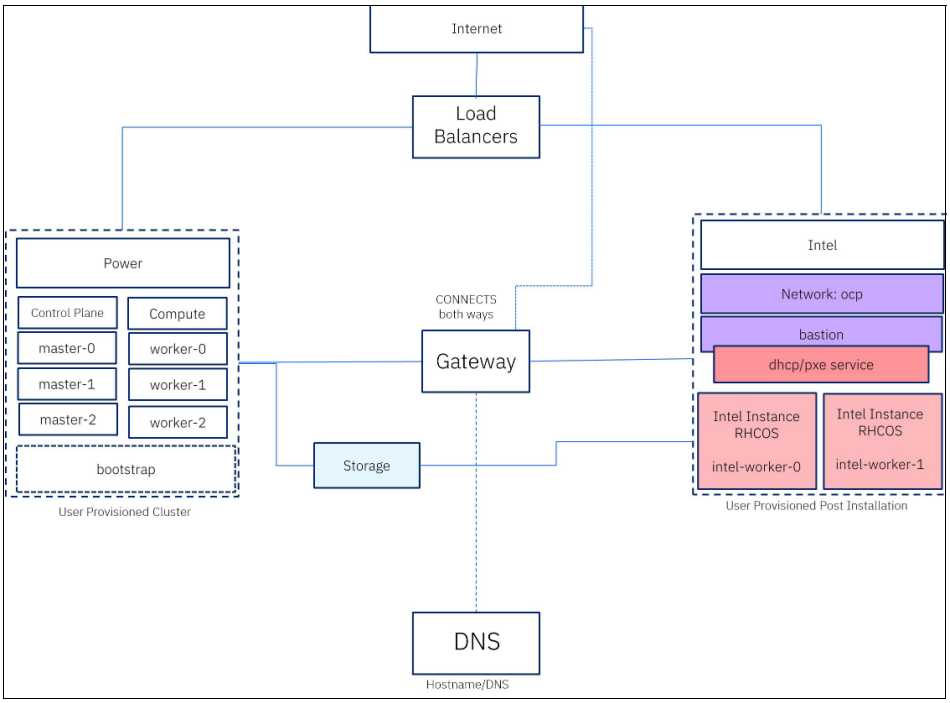


Figure 3-2 Power cluster with Intel/x86 worker nodes

3.2 Key concepts in multi-architecture containerization

Understanding the key concepts in multi-architecture containerization is essential for successfully implementing this strategy in a diverse hardware environment. These concepts are foundational to ensuring that applications are portable, efficient, and capable of running on different types of hardware architectures, such as x86_64, ARM, and IBM Power Systems. Here's a detailed look at these core concepts:

Container images for multiple architectures

In a multi-architecture environment, managing container images that are compatible with different hardware architectures is crucial. This involves creating and maintaining separate images for each target architecture; each optimized to make the best use of the specific hardware features.

Architecture-specific base images

Each architecture requires a specific base image that includes the necessary system libraries and other dependencies that are compatible with that architecture. For instance, an ARM-based image might start from arm64v8/ubuntu, whereas an x86_64 image might use amd64/ubuntu.

Manifest lists (multi-arch images)

Manifest lists serve as a cornerstone in streamlining image management across disparate architectures. They represent a single image tag pointing to multiple architecture-specific images. Upon pulling an image from a registry, the container runtime automatically selects the suitable version based on the host machine's architecture. This process occurs seamlessly without requiring input from the user.

Utilizing tools like Docker Buildx enables the generation of manifest lists by concurrently constructing images for multiple architectures and pushing them to a container registry under a single, unified tag.

Build and deployment automation

Streamlining automated build and deployment processes across multiple architectures is crucial for preserving efficiency and consistency within environments. Continuous Integration and Continuous Deployment (CI/CD) pipelines are commonly employed to handle these tasks. Build systems should provide cross-compilation capabilities, enabling source code to be compiled on a host machine of one architecture (for example, x86_64) to generate executables for another architecture (for example, ARM).

Automated build integrations within CI/CD pipelines can automatically trigger builds for all supported architectures upon detecting modifications to the codebase, guaranteeing access to the most recent code as a container image for any architecture.

Orchestration and Scheduling

Effective orchestration and scheduling tools, such as Red Hat OpenShift, must account for the architecture of the underlying nodes to properly schedule pods. This is achieved through labels and selectors. Each node in a Red Hat OpenShift cluster can be tagged with its architecture.

Pod specifications can incorporate node selectors or affinity rules that guide the orchestration system to allocate pods onto nodes of a particular architecture. This ensures that the container images, designed for specific architectures, operate on compatible hardware.

Optimization and Tuning

Applications may require customization based on the architecture to optimize hardware utilization fully. This can entail modifying memory allocation, CPU utilization, and the application's internal algorithms.

Regularly profiling applications on each target architecture helps identify bottlenecks and potential areas for enhancement. Architecture-Specific Tuning Adjustments might be required in configurations or code levels to improve performance on specific processor types or address memory bandwidth and latency concerns peculiar to certain architectures.

Security and Compliance

Maintaining security and compliance across multiple architectures demands consistent enforcement of security policies and timely updates across all environments. Establish a universal security baseline encompassing architecture-specific considerations where needed. Implement scanning tools supporting multiple architectures to verify images are devoid of known vulnerabilities prior to deployment.

3.3 Implementing Multi-Architecture containerization

Implementing Multi-Architecture Containerization involves several critical steps that ensure applications are effectively designed, built, and deployed to operate across various hardware architectures. This process requires meticulous planning, tool selection, and execution strategies. Here's a detailed exploration of how to implement Multi-Arch Compute in an organizational environment:

Setting Up the development environment

Before diving into containerization, ensure that the development environment supports multi-architecture development. This includes setting up cross-compilation tools, multi-arch build systems, and emulators or simulators for testing.

Use tools like GCC or Clang configured for cross-compilation to compile software for different architectures from a single source platform. Also, tools like QEMU can emulate different architectures, allowing developers to test builds on architectures they do not physically possess.

Designing architecture-agnostic applications

The application code should be designed to be architecture-agnostic as much as possible, minimizing the use of architecture-dependent code unless necessary.

Implement abstraction layers in your application to handle architecture-specific functionalities. This can help isolate parts of the code that are dependent on the architecture and reduce the impact on the main codebase. Use conditional compilation directives to include or exclude code segments based on the target architecture. This is useful for optimizing performance or utilizing architecture-specific features.

Building multi-architecture images

When building containers for multiple architectures, it is crucial to create multi-architecture images that include manifests specifying supported platforms. Tools like Buildah or Docker Buildx can help automate this process by enabling builds for different architectures from a single Dockerfile or BuildConfig. For instance, you can use Docker Buildx to build images for both x86_64 and ppc64 architectures simultaneously, ensuring compatibility across your OpenShift cluster.

Once built, deploying these multi-architecture containers on OpenShift involves tagging and pushing them to a container registry that OpenShift can access. OpenShift's native support for Kubernetes manifests ensures that these images can be deployed and managed seamlessly across nodes of different architectures within the cluster. This approach not only optimizes resource utilization but also facilitates the development of applications that are resilient to hardware changes or optimizations, making OpenShift a versatile platform for diverse computing environments.

Configure CI/CD pipelines to automatically build images for all supported architectures whenever code changes are pushed to the repository. This ensures that all architecture-specific images are always up to date. Figure 3-3 illustrates a pipeline concept that can be easily adapted to your environment.

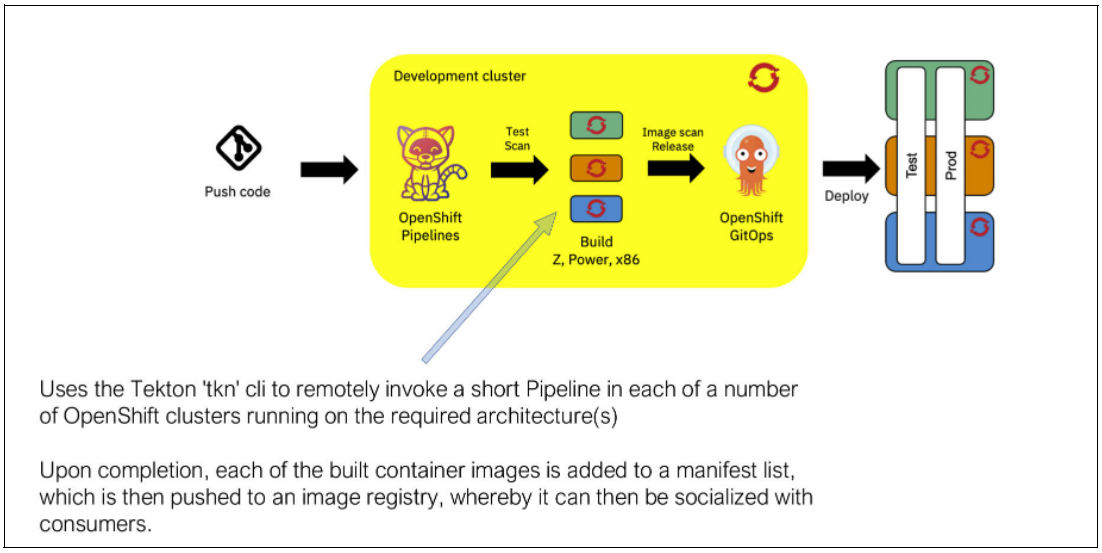


Figure 3-3 Red Hat OpenShift pipeline for building a multi architecture image

Figure 3-4 shows how the pipeline is built on Red Hat OpenShift using Tekton.

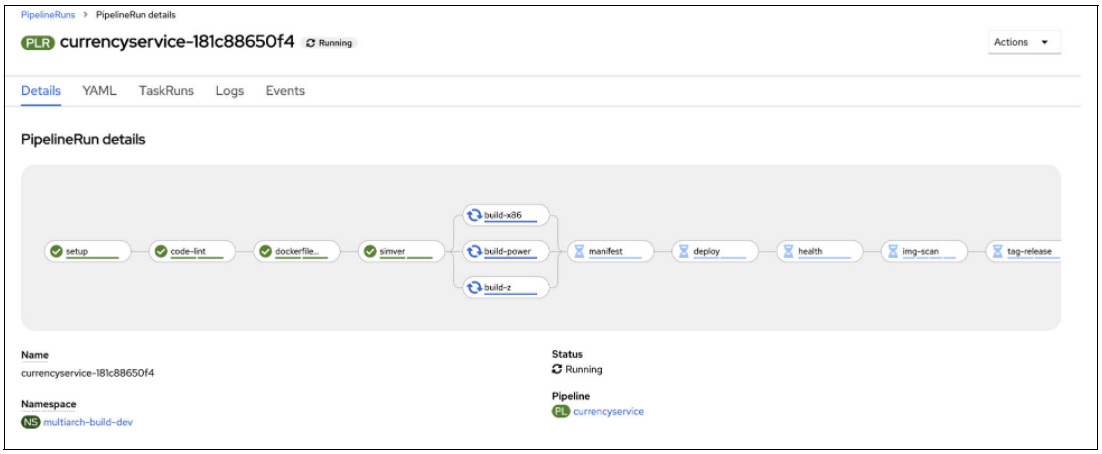


Figure 3-4 View of Tekton pipeline in Red Hat OpenShift for multi architecture image build

For an example of building a CI/CD pipeline using GitLab to build multi architecture images, see the IBM developer blog.

Configuring the container registry

Ensure your container registry can handle multi-architecture images. Modern registries like Docker Hub, Google Container Registry, and AWS ECR support multi-arch images via manifest lists.

Adopt a consistent tagging strategy that makes it easy to manage versions and architectures. Common practices include using tags that reflect the version, build, or commit hash and the architecture. Organize the registry to ensure that images are easy to find and manage. This may involve separating staging and production images or organizing images by application and architecture.

Orchestration

Modify the Red Hat OpenShift configuration to ensure it can intelligently schedule pods to the appropriate hardware based on the architecture. Label each node in the Red Hat OpenShift cluster with its architecture type. Use affinity and anti-affinity settings in your pod specifications to control pod scheduling. This helps ensure that pods are deployed onto nodes with compatible architectures.

Testing and monitoring

Implement a comprehensive testing strategy that covers all target architectures to ensure the application behaves consistently and performs well across all platforms. Extend your CI/CD pipelines to include tests running on all supported architectures using either emulation or actual hardware. Conduct performance tests to understand how the application performs on different architectures and identify any architecture-specific tuning that may be needed.

Once deployed, use monitoring tools to track the performance and health of your application across different architectures. Use tools to collect and display metrics from different architectures in a unified dashboard. Regularly review performance data to identify opportunities for optimizing code and resources specific to each architecture.

Implementing Multi-Arch Compute requires building a flexible and adaptable development environment that can respond to the evolving needs of diverse hardware platforms. By following these steps, organizations can ensure that their applications are portable across different architectures and optimized for performance and efficiency in each environment.

3.4 Challenges and solutions in multi-architecture containerization

Implementing Multi-Architecture Containerization poses a set of unique challenges due to the complexity of managing and deploying applications across diverse hardware architectures. Addressing these challenges effectively is crucial for organizations aiming to leverage the full potential of their hardware environments. Here is a detailed look at common challenges and practical solutions in Multi-Arch Compute.

Compatibility issues

Ensuring that applications run consistently across various architectures can be difficult, especially when dealing with differences in operating systems, system libraries, and hardware capabilities.

Solution:

- **Standardized environment**: Use containers to standardize the runtime environment across architectures, minimizing the differences seen by the application.

- **Conditional compilation**: Implement conditional compilation in your code to handle architecture-specific code branches and dependencies.
- **Extensive testing**: Establish a rigorous testing regime that includes unit, integration, and system tests on all target architectures. Emulation tools and CI/CD automation can facilitate this process by enabling testing even when physical hardware isn't readily available.

Build and deployment complexity

Managing build and deployment processes for multiple architectures can become complex, increasing the risk of errors and inconsistencies.

Solution:

- **Automated build systems**: Utilize tools like Docker Buildx to automate the creation of multi-architecture images from a single build process. Integrate these tools into CI/CD pipelines to ensure consistent and error-free builds.
- **Configuration management**: Use Infrastructure as Code (IaC) tools to reliably manage and replicate deployment configurations across different environments and architectures.

Performance optimization

Different architectures may have vastly different performance characteristics, and optimizing an application to run efficiently on one may adversely affect its performance on another.

Solution:

- **Architecture-specific tuning**: Identify critical performance bottlenecks for each architecture and apply architecture-specific optimizations. This might involve tweaking algorithm implementations or optimizing resource usage patterns.
- **Profiling and benchmarking**: Regularly profile the application on each architecture. Use profiling data to guide optimizations and ensure that changes benefit the overall performance across all architectures.

Resource management

Efficiently managing resources and ensuring optimal utilization across diverse architectures, especially in hybrid environments with cloud and on-premise solutions, can be challenging.

Solution:

- **Resource abstraction**: Implement abstraction layers that manage resources in a way that is agnostic to the underlying hardware. Kubernetes, for instance, abstracts resource allocation and scaling.
- **Dynamic resource allocation**: Use Kubernetes features like Horizontal Pod Autoscaling and Vertical Pod Autoscaling to adjust resources based on demand and performance metrics dynamically.

Security and compliance

Maintaining a consistent security posture and ensuring compliance across multiple architectures and deployment environments can be difficult due to varying security capabilities and requirements.

Solution:

- **Unified security framework**: Adopt a comprehensive security framework that applies uniformly across all architectures. Implement centralized logging, monitoring, and security policy enforcement.
- **Regular security audits**: Conduct regular security audits and vulnerability assessments for each architecture. Automate security scans as part of the CI/CD pipeline to catch issues early.

Maintenance and support

Continuous maintenance and support for applications across multiple architectures can strain resources and complicate operational processes.

Solution:

- **Documentation and training**: Maintain detailed documentation and provide training for development and operations teams on the nuances of multi-architecture support.
- **Dedicated teams**: Consider establishing architecture-specific teams that specialist in each hardware environment's unique operational and support challenges.

Addressing these challenges requires a strategic approach that combines the right tools, processes, and practices. By implementing these solutions, organizations can harness the benefits of Multi-Architecture Containerization, ensuring that their applications are robust, efficient, and capable of operating seamlessly across any hardware platform.

Multi-Architecture Containerization is a strategy that, while complex, offers significant advantages in a global, diverse hardware environment. By understanding and implementing Multi-Arch Compute, organizations can ensure their applications are robust, performant, and future-proof across various computing platforms.

Part 2

Getting Started With Red Hat OpenShift on Power

In Part 1, “Foundations” on page 1, we introduced you to Red Hat OpenShift and the new Multi Architecture capabilities that simplify the management of your Red Hat OpenShift environment on both x86 and IBM Power servers. We also introduced you to the IBM Power servers and showed why running Red Hat OpenShift on Power servers is a good choice for your hybrid cloud environment.

In this part we demonstrate how to set up a Red Hat OpenShift cluster on your IBM Power servers. The following topics are included:

- Chapter 4, “Setting up Your Environment” on page 45
- Chapter 5, “Installing Red Hat OpenShift on IBM Power Systems” on page 67

4

Setting up Your Environment

Mission-critical applications and their associated data often reside on IBM Power platforms to leverage their reliability, security, and performance benefits. When modernizing legacy applications, such as integrating AI to derive additional insights from data, placing both the application and its data on IBM Power systems can enhance performance.

However, business requirements may sometimes necessitate running certain applications on alternative hardware architectures, such as x86 or ARM. Previously, managing different architecture compute nodes required separate clusters for each architecture, which increased complexity and resource consumption.

To address this issue, Red Hat OpenShift introduced the Multi-Architecture Compute feature. This capability allows for the management of both x86 and Power worker nodes within a single cluster. It provides the flexibility to choose the underlying hardware architecture while simplifying cluster management by eliminating the need for multiple clusters.

Enabling Multi-Architecture Compute functionality is a post-installation step for Red Hat OpenShift clusters.

This chapter covers pre-installation considerations for setting up a new Red Hat OpenShift v4.15 cluster on IBM Power. The following topics are presented:

4.1 Setting up the environment

With Red Hat OpenShift 4.14, you can define IBM Power-based worker nodes within a Red Hat OpenShift cluster managed by x86 master nodes. With the release of Red Hat OpenShift 4.15, it is now possible to deploy the control plane (master nodes) on IBM Power while hosting the data plane (worker nodes) on both IBM Power and x86 systems. These mixed-architecture clusters exemplify Multi-Architecture Compute.

Choosing a Multi-Architecture Compute approach for a Red Hat OpenShift cluster depends on factors such as application requirements, available hardware, and other considerations. Existing Red Hat OpenShift clusters running earlier versions on IBM Power can be upgraded to Red Hat OpenShift 4.15 to leverage Multi-Architecture Compute capabilities.

4.2 Installing Red Hat OpenShift on IBM Power

The initial step in creating a Multi-Architecture Compute cluster on IBM Power is to install a Red Hat OpenShift cluster on IBM Power nodes. After setting up the cluster, you can add additional x86 or IBM Power-based worker nodes.

There are two primary methods for provisioning a Red Hat OpenShift cluster: Installer Provisioned Infrastructure (IPI) and User Provisioned Infrastructure (UPI). Below is a description of each method and considerations for choosing between them.

4.2.1 Installer Provisioned Infrastructure (IPI)

IPI installations are generally simpler than UPI installations because the IPI installer includes built-in logic to provision all required components. Currently, the only environment on IBM Power that supports IPI is IBM Power Virtual Server (PowerVS). Other on-premises implementations on IBM Power require UPI.

For Power Virtual Servers, the IPI process handles the provisioning of:

- Power compute nodes (LPARs)
- Networking
- Load Balancers
- Access Policies and Service IDs
- DNS records

For more information, see IBM Power Systems Virtual Server now offering Red Hat OpenShift Container Platform.

Upon completing the installation, you will have a fully functional Red Hat OpenShift cluster. You can then enable Multi-Architecture support and add x86 worker nodes, as described in section 5.2.5, “Adding an x86 compute node to an IBM Power Red Hat OpenShift cluster” on page 86.

4.2.2 User provisioned infrastructure

User Provisioned Infrastructure (UPI) involves a more customized deployment process where users are responsible for defining and setting up the infrastructure required for a Red Hat OpenShift cluster before initiating the deployment.

UPI is often preferred for production environments due to the need for customization that aligns with an enterprise's specific policies and requirements.

Connected or Air-Gapped Environments

The Red Hat OpenShift installer supports both connected and air-gapped installations. A connected environment has direct internet access and can retrieve installation images directly from online sources. In contrast, an air-gapped environment lacks direct internet connectivity. In such cases, a private registry is needed to mirror Red Hat OpenShift images for installation, as these environments typically do not have internet access.

Installing Prerequisites and Dependencies

For clusters utilizing User Provisioned Infrastructure, all necessary machines must be deployed prior to installing Red Hat OpenShift. The following outlines the requirements for setting up Red Hat OpenShift Container Platform on a user-provisioned infrastructure.

Required Machines for Cluster Installation

The smallest Red Hat OpenShift Container Platform clusters require the hosts shown in Table 4-1.

Table 4-1 Minimum required hosts

Hosts	Description
One temporary bootstrap machine	The cluster requires the bootstrap machine to deploy the Red Hat OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.
Three control plane machines	The control plane machines run the Kubernetes and Red Hat OpenShift Container Platform services that form the control plane
At least two compute machines, which are also known as worker machines	The workloads requested by Red Hat OpenShift Container Platform users run on the compute machine

Note: To maintain high availability of your cluster, use separate physical hosts for these cluster machines. The bootstrap, control plane, and compute machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system. RHCOS is based on Red Hat Enterprise Linux (RHEL) 9.2 and inherits all of its hardware certifications and requirements.

Minimum resource requirements for cluster installation

Each cluster machine must meet the minimum requirements shown in Table 4-2.

Table 4-2 Minimum resource requirements

Machine	Operating System	vCPU[a]	Virtual RAM	Storage	Input/Output Per Second (IOPS)[b]
Bootstrap	RHCOS	2	16 GB	100 GB	300
Control Plane	RHCOS	2	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300

a. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.
b. Red Hat OpenShift Container Platform and Kubernetes are sensitive to disk performance and faster storage is recommended, particularly for etcd on the control plane nodes. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

Minimum IBM Power requirements

Red Hat OpenShift Container Platform 4.15 can be installed on the IBM Power 9 or IBM Power 10 processor-based systems.

Important: Support for RHCOS functionality for all IBM Power 8 models, IBM Power AC922, IBM Power IC922, and IBM Power LC922 is deprecated in Red Hat OpenShift Container Platform 4.15. Red Hat recommends that you use later hardware models.

The following is the recommended IBM Power system configuration:

- Six LPARs across multiple PowerVM servers
- One instance of an IBM Power 9 or IBM Power 10 processor-based system

On your IBM Power instance, set up:

- Three LPARs for Red Hat OpenShift Container Platform control plane machines
- Two LPARs for Red Hat OpenShift Container Platform compute machines
- One LPAR for the temporary Red Hat OpenShift Container Platform bootstrap machine

Disk storage for the IBM Power guest virtual machines

Storage can be one of the following:

- local storage
- storage provisioned by the Virtual I/O Server using:
  - vSCSI
  - NPIV (N-Port ID Virtualization)
  - SSP (shared storage pools)

Network for the PowerVM guest virtual machines

Networking can be:

- Dedicated physical adapter
- SR-IOV virtual function
- Virtualized by the Virtual I/O Server using Shared Ethernet Adapter
- Virtualized by the Virtual I/O Server using IBM vNIC

Storage / main memory

The recommended storage and memory configuration is:

- 120 GB / 32 GB for Red Hat OpenShift Container Platform control plane machines
- 120 GB / 32 GB for Red Hat OpenShift Container Platform compute machines
- 120 GB / 16 GB for the temporary Red Hat OpenShift Container Platform bootstrap machine

Certificate signing requests management

In UPI, the cluster has limited access to automatic machine management for signing certificate requests post installation. The *kube-controller-manager* only approves the kubelet client certificate signing requests (CSRs).

The machine-approver cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

4.3 Network and storage design in Red Hat OpenShift

Setting up network and storage resources is a fundamental step in preparing an environment for Red Hat OpenShift – especially when deploying on different hardware architectures such as IBM Power Systems. Effective configuration of these resources ensures that applications perform optimally, are resilient, and scale effectively.

Red Hat OpenShift network design considerations

Networking is an essential component within a Red Hat OpenShift environment. It is essential that users of the cluster resources be able to connect to the appropriate services and even more important is that the security of the cluster is maintained - especially in multi-tenant environments. The underlying basis for networking in the Red Hat OpenShift environment is a software-defined network.

Software Defined Networking

Red Hat OpenShift employs a software-defined networking (SDN) approach to create an efficient and scalable network that manages communication between pods across the cluster. This SDN setup abstracts the underlying network infrastructure to provide advanced networking features.

Within the SDN, Red Hat OpenShift uses overlay networks to allow pods to communicate across different nodes. This involves encapsulating packet data using network protocols such as VXLAN, which helps in creating a virtual network over the physical network. This setup is crucial for pod-to-pod communication across nodes without requiring complex physical network configurations.

The use of SDN in Red Hat OpenShift simplifies network management and enhances security by providing network isolation. It also improves scalability. This is all done through decoupling the network configuration from the underlying physical infrastructure.

Network policies

Network policies in Red Hat OpenShift are used to control the flow of traffic in and out of containers and services. Network policies are Kubernetes resources that define rules for ingress and egress traffic – specifying which traffic is allowed to enter and exit the pods. Network policies are the base for networking security within a Red Hat OpenShift environment.

Administrators can define fine-grained network policies that specify which pods can communicate with each other and with other network endpoints. These policies are essential for enforcing security rules and maintaining compliance within the cluster. Network policies are particularly useful in multi-tenant environments, where strict network isolation between different organizational units or projects is required.

Load balancing

Load balancing in Red Hat OpenShift aims to distribute incoming traffic evenly among multiple pods, thereby ensuring high availability and fault tolerance of applications. This is crucial for maintaining service performance and reliability, particularly during peak loads.

Red Hat OpenShift supports integration with both external and internal load balancers. Cloud platform providers like Amazon Web Services (AWS) Elastic Load Balancing (ELB), Microsoft Azure Load Balancer, or hardware load balancers can manage incoming internet traffic. internally, Red Hat OpenShift uses HAProxy or software-defined load balancers to disperse internal traffic among pods.

Configuring load balancing entails creating service objects within Kubernetes defining how to interact with the applications. These services can then be exposed through various methods, including NodePort, LoadBalancer, or Cluster IP, according to specific requirements.

Ingress control

Ingress controllers in Red Hat OpenShift manage access to services from outside the Kubernetes cluster. They act as a reverse proxy and traffic router, directing client requests to the appropriate backend services.

Ingress can also handle SSL/TLS termination, offloading the cryptographic workload from backend pods. This setup improves security by enabling encryption of data in transit and simplifies certificate management.

Ingress rules can be customized to include URL rewriting, host-based routing, and load balancing methods, providing flexibility in how applications are exposed and managed.

DNS and routing

The ability to find services within the cluster and services that the cluster needs to connect to requires that IP addresses be associated with a service name through the use of Domain Name Services (DNS). As services are added or removed, the DNS must be updated appropriately so that users and services are connected and the traffic is routed efficiently.

Internal DNS

Red Hat OpenShift automatically assigns DNS names to services and pods, facilitating easy service discovery within the cluster. This internal DNS is managed by CoreDNS or a similar tool, which resolves service names to the appropriate internal IP addresses.

Automatic DNS configuration simplifies the deployment and scaling of applications by allowing services to communicate with each other using easily recognizable names rather than IP addresses.

External DNS integration

For services that need to be accessible externally, integration with external DNS providers is crucial. This ensures that DNS entries are automatically updated when services are exposed externally.

Red Hat OpenShift can integrate with systems like ExternalDNS to automatically update DNS records in external DNS providers whenever services are exposed or scaled. This is especially important in dynamic environments where IP addresses can frequently change.

Setting up external DNS integration involves configuring credentials and permissions for Red Hat OpenShift to interact with external DNS services, and defining the DNS zones and records that should be managed.

Storage configuration in Red Hat OpenShift

Data resources within a Red Hat OpenShift cluster are by default ephemeral, meaning that when a node or pod is taken down, the data stored is not kept. To maintain data persistence in the cluster, utilize persistent storage mechanisms.

Red Hat OpenShift leverages the Kubernetes persistent volume (PV) framework to enable cluster administrators to create and manage persistent volumes for the cluster. Users can submit persistent volume claims (PVCs) to request PV resources without needing explicit knowledge of the underlying storage infrastructure.

PV resources are not confined to individual projects; instead, they can be accessed by every project in the OpenShift platform. Conversely, PVCs are exclusive to a particular project and serve as a method for users to employ a PV. Once a PV is associated with a PVC, it cannot subsequently bind to extra PVCs. Consequently, a bound PV pertains solely to the namespace of the binding project.

Persistent Volumes are represented by a Persistent Volume API object, which signifies pre-existing storage in the cluster that was manually provisioned by the cluster admin or automatically provisioned utilizing a Storage Class object. A Persistent Volume acts similarly to a node in the cluster—both represent resources.

Storage classes

Storage Classes in Red Hat OpenShift facilitate the classification of storage based on factors like performance traits, underlying tech, and pricing. By employing these classes, administrators can group storage into categories such as solid-state drives (SSDs) for high-performance necessities and hard disk drives (HDDs) for budgetary, mass storage demands.

While configuring a storage class, administrators outline parameters like storage type, input/output operations per second (IOPS), and redundancy settings. Red Hat OpenShift integrates seamlessly with various storage vendors, including NFS, iSCSI, Fibre Channel, cloud-based offerings like Amazon Elastic Block Store (EBS), Google Persistent Disks, or Microsoft Azure Disk Storage.

For instance, a database requiring high IOPS may benefit from a storage class linked to SSDs, whereas a file storage app demanding extensive capacity yet lower performance could utilize a class connected to HDDs.

Dynamic provisioning

Dynamic provisioning in Red Hat OpenShift uses the concept of Persistent Volume Claims (PVCs) to automatically provision storage as it is needed without manual administrator intervention. This system decouples the storage needs from the underlying storage environment, providing a more flexible and efficient way to manage storage resources.

When a PVC is created, it requests a specific amount of storage and a storage class. The cluster then dynamically provisions a Persistent Volume (PV) that meets these specifications, ensuring that applications always have the storage they require without needing to manually pre-provision storage.

This approach significantly speeds up the deployment process and ensures better utilization of storage resources, reducing waste and potentially lowering costs.

Data redundancy and high availability

An important consideration for persistent storage defined within a Red Hat OpenShift cluster is data redundancy and availability. Data redundancy and high availability (HA) needs to be planned for when building the storage resources for the Red Hat OpenShift cluster.

Replication

Replication of storage volumes is critical for ensuring data durability and availability, particularly for stateful applications that cannot afford data loss, such as databases or file storage systems.

This can be achieved through software-defined storage solutions that are integrated into Red Hat OpenShift, or through external storage systems that provide native support for replication.

Administrators have the ability to configure replication settings at the storage class level, which includes selecting the preferred replication factor and defining data locality policies (e.g., replicating data across various physical sites or data centers). This approach aims to strike a balance between ensuring high availability and maintaining optimal performance levels.

Backup and recovery

Implementing thorough backup and recovery protocols is crucial to safeguard data from potential losses caused by equipment malfunctions, user mistakes, or unforeseen incidents. These measures encompass frequent data backups and guarantee swift and efficient restoration capabilities.

Red Hat OpenShift facilitates integration with diverse backup systems, enabling automated backup of persistent volumes and application data. Solutions such as Velero (previously known as Ark) empower users to back up Kubernetes resources and persistent volumes efficiently.

Recommended best practices for backup involve scheduling periodic backups, maintaining duplicate copies in distinct geographical locations to prevent regional failures from impacting both primary and backup datasets, and periodically validating recovery workflows to confirm their flawless performance.

4.3.1 Networking in Red Hat OpenShift on Power Systems

Deploying Red Hat OpenShift on IBM Power Systems involves tailoring the networking infrastructure to fully leverage the robust capabilities and unique architecture of Power Systems. This entails optimizing network performance for high throughput and low latency, ensuring network security and compliance, and managing network configurations to meet the specific demands of enterprise-level deployments.

Optimizing network performance

Power Systems are built with flexibility and performance in mind and there are many options available, both in the physical hardware choices and specific tuning options that ensure that the networking capabilities of the underlying Power infrastructure components provide the performance that meets the needs of your applications. Consider and plan for the following:

1. High-Bandwidth Networking

   Building an efficient network infrastructure necessitates employing the best-suited adapters and features. To achieve optimal networking performance, we recommend:

   – **Using advanced network adapters**: (NICs), engineered to manage substantial datasets and rapid communication requirements efficiently. By integrating these high-performance NICs, businesses can optimize data transfer rates and reduce latency—essential for applications demanding quick data retrieval and instant processing. Adapter capabilities such as hardware acceleration and sophisticated queue management enhance total data workflow efficiency.

- **Using Virtual NICs (vNIC)**: Utilizing virtual NICs in Power Systems allows for flexible and efficient network traffic management. vNICs can be dynamically configured to meet the changing demands of applications and workloads, allowing better resource allocation and network isolation. This setup supports scaling network resources up or down as needed without disrupting underlying physical configurations.

2. Network tuning

   Tuning your network effectively can significantly impact user satisfaction, with a satisfactory experience being more likely compared to an unsatisfactory one. To achieve this, we recommend employing the following strategies:

   - **Jumbo frames**: Configuring jumbo frames on Power Systems can significantly enhance network performance, especially for data-intensive applications. Jumbo frames allow more data to be packed into a single packet, reducing overhead and increasing the efficiency of data transmission across the network. This is particularly beneficial in environments where large datasets are transferred frequently, such as in big data analytics and video streaming applications.
   - **TCP/IP stack tuning**: Adjusting the TCP/IP stack settings can optimize the handling of network traffic on Power Systems. This includes tuning parameters like the TCP window size, buffer sizes, and the number of allowable connections. These adjustments help accommodate larger volumes of network traffic and improve the resilience and responsiveness of network communications.

Security and compliance

Power Systems provide a high level of security capabilities which can keep your infrastructure secure and allow you to meet regulatory compliance requirements. Consider and plan for the following:

1. Network Isolation

   Isolating network traffic is a fundamental aspect of creating secure networks, particularly in multi-tenant environments. By doing so, individual tenants' data and communications remain confidential and protected from unauthorized access.

   - **VLANs and VXLANs**: VLANs (Virtual Local Area Networks) and VXLANs (Virtual Extensible LAN) are technologies used to create logically segmented networks within the same physical network infrastructure. In Red Hat OpenShift on Power Systems, using VLANs and VXLANs helps in creating isolated networks for different applications or tenants, enhancing security by ensuring that network traffic is segregated and cannot cross over into other segments unless explicitly permitted. This is crucial for maintaining the integrity and confidentiality of sensitive or critical data.

2. Regulatory Compliance

   Ensuring compliance with security regulations is vital, as it helps maintain user trust and protect your data. The use of encryption for data in motion is critical. Here's what we suggest:

   - **Encrypted transmissions**: To meet regulatory compliance standards and protect data privacy, it is essential to ensure that all data transmitted over the network is encrypted. This includes not only data at rest but also data in transit. Encrypting network transmissions prevents unauthorized access and data breaches, and is a requirement in many industries, particularly those handling sensitive information like finance, health care, and public services. Encryption techniques such as TLS (Transport Layer Security) and IPSec (Internet Protocol Security) can be implemented to secure data as it moves across the network.

4.3.2 Integration with existing infrastructure

The integration of Red Hat OpenShift running on IBM Power servers with existing infrastructure involves strategic networking solutions that bridge on-premises systems with your new cloud environment. The integration of these systems enables organizations to leverage the strengths of both infrastructures for enhanced flexibility, scalability, and resilience. The following lists some considerations and technologies for integrating Red Hat OpenShift on Power into your existing infrastructure:

1. Hybrid cloud connectivity

   Hybrid cloud connectivity involves establishing robust communication links between on-premises data centers and public clouds. This setup allows organizations to keep some data and applications on-premises for security and compliance reasons while leveraging the scalability and advanced services available in the cloud. Hybrid environments are particularly useful for disaster recovery, global application deployment, and data locality.

   There are several technologies that can provide the needed connectivity, such as:

   - **VPN (Virtual Private Network)**: A VPN can be used to create a secure and encrypted tunnel between the on-premises network and the cloud. This is suitable for medium-scale integration where data transmission security is a priority.
   - **Direct connect solutions**: Services like AWS Direct Connect, Azure ExpressRoute, or Google Cloud Interconnect provide dedicated network connections between on-premises infrastructures and cloud data centers. These connections offer higher bandwidth and lower latency than typical internet-based connections, making them ideal for high-volume, mission-critical applications.
   - **API management**: Implement API gateways to manage and secure communications between on-premises applications and cloud services. This includes handling authentication, rate limiting, and data caching to optimize network usage and security.

2. Multiprotocol Label Switching (MPLS)

   MPLS is a flexible, high-performance routing technique that directs data from one network node to the next based on short path labels rather than long network addresses. Unlike traditional IP routing, where each data packet has to be inspected at each router, MPLS assigns a label to each data packet which allows routers to forward the data without extensive analysis. This can reduce the complexity and increase the speed of data flows across a distributed network. MPLS can provide the following functions for your networking environment:

   - **Traffic engineering**: MPLS can be used for traffic engineering by setting the path that traffic will follow across the network. This is particularly useful in hybrid environments where control over traffic routes can significantly improve performance and reduce congestion.
   - **Quality of Service (QoS)**: MPLS supports QoS by allowing network operators to differentiate traffic based on labels. This means that critical business applications can be prioritized over less critical traffic, ensuring consistent performance levels.
   - **Disaster recovery and business continuity**: By using MPLS, organizations can reroute traffic around failed links and nodes automatically and quickly, which is crucial for maintaining business operations during network failures or other disruptions.

Challenges and considerations

Consider the following when deciding on a networking solution for your hybrid cluster environment:

- **Complexity and cost**: Both VPN and MPLS solutions add complexity to network management and can involve significant costs, especially for high-bandwidth direct connect solutions or extensive MPLS networks.
- **Security and compliance**: Ensuring security in a hybrid environment requires careful configuration of encryption, network security policies, and consistent compliance practices across both on-premises and cloud components.
- **Latency and performance optimization**: When designing hybrid networks, attention must be paid to minimize latency and optimizing performance, particularly when integrating applications that are sensitive to delays, such as real-time data analytics or customer-facing applications.

Integration of Red Hat OpenShift on IBM Power Systems with existing infrastructure using hybrid networking strategies provides organizations the ability to extend their capabilities and achieve more flexible, scalable, and resilient IT operations. This approach is essential for businesses looking to innovate while still maintaining control over critical IT resources.

4.3.3 Storage options

There are multiple options available to provide persistent storage to your Red Hat OpenShift cluster. Depending on your requirements you can use attached block storage – which generally provides higher performance – or you can use object storage for storage of unstructured data.

ODF

Red Hat OpenShift Data Foundation (ODF) is a solution that integrates various open-source technologies to simplify storage management within Kubernetes clusters. It offers built-in services for block, object, and file system storage, catering to all persistent storage requirements. In essence, ODF serves as a comprehensive platform for managing storage resources in a streamlined manner.

The key components of ODF include:

- Ceph Data Storage Platform: This is an open-source storage platform that uses software-defined storage with various levels of redundancy based on object storage. It consists of three main layers: the object storage layer called RADOS, which uses the CRUSH algorithm to distribute storage objects across nodes, forming the Ceph Storage Cluster.
- Classic Block Devices (also known as RBD or RADOS Block Device): These provide local disk-like performance for applications requiring low-latency access to their stored data. They operate similarly to iSCSI initiators connecting to Fibre Channel disks.
- Ceph File System (or CephFS): This is a POSIX-compliant file system built on top of the Ceph Storage Cluster. It allows users to mount a traditional file system hierarchy onto the Ceph cluster, providing compatibility with existing tools and workflows.
- Object Storage Through S3 Compatible and Swift APIs: Ceph supports storing and retrieving data via industry-standard protocols such as Amazon S3 and OpenStack Swift. This enables seamless integration with cloud services and applications designed for those platforms.

Additionally, two projects complement ODF:

- Rook Kubernetes Operator: Rook is a CNCF-backed open-source project designed to manage Ceph components within Kubernetes clusters. By integrating Ceph with Kubernetes, Rook simplifies the deployment, scaling, and maintenance of distributed storage systems.

- NooBaa Multi-Cloud Object Storage Gateway: NooBaa is a multi-cloud object storage gateway that provides a unified interface for managing object storage across different cloud providers. It helps organizations leverage the benefits of object storage while maintaining data consistency and security across various environments.

These components together provide a set of storage classes to enable block, file storage and object storage API endpoints for the Red Hat OpenShift cluster.

Red Hat OpenShift Data Foundation supports deployment on existing Red Hat OpenShift clusters running on IBM Power in connected or disconnected environments and includes out-of the box support for proxy environments. See Planning your deployment and Preparing to deploy Red Hat OpenShift Data Foundation for more information about deployment requirements.

The Red Hat OpenShift Data Foundation on IBM Power Systems is compatible with two on-premises cloud setups. One configuration is built upon IBM PowerVC, while the other relies on IBM PowerVM along with the IBM Power Hardware Management Console (HMC). Additionally, there's a public cloud option, which utilizes IBM Power Systems Virtual Servers available within the IBM Cloud environment.

Deploying ODF

Red Hat OpenShift Data Foundation supports deployment into Red Hat OpenShift Container Platform clusters deployed on installer-provisioned or user-provisioned infrastructure. You have two options for deploying Red Hat OpenShift Data Foundation:

1. Deploy it wholly inside OpenShift Container Platform, which we refer to as the Internal approach.
2. Alternatively, you can expose the services from a separate cluster located externally, allowing them to be utilized without being fully integrated into OpenShift Container Platform. We call this the External approach.

To install Red Hat OpenShift Data Foundation within a cluster requires certain minimum requirements:

- The cluster must consist of at least three Red Hat OpenShift worker nodes in the cluster with locally attached storage devices on each of them.
  - Each of the three selected nodes must have at least one raw block device available to be used by Red Hat OpenShift Data Foundation.
- The devices to be used must be empty, that is, there should be no persistent volumes (PVs), volume groups (VGs), or local volumes (LVs) remaining on the disks.
- You must have a minimum of three labeled worker nodes.
  - Each node that has local storage devices to be used by Red Hat OpenShift Data Foundation must have a specific label to deploy Red Hat OpenShift Data Foundation pods.

To ensure a smooth Red Hat ODF deployment and operation, it is essential to have proficiency in several key areas:

- Familiarity with Red Hat OpenShift Container Platform deployed on IBM Power architecture.
- Knowledge of storage infrastructure within Kubernetes/Red Hat OpenShift, encompassing Ceph and Rook components.
- Understanding of the hierarchical organization of storage classes, persistent volumes, and persistent volume claims within the Kubernetes/Red Hat OpenShift framework.
- Experience integrating Red Hat OpenShift Cluster Block Storage and File Storage into applications.

- Ability to generate YAML files for creating persistent volume claims utilizing designated storage classes and deploying basic application pods employing an NGINX image with a persistent volume claim attached.
- Proficiency in working with the ceph-tools pod to manage and interact with embedded Ceph within Red Hat OpenShift Container Platform.

With these understandings you will be able to create a persistent volume claim from a user-provided YAML file and deploy a simple application pod using NGINX that mounts that persistent volume.

After you have addressed the above, perform the following steps:

1. Install Local Storage Operator.
2. Install the Red Hat OpenShift Data Foundation Operator.
3. Find available storage devices.
4. Create a Red Hat OpenShift Data Foundation cluster on IBM Power.

Detailed implementation instructions can be found in the Red Hat documentation.

Optional encryption enablement

If you want to enable cluster-wide encryption using the external Key Management System (KMS) follow these steps:

- Ensure that you have a valid Red Hat OpenShift Data Foundation Advanced subscription. To know how subscriptions for Red Hat OpenShift Data Foundation work, see this knowledgebase article on Red Hat OpenShift Data Foundation subscriptions (Red Hat login required).
- When the Token authentication method is selected for encryption then refer to Enabling cluster-wide encryption with the Token authentication using KMS.
- When the Kubernetes authentication method is selected for encryption then refer to Enabling cluster-wide encryption with the Kubernetes authentication using KMS.
- Ensure that you are using signed certificates on your Vault servers.

IBM Storage Scale

IBM Storage Scale is the IBM strategic high-performance parallel file system. IBM Storage Scale provides a shared storage platform for an end-to-end collaborative common enterprise data platform. The solution is architected to manage up to exabyte scale capacity with various configurations suitable for analytics, big data, artificial intelligence, and enterprise data lake workloads. An overview of IBM Storage Scale is Shown in Figure 4-1 on page 58.

IBM Spectrum Scale in containers (IBM Spectrum Scale Container Native Storage Access) allows the deployment of the cluster file system in a Red Hat OpenShift cluster. Using a remote mount-attached IBM Spectrum Scale file system, the IBM Spectrum Scale solution provides a persistent data store to be accessed by the applications via the IBM Spectrum Scale Container Storage Interface (CSI) driver using persistent volumes (PVs).

IBM Storage Scale is designed to provide the following major value propositions.

- Simplified data management by supporting enterprise workflows on a single common enterprise data platform
- A single global namespace that supports enterprise-level data over high-performance networks
- Enables intelligent automatic tiering of data between storage pools, externally to tape, to object-based and cloud resources. This delivers cost-effective storage economics by automatically managing and tiering data to different classes of storage

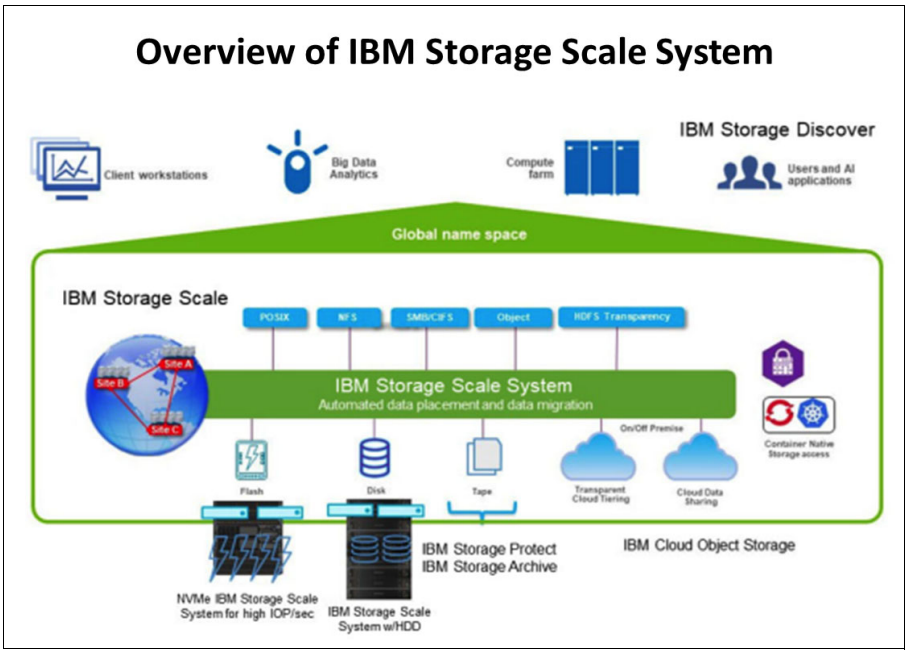


Figure 4-1 IBM Storage Scale architecture overview

Although multiple types of IBM Storage Scale cluster configurations are available, the configuration into which IBM Storage Scale System is commonly deployed is the IBM Storage Scale Network Shared Disk (NSD) configuration, as shown in Figure 4-2.

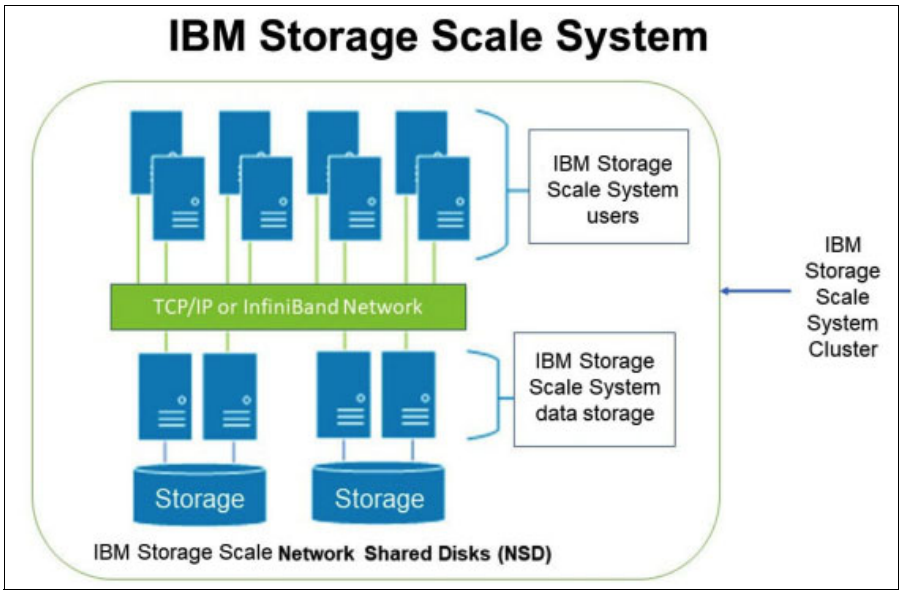


Figure 4-2 Network Shared Disk configuration of IBM Storage Scale

The IBM Storage Scale System shown in Figure 4-2 contains a pair of IBM Storage Scale NSD Data Servers, which are configured together as a tested, integrated, highly available, and reliable IBM Storage Scale storage building-block-based solution. As shown in the example IBM Storage Scale cluster, eight IBM Storage Scale nodes are application servers, and four nodes are IBM Storage Scale data server nodes. The IBM Storage Scale client achieves high performance by performing simultaneous real-time parallel I/O to all IBM Storage Scale data servers, storage volumes, and NSDs.

The cluster can scale in capacity by adding additional storage capacity (NSDs) behind the existing NSD servers. If additional performance or throughput is required, additional NSD servers can be added to the cluster.

An IBM Storage Scale cluster can provide 1 to 256 logical POSIX file systems to users and workstations. The IBM Storage Scale client provides the appearance of a mountable POSIX file system to the applications and users on the workstation where it is installed. IBM Storage Scale users are unaware of the physical distribution of data in the IBM Storage Scale data server storage pools. The automatically balanced data distribution is seamlessly determined by the IBM Storage Scale policy engine at the time the data is imported. The policy engine can also transparently move data from one storage pool to another storage pool while the data is accessed and active.

The IBM Storage Scale parallel file system provides enterprises with the capability for data management over large amounts of data while also performing constant auto-balancing of workload and storage by equally distributing I/O and data within a storage pool or among different storage pools.

The preferred method of accessing IBM Storage Scale data is to install the IBM Storage Scale client on every workstation or server that needs to access it. The IBM Storage Scale client provides the multiple threads and communication with multiple data servers to provide high-performance parallel throughput. IBM Storage Scale also manages full read/write data integrity between multiple users who are working with the data in the file system.

As an optional part of the IBM Storage Scale solution, protocol nodes allow access to IBM Storage Scale data using NFS, SMB, or object protocols without installing IBM Storage Scale software on the node. The primary benefit of protocol nodes is that applications, workstations, and users that do not have the IBM Storage Scale client can still access IBM Storage Scale data through one of these configured protocols. IBM Storage Scale is often used as an enterprise data lake or a central data repository.

IBM Storage CEPH

IBM Storage Ceph is an enterprise-grade, distributed, universal, software-defined storage solution, proven at scale, providing a single, efficient, unified storage platform for object, block, and file storage.

An IBM Storage Ceph cluster can be installed by running a single command and also features a dashboard UI and APIs for lights-out data center operations.

IBM Storage Ceph, a software-defined storage solution, can be consumed either as a software-only delivery to run on industry-standard x86 hardware of choice or as a combined software and hardware solution. The Ready Nodes hardware and software combined solution is fully validated and supported by IBM, providing a single point of contact for both acquisition and support services.

The following are common use cases for IBM Storage Ceph:

- Object storage as-a-service:
  - IBM Storage Ceph is an AWS S3-compatible on-premise object storage solution, with high fidelity to AWS services such as IAM, SSE, Object Lock, and more.
  - Replication between multiple locations, including public cloud, is also possible.
- AI/ML and Data Lakes
  - An ideal storage platform for AI/ML and Analytics workloads. Scale-out object storage for object, file and block storage from a single solution.

- Replace Hadoop HDFS with S3A and IBM Storage Ceph services to achieve greater freedom of choice, with handy features like snapshots and clones, and increased flexibility.

- File as-a-service
  - IBM Storage Ceph provides CephFS, a scalable and reliable file system.
  - Natively accessible by Linux systems and by NFS for non-Linux clients.
- Cloud Native (S3) Data Pipelines
  - IBM Storage Ceph can be useful in creating data pipelines by leveraging bucket notifications.
  - Whenever a bucket state change occurs, IBM Storage Ceph can trigger a follow-up process or activity by sending a bucket notification.
  - This enables the creation of AWS-like Lambda serverless services on-premise or in hybrid settings.
- Active Archive & Near-line storage
  - Use IBM Storage Ceph as an active target for archiving, rather than using offline media.
  - The near-line storage use case is to move cold data from more expensive and high-performance storage to IBM Storage Ceph.
  - This frees up more expensive capacity from the main tier, while still keeping it available and accessible on IBM Storage Ceph.
- Platform storage for Kubernetes and OpenStack
  - IBM Storage Ceph can be used as Kubernetes (K8s) storage by using the CSI (Container Storage Interface) API.

IBM Storage Ceph high-level architecture

Figure 4-3 illustrates the IBM Ceph storage high-level architecture, offering an overview of its key components.

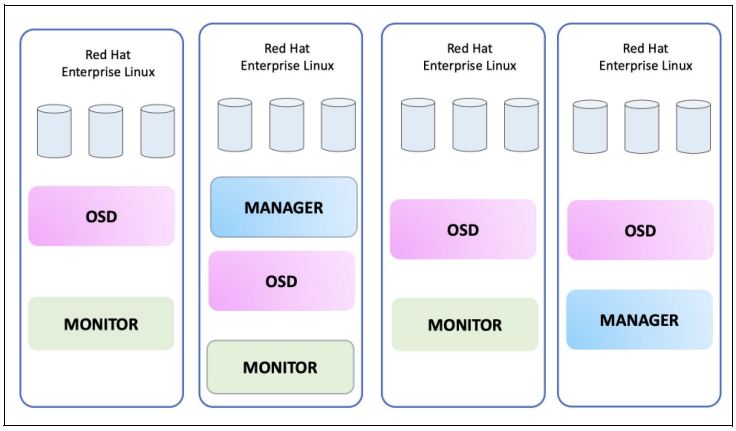


Figure 4-3 IBM Storage Ceph high level architecture

Ceph Manager

The Ceph Manager provides detailed information about placement groups, process metadata, and host metadata, significantly enhancing performance at scale compared to the Ceph Monitor.

It manages the execution of many read-only Ceph CLI queries, such as placement group statistics, and offers RESTful monitoring APIs. The system can run two manager instances—one active and one standby—to ensure failover and HA.

Ceph Monitor

A Ceph Monitor maintains the master copy of the cluster map, which reflects the current state of the IBM Ceph Storage cluster. Monitors are designed to ensure high consistency and use the Paxos consensus protocol to agree on the cluster's state.

OSD (Object Storage Daemon)

Ceph OSDs handle the storage of data on behalf of Ceph clients. They also leverage the CPU, memory, and network resources of Ceph nodes to perform various functions, including data replication, erasure coding, re-balancing, recovery, monitoring, and reporting.

IBM Storage Ceph and Red Hat Packaging.

IBM Ceph can be delivered in several ways from both IBM and Red Hat. Table 4-3 shows some of the packaging options you can choose.

Table 4-3 IBM Storage Ceph and Red Hat packaging

Red Hat OpenStack Platform: Red Hat Ceph Storage	**IBM Storage Fusion: a container native implementation of Ceph for Red Hat OpenShift**	**IBM Storage Ceph: on-prem S3 storage at scale and performance**
► Cinder block storage ► Nova ephemeral storage ► Glance image storage ► Swift object storage ► Manila file storage ► Advanced integration ► Unified management	► Self-managinf storage powered by Red hat Ceph Storage ► Automated by Rook and completed with Multicloud object gateway ► Advanced integration, automation, ease of use ► Persistent storage for Red Hat OpenShift steful workloads	► Object storage ► Block storage ► File storage ► S3 compatible with AWS

IBM Storage Fusion

For clients seeking a more turnkey solution, many of the storage technologies mentioned are included in our IBM Storage Fusion offerings.

The IBM Storage Fusion HCI System is a comprehensive "bare metal" Red Hat OpenShift cluster-in-a-box. It simplifies the deployment of Red Hat OpenShift on bare metal, while still delivering its full benefits. This solution includes a complete hardware and software stack, all supported by IBM. It features data protection, resilience capabilities, and enterprise-class high-performance storage.

For those not requiring a full hardware solution, the IBM Storage Fusion SDS provides the same software stack found in the IBM Storage Fusion HCI System. This software can be deployed on custom-built Red Hat OpenShift clusters, whether in private or public cloud environments.

For more information on IBM Storage Fusion, see *IBM Storage Fusion Product Guide*, REDP-5688.

4.3.4 Storage backup options and tools

Backing up a Red Hat OpenShift cluster deployed on IBM Power involves ensuring the integrity and availability of the cluster's configuration, state, and persistent data.

Application backup and restore operations

As a cluster administrator, you can use Red Hat OpenShift API for Data Protection (OADP) to perform backup and restore operations for applications running on Red Hat OpenShift Container Platform.

OADP allows for backing up and restoring Kubernetes resources and internal images at the namespace level. OADP facilitates the backup and restoration of persistent volumes (PVs) using either snapshots or Restic.

For more information, see Red Hat OpenShift Backup and restore.

Note: The following are the supported object storage options for storing backups:

- Red Hat OpenShift Data Foundation
- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- S3-compatible object storage
- IBM Cloud Object Storage S3

Velero

Velero is a powerful, open-source tool for Kubernetes and Red Hat OpenShift backup and restore, disaster recovery, and migration providing the following features:

- Backup and restore of Kubernetes/Red Hat OpenShift resources and persistent volumes.
- Scheduled backups.
- Supports multiple storage back-ends (AWS S3, GCP, Azure Blob Storage, IBM Cloud Object Storage for example).
- Integrates with plug-ins for additional functionality.

Prerequisites

Consider the following prerequisites:

- Ensure you have a Red Hat OpenShift cluster running on IBM Power.
- Download and install the Velero CLI on your local machine.
- Set up object storage (e.g., IBM Cloud Object Storage) for storing backups.

Installation and configuration

To install follow the following steps:

1. Download Velero CLI, as shown in Example 4-1.

Example 4-1 Download Velero CLI

```
root@yslpower:~# wget
https://github.com/vmware-tanzu/velero/releases/download/v1.13.2/velero-v1.13.2-li
nux-ppc64le.tar.gz
root@yslpower:~# tar -xvf  velero-v1.13.2-linux-ppc64le.tar.gz
root@yslpower:~# sudo mv velero-v1.13.2-linux-ppc64le/velero /usr/local/bin/
```

2. Configure Object Storage

- Create IBM Cloud Object Storage Bucket:
  - Go to the IBM Cloud Console.
  - Create a new bucket for Velero backups.
- Create Service Credentials:
  - Generate the service credentials for accessing the bucket.
  - Save the credentials to a file, e.g., `credentials-velero`.

3. Deploy Velero on Red Hat OpenShift
   - Install Velero, as shown in Example 4-2. Replace `<YOUR_BUCKET>`, `<YOUR_REGION>`, and `<YOUR_COS_ENDPOINT>` with your IBM Cloud Object Storage details.

Example 4-2 Deploy Velero on Red Hat OpenShift

```
$ velero install \
  --provider aws \
  --bucket <YOUR_BUCKET> \
  --secret-file ./credentials-velero \
  --use-restic \
  --backup-location-config
region=<YOUR_REGION>,s3ForcePathStyle=true,s3Url=https://<YOUR_COS_ENDPOINT>
```

   - Verify Installation, as shown in Example 4-3, and ensure all Velero pods are running correctly.

Example 4-3 Verify installation

```
$ kubectl get pods -n velero
```

4. Create a Backup, as shown in Example 4-4. Replace `<BACKUP_NAME>` with a name for your backup and `<NAMESPACE>` with the namespace you want to include in the backup.

Example 4-4 Create backup

```
$ velero backup create <BACKUP_NAME> --include-namespaces <NAMESPACE>
```

5. Restore from Backup. Replace `<BACKUP_NAME>` with the name of the backup you want to restore from, as shown in Example 4-5.

Example 4-5 Restore from backup

```
$ velero restore create --from-backup <BACKUP_NAME>
```

IBM Spectrum Protect Plus

IBM Spectrum Protect Plus is a data protection solution that supports Kubernetes and Red Hat OpenShift environments, including those on IBM Power providing the following features:

- Backup, recovery, and replication for containerized applications.
- Application-consistent backups.
- Supports multiple storage targets.
- Integration with IBM Spectrum Scale for efficient data management.

For more information, refer to *IBM Spectrum Protect Plus: Protecting Red Hat OpenShift Containerized Environments*, REDP-5636.

Custom scripting and automation

Backing up a Red Hat OpenShift cluster using custom scripting and automation involves creating scripts to back up critical components of the cluster, including the etcd database, Persistent Volumes (PVs), and configuration files. Below is an outline of the steps and example scripts for performing a comprehensive backup.

Prerequisites

Consider the following prerequisites:

- Ensure you have administrative access to your Red Hat OpenShift cluster.
- Set up a remote storage solution (for example NFS or S3) to store backups.
- Install necessary tools like **`oc`** (Red Hat OpenShift CLI), **`kubectl`**, and **`rsync`**.

Components to back up

The following components should be backed up:

- The `etcd` Database: Stores the state of the cluster.
- Persistent Volumes (PVs): Store application data.
- Configuration Files: Include manifests, secrets, and config maps.

Backup sample scripts

This section provides some sample scripts to use.

Note: The following scripts have been tested in our development environment and are not intended for direct use in a production setting. They should be carefully reviewed and customized to suit the specific requirements and configurations of your own environment before deployment.

1. To backup the `etcd` database, create a script named *backup_etcd.sh*, as shown in Example 4-6.

Example 4-6 Backup etcd databases

```
#!/bin/bash
# Define variables
BACKUP_DIR=/path/to/backup/etcd
TIMESTAMP=$(date +%F-%H-%M-%S)
ETCD_BACKUP_DIR=$BACKUP_DIR/$TIMESTAMP
# Create a backup directory
mkdir -p $ETCD_BACKUP_DIR
# Export etcd backup
oc exec -n openshift-etcd etcd-<master-node-name> -- etcdctl snapshot save
/tmp/snapshot.db
oc cp openshift-etcd/etcd-<master-node-name>:/tmp/snapshot.db
$ETCD_BACKUP_DIR/snapshot.db
# Verify backup
oc exec -n openshift-etcd etcd-<master-node-name> -- etcdctl snapshot status
/tmp/snapshot.db
echo "etcd backup completed and saved to $ETCD_BACKUP_DIR"
```

2. To backup Persistent Volumes, create a script named *backup_pvs.sh*, `as shown` in Example 4-7.

Example 4-7 Backup persistent volumes

```
#!/bin/bash
```

```
# Define variables
BACKUP_DIR=/path/to/backup/pvs
TIMESTAMP=$(date +%F-%H-%M-%S)
PV_BACKUP_DIR=$BACKUP_DIR/$TIMESTAMP
# Create a backup directory
mkdir -p $PV_BACKUP_DIR
# Get list of PVs
PVS=$(oc get pv --no-headers -o custom-columns=":metadata.name")
# Backup each PV
for PV in $PVS; do
    PVC=$(oc get pv $PV -o
jsonpath='{.spec.claimRef.namespace}/{.spec.claimRef.name}')
    BACKUP_PATH=$PV_BACKUP_DIR/$PVC
    mkdir -p $BACKUP_PATH
    oc rsync /mnt/$PVC $BACKUP_PATH
done
echo "Persistent volumes backup completed and saved to $PV_BACKUP_DIR"
```

3. To back up configuration Files, create a script named *backup_configs.sh*, as shown in Example 4-8.

Example 4-8 Backup configuration files

```
#!/bin/bash
# Define variables
BACKUP_DIR=/path/to/backup/configs
TIMESTAMP=$(date +%F-%H-%M-%S)
CONFIG_BACKUP_DIR=$BACKUP_DIR/$TIMESTAMP
# Create a backup directory
mkdir -p $CONFIG_BACKUP_DIR
# Backup all namespaces configurations
oc get namespaces -o json | jq -r '.items[].metadata.name' | while read NAMESPACE;
do
    echo "Backing up namespace: $NAMESPACE"
    mkdir -p $CONFIG_BACKUP_DIR/$NAMESPACE
    # Backup resources in namespace
    for RESOURCE in secrets configmaps; do
        oc get $RESOURCE -n $NAMESPACE -o yaml >
$CONFIG_BACKUP_DIR/$NAMESPACE/${RESOURCE}.yaml
    done
done
echo "Configuration files backup completed and saved to $CONFIG_BACKUP_DIR"
```

5

Installing Red Hat OpenShift on IBM Power Systems

After setting up your environment as discussed in Chapter 4, you are ready to install your Red Hat OpenShift cluster on your IBM Power infrastructure. This chapter describes the installation of Red Hat OpenShift on IBM Power servers.

It covers the following topics:

5.1 Installation methods

Red Hat OpenShift Container Platform is designed to be installed in a variety of environments and provides significant flexibility in how you install your cluster. There are four primary methods of deploying a cluster:

- **Interactive:** This approach utilizes the web-based Assisted Installer for online networks, providing smart defaults, validation checks, and a RESTful API for automated processes.
- **Local Agent-based:** Suitable for offline or limited networks, this method involves manually installing and configuring the Agent-based Installer using the command line.
- **Automated:** Deploys clusters on installer-provisioned infrastructure using the baseboard management controller (BMC) of each host, suitable for both connected and disconnected environments.
- **Full Control:** Aimed at maximum customization, this method enables deployment onto self-managed infrastructure, adaptable for any networking scenario.

Each method ensures a highly available infrastructure with no single points of failure and allows administrators to manage updates.

5.1.1 About the Red Hat OpenShift Container Platform installation

The Red Hat OpenShift Container Platform installation program enables deploying various cluster types by generating main assets like Ignition config files for bootstrap, control plane, and compute machines. These three machine configurations can start a Red Hat OpenShift Container Platform cluster when the infrastructure is correctly set up.

The program utilizes a system of targets and dependencies to manage installations. Each target concentrates on its dependencies, enabling parallel execution with the aim of creating a running cluster. Recognized existing components are leveraged to prevent redundant production, thereby simplifying the installation process. Figure 5-1 depicts the Red Hat OpenShift Cluster Platform's installation targets and dependencies.

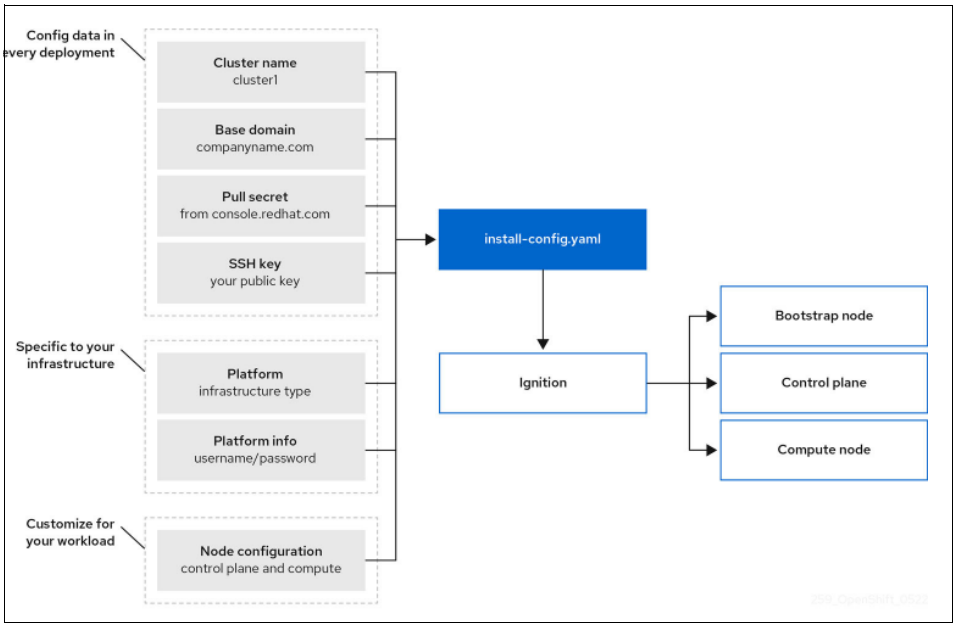


Figure 5-1 Red Hat OpenShift Container Platform installation targets and dependencies

5.1.2 Introduction to Red Hat Enterprise Linux CoreOS (RHCOS)

After installation, each cluster machine runs Red Hat Enterprise Linux CoreOS (RHCOS), which is an immutable container host version of Red Hat Enterprise Linux (RHEL) operating system. RHCOS features a RHEL kernel with SELinux enabled by default and includes `kubelet` – the Kubernetes node agent – and the CRI-O container runtime to provide an environment optimized for Kubernetes.

In Red Hat OpenShift Container Platform 4.15, the control plane machines use RHCOS with Ignition for initial provisioning. Updates are managed with OSTree and delivered as bootable container images which are applied in-place through rpm-ostree. This process is coordinated by the Machine Config Operator. to ensure seamless, consistent updates across the cluster – reducing operational complexity. Only the installation program and Machine Config Operator can modify machines through the use of Ignition configs and post-install updates.

5.1.3 Common terms for Red Hat OpenShift Container Platform installation

This glossary defines key terms related to Red Hat OpenShift Container Platform installation to aid in understanding the process.

Bootstrap Node: A temporary machine running minimal Kubernetes used to deploy the control plane.

Control Plane: The orchestration layer that manages the containers and container lifecycle. Consists of one or more as control plane nodes.

Compute Node: Nodes executing user workloads, also called worker nodes.

Infrastructure Node: Infrastructure nodes are a special class of compute or worker nodes that host only infrastructure components, such as the default router, the integrated container image registry, and the components for cluster metrics and monitoring. These infrastructure machines are not counted toward the total number of subscriptions that are required to run the environment.

Red Hat OpenShift installation program: Provisions infrastructure and deploys the cluster.

Assisted Installer: A web-based tool at console.redhat.com that provides a user interface or RESTful API for cluster configuration. The installer generates a discovery image for cluster machines.

Installer-Provisioned Infrastructure: The installation program deploys and configures cluster infrastructure.

User-Provisioned Infrastructure: Installing Red Hat OpenShift on pre-existing infrastructure, with the installation program generating the necessary assets and deploying the cluster.

Disconnected Installation: Installing Red Hat OpenShift in environments without internet access by pre-downloading required software and images.

Agent-based Installer: Similar to the Assisted Installer but designed for disconnected environments. Figure 5-2 on page 70 illustrates the process of downloading and installing Red Hat OpenShift on bare metal locally using the Agent-based installer.

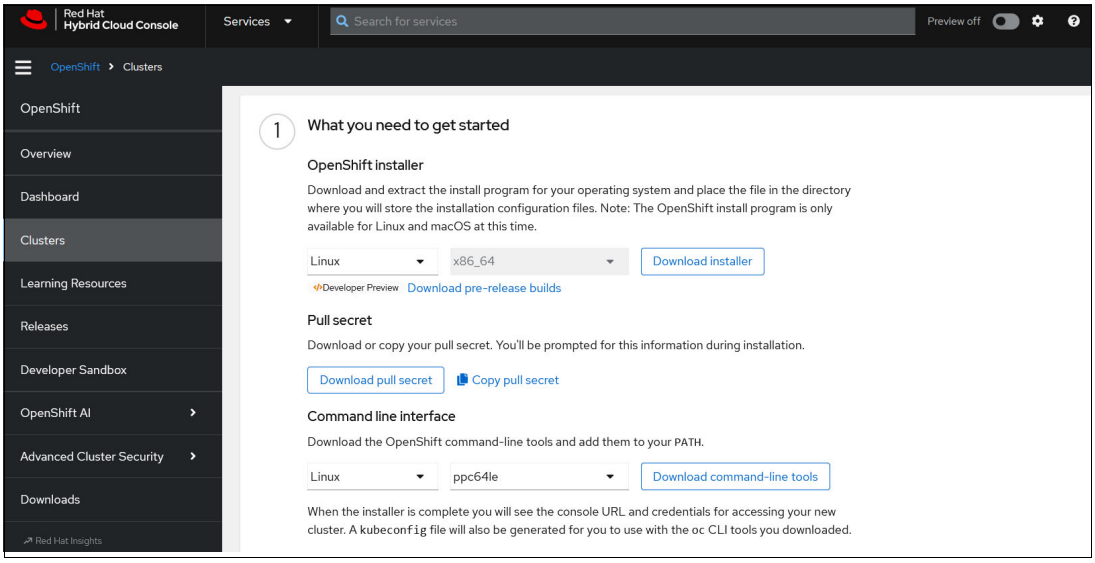


Figure 5-2 Install Red Hat OpenShift on Bare Metal locally with Agent installer.

Ignition config files: Used by the Ignition tool to configure RHCOS during initialization, generated for bootstrap, control plane, and worker nodes.

Kubernetes manifests: JSON or YAML specifications of Kubernetes API objects.

Kubelet: The primary node agent ensuring containers run in a pod.

Load balancers: Distribute incoming API traffic across control plane nodes.

Machine config operator: Manages and updates base OS and container runtime configurations for cluster nodes.

Operators: Packages, deploys, and manages Kubernetes applications in Red Hat OpenShift, encoding human operational knowledge into software.

5.1.4 Installation process

With the exception of installs using the Assisted Installer, you must download the installation program from the appropriate Cluster Type page on the Red Hat OpenShift Cluster Manager Hybrid Cloud Console. This console manages account REST APIs, registry tokens (pull secrets), and cluster registration for your Red Hat account.

In Red Hat OpenShift Container Platform 4.15 the installation program, – a Go binary file – performs file transformations on a set of assets. The interaction with the program varies by installation type:

- **Assisted Installer:** When using the Assisted Installer you first configure cluster settings via the installer. Once the cluster configuration is complete, a discovery ISO is downloaded and used to boot the cluster machines. This option is suitable for bare metal, vSphere, or Nutanix environments and can also be used on other platforms with or without integration.
- **Agent-based Installer:** Download and configure the Agent-based Installer to generate a discovery image. Boot cluster machines with this image for agent-based provisioning. Ideal for disconnected environments, requiring full provision of cluster infrastructure and resources.

- **Installer-Provisioned Infrastructure:** The installation program handles infrastructure bootstrapping and provisioning, creating necessary networking, machines, and OS, except for bare metal, where you must provide infrastructure.
- **User-Provisioned Infrastructure:** Provide all cluster infrastructure and resources, including bootstrap machine, networking, load balancing, storage, and individual machines.

The installation program uses three file sets: install-config.yaml, Kubernetes manifests, and Ignition config files.

Note: During installation, Kubernetes and Ignition config files can be altered to control the underlying RHCOS operating system. However, these modifications lack validation to ensure their suitability. Changes can potentially render your cluster non-functional. Therefore, it is not recommended to modify these files unless you are following official documented procedures or have explicit instructions from Red Hat support. This precaution helps maintain the stability and functionality of your Red Hat OpenShift cluster.

The installation configuration file is converted into Kubernetes manifests, which are then encapsulated into Ignition config files. These Ignition config files are used by the installation program to create the cluster. When running the installation program, all installation configuration files are pruned. Therefore, make sure to back up any configuration files you want to reuse.

Note: Parameters set during installation cannot be changed, but many cluster attributes can be modified after the installation is complete.

Assisted installer process

The Assisted Installer streamlines the Red Hat OpenShift Container Platform installation by creating a cluster configuration interactively through a web-based user interface or RESTful API. It prompts for necessary inputs and offers default values for other settings, which you can adjust directly in the UI or via the API. The system then generates a discovery image for booting cluster machines, which automates the installation of RHCOS and the provisioning agent. This method supports full integration on platforms like Nutanix, vSphere, and bare metal, and allows installations on other platforms without integration.

Agent-based installation process

Similar to the Assisted Installer, the Agent-based method requires downloading and setting up the Agent-based Installer first. This approach is ideal for environments that require the Assisted Installer's convenience but are disconnected and cannot directly use the web-based services.

Installer-provisioned infrastructure process

This default installation method uses an installation wizard to guide through the setup, filling in where automatic detections are unavailable and providing defaults otherwise. The installer handles infrastructure provisioning, accommodating both standard and customized cluster installations based on detailed specifications.

User-provisioned Infrastructure process

For environments where the installer-provisioned approach isn't feasible, you can use the installation program to generate the necessary assets and manually set up the cluster on self-managed infrastructure. This method involves extensive manual management of resources including load balancers, networking, and storage.

In all methods, Red Hat OpenShift Container Platform manages the cluster's broader aspects, ensuring self-sufficiency through configurations that allow for self-management and updates.

5.1.5 Cluster installation overview

The provisioning of a cluster consists of several stages where each node (or machine) is set up with crucial cluster details. Red Hat OpenShift Container Platform utilizes a temporary bootstrap machine to kickstart this procedure. This bootstrap machine initiates the setup by loading an ignition configuration file. The ignition file contains the necessary commands to form the cluster. In the initial phase, the bootstrap machine establishes the control plane machines. Once fully functional, these control plane machines manage the creation of further nodes, known as worker machines. A graphical representation of this workflow is presented in Figure 5-3.

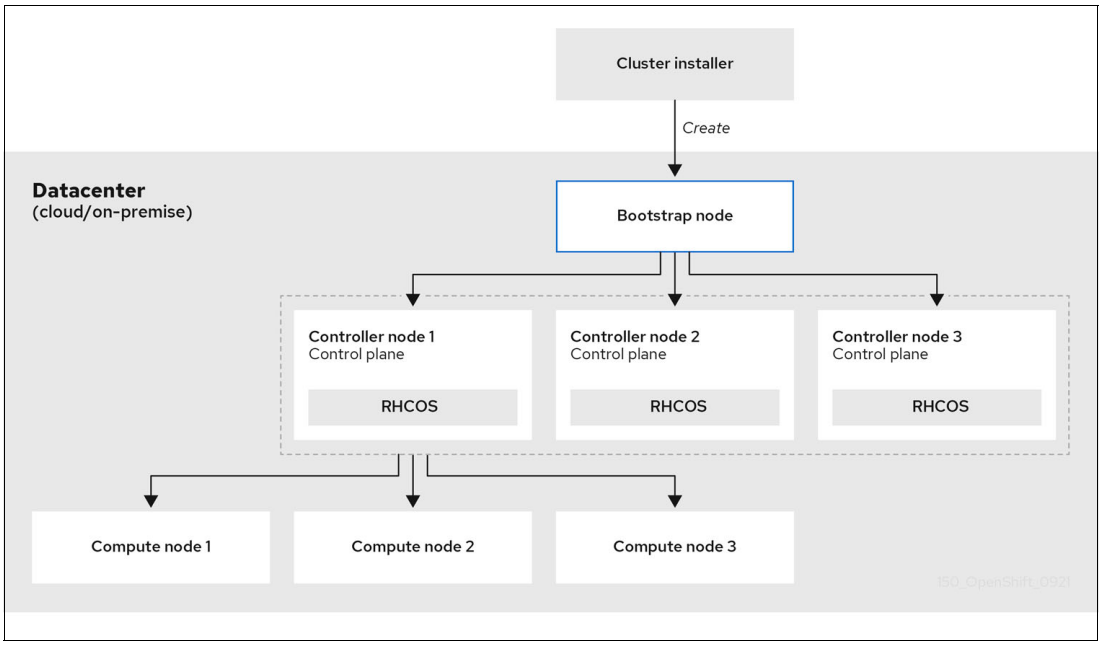


Figure 5-3 The installation process: setting up the bootstrap, control plane, and compute machines.

Once the cluster machines are initialized, the bootstrap machine is destroyed. All clusters utilize the bootstrap process for initialization. However, if you are provisioning the infrastructure for your cluster manually, you will need to handle many of these steps yourself.

Note: The installation program generates Ignition config files containing certificates that expire after 24 hours and are renewed at that interval. If the cluster is shut down and restarted after these 24 hours, it will automatically restore the expired certificates. However, to recover kubelet certificates, you must manually approve the pending node-bootstrapper certificate signing requests (CSRs).

It is advisable to utilize Ignition config files within 12 hours of their generation. The certificates in these files rotate between 16 and 22 hours after cluster installation. Using the files within this 12-hour window helps prevent installation failures that can occur if the certificates update while the installation is in progress.

5.1.6 Bootstrapping a cluster

Bootstrapping a cluster includes these steps:

1. The bootstrap machine boots up, providing the necessary remote resources for the control plane machines. Manual intervention is needed if you are provisioning the infrastructure.
2. The bootstrap machine initiates a single-node cluster and a temporary Kubernetes control plane.
3. Control plane machines access remote resources from the bootstrap machine to complete their setup. Manual intervention is required if you are provisioning the infrastructure.
4. The temporary control plane orchestrates the transfer of duties to the production control plane machines.
5. The Cluster Version Operator (CVO) activates and installs the **`etcd`** Operator, which then expands etcd across all control plane nodes.
6. After transferring responsibilities, the temporary control plane is destroyed.
7. The bootstrap machine deploys Red Hat OpenShift Container Platform components into the production control plane.
8. The installation program deactivates the bootstrap machine, necessitating manual intervention if you are provisioning the infrastructure.
9. The control plane configures the compute nodes.
10. The control plane also deploys additional services through various Operators.

This bootstrapping sequence results in an operational Red Hat OpenShift Container Platform cluster. The cluster proceeds to download and set up further components essential for routine operations, including provisioning compute machines in supported configurations.

5.1.7 Post-Installation node verification

The installation of the Red Hat OpenShift Container Platform is deemed complete once the following health checks are successfully met:

- The provisioner has access to the Red Hat OpenShift Container Platform web console.
- All control plane nodes are in a 'ready' state.
- All cluster Operators are reported as available.

After the installation is complete, you should continue monitoring the status of the nodes in your cluster. Figure 5-4 illustrates the node details of a cluster with multi-architecture compute machines.

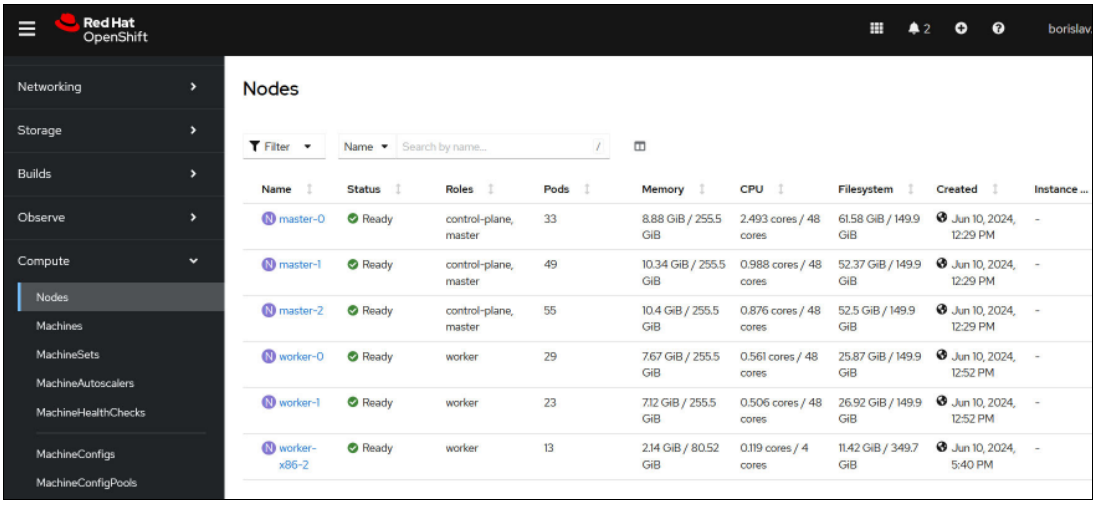


Figure 5-4 Node details of a cluster with multi-architecture compute machines.

Note: Once the installation is finalized, the cluster Operators designated for managing the worker nodes will continuously work to provision all worker nodes. It may take some time for all worker nodes to reach a **READY** state. For bare metal installations, it is recommended to wait at least 60 minutes before starting any troubleshooting on a worker node. For installations on other platforms, a waiting period of at least 40 minutes is advised before initiating troubleshooting. Note that a **DEGRADED** state in the cluster Operators responsible for worker nodes reflects the status of the Operators' resources and is not indicative of the worker nodes' conditions.

For comprehensive guidance and further information on Red Hat OpenShift Container Platform installation and configuration, please see Red Hat OpenShift Documentation and the IBM Redbooks publication *Implementing, Tuning, and Optimizing Workloads with Red Hat OpenShift on IBM Power*, SG24-8537.

5.2 Installing a Red Hat OpenShift cluster on IBM Power

This section describes the step-by-step installation process of a Red Hat OpenShift cluster on an IBM Power server with a single-architecture payload. We then describe the process of updating the cluster to support a multi-architecture payload so that x86-based worker nodes can be added to the cluster.

You can deploy a Red Hat OpenShift cluster on IBM Power Virtual Server or on your IBM Power infrastructure. You can use one of the following approaches when deploying a cluster on IBM Power infrastructure:

- **Standard Installation**

  This method is used to install the Red Hat OpenShift Container Platform on IBM Power servers that you manage.

- **Restricted Network Installation**

  This method is used when installing a Red Hat OpenShift Container Platform on IBM Power Servers in a restricted or disconnected network environment. This method involves using an internal mirror of the installation release content, allowing the setup of a cluster without needing an active internet connection for software components. Additionally, this approach helps ensure that the clusters use only container images that comply with your organization's external content policies.

Prerequisites

The installation process must complete successfully in the terminal. For detailed guidance, refer to “Installing Prerequisites and Dependencies” on page 47.

5.2.1 Installing a cluster on IBM Power user-provisioned infrastructure with internet access

This section covers the user-provided infrastructure (UPI) installation method with internet access, designed for enterprise customers with their own network infrastructure to handle load balancing, firewall, and DNS. For step-by-step installation instructions, see Section 3.2, “Red Hat OpenShift V4.3 deployment with internet connection stand-alone installation” in *Red Hat OpenShift V4.3 on IBM Power Systems Reference Guide*, REDP-5599.

As automation becomes increasingly important in many businesses today, this section focuses on 'Automation for OpenShift on IBM Power Platform.' The ocp4-upi-powervm project is part of 'Automation for OpenShift on IBM Power Platform' and provides Terraform-based automation code to help deploy Red Hat OpenShift Container Platform (OCP) 4.x on PowerVM systems managed by PowerVC. For users managing standalone PowerVM, there is a quickstart guide available. This guide uses an Ansible playbook to create a bastion host required for OCP deployment. Furthermore, the project leverages the same Ansible playbook to install OCP on PowerVM LPARs managed by PowerVC.

The ocp4-upi-multiarch-compute project offers code that simplifies the addition of Red Hat OpenShift Container Platform (OCP) 4.x compute workers post-installation across multiple platforms. This project can be used to add nodes (x86 or ppc64le) to an existing cluster with multi-architecture compute machines. For more information see: https://github.com/ocp-power-automation/ocp4-upi-multiarch-compute

In the following steps, we will cover Red Hat OpenShift deployment on a PowerVM server managed by IBM PowerVC using the ocp4-upi-powervm project:

Note: In order to successfully complete the steps described in this section, your IBM PowerVC environment must be configured. For more information on how to configure your environment, please refer to *IBM PowerVC Version 2.0 Introduction and Configuration*, SG24-8477.

1. The first step is to clone the ocp4-upi-powervm project and navigate to the *ocp4-upi-powervm* directory. This is shown in Example 5-1.

Example 5-1 Clone the ocp4-upi-powervm project using the git command.

```
# git clone https://github.com/ocp-power-automation/ocp4-upi-powervm.git
Cloning into 'ocp4-upi-powervm'...
remote: Enumerating objects: 2284, done.
remote: Counting objects: 100% (2284/2284), done.
remote: Compressing objects: 100% (808/808), done.
remote: Total 2284 (delta 1472), reused 2243 (delta 1448), pack-reused 0
Receiving objects: 100% (2284/2284), 500.30 KiB | 872.00 KiB/s, done.
Resolving deltas: 100% (1472/1472), done.

# cd ocp4-upi-powervm/

#ls -ltr
total 88
-rw-r--r--. 1 root root  2048 Jun 18 12:07 README.md
-rw-r--r--. 1 root root   156 Jun 18 12:07 OWNERS
-rw-r--r--. 1 root root 11357 Jun 18 12:07 LICENCE.txt
drwxr-xr-x. 2 root root   185 Jun 18 12:07 docs
-rw-r--r--. 1 root root  1265 Jun 18 12:07 DCO1.1.txt
drwxr-xr-x. 2 root root    24 Jun 18 12:07 data
-rw-r--r--. 1 root root  1868 Jun 18 12:07 CONTRIBUTING.md
-rw-r--r--. 1 root root  5031 Jun 18 12:07 CODE_OF_CONDUCT.md
-rw-r--r--. 1 root root  1011 Jun 18 12:07 versions.tf
-rw-r--r--. 1 root root  7885 Jun 18 12:07 var.tfvars
-rw-r--r--. 1 root root 17692 Jun 18 12:07 variables.tf
-rw-r--r--. 1 root root  2401 Jun 18 12:07 outputs.tf
-rw-r--r--. 1 root root 12779 Jun 18 12:07 ocp.tf
drwxr-xr-x. 7 root root    92 Jun 18 12:07 modules
```

2. Download the latest Terraform binary for the `ppc64le` architecture to the `ocp4-upi-powervm` directory as shown in Example 5-2 on page 76.

Example 5-2 Download the latest Terraform binary for the ppc64le architecture.

```
curl -L https://ftp2.osuosl.org/pub/ppc64el/terraform/terraform-1.4.6 -o terraform && chmod +x
terraform

# ls -ltr terraform
-rwxr-xr-x. 1 root root 87321914 Jun 18 12:37 terraform

# ./terraform --version
Terraform v1.4.6
```

3. Download and extract the required version of `archive.zip` from terraform-providers-power as shown in Example 5-3.

Example 5-3 Download and extract Terraform plugins for IBM Power.

```
# curl -L
https://github.com/ocp-power-automation/terraform-providers-power/releases/download/v0.19/archiv
e.zip -o archive.zip && unzip archive.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  115M  100  115M    0     0  35.1M      0  0:00:03  0:00:03 --:--:-- 54.8M

Archive:  archive.zip
   creating: registry.terraform.io/
   creating: registry.terraform.io/community-terraform-providers/
   creating: registry.terraform.io/community-terraform-providers/ignition/
   creating: registry.terraform.io/community-terraform-providers/ignition/2.1.3/
   creating: registry.terraform.io/community-terraform-providers/ignition/2.1.3/linux_ppc64le/
  inflating:
registry.terraform.io/community-terraform-providers/ignition/2.1.3/linux_ppc64le/terraform-provi
der-ignition
   creating: registry.terraform.io/terraform-provider-openstack/
   creating: registry.terraform.io/terraform-provider-openstack/openstack/
   creating: registry.terraform.io/terraform-provider-openstack/openstack/1.32.0/
   creating: registry.terraform.io/terraform-provider-openstack/openstack/1.32.0/linux_ppc64le/
  inflating:
registry.terraform.io/terraform-provider-openstack/openstack/1.32.0/linux_ppc64le/terraform-prov
ider-openstack
   creating: registry.terraform.io/hashicorp/
   creating: registry.terraform.io/hashicorp/random/
   creating: registry.terraform.io/hashicorp/random/3.4.3/
   creating: registry.terraform.io/hashicorp/random/3.4.3/linux_ppc64le/
  inflating:
registry.terraform.io/hashicorp/random/3.4.3/linux_ppc64le/terraform-provider-random
   creating: registry.terraform.io/hashicorp/null/
   creating: registry.terraform.io/hashicorp/null/3.2.1/
   creating: registry.terraform.io/hashicorp/null/3.2.1/linux_ppc64le/
  inflating: registry.terraform.io/hashicorp/null/3.2.1/linux_ppc64le/terraform-provider-null
   creating: registry.terraform.io/hashicorp/time/
   creating: registry.terraform.io/hashicorp/time/0.10.0/
   creating: registry.terraform.io/hashicorp/time/0.10.0/linux_ppc64le/
  inflating: registry.terraform.io/hashicorp/time/0.10.0/linux_ppc64le/terraform-provider-time
   creating: registry.terraform.io/terraform-providers/
   creating: registry.terraform.io/terraform-providers/ignition/
   creating: registry.terraform.io/terraform-providers/ignition/1.2.1/
   creating: registry.terraform.io/terraform-providers/ignition/1.2.1/linux_ppc64le/
  inflating:
registry.terraform.io/terraform-providers/ignition/1.2.1/linux_ppc64le/terraform-provider-igniti
on
   creating: registry.terraform.io/IBM-Cloud/
   creating: registry.terraform.io/IBM-Cloud/ibm/
   creating: registry.terraform.io/IBM-Cloud/ibm/1.60.0/
```

```
   creating: registry.terraform.io/IBM-Cloud/ibm/1.60.0/linux_ppc64le/
  inflating: registry.terraform.io/IBM-Cloud/ibm/1.60.0/linux_ppc64le/terraform-provider-ibm
```

4. Ensure that the plugins are present in the current directory. This is shown in Example 5-4.

Example 5-4 Validate that the plugins are present in the current directory.

```
# ls -lrt registry.terraform.io/
total 0
drwxrwxr-x. 3 root root 22 Feb  5 10:06 community-terraform-providers
drwxrwxr-x. 3 root root 22 Feb  5 10:06 terraform-providers
drwxrwxr-x. 3 root root 23 Feb  5 10:08 terraform-provider-openstack
drwxrwxr-x. 3 root root 17 Feb  5 10:13 IBM-Cloud
drwxrwxr-x. 5 root root 44 Feb  5 10:13 hashicorp
```

5. Run the **terraform init** command to utilize the locally downloaded plugins as shown in Example 5-5.

*Example 5-5 Running the **terraform init** command to utilize the local plugins.*

```
# ./terraform init --plugin-dir ./

Initializing the backend...
Initializing modules...
- bastion in modules/1_bastion
- bootstrapcomplete in modules/5_install/5_3_bootstrapcomplete
- bootstrapconfig in modules/5_install/5_2_bootstrapconfig
- bootstrapnode in modules/4_nodes/4_1_bootstrapnode
- helpernode in modules/3_helpernode
- install in modules/5_install/5_4_installcomplete
- installconfig in modules/5_install/5_1_installconfig
- masternodes in modules/4_nodes/4_2_masternodes
- network in modules/2_network
- workernodes in modules/4_nodes/4_3_workernodes

Initializing provider plugins...
- Finding terraform-provider-openstack/openstack versions matching "~> 1.32"...
- Finding hashicorp/random versions matching "~> 3.4"...
- Finding community-terraform-providers/ignition versions matching "~> 2.1.0"...
- Finding hashicorp/null versions matching "~> 3.2"...
- Installing terraform-provider-openstack/openstack v1.32.0...
- Installed terraform-provider-openstack/openstack v1.32.0 (unauthenticated)
- Installing hashicorp/random v3.4.3...
- Installed hashicorp/random v3.4.3 (unauthenticated)
- Installing community-terraform-providers/ignition v2.1.3...
- Installed community-terraform-providers/ignition v2.1.3 (unauthenticated)
- Installing hashicorp/null v3.2.1...
- Installed hashicorp/null v3.2.1 (unauthenticated)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

?
? Warning: Incomplete lock file information for providers
?
? Due to your customized provider installation methods, Terraform was forced to calculate lock
file checksums locally for the
? following providers:
?   - community-terraform-providers/ignition
?   - hashicorp/null
?   - hashicorp/random
```

```
?   - terraform-provider-openstack/openstack
?
? The current .terraform.lock.hcl file only includes checksums for linux_ppc64le, so Terraform
running on another platform will
? fail to install these providers.
?
? To calculate additional checksums for another platform, run:
?   terraform providers lock -platform=linux_amd64
? (where linux_amd64 is the platform to generate)
?

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

The message 'Terraform has been successfully initialized!' indicates a successful initialization. You can now run Terraform commands to create your cluster from an IBM Power server. For more information about IBM PowerVC integration with Terraform and Ansible, refer to Chapter 6, “IBM PowerVC integration with Terraform and Ansible”, in *IBM PowerVC Version 2.0 Introduction and Configuration*, SG24-8477.

6. Follow the steps to create a Red Hat CoreOS (RHCOS) image using IBM PowerVC as described in *IBM PowerVC Version 2.0 Introduction and Configuration*, SG24-8477.
7. Specify the set of variables in the **`var.tfvars`** file for your PowerVC environment. For more information, see the var.tfvars-doc.md. This is shown in Example 5-6.

*Example 5-6 Specify the set of variables in the **`var.tfvars`** file for your PowerVC environment.*

```
# nano var.tfvars
  GNU nano 2.9.8                                                  var.tfvars

### PowerVC Details
auth_url                    = "<https://<HOSTNAME>:5000/v3/>"
user_name                   = "<powervc-login-user-name>"
password                    = "<powervc-login-user-password>"
tenant_name                 = "<tenant_name>"
domain_name                 = "Default"
openstack_availability_zone = ""

network_name = "<network_name>"

### OpenShift Cluster Details

bastion   = { instance_type = "<bastion-compute-template>", image_id = "<image-uuid-rhel>",
"count" = 1 }
bootstrap = { instance_type = "<bootstrap-compute-template>", image_id = "<image-uuid-rhcos>",
"count" = 1 }
master    = { instance_type = "<master-compute-template>", image_id = "<image-uuid-rhcos>",
"count" = 3 }
worker    = { instance_type = "<worker-compute-template>", image_id = "<image-uuid-rhcos>",
"count" = 2 }

rhel_username                  = "root" #Set it to an appropriate username for non-root user
access
public_key_file                = "data/id_rsa.pub"
private_key_file               = "data/id_rsa"
```

```
rhel_subscription_username      = "<subscription-id>"       #Leave this as-is if using CentOS as
bastion image
rhel_subscription_password      = "<subscription-password>" #Leave this as-is if using CentOS as
bastion image
rhel_subscription_org           = ""                        # Define it only when using
activationkey for RHEL subscription
rhel_subscription_activationkey = ""                        # Define it only when using
activationkey for RHEL subscription

connection_timeout = 45
jump_host          = ""

### OpenShift Installation Details

openshift_install_tarball =
"https://mirror.openshift.com/pub/openshift-v4/ppc64le/clients/ocp/stable/openshift-install-linu
x.tar.$
openshift_client_tarball  =
"https://mirror.openshift.com/pub/openshift-v4/ppc64le/clients/ocp/stable/openshift-client-linux
.tar.g$
pull_secret_file          = "data/pull-secret.txt"

cluster_domain    = "ibm.com"  # Set domain to nip.io or xip.io if you prefer using online
wildcard domain and avoid modifying /etc/hosts
cluster_id_prefix = "test-ocp" # Set it to empty if just want to use cluster_id without prefix
cluster_id        = ""         # It will use random generated id with cluster_id_prefix if this
is not set

###
scg_id                     = "df21cec9-c244-4d3d-b927-df1518672e87"
###

storage_type                = "nfs"
volume_size                 = "300" # Value in GB
volume_storage_template     = ""
```

8. Next, we create a 'logs' directory to collect logs from the Terraform execution for future review and troubleshooting purposes. Additionally, we will create four shell scripts: ***initiate.sh***, ***state.sh***, ***install.sh***, and ***uninstall.sh.*** This is shown in Example 5-7.

Example 5-7 Create a 'logs' directory and four shell scripts.

```
# mkdir logs

echo -e '#!/bin/sh\n./terraform init --plugin-dir ./ | tee logs/init_ocp_multi_$(date
+'%Y-%m-%d-%H-%M-%S').log' > initiate.sh && chmod +x initiate.sh

echo -e '#!/bin/sh\n./terraform apply -auto-approve -var-file var.tfvars | tee
logs/install_ocp_multi_$(date +'%Y-%m-%d-%H-%M-%S').log' > install.sh && chmod +x install.sh

echo -e '#!/bin/sh\n./terraform destroy -auto-approve -var-file var.tfvars | tee
logs/uninstall_ocp_multi_$(date +'%Y-%m-%d-%H-%M-%S').log' > uninstall.sh && chmod +x
uninstall.sh

# echo -e '#!/bin/sh\n./terraform state list | tee logs/state_ocp_multi_$(date
+'%Y-%m-%d-%H-%M-%S').log' > state.sh && chmod +x state.sh
```

9. Check the shell script by initiating Terraform with `./initiate.sh` and ensure it correctly writes the data to the `logs` directory as shown in Example 5-8.

Example 5-8 Check the shell script by initiating it.

```
# ./initiate.sh
```

```
Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of terraform-provider-openstack/openstack from the dependency lock
file
- Reusing previous version of hashicorp/random from the dependency lock file
- Reusing previous version of community-terraform-providers/ignition from the dependency lock
file
- Reusing previous version of hashicorp/null from the dependency lock file
- Using previously-installed terraform-provider-openstack/openstack v1.32.0
- Using previously-installed hashicorp/random v3.4.3
- Using previously-installed community-terraform-providers/ignition v2.1.3
- Using previously-installed hashicorp/null v3.2.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

# ls -ltr logs/
total 4
-rw-r--r--. 1 root root    0 Jun 18 13:37 state_ocp_multi_2024-06-18-13-37-55.log
-rw-r--r--. 1 root root 1204 Jun 18 13:42 init_ocp_multi_2024-06-18-13-42-32.log
```

Note: As an alternative to Terraform, we can use OpenTofu, which is supported on Power and facilitates the deployment of Red Hat OpenShift on IBM Power servers. For additional information on using OpenTofu, see this IBM community article.

10. Verify the variables in the ***var.tfvars*** file and run the ***install.sh*** script to start the Red Hat OpenShift installation as shown in Example 5-9.

Example 5-9 Run the install.sh script to start the Red Hat OpenShift installation.

```
./install.sh
module.bootstrapnode.data.ignition_file.b_hostname: Reading...
module.workernodes.data.ignition_file.w_hostname[0]: Reading...
module.masternodes.data.ignition_file.m_hostname[2]: Reading...
module.masternodes.data.ignition_file.m_hostname[0]: Reading...
module.masternodes.data.ignition_file.m_hostname[1]: Reading...
module.workernodes.data.ignition_file.w_hostname[1]: Reading...
module.workernodes.data.ignition_file.w_hostname[0]: Read complete after 0s
[id=28b9dcc333049039879c9c1e94f95816f0341945047e8ae59674e1233f72be83]
module.masternodes.data.ignition_file.m_hostname[2]: Read complete after 0s
[id=0c163cb1c5dd992d636e27b372616311c7c43967102261b4f85f9ca00dadf964]
module.workernodes.data.ignition_file.w_hostname[1]: Read complete after 0s
[id=82d7ac7858b4e7143397c00ce94a148a055fdd676699a6a0176fbc45c50731f8]
module.masternodes.data.ignition_file.m_hostname[0]: Read complete after 0s
[id=7551bfa9e87523c711bf18607b8af5ccfee1657ea6c4817bbc3dd2186602f590]
module.masternodes.data.ignition_file.m_hostname[1]: Read complete after 0s
[id=3a19b43cbd34e4811a658630d198c68ef3f2a95ead84d014b2ec674c9602bd6e]
module.bootstrapnode.data.ignition_file.b_hostname: Read complete after 0s
[id=1ec8928da9e89f9b35deb26dd484665fda91d99d73e31330dce71edf3a4e19cc]
module.workernodes.random_id.label[0]: Refreshing state... [id=S4M]
module.bastion.random_id.label[0]: Refreshing state... [id=AUQ]
module.bootstrapnode.random_id.label[0]: Refreshing state... [id=Q-U]
```

```
module.masternodes.random_id.label[0]: Refreshing state... [id=5kk]
module.bastion.data.openstack_compute_flavor_v2.bastion: Reading...
module.bootstrapnode.data.openstack_compute_flavor_v2.bootstrap: Reading...
module.network.data.openstack_networking_network_v2.network: Reading...
module.workernodes.data.openstack_compute_flavor_v2.worker: Reading...
module.bastion.openstack_compute_keypair_v2.key-pair[0]: Refreshing state... [id=ocp-10-keypair]
module.masternodes.data.openstack_compute_flavor_v2.master: Reading...
module.bastion.openstack_blockstorage_volume_v2.storage_volume[0]: Refreshing state...
[id=198fffe5-642d-4f57-b19e-69be7bcbdbd8]
module.workernodes.data.openstack_compute_flavor_v2.worker: Read complete after 1s
[id=b40f0c57-6009-480d-9319-1bbc6f5469c9]
module.network.data.openstack_networking_network_v2.network: Read complete after 1s
[id=8adb6845-1d3c-4099-a5d7-fe86e895be21]
module.network.data.openstack_networking_subnet_v2.subnet: Reading...
module.workernodes.openstack_compute_flavor_v2.worker_scg[0]: Refreshing state...
[id=0db9f7df-9ce8-48c4-9c44-219d0ca21951]
...
module.install.null_resource.install (remote-exec): TASK [ocp-customization : Get all Nodes]
*************************************
module.install.null_resource.install (remote-exec): skipping: [ocp-10-bastion-0] => {"changed":
false, "false_condition": "node_labels", "skip_reason": "Conditional result was False"}
module.install.null_resource.install (remote-exec): TASK [ocp-customization : Automate Node
labels] *******************************
module.install.null_resource.install (remote-exec): skipping: [ocp-10-bastion-0] => {"changed":
false, "false_condition": "node_labels", "skip_reason": "Conditional result was False"}
module.install.null_resource.install (remote-exec): PLAY [OCP post-install HA]
************************************************
module.install.null_resource.install (remote-exec): skipping: no hosts matched
module.install.null_resource.install (remote-exec): PLAY RECAP
*********************************************************************
module.install.null_resource.install (remote-exec): ocp-10-bastion-0          : ok=19
changed=18   unreachable=0    failed=0    skipped=8    rescued=0    ignored=0
module.install.null_resource.install (remote-exec): worker-0                  : ok=2
changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
module.install.null_resource.install (remote-exec): worker-1                  : ok=2
changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
module.install.null_resource.install: Creation complete after 17m8s [id=2065901463032178167]
module.install.null_resource.upgrade[0]: Creating...
module.install.null_resource.upgrade[0]: Provisioning with 'file'...
module.install.null_resource.upgrade[0]: Provisioning with 'remote-exec'...
module.install.null_resource.upgrade[0] (remote-exec): Connecting to remote host via SSH...
module.install.null_resource.upgrade[0] (remote-exec):   Host: x.x.x.142
module.install.null_resource.upgrade[0] (remote-exec):   User: root
module.install.null_resource.upgrade[0] (remote-exec):   Password: false
module.install.null_resource.upgrade[0] (remote-exec):   Private key: true
module.install.null_resource.upgrade[0] (remote-exec):   Certificate: false
module.install.null_resource.upgrade[0] (remote-exec):   SSH Agent: false
module.install.null_resource.upgrade[0] (remote-exec):   Checking Host Key: false
module.install.null_resource.upgrade[0] (remote-exec):   Target Platform: unix
module.install.null_resource.upgrade[0] (remote-exec): Connected!
module.install.null_resource.upgrade[0] (remote-exec): Running ocp upgrade playbook...
module.install.null_resource.upgrade[0] (remote-exec): Using /root/ocp4-playbooks/ansible.cfg as
config file
module.install.null_resource.upgrade[0] (remote-exec): [WARNING]: Found both group and host with
same name: bootstrap

module.install.null_resource.upgrade[0] (remote-exec): PLAY [Upgrade OCP cluster]
****************************************************

module.install.null_resource.upgrade[0] (remote-exec): TASK [Gathering Facts]
****************************************************
module.install.null_resource.upgrade[0] (remote-exec): ok: [ocp-10-bastion-0]
```

```
module.install.null_resource.upgrade[0] (remote-exec): TASK [include_role : ocp-upgrade]
*********************************************
module.install.null_resource.upgrade[0] (remote-exec): skipping: [ocp-10-bastion-0] =>
{"changed": false, "false_condition": "(upgrade_version is defined and upgrade_version != \"\")
or (upgrade_image is defined and upgrade_image != \"\") or (eus_upgrade_channel is defined and
eus_upgrade_channel != \"\") or (eus_upgrade_image is defined and eus_upgrade_image != \"\")\n",
"skip_reason": "Conditional result was False"}

module.install.null_resource.upgrade[0] (remote-exec): PLAY RECAP
*********************************************************************
module.install.null_resource.upgrade[0] (remote-exec): ocp-10-bastion-0           : ok=1
changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

module.install.null_resource.upgrade[0]: Creation complete after 3s [id=5032405106525673703]

Apply complete! Resources: 6 added, 0 changed, 2 destroyed.

Outputs:

bastion_ip = "x.x.x.142"
bastion_ssh_command = "ssh -i data/id_rsa root@x.x.x.142"
bootstrap_ip = "x.x.x.140"
cluster_id = "ocp-10"
etc_hosts_entries = <<EOT

x.x.x.142 api.ocp-10.rchland.ibm.com console-openshift-console.apps.ocp-10.rchland.ibm.com
integrated-oauth-server-openshift-authentication.apps.ocp-10.rchland.ibm.com
oauth-openshift.apps.ocp-10.rchland.ibm.com
prometheus-k8s-openshift-monitoring.apps.ocp-10.rchland.ibm.com
grafana-openshift-monitoring.apps.ocp-10.rchland.ibm.com example.apps.ocp-10.rchland.ibm.com

EOT
install_status = "COMPLETED"
master_ips = [
  "x.x.x.141",
  "x.x.x.138",
  "x.x.x.143",
]
oc_server_url = "https://api.ocp-10.rchland.ibm.com:6443"
storageclass_name = "nfs-storage-provisioner"
web_console_url = "https://console-openshift-console.apps.ocp-10.rchland.ibm.com"
worker_ips = [
  "x.x.x.137",
  "x.x.x.139",
]
[root@thel89-terra ocp4-upi-powervm]#
```

11. The next step is to go to the HMC and verify the LPARs installation. Figure 5-5 on page 83 illustrates the HMC GUI and the LPARs installations.

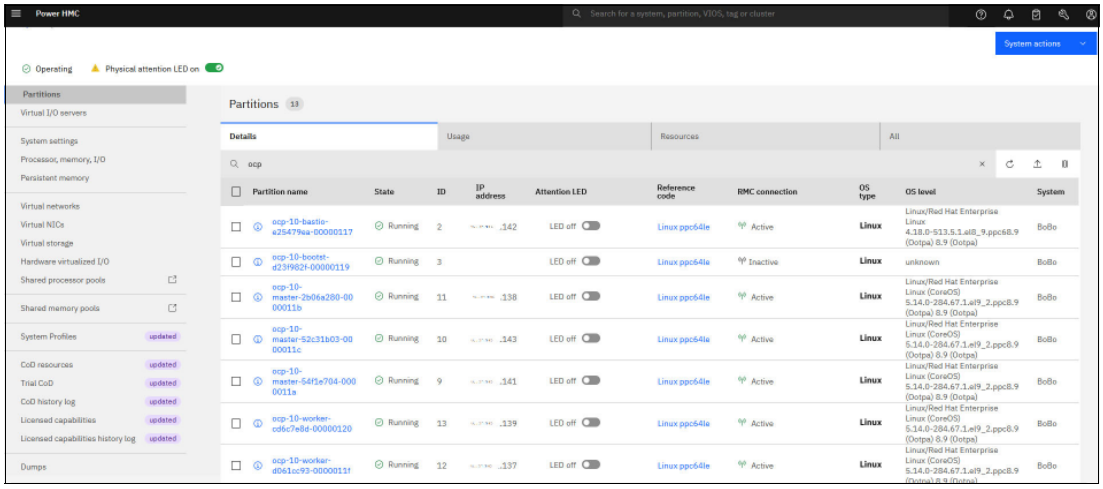


Figure 5-5 The HMC GUI and the LPARs installations.

12. Navigate to PowerVC and verify the LPARs installations as well. Figure 5-6 illustrates the PowerVC UI and the LPARs installations.

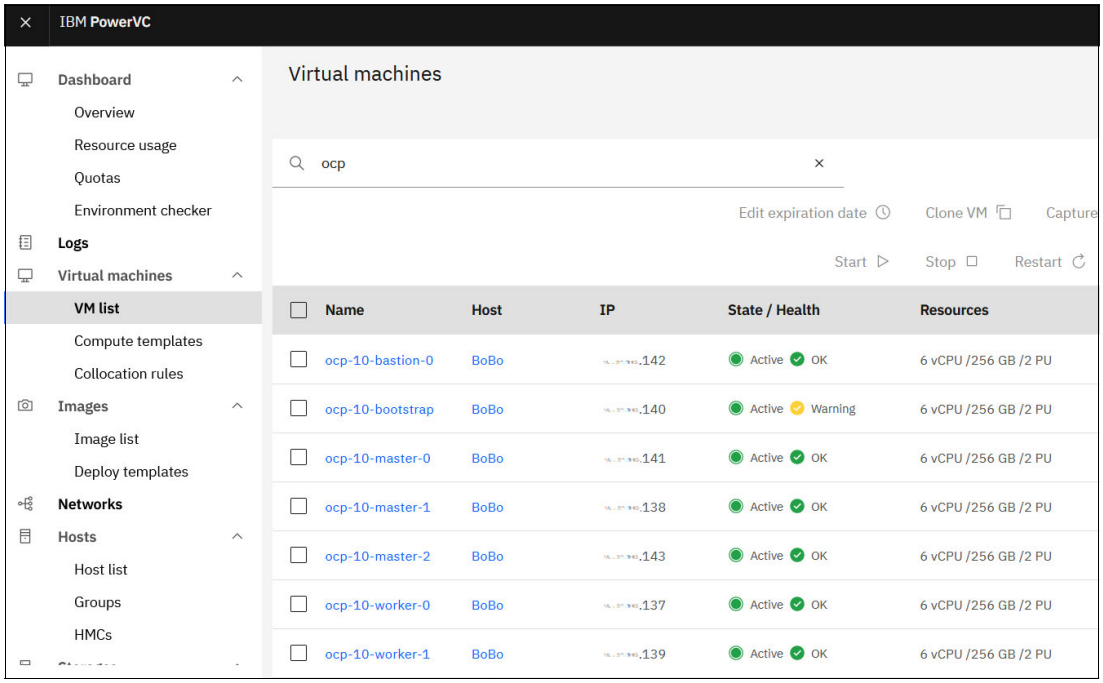


Figure 5-6 The PowerVC UI and the LPARs installations.

13. Open HAProxy by accessing the URL `http://<http_Openshift_bastion_IP><:9000>/` to ensure all services are running.

    Figure 5-7 on page 84 illustrates the details of HAProxy.

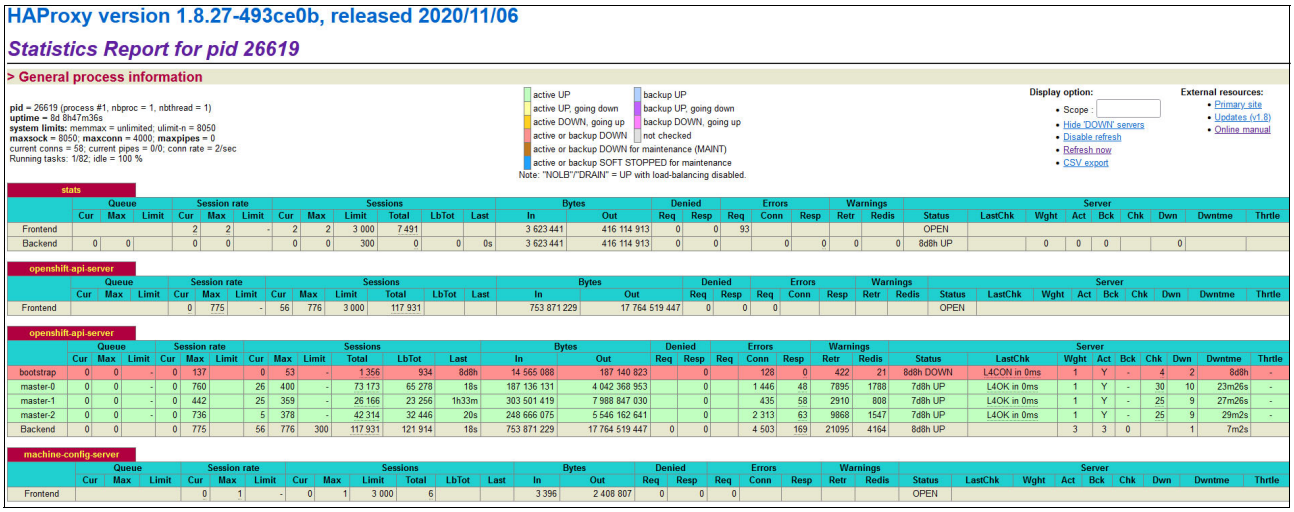


Figure 5-7 The details of HAProxy.

Figure 5-8 illustrates additional details of HAProxy.

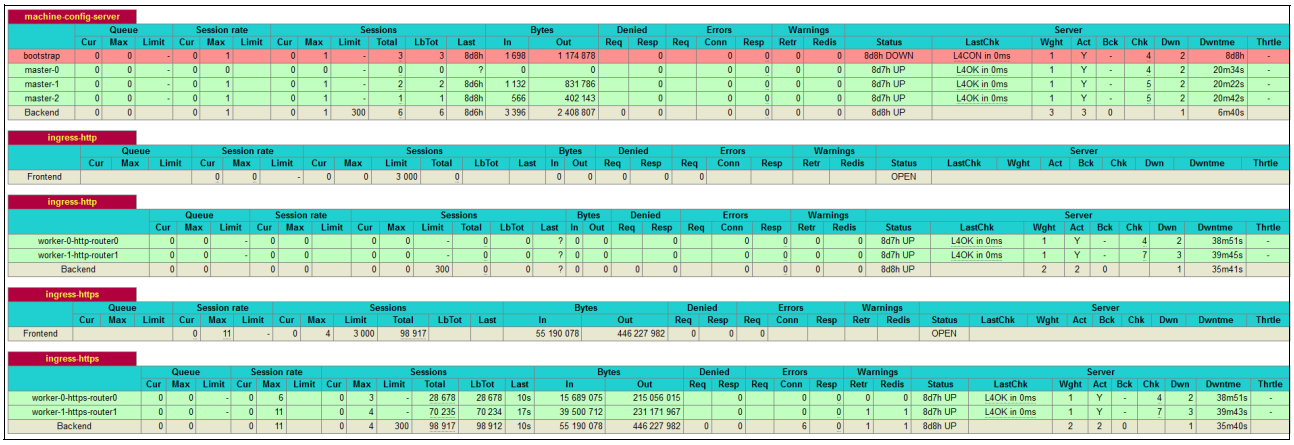


Figure 5-8 Additional details of HAProxy.

14. After the installation is complete, you can monitor the condition of the nodes in your cluster. Open your cluster URL and log in to check the status of your nodes. Figure 5-9 shows the Red Hat OpenShift login screen.

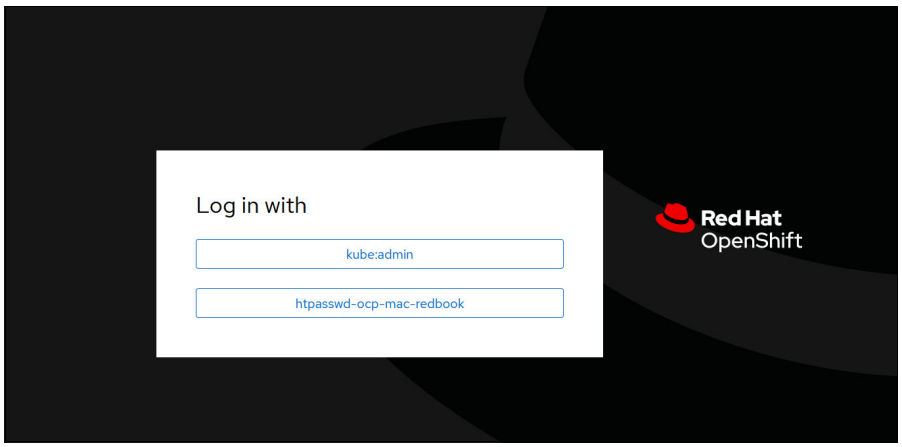


Figure 5-9 The Red Hat OpenShift login screen.

Figure 5-10 shows the status of all nodes.

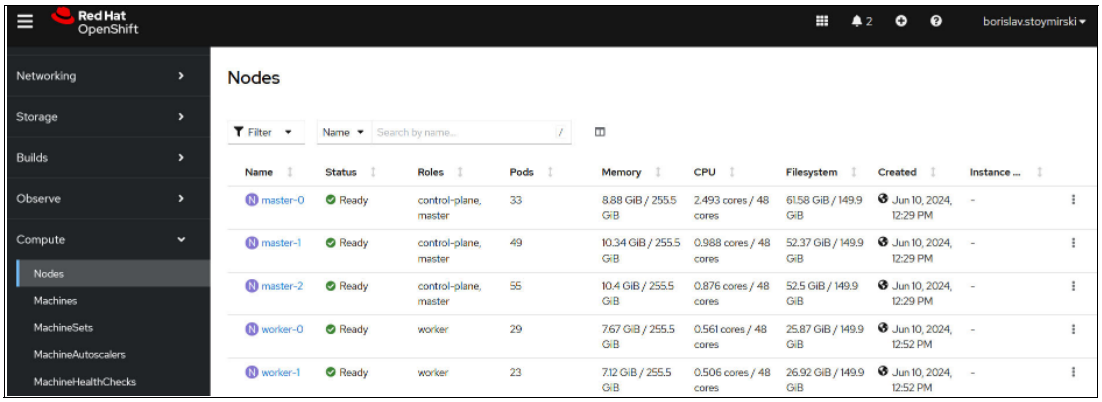


Figure 5-10 Details of all five nodes in the Red Hat OpenShift GUI.

5.2.2 Installing a cluster on IBM Power user-provisioned infrastructure with restricted network

If your enterprise does not have an internet connection – even through a proxy – for security reasons, you can still install Red Hat OpenShift Container Platform on Power servers using the ocp4-upi-powervm project. Locate a bastion host with internet access that is reachable by your cluster. This server functions solely as a registry for the Red Hat OpenShift Platform and is not used as a router or for any other network services. It is only utilized for maintenance and is not part of the production path. For more information, refer to the online documentation on this Red Hat website.

5.2.3 Installing a cluster on IBM Power Virtual Server

IBM Power Virtual Server (PowerVS) is an Infrastructure as a service (IaaS) offering in the IBM Cloud providing configurable, multi-tenant virtual servers and storage integrated with IBM Cloud services. Enterprises can access IBM Power servers utilizing a cloud based pay-as-you-go model to run AIX, IBM i, and Linux workloads including Red Hat OpenShift. T

he ocp4-upi-powervs project offers Red Hat Ansible and Terraform-based automation code designed to facilitate the deployment of Red Hat OpenShift Container Platform (OCP) 4.x on IBM PowerVS.

Internally, this project utilizes the helpernode Ansible playbook for OCP deployment on IBM PowerVS. For more information, please visit the ocp4-upi-powervs project on github, The project is part of the 'Automation for OpenShift on IBM Power Platform' initiative. Additional information on IBM PowerVS and Red Hat OpenShift can be found at 'IBM Power Systems Virtual Server on IBM Cloud' and this Red Hat OpenShift Documentation. For step-by-step installation instructions, see Chapter 6, “Deployment scenarios”, in *Red Hat OpenShift V4.X and IBM Cloud Pak on IBM Power Systems Volume 2*, SG24-8486.

5.2.4 Installing a cluster on IBM Cloud

Red Hat OpenShift Container Platform can be installed on IBM Cloud (x86 based architecture) using installer-provisioned infrastructure. This method utilizes an installation program to set up the necessary cluster infrastructure. After the cluster installation on x86 based infrastructure, you can update to multi-architecture payloads and add Power based worker nodes.

Currently, installing Red Hat OpenShift Container Platform on IBM Cloud with user-provisioned infrastructure is not supported. For more details, see the Red Hat documentation.

Note: IBM Cloud Classic is currently unsupported. For more information on the differences between Classic and VPC infrastructures, see IBM documentation.

5.2.5 Adding an x86 compute node to an IBM Power Red Hat OpenShift cluster

Important: This is considered a Post Installation Step - Day-2 operation**.**

In this section, we will cover the manual method to add an `x86` node to a multi-architecture Red Hat OpenShift cluster with the main architecture being '`ppc64le`'. While automation is the preferred method for most companies – typically achieved using Ansible playbooks and Terraform –we have chosen to present a manual step-by-step process for demonstration purposes to detail the installation. For information on automating the process using Ansible playbooks, refer to “Prerequisites” on page 74.

Prerequisites

- The multi-architecture Red Hat OpenShift CLI (**`oc`** and **`kubectl`**) binaries must be installed.
- The new compute nodes should be added to your DNS including reverse DNS lookup.

Note: Before integrating x86 or ppc64le nodes into a Red Hat OpenShift cluster, ensure the cluster is upgraded to a version that supports the multi-architecture payload.

Steps

The following steps will help you set up an x86 worker node in an IBM Power cluster:

1. Find the latest Power version of the OpenShift client at:
   `https://mirror.openshift.com/pub/openshift-v4/ppc64le/clients/ocp/`

   Remember to get version 4.15.8 or later.

   Download and install the multi-architecture Red Hat OpenShift CLI (**`oc`** and **`kubectl`**) binaries as shown in Example 5-10.

Example 5-10 Installing the multi-architecture Red Hat OpenShift CLI on Linux (ppc64le).

```
# wget
https://mirror.openshift.com/pub/openshift-v4/ppc64le/clients/ocp/4.15.8/openshift-client-linux-
ppc64le-rhel9-4.15.8.tar.gz
# tar -xf openshift-client-linux-ppc64le-rhel9-4.15.8.tar.gz
# cp oc /usr/local/bin/oc
cp: overwrite '/usr/local/bin/oc'? y
# whereis kubectl
kubectl: /usr/local/bin/kubectl
# cp kubectl /usr/local/bin/kubectl
cp: overwrite '/usr/local/bin/kubectl'? y
```

2. Make sure you have Red Hat OpenShift version 4.15.8 or higher and that it supports multiarch as shown in Example 5-11

Example 5-11 Verifying correct Red Hat OpenShift version and multiarch support

```
# oc get clusterversion -o json|jq ".items[0].spec"
{
```

```
  "channel": "stable-4.15",
  "clusterID": "<cluster ID>",
  "desiredUpdate": {
    "architecture": "Multi",
    "force": false,
    "image": "",
    "version": "4.15.8"
  }
}

# oc adm release info -o jsonpath="{ .metadata.metadata}"
{"release.openshift.io/architecture":"multi","url":"https://access.redhat.com/errata/RHSA-2024:1
668"}
```

Figure 5-11 illustrates how to check the Red Hat OpenShift Payload loaded version and architecture.

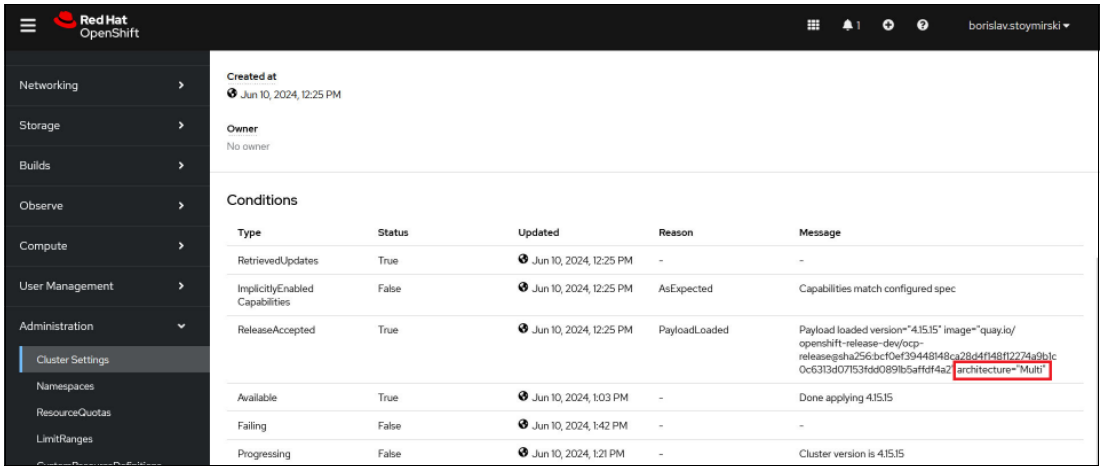


Figure 5-11 Red Hat OpenShift payload version and architecture.

Note: Migration from a multi-architecture payload to a single-architecture payload is not supported. Once a cluster has transitioned to a multi-architecture payload, it cannot revert to a single-architecture update payload.

If this is not the case, update the cluster version and channel to meet the requirements shown in Example 5-11 on page 86 and Figure 5-11. As the next step is to upgrade the payload to 'Multi' architecture and monitor the process, execute the commands shown in Example 5-12.

Example 5-12 Updating the payload to 'Multi' architecture.

```
[root@ocp-10-bastion-0 ~]# oc version
Client Version: 4.15.15
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: 4.15.15
Kubernetes Version: v1.28.9+416ecaf
[root@ocp-10-bastion-0 ~]#
[root@ocp-10-bastion-0 ~]# oc adm upgrade --to-multi-arch
Requested update to multi cluster architecture
[root@ocp-10-bastion-0 ~]# oc adm upgrade status
```

Figure 5-12 shows the single-architecture payload (ppc64le).

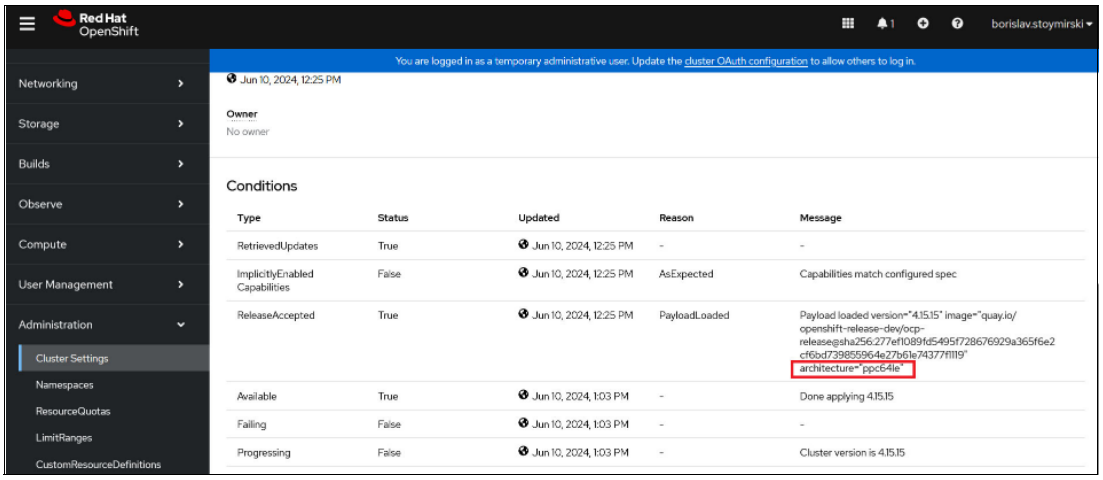


Figure 5-12 Cluster with single-architecture payload (ppc64le).

Figure 5-13 illustrates the upgrade process from a single-architecture payload to a multi-cluster architecture payload after executing the **`oc adm upgrade --to-multi-arch`** command.

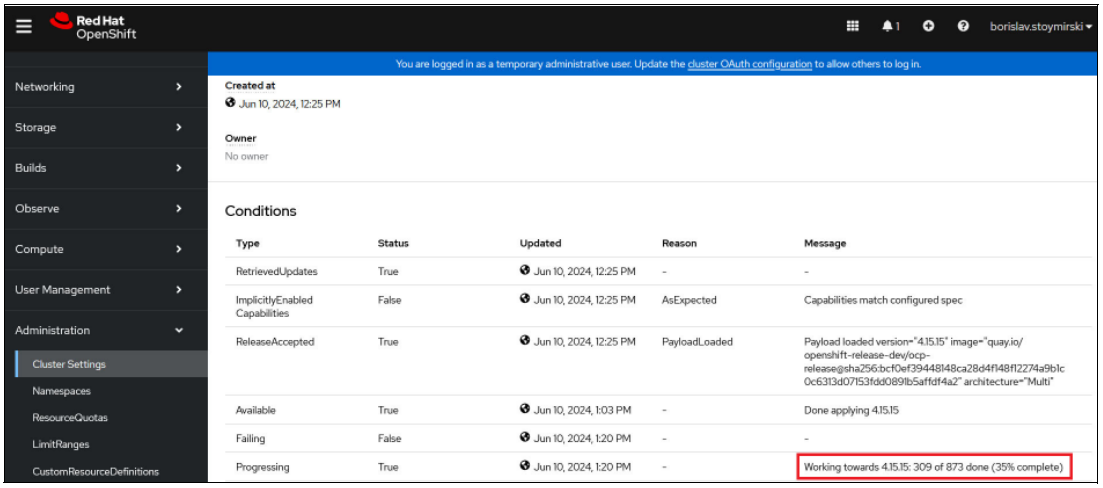


Figure 5-13 Update process in place from single-architecture payload to multi cluster architecture payload.

Figure 5-14 on page 89 shows that the update process to a multi-architecture payload completed successfully.

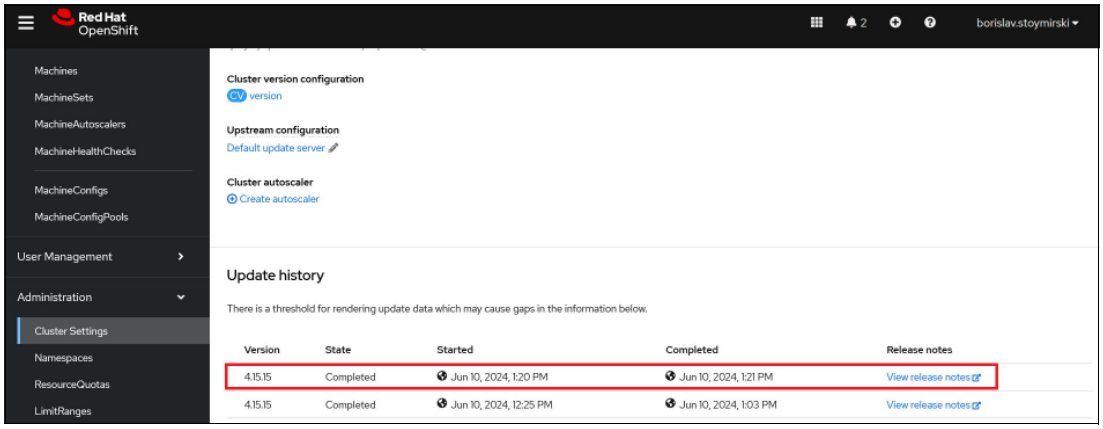


Figure 5-14 The update process from a single-architecture payload to a multi-cluster architecture payload completed successfully.

3. In our example, we will get RHCOS live ISO for booting the x86 VM. The command is shown in Example 5-13.

Example 5-13 Acquiring the ISO for Booting an x86 VM.

```
# oc -n openshift-machine-config-operator get configmap/coreos-bootimages -o
jsonpath='{.data.stream}' | jq -r
'.architectures.x86_64.artifacts.metal.formats.iso.disk.location'
https://rhcos.mirror.openshift.com/art/storage/prod/streams/4.15-9.2/builds/415.92.202402201450-
0/x86_64/rhcos-415.92.202402201450-0-live.x86_64.iso
```

4. Create a new VM and boot the ISO on your x86 architecture server.

Figure 5-15 shows the process of creating a new VM on VMware ESXi.

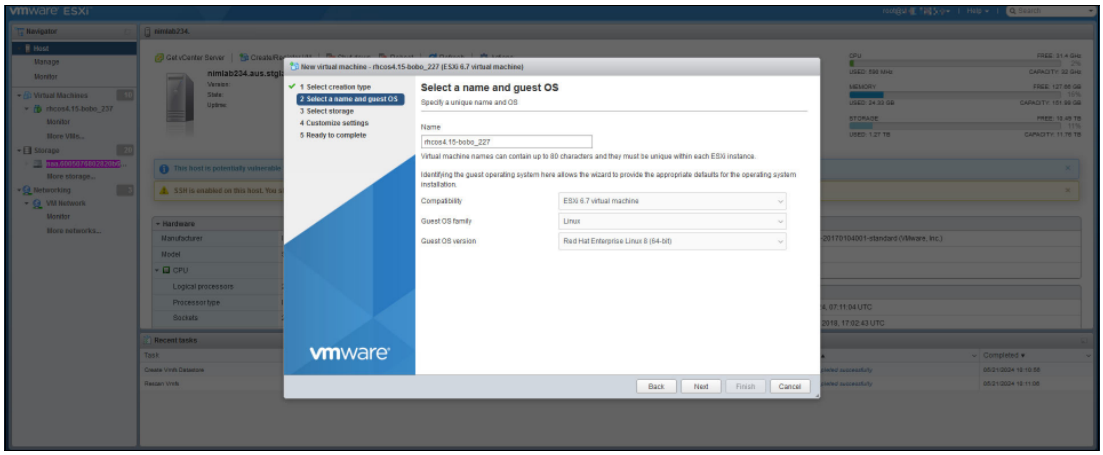


Figure 5-15 Creating a new VM on VMware ESXi.

Figure 5-16 on page 90 illustrates the process of booting the VM from a virtual CD-ROM with the loaded Red Hat CoreOS image.

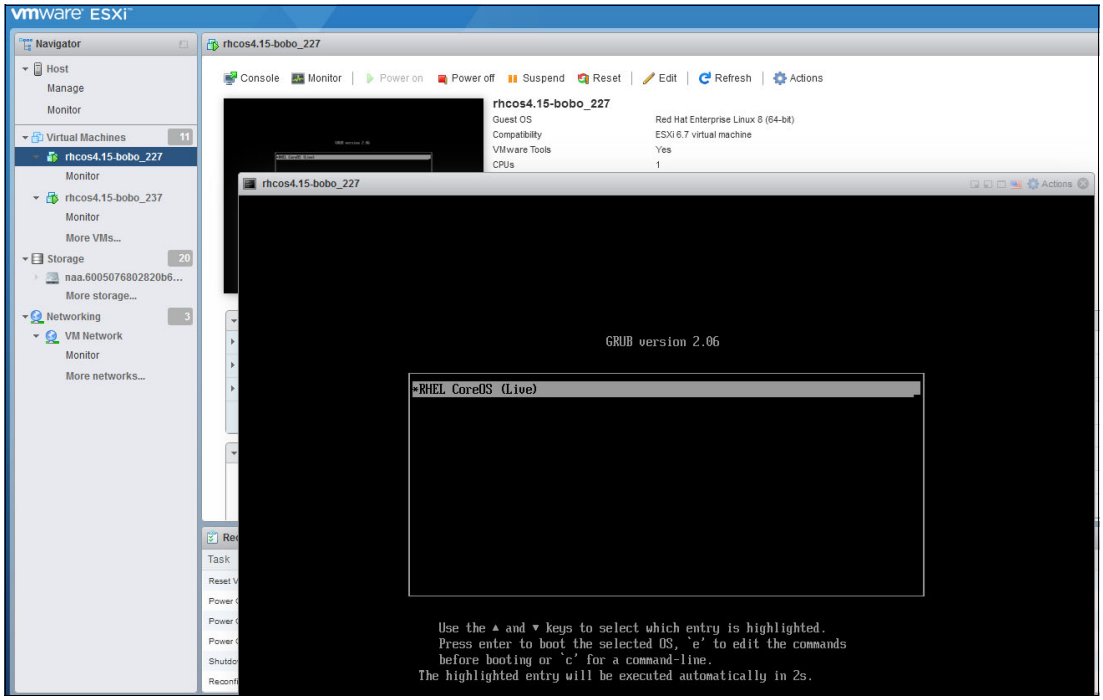


Figure 5-16 Booting the VM from a virtual CD-ROM with the loaded Red Hat CoreOS image.

5. Format the disk `/dev/sda` using the **dd** command before proceeding with the Red Hat CoreOS installation as shown in Example 5-14.

Example 5-14 Formatting the disk `/dev/sda` using the dd command.

```
sudo dd if=/dev/zero of=/dev/sda bs=4000 count=1000
1000+0 records in
1000+0 records out
```

IMPORTANT: Pay special attention when using the **dd** command and its syntax. Failure to do so might permanently destroy the disk data.

6. Set up the network using ***nmcli*** as shown in Example 5-15:

Example 5-15 Set up network connection using nmcli.

```
nmcli conn modify "Wired connection 1" ipv4.addresses <IP adress>/<CIDR>
nmcli conn modify "Wired connection 1" ipv4.method manual
nmcli conn modify ipv4.dns <DNS server>   # in some cases could be setup as Bastion IP
nmcli conn modify ipv4.gateway <ip_gateway>
nmcli conn up "Wired connection 1”
```

7. From your Red Hat OpenShift cluster console make your ignition file available through `http` as shown in Example 5-16.

Example 5-16 Creating the ignition file.

```
# oc extract -n openshift-machine-api secret/worker-user-data-managed --keys=userData --to=- >
/var/www/html/ignition/computex86.ign
```

The ignition file will now be available at:

http://<http_IP><:_port_if_needed>/ignition/computex86.ign

Important: This process assumes that you have the DNS set up properly. If DNS is not set up properly, then you may have issues.

8. Edit the ignition file to set the hostname according to your requirements, as shown in Example 5-17.

Example 5-17 Setting the hostname in the ignition file.

```
# nano computex86.ign
"storage":{"files": [{"path":"/etc/hostname","contents":{"source":"data:,worker-x86-2"},"mode":
420}]},"systemd":{}}
```

9. The following steps and commands, shown in Example 5-18, will set up additional environment variables and initiate the installation of the x86 worker node.

Example 5-18 Final setup and install.

```
hostnamectl set-hostname <node_hostname>
timedatectl set-timezone <time_zone_your_cluster>

cat >/etc/resolv.conf <<EOL
search <doamin_cluster>
nameserver <ip_dns_server>
EOL

wrksha=$(curl -s http://<ignition_file_url.ign> | sha512sum | awk '{print $1;}')

iptables -I INPUT -m state --state NEW -p tcp --destination-port 22623 -j ACCEPT

coreos-installer install --copy-network -p vmware --ignition-url=http://<ignition_file_url.ign>
--ignition-hash=sha512-$wrksha /dev/sda
```

10. Reboot the VM from the disk drive without the live ISO.

5.3 Post-installation configuration and verification

Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine. It may take up to 15 minutes for the certificates to appear as pending. Ensure these CSRs are approved, or approve them manually if needed. Client requests should be approved first, followed by server requests.

1. Monitor for pending CSRs and verify that the client requests for each machine added to the cluster are approved. See Example 5-19.

Example 5-19 CSRs approval process.

```
# oc get csr
NAME        AGE   SIGNERNAME                                    REQUESTOR REQUESTEDDURATION   CONDITION
csr-ftk57   39s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-qzl69   15m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending

# 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-ftk57 approved
certificatesigningrequest.certificates.k8s.io/csr-qzl69 approved
```

```
# oc get csr
NAME        AGE   SIGNERNAME                                    REQUESTOR REQUESTEDDURATION   CONDITION
csr-ftk57   57s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-qzl69   15m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
```

In Example 5-19 on page 91 the command to approve the certificates could also use the command: **oc get csr -oname | xargs** instead of the lengthy version shown.

During the CSR approval process, you might see additional approved CSRs in the list.

Note: Since CSRs rotate automatically, ensure you approve your CSRs within an hour of adding the machines to the cluster. If not approved within this time frame, the certificates will rotate, resulting in multiple certificates for each node. You will need to approve all of these certificates.

2. Validate that the node has been added to the cluster with a '***Ready***' status and check its architecture using the following command:

   # oc get nodes -o wide

   The results are shown in Example 5-20.

Example 5-20 The node details show that two additional nodes with x86 architecture have been successfully added.

```
NAME                                STATUS   ROLES                  AGE    VERSION           INTERNAL-IP    EXTERNAL-IP
OS-IMAGE                                                KERNEL-VERSION                  CONTAINER-RUNTIME
master-0       Ready    control-plane,master   23d   v1.28.9+416ecaf   x.x.x.141     <none>        Red Hat Enterprise Linux
CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.ppc64le   cri-o://1.28.6-7.rhaos4.15.git7Oc2e96.el9
master-1       Ready    control-plane,master   23d   v1.28.9+416ecaf   x.x.x.138     <none>        Red Hat Enterprise Linux
CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.ppc64le   cri-o://1.28.6-7.rhaos4.15.git7Oc2e96.el9
master-2       Ready    control-plane,master   23d   v1.28.9+416ecaf   x.x.x.143     <none>        Red Hat Enterprise Linux
CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.ppc64le   cri-o://1.28.6-7.rhaos4.15.git7Oc2e96.el9
worker-0       Ready    worker                 23d   v1.28.9+416ecaf   x.x.x.137     <none>        Red Hat Enterprise Linux
CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.ppc64le   cri-o://1.28.6-7.rhaos4.15.git7Oc2e96.el9
worker-1       Ready    worker                 23d   v1.28.9+416ecaf   x.x.x.139     <none>        Red Hat Enterprise Linux
CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.ppc64le   cri-o://1.28.6-7.rhaos4.15.git7Oc2e96.el9
worker-x86-2   Ready    worker                 23d   v1.28.9+416ecaf   x.x.x.237     <none>        Red Hat Enterprise Linux
CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64    cri-o://1.28.6-7.rhaos4.15.git7Oc2e96.el9
worker-x86-3   Ready    worker                 23d   v1.28.9+416ecaf   x.x.x.227     <none>        Red Hat Enterprise Linux
CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64    cri-o://1.28.6-7.rhaos4.15.git7Oc2e96.el9
```

3. Monitor your cluster within the next 24 hours, as additional CSRs may appear and require manual approval as shown in Example 5-21.

Example 5-21 Additional CSRs approval process.

```
# oc get csr
NAME        AGE    SIGNERNAME                                    REQUESTOR REQUESTEDDURATION   CONDITION
csr-242nq   72m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-469nl   41m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-4x4s8   87m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-82zxl   164m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-cc4dc   149m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
```

```
csr-gsdlg   118m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-h6ltc   179m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-j5cs4   134m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-npr7b   26m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-phcdr   10m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-qtftw   179m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-rtvpb   57m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending
csr-xm9sf   103m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>              Pending

# 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-242nq approved
certificatesigningrequest.certificates.k8s.io/csr-469nl approved
certificatesigningrequest.certificates.k8s.io/csr-4x4s8 approved
certificatesigningrequest.certificates.k8s.io/csr-58xvp approved
certificatesigningrequest.certificates.k8s.io/csr-82zxl approved
certificatesigningrequest.certificates.k8s.io/csr-cc4dc approved
certificatesigningrequest.certificates.k8s.io/csr-gsdlg approved
certificatesigningrequest.certificates.k8s.io/csr-h6ltc approved
certificatesigningrequest.certificates.k8s.io/csr-j5cs4 approved
certificatesigningrequest.certificates.k8s.io/csr-npr7b approved
certificatesigningrequest.certificates.k8s.io/csr-phcdr approved
certificatesigningrequest.certificates.k8s.io/csr-qtftw approved
certificatesigningrequest.certificates.k8s.io/csr-rtvpb approved
certificatesigningrequest.certificates.k8s.io/csr-xm9sf approved
[root@ocp-10-bastion-0 ~]#

# oc get csr
NAME        AGE    SIGNERNAME                                    REQUESTOR REQUESTEDDURATION   CONDITION
csr-242nq   77m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-469nl   46m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-4x4s8   93m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-58xvp   20s    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-82zxl   169m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-cc4dc   154m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-gsdlg   124m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-h6ltc   3h4m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-j5cs4   139m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-npr7b   31m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-phcdr   15m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-qtftw   3h4m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-rtvpb   62m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
```

```
csr-xm9sf   108m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none> Approved,Issued
csr-xqhsm  16s   kubernetes.io/kubelet-serving           system:node:worker-x86-2
<none>              Pending

# 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-xqhsm approved
```

Ensure that your nodes are properly authorized using the certificate request process built in to OpenShift. The certificate requet process is critical in ensuring that only the nodes you authorize are allowed to be part of your cluster.

Part 3

Deep Dive into Red Hat OpenShift on IBM Power

The previous sections provided insight into the basics of Red Hat OpenShift and how the new multiple architecture cluster capabilities can help you create a more efficient and flexible solution for managing your cloud workloads, whether they are running on-premises or in the cloud.

This section provides a more in-depth look at how to implement Red Hat OpenShift and the multiple architecture cluster technology in your IBM Power system.

The following topics are included in this section:

6

Building and Managing Containers

Adding multi-architecture support to a Red Hat OpenShift cluster requires careful consideration of application compatibility across different architectures within the cluster. As such, it is crucial to comprehend the supported architectures for your applications and their execution environments. This knowledge will help ensure seamless container build and management throughout the cluster.

This chapter will cover the following topic:

6.1 Container images for multiple architectures

Containers are lightweight software packages that contain all the dependencies required to execute the contained software application. These dependencies include things like system libraries, external third-party code packages, and other operating system level applications. The dependencies included in a container exist in stack levels that are above the operating system.

A container image once built is immutable, which means that a container image will not be modified during its lifecycle, no updates, no patches, no configuration changes. For any changes, the Image has to be rebuilt with the necessary changes to the configuration file.

As the demand for containerization continues to grow, developers face the challenge of deploying applications across various architectures and platforms. The Red Hat OpenShift containerization platform provides a solution with its support for multi-architecture container images.

In this chapter we will explore the process of building multi-architecture container images and the benefits it offers.

6.1.1 Understanding multi-architecture containers

Previously, container images were tailored for a particular architecture like x86, ARM, IBM Power Systems, or IBM Z. But due to the growth of various hardware architectures and cloud platforms, there is a pressing demand for multi-architecture support. Multi-architecture containers allow you to deploy the same image across different architectures smoothly.

6.1.2 Container manifests

To achieve multi-architecture support, Red Hat OpenShift introduced the concept of manifests. A manifest is a file that describes a set of images for different platforms, collectively known as a manifest list. It provides a way to associate multiple image layers with a single reference. Docker automatically selects the appropriate image for the target platform during the container deployment.

6.1.3 Building multi-architecture images

To build multi-architecture container images, we need to follow these steps:

1. Create Dockerfiles: Start by creating separate Dockerfiles for each architecture you want to support. These Dockerfiles will define the instructions for building the container images.
2. Build images: Use Docker's build command to build the images for each architecture. Specify the appropriate Dockerfile for each build, along with the target architecture. For example, you can use the --platform flag to specify the architecture explicitly.
3. Tag images: Once the images are built, tag them with the appropriate platform-specific tags. These tags typically include the architecture name, such as AMD64, arm64, or ppc64le.
4. Push images: Push the tagged images to a Docker registry or any other container registry of your choice. Make sure the registry supports multi-architecture manifest lists.
5. Create a manifest list: Finally, create a manifest list that references the different platform-specific images.

The manifest list acts as a single entry point for pulling and deploying the multi-architecture container. Use the podman manifest create command to create the manifest list.

6. Push manifest list: Push the manifest list to the registry, linking it to the appropriate images for each architecture. This ensures that Docker pulls the correct image based on the target platform.

6.1.4 Testing multi-architecture containers

To guarantee seamless operation across various systems, it is vital to meticulously test multi-architecture containers. Establish a testing setup comprising physical or virtual machines of the targeted platforms. Then, pull the container images from the repository and install them on every platform to validate their functionalities and performance.

6.1.5 Benefits of multi-architecture containers

Deploying multi-architecture containers offers several benefits:

Fit for Purpose: Different architectures provide different capabilities. Using multiple architecture clusters allows you to take advantage of those unique features only provided by a specific architecture.

Portability: Applications packaged as multi-architecture containers can run seamlessly on different hardware architectures, making it easier to migrate between platforms or cloud providers.

Scalability: With multi-architecture support, scaling applications becomes more flexible, as you can leverage a diverse range of hardware architectures.

Development efficiency: You can build, test, and distribute applications across various architectures simultaneously, reducing the time and effort required for cross-platform deployment.

Future-proofing: Multi-architecture containers future-proof your applications, ensuring they can run on new architectures as they emerge.

6.1.6 Scheduling applications on the appropriate architecture machine

In a multi-architecture cluster, where some nodes may have different architectural configurations, it's crucial that the container images used within the cluster align with the node's architecture. To accomplish this, each image must specify the architecture it supports. This alignment ensures that pods are correctly assigned to the appropriate node and match their respective image architecture.

To address the case where an application is solely compatible with specific platforms in a multi arch clusters, Red Hat OpenShift offers node affinity, taints, and tolerances. These features enable precise control over scheduling, ensuring that applications land on designated target platforms.

Using node affinity to schedule workloads on a node

You can allow a workload to be scheduled on only a set of nodes with architectures supported by its images. To do so you can set the spec.affinity.nodeAffinity field in your pod’s template specification. This is demonstrated in Example 6-1 on page 100 where the nodeAffinity is set for ppc64le for IBM Power architecture nodes.

Example 6-1 Example deployment with the nodeAffinity set to certain architectures

```
apiVersion: apps/v1
kind: Deployment
metadata: # ...
spec:
   # ...
  template:
     # ...
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - ppc64le
```

Understanding taints and tolerations

Taints and tolerations allow the node to control which pods should (or should not) be scheduled on them. A taint allows a node to refuse a pod to be scheduled unless that pod has a matching toleration.

You apply taints to a node through the Node specification (NodeSpec) and apply tolerations to a pod through the Pod specification (PodSpec). When you apply a taint to a node, the scheduler cannot place a pod on that node unless the pod can tolerate the taint. Example 6-2 shows specifying a taint in a node specification. In this case the node is not a Power node (architecture ppc64le) and this taint would keep a ppc64le image from being scheduled on the node.

Example 6-2 Example taint in a node specification

```
apiVersion: v1
kind: Node
metadata:
  name: my-node
#...
spec:
  taints:
  - effect: NoSchedule
    key: kubernetes.io/arch
    value: ppc64le
#...
```

Example 6-3 shows the toleration specified in the Pod specification.

Example 6-3 Example toleration in a Pod spec

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
#...
```

```
spec:
  tolerations:
  - key: "kubernetes.io/arch"
    operator: "Equal"
    value: "ppc64le"
    effect: "NoExecute"
    tolerationSeconds: 3600
#...
```

Multiarch Tuning Operator

The Multiarch Tuning Operator is available as a technical preview in Red Hat OpenShift 4.16. Technology Preview features are not supported by Red Hat production service level agreements, instead they provide an early view of upcoming product updates.

As multi-architecture compute nodes become more prevalent in OpenShift clusters, users and administrators encounter new challenges when deploying workloads. The main issue is that the scheduler does not account for the compatibility of container images with the CPU architectures of the nodes. Consequently, workloads may be compatible with only a subset of the cluster's architectures.

Typically, users address this by manually adding affinity rules to pod specifications, ensuring that pods are scheduled on nodes where their image binaries can run. However, this method significantly impacts the user experience: it ties compatible architectures to the pod specification, making it difficult to scale and maintain.

The Multiarch Tuning Operator enhances the operational experience within multi-architecture clusters, and single-architecture clusters that are migrating to a multi-architecture compute configuration. This Operator implements the `clusterpodplacementconfigs` custom resource (CR) to support architecture-aware workload scheduling. Scheduling gates inspect the container image to patch Pod specs so that a NodeAffinity is used to influence workload scheduling.

- Architecture-aware scheduling

  Automatically patch pods by adding the node affinity terms with the information about the architectures supported by the workloads.

- Support migration from single-arch to multi-arch OpenShift

  Support migration from single-arch to multi-arch OpenShift ensuring any previously deployed workloads are placed in nodes supporting the secondary architectures, without the need for manual rollouts or changes to the cluster infrastructure.

Note: The openshift-*, kube-* and hypershift-* namespaces are excluded from management by the Multiarch Tuning Operator.

For more information, see Managing workloads on multi-architecture clusters by using the Multiarch Tuning Operator.

6.2 Installing ODF and local storage operator

OpenShift Data Foundation (ODF) provides comprehensive multi-cloud storage for containerized workloads. It is deeply integrated into Red Hat OpenShift and designed to handle persistent storage requirements with efficiency and resilience.

The Local Storage Operator (LSO), on the other hand, allows users to manage local storage devices directly from the Red Hat OpenShift environment.

6.2.1 Installing OpenShift Data Foundation

Before installing OpenShift Data Foundation (ODF) ensure you have an OpenShift cluster running with appropriate administrative privileges. Also confirm that your cluster meets the hardware and software requirements specific to ODF, including node sizing, network configuration, and available storage.

Command Line Installation

To install ODF from the command line, follow these steps:

1. Create the project using this command:

   `oc adm new-project openshift-storage`

2. Next create the yaml file shown in Example 6-4 to setup the OperatorGroup and Subscription.

Example 6-4 odf.yaml

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-storage-operatorgroup
  namespace: openshift-storage
spec:
  targetNamespaces:
  - openshift-storage

---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: odf-operator
  namespace: openshift-storage
spec:
  channel: stable-4.14
  installPlanApproval: Automatic
  name: odf-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

3. Define the object with this command:

   `oc apply -f odf.yaml`

> **Note:** ODF takes a few minutes to install.

4. Verify the operator by checking subscriptions in the openshift-storage namespace as shown in Example 6-5.

Example 6-5 Verify the operator

```
[redhat@bastion ~]$ oc get subscriptions -n openshift-storage
```

```
NAME
PACKAGE                    SOURCE             CHANNEL
mcg-operator-stable-4.14-redhat-operators-openshift-marketplace
mcg-operator               redhat-operators   stable-4.14
ocs-operator-stable-4.14-redhat-operators-openshift-marketplace
ocs-operator               redhat-operators   stable-4.14
odf-csi-addons-operator-stable-4.14-redhat-operators-openshift-marketplace
odf-csi-addons-operator    redhat-operators   stable-4.14
odf-operator
odf-operator               redhat-operators   stable-4.14
```

5. When the ODF operator is available, enable the Web Console plug-in using this command:

   oc patch console.operator cluster -n openshift-storage --type json -p '[{"op": "add", "path": "/spec/plugins", "value": ["odf-console"]}]'

Web console installation

If the Web console is not already intalled, the following steps will install it:

1. Navigate to the OpenShift Container Platform web console.
2. Go to the "Operators" section, then "OperatorHub".
3. Search for "OpenShift Data Foundation" and select it.
4. Click "Install" and choose the appropriate namespace (typically openshift-storage), approval strategy, and update channel as per your cluster configuration.

6.2.2 Local Storage Operator

ODF can use a variety of back-end storage devices and to simplify the discovery of block devices directly attached to infrastructure or worker nodes in the cluster, it is recommended to use the Local Storage Operator (LSO). The example below assumes block devices (HDD/SSD disks) are locally attached to each node. To install the LSO through the command line follow these steps:

1. Create a project to install the LSO in.

   LSO is installed, by default, into its own project called “openshift-local-storage”. using a user profile with cluster admin privileges, create the project with this command:

   oc adm new-project openshift-local-storage

   By default, LSO discovers locally attached devices on worker nodes only. To expand the scope for LSO to include infrastructure nodes, create the following namespace annotation:

   oc annotate namespace openshift-local-storage openshift.io/node-selector=''

Note: Every project object in Red Hat OpenShift has a corresponding namespace Kubernetes object. All custom annotations should be created in the respective namespace object.

2. Create an Operator Group for the LSO.

   To create an operator group for the local storage operator follow these steps:

   a. Create a yaml file as shown In Example 6-6 on page 104 that will be used to create an operator group for the Local Storage Operator.

Example 6-6 lso-og.yaml used to define the operator group

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
name: local-operator-group
namespace: openshift-local-storage
spec:
targetNamespaces:
- openshift-local-storage
```

b. Define the object using this command:

oc apply -f lso-og.yaml

c. Verify the object was created by running the command shown in Example 6-7 and look for output as seen there.

Example 6-7 Verify that the operator group id defined

```
[redhat@bastion ~]$oc get OperatorGroup -n openshift-local-storage
NAME                           AGE
openshift-local-storage        12d
```

d. To install the LSO, create a subscription using the definition shown in Example 6-8.

Example 6-8 Define lso-sub.yaml to create a subscription for the LSO

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: local-storage-operator
  namespace: openshift-local-storage
spec:
  channel: stable
  installPlanApproval: Automatic
  name: local-storage-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

e. Define the object using this command:

oc apply -f lso-sub.yaml

f. Verify the object exists and the operator pods are running using the command shown in Example 6-9.

Example 6-9 Verify installation

```
[redhat@bastion ~]$oc get csvs -n openshift-local-storage
NAME DISPLAY VERSION REPLACES PHASE
local-storage-operator.v4.14.0-202404250639 Local Storage 4.14.0-202404250639
Succeeded

[redhat@bastion ~]$oc get pods -n openshift-local-storage
NAME                                     READY   STATUS    RESTARTS      AGE
local-storage-operator-68bb94c7d9-8n6wr   1/1     Running   1 (30h ago)   3d9h
```

Note: When a discovery session is run in the LSO there will be more pods created in this namespace to reflect the discovery pods running on all nodes where the LSO is allowed to discover devices.

3. To run the discovery process, prepare nodes for discovery using the LSO.
   a. All nodes that contain locally attached devices to be used by ODF should be labeled using the following command:

      `oc label node <NodeName> cluster.ocs.openshift.io/openshift-storage=''`

   b. Create the discovery object using the yaml file shown in Example 6-10:

Example 6-10 lso-discovery.yaml

```
apiVersion: local.storage.openshift.io/v1alpha1
kind: LocalVolumeDiscovery
metadata:
  name: auto-discover-devices
  namespace: openshift-local-storage
spec:
  nodeSelector:
    nodeSelectorTerms:
      - matchExpressions:
        - key: cluster.ocs.openshift.io/openshift-storage
          operator: Exists
```

   c. Apply the discovery file using this command:

      `oc apply -f lso-discovery.yaml`

      Alternatively, nodes for discovery can be explicitly specified through the Kubernetes nodeSelector.

Defining Resources

To create StorageSystem and corresponding storage classes, use the Red Hat OpenShift Web Console. While it is possible to define a storage system through a combination of yaml definitions, the web GUI provides a simple three-step process. First **`Login`** to the Red Hat OpenShift Web Console as a user with cluster admin privileges. Then follow these steps:

1. Navigate to **Operators** → **Installed Operators** and choose **OpenShift Data Foundation**
2. Under **Provided APIs** → **StorageSystem** click on "**Create instance**"
3. Under "**Backing storage type**" choose **option 2, "Create a new StorageClass using local storage devices**" and click through the wizard to enable ODF on devices discovered by the LSO.

Monitoring and protecting your local storage

Both ODF and Local Storage Operator come with monitoring tools that should be configured to alert administrators about performance issues, capacity limits, and operational status. It is also important to implement robust backup and disaster recovery strategies to protect the data managed by ODF and local storage. To further protect your data, ensure that access to storage management interfaces is secured and that role-based access control (RBAC) policies are correctly applied

6.3 Using Buildah and Podman on Power Systems

Buildah and Podman are two complementary open-source projects that are available on most Linux platforms and both projects reside at GitHub with Buildah and Podman. Both Buildah and Podman are command line tools that work on Open Container Initiative (OCI) images and containers.

Podman

Podman (short for pod manager) is an open source tool for developing, managing, and running containers. Developed by Red Hat engineers along with the open source community, Podman manages the entire container ecosystem using the libpod library.

Podman's daemon-less and inclusive architecture makes it an accessible, security-focused option for container management. Its accompanying tools and features, such as Buildah and Skopeo, let developers customize their container environments to suit their needs. Developers can also take advantage of Podman Desktop, a graphical user interface (GUI) for using Podman in local environments.

Users can run Podman on various Linux distributions, such as Red Hat Enterprise Linux, Fedora, CentOS, and Ubuntu.

Buildah

Buildah's flexibility in creating images without requiring Dockerfiles enables the incorporation of additional scripting languages into the build process. This streamlined approach also enhances efficiency by utilizing external build tools outside the container image itself. As a result, developers can accelerate innovation and bring new ideas to life more swiftly. Images can be constructed and customized efficiently, minimizing the need for extensive setup procedures. Buildah allows you to:

- Inspect, verify, and modify images.
- Push containers and images from local storage to a public or private registry or repository.
- Push or pull images from the Docker Hub.
- Remove locally stored container images.
- Mount and unmount a working container's root file system.
- Use the updated contents of a container's root filesystem as a filesystem layer for a new image.

Differentiating Buildah and Podman

Buildah and Podman differ in their focus. Buildah excels at creating Open Container Initiative (OCI) images through its specialized capabilities. Its commands mimic those in a Dockerfile, enabling image creation with or without Dockerfiles while avoiding the need for root permissions. Buildah aims to offer a low-level core utilities interface for building images, fostering integration of various scripting languages into the construction process. Adhering to a straightforward fork- exec model, Buildah relies on a robust golang API, which can be incorporated into other tools.

Podman, on the other hand, concentrates on managing and modifying OCI images, including pulling, tagging, and creating, running, and maintaining containers derived from those images. When constructing container images utilizing Dockerfiles, Podman leverages Buildah's go API and can be installed separately from Buildah.

The primary distinction between Buildah and Podman lies in their interpretation of a container. Podman permits the formation of traditional containers, designed for extended usage. In contrast, Buildah containers serve merely to facilitate content addition to the container image. By analogy, the buildah run command replicates the RUN directive in a Dockerfile, whereas the podman run command mirrors the docker run function. Due to these distinctions and their respective storage approaches, Podman containers cannot be viewed inside Buildah nor vice versa.

In summary, Buildah streamlines OCI image creation, while Podman facilitates image management and maintenance in a production setting using customary container CLI commands. For more details, see the Container Tools Guide.

6.4 Managing container registries

A container image consists of the files and elements required to operate an application. Unlike traditional virtual machines (VMs), containers are streamlined bundles of software that ride atop the Linux OS. Containers can be replicated to adapt to fluctuating workloads. An open-source utility akin to Buildah enables creating container images. To facilitate storing, sharing, and accessing these images during development, a container repository serves as a hub.

A container registry functions as a digital warehouse for developers to store container images and disseminate them through the process of transferring (pushing) to the registry and retrieving (pulling) onto another system, such as a Kubernetes cluster.

Two categories of container registries exist: public and private. Public registries are typically utilized by individual contributors or small teams seeking expediency upon initial setup. Nevertheless, as organizational growth occurs, public registries can introduce additional intricate security challenges including patch management, data privacy, and access controls.

Private registries furnish a solution for integrating security and privacy within enterprise container image storage, either hosted externally or internally. These exclusive repositories usually accompany enhanced security features and professional assistance. Many cloud service providers supply private image registry solutions:

- Google offers the Google Container Registry
- Amazon provides Amazon Elastic Container Registry (ECR)
- Microsoft hosts the Azure Container Registry
- IBM operates IBM Cloud Container Registry

Red Hat OpenShift facilitates integration with alternative private registries you might already own, such as JFrog's Artifactory and Sonatype Nexus.

Integrated Red Hat OpenShift Container Platform registry

Red Hat OpenShift Container Platform provides a built-in container image registry that runs as a standard workload on the cluster. The registry is configured and managed by an infrastructure Operator. It provides an out-of-the-box solution for users to manage the images that run their workloads, and runs on top of the existing cluster infrastructure. This registry can be scaled up or down like any other cluster workload and does not require specific infrastructure provisioning. In addition, it is integrated into the cluster user authentication and authorization system, which means that access to create and retrieve images is controlled by defining user permissions on the image resources.

The registry is typically used as a publication target for images built on the cluster, as well as being a source of images for workloads running on the cluster.

When a new image is pushed to the registry, the cluster is notified of the new image and other components can react to and consume the updated image.

Image data is stored in two locations. The actual image data is stored in a configurable storage location, such as cloud storage or a filesystem volume. The image metadata, which is exposed by the standard cluster APIs and is used to perform access control, is stored as standard API resources, specifically images and imagestreams.

Red Hat Quay registries

If you need another option for an enterprise-quality container image registry, Red Hat Quay is available – both as a hosted service and as software you can install in your own data center or cloud environment. Advanced registry features in Red Hat Quay include geo-replication, image scanning, and the ability to roll back images.

7

Deploying Applications

This chapter discusses essential factors to consider when deploying applications in a Red Hat OpenShift environment utilizing Multi-Architecture Cluster features. Crucial insights into scheduling applications across various architecture components of the cluster are presented. Key topics include load balancing, high availability, and monitoring strategies.

The following topics are presented:

7.1 Deploying multi-architecture applications

In today's diversifying computing landscape, numerous organizations operate systems based on various central processing unit (CPU) architectures, including x86_64, ARM, and IBM Power Systems. To address this challenge, Red Hat OpenShift offers a robust solution for deploying applications that run seamlessly across multiple architectures without compromising compatibility or performance. This chapter outlines the steps required to deploy multi-architecture applications using Red Hat OpenShift.

7.1.1 Understanding Multiple Architecture Support

Multiple architecture support enables a single Red Hat OpenShift Cluster's control plane and worker nodes to operate across diverse hardware architectures concurrently. To ensure applications' compatibility with their target hardware, scheduling them onto appropriate worker nodes hosting compatible architected infrastructure is mandatory. Application container images should be built according to the distinct hardware architectures they aim to execute upon.

Adopting multiple architecture support offers several benefits, including leveraging various public clouds and on-premises hardware resources without being confined to specific vendors, thereby minimizing costs and enhancing overall efficiency. Key components associated with managing a multi-architecture cluster encompass.

Container image management

When managing a multiple architecture environment, it's crucial to have a reliable container image management system. This setup guarantees that, upon scheduling a container onto a node in the cluster, the correct application code matching the node's architecture is utilized. This objective is achieved through two primary methods in Red Hat OpenShift:

- Architecture-Specific Tags: Container images within the same repository can be labeled with architecture-specific tags. These tags point to builds of the container tailored for specific architectures, such as:
  - app:latest-amd64
  - app:latest-arm64
  - app:latest-ppc64le
- Manifest Lists: Red Hat OpenShift allows for the utilization of manifest lists, which act as a higher-level abstraction containing multiple container images, each designed for a particular architecture. A manifest list exists to allow a single image to support multiple platform architectures. Upon pulling a manifest list from a registry, the system then needs to choose a manifest in the list that aligns with the architecture of the node to which it is assigned.

Node selectors and affinity

One option for ensuring that the application scheduled on a node in a multiple architecture cluster is to use node selectors and node affinity labels.

- Node Labels: In Red Hat OpenShift, each node can be labeled with its architecture type, such as kubernetes.io/arch=amd64 or kubernetes.io/arch=ppc64le. These labels allow deployments to specify where applications should run.
- Pod Affinity/Anti-Affinity: To control the placement of pods more granularly, Red Hat OpenShift allows you to define affinity and anti-affinity rules. Affinity rules can be used to run pods on nodes with specific labels (for example, running on ppc64le nodes).

Anti-affinity ensures that pods are spread across different architectures to enhance fault tolerance and availability.

Scheduler enhancements

Red Hat OpenShift can be configured to use custom schedulers that are aware of multi-architecture constraints and can make intelligent decisions about where to place pods based on the architecture.

7.1.2 Practical deployment strategies

Choosing a deployment strategy applications in your multiple architecture cluster is dependent on your specific needs. Here are a couple of strategies that you might consider:

- Single Deployment, Multiple Architectures: You can create a single deployment configuration that references a manifest list. The Red Hat OpenShift scheduler automatically pulls the appropriate image based on the node's architecture. This approach simplifies management because you manage a single deployment rather than multiple architecture-specific deployments.
- Architecture-Specific Deployments: In some cases, you may need to optimize configurations or resource requests for specific architectures. Red Hat OpenShift allows you to create separate deployment configurations for each architecture, which can be managed independently but originate from the same base configuration or Docker file.

7.1.3 Challenges and considerations

Managing a multiple architecture cluster offers several benefits as have been described earlier. It can also provide additional challenges which need to be addressed. Consider the following:

- Consistent Configuration Management: Managing configurations across multiple architectures can be challenging. It is essential to ensure that environment variables, config maps, and security settings are consistent across all deployments unless specific differences are required for architectural reasons.
- Testing and Validation: Applications should be thoroughly tested on all target architectures. This might require investment in CI/CD pipelines that are capable of building and testing applications across multiple architectures.
- Performance Optimization: Each architecture may have different performance characteristics. Profiling and tuning applications for each architecture is crucial to ensure that they perform well regardless of the underlying hardware.

By strategically managing multi-architecture deployments in Red Hat OpenShift, organizations can ensure their applications are robust, flexible, and capable of running efficiently across diverse computing environments. This approach not only maximizes resource utilization but also enhances the application's resilience by spreading workloads across different hardware platforms.

7.2 Service exposure and load balancing

In Red Hat OpenShift, exposing services to the outside world and efficiently managing traffic is crucial for the accessibility and performance of applications. Red Hat OpenShift provides a robust set of tools to manage service exposure and load balancing, integrating Kubernetes native capabilities with additional enhancements for more advanced needs.

Understanding service exposure

Service exposure in Red Hat OpenShift is about making internal services accessible from outside the Kubernetes cluster. This is vital for any application that needs to interact with users or external systems. Red Hat OpenShift accomplishes this through Routes, Services, and Ingress Controllers.

Services in Kubernetes

Services provide important functions within the cluster. Here is a list of some of those:

- ClusterIP: ClusterIP is the default Kubernetes service that assigns internal IP addresses for intra-cluster communication in order to allow communication within the cluster. It is not accessible from outside the cluster.
- NodePort: The NodePort service exposes applications to external clients via specific ports on worker nodes. A NodePort service is accessible from outside the cluster.
- LoadBalancer: The LoadBalancer connects externally using a cloud provider's load balancer. This service type enhances the NodePort Service by adding a load-balancing functionality to distribute traffic among nodes.

Routes and ingress controllers

In Kubernetes, Ingress and Routes are both tools that can be used for traffic routing. They are briefly defined here:

- Routes: Red Hat OpenShift enhances Kubernetes' Ingress capability with Routes, which allow you to define rules to handle inbound requests to the cluster's services. You can assign custom hostnames, paths, and TLS (Transport Layer Security) termination settings at the route level.
- Ingress Controllers: Managed by Routes in Red Hat OpenShift, these are responsible for implementing the Ingress rules, managing redirection, and forwarding rules. They typically handle HTTPS termination, setting headers, and other necessary functions for web traffic management.

Load balancing strategies

Load balancing is about distributing network traffic across multiple servers or pods to ensure no single node bears too much demand, enhancing the application's responsiveness and availability.

- Internal Load Balancing: This is managed by Kubernetes Services (like ClusterIP and LoadBalancer), which use kube-proxy to route traffic to available pods based on the selected service type.
- External Load Balancing: This involves using external load balancers that integrate with Red Hat OpenShift, which might be provided by a cloud platform (like AWS ELB, Azure, Load Balancer) or implemented using dedicated hardware in on-premises setups. These are typically used with the LoadBalancer service type.
- HAProxy Router: Red Hat OpenShift uses an integrated HAProxy router as the default router-based load balancer. It handles incoming external traffic and routes it to the correct services based on the defined routes.

Implementing load balancing

Choosing a methodology to implement your load balancing should be done based on your specific needs. These options are available:

- Round Robin: The simplest form of load balancing, where each server or pod gets traffic in turn. This is the default method in Red Hat OpenShift's integrated HAProxy router.

- Sticky Sessions: Also known as session affinity, which can be configured to route the requests from the same client to the same pod each time. This is useful for applications that maintain session state between requests.
- Weighted Routing: Traffic is distributed to different servers based on weights assigned to each server. This can be useful when servers have different capabilities or when deploying new versions gradually (canary deployments).

High availability and scalability

Red Hat OpenShift clusters often host applications that are business critical. In addition, the clusters can be designed to support varying workflow patterns, scaling up and down based on your user requirements. These are accomplished through:

- Redundancy: To achieve high availability, Red Hat OpenShift clusters should deploy multiple instances of routers in different availability zones or nodes.
- Scalability: Red Hat OpenShift routers can scale out (adding more routers) or scale up (enhancing the capacity of existing routers) based on traffic requirements.

Security considerations

Security is critical as well. Protecting the security of your data and preventing unauthorized access to the cluster is important. Here are some functions that allow you to keep your cluster secure.

- TLS/SSL Termination: Routes in Red Hat OpenShift can be configured to handle SSL termination, off loading the CPU-intensive encryption and decryption tasks from the backend pods.
- Edge, Pass-through, and Re-encrypt: These are the types of TLS terminations supported by Red Hat OpenShift Routes. Edge termination decrypts requests at the HAProxy router, Pass-through leaves encryption intact until it reaches the pod, and Re-encrypt decrypts and re-encrypts the request so a secure connection is maintained all the way to the pod.

By effectively using these tools and strategies for service exposure and load balancing, Red Hat OpenShift ensures that applications are not only accessible and responsive but also secure and capable of handling varying loads efficiently. This is essential for maintaining service quality as user demand and data traffic fluctuate.

7.3 Application scaling and management

Application scaling and management in Red Hat OpenShift are critical for adapting to varying workloads, ensuring efficient resource use, and maintaining application performance and availability. Red Hat OpenShift provides a robust suite of tools to automate scaling operations and manage application lifecycles efficiently.

Scaling in Red Hat OpenShift can be categorized into two main types: horizontal scaling (out or in) and vertical scaling (up or down).

- Horizontal Scaling (Scaling Out/In): This involves increasing or decreasing the number of pod replicas to handle changes in load. It is effective for applications designed to run concurrently across multiple instances.
- Vertical Scaling (Scaling Up/Down): This entails adjusting the amount of CPU and memory allocated to the pods. It is suitable for applications that require more resources per instance rather than multiple parallel instances.

Implementing Horizontal Pod Autoscaler

The Horizontal Pod Autoscaler (HPA) automatically adjusts the number of pod replicas in a deployment or replica set based on observed CPU utilization or other customizable metrics such as:

- Metrics Monitoring: HPA uses metrics from the Metrics Server in Red Hat OpenShift, which collects resource usage data from each node and pod. Users can specify custom metrics for more fine-grained control over the scaling process.
- Configuration: Administrators define HPA policies specifying the minimum and maximum number of pods, as well as the target CPU or memory usage percentages that trigger scaling actions.
- Responsive Scaling: HPA adjusts the number of pods dynamically in response to the specified metrics, ensuring the application meets performance standards without over-utilizing resources.

Vertical Pod Autoscaler

Vertical Pod Autoscaler (VPA) reallocates memory and CPU resources among pods in a deployment, automatically adjusting their requests and limits to fit the workload.

- Analysis and Adjustment: VPA continuously analyzes the historical and current resource consumption of pods and adjusts their CPU and memory requests to optimize for efficiency and performance.
- Modes of Operation: VPA can operate in three modes: “Off” (only recommendations are provided, no automatic changes), “Initial” (applies recommended values on pod start, no further changes), and “Auto” (automatically updates running pods).
- Use Cases: VPA is particularly useful for applications with variable resource demands that are difficult to predict manually.

Manual scaling

In addition to automatic scaling, Red Hat OpenShift also supports manual scaling, which allows administrators to adjust the number of pods or resource allocations based on anticipated changes in load. This can be done with:

- CLI and UI Tools: Users can scale applications directly from the Red Hat OpenShift CLI (**oc**) or through the web console, providing flexibility in how they manage scaling.
- Scripting and Automation: Manual scaling commands can be integrated into scripts or CI/CD pipelines to automate scaling based on non-standard events or custom triggers.

7.3.1 Managing application deployments

Effective application management involves more than just scaling. Red Hat OpenShift provides comprehensive tools to manage the entire lifecycle of applications. Tools are available to manage:

- Rolling Updates and Rollbacks: Red Hat OpenShift supports rolling updates, which gradually replace the old version of an application with a new one with minimal downtime. If issues arise, rollbacks can be easily performed to return to a previous state.
- Deployment Strategies: Administrators can choose from several deployment strategies, such as Recreate (all pods are replaced at once, simple but with downtime) and Blue-Green Deployment (two versions run simultaneously for testing before full switch-over).

- Resource Quotas and Limits: To prevent any application from monopolizing cluster resources, administrators can set quotas and limits on the resources a namespace or application can consume.

7.3.2 Observability and monitoring

Understanding the current state of a system or application through cumulative information gathered from its parts is known as observability. This concept typically emphasizes observing the entire system or application rather than individual components. Monitoring plays a crucial role in observability, tracking external metrics to identify anomalies like errors or security incidents. By being proactive, monitoring serves as a primary shield against system failures. Often used alongside alerting systems, it effectively highlights potential problems in areas such as network performance and infrastructure health. Within the Red Hat OpenShift environment observability and monitoring can be accomplished by:

- Built-in Monitoring Tools: Red Hat OpenShift integrates with tools like Prometheus for monitoring metrics and Alertmanager for configuring alerts based on specific conditions. This observability is crucial for proactive management and incident response.
- Logging and Tracing: Comprehensive logging (using tools like Elasticsearch, Fluentd, and Kibana) and tracing capabilities are essential for diagnosing issues and optimizing application performance.

By leveraging these tools and practices for application scaling and management, Red Hat OpenShift ensures that applications can perform efficiently and reliably under varying load conditions. This adaptability is crucial for maintaining user satisfaction and operational stability in dynamic environments.

These capabilities ensure that applications deployed on Red Hat OpenShift can effectively handle varying loads, maintain high availability, and respond dynamically to the changing demands of the business environment. Each of these sections could be elaborated with real-world examples, best practices, and detailed step-by-step guidance to provide comprehensive coverage for readers interested in deploying robust, scalable applications on Red Hat OpenShift.

8

Security

This chapter focuses on Security within the Red Hat OpenShift ecosystem. Given that contemporary containerized cloud architectures may be less fortified compared to their traditional counterparts, it's crucial to devise and integrate suitable security measures and solutions. This compilation doesn't encompass every potential approach, yet presents various considerations to help fortify your Red Hat OpenShift deployment. To explore security aspects comprehensively, refer to *Security Implementation with Red Hat OpenShift on IBM Power Systems*, REDP-5690.

This chapter contains the following topics.

8.1 Cluster security

OpenShift offers a method for building applications as a collection of independent containerized microservices, sharing a host among multiple tenants while maintaining their dependencies. This approach represents a departure from traditional monolithic application design, enabling quicker release cycles and seamless scaling of microservices based on fluctuating business demands – such as unexpected spikes in client workload. Adopting Red Hat OpenShift delivers significant operational benefits to businesses, including improved scalability, adaptability, and maintenance efficiency. As organizations embrace application modernization efforts, selecting the optimal container platform becomes crucial, with security emerging as a primary concern. Indeed, microservice architectures introduce unique security challenges that require careful consideration from security teams within the organization.

In summary, Red Hat OpenShift facilitates rapid application development through containerization, offering enhanced agility and resource optimization. Meanwhile, security remains a top priority as organizations transition towards modernized applications built upon microservices principles.

The major security aspects that need to be addressed are the following:

Complexity and visibility Microservices challenge security due to the distributed nature of its building blocks. As containers are independent and built on various frameworks (e.g., different language, different libraries), the security challenges require a preventive strategy to monitor containers.

Communication Microservices communicate with each other via APIs, increasing the attack surface. Encryption of data and authentication are measures that are needed to address the issue effectively.

Access control Access needs to be monitored granularly, and specific policies are required to guarantee a balance between a smooth development workflow and a highly secure environment.

8.1.1 Best practices for container security

Red Hat OpenShift and containers can be a complex environment. There are several aspects to consider as you configure the environment to properly configure and manage a secure K8s environment:

- Secure container images in the container registry

  Developers must adapt the process of creating a secure image that is built on the secure application code. They must implement a security and vulnerability scanner in the CI/CD pipeline. If the code is not secure and contains vulnerabilities, then the container can be vulnerable and prone to attacks.

- Node security

  The cluster nodes need to be secured. This requires that you:

  - Apply patches for the OS.
  - Configure firewalls.
  - Use the principle of least privilege.
  - Block public access to the nodes.
  - Follow the best practices that are mentioned in the Center for Internet Security (CIS) benchmarks.

- Secure apiserver

  Because all communication to the containers and in the cluster go through the API server, implement TLS for apiserver communication.

- Role-based access control (RBAC)

  Limit the access to the cluster with RBAC.

- Principle of least privilege

  Provide the required minimum and limited access to service accounts and users.

- Network security

  Implement proper ingress and egress rules and Container Network Interface (CNI) network policies for workloads in the cluster. You should:

  – Implement a service mesh, if appropriate.
  – Leverage side-car proxy and mutual Transport Layer Security (mTLS) for secure communication between microservices in the cluster.

  For more information, see Controlling traffic with network policies.

- Pod security

  Configure an appropriate pod security standards policy. Pod security is managed by Pod Security Admission policies in the current version of Red Hat OpenShift. For more information, see Pod Security Admission OpenShift 4.11.

- Secrets

  Do not use a configmap to keep a password or other authentication tokens; instead, use secrets. If appropriate, use a third-party vault to inject a secret into the pod.

- Version control

  Keep your Red Hat OpenShift cluster up to date.

- Monitor

  Set up monitoring and observability in your environment.

8.1.2 Designed for Enterprise-Grade security

Red Hat OpenShift is designed with security as a foundational aspect, integrating robust security features that support the demanding requirements of enterprise environments. This includes everything from strict access controls to ensuring container and platform integrity. Here is a deeper look into how Red Hat OpenShift delivers enterprise-grade security:

- Security Context Constraints (SCC)

  Red Hat OpenShift enhances the security of container environments by using Security Context Constraints (SCC) to define a set of conditions that a container must comply with to run on the platform. These role-based constraints can limit the actions that a pod can perform and the resources it can access, significantly reducing the risk of unauthorized actions.

  Administrators can use SCCs to manage permissions at a granular level, controlling whether pods can run as privileged containers, access sensitive volumes, or use host networking and ports, among other security settings.

- Integrated Authentication and Authorization

  Red Hat OpenShift integrates with existing enterprise authentication systems, such as LDAP, Active Directory, and public OAuth providers, to provide a robust user authentication process seamlessly across the organization.

RBAC (role based access control) in Red Hat OpenShift allows administrators to regulate access to resources based on the roles of individual users within the enterprise. This ensures that only authorized users have access to control critical operations, thereby securing the environment against internal and external threats.

- Network Policies and Encryption

  Red Hat OpenShift allows administrators to define network policies that govern how pods communicate with each other and with other network endpoints. This ensures that applications are isolated and protected from network-based attacks.

  Data in transit and at rest can be encrypted, providing an additional layer of security. Red Hat OpenShift supports TLS for all data in transit and can integrate with enterprise key management solutions to manage encryption keys for data at rest.

- Security Enhancements and Compliance

  Red Hat OpenShift provides automated mechanisms to apply security patches and updates to the container host, runtime, and the application containers themselves. This helps in maintaining security compliance and reducing the vulnerability window.

  Red Hat OpenShift includes features to support compliance with various regulatory requirements such as PCI DSS, HIPAA, and GDPR. It provides extensive logging and auditing capabilities that help in tracking all user actions and system changes, crucial for forensic analysis and compliance reporting.

- Container Security and Image Assurance

  Red Hat OpenShift integrates with tools like Quay.io to provide automated container image scanning. This scans images for vulnerabilities before they are deployed, and image signing ensures that only approved and verified images are used in the environment.

  Red Hat OpenShift runs on Red Hat Enterprise Linux CoreOS (RHCOS) and leverages SELinux to enforce mandatory access control policies that isolate containers from each other and from the host system. This prevents a compromised container from affecting others or gaining undue access to host resources.

- Secure Default Settings and Practices

  Red Hat OpenShift encourages the use of minimal base images that contain only the essential packages needed to run applications, reducing the potential attack surface. Red Hat OpenShift is preconfigured with security best practices and regularly updated security benchmarks that guide users in setting up and maintaining a secure environment.

By providing these comprehensive security features, Red Hat OpenShift addresses the complex security challenges faced by enterprises today, ensuring that their deployments are secure by design, compliant with industry standards, and capable of withstanding modern cybersecurity threats. This security-first approach is integral to maintaining trust and integrity in enterprise applications and data.

8.2 Securing containerized applications

Understanding that Red Hat OpenShift provides a framework for security in your cloud native container environment, it is imperative that you plan for and implement good security processes. This section provides a brief look at several places in your environment where you should implement security practices to protect your applications and your data. This is not an exhaustive list and you can find more detail in this IBM Redpaper: *Security Implementation with Red Hat OpenShift on IBM Power Systems*, REDP-5690.

Network isolation and API endpoint security

When working with containerized applications that are deployed across multiple distributed hosts or nodes, it becomes critical to secure the network topology.

Network namespaces usually assign a port range and IP address to a collection of containers, which help to distinguish and isolate pods from each other. By default, pods of different namespaces cannot send or receive data packets unless exceptions are made by the system administrator.

Red Hat OpenShift uses software-defined networking (SDN) to provide a unified cluster networking approach:

- Namespaces for container collections simplify network security architectures.
- The platform controls egress traffic by using a router or firewall so that clients can conduct IP whitelisting.

Red Hat OpenShift comes with many API authentication and authorization services that customers can readily integrate throughout application and platform endpoints. The most prominent one is Red Hat Single Sign-On (RH-SSO), which provides Security Assertion Markup Language (SAML) 2.0 and OpenID Connect based authentication.

Container registry

Red Hat OpenShift provides a unified image registry that is on the infrastructure nodes of the cluster. This setup allows organizations to avoid third-party hosting services and public image storage services such as Docker Hub. By keeping all required images within the cluster, organizations can avoid reliance on third-party services and associated outages.

The container registry stores container images for the following reasons:

- Make images accessible to other users.
- Organize images in repositories that can contain multiple versions of images.
- Restrict access to images based on different authentication methods.

Here are some best practices to use to ensure secure container registries:

- Scan and track the contents of downloaded container images and add a layer of protection by using only trusted sources that are known to be free of vulnerabilities in all layers.
- Use immutable containers:
  - Rebuild and redeploy updated container images instead of changing them.
  - Use Red Hat certified images.
- Use Red Hat Security Advisories to alert you to any newly discovered issues in Red Hat-certified container images and direct you to the updated image.
- Check the Red Hat Ecosystem Catalog to look up security-related issues for each Red Hat image.
- Use RBAC to manage who can pull and push each container image.
- Use private Red Hat OpenShift Container Platform registries such as Red Hat Quay.
- Integrate CI/CD pipelines and image registries with Red Hat Advanced Cluster Security for Kubernetes for continuous scanning and assurance.

Red Hat OpenShift build process security

For containers, the “Build phase” of an application's lifecycle occurs when application code is integrated with runtime libraries and other dependencies. Defining and protecting the build process is critical to securing a container that might be deployed many times over its lifecycle.

Red Hat OpenShift uses the “Source-2-Image” (S2I) open-source framework for build management and image security. As developer code is built and committed to a repository through S2I, Red Hat OpenShift can trigger CI/CD processes to automatically assemble a new container image by using the freshly committed code, deploy that image for testing, and promote the tested image to full production status.

As a best practice, integrate automated security testing into the CI/CD pipelines by using Red Hat OpenShift. Leveraging the platform's RESTful APIs, you can integrate Static Application Security Testing (SAST) or Dynamic Application Security Testing (DAST) tools like IBM Rational® AppScan.

Ultimately, this approach of securing the software build process allows operations teams to manage base images, architects to manage middleware and software that are needed by the application layer, and developers to focus on writing better code.

Red Hat OpenShift deployment process security

Tools for automated, policy-based deployments can further secure containers beyond the software build process, and into the production deployment phase.

Red Hat OpenShift comes with Security Context Constraints (SCCs) that define a set of conditions that must be met before a collection of containers can be deployed.

Using SCCs, you can control the following items:

- How and where privileged containers can run.
- The capabilities that a running container may request.
- Allow or deny access to volumes.
- A containers user ID.
- The Security Enhanced Linux (SELinux) context of the container.

Security consideration for federation of containerized applications

Federation is invaluable when deploying and accessing applications that are running across multiple distributed data centers or clouds. Red Hat OpenShift and Kubernetes orchestration supports and facilitates federation in two different ways:

- Federated secrets automatically create and manage all authentication and authorization “secrets” across all clusters that belong to the federation.
- Federated namespaces ensure that pods (groups of containers) have consistent IP addresses and port ranges that are assigned to them.

8.3 Implementing security policies and compliance

In Red Hat OpenShift Container Platform, you can use security context constraints (SCCs) to control permissions for the pods in your cluster. Default SCCs are created during installation and when you install some Operators or other components. As a cluster administrator, you can also create your own SCCs by using the Red Hat OpenShift CLI (oc).

By default, Red Hat OpenShift prevents the containers running in a cluster from accessing protected functions. These functions – Linux features such as shared file systems, root access, and some core capabilities such as the KILL command -– can affect other containers running in the same Linux kernel, so the cluster limits access to them. Most cloud-native applications work fine with these limitations, but some (especially stateful workloads) need greater access. Applications that need these functions can still use them, but they need the cluster's permission.

The application's security context (SC) specifies the permissions that the application needs, while the cluster's SCCs specify the permissions that the cluster allows. An SC with an SCC enables an application to request access while limiting the access that the cluster will grant. Access to protected functionality is not controlled by the application, but by the application's environment. This way, even a rogue or hacked application cannot grant itself access to protected functionality. The access is configured not by the application (which could be compromised) but rather by the pod that creates the application container and by the cluster that runs it. The application can access only the functions that the pod requests and that the cluster approves.

When the pod creates the application's container, it configures the container to allow the access it specifies. If the application tries to perform a protected function, Linux will block it unless the pod has configured the container to allow access to that function.

An application's access to protected functions is an agreement between three personas:

1. A developer who writes an application that performs protected functions.
2. A deployer who writes the deployment manifest that specifies which type of access the application requires.
3. An administrator who decides whether to grant the deployment the access it specifies.

Figure 8-1 illustrates the components and process that allow an application to access functions.

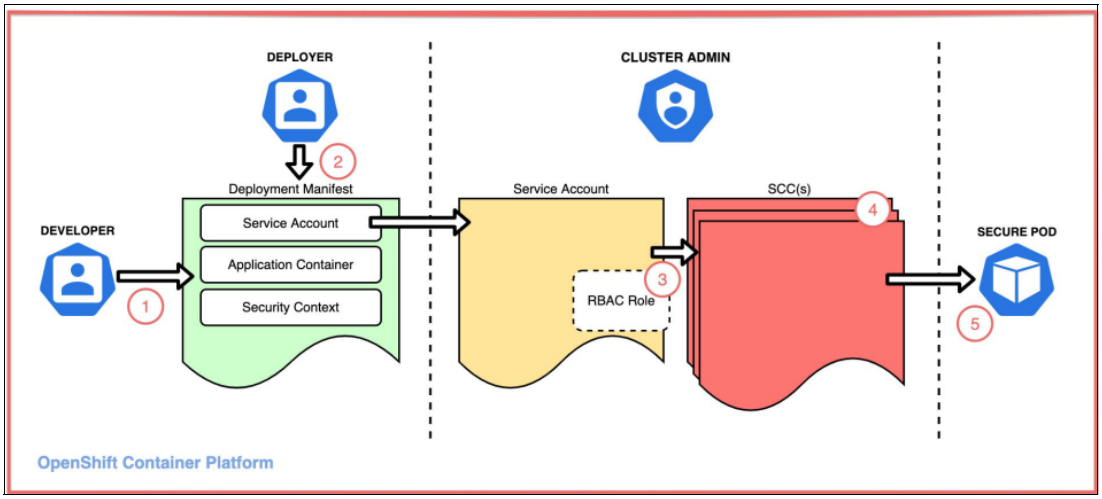


Figure 8-1 How SCCs and SCs protect your environment

1. A developer writes an application that needs access to protected functions.
2. A deployer creates a deployment manifest to deploy the application with a pod specification that configures:
   - A security context (for the pod and/or for each container) that specifies the access needed by the application, thereby requesting it
   - A service account to grant the requested access
3. An administrator assigns a security context constraint to the service account that grants the requested access, thereby allowing the pod to configure Linux as specified.
   - The SCC can be assigned directly to the service account or indirectly through a RBAC role or group.
4. The SCC may be one of Red Hat OpenShift's predefined SCCs or it may be a custom SCC.
5. If the SCC grants the access, the admission process allows the pod to deploy and the pod configures the container as specified.

For more information, see Get started with security context constraints on Red Hat OpenShift.

8.4 Security policies and compliance solutions

One could argue that security is more important in modern IT solutions and systems than high availability or performance. This is because more than ever, availability and performance depend on resilience against increasingly frequent and powerful attacks on the confidentiality, availability and integrity of IT systems and their data.

Our systems are not only being threatened with continuous and automated attacks by internal and external parties, but also with highly sophisticated approaches such as Advanced Persistent Threats (APTs), Ransomware, Side Channel Attacks, Supply Chain Exploitations and so on.

As large-scale IT systems become more automated and dynamically reconfigured, the data and infrastructure objects requiring protection frequently shift beyond visible range, making it difficult to adjust security measures accordingly to keep pace with constantly evolving informational landscapes. This presents a significant challenge.

We require comprehensive solutions or tools that can assist in adhering to industry standards' compliance, encompassing at minimum general security principles like overall hardening, Zero Trust, Need-to-Know, or Least Privilege. These concepts have widespread applicability – independently of specific implementation details – thereby necessitating their inclusion in our approach.

Below are some security policies and compliance solution and tools that could be helpful for our further reference.

8.4.1 Red Hat Advanced Cluster Security for Kubernetes

Red Hat Advanced Cluster Security (RHACS) for Kubernetes is a Kubernetes-native security platform that equips you to build, deploy, and run cloud-native applications with security in mind. The solution helps protect containerized Kubernetes workloads in all major clouds and hybrid platforms, including Red Hat OpenShift, Amazon Elastic Kubernetes Service (EKS), Microsoft Azure Kubernetes Service (AKS), and Google Kubernetes Engine (GKE).

Red Hat Advanced Cluster Security for Kubernetes is included with Red Hat OpenShift Platform Plus, a complete set of powerful, optimized tools to secure, protect, and manage your apps such as:

- Vulnerability management

  Identify and fix vulnerabilities in both container images and Kubernetes across the entire software development life cycle.

- Compliance

  Audit your systems against CIS Benchmarks, NIST, PCI, and HIPAA, with interactive dashboards and one-click audit reports.

- Network segmentation

  Visualize existing versus allowed network traffic and enforce network policies and tighter segmentation using Kubernetes-native controls.

- Risk profiling

See all your deployments ranked by risk level, using context from Kubernetes' declarative data, to prioritize remediation.

- Configuration management

  Apply best practices to hardening your Kubernetes environments and workloads for a more secure and stable application.

- Detection and response

  Use rules, allowlists, and base lining to identify suspicious activity, and take action to thwart attacks, using Kubernetes for enforcement.

Red Hat OpenShift Platform Plus is a unified platform designed to build, modernize, and deploy applications at scale. Multi-cluster security, compliance, application and data management all work across multiple infrastructures to provide consistency throughout the software supply chain. Red Hat OpenShift Platform Plus helps you work smarter and faster with a complete set of services for bringing apps to market on your hybrid cloud. Red Hat Advanced Cluster Security (RHACS) is a part of the Red Hat OpenShift Platform Plus bundles.

Red Hat Advanced Cluster Security Architecture Overview

Red Hat Advanced Cluster Security for Kubernetes uses a distributed architecture that supports high-scale deployments and is optimized to minimize the impact on the underlying Red Hat OpenShift Container Platform or Kubernetes nodes. RHACS architecture is shown in Figure 8-2.

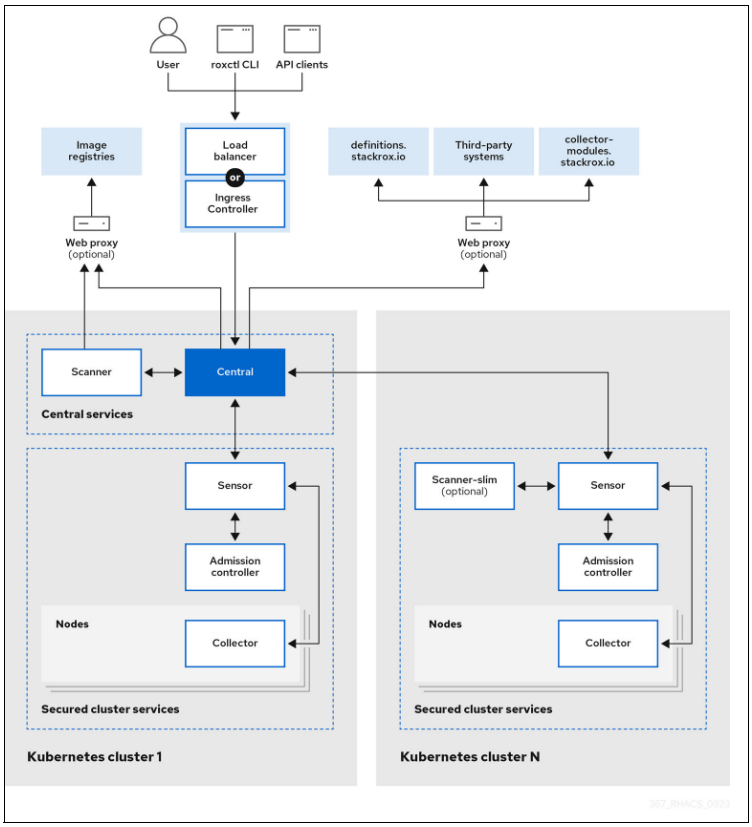


Figure 8-2 Red Hat Advanced Cluster Security for Kubernetes architecture for Kubernetes

RHACS is installed as a set of containers in your Red Hat OpenShift Container Platform or Kubernetes cluster. RHACS includes:

- Central services (CS) you install on one cluster.
- Secured cluster services you install on each cluster you want to secure using RHACS.

In addition to these primary services, RHACS also interacts with other external components to enhance your clusters' security.

The Central component provides the policy and violation management interface, data persistence and image scanning. The Secured Cluster provides components for monitoring a cluster and its workload activity and enforcing security policies. Central is typically installed on one cluster with multiple Secured Clusters connected to it.

RHACS Cloud Service (RHACS CS or Cloud Service) is a Red Hat offering where Central is deployed to Red Hat managed infrastructure, and is upgraded, monitored and managed by Red Hat Software Engineering and Site Reliability Engineering (SRE) teams.

RHACS Secured Cluster services work with either self-managed RHACS Central or with an instance of RHACS CS. In both cases, Secured Cluster is the same set of components which are deployed, upgraded and managed by the User.

Red Hat Advanced Cluster Security support

The currently supported versions of Red Hat Advanced Cluster Security (RHACS) are compatible with all currently supported versions of Red Hat OpenShift Container Platform. Specifically, if RHACS 4.2 is installed onto Red Hat OpenShift Container Platform 4.10, which has reached its end-of-life, RHACS 4.2 will continue to function without issue. However, Red Hat does not formally support combinations where the major version of either product falls outside their respective life cycles.

Additionally, RHACS supports various architectures within OpenShift Container Platform, including x86_64, ppc64le (IBM Power), and s390x (IBM Z and IBM LinuxONE).

8.4.2 Compliance operator

With Red Hat OpenShift Compliance Operator, an administrator can run compliance scans and provide remediations for anomalies found in a Red Hat OpenShift cluster and the worker machines (nodes) running the cluster. Red Hat OpenShift Compliance Operator leverages OpenSCAP and the community-based compliance content that is developed in the ComplianceAsCode content project. This content project provides a bundle of security policies, along with default profiles for various operating system platforms, and security standards such as the Center for Internet Security (CIS) benchmark, HIPPA, NIST 800-53 Moderate, and Australian Cyber Security Centre (ACSC) Essential Eight.

Red Hat OpenShift Compliance Operator does not require any additional subscriptions.

For more information, see How to use the Compliance Operator in Red Hat OpenShift Container Platform 4 and 8.4 “Red Hat OpenShift Compliance Operator” in *Security Implementation with Red Hat OpenShift on IBM Power Systems*, REDP-5690.

8.4.3 Insights operator

Insights Advisor can be used to assess and monitor the health of your Red Hat OpenShift Container Platform clusters. Whether concerned about individual clusters, or with the whole infrastructure, it is important to be aware of the exposure of the cluster infrastructure to issues that can affect service availability, fault tolerance, performance, or security.

Using cluster data collected by the Insights Operator, Insights repeatedly compares that data against a library of recommendations. Each recommendation is a set of cluster-environment conditions that can leave Red Hat OpenShift Container Platform clusters at risk. The results of the Insights analysis are available in the Insights Advisor service on Red Hat Hybrid Cloud Console. In the Console, you can perform the following actions:

- See clusters impacted by a specific recommendation.
- Use robust filtering capabilities to refine your results to those recommendations.
- Learn more about individual recommendations, details about the risks they present, and get resolutions tailored to your individual clusters.
- Share results with other stakeholders.

Using the Insights Operator

The Insights Operator periodically gathers configuration and component failure status and, by default, reports that data every two hours to Red Hat. This information enables Red Hat to assess configuration and deeper failure data than is reported through Telemetry. Users of Red Hat OpenShift Container Platform can display the report in the Insights Advisor service on Red Hat Hybrid Cloud Console.

The Insights Operator is installed and enabled by default. If you need to opt out of remote health reporting, see Opting out of remote health reporting.

For more information on using Insights Advisor to identify issues with your cluster, see Using Insights to identify issues with your cluster.

8.4.4 Network policy

By default, all pods in a project are accessible from other pods and network endpoints. By using Network Policy we can isolate one or more pods in a project. This is done by creating NetworkPolicy objects in the application project to indicate the allowed incoming connections Using Role-based access control (RBAC), A project administrator can create and delete NetworkPolicy objects within their own project.

The Center for Internet Security Kubernetes security benchmark recommends that all application Namespaces (projects) have Network Policies defined. This is identified as a high severity levels. Figure 8-3 illustrates how network policies allow or deny connectivity between different namespaces.

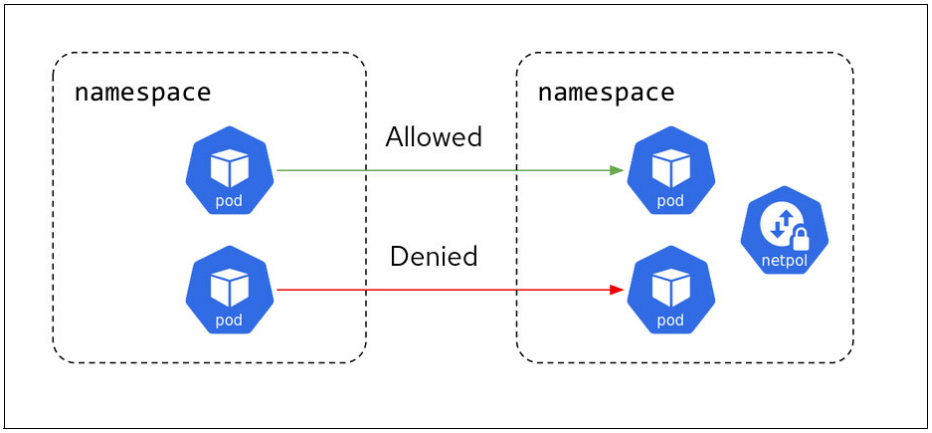


Figure 8-3 Network Policy

A network policy applies to only the TCP, UDP, ICMP, and SCTP protocols – other protocols are not affected. Best practices recommend:

Deny all traffic and allow access only when required. For example:

- Allow only connections from the Red Hat OpenShift Container Platform Ingress Controller.
- Accept only connections from pods within a project.
- Allow only HTTP and HTTPS traffic based on pod labels.
- Accept connections by using both namespace and pod selectors.

For more information, see Network Policies: Controlling Cross-Project Communication on OpenShift and About network policy.

Part 4

Advanced Topics

This section introduces additional topics to consider when implementing Red Hat OpenShift in your IBM Power environment.

The following topics are included:

9

Management using Red Hat Advanced Cluster Management

This chapter introduces Red Hat Advanced Cluster Management (RHACM). Red Hat Advanced Cluster Management for Kubernetes offers unified control over clusters and applications through a single interface, incorporating built-in security policies. Integrated in Red Hat OpenShift Platform, RHACM allows you to deploy applications, administer multiple clusters, and enforce policies consistently across numerous clusters at large scales. This comprehensive solution guarantees compliance, tracks resource utilization, and preserves uniformity.

This chapter will cover the following topics:

9.1 Introducing Red Hat Advanced Cluster Management

Users, such as administrators and site reliability engineers, face challenges as they work across a range of environments, including multiple data centers, private clouds, and public clouds that run Kubernetes clusters. Red Hat Advanced Cluster Management for Kubernetes provides the tools and capabilities to address these common challenges.

Red Hat Advanced Cluster Management for Kubernetes provides end-to-end management visibility and control to manage your Kubernetes environment. Take control of your application modernization program with management capabilities for cluster creation, application lifecycle, and provide security and compliance for all of them across data centers and hybrid cloud environments.

Clusters and applications are all visible and managed from a single console, with built-in security policies. Run your operations from anywhere that Red Hat OpenShift runs, and manage any Kubernetes cluster in your Red Hat Advanced Cluster Management for Kubernetes:

- Offers end-to-end management, visibility, and control of cluster and application life cycle, along with improved security and compliance of the entire Kubernetes domain – across multiple datacenters and public cloud environments.
- Provides a hybrid cloud management platform and capabilities that address common challenges faced by administrators and site reliability engineers (SREs) as they work across a range of environments such as multiple datacenters and private and public cloud environments that run Kubernetes clusters – including remote edge sites.
- Provides FIPS mode support and allows management of Kubernetes clusters from one place. Kubernetes clusters can range from public cloud (AWS, Google, Microsoft), private cloud (OpenStack, Virtualization) to on-premises (bare metal servers with x86_64, IBM Power, and LinuxONE or Z systems) and even to edge environments (ARM).
- Multi-cluster observability for cluster health along with optimization capabilities deliver an enhanced SRE experience with out-of-the-box multi cluster dashboards that can store long-term historical data and provide an overview of your full cloud-ready.
- Unified multi cluster life cycle management allows the creation, upgrade, and destruction of Kubernetes clusters reliably, consistently, and at scale, using an open-source programming model that supports and encourages infrastructure as code (IaC) best practices and design principles.
- Policy-based governance, risk, and compliance allow the application of a policy-based governance approach to automatically monitor and ensure desired best practices configuration state for controls related to security, resiliency, and software engineering so that these controls are operated to industry compliance standards or self-imposed corporate standards.
- Advanced application lifecycle management integrates open standards and deploys applications using placement rules that are integrated into existing CI/CD pipelines and governance controls.
- Allows Edge management at scale. With single-node Red Hat OpenShift clusters and Red Hat Advanced Cluster Management, continuously scale while enabling availability in high-latency, low-bandwidth edge use cases.
- Provides Business Continuity. Using Red Hat Advanced Cluster Management along with the broader Red Hat portfolio ensures that the applications and stateful applications your business relies on are always up and running.

9.2 Architecture

Red Hat Advanced Cluster Management for Kubernetes consists of several multi-cluster components, which are used to access and manage your clusters. The hub cluster is the common term that is used to define the central controller that runs in a Red Hat Advanced Cluster Management for Kubernetes cluster. From the hub cluster, you can access the console and product components, as well as APIs such as the rcm-api, which handles API requests related to cluster lifecycle management.

Figure 9-1 shows the hub cluster and a single managed cluster. RHACM can manage multiple clusters from a single hub cluster.

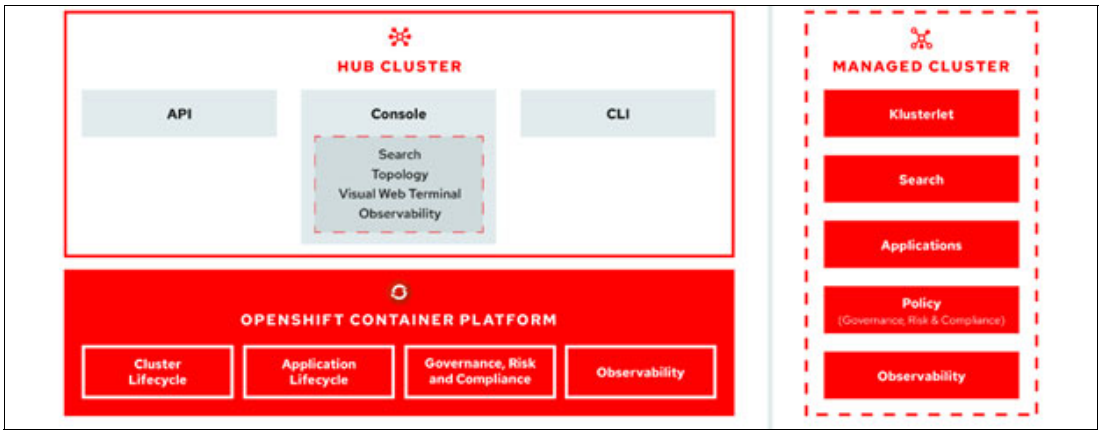


Figure 9-1 RHACM hub cluster and managed cluster

The hub cluster aggregates information from multiple clusters and maintains the state of the managed clusters and the applications that run them. RHACM provides a set of REST APIs to support the various functions it uses for management. A managed cluster is one of the clusters which are managed by the hub cluster. The *Klusterlet* agent manages the connection to the hub cluster.

Figure 9-2 shows a more detailed view of the HRHACM components.

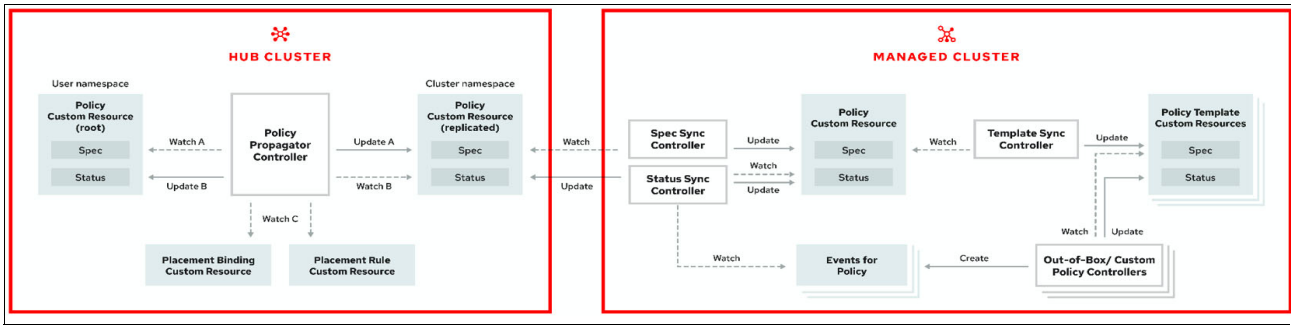


Figure 9-2 Cluster components

The RHACM hub cluster is the central controller providing the console and product components. The hub cluster uses APIs to manage API requests related to cluster lifecycle management. The APIs can look for resources across the clusters, run commands from the Visual Web Terminal, and view the topology of your environment. The hub cluster provides observability and collects metrics from your managed clusters and cloud providers.

The hub cluster aggregates information from multiple clusters using an asynchronous work request model. The hub cluster maintains the state of clusters and the applications running on those clusters using a graph database and it uses etcd, a distributed key value store, to store the state of work requests and results from multiple clusters.

The managed cluster receives and applies requests, then returns the results. It sends metrics to the hub cluster using the observability service.

RHACM can create clusters and import existing clusters as well as upgrade, destroy, and manage clusters across on-premise, public and private clouds. providing an aggregated view of all cluster health statuses or a view of individual health metrics of any managed cluster.

RHACM helps manage the application lifecycle and can manage application resources on managed clusters. This includes automation of the deployment and lifecycle management of resources across the managed clusters. It exposes easy-to-reconcile service routes and provides full control of Kubernetes resource updates to manage all aspects of the application.

RHACM allows governance and risk management by defining the processes that are used to manage security and compliance from a central interface page. Cluster managers can view and create policies with the Red Hat Advanced Cluster Management policy framework, create RBAC controls, and define security policies.

The observability component is a service used to understand the health and utilization of clusters across a fleet. By default, the multicluster observability operator is enabled during the installation of Red Hat Advanced Cluster Management. Observability on RHACM includes the definition for multiple components:

- Observability architecture
- Observability configuration
- Enabling the observability service
- Using observability
- Customizing observability
- Observability alerts
- Searching in the console introduction
- Using observability with Red Hat Insights

9.2.1 Observability architecture

You can use Red Hat Advanced Cluster Management for Kubernetes to gain insight and optimize your managed clusters. Enable the observability service operator, multicluster-observability-operator, to monitor the health of your managed clusters. By default, observability is included with the product installation, but is not enabled due to the requirement for persistent storage.

The multiclusterhub-operator enables the multicluster-observability-operator pod by default. The user must configure the multicluster-observability-operator pod. When the Observability service is enabled by defining a MultiClusterObservability custom resource, the observability-endpoint-operator is automatically deployed to each imported or created managed cluster.

This controller starts a metrics collector that collects the data from Red Hat OpenShift Container Platform *Prometheus*, then sends the data to the Red Hat Advanced Cluster Management hub cluster. Figure 9-3 on page 135 illustrates the observability components

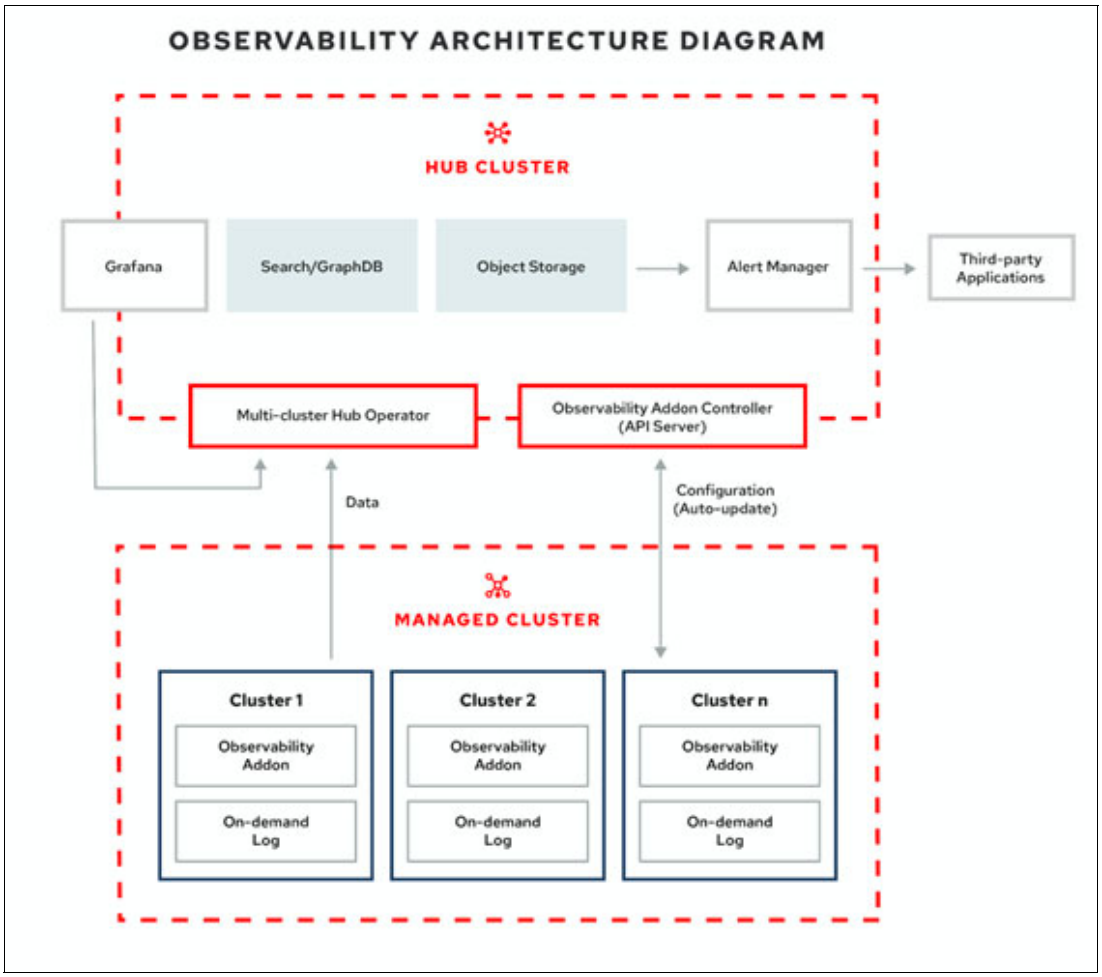


Figure 9-3 Observability Architecture diagram

When the Observability service is enabled, the hub cluster is always configured to collect and send metrics to the configured Thanos instance, regardless of whether hub self-management is enabled or not. When the hub cluster is self-managed, the disableHubSelfManagement parameter is set to false, which is the default setting. Metrics and alerts for the hub cluster appear in the local-cluster namespace. The local-cluster only appears in the cluster list drop-down menu if hub self-management is enabled. You can query the local-cluster metrics in the Grafana explorer.

The components of the observability architecture include the following items:

- The multicluster hub operator, also known as the multiclusterhub-operator pod, deploys the multicluster-observability-operator pod. It sends hub cluster data to your managed clusters.
- The observability add-on controller is the API server that automatically updates the log of the managed cluster.
- The Thanos infrastructure includes the Thanos Compactor, which is deployed by the multicluster-observability-operator pod. The Thanos Compactor ensures that queries are performing well by using the retention configuration, and compaction of the data in storage. To help identify when the Thanos Compactor is experiencing issues, use the four default alerts that are monitoring its health.

The observability component deploys an instance of Grafana to enable data visualization with dashboards (static) or data exploration. Red Hat Advanced Cluster Management supports version 8.5.20 of Grafana. You can also design your Grafana dashboard. For more information, see Designing your Grafana dashboard.

9.2.2 Alert Manager

The Prometheus Alertmanager enables alerts to be forwarded with third-party applications. You can customize the observability service by creating custom recording rules or alerting rules. Red Hat Advanced Cluster Management supports Prometheus Alertmanager 0.25. Figure 9-4 illustrates how the alert manager functions.

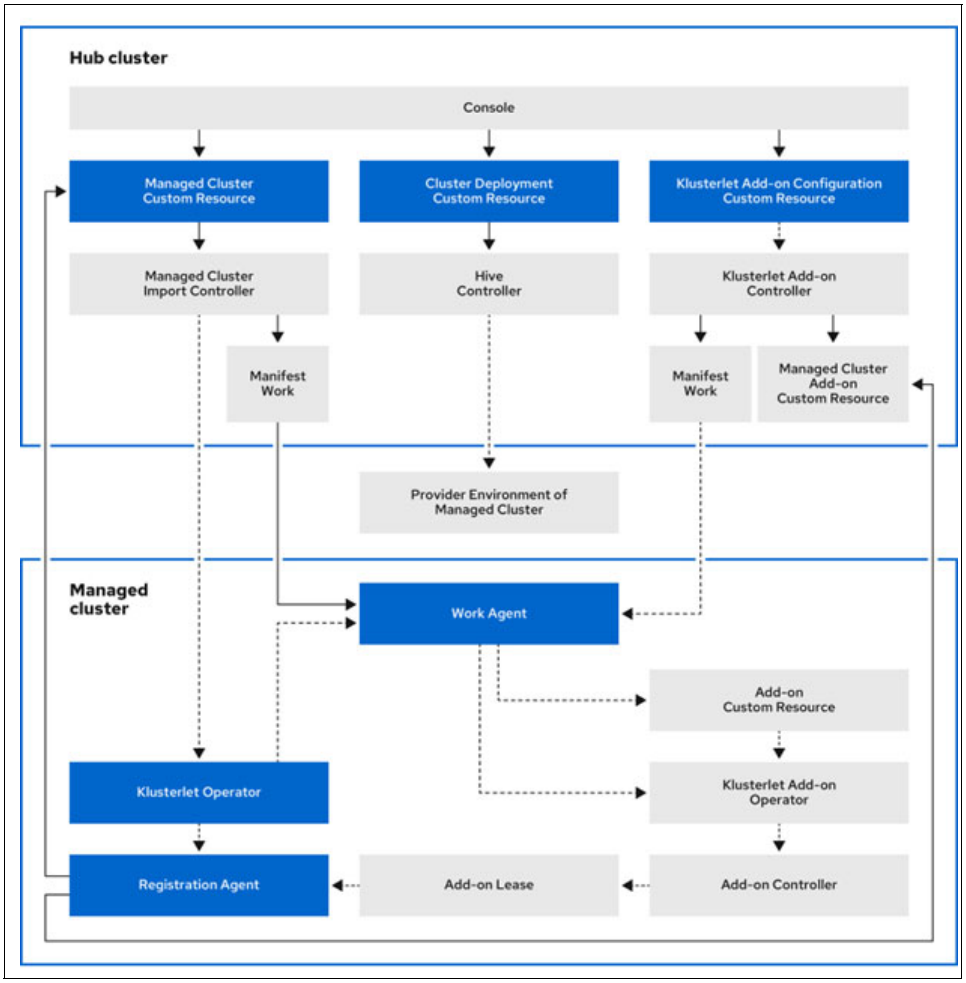


Figure 9-4 High level alert manager

There are predefined alerts that are managed. See Table 9-1 for a list of default alerts.

Table 9-1 List of default alerts:

Alert	Severity	Description
ACMThanosCompactHalted	Critical	An alert is sent when the compactor stops.
ACMThanosCompactHighCompactionFailures	Warning	An alert is sent when the compaction failure rate is greater than 5 percent.

Alert	Severity	Description
ACMThanosCompactBucketHighOperationFailures	Warning	An alert is sent when the bucket operation failure rate is greater than 5%.
ACMThanosCompactHasNotRun	Warning	An alert is sent when the compactor has not uploaded anything in the last 24 hours.

9.2.3 Sizing for Red Hat Advanced Cluster Management

The sizing requirements for Red Hat Advanced Cluster Manager is shown in Table 9-2. These are the suggested minimum compute resources in addition to any Red Hat OpenShift Container Platform requirement:

Table 9-2 Minimum sizes of compute resources

Node role	Minimum no. of nodes	Data stores	Total reserved memory (Lower bound) per node	Total reserved CPU (lower bound) per node
Master	3	etcd x 3	Per Red Hat OpenShift Container Platform sizing guidelines	Per Red Hat OpenShift sizing guidelines
Worker	3	redisgraph redis x 1	16 GB	6 CPU

9.3 Setting up Red Hat Advanced Cluster Management for Kubernetes on IBM Power

This section provides instructions for deploying RHACM as well as providing instructions for additional setup tasks.

9.3.1 Deploying RHACM

To deploy a Red Hat Advanced Cluster Management for Kubernetes console on an IBM Power Systems virtual machine requires the deployment of Red Hat OpenShift Container Platform on the IBM POWER server.

Using the Operator Hub on Red Hat OpenShift Container Platform, follow these steps to deploy Advanced Cluster Management:

1. Click on Operator Hub.
2. Search for Advanced Cluster Management.

   RHACM is easily installed using the OperatorHub. For detailed installation instructions, see the Red Hat Advanced Cluster Management for Kubernetes installation documentation.
3. Open Advanced Cluster Management Operator
4. Review the Configuration
5. Click Install

6. Verify the availability of Advanced Cluster Management in the list of installed Operators on Red Hat OpenShift
7. Open Advanced Cluster Management Console on Red Hat OpenShift

   To open the Red Hat Advanced Cluster Management for Kubernetes console, from the Red Hat OpenShift Container Platform console of your cluster, click the matrix icon at the upper-right side and select **Red Hat Advanced Cluster Management for Kubernetes** as shown in Figure 9-5.

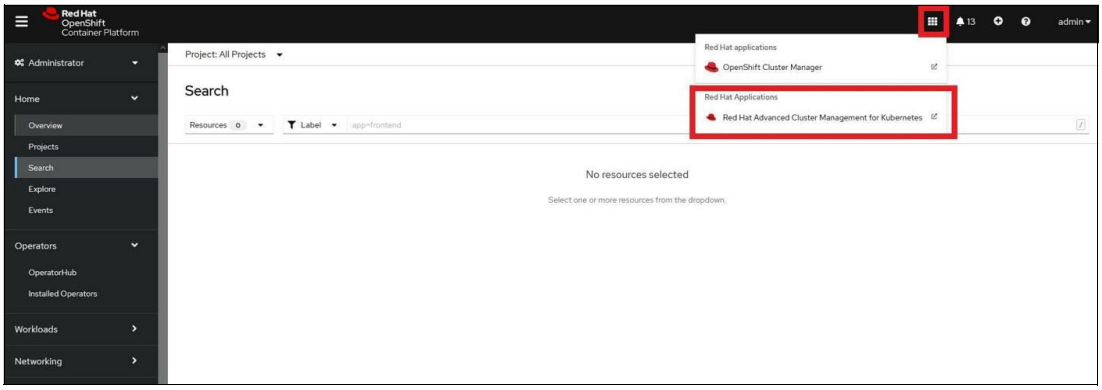


Figure 9-5 RH advanced Cluster Management for Kubernetes

Once installed you can use the RHACM console to quickly and easily deploy applications and policies on one or multiple clusters.

Red Hat Advanced Cluster Management for Kubernetes provides a secure and quick way to manage the lifecycle of your Red Hat OpenShift Container Platform clusters, applications, and operators from a secure single console.

8. Import the cluster into your Red Hat Advanced Cluster Management for Kubernetes console

   On installation of Red Hat Advanced Cluster Management for Kubernetes on a cluster, the local cluster is automatically imported into Red Hat Advanced Cluster Management for Kubernetes. This step is only necessary to manage multiple clusters. A user can import additional clusters to Red Hat Advanced Cluster Management for Kubernetes by following the import a target cluster documentation available at Red Hat

   Users can import clusters from different Kubernetes cloud providers. After import, the targeted cluster becomes a managed cluster for the Red Hat Advanced Cluster Management for Kubernetes hub cluster.

> **Tip:** A hub cluster cannot manage any other hub cluster, but can manage itself. The hub cluster is configured to be automatically imported and self-managed. You do not need to manually import the hub cluster. However, if a hub cluster is removed and you need to import it again, the `local-cluster:true` label needs to be added.

9.3.2 Importing additional clusters for management

After you install Red Hat Advanced Cluster Management for Kubernetes, you are ready to import a cluster to manage. This can be done either from the console or from the CLI. This requires that you have cluster administration privileges for the user doing the import.

Prerequisites

Ensure that you have met the following prerequisites:

- You need a Red Hat Advanced Cluster Management for Kubernetes hub cluster that is deployed. If you are importing bare metal clusters, you must have the hub cluster installed on Red Hat OpenShift Container Platform version 4.6 or later.
- You need a cluster that you want to manage and Internet connectivity.
- You need to Install kubectl. To install kubectl, see Install and Set Up kubectl in the Kubernetes documentation.
- You need the base64 command line tool.
- If you are importing a cluster that was not created by Red Hat OpenShift Container Platform, you need a *multiclusterhub.spec.imagePullSecret* defined. See section 9.4, “Custom image pull secret” on page 142 for more information about defining the secret.
- If you are importing in a Red Hat OpenShift Dedicated environment:
  - You must have the hub cluster deployed in a Red Hat OpenShift Dedicated environment.
  - The default permission in Red Hat OpenShift Dedicated is dedicated-admin, but that does not contain all of the permissions to create a namespace. You must have cluster-admin permissions to import and manage a cluster with Red Hat Advanced Cluster Management for Kubernetes.

Note: A hub cluster cannot manage a different hub cluster. A hub cluster is set up to automatically import and manage itself, so you do not have to manually import a hub cluster to manage itself.

Importing cluster

To import an existing cluster follow these steps:

1. From the navigation menu, select Infrastructure > Clusters.
2. Click **Add a cluster**.
3. Click **Import an existing cluster**.
4. Provide a name for the cluster. By default, the namespace is used for the cluster name and namespace.
5. Optional: Add any additional labels.

Note: If you import a Red Hat OpenShift Dedicated cluster and do not specify a vendor by adding a label for vendor=OpenShiftDedicated, or if you add a label for vendor=auto-detect, a managed-by=platform label is automatically added to the cluster. You can use this added label to identify the cluster as a Red Hat OpenShift Dedicated cluster. This allows you to retrieve any Red Hat OpenShift Dedicated clusters as a group.

6. Select the import mode that you want to use to identify the cluster that you are importing from the following options:
   a. Run import commands manually:
      i. Generate import commands that you can copy and run, based on the information that you provided. Click **`Save import and generate code`** to generate the command that you use to deploy the *open-cluster-management-agent-addon*. A confirmation message is displayed.

ii. In the Import an existing cluster window, select the Copy command to copy the generated command and token to the clipboard.

iii. Log in to the managed cluster that you want to import.

Important: The command contains pull secret information that is copied to each of the imported clusters. Anyone who can access the imported clusters can also view the pull secret information. Consider creating a secondary pull secret at https://cloud.redhat.com/ or create a service account to protect your personal credentials. See Using image pull secrets or Understanding and creating service accounts for more information about pull secrets.

b. For the Red Hat OpenShift Dedicated environment only Complete the following steps:

i. Create the open-cluster-management-agent and open-cluster-management namespaces or projects on the managed cluster.

ii. Find the klusterlet operator in the Red Hat OpenShift Container Platform catalog.

iii. Install it in the open-cluster-management namespace or project that you created.

Important: Do not install the operator in the open-cluster-management-agent namespace.

7. Extract the bootstrap secret from the import command by completing the following steps:

a. Select **Infrastructure** → **Clusters** from the Red Hat Advanced Cluster Management console main navigation.

b. Select **Add a cluster** → **Import an existing cluster**.

c. Add the cluster information, and select **Save import and generate code**.

d. Copy the import command.

e. Paste the import command into a file that you create named import-command.

f. Run the following command to insert the content into the new file:

```
cat import-command | awk '{split($0,a,"&&"); print a[3]}' | awk
'{split($0,a,"|"); print a[1]}' | sed -e "s/^ echo //" | base64 -d
```

g. Find and copy the secret with the name bootstrap-hub-kubeconfig in the output.

h. Apply the secret to the open-cluster-management-agent namespace on the managed cluster.

i. Create the klusterlet resource using the example in the installed operator, the clusterName should be changed to the same name as the cluster name that was set during the import.

Note: When the managedcluster resource is successfully registered to the hub, there are two klusterlet operators installed. One klusterlet operator is in the open-cluster-management namespace, and the other is in the open-cluster-management-agent namespace. Multiple operators does not affect the function of the klusterlet.

8. For cluster imports that are not in the Red Hat OpenShift Dedicated environment Complete the following steps:

a. If necessary, configure your kubectl commands for your managed cluster.

See Supported clouds to learn how to configure your kubectl command line interface (CLI).

b. To deploy the open-cluster-management-agent-addon to the managed cluster, run the command and token that you copied.

c. Select View cluster to view a summary of your cluster in the Overview page.

d. Enter your server URL and API token for the existing cluster: Provide the server URL and API token of the cluster that you are importing.

e. Kubeconfig: Copy and paste the content of the kubeconfig file of the cluster that you are importing.

9. Optional: Configure the Cluster API address that is on the cluster details page by configuring the URL that is displayed in the table when you run the **`oc get managedcluster`** command.

   a. Log in to your hub cluster with an ID that has cluster-admin permissions.

   b. Configure your kubectl for your targeted managed cluster.

   c. Edit the managed cluster entry for the cluster that you are importing by entering the following command:

      `oc edit managedcluster <cluster-name>`

      Replace cluster-name with the name of the managed cluster.

   d. Add the ManagedClusterClientConfigs section to the ManagedCluster spec in the YAML file, as shown in Example 9-1.

Example 9-1

```
spec:
hubAcceptsClient: true
managedClusterClientConfigs:
url: https://multicloud-console.apps.new-managed.dev.redhat.com
```

      Replace the value of the URL with the URL that provides external access to the managed cluster that you are importing.

   Your cluster is imported. You can import another by selecting Import another.

10. View your managed clusters.

    With Red Hat Advanced Cluster Management for Kubernetes instance running on hub cluster and one or more clusters managed with Red Hat Advanced Cluster Management for Kubernetes. To see your managed clusters from the console, in the left panel, click **Infrastructure** → **Clusters**.

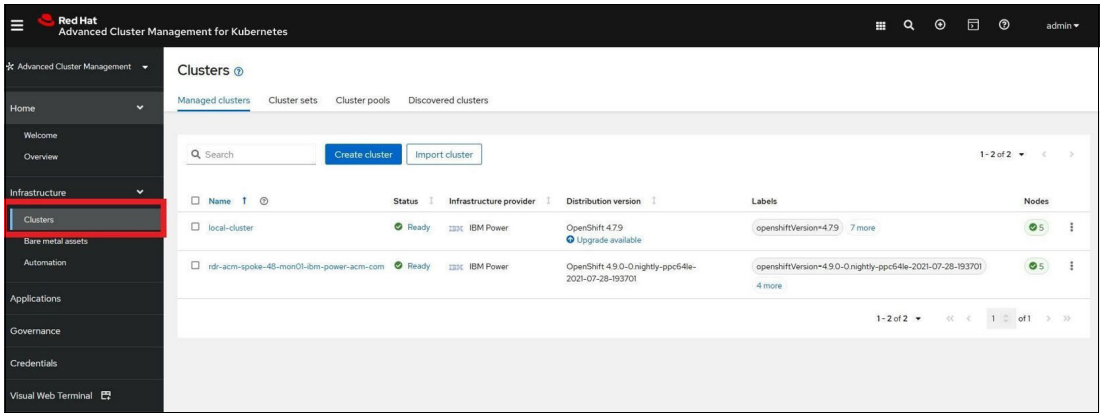


Figure 9-6 Infrastructure number

The number of entries in the cluster tab can vary depending on the number of clusters imported.

9.4 Custom image pull secret

If you plan to import Kubernetes clusters that were not created by Red Hat OpenShift Container Platform or Red Hat Advanced Cluster Management, generate a secret that contains your Red Hat OpenShift Container Platform pull secret information to access the entitled content from the distribution registry.

The secret requirements for Red Hat OpenShift Container Platform clusters are automatically resolved by Red Hat OpenShift Container Platform and Red Hat Advanced Cluster Management, so you do not have to create the secret if you are not importing other types of Kubernetes clusters to be managed. Your Red Hat OpenShift Container Platform pull secret is associated with your Red Hat Customer Portal ID, and is the same across all Kubernetes providers.

Important: These secrets are namespace-specific, so make sure that you are in the namespace that you use for your hub cluster.

Follow these steps:

1. Go to cloud.redhat.com/openshift/install/pull-secret to download the OpenShift Container Platform pull secret file.
2. Click Download pull secret.
3. Run the following command to create your secret:

   ```
   oc create secret generic <secret> -n <namespace>
   --from-file=.dockerconfigjson=<path-to-pull-secret>
   --type=kubernetes.io/dockerconfigjson
   ```

   Replace <**secret**> with the name of the secret that you want to create, replace **<namespace>** with your project namespace (secrets are namespace-specific), and replace <**path-to-pull-secret**> with the path to your Red Hat OpenShift Container Platform pull secret that you downloaded.

   Example 9-2 displays the **spec.imagePullSecret** template to use if you want to use a custom pull secret. Replace <secret> in the template with the name of your pull secret.

Example 9-2 Pull secret templet

```
apiVersion: operator.open-cluster-management.io/v1
kind: MultiClusterHub
metadata:
  name: multiclusterhub
  namespace: <namespace>
spec:
  imagePullSecret: <secret>
```

10

Monitoring and Logging

Running a Red Hat OpenShift cluster involves keeping track of its overall health. Crucial aspects include understanding the cluster's status, tracking application performance, and looking out for unusual activities signaling potential security issues. This chapter focuses on the log files generated within the Red Hat OpenShift ecosystem and offers suggestions on which metrics to watch to maintain smooth operation of your cloud-native applications.

The following topics are covered:

10.1 Monitoring tools and techniques

Managing contemporary software applications has become increasingly difficult due to their migration towards cloud-based Agile development approaches. Traditional monitoring and logging tools have been unable to keep pace with this shift. To address this issue, modern infrastructure administrators need tools and platforms that emphasize observability within the infrastructure itself.

Observability goes beyond traditional monitoring by attempting to comprehend the inner workings of a system through understanding the majority of potential external outcomes it generates. This heightened awareness enables quicker issue detection and resolution. In essence, observability represents an advancement of conventional application performance monitoring, which equips organizations with essential management capabilities for complex, dynamically adjusting application ecosystems characterized by frequent modifications to their operational components. Conventional monitoring techniques, including those associated with Application and Performance Monitoring (APM), which sample data only once every minute, struggle to maintain relevance amidst these evolving conditions.

Observability encompasses the collection and analysis of various types of telemetry data—logs, metrics, traces, and dependencies—to provide comprehensive, contextually rich insights. These insights empower SREs, DevOps teams, and other IT professionals to swiftly identify and rectify performance bottlenecks, for instance. A crucial characteristic of observability solutions is their inherent automation, enabling them to automatically detect newly integrated telemetry sources and filter out irrelevant data, thereby distinguishing themselves from traditional monitoring methods. The primary advantage of observability lies in its capacity to uncover and tackle the "unknowns," offering superior value compared to traditional monitoring approaches.

This chapter provides an insight into some of the tools available to assist in managing modern application environments.

10.1.1 Sysdig

Sysdig is a software as a service (SaaS) provider for security and monitoring tools. It can help embed security and compliance management into DevOps workflows.

Sysdig is designed to:

- Provide Infrastructure as code (IaC) security providing the ability to shift security further left. Shift left security is a proactive strategy that involves integrating security practices and testing early in the software development lifecycle (SDLC). This approach aims to identify and fix defects before they become more difficult to address later in the process.
- Provide ongoing Cloud Security Posture Management through constant monitoring of cloud-based systems, identifying misconfiguration, potential threats, and unwarranted access levels.
- Implement vulnerability management through consolidation of container and host security scans within development operations workflows.
- Combine and harmonize threat detection with incident response across container environments, Kubernetes deployments, and clouds.
- Increase network security through the use of micro-segmentation within the network, implementing and automating a native network policy utilizing Kubernetes features.
- Provide monitoring with deep insight by using Kubernetes-native Prometheus.

- Validate compliance with major security standards like PCI, SOC2, and NIST for containers and hosts.

The Sysdig platform is split into two major parts: Sysdig Secure and Sysdig Monitor.

Sysdig Secure

Sysdig Secure is a key component of Sysdig's container intelligence platform, offering a unified solution for security, monitoring, and forensics within cloud, container, and microservices environments. Integrated with Docker and Kubernetes, Sysdig Secure provides a services-aware approach to protect workloads while delivering comprehensive visibility across cloud and container ecosystems. Figure 10-1 shows the capabilities of Sysdig Secure.

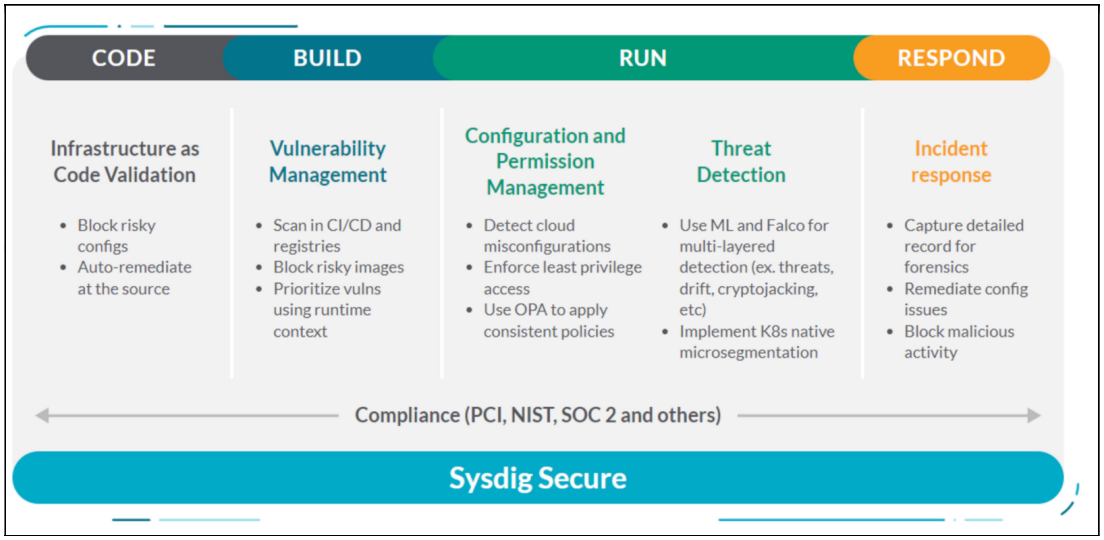


Figure 10-1 Sysdig Secure

Key features of Sysdig Secure include:

- Posture Management: Compliance and benchmark assessments, along with Cloud Infrastructure Entitlement Management (CIEM).
- Vulnerability Scanning: Identifying and addressing vulnerabilities in your environment.
- Forensics: Detailed analysis and investigation of security incidents.
- Threat Detection and Blocking: Real-time identification and prevention of potential threats.
- Sysdig Secure ensures robust protection and visibility for your cloud-native applications.

Sysdig Monitor

Figure 10-2 on page 146 shows Sysdig Monitor.

Sysdig Monitor is a comprehensive suite for monitoring, troubleshooting, cost optimization, and alerting, offering in-depth, process-level visibility into dynamic and distributed production environments. It captures, correlates, and visualizes full-stack data, providing dashboards designed to monitor cloud-native environments effectively.

The Sysdig agent operates on the monitored hosts, collecting relevant metrics and events. By default, the agent reports a broad range of predefined metrics. For additional metrics and custom parameters, you can use agent configuration files and integrations with monitoring and cloud services.

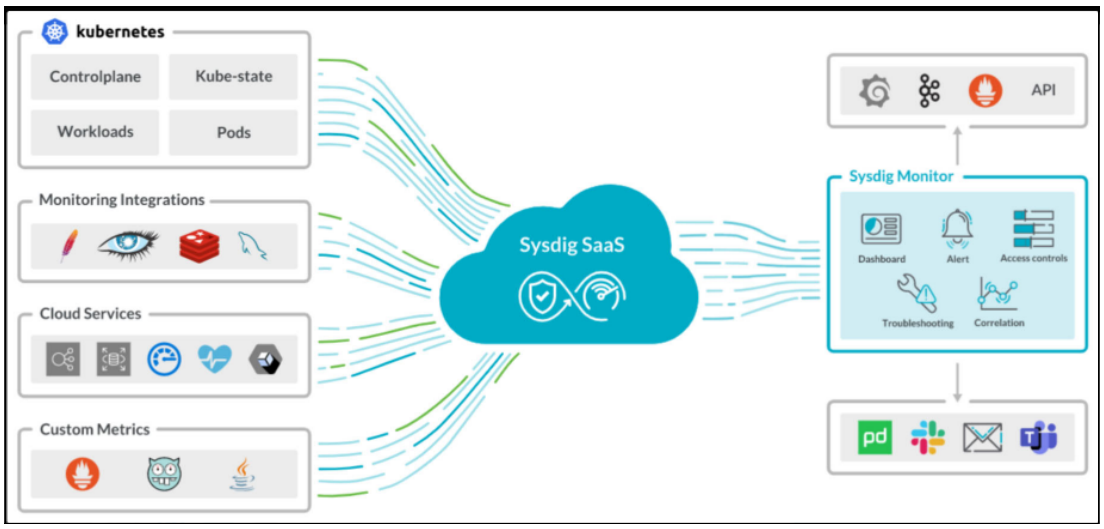


Figure 10-2 Sysdig monitor

Sysdig is built on an open-source security stack that provides accelerated innovation and standardization. Its monitoring capabilities are built on Prometheus, so Red Hat OpenShift workloads on IBM Power Architecture® based servers can be monitored in a standard way. *Sysdig* is an agent-based service. Monitored sources can be Kubernetes clusters, Red Hat OpenShift clusters, Linux machines, and Docker containers.

10.1.2 Prometheus

Prometheus is an open-source systems monitoring and alerting toolkit that originally was built by SoundCloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has an active developer and user community. Prometheus is now a stand-alone, open-source project that is maintained independently of any company. To emphasize this situation and clarify the project's governance structure, Prometheus joined the Cloud Native Computing Foundation (CNCF) in 2016 as the second hosted project after Kubernetes.

Prometheus collects and stores its metrics as time series data, that is, metrics are stored with the timestamp at which it was recorded, along with optional key-value pairs that are called labels.

Red Hat OpenShift contains a full-featured monitoring stack that you can use to monitor the cluster health by default, but you can use it for user projects too. A key part of this stack is Prometheus.

Figure 10-3 on page 147 shows the parts of the monitoring stack that are installed by default. They monitor the Red Hat OpenShift platform base and controlling components. These components are deployed into the `openshift-monitoring` namespace.

Also, user-defined projects and applications can be monitored if enabled. In this case, there will be a new namespace that is created and named `openshift-user-workload-monitoring`.

Prometheus, as part of Red Hat OpenShift Monitoring, offers a time-series database and a rule engine for metrics while sending alerts to Alertmanager. Under its control, the Prometheus Operator manages Prometheus instances and automatically creates monitoring target configurations according to Kubernetes labels. The monitoring of operator system metrics and nodes is handled by a node-exporter agent.

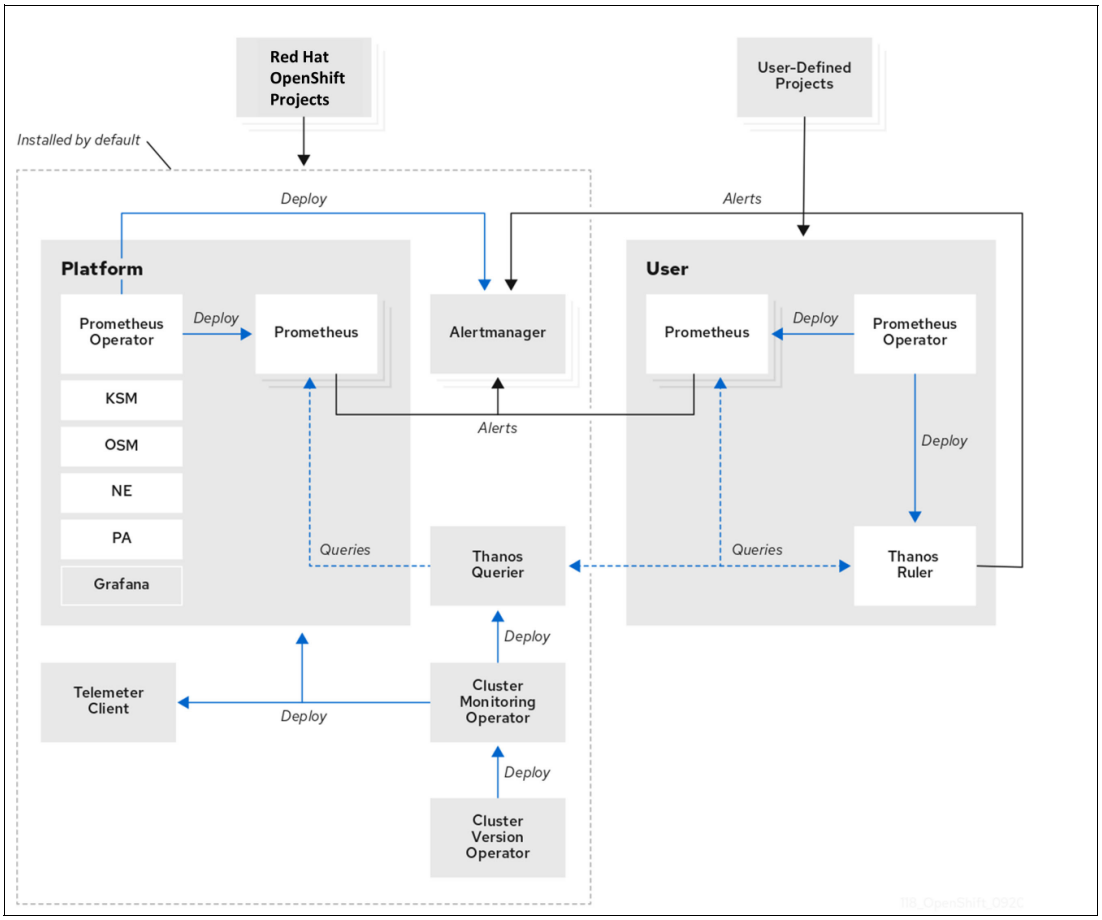


Figure 10-3 Red Hat OpenShift Monitoring stack

Red Hat OpenShift provides an observability framework, and the GUI has the following features for this framework These are found in the **Observe** menu:

- Alerting

  This provides visibility to alerting rules that are based on metrics that are collected by Prometheus. This includes silencers (to silence a predefined alerting rule) and the alerts that are fired by alerting rules.

- Metrics

  This window provides a GUI to build custom queries that are based on the metrics that are collected by Red Hat OpenShift Monitoring.

- Dashboards

  Preconfigured dashboards are provided for monitoring. Clicking **Inspect** on a dashboard element allows the user to dig deeper into that element as well as showing or editing the query that produced that element.

- Targets

  Shows all the monitoring targets that are supported by the platform.

If you go to the Dashboard window by selecting **Observe** → **Dashboard**, you see two dashboards that are based on `node-export` by default:

- Node Exporter / USE Method / Cluster:

  This dashboard has cluster-wide data by nodes.

- Node Exporter / USE Method / Node: This dashboard has the same data, but only for the selected node.

Figure 10-4 shows the cluster dashboard.

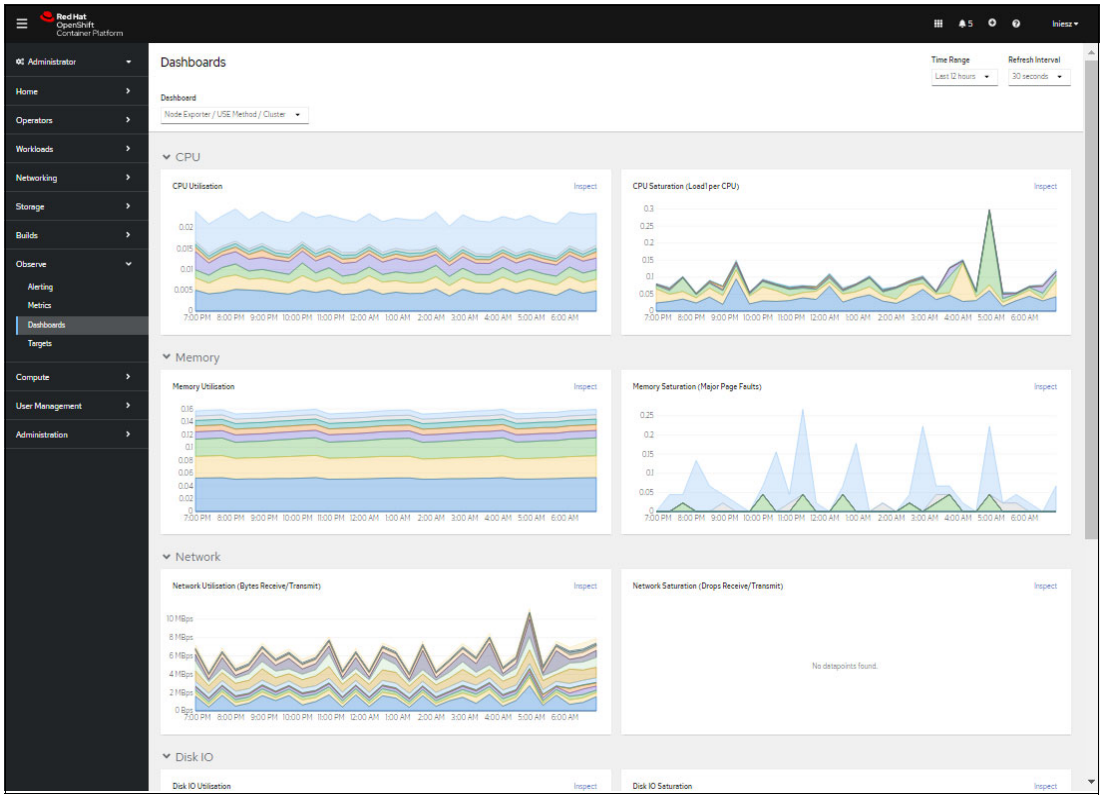


Figure 10-4 Node Exporter / USE Method / Cluster

10.1.3 Grafana

Grafana is a widely used operational dashboard that can be connected to many metric collection systems, such as Prometheus. Grafana provided the dashboard service in earlier versions of Red Hat OpenShift, but the service is deprecated in Version 4.10 and was removed starting with Version 4.11. A similar function is available by using Red Hat OpenShift dashboards, which you can view by selecting **Observe** → **Dashboards**.

Configure Grafana by defining the data sources, typically the integrated Prometheus service in Red Hat OpenShift. To do this, create a secret named grafana-datasources-v2 containing the default data source configuration (shown in Example 10-1) or specify custom data sources during setup.

Example 10-1 The prometheus.yaml file

```
{
    "apiVersion": 1,
    "datasources": [
        {
            "access": "proxy",
            "basicAuth": true,
            "secureJsonData": {
                "basicAuthPassword":
"8BCcK+MXnQUwzgYGcKEq5k+InSNWXZkueMRbZQvn8T/DwdFQ4XpfcKv5g76gjKfZgxQZBAJajyiQkRKJw
```

```
kRJk4hecJUHwlSCuOwGTml6/NCxVomCWvnMNb7pYZYWSDvOQGDpK6VCO66RdLhX7vF+SUZEe33/x4A+5iw
EN78wWtoMx+Ehb7WboET7jESxqpr1Yanj5Nr+bLeLslcwj2qYHOgxkrwKZBmO6YqKrYSeNqbJfmOkvwuFa
/QPQJBt8hEI4xBFWchTSOBrHaSjKhC/8zd+Z7mtK/i9VGioKEqEXOzYRAxGBh4GqdjXlTn233tFBgypScD
tZ9FgJawhMIoy"
            },
            "basicAuthUser": "internal",
            "editable": false,
            "jsonData": {
                "tlsSkipVerify": true
            },
            "name": "prometheus",
            "orgId": 1,
            "type": "prometheus",
            "url": "https://prometheus-k8s.openshift-monitoring.svc:9091",
            "version": 1
        }
    ]
}
```

This secret and all others are stored in `/etc/grafana/provisioning/datasources` of the Grafana pod directory. You can choose these secrets to create a dashboard from the collected data.

The data source can be local, in which case use the internal service URL, or it can be remote. A single Grafana server can show dashboards for many remote data sources. Dashboards are configured as JSON files. In Red Hat OpenShift, the configuration of the dashboards is stored in ConfigMaps, which are stored in a directory that is specified by another configmap called `grafana-dashboards`. Many specialized dashboard configurations can be downloaded from Grafana Dashboards.

10.1.4 IBM Instana

IBM's Instana offers an Enterprise Observability Platform designed for businesses operating intricate, contemporary, and cloud-native applications across various environments, including on-premises infrastructure, public and private clouds, mobile devices, and IBM Z mainframe computers. This platform integrates automated application performance monitoring capabilities powered by AI-driven discovery of deep contextual dependencies within hybrid applications.

Instana ensures comprehensive visibility into development pipelines, fostering closed-loop DevOps automation. By providing actionable insights, it enables organizations to enhance application performance, stimulate innovation, and minimize risks. Consequently, DevOps teams can improve operational efficiencies, contribute more significantly to software delivery pipelines, and meet their service-level and business-level goals.

Features

Instana provides the following features:

- Automated discovery by using a lightweight agent and sensors that automatically collect data with a 1-second granularity. Every request that is made by microservices is traced, and the response time and context is captured. This data is enhanced with other related metrics to produce a complete picture of the applications and infrastructure.
- A dependency map built using the gathered data.

- Assist with root cause analysts by analyzing the incoming data in real time and creating issues and incidents that are raised if users are impacted. An incident includes metrics, traces, exceptions, logged data, and configuration data, which are correlated through the Dynamic Graph.
- Performance optimization through Unbounded Analytics, which uses collected trace information. The information can be filtered for performance outliers, patterns of known problem, and traces that are tagged uniquely.

Instana uses sensors to provide automated infrastructure and application monitoring with no plug-ins or application restarts. Each sensor supports an application component, middleware component, operating system, or other integration point so that you can manage and monitor your infrastructure. At the time of writing, Instana has the following integrations listed by the type of integration and the number of different tools:

- AI Ops integrations (18):
  - CI/CD Automation (7)
  - DevOps Tools (9)
- Cloud Operations (24)
- Containers and Orchestration (23)
- User Monitoring (3)
- Infrastructure and Middleware Components (114):
  - Database (37)
  - Messaging (25)
  - OS (11)
  - Web / App Servers (39)
- Kubernetes Distributions (5)
- Legacy Middleware (3)
- Secrets and Identity Management (2)
- Serverless (4)
- Tracing, Supported Languages, and Frameworks (34):
  - Application Frameworks (7)
  - Application Monitoring (17)
  - Proxies and Service Meshes (4)
  - Tracing Technology (7)

Instana has a GUI that can be used through a web browser.

Instana architecture

Instana provides automatic, continuous discovery of your application stack. A single, lightweight agent per host continually discovers all the components and deploys sensors that are crafted to monitor each technology. With no human intervention, sensors automatically collect configuration, changes, metrics, and events. Metrics from all components are collected in high fidelity with a 1-second data granularity as every request is traced across each microservice, automatically capturing the response time and context.

To understand how a system of services works together and the impact of component failure, Instana enhances traces with information about the underlying service, application, and system infrastructure by using the Dynamic Graph. The Dynamic Graph provides a dependency map so that you can get to the root cause of issues quickly.

Figure 10-5 shows the architecture of the Instana Enterprise Observability platform.

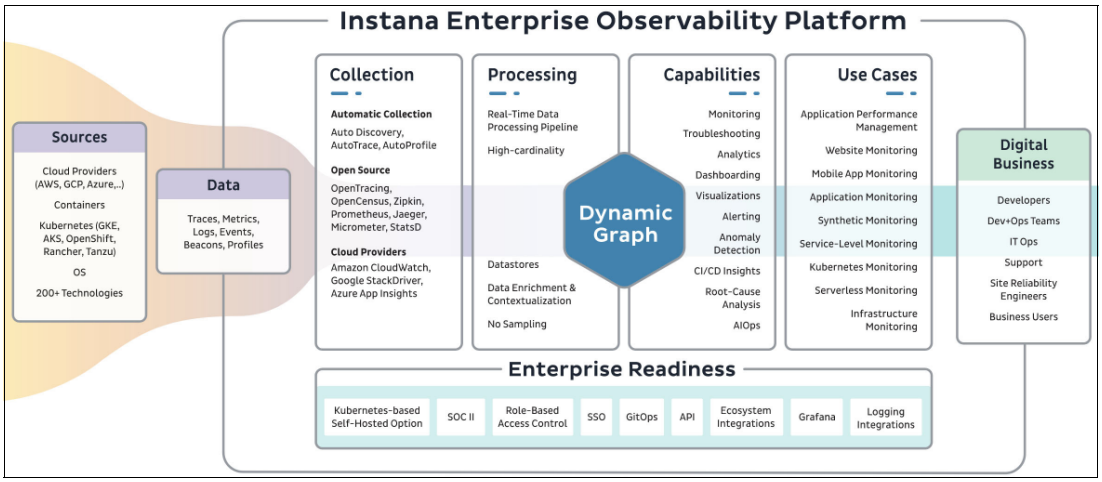


Figure 10-5 Instana architecture

On Red Hat OpenShift and Kubernetes clusters, Instana agents are defined and running as pods that are managed by a DaemonSet, which means that all worker nodes have an agent running with the same configuration by default. The configuration is managed as a configmap in Red Hat OpenShift. The Red Hat OpenShift cluster nodes can be seen in the Instana Infrastructure window, as shown in Figure 10-6.

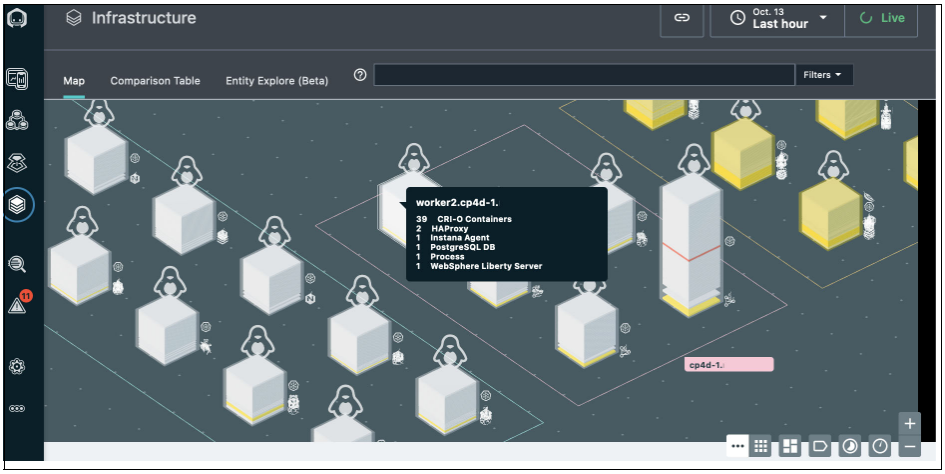


Figure 10-6 Instana: Infrastructure window

10.2 Log management in Red Hat OpenShift

In the world of container orchestration, logging is indispensable for monitoring, troubleshooting, and securing applications. Red Hat OpenShift, built on Kubernetes, provides robust solutions for managing container logs. This chapter delves into the various logging mechanisms available in Red Hat OpenShift, outlining their functionalities, configurations, and best practices for effective log management.

10.2.1 Understanding logs in Red Hat OpenShift

Red Hat OpenShift generates several types of logs, crucial for the administration and troubleshooting of both the platform and the applications running on it.

Container logs

Controller logs contain stdout and stderr streams from containers, capturing application behavior and errors. These logs provide insights into the application behavior and are vital for diagnosing issues with applications deployed on Red Hat OpenShift.

Consider these items as best practices in managing container logs

- Implement structured logging in applications to facilitate easier querying and analysis.
- Set appropriate log rotation and retention policies to manage storage consumption effectively.
- Use labels and annotations judiciously to enrich log entries with context, aiding in more precise filtering and searches.

Node logs

Node logs contain system logs related to Red Hat OpenShift nodes, including system services and container runtime operations.

These are the node logs captured by Red Hat OpenShift:

- Systemd Logs - Generated by the system and service manager on nodes.
- Kubelet Logs - Related to the management of containers on the node.
- API Server Logs, etcd Logs, etc. - For master nodes, logs from various control plane components are crucial.
- Audit Logs - Security-related logs that record sequences of activities or changes that affect the operation of Red Hat OpenShift.

These are the suggestions for best practices in managing node logs:

- Adjust collection strategies as necessary by monitoring log volume and any performance impact from log collection.
- Secure access to node logs, given they can contain sensitive information about the node and cluster operations.
- Integrate log monitoring with system health monitoring for proactive issue detection and resolution.

Audit logs

Audit logs provide a record of the events that have occurred within the Red Hat OpenShift cluster, particularly with respect to the API server. They are crucial for security and compliance, helping administrators track changes to the cluster and understand who did what, when, and from where. Audit logs typically contain metadata such as timestamps, user and source IP address. In addition to api endpoint, resource requests and responses

The best practice recommendations for audit log management are:

- Define audit log granularity based on security requirements and compliance needs.
- Regularly review and analyze audit logs to detect unauthorized access or anomalous activities early.
- Ensure audit logs are included in backup procedures and protected from unauthorized access.

10.2.2 Relevance in Multi-Architecture clusters

In a multi-architecture cluster environment, where nodes might have different CPU architectures (such as x86, ARM, and IBM Power), managing logging can become complex due to the heterogeneity of the environment. The Red Hat OpenShift Cluster Logging Operator is particularly relevant in such scenarios for the following reasons.

Unified logging across architectures:

The operator ensures a consistent logging experience across different node architectures. It abstracts the underlying differences and provides a uniform interface for log collection and management.

Compatibility and portability

By supporting multiple architectures, the operator ensures that the logging stack components (Elasticsearch, Fluentd, and Kibana) can run on various CPU architectures without compatibility issues. This is crucial for organizations leveraging diverse hardware to optimize performance and cost.

Centralized management

The operator enables centralized management of the logging infrastructure, simplifying the administrative burden in a heterogeneous cluster. Administrators can define and apply logging policies uniformly across all nodes, regardless of their architecture.

Resource optimization

In multi-architecture clusters, different nodes may have varying resource capacities. The operator helps optimize resource usage by efficiently distributing the logging workload based on node capabilities, ensuring balanced performance and avoiding bottlenecks.

Resilience and redundancy

The operator enhances the resilience of the logging infrastructure by supporting distributed deployments across multiple architectures. This redundancy ensures that logging services remain available even if some nodes fail, contributing to the overall reliability of the cluster.

Compliance and security

Consistent logging across different architectures aids in maintaining compliance with industry standards and regulations. The operator ensures that security logs are collected uniformly, facilitating comprehensive security monitoring and incident response.

10.2.3 Installing cluster logging operator

Before installing the Red Hat OpenShift Cluster Logging Operator, ensure that your environment meets the following prerequisites:

1. Ensure you have a Red Hat OpenShift cluster running version 4.5 or later.
2. You need cluster-admin privileges to install and configure the logging operator.
3. Ensure that Persistent Volume (PV) storage is available and configured in your cluster for Elasticsearch.

Installation guide

Follow these steps to install the Red Hat OpenShift Cluster Logging Operator:

1. Create the `openshift-logging` Project

   Start by creating a new project for the logging components:

   ```
   oc create namespace openshift-logging
   ```

2. Create the `openshift-operators-redhat` Project

   Create another project for the logging operator to reside:

   ```
   oc create namespace openshift-operators-redhat
   ```

3. Install the Elasticsearch Operator

   The Cluster Logging Operator relies on the Elasticsearch Operator for managing Elasticsearch instances. Install it first:

   a. Navigate to the Red Hat OpenShift Web Console.
   b. Go to the *OperatorHub*.
   c. Search for *Elasticsearch Operator*.
   d. Click *Install* and choose the *openshift-operators-redhat* namespace for the installation.

4. Install the Cluster Logging Operator

   a. In the Red Hat OpenShift Web Console, go to the *OperatorHub*.
   b. Search for *Red Hat OpenShift Logging*.
   c. Click *Install* and choose the *openshift-operators-redhat* namespace for the installation.

5. Create an Instance of the Cluster Logging Operator

   Once the operator is installed, you need to create an instance to deploy the logging stack:

   a. Create a *ClusterLogging* custom resource (CR) to deploy the logging components. Save the YAML definition shown in Example 10-2 to a file named *cluster-logging.yaml.*

Example 10-2 Custom resource definition for cluster logging

```
apiVersion: "logging.openshift.io/v1"
    kind: "ClusterLogging"
    metadata:
      name: "instance"
      namespace: "openshift-logging"
    spec:
      managementState: "Managed"
      logStore:
        type: "elasticsearch"
        retentionPolicy:
          application:
            maxAge: 7d
          infra:
            maxAge: 7d
        elasticsearch:
          nodeCount: 3
          storage:
            storageClassName: "gp2"
            size: "200Gi"
          resources:
            limits:
              memory: 16Gi
            requests:
              memory: 16Gi
```

```
visualization:
  type: "kibana"
  kibana:
    replicas: 1
collection:
  logs:
    type: "fluentd"
    fluentd: {}
```

b. Apply the YAML definition to your Red Hat OpenShift cluster:

```
oc apply -f cluster-logging.yaml
```

6. Verify the Installation

   Monitor the status of the logging components to ensure they are deployed correctly:

   ```
   oc get pods -n openshift-logging
   ```

   You should see pods for Elasticsearch, Fluentd, and Kibana running in the `openshift-logging` namespace.

Post-installation configuration

After the installation, you may want to customize the logging configuration based on your requirements:

1. Adjust Retention Policies

   Modify the retention policies for application and infrastructure logs to meet your organization's data retention needs.

2. Resource Allocation

   Ensure that the resource requests and limits for Elasticsearch and other components are appropriately set to handle your log volume.

3. Access Control

   Configure role-based access control (RBAC) to restrict access to the Kibana dashboards and other logging components.

10.3 Performance Tuning and Optimization

Performance tuning and optimization are essential for attaining maximum efficiency and response times for applications hosted within Red Hat OpenShift. To accomplish this, adjustments should be made to both the Red Hat OpenShift platform and the applications themselves. Based on our expertise, here are some practical performance tuning and optimization techniques for Red Hat OpenShift :

- Properly configure resource requests and limits for pods to ensure that applications have enough resources to perform optimally while preventing any single application from consuming excessive cluster resources.
- Use Quality of Service (QoS) classes in Red Hat OpenShift to manage resource allocation policies effectively.
- Utilize the Node Tuning Operator to automatically apply node-level tuning measures. This operator can adjust system parameters on Red Hat OpenShift nodes according to predefined or custom profiles to optimize the performance of both the nodes and the workloads they host.

- Optimize the Red Hat OpenShift scheduler to ensure it places pods on nodes in a way that balances the load effectively and reduces resource contention. Scheduler enhancements might include configuring pod affinity and anti-affinity, taints, and tolerations to control pod placement based on your workload needs.
- Define and implement network policies that control the flow of traffic between pods. Efficient network policies not only secure pod traffic but also reduce unnecessary network overhead by restricting the communications to only required pathways.
- Use Red Hat OpenShift's router sharding and load balancing features to distribute client requests efficiently across multiple pods. Fine-tuning these settings can reduce latency and increase the throughput of applications.
- For applications that require high network performance, consider enabling features like SR-IOV (Single Root Input/Output Virtualization) or deploying network plugins that support Multi-Protocol Label Switching (MPLS), especially in data-intensive environments.
- Choose the appropriate storage solutions based on your performance needs. For I/O-intensive applications, use faster storage classes such as SSD-based persistent volumes. Additionally, configure volume IOPS limits and throughput parameters to align with your application needs.
- Implement caching mechanisms to reduce read access times and offload I/O operations from the primary storage. Caches can be configured at various layers, including within applications, databases, or by using Red Hat OpenShift integrated caching solutions like Redis or Memcached.
- Utilize data locality features to place pods close to their data sources in multi-node clusters, reducing latency and improving response times.
- Regularly profile application performance using tools integrated into Red Hat OpenShift like Prometheus and Grafana. Identify bottlenecks and optimize code or architecture based on real usage data.
- For applications structured as microservices, ensure that each microservice is scaled appropriately and is independently deployable, which can improve the overall resilience and efficiency of the application.
- Manage application concurrency settings to optimize the use of CPU and memory resources. This includes tuning thread pools, database connections, and other concurrent processes to match the load.
- Implement a continuous performance optimization strategy where feedback from monitoring tools is regularly analyzed, and improvements are systematically implemented.
- Maintain environments that mirror the production settings as closely as possible. Use these environments for load testing and performance experiments to fine-tune settings before rolling them out to production.
- Use progressive rollout strategies such as canary deployments or blue-green deployments to minimize the impact of changes and assess performance improvements in a controlled manner.

By systematically addressing these areas, Red Hat OpenShift administrators and developers can ensure that their clusters and applications are not only stable and secure but also optimized for high performance. This holistic approach to performance tuning helps in creating a robust, scalable, and efficient cloud-native environment.

For additional information on performance in Red Hat OpenShift reference *Implementing, Tuning, and Optimizing Workloads with Red Hat OpenShift on IBM Power*, SG24-8537

11

Troubleshooting and Support

Running a Red Hat OpenShift environment can be challenging due to its numerous components and potential configuration issues. To effectively address problems, it is crucial to establish a systematic approach and a well-thought-out plan. A key aspect of problem resolution involves understanding how to reach assistance from support professionals and gathering essential information to diagnose and resolve any encountered challenges. This chapter is intended to give you some suggestions on troubleshooting any issue you might encounter and to provide you with some pointers on how to get support from experts.

This chapter covers the following topics:

11.1 Common issues and their resolutions

When encountering issues with OpenShift Multi-Architecture Compute on your IBM Power platform, start by identifying the problem type – installation, networking, control panel, or another issue. Different approaches require distinct sets of logs:

- Installation issue: Collect bootstrap and control plane machine logs for further investigation. See troubleshooting installation issues for details.
- Operating system issues: Focus on kdump logs for analysis. See troubleshooting operating system issues for guidance.
- Authentication and authorization concerns: Verify the user running the command with the command **`oc whoami`**. This helps identify whether the issue stems from insufficient privileges or misconfigured roles.

  If you are troubleshooting authentication and authorization problems, remember that Red Hat OpenShift uses advanced Role-Based Access Control (RBAC). By default, Red Hat OpenShift uses Kubernetes user accounts, which consist of trusted TLS certificates. The user *kubeadmin* is the cluster admin with access to everything, while a developer is defined with limited permissions.
- Application startup issues: Ensure resource commitment to the etcd database and initiate the containerized entry point application. Utilize the following commands for additional insights:
  - **`oc describe`** may reveal crashloopback errors.
  - **`oc logs`** shows the application's standard output.

Further troubleshooting guidance is provided in the next section.

11.2 Troubleshooting guidance

Troubleshooting Red Hat OpenShift installation, deployment, and performance issues require a systematic approach and understanding of the platform's architecture. Here are some best practices and examples to help you.

11.2.1 Installation Issues

For installation issues, follow these recommendations:

1. Check Documentation and Requirements:
   a. Ensure all prerequisites are met according to Red Hat documentation.
   b. Verify hardware and software compatibility with OpenShift versions.
   c. Review any specific network or storage requirements.
2. Logs and Diagnostics:
   a. Check installation logs (**`openshift-install.log`**) for errors or warnings.
   b. Check the status of cluster components (**`oc get pods --all-namespaces`**) for any failing pods.
   c. Run cluster health checks (**`oc adm top nodes, oc adm top pods`**) to monitor resource utilization, as shown in Example 11-1 on page 159.

Example 11-1 Check the status of cluster components

```
$ oc adm top nodes
W0705 16:21:07.555429   66389 top_node.go:119] Using json format to get metrics. Next
release will switch to protocol-buffers, switch early by passing --use-protocol-buffers
flag
NAME CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
p1296-master.p1296.cecc.ihost.com   1616m 10%    16510Mi 36%
Macs-MacBook-Pro-321:okd root# ./oc get pods --all-namespaces
NAMESPACE NAME READY   STATUS RESTARTS AGE
nfs-provisioner nfs-client-provisioner-57476c4b77-zm8zx 1/1 Running 13 (20h ago) 43h
openshift-apiserver-operator openshift-apiserver-operator-55fdbf86f7-sp9j4 1/1     Running
1 (43h ago)    43h
openshift-apiserver apiserver-ff76b5b8b-v6r4c 2/2 Running 0 43h
openshift-authentication-operator authentication-operator-6f6b668dcb-2cpbs 1/1     Running
1 (43h ago) 43h
openshift-authentication oauth-openshift-756b7f6b9f-b5tb2 1/1 Running 0 43h

(...)

openshift-service-ca service-ca-5746b94c7d-fw2qv 1/1     Running 0 43h
powervm-rmc powervm-rmc-ndt9z 1/1 Running 0 43h
ysledbooks django-psql-example-1-2j2tt 1/1     Running 0 5m44s
ysledbooks django-psql-example-1-build 0/1     Completed 0 6m57s
ysledbooks django-psql-example-1-deploy 0/1     Completed 0 5m45s
ysledbooks eap-xp4-basic-app-1-75vgd 1/1     Running 0 6m11s
ysledbooks eap-xp4-basic-app-1-deploy 0/1     Completed 0 6m12s
ysledbooks eap-xp4-basic-app-2-build 0/1     Completed 0 7m13s
ysledbooks eap-xp4-basic-app-build-artifacts-1-build 0/1     Completed 0 9m6s
ysledbooks httpd-example-1-697g5 1/1     Running 0 12m
ysledbooks httpd-example-1-build 0/1     Completed 0 13m
ysledbooks httpd-example-1-deploy 0/1     Completed 0 12m
ysledbooks ibm-spectrum-scale-csi-operator-7874679448-btjtc 1/1 Running 0 15m
ysledbooks jenkins-1-deploy 0/1     Completed 0 8m4s
ysledbooks jenkins-1-xf6t5 1/1     Running 0 8m3s
ysledbooks mariadb-1-deploy 0/1     Completed 0 13m
ysledbooks mariadb-1-z6knb 1/1     Running 0 13m
ysledbooks postgresql-1-deploy 0/1     Completed 0 6m56s
ysledbooks postgresql-1-fj4c6 1/1     Running 0 6m55s
ysledbooks sso-1-deploy 0/1     Completed 0 3m45s
ysledbooks sso-1-h4hpt 1/1     Running 1 (38s ago)    3m43s
ysledbooks sso-postgresql-1-6h4gj 1/1     Running 0 3m44s
ysledbooks sso-postgresql-1-deploy 0/1     Completed 0 3m45s
```

d. Verify that cluster components like etcd, Kubernetes controllers, and OpenShift operators are healthy (**`oc get clusterversion`**), as shown in Example 11-2.

Example 11-2 Check Cluster Version

```
$ oc get clusterversion
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.14.31   True        False         42h     Cluster version is 4.14.31
```

11.2.2 Deployment issues

For deployment issues, follow these recommendations:

Application Deployment Failures:

a. Check application deployment logs (**`oc logs <pod_name>`**) for errors, as shown in Example 11-3.

Example 11-3 Check application deployment logs

```
$ oc logs postgresql-1-fj4c6
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.utf8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/pgsql/data/userdata ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

pg_ctl -D /var/lib/pgsql/data/userdata -l logfile start

waiting for server to start....2024-07-05 15:14:34.740 UTC [39] LOG:  starting PostgreSQL
12.18 on powerpc64le-redhat-linux-gnu, compiled by gcc (GCC) 8.5.0 20210514 (Red Hat
8.5.0-20), 64-bit
2024-07-05 15:14:34.741 UTC [39] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2024-07-05 15:14:34.742 UTC [39] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
2024-07-05 15:14:34.749 UTC [39] LOG:  redirecting log output to logging collector process
2024-07-05 15:14:34.749 UTC [39] HINT:  Future log output will appear in directory "log".
 done
server started
/var/run/postgresql:5432 - accepting connections
=> sourcing /usr/share/container-scripts/postgresql/start/set_passwords.sh ...
ALTER ROLE
waiting for server to shut down.... done
server stopped
Starting server...
2024-07-05 15:14:35.238 UTC [1] LOG:  starting PostgreSQL 12.18 on
powerpc64le-redhat-linux-gnu, compiled by gcc (GCC) 8.5.0 20210514 (Red Hat 8.5.0-20),
64-bit
2024-07-05 15:14:35.239 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2024-07-05 15:14:35.239 UTC [1] LOG:  listening on IPv6 address "::", port 5432
2024-07-05 15:14:35.242 UTC [1] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2024-07-05 15:14:35.244 UTC [1] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
2024-07-05 15:14:35.252 UTC [1] LOG:  redirecting log output to logging collector process
```

```
2024-07-05 15:14:35.252 UTC [1] HINT:  Future log output will appear in directory "log".
```

b. Verify resource requests and limits (**oc describe pod <pod_name>**), as shown in Example 11-4.

Example 11-4 Verify resource requests and limits

```
$ oc describe pod postgresql-1-fj4c6
Name:         postgresql-1-fj4c6
Namespace:    ysledbooks
Priority:     0
Node:         p1296-master.p1296.cecc.ihost.com/129.40.94.251
Start Time:   Fri, 05 Jul 2024 16:14:17 +0100
Labels:       deployment=postgresql-1
              deploymentconfig=postgresql
              name=postgresql
Annotations:  k8s.ovn.org/pod-networks:

{"default":{"ip_addresses":["10.128.1.70/23"],"mac_address":"0a:58:0a:80:01:46","gateway_ip
s":["10.128.0.1"],"routes":[{"dest":"10.128.0.0...
              k8s.v1.cni.cncf.io/network-status:
                [{
                    "name": "ovn-kubernetes",
                    "interface": "eth0",
                    "ips": [
                        "10.128.1.70"
                    ],
                    "mac": "0a:58:0a:80:01:46",
                    "default": true,
                    "dns": {}
                }]
              openshift.io/deployment-config.latest-version: 1
              openshift.io/deployment-config.name: postgresql
              openshift.io/deployment.name: postgresql-1
              openshift.io/scc: restricted-v2
              seccomp.security.alpha.kubernetes.io/pod: runtime/default
Status:       Running
IP:           10.128.1.70
IPs:
  IP:           10.128.1.70
Controlled By:  ReplicationController/postgresql-1
Containers:
  postgresql:
    Container ID:
cri-o://3d8b5f07792c4cc9ea2810bff9ccfd4aa57b4388eafdc4ca7202809013a03054
    Image:
image-registry.openshift-image-registry.svc:5000/openshift/postgresql@sha256:58883d18f223de
d6f18ae6c144890c7c0e61729f402689ca9d0524bae467039d
    Image ID:
image-registry.openshift-image-registry.svc:5000/openshift/postgresql@sha256:58883d18f223de
d6f18ae6c144890c7c0e61729f402689ca9d0524bae467039d
    Port:           5432/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 05 Jul 2024 16:14:33 +0100
    Ready:          True
    Restart Count:  0
    Limits:
      memory:  512Mi
    Requests:
      memory:   512Mi
```

```
    Liveness:   exec [/usr/libexec/check-container --live] delay=120s timeout=10s
period=10s #success=1 #failure=3
    Readiness:  exec [/usr/libexec/check-container] delay=5s timeout=1s period=10s
#success=1 #failure=3
    Environment:
      POSTGRESQL_USER:      <set to the key 'database-user' in secret
'django-psql-example'>      Optional: false
      POSTGRESQL_PASSWORD:  <set to the key 'database-password' in secret
'django-psql-example'>  Optional: false
      POSTGRESQL_DATABASE:  default
    Mounts:
      /var/lib/pgsql/data from data (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-kkbdm (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  data:
    Type:       EmptyDir (a temporary directory that shares a pod's lifetime)
    Medium:
    SizeLimit:  <unset>
  kube-api-access-kkbdm:
    Type:                     Projected (a volume that contains injected data from multiple
sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
    ConfigMapName:           openshift-service-ca.crt
    ConfigMapOptional:       <nil>
QoS Class:                   Burstable
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/memory-pressure:NoSchedule op=Exists
                             node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason          Age   From               Message
  ----    ------          ----  ----               -------
  Normal  Scheduled       16m   default-scheduler  Successfully assigned
ysledbooks/postgresql-1-fj4c6 to p1296-master.p1296.cecc.ihost.com
  Normal  AddedInterface  16m   multus             Add eth0 [10.128.1.70/23] from
ovn-kubernetes
  Normal  Pulling         16m   kubelet            Pulling image
"image-registry.openshift-image-registry.svc:5000/openshift/postgresql@sha256:58883d18f223d
ed6f18ae6c144890c7c0e61729f402689ca9d0524bae467039d"
  Normal  Pulled          15m   kubelet            Successfully pulled image
"image-registry.openshift-image-registry.svc:5000/openshift/postgresql@sha256:58883d18f223d
ed6f18ae6c144890c7c0e61729f402689ca9d0524bae467039d" in 14.735953703s (14.735970582s
including waiting)
  Normal  Created         15m   kubelet            Created container postgresql
  Normal  Started         15m   kubelet            Started container postgresql
```

c. Check deployment configurations (**oc get deployments, oc describe deployment <deployment_name>**), as shown in Example 11-5.

Example 11-5 Check deployment configurations

```
$ oc get deployments
```

```
NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
ibm-spectrum-scale-csi-attacher      0/2     0            0           22m
ibm-spectrum-scale-csi-operator      1/1     1            1           25m
ibm-spectrum-scale-csi-provisioner   0/1     0            0           22m
ibm-spectrum-scale-csi-resizer       0/1     0            0           22m
ibm-spectrum-scale-csi-snapshotter   0/1     0            0           22m

$ oc describe deployment  ibm-spectrum-scale-csi-attache
Name:                   ibm-spectrum-scale-csi-attacher
Namespace:              ysledbooks
CreationTimestamp:      Fri, 05 Jul 2024 16:08:46 +0100
Labels:                 app.kubernetes.io/instance=ibm-spectrum-scale-csi-operator
                        app.kubernetes.io/managed-by=ibm-spectrum-scale-csi-operator
                        app.kubernetes.io/name=ibm-spectrum-scale-csi-operator
                        product=ibm-spectrum-scale-csi
                        release=ibm-spectrum-scale-csi-operator
Annotations:            deployment.kubernetes.io/revision: 1
                        productID: ibm-spectrum-scale-csi-operator
                        productName: IBM Spectrum Scale CSI Operator
                        productVersion: 2.6.0
Selector:               app=ibm-spectrum-scale-csi-attacher,product=ibm-spectrum-scale-csi
Replicas:               2 desired | 0 updated | 0 total | 0 available | 2 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  50% max unavailable, 25% max surge
Pod Template:
  Labels:           app=ibm-spectrum-scale-csi-attacher
                    app.kubernetes.io/instance=ibm-spectrum-scale-csi-operator
                    app.kubernetes.io/managed-by=ibm-spectrum-scale-csi-operator
                    app.kubernetes.io/name=ibm-spectrum-scale-csi-operator
                    product=ibm-spectrum-scale-csi
                    release=ibm-spectrum-scale-csi-operator
  Annotations:      productID: ibm-spectrum-scale-csi-operator
                    productName: IBM Spectrum Scale CSI Operator
                    productVersion: 2.6.0
  Service Account:  ibm-spectrum-scale-csi-attacher
  Containers:
   ibm-spectrum-scale-csi-attacher:
    Image:
us.gcr.io/k8s-artifacts-prod/sig-storage/csi-attacher@sha256:8b9c313c05f54fb04f8d430896f5f5
904b6cb157df261501b29adc04d2b2dc7b
    Port:       8080/TCP
    Host Port:  0/TCP
    Args:
      --v=5
      --csi-address=$(ADDRESS)
      --resync=10m
      --timeout=2m
      --leader-election=true
      --leader-election-lease-duration=$(LEADER_ELECTION_LEASE_DURATION)
      --leader-election-renew-deadline=$(LEADER_ELECTION_RENEW_DEADLINE)
      --leader-election-retry-period=$(LEADER_ELECTION_RETRY_PERIOD)
      --http-endpoint=:8080
    Liveness:  http-get http://:http-endpoint/healthz/leader-election delay=10s timeout=10s
period=20s #success=1 #failure=1
    Environment:
      ADDRESS:
/var/lib/kubelet/plugins/spectrumscale.csi.ibm.com/csi.sock
      LEADER_ELECTION_LEASE_DURATION:  137s
      LEADER_ELECTION_RENEW_DEADLINE:  107s
```

```
      LEADER_ELECTION_RETRY_PERIOD:    26s
    Mounts:
      /var/lib/kubelet/plugins/spectrumscale.csi.ibm.com from socket-dir (rw)
  Volumes:
   socket-dir:
    Type:          HostPath (bare host directory volume)
    Path:          /var/lib/kubelet/plugins/spectrumscale.csi.ibm.com
    HostPathType:  DirectoryOrCreate
Conditions:
  Type             Status  Reason
  ----             ------  ------
  Available        False   MinimumReplicasUnavailable
  ReplicaFailure   True    FailedCreate
  Progressing      False   ProgressDeadlineExceeded
OldReplicaSets:    <none>
NewReplicaSet:     ibm-spectrum-scale-csi-attacher-5dcb4978b4 (0/2 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  30m   deployment-controller  Scaled up replica set
ibm-spectrum-scale-csi-attacher-5dcb4978b4 to 2
```

Persistent Storage Issues

To look for persistent storage issues:

a. Ensure storage classes are available and properly configured (**`oc get storageclass`**), as shown in Example 11-6.

Example 11-6 Check Storage classes

```
$ oc get storageclass
NAME                                 PROVISIONER                    RECLAIMPOLICY
VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
nfs-storage-provisioner (default)   nfs-storage                    Delete
Immediate              false                  43h
portworx-class                       kubernetes.io/portworx-volume   Delete
WaitForFirstConsumer   true                   10s
spectrum-scale-ysl                   spectrumscale.csi.ibm.com       Delete
WaitForFirstConsumer   true                   58s
```

b. Check PV (PersistentVolume) and PVC (PersistentVolumeClaim) statuses (`oc get pv`, `oc get pvc`), as shown in Example 11-7.

Example 11-7 Check PV and PVC statuses

```
$ oc get pv
NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE
example 5Gi RWO Retain Available slow 2m46s
pvc-5ec50571-e938-4bca-be04-16d6485e942d 20Gi RWX Delete Bound
openshift-image-registry/registry-pvc nfs-storage-provisioner 43h
pvc-6972b773-7f85-4c9b-bb70-4f8e64875aa8 1Gi RWO Delete Bound
ysledbooks/sso-postgresql-claim nfs-storage-provisioner 25m
pvc-71f9d4d1-0afd-4834-a130-86e04b732b19 1Gi RWO Delete Bound ysledbooks/jenkins
nfs-storage-provisioner 29m
pvc-934cb2ba-4a1d-4477-8873-88312116e4d0 1Gi RWO Delete Bound ysledbooks/mariadb
nfs-storage-provisioner 34m
pvc-d9841f68-4856-4abe-bb90-bb1a0b0296b3 1Gi RWO Delete Bound ysledbooks/postgresql
nfs-storage-provisioner 27m

$ oc get pvc
```

```
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
jenkins Bound pvc-71f9d4d1-0afd-4834-a130-86e04b732b19 1GiRWO nfs-storage-provisioner 29m
mariadb Bound pvc-934cb2ba-4a1d-4477-8873-88312116e4d0 1Gi RWO
nfs-storage-provisioner 35m
postgresql Bound pvc-d9841f68-4856-4abe-bb90-bb1a0b0296b3 1Gi RWO nfs-storage-provisioner
27m
sso-postgresql-claim Bound pvc-6972b773-7f85-4c9b-bb70-4f8e64875aa8 1Gi       RWO
nfs-storage-provisioner 25m
```

c. Verify access modes and storage capacities.

Network Configuration

For network configuration issues:

a. Ensure network policies (**oc get networkpolicy**) are correctly configured, as shown in Example 11-8.

Example 11-8 Check network policies

```
$ oc get networkpolicy
NAME                  POD-SELECTOR         AGE
ysl-policy-preprod    prepared=s-preprod   67s
ysl-policy-prod       prod=s-prod          31s
ysl-policy-training   train=s-train        8s
```

b. Check pod networking (**oc get pods --all-namespaces -o wide**) for IP allocation and connectivity issues, as shown in Example 11-9.

Example 11-9 Check pod networking

```
$ oc get pods --all-namespaces -o wide
NAMESPACE NAME READY   STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES
nfs-provisioner nfs-client-provisioner-57476c4b77-zm8zx 1/1 Running 13 (20h ago) 43h
10.128.0.117 p1296-master.p1296.cecc.ihost.com   <none> <none>
openshift-apiserver-operator openshift-apiserver-operator-55fdbf86f7-sp9j4 1/1     Running
1 (44h ago) 44h 10.128.0.27 p1296-master.p1296.cecc.ihost.com <none> <none>
openshift-apiserver apiserver-ff76b5b8b-v6r4c 2/2 Running 0 43h 10.128.0.133
p1296-master.p1296.cecc.ihost.com <none> <none>
openshift-authentication-operator authentication-operator-6f6b668dcb-2cpbs 1/1 Running 1
(44h ago) 44h 10.128.0.23 p1296-master.p1296.cecc.ihost.com <none> <none>

(...)

powervm-rmc powervm-rmc-ndt9z 1/1 Running 0 43h 129.40.94.251
p1296-master.p1296.cecc.ihost.com <none> <none>
ysledbooks django-psql-example-1-2j2tt 1/1 Running 2 (82s ago) 32m 10.128.1.75
p1296-master.p1296.cecc.ihost.com   <none> <none>
ysledbooks django-psql-example-1-build 0/1 Completed 0 34m 10.128.1.68
p1296-master.p1296.cecc.ihost.com   <none> <none>
ysledbooks django-psql-example-1-deploy 0/1 Completed 0 32m 10.128.1.74
p1296-master.p1296.cecc.ihost.com   <none> <none>
ysledbooks eap-xp4-basic-app-1-75vgd 1/1 Running 0 33m 10.128.1.73
p1296-master.p1296.cecc.ihost.com <none> <none>
ysledbooks eap-xp4-basic-app-1-deploy 0/1 Completed 0 33m 10.128.1.72
p1296-master.p1296.cecc.ihost.com   <none> <none>
ysledbooks eap-xp4-basic-app-2-build 0/1 Completed 0 34m 10.128.1.67
p1296-master.p1296.cecc.ihost.com   <none> <none>
ysledbooks eap-xp4-basic-app-build-artifacts-1-build 0/1 Completed 0 36m 10.128.1.64
p1296-master.p1296.cecc.ihost.com <none> <none>
ysledbooks httpd-example-1-697g5 1/1 Running 0 39m 10.128.1.60
p1296-master.p1296.cecc.ihost.com <none> <none>
```

```
ysledbooks httpd-example-1-build 0/1 Completed 0 40m 10.128.1.58
p1296-master.p1296.cecc.ihost.com   <none> <none>
ysledbooks httpd-example-1-deploy 0/1 Completed 0 39m 10.128.1.59
p1296-master.p1296.cecc.ihost.com   <none> <none>
ysledbooks ibm-spectrum-scale-csi-operator-7874679448-btjtc 1/1 Running            3 (2m23s
ago) 42m 10.128.1.52 p1296-master.p1296.cecc.ihost.com <none> <none>
ysledbooks jenkins-1-deploy 0/1 Completed 0 35m
10.128.1.65p1296-master.p1296.cecc.ihost.com <none> <none>
ysledbooks jenkins-1-xf6t5 1/1 Running 0 35m 10.128.1.66 p1296-master.p1296.cecc.ihost.com
<none> <none>
ysledbooks mariadb-1-deploy 0/1 Completed 0 40m 10.128.1.55
p1296-master.p1296.cecc.ihost.com <none> <none>
ysledbooks mariadb-1-z6knb 1/1 Running 0 40m 10.128.1.56 p1296-master.p1296.cecc.ihost.com
<none> <none>
ysledbooks postgresql-1-deploy 0/1 Completed 0 34m 10.128.1.69
p1296-master.p1296.cecc.ihost.com   <none> <none>
ysledbooks postgresql-1-fj4c6 1/1 Running 0 34m 10.128.1.70
p1296-master.p1296.cecc.ihost.com   <none> <none>
```

11.2.3 Performance issues

For performance try the following recommendations:

Monitoring and Metrics

Gather metrics to isolate the issue:

a. Use monitoring tools like Prometheus (**`oc get routes -n openshift-monitoring`** to access Grafana dashboards), as shown in Example 11-10.

Example 11-10 Check monitoring

```
$ oc get routes -n openshift-monitoring
NAME HOST/PORT PATH SERVICES PORT   TERMINATION WILDCARD
alertmanager-main alertmanager-main-openshift-monitoring.apps.p1296.cecc.ihost.com
/api alertmanager-main web reencrypt/Redirect None
prometheus-k8s prometheus-k8s-openshift-monitoring.apps.p1296.cecc.ihost.com
/api prometheus-k8s web reencrypt/Redirect None
prometheus-k8s-federate
prometheus-k8s-federate-openshift-monitoring.apps.p1296.cecc.ihost.com   /federate
prometheus-k8s web reencrypt/Redirect None
thanos-querier thanos-querier-openshift-monitoring.apps.p1296.cecc.ihost.com
/api thanos-querier web reencrypt/Redirect None
```

b. Monitor cluster resource usage (**oc adm top nodes, oc adm top pods**), as shown in Example 11-11.

Example 11-11 Monitor cluster resource usage

```
$ oc adm top pods
W0705 16:18:17.470687   66341 top_pod.go:140] Using json format to get metrics. Next
release will switch to protocol-buffers, switch early by passing --use-protocol-buffers
flag
NAME                                          CPU(cores)   MEMORY(bytes)
django-psql-example-1-2j2tt                   1m           392Mi
eap-xp4-basic-app-1-75vgd                     27m          383Mi
httpd-example-1-697g5                         0m           79Mi
ibm-spectrum-scale-csi-operator-7874679448-btjtc   6m          25Mi
jenkins-1-xf6t5                               6m           717Mi
mariadb-1-z6knb                               3m           66Mi
```

```
postgresql-1-fj4c6                                3m           53Mi

$ oc adm top nodes
W0705 16:21:07.555429   66389 top_node.go:119] Using json format to get metrics. Next
release will switch to protocol-buffers, switch early by passing --use-protocol-buffers
flag
NAME                                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
p1296-master.p1296.cecc.ihost.com   1616m        10%    16510Mi         36%
```

Node and Pod Optimization

Validate node and pod optimization settings:

a. Review and adjust resource requests and limits (**oc edit pod <pod_or_deployment>**), as shown in Example 11-12.

Example 11-12 Adjust pod resource requests and limits

```
$ oc edit pod postgresql-1-fj4c6

# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Pod
metadata:
  annotations:
    k8s.ovn.org/pod-networks:
'{"default":{"ip_addresses":["10.128.1.70/23"],"mac_address":"0a:58:0a:80:01:46","gateway_i
ps":["10.128.0.1"],"routes":[{"dest":"10.128.0.0/14","nextHop":"10.128.0.1"},{"dest":"172.3
0.0.0/16","nextHop":"10.128.0.1"},{"dest":"100.64.0.0/16","nextHop":"10.128.0.1"}],"ip_addr
ess":"10.128.1.70/23","gateway_ip":"10.128.0.1"}}'
    k8s.v1.cni.cncf.io/network-status: |-
      [{
          "name": "ovn-kubernetes",
          "interface": "eth0",
          "ips": [
              "10.128.1.70"
          ],
          "mac": "0a:58:0a:80:01:46",
          "default": true,
          "dns": {}
      }]
    openshift.io/deployment-config.latest-version: "1"
    openshift.io/deployment-config.name: postgresql
    openshift.io/deployment.name: postgresql-1
    openshift.io/scc: restricted-v2
    seccomp.security.alpha.kubernetes.io/pod: runtime/default
  creationTimestamp: "2024-07-05T15:14:17Z"
  generateName: postgresql-1-
  labels:
    deployment: postgresql-1
    deploymentconfig: postgresql
    name: postgresql
  name: postgresql-1-fj4c6
  namespace: ysledbooks
  ownerReferences:
  - apiVersion: v1
    blockOwnerDeletion: true
    controller: true
```

```
    kind: ReplicationController
    name: postgresql-1
    uid: c36e87f9-2dd3-4887-8fae-52456511916b
  resourceVersion: "495765"
  uid: 51ba91df-2311-43c5-b195-835c55faa865
spec:
  containers:
  - env:
    - name: POSTGRESQL_USER
      valueFrom:
        secretKeyRef:
          key: database-user
          name: django-psql-example
    - name: POSTGRESQL_PASSWORD
      valueFrom:
        secretKeyRef:
          key: database-password
          name: django-psql-example
    - name: POSTGRESQL_DATABASE
      value: default
    image:
image-registry.openshift-image-registry.svc:5000/openshift/postgresql@sha256:58883d18f223de
d6f18ae6c144890c7c0e61729f402689ca9d0524bae467039d
    imagePullPolicy: IfNotPresent
    livenessProbe:
      exec:
        command:
        - /usr/libexec/check-container
        - --live
      failureThreshold: 3
      initialDelaySeconds: 120
      periodSeconds: 10
      successThreshold: 1
      timeoutSeconds: 10
    name: postgresql
    ports:
    - containerPort: 5432
```

b. Consider node scaling.

Application and Database Tuning

Optimize the tuning parameters for your application and database.

c. Optimize application performance by tuning JVM options, database configurations, and other relevant settings.

d. Monitor application logs (**`oc logs <pod_name>`**) for performance-related messages.

11.2.4 Authentication Issues

For authentication issues, follow these steps for troubleshooting:

Check Identity Providers

Check your identity provider configuration:

e. Verify the configuration of identity providers such as LDAP, HTPasswd, OAuth, and others.

f. Ensure the IDP configuration matches the actual identity provider settings (server URL, credentials, and other relevant details).

Review Authentication Logs

Review the authentication logs:

g. Access authentication logs (**`oc get pods -n openshift-authentication`** and **`oc logs <Name> -n openshift-authentication`**) to identify errors or warnings related to authentication, as shown in Example 11-13.

Example 11-13 Access authentication logs

```
$ oc get pods -n openshift-authentication
NAME                               READY   STATUS    RESTARTS   AGE
oauth-openshift-756b7f6b9f-b5tb2   1/1     Running   0          44h

$ oc logs oauth-openshift-756b7f6b9f-b5tb2 -n openshift-authentication
Copying system trust bundle
I0703 19:58:44.394973       1 dynamic_serving_content.go:113] "Loaded a new cert/key pair"
name="serving-cert::/var/config/system/secrets/v4-0-config-system-serving-cert/tls.crt::/va
r/config/system/secrets/v4-0-config-system-serving-cert/tls.key"
I0703 19:58:44.395509       1 dynamic_serving_content.go:113] "Loaded a new cert/key pair"
name="sni-serving-cert::/var/config/system/secrets/v4-0-config-system-router-certs/apps.p12
96.cecc.ihost.com::/var/config/system/secrets/v4-0-config-system-router-certs/apps.p1296.ce
cc.ihost.com"
I0703 19:58:44.812527       1 audit.go:350] Using audit backend: ignoreErrors<log>
I0703 19:58:44.824793       1 requestheader_controller.go:244] Loaded a new request header
values for RequestHeaderAuthRequestController
I0703 19:58:44.829628       1 maxinflight.go:140] "Initialized nonMutatingChan" len=400
I0703 19:58:44.829644       1 maxinflight.go:146] "Initialized mutatingChan" len=200
I0703 19:58:44.829660       1 maxinflight.go:117] "Set denominator for readonly requests"
limit=400
I0703 19:58:44.829668       1 maxinflight.go:121] "Set denominator for mutating requests"
limit=200
I0703 19:58:44.834915       1 secure_serving.go:57] Forcing use of http/1.1 only
I0703 19:58:44.834936       1 genericapiserver.go:490] MuxAndDiscoveryComplete has all
endpoints registered and discovery information is complete
I0703 19:58:44.837083       1 requestheader_controller.go:169] Starting
RequestHeaderAuthRequestController
I0703 19:58:44.837089       1 configmap_cafile_content.go:202] "Starting controller"
name="client-ca::kube-system::extension-apiserver-authentication::client-ca-file"
I0703 19:58:44.837133       1 shared_informer.go:273] Waiting for caches to sync for
client-ca::kube-system::extension-apiserver-authentication::client-ca-file
I0703 19:58:44.837107       1 configmap_cafile_content.go:202] "Starting controller"
name="client-ca::kube-system::extension-apiserver-authentication::requestheader-client-ca-f
ile"
I0703 19:58:44.837178       1 shared_informer.go:273] Waiting for caches to sync for
client-ca::kube-system::extension-apiserver-authentication::requestheader-client-ca-file
I0703 19:58:44.837133       1 shared_informer.go:273] Waiting for caches to sync for
RequestHeaderAuthRequestController
I0703 19:58:44.837925       1 dynamic_serving_content.go:132] "Starting controller"
name="serving-cert::/var/config/system/secrets/v4-0-config-system-serving-cert/tls.crt::/va
r/config/system/secrets/v4-0-config-system-serving-cert/tls.key"
I0703 19:58:44.840367       1 tlsconfig.go:200] "Loaded serving cert"
certName="serving-cert::/var/config/system/secrets/v4-0-config-system-serving-cert/tls.crt:
:/var/config/system/secrets/v4-0-config-system-serving-cert/tls.key"
certDetail="\"oauth-openshift.openshift-authentication.svc\" [serving]
validServingFor=[oauth-openshift.openshift-authentication.svc,oauth-openshift.openshift-aut
hentication.svc.cluster.local] issuer=\"openshift-service-serving-signer@1720035339\"
```

```
(2024-07-03 19:35:58 +0000 UTC to 2026-07-03 19:35:59 +0000 UTC (now=2024-07-03
19:58:44.840307935 +0000 UTC))"
I0703 19:58:44.840874       1 dynamic_serving_content.go:132] "Starting controller"
name="sni-serving-cert::/var/config/system/secrets/v4-0-config-system-router-certs/apps.p12
96.cecc.ihost.com::/var/config/system/secrets/v4-0-config-system-router-certs/apps.p1296.ce
cc.ihost.com"

(...)

I0704 19:24:23.584573       1 named_certificates.go:53] "Loaded SNI cert" index=1
certName="sni-serving-cert::/var/config/system/secrets/v4-0-config-system-router-certs/apps
.p1296.cecc.ihost.com::/var/config/system/secrets/v4-0-config-system-router-certs/apps.p129
6.cecc.ihost.com" certDetail="\"*.apps.p1296.cecc.ihost.com\" [serving]
validServingFor=[*.apps.p1296.cecc.ihost.com] issuer=\"ingress-operator@1720035359\"
(2024-07-03 19:35:59 +0000 UTC to 2026-07-03 19:36:00 +0000 UTC (now=2024-07-04
19:24:23.584535393 +0000 UTC))"
I0704 19:24:23.585561       1 named_certificates.go:53] "Loaded SNI cert" index=0
certName="self-signed loopback" certDetail="\"apiserver-loopback-client@1720036724\"
[serving] validServingFor=[apiserver-loopback-client]
issuer=\"apiserver-loopback-client-ca@1720036724\" (2024-07-03 18:58:44 +0000 UTC to
2025-07-03 18:58:44 +0000 UTC (now=2024-07-04 19:24:23.585528386 +0000 UTC))"
E0705 15:14:33.739096       1 osinserver.go:103] osin: error=invalid_request,
internal_error=&errors.errorString{s:"Request must be POST"} access_request=request must be
POST
E0705 15:14:33.739259       1 osinserver.go:111] internal error: Request must be POST
E0705 15:14:35.275965       1 osinserver.go:103] osin: error=invalid_request,
internal_error=&errors.errorString{s:"Request must be POST"} access_request=request must be
POST
E0705 15:14:35.275989       1 osinserver.go:111] internal error: Request must be POST
E0705 15:14:36.886089       1 osinserver.go:103] osin: error=invalid_request,
internal_error=&errors.errorString{s:"Request must be POST"} access_request=request must be
POST
E0705 15:14:36.886166       1 osinserver.go:111] internal error: Request must be POST
E0705 15:14:37.795196       1 osinserver.go:103] osin: error=invalid_request,
internal_error=&errors.errorString{s:"Request must be POST"} access_request=request must be
POST
E0705 15:14:37.795251       1 osinserver.go:111] internal error: Request must be POST
```

h. Look for messages indicating failed login attempts or issues with token generation (**oc describe secret oauth-openshift -n openshift-authentication**), as shown in Example 11-14.

Example 11-14 Check messages for failed login attempts with token generation

```
$ oc describe secret oauth-openshift -n openshift-authentication
Name:         oauth-openshift-dockercfg-vsfxx
Namespace:    openshift-authentication
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: oauth-openshift
              kubernetes.io/service-account.uid: 5a99f846-aa16-48be-84c9-83dd488a52b5
              openshift.io/token-secret.name: oauth-openshift-token-6pwhr
              openshift.io/token-secret.value:

eyJhbGciOiJSUzI1NiIsImtpZCI6Imh5VFhyLUpzUV9QMS1kQWIwNlBoMllvWGRORzBHREYOb1FRZFM5OE54S1EifQ.
eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW5OIiw...

Type:  kubernetes.io/dockercfg

Data
====
```

```
.dockercfg:  9774 bytes

Name:         oauth-openshift-token-6pwhr
Namespace:    openshift-authentication
Labels:       <none>
Annotations:  kubernetes.io/created-by: openshift.io/create-dockercfg-secrets
              kubernetes.io/service-account.name: oauth-openshift
              kubernetes.io/service-account.uid: 5a99f846-aa16-48be-84c9-83dd488a52b5

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:          7234 bytes
namespace:       24 bytes
service-ca.crt:  8447 bytes
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6Imh5VFhyLUpzUV9QMS1kQWIwNlBoMllvWGRORzBHREYOb1FRZFM5OE54S1EifQ.
eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW5OIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1
lc3BhY2UiOiJvcGVuc2hpZnQtYXVOaGVudGljYXRpb24iLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW5OL3NlY3
JldC5uYW1lIjoib2F1dGgtb3BlbnNoaWZOLXRva2VuLTZwd2hyIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3Vud
C9zZXJ2aWNlLWFjY291bnQubmFtZSI6Im9hdXRoLW9wZW5zaGlmdCIsImt1YmVybmVOZXMuaW8vc2VydmljZWFjY291
bnQvc2VydmljZS1hY2NvdW5OLnVpZCI6IjVhOTlmODQ2LWFhMTYtNDhiZSO4NGM5LTgzZGQOODhhNTJiNSIsInN1YiI
6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpvcGVuc2hpZnQtYXVOaGVudGljYXRpb246b2F1dGgtb3BlbnNoaWZOInO.Eq
lF4jLjadVOea41U2NG6tVoo8CvfEB3Fa_GEsD_UFFRALLefpZbt7CO7GSIvaYjnlmcMmvloSd4LGU6L7PtvLusVjv3k
La7QWx9YuHeTtc8RoxFjyM9WYLUGm_yeaidS7GSZNMaAjVBHDD6ftP4StaAoUqdh-z47O3ShtEES1vAVv67rPMvAdWK
w2sOPT-oU2mWuNrYSXSOkS7I7Iwgc-6pWPhxfghMiXA-vB496Dh4aiUJRucqH9Yv6snRsrrJ6ntgdCqmAKjxIiBamc-
9jZHuwS1BMGvbJhzbeqAtfyKTt5tmF9Y1mViPGdT_-oD_ZwzK432iAuiWrk7AkeOHew
```

Test Authentication Flow

Check the authentication flow:

a. Use oc login with different user credentials to test authentication.
b. Check if users are correctly mapped to OpenShift roles.

Check cluster-wide authentication configuration

Validate your cluster authentication configuration:

a. Review OAuth server configuration (**`oc get oauth`** and **`oc describe oauth`**) for any misconfiguration or inconsistencies, as shown in Example 11-15.

Example 11-15 Review OAuth server configuration

```
$ oc get oauth
NAME      AGE
cluster   44h
Macs-MacBook-Pro-321:okd root# ./oc describe oauth
Name:         cluster
Namespace:
Labels:       <none>
Annotations:  include.release.openshift.io/ibm-cloud-managed: true
              include.release.openshift.io/self-managed-high-availability: true
              include.release.openshift.io/single-node-developer: true
              release.openshift.io/create-only: true
API Version:  config.openshift.io/v1
Kind:         OAuth
Metadata:
  Creation Timestamp:  2024-07-03T19:24:07Z
  Generation:          2
```

```
  Managed Fields:
    API Version:  config.openshift.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          .:
          f:include.release.openshift.io/ibm-cloud-managed:
          f:include.release.openshift.io/self-managed-high-availability:
          f:include.release.openshift.io/single-node-developer:
          f:release.openshift.io/create-only:
        f:ownerReferences:
          .:
          k:{"uid":"3e66d26d-85ed-46ed-9d45-1b915e68a8e0"}:
      f:spec:
    Manager:      cluster-version-operator
    Operation:    Update
    Time:         2024-07-03T19:24:07Z
    API Version:  config.openshift.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          f:kubectl.kubernetes.io/last-applied-configuration:
      f:spec:
        f:identityProviders:
    Manager:    kubectl-client-side-apply
    Operation:  Update
    Time:       2024-07-03T19:57:43Z
  Owner References:
    API Version:     config.openshift.io/v1
    Kind:            ClusterVersion
    Name:            version
    UID:             3e66d26d-85ed-46ed-9d45-1b915e68a8e0
  Resource Version:  22378
  UID:               5b15cc9d-7220-4f28-9b83-4803a00c4e72
Spec:
  Identity Providers:
    Htpasswd:
      File Data:
        Name:        htpass-secret
    Mapping Method:  claim
    Name:            htpasswd
    Type:            HTPasswd
Events:              <none>
```

11.2.5 Authorization Issues

For authorization issues follow these suggested steps:

1. Verify Role-Based Access Control (RBAC):
   a. Check roles and role bindings (**`oc get roles, oc get rolebindings`**) to ensure users or groups have appropriate permissions, as shown in Example 11-16.

Example 11-16 Check roles and role bindings

```
$ oc get rolebindings
NAME ROLE AGE
```

```
admin ClusterRole/admin 85m
ibm-spectrum-scale-csi-operator.v2.6.0 Role/ibm-spectrum-scale-csi-operator.v2.6.0   52m
jenkins_edit ClusterRole/edit 44m
system:deployers ClusterRole/system:deployer 85m
system:image-builders ClusterRole/system:image-builder 85m
system:image-pullers ClusterRole/system:image-puller 85m
```

b. Verify cluster roles and cluster role bindings (**`oc get clusterroles, oc get clusterrolebindings`**) for global permissions, as shown in Example 11-17.

Example 11-17 Check cluster role bindings

```
$ oc get clusterroles
NAME CREATED AT
3scale-community-operator.v0.11.0-3scale-operator-68b7f98f66 2024-07-05T14:33:11Z
3scale-community-operator.v0.11.0-6d594c4565 2024-07-05T14:33:12Z
admin 2024-07-03T19:23:00Z
aggregate-olm-edit 2024-07-03T19:24:18Z
aggregate-olm-view 2024-07-03T19:24:19Z
alert-routing-edit 2024-07-03T19:37:33Z
alertmanager-main 2024-07-03T19:47:45Z
backups.postgresql.k8s.enterprisedb.io-v1-admin 2024-07-05T14:41:28Z
backups.postgresql.k8s.enterprisedb.io-v1-crdview 2024-07-05T14:41:28Z
backups.postgresql.k8s.enterprisedb.io-v1-edit 2024-07-05T14:41:28Z
backups.postgresql.k8s.enterprisedb.io-v1-view 2024-07-05T14:41:28Z
basic-user 2024-07-03T19:37:56Z
cloud-controller-manager 2024-07-03T19:24:45Z
cloud-credential-operator-role 2024-07-03T19:24:00Z
cloud-native-postgresql.v1.23.2-75b5684775 2024-07-05T14:41:12Z
cloud-node-manager 2024-07-03T19:24:45Z
cluster-admin 2024-07-03T19:23:00Z
cluster-autoscaler 2024-07-03T19:24:39Z
cluster-autoscaler-operator 2024-07-03T19:23:56Z
cluster-autoscaler-operator:cluster-reader 2024-07-03T19:37:21Z
cluster-baremetal-operator 2024-07-03T19:24:44Z
cluster-debugger 2024-07-03T19:37:56Z
cluster-image-registry-operator 2024-07-03T19:24:13Z
cluster-monitoring-operator 2024-07-03T19:24:10Z
cluster-monitoring-operator-namespaced 2024-07-03T19:24:09Z
cluster-monitoring-view 2024-07-03T19:37:33Z
cluster-node-tuning-operator 2024-07-03T19:24:10Z
cluster-node-tuning:tuned 2024-07-03T19:24:13Z
cluster-reader 2024-07-03T19:37:56Z
clusterimagecatalogs.postgresql.k8s.enterprisedb.io-v1-view 2024-07-05T14:41:28Z
clusters.postgresql.k8s.enterprisedb.io-v1-admin 2024-07-05T14:41:28Z
clusters.postgresql.k8s.enterprisedb.io-v1-crdview 2024-07-05T14:41:28Z
clusters.postgresql.k8s.enterprisedb.io-v1-edit 2024-07-05T14:41:28Z
clusters.postgresql.k8s.enterprisedb.io-v1-view 2024-07-05T14:41:28Z

(...)

ysledbooks-lv48z-admin 2024-07-05T15:05:33Z
ysledbooks-lv48z-edit 2024-07-05T15:05:33Z
ysledbooks-lv48z-view 2024-07-05T15:05:33Z
```

Review Service Accounts and Tokens

Review service accounts and validate tokens:

a. Ensure service accounts and service account tokens are properly configured (**`oc get sa, oc describe sa <service_account>`**), as shown in Example 11-18 on page 174.

Example 11-18 Check service accounts

```
$ oc get sa
NAME                                SECRETS   AGE
builder                             1         87m
default                             1         87m
deployer                            1         87m
ibm-spectrum-scale-csi-attacher     1         50m
ibm-spectrum-scale-csi-node         1         50m
ibm-spectrum-scale-csi-operator     1         53m
ibm-spectrum-scale-csi-provisioner  1         50m
ibm-spectrum-scale-csi-resizer      1         50m
ibm-spectrum-scale-csi-snapshotter  1         50m
jenkins                             1         46m

$ oc describe sa jenkins
Name:                jenkins
Namespace:           ysledbooks
Labels:              app=jenkins-persistent
                     template=jenkins-persistent-template

template.openshift.io/template-instance-owner=bf8ca4b4-1ab7-40cc-a4ef-53e63ec4412d
Annotations:         serviceaccounts.openshift.io/oauth-redirectreference.jenkins:

{"kind":"OAuthRedirectReference","apiVersion":"v1","reference":{"kind":"Route","name":"jenk
ins"}}
Image pull secrets:  jenkins-dockercfg-lnwxn
Mountable secrets:   jenkins-dockercfg-lnwxn
Tokens:              jenkins-token-rpst5
Events:              <none>
```

b. Check if the service account has the necessary permissions (**oc describe secret <service_account_token>**).

Audit Logs for Authorization

If not already enabled, ensure that audit logging is configured in your OpenShift cluster. You typically configure this through the API server configuration.

a. Use the **oc edit** command to modify the API server configuration and define an audit policy. This policy specifies what events should be audited, including authorization-related events, as shown in Example 11-19.

Example 11-19 Edit apiserver cluster

```
$ oc edit apiserver cluster
Edit cancelled, no changes made.
Macs-MacBook-Pro-321:okd root#

# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: config.openshift.io/v1
kind: APIServer
metadata:
  annotations:
    include.release.openshift.io/ibm-cloud-managed: "true"
    include.release.openshift.io/self-managed-high-availability: "true"
    include.release.openshift.io/single-node-developer: "true"
    oauth-apiserver.openshift.io/secure-token-storage: "true"
```

```
    release.openshift.io/create-only: "true"
  creationTimestamp: "2024-07-03T19:23:51Z"
  generation: 1
  name: cluster
  ownerReferences:
  - apiVersion: config.openshift.io/v1
    kind: ClusterVersion
    name: version
    uid: 3e66d26d-85ed-46ed-9d45-1b915e68a8e0
  resourceVersion: "769"
  uid: 3d869b7a-d3f2-4d46-91b0-d70a992b6712
spec:
  audit:
    profile: Default
~
```

b. Save the changes to the configuration file. OpenShift will automatically apply the updated audit policy.

c. Once audit logging is enabled and configured, you can access and review the audit logs to monitor authorization activities, as shown in Example 11-20.

Example 11-20 Access audit logs

```
$ oc get pods -n openshift-logging
NAME                                           READY   STATUS    RESTARTS   AGE
cluster-logging-operator-7f9f5d96bb-mthwq   1/1     Running   0          41s

$ oc logs -n openshift-logging cluster-logging-operator-7f9f5d96bb-mthwq --since=1h
{"_ts":"2024-07-05T16:30:55.170670998Z","_level":"0","_component":"cluster-logging-operator
","_message":"starting up...","go_arch":"ppc64le","go_os":"linux","go_version":"go1.21.9
(Red Hat 1.21.9-1.el9_4)","operator_version":"5.9.0"}
{"_ts":"2024-07-05T16:30:55.186310278Z","_level":"0","_component":"cluster-logging-operator
","_message":"migrating resources provided by the manifest"}
{"_ts":"2024-07-05T16:30:55.189417649Z","_level":"0","_component":"cluster-logging-operator
","_message":"Registering Components."}
{"_ts":"2024-07-05T16:30:55.199554206Z","_level":"0","_component":"cluster-logging-operator
","_message":"Starting the Cmd."}
{"_ts":"2024-07-05T16:30:55.199747089Z","_level":"0","_component":"cluster-logging-operator
_controller-runtime_metrics","_message":"Starting metrics server"}
{"_ts":"2024-07-05T16:30:55.200211169Z","_level":"0","_component":"cluster-logging-operator
_controller-runtime_metrics","_message":"Serving metrics
server","bindAddress":":8686","secure":false}
{"_ts":"2024-07-05T16:30:55.300422794Z","_level":"0","_component":"cluster-logging-operator
","_message":"Starting
EventSource","controller":"clusterlogging","controllerGroup":"logging.openshift.io","contro
llerKind":"ClusterLogging","source":"kind source: *v1.ClusterLogging"}
{"_ts":"2024-07-05T16:30:55.300452417Z","_level":"0","_component":"cluster-logging-operator
","_message":"Starting
EventSource","controller":"clusterlogforwarder","controllerGroup":"logging.openshift.io","c
ontrollerKind":"ClusterLogForwarder","source":"kind source: *v1.ClusterLogForwarder"}

(...)

{"_ts":"2024-07-05T16:30:55.663753491Z","_level":"0","_component":"cluster-logging-operator
","_message":"Starting
workers","controller":"clusterlogging","controllerGroup":"logging.openshift.io","controller
Kind":"ClusterLogging","worker count":1}
{"_ts":"2024-07-05T16:30:55.663894815Z","_level":"0","_component":"cluster-logging-operator
","_message":"Starting
workers","controller":"logfilemetricexporter","controllerGroup":"logging.openshift.io","con
trollerKind":"LogFileMetricExporter","worker count":1}
```

11.2.6 General Best Practices

These are some general best practices for maintaining your environment:

1. Backup and Restore:
   a. Implement regular backups (**`oc create backup`**).
   b. Practice disaster recovery scenarios (**`oc create restore`**).
2. Stay Updated:
   c. Keep OpenShift and its components up to date (**`oc adm upgrade`**).
   d. Monitor Red Hat advisories for patches and updates.

11.3 Accessing IBM and Red Hat support resources

There are many places where you can get support for OpenShift. For comprehensive resources on IBM and Red Hat support for OpenShift, explore the following:

IBM Cloud Documentation

IBM provides extensive documentation on OpenShift, covering various aspects such as installation, configuration, usage, and troubleshooting. For more information, see IBM Cloud documentation.

Red Hat Documentation

Red Hat offers detailed documentation on OpenShift, including user guides, administration guides, and troubleshooting guides. For more information, see the Red Hat Customer Portal.

Red Hat Knowledge Base

The Red Hat Knowledge Base contains a wealth of articles, solutions, and best practices for working with OpenShift. For more information, see the Red Hat Knowledge Base.

Red Hat support

Red Hat offers support services for OpenShift, including technical support and consulting. For more information, see Red Hat Support.

IBM support

If you're using OpenShift on IBM Cloud, you can also leverage IBM support services. IBM offers various support options, including online support, phone support, and technical account management.

Community forums and discussion groups

Various online forums and discussion groups allow users to share experiences, ask questions, and provide help related to Red Hat OpenShift. Examples include the Red Hat Customer Portal forums, Stack Overflow, and the Red Hat OpenShift subreddit.

Training and certification

Both IBM and Red Hat offer training and certification programs for OpenShift. These programs cover topics from basic to advanced concepts and can help you gain expertise in deploying, managing, and troubleshooting Red Hat OpenShift environments.

By leveraging these resources, you can access the support and guidance you need to effectively deploy, manage, and troubleshoot Red Hat OpenShift environments in your organization.

12

Case Studies

In this chapter, we provide a path to help you consider how to employ Multi-Architecture Compute to solve problems in your environment.

12.1 Building a multi-architecture image

Building an OpenShift multi-architecture image involves creating a Docker image that can run on different CPU architectures supported by OpenShift, such as x86_64, ARM, IBM Power, and others. Here is a step-by-step guide to building a multi architecture image for OpenShift:

Prerequisites

Ensure that you meet the following prerequisites:

- Have an application Dockerfile, which defines the instructions to build your application or service.
- The OpenShift CLI (oc) should be installed and configured to interact with your OpenShift cluster.
- Have access to Docker Hub or another container registry where you will push your multi-architecture image.

Steps to build a multi architecture Image

Follow these steps to build your multi architecture image:

- Create a Dockerfile for your application. This file contains instructions to build your application inside a Docker container, as shown in Example 12-1.

Example 12-1 Example of Dockerfile

```
# Use a base image that supports multiple architectures
FROM --platform=${BUILDPLATFORM:-linux/amd64} golang:1.17 AS builder
WORKDIR /app
# Copy the local package files to the container's workspace.
COPY . .
# Build the application
RUN go build -o myapp
# Final image
FROM --platform=${TARGETPLATFORM:-linux/amd64} alpine:latest
WORKDIR /root/

# Copy the binary from the builder stage
COPY --from=builder /app/myapp .
# Set the binary as executable
RUN chmod +x myapp
# Run the application
CMD ["./myapp"]
```

- Use Docker Buildx to build your multi architecture image. Docker Buildx is a CLI plug-in that extends the Docker CLI to enable building and pushing images to multiple platforms. Replace your `username/myapp:latest` with your Docker Hub username and image name, as shown in Example 12-2:

Example 12-2 Build Multiachitecture image

```
# Initialize Docker Buildx (if not already initialized)
$ docker buildx create --use

# Build the multiarchitecture image
$ docker buildx build --platform linux/amd64,linux/arm64,linux/ppc64le -t
yourusername/myapp:latest .
```

- Once the image is built, push it to a container registry accessible to your OpenShift cluster, as shown in Example 12-3:

Example 12-3 Push image to Registry

```
$ docker push yourusername/myapp:latest
```

- Deploy your multi architecture image on OpenShift using the oc CLI or the OpenShift Web Console, as shown in Example 12-4:

Example 12-4 Deploy the image on OpenShift

```
$ oc create deployment myapp --image=yourusername/myapp:latest
```

12.2 Heterogeneous examples

This section provides several examples of how using a multiple architecture compute cluster can help simplify your cloud-based infrastructure.

12.2.1 Architecture migration

It offers an easy way to try out specific applications or services on a different architecture while maintaining your current environment. A gradual roll out of another architecture means that you can avoid a "big bang" approach where you might need to change everything at once. This is shown in Figure 12-1.

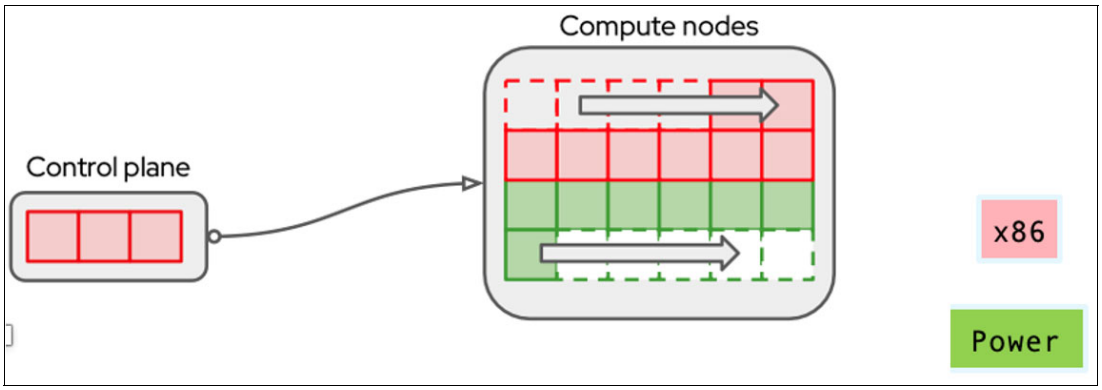


Figure 12-1 IBM Power migration diagram

12.2.2 Legacy applications

Have an application that you may not have the source code for? It is likely too expensive to perform a complete re-write of these applications. You can handle gaps in the new architecture's ecosystem as illustrated in Figure 12-2 on page 182.

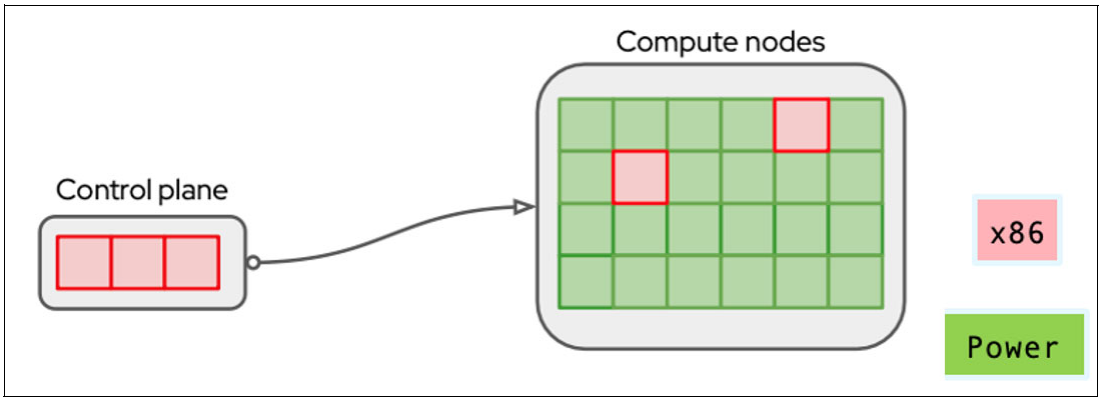


Figure 12-2 IBM Power and x86 mixed applications

12.2.3 Cost optimization

Think of using a different architecture control plane that is made up of cheaper systems. It poses an interesting case for Power systems, as shown in Figure 12-3.

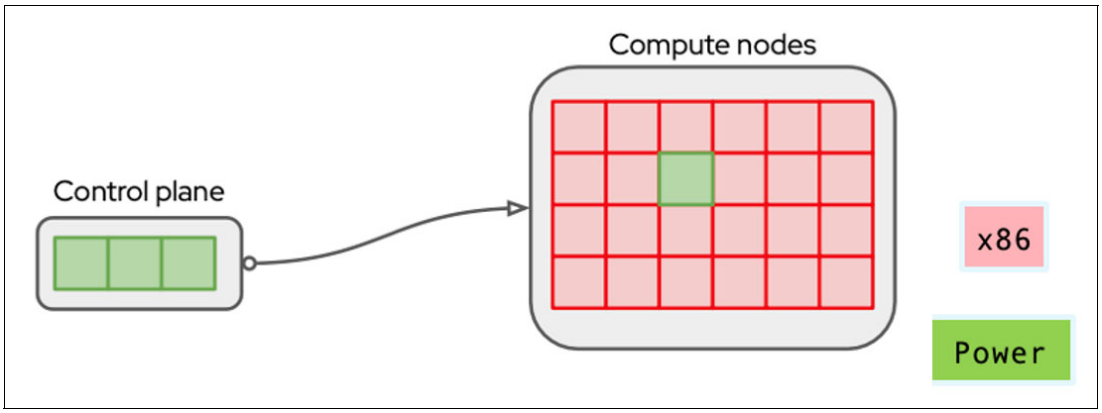


Figure 12-3 Cost optimization strategy

12.3 Success stories of Red Hat OpenShift Multi-Arch compute on IBM Power Systems

This section discusses different reasons that a client might want to use a Red Hat OpenShift multi-architecture compute cluster

12.3.1 Missing ecosystem component

If you comprehend the benefits of employing Power for your cloud-based application infrastructure and have deployed Red Hat OpenShift in Power servers, yet certain crucial services – such as a key Cloud Pak component, an ISV application or service, or an open source package – are absent from the Power ecosystem, a multi-architecture cluster comprising both Power and x86 worker nodes can serve as an ideal solution. This configuration allows you to run any components that are not available on Power on the x86 worker nodes.

Prior to multiple architecture support:

- You must run software on separate OCP clusters for each platform architecture (x86, Power, Z/LinuxOne).
- Should a single software component not be available on Power, the client must run all software on x86 architecture.
- Client is unable to exploit the value of incrementally modernizing applications with Red Hat OpenShift on Power.

With multiple architecture suppor:

- You may run ecosystem elements on x86 that are unavailable for Power in the same Red Hat OpenShift cluster
- Provides lower risk and simpler implementation by managing only one Red Hat OpenShift cluster.
- Observe faster time to value by modernizing applications and infrastructure on IBM Power and x86.

12.3.2 Best fit platform

There are some workloads that are more suited to a specific compute architecture when you consider cost, performance, sustainability, resiliency, and security. With multi-arch computing, you can choose the best fit architecture for each of the services you are providing.

Before multiple architecture support:

- One size fit all architecture is unable to efficiently run workloads that require higher performance or need a more secure and resilient architectures without the complexity of running two different clusters.
- Unable to try out specific apps on different architectures or use lower cost platforms for the control plane.
- Solutions that require mixed compute architectures require multiple separate clusters.

After:

- Clients can exploit best fit architecture for each component.
- Clients can reduce the cost and complexity of solutions that require workloads to run on multiple compute architectures.
- Enable dynamic scheduling of workloads across best fit architecture to support peak demand and SLAs.

12.3.3 DevOps across multiple architectures

The ability to have a simple and consistent DevOps environment across multiple architectures is a significant advantage and can save you time and effort as you develop software for multiple architectures.

Before multiple architecture support:

- Separate clusters are required for each architecture.
- Separate build, test, release, repositories are required.
- Not as easy to be deterministic when there are three build/test/run paths.

After:

- Clients will observe improved productivity for developers and IT operations.
- Faster iterations, which lead to faster releases and improved time to market.
- Supports deterministic properties i.e. starting from the same point and using the same process, on the same cluster, develop once and deploy many.

12.4 Hybrid cloud use cases

Hybrid cloud environments, offer versatile solutions that leverage both public and private cloud infrastructures. This integration enables organizations to achieve enhanced flexibility, scalability, and resilience in their IT operations.

12.4.1 Burst to cloud

In this case, a client wants to use the cloud as a scale-out location to run their application. During peak periods, such as seasonal sales or promotional events, the demand on the e-commerce platform's infrastructure exceeds the capacity of their on-premises IBM Power Systems running Red Hat OpenShift.

Using the built-in capabilities of OpenShift, the customer wants to set up the ability to scale to the cloud when the on-premises systems are running out of capacity. This involves:

- Dynamic scaling can be defined to configure OpenShift to utilize IBM Power Virtual Server instances in the cloud as burst capacity. This involves setting up OpenShift nodes on IBM Power Virtual Server instances, which can be provisioned and scaled up/down dynamically based on workload demands.

  OpenShift supports multiple CPU architectures, including IBM Power (ppc64le) and x86_64. This allows applications running on OpenShift to seamlessly migrate between on-premises IBM Power Systems or x86 and IBM Power Virtual Server in the cloud without modification

  There are some challenges that need to be addressed to enable a successful implementation of the burst-to-cloud environment.
- Ensuring low-latency network connectivity between on-premises data centers and IBM Power Virtual Server instances in the cloud are crucial for maintaining application performance. IBM Cloud and PowerVS provide many options for optimizing the connection between your on-premises infrastructure and the IBM Cloud resources.
- Careful planning is required to ensure data consistency and synchronization between on-premises databases and cloud-hosted applications to prevent data discrepancies or loss during burst scaling.
- Resource allocation and deallocation policies need to be defined within the cluster based on application workloads to allow the efficient management of resources (CPU, memory, storage) across hybrid cloud environments to optimize performance and cost-effectiveness.

12.4.2 DR in the cloud

This section discusses using a cloud instance as your DR target. In this case a financial institution requires a robust disaster recovery (DR) solution for their critical banking applications running on Red Hat OpenShift deployed on IBM Power Systems.

Using data replication from their on-premises Power systems to IBM Power instances in IBM Power Virtual Server the customer can:

- Establish IBM Power Virtual Server instances in the cloud as a secondary DR site for the OpenShift cluster. This involves replicating application data and configurations from on-premises IBM Power Systems to IBM Power Virtual Server.
- Implement automated failover mechanisms within OpenShift to detect and respond to primary site failures by redirecting traffic and workload execution to the cloud-based DR site.
- Test and ensure that the applications and services running on OpenShift can seamlessly failover between on-premises IBM Power Systems and IBM Power Virtual Server instances in the cloud.

Ensure that the following challenges are addressed:

- Data needs to be replicated to the DR location by implementing efficient data replication mechanisms (e.g., asynchronous replication) between on-premises and cloud environments to ensure data consistency and minimize recovery point objectives (RPO).
- Automate and streamline the failover processes to minimize the time required to activate the DR site and resume operations in the cloud, ensuring business continuity and meeting SLAs.
- Addressing any security concerns related to data protection, access control, and compliance requirements when replicating sensitive banking data to a cloud-based DR site.

12.5 Power10 MMA AI inferencing

Over the past few years, customers have been actively exploring AI capabilities using our platforms. They have deployed AI models and made them accessible through REST endpoints. Many customers have integrated these solutions into operational applications that interact with these remote REST endpoints. However, this approach comes with several challenges:

- There are complexities involved in extracting data from operational platforms during model execution, as well as overhead in transferring this data to the AI platform. This process often introduces latency when receiving predictions back from the AI platform to the operational environment.
- Monitoring and governing the execution of these models also pose significant challenges. Ensuring end-to-end availability becomes difficult when operational applications rely on external model invocation services hosted on AI platforms. Moreover, the remote invocation of models, which involves shipping data back and forth for predictions, increases the risk of breaching service level agreements.
- Furthermore, the maintenance of multiple environments dedicated to AI model execution and the need to reserve capacity, add to both the cost and complexity of IT operations.

IBM Power10 addresses these challenges through advanced hardware and software innovations. It supports the deployment of AI models trained on any cloud, IBM Power, or x86 platform, including those with different endian-ness, without requiring modifications. Power10's compute cores feature 4x Matrix Math Accelerators (MMA) Engines optimized for matrix and tensor computations, enabling applications to execute models efficiently with data located on the same hardware. This eliminates the need for external accelerators, GPUs, or additional AI platforms.

Power10 adopts the principle of "Train Anywhere, Deploy Here," streamlining the operationalization of AI. Here's how AI can be effectively operationalized using Power10:

- Initially, a model trained on a public or private cloud is registered along with its version in a "model vault," typically a VM or LPAR equipped with tools like IBM Watson® Openscale, BentoML, or Tensorflow Serving.
- Next, the model is deployed to its destination, such as a VM running alongside a database and application. These components, positioned on the left-hand side of the operational chart, can directly utilize the model.
- Transactions received by the database and application trigger the execution of the deployed model, generating predictions or classifications. These outcomes can also be stored locally, such as risk assessments for transactions or product classifications for downstream applications.
- Copies of the model's outputs (predictions or classifications) are sent to the ModelOps engine. Here, drift calculations against Ground Truth data are performed. If drift exceeds a specified threshold, this step initiates triggers for model retraining.

Through a synergy of software and hardware innovations, IBM Power10 ensures it meets the stringent performance, response time, and throughput Key Performance Indicators (KPIs) required by databases and AI-integrated applications. This integrated approach not only enhances operational efficiency but also reduces the complexity associated with maintaining multiple AI execution environments.

Figure 12-4 shows an example of a Low-Latency, High-Throughput Microservice Portal using a hybrid architecture.

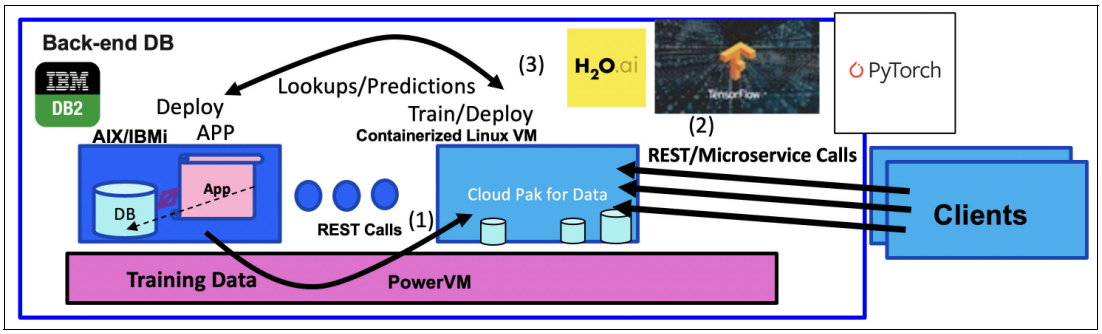


Figure 12-4 Low-Latency, High-Throughput Microservice Portal

In this scenario, the backend database resides on an AIX or IBM i LPAR, ensuring robust and reliable data management. Utilizing IBM Cloud Pak for Data (CP4D) and Red Hat OpenShift, a microservice portal is implemented to handle client requests efficiently.

Workflow Overview

The following overview demonstrates the workflow.

1. Client Interaction

   Clients initiate REST API calls to the microservice portal hosted on IBM Cloud Pak for Data running on Red Hat OpenShift.

2. Microservice Execution

   Within the IBM Cloud Pak for Data environment, microservices are designed to handle various tasks such as predicting inventory levels, forecasting future demand, and authenticating requests for fraud detection.

These microservices issue queries and lookups to the backend database located on the AIX or IBM i LPAR.

- Processing in AIX/IBM i LPAR

  The backend database, residing in the AIX or IBM i LPAR, processes the requests from IBM CP4D. It manages data storage, retrieval, and updates based on the queries received.

- Prediction and Classification

  Upon receiving requests from IBM Cloud Pak for Data, the backend database executes operations related to prediction and classification. The results of these operations, such as inventory predictions or fraud classifications, are generated within the AIX/IBM i LPAR.

- Response to Clients

  The prediction and classification responses are sent back to IBM Cloud Pak for Data where they are formatted and returned to the requesting clients via the microservice portal. Clients receive updated information and insights based on the predictions made by the system.

Technology Stack

The technology stack consists of:

- An AIX or IBM i LPAR which is hosting the backend database, ensuring reliable data management and processing.
- IBM Cloud Pak for Data running on Red Hat OpenShift which implements the microservice portal. Orchestrating requests and responses between clients and the backend database.
- The AI Stack within IBM Cloud Pak for Data for performing predictive analytics and classification tasks, enhancing decision-making capabilities.

This architecture leverages the strengths of both traditional on-premises systems (AIX/IBM i LPAR) and modern cloud-based microservices (IBM CP4D/OpenShift), ensuring low-latency, high-throughput operations for handling critical business tasks such as inventory management, demand forecasting, and fraud detection.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *Introduction to IBM PowerVM*, SG24-8535
- *IBM PowerVC Version 2.0 Introduction and Configuration*, SG24-8477
- *IBM Storage Fusion Product Guide*, REDP-5688
- *Implementing, Tuning, and Optimizing Workloads with Red Hat OpenShift on IBM Power, SG24-8537*
- *Red Hat OpenShift and IBM Cloud Paks on IBM Power Systems: Volume 1*, SG24-8459
- *Red Hat OpenShift V4.X and IBM Cloud Pak on IBM Power Systems Volume 2*, SG24-8486
- *Security Implementation with Red Hat OpenShift on IBM Power Systems*, REDP-5690

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- Creating a cluster with multi-architecture compute machines on IBM Power
- Managing your cluster with multi-architecture compute machines
- Red Hat OpenShift Documentation: Installing on IBM Power
- Red Hat OpenShift Documentation: Installing on IBM Power Virtual Server

Help from IBM

IBM Support and downloads

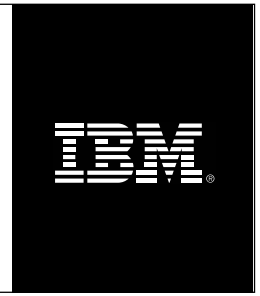
ibm.com/support

IBM Global Services

ibm.com/services

Creating OpenShift Multiple Architecture Clusters with IBM

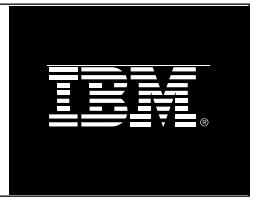
SG24-8565-00
ISBN 0738461865



(1.5” spine)
1.5”<-> 1.998”
789 <->1051 pages

Creating OpenShift Multiple Architecture Clusters with IBM

SG24-8565-00
ISBN 0738461865



(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Creating OpenShift Multiple Architecture Clusters with IBM Power

SG24-8565-00
ISBN 0738461865

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Creating OpenShift Multiple Architecture Clusters with IBM Power

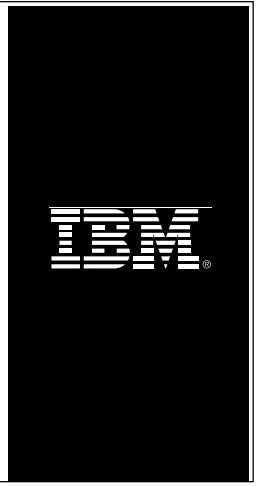
(0.2”spine)
0.17”<->0.473”
90<->249 pages

(0.1”spine)
0.1”<->0.169”
53<->89 pages

Creating OpenShift Multiple Architecture

SG24-8565-00

ISBN 0738461865

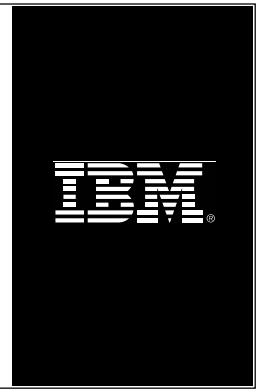


(2.5” spine)
2.5”<->nnn.n”
1315<-> nnnn pages

Creating OpenShift Multiple Architecture Clusters with IBM Power

SG24-8565-00

ISBN 0738461865



(2.0” spine)
2.0” <-> 2.498”
1052 <-> 1314 pages

SG24-8565-00

ISBN 0738461865

Printed in U.S.A.