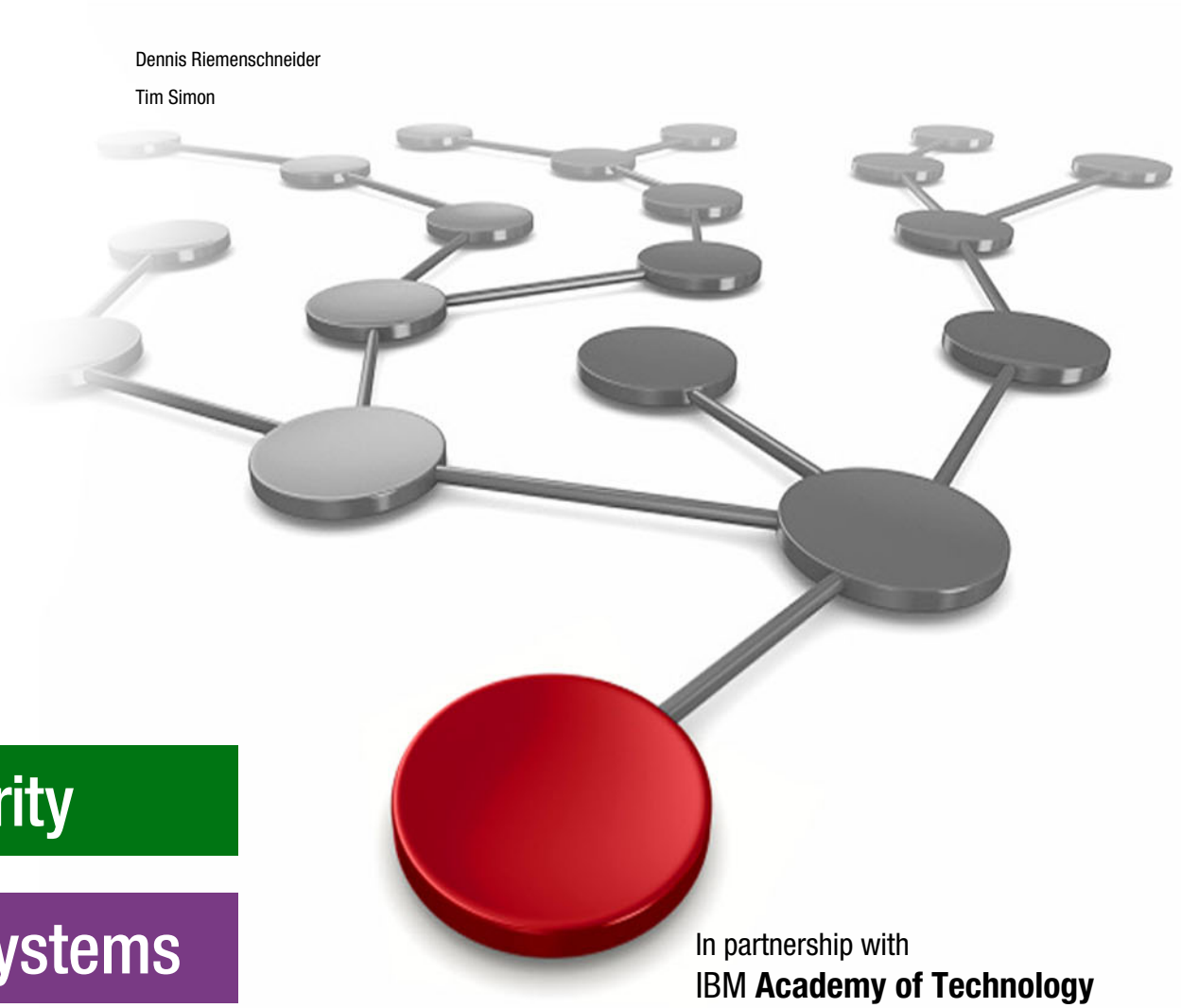
Redbooks

ibm.com/redbooks

Security Implementation with Red Hat OpenShift on IBM Power Systems

Dino Quintero
John Adegbile
Faraz Ahmad
Sambasiva Andaluri
Agustin Barreto
Olavo Borges
Ivaylo Bozhinov
Daniel Casali
Gayathri Gopalakrishnan
Nilabja Haldar
Abhishek Jain
Josephine Eskaline Joyce
Youssef Largou
Amrita Maitra
David Pearson
João André Pellizzari
Dennis Riemenschneider
Tim Simon



Security

Power Systems

In partnership with
IBM **Academy of Technology**

IBM

Redpaper

IBM Redbooks

Security Implementation with Red Hat OpenShift on IBM Power Systems

April 2023

REDP-5690-00

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

First Edition (April 2023)
This edition applies to the following software versions:
- Red Hat OpenShift Container Platform 4.9 for IBM Power
- Red Hat OpenShift Container Platform 4.10 for IBM Power
- Red Hat OpenShift Container Platform 4.11 for IBM Power
- Red Hat Advanced Cluster Security for Kubernetes (K8s) 3.72.1
- Red Hat OpenShift Data Foundation 4.10.7
- Red Hat OpenShift Data Foundation 4.11
- Red Hat OpenShift APIs for Data Protection 4.11
- Red Hat OpenShift Pipelines 1.8.0
- Red Hat OpenShift File Integrity Operator 0.1.32
- IBM Cloud Shell 1.0.67
- Red Hat OpenShift Compliance Operator 0.1.57.
- IBM Security Guardium Data Encryption 5.0.
- OpenSCAP Scanner 1.3.4.
- IBM Spectrum Virtualize 8.5
- IBM Spectrum Protect Plus 10.1.10
- Velero 1.9
- Aqua Starboard 0.15.8
- Aqua Trivy v0.36
- Aqua Security 2022.4.196
- IBM PowerVM 3.1.3
- IBM PowerVM 3.1.4

Contents

Notices

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX®	IBM Consulting™	POWER9™
Cognos®	IBM FlashSystem®	PowerHA®
Db2®	IBM Security®	PowerVM®
DS8000®	IBM Spectrum®	QRadar®
Guardium®	IBM Watson®	Rational®
IBM®	Instana®	Redbooks®
IBM Cloud®	OS/400®	Redbooks (logo)  ®
IBM Cloud for Financial Services®	Power Architecture®	Spectrum Fusion™
IBM Cloud Pak®	POWER8®	WebSphere®

The following terms are trademarks of other companies:

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

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

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

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

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

Red Hat OpenShift is a powerful and flexible container orchestration platform that enables organizations to build, deploy, and manage applications in a cloud-native environment. As with any production system, you must ensure the security of an Red Hat OpenShift deployment, which includes secure deployment and configuration of the Red Hat OpenShift components, and ongoing maintenance and monitoring to ensure the continued security of the environment.

This IBM® Redpaper publication provides a comprehensive overview of the security best practices for deploying Red Hat OpenShift on IBM Power. It covers the essential steps to secure your Red Hat OpenShift environment and ensure the confidentiality, integrity, and availability of your data and applications.

Authors

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

Dino Quintero is a Systems Technology Architect with IBM Redbooks®. He has 28 years of experience with IBM Power technologies and solutions. Dino shares his technical computing passion and expertise by leading teams developing technical content in the areas of enterprise continuous availability, enterprise systems management, high-performance computing (HPC), cloud computing, artificial intelligence (AI) (including machine and deep learning), and cognitive solutions. He is a Certified Open Group Distinguished Technical Specialist. Dino is formerly from the province of Chiriqui in Panama. Dino holds a Master of Computing Information Systems degree and a Bachelor of Science degree in Computer Science from Marist College.

John Adegbile is an independent middleware consultant with a focus on enterprise messaging and Enterprise Service Bus (ESB) technologies such as IBM MQ, IBM Integration Bus, IBM App Connect Enterprise, and Kafka. For the last 14 years, he has worked with clients in Canada, the US, and Europe in various industries, such as insurance, rail, banking, retail, media, and advertising. Before becoming a freelance consultant, John worked for 13 years as an IT Specialist with IBM Global Services UK.

Faraz Ahmad is an IBM Power Systems solution architect working at IBM Technology Services, India. Faraz has experience in various areas of IT, including software development, solution design, and IT consultation. He specializes in cybersecurity, and in his current role, he designs security solutions for IBM customers. He is a geography lead and mentors security consultants in Central and Eastern Europe, the Middle East, and Africa regions. His other areas of expertise include IBM PowerHA®, IBM AIX®, Linux, networking, and virtualization. He is the author of multiple patents and recognized as an Invention Plateau holder. He has a degree in Computer Science from the Birla Institute of Technology, Ranchi, India.

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- and post-sales solution architect for trading systems at Fidessa, a pre-sales solution architect at AWS, and a Site Reliability Engineer (SRE) for onboarding ISVs for Google Marketplace at a Business Partner. He brings multifaceted experience to the table, and is a continuous learner and a strong supporter of STEM.

Agustin Barreto is a cloud architect who holds the position of Chief Technology Officer at Inco S.A., one of the top IBM vendors in Uruguay. His work is focused on discovering new technologies and helping customers with their modernization process. He is studying system engineering at Universidad de la República. He has worked on several open-source projects as a cloud-native developer consultant. He has the role of door opener within the company, delegating responsibilities and consolidating departmental structures to expand the service portfolio. Most recently, he works on Lift and Shift modernization projects for migrating services to the cloud.

Olavo Borges has more than 20 years of experience in the IT industry. He works at IBM supporting large financial services and public sector customers. He holds over 30 certifications in the IT industry, including technical certifications from cloud providers like IBM, Microsoft, AWS, and Google; certifications in technologies like Red Hat OpenShift; and management and soft-skills certifications like Project Management Professional (PMP). He is a continuous learner and a Georgia Tech Cybersecurity Master's student.

Ivaylo Bozhinov is a Technical Support Professional SME for Flexible Service Processor (FSP), Hypervisor, and enterprise Baseboard Management Controller (eBMC)) for the IBM Power 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 a master's degree in Cybersecurity from New Bulgarian University. He supports many clients in the banking industry and telecom and retail sector.

Daniel Casali is a Thought Leader Information Technology Specialist who has been working at IBM for 15 years in the IBM Power, HPC, big data, and storage areas. His role at IBM is realizing solutions that address client's needs by exploring new technologies for different workloads. He explores multi-cloud implementations to abstract and simplify the new challenges of the heterogeneous architectures of this model for both on-premises and in the public cloud.

Gayathri Gopalakrishnan works at 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. She is a recognized leader that applies 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 convert business requirements into technical solutions.

Nilabja Haldar is an experienced Cloud Architect and SRE, and a certified AWS, Google Cloud Platform (GCP), Azure, IBM Cloud®, and Red Hat OpenShift solution Architect, with 15 years of experience in various IT domains like public and hybrid multicloud, technical consultation, solution design, design, implementation, transformation and migration, and data center consolidation for organizations world wide. He works in IBM Consulting™ as an Infrastructure and Cloud architect, DevOps, and SRE. He has BTech degree in Computer Sc, and his technical skills cover hybrid cloud, GCP, Azure, IBM Cloud, Kubernetes (K8s), Red Hat OpenShift, DevOps, security, observability, integration, and open-source software.

Abhishek Jain is an IBM Master Inventor and an Automation Architect with IBM Systems Development Labs, India. He has more than 10 years of experience working on various storage technologies, such as scale-out file systems, and block and object and container native storage. In his current role as Test Architect, he leads a quality process for IBM Container Native Storage Access and IBM Spectrum® Scale CSI Driver deliveries. He holds a Bachelor of Technology degree in Computer Engineering.

Josephine Eskaline Joyce is an IBM Master Inventor and Cloud Architect at IBM India Software Labs, with 20+ years of experience in the IT industry. She has extensive experience in enterprise application architectures, designing various customer solutions, and developing many IBM products. She has several patents and paper presentations to her credit. She authored a book on mobile application development, and is a research scholar in cloud computing.

Youssef Largou is the founding director of PowerM, a platinum IBM Business Partner in Morocco. He has 21 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®, and Red Hat OpenShift. He has worked within numerous industries with many technologies. Youssef is an IBM Champion 2020, 2021, and 2022, an IBM Redbooks Gold Author, and has designed many reference architectures. He has been recognized as an IBM Beacon Award Finalist in Storage, Software-Defined Storage, and LinuxONE five times. He holds an engineer degree in Computer Science from the Ecole Nationale Supérieure des Mines de Rabat and Excecutif MBA from EMLyon.

Amrita Maitra is a certified Application Architect who specializes in the migration of client workloads to AWS. She is an expert in .NET technologies and extensively uses design thinking in customer deliverables. She works with clients from various sectors, predominantly the finance domain. Designing for resiliency, infrastructure sizing, costs, and high availability are some of the key areas of her work.

David Pearson works for IBM UK and has over 25 years' experience as a technical solution and enterprise architect, working in both pre-sales and consulting roles throughout his career. He provides technical leadership to clients, helping them align their business goals and technical challenges with open-source software and cloud technologies that are focused on the IBM and Red Hat platforms. He is a Chartered Engineer (CEng), and his technical skills cover social software, mobile, hybrid cloud, K8s, security, integration, and open-source software. He has deep expertise with architectural frameworks and design thinking approaches, and has delivered hundreds of consulting engagements globally during his career.

João André Pellizzari is a certified senior architect with over a decade of experience in the field. He has a wealth of knowledge in architectural thinking, enterprise architecture, and technical leadership, and expertise in IBM Cloud, Docker, K8s, and Red Hat OpenShift. In his most recent role as the chief architect of the Configure to Order subdomain, he is responsible for transforming a portfolio to leverage IBM Cloud and AI. Before taking this role, he served as the lead architect of a global transformation initiative that aimed to simplify and transform the end-to-end selling process for IBM Systems hardware. He also previously worked as a Solutions Architect, where he worked on the first internal IBM Watson® implementation in Latin America, where he was recognized for his technical achievements.

Dennis Riemenschneider works for IBM Germany. He has 22 years of experience in IT. After his IBM apprenticeship, he started with IBM Hardware Support Services, followed by Software Support Services, IT Consulting, and Platform Engineering. He is an IT software specialist and solution architect. His extensive and fundamental hardware knowledge of servers, storage, and networking is complemented with deeper knowledge of open-source software, Linux, automation, and his current fields of expertise: Infrastructure as Code (IaC), cloud computing, K8s, and Red Hat OpenShift.

Tim Simon is an IBM Redbooks Project Leader in Tulsa, Oklahoma, US. 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 by using IBM Power, IBM Storage, and IBM zSystems throughout his career.

Now you can become a published author, too!

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

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

Stay connected to IBM Redbooks

- Find us on LinkedIn:

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

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

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

- Stay current on recent Redbooks publications with RSS Feeds:

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

1

Introduction

This publication is intended for readers of all levels. The reader should understand containers, Kubernetes (K8s), IBM Power Systems Virtual Server (IBM PowerVS) and Red Hat OpenShift. The typical targeted audiences of this publication are developers, security consultants, architects, and technical project managers.

This chapter describes the following topics:

- Purpose
- Scope
- Audience
- Challenges in the cloud-native world

1.1 Purpose

It is a business imperative for many organizations to improve their cybersecurity stance. With the increasing trend toward hybrid cloud, application modernization, automation, and many more workload scenarios, leveraging the existing abilities of the IBM Power platform along with modern containerization solutions such as Red Hat OpenShift is critical to success. There are many considerations when planning, designing, deploying, and managing a secure K8s or Red Hat OpenShift architecture:

- Evaluating security frameworks
- Assessing your security stance
- Understanding the nature of attacks and approaches that attackers use to compromise your environment
- Understanding the technical building blocks
- Understanding how to define a secure architecture

These critical aspects must be addressed as you implement a “security first” approach.

The purpose of this publication is to provide insight, education, and guidance into the range of considerations for the optimal design, implementation, and secure operation of a K8s or Red Hat OpenShift environment when using IBM Power servers.

This publication aims to provide an insight into IBM Power capabilities that you can use to successfully enable a secure hosting environment for your business-critical applications on a K8s or Red Hat OpenShift architecture.

1.2 Scope

In this publication, we focus on Red Hat OpenShift 4 on the IBM POWER9™ and IBM Power10 platforms. We provide an overview of the architectural building blocks, along with an explanation of security frameworks and attack vectors so that you can develop a layered defense approach when designing and implementing your environment.

We cover the usage of IBM PowerVS capabilities including monitoring and logging capabilities, and provide insight into the aspects of regulations and compliance requirements that your architecture might need to address.

1.3 Audience

This publication is intended for readers of all levels. The reader should understand containers, K8s, IBM PowerVS, and Red Hat OpenShift. The typical targeted audience of this publication is developers, security consultants, architects, and technical project managers. For novice users, understanding the information at the following links provides a basis for better understanding the contents of this publication:

- What are containers?
- What is Kubernetes?
- IBM Power Systems Virtual Server
- Red Hat OpenShift

- IBM Developer
- IBM Developer at YouTube

1.4 Challenges in the cloud-native world

For large customers to survive and thrive in an era of constant change, they must redesign many of their processes and platforms. Current business models and systems are not designed to respond to marketplace shifts and regulatory demands with the speed and agility that are required to stay relevant, competitive, and compliant.

1.4.1 Understanding cloud-native security challenges

A corresponding combination of measures focusing on trust, service, and economics is needed to counteract the forces of cybercrime while still providing a strong ecosystem that empowers organizations to use their resources to provide better business outcomes. Together, these initiatives can be enabled by shifting of existing business models from discrete offerings to the automated orchestration of a secure, experience-based ecosystem of new and existing products.

To achieve this change, consumer experiences that were previously provided by people and physical locations and helped by technology must now be provided by technology. IT infrastructure underpins this transformation. Business strategy and technology have become inseparable, and architectural decisions are now critical business decisions.

Here are the major disruptive trends for any customer embracing a digital transformation journey:

- Customer behaviors and expectations: Competitors raised the bar with innovative business models in the battle to win the customer. Business to business and business to customers providers are responding with massive investments to understand and enhance the user experience by offering self-service options and unbundling products to improve choice.
- Digital competition: New and existing competitors are leveraging new digital technologies to deploy non-traditional business, operating, and technology models that are user-centric, data-intensive, and cloud-native.
- Competition for talent: Fewer customers and institutions believe that their organization offers their employees the resources or skill-development opportunities that they need to thrive in a digital environment. Digital behaviors create the environment so that firms can do business digitally by providing agility, collaboration, distributed organizational structures, a bolder risk appetite, and true customer centricity.
- Regulatory compliance: Regulatory compliance activities continue to be complex and costly. The sheer volume and growth of compliance regulation, when combined with the costs that are related to non-compliance, make this area a continued focus.
- Security and fraud: Expanding channels and partner ecosystems, cloud-based solutions, digital platforms, and 24x7 services drive increased security risks and threats, and increase the opportunity for fraud, which raises the need to address security.
- Data, analytics, AI and cognitive: The industry has access to vast amounts of rich data about its clients. The new and winning customer business models of tomorrow will succeed by delivering on insights from their vast stores of data, which are enabled through intelligent analytics, and driven by cognitive processes at the core of their models.

As cloud platforms become the technical backbone of any core modernization project including digital transformation and compliance initiatives for a customer, the following challenges and concerns have exponentially increased:

- Cloud-native: Customers require flexible computing with consistent development, deployment, management, orchestration, and automation. Enabling a hybrid cloud enterprise is key to improving internal operations and efficiency, retaining and growing clients, and reducing costs. Cloud-native helps by automating labor-intensive work, expanding the ecosystem through cloud-based interaction with core systems, and capturing high business value insights from customer data.
- Encryption everywhere: As customers expand their touch points, channels, and business partners, they expose themselves to bigger risks for security breaches, fraud, and cybercrime, which leads to a lack of customer trust and can decrease your profitability. Customers are demanding security for their whole hybrid cloud enterprise. Customers must ensure the privacy of their data 100% of the time wherever it is across the cloud infrastructure.
- Cyber resilience: Because more customers are operating on “Always On Mode” and continue to expand, they must have systems that are reliable, especially for their security updates and planned outages. They must move from manual to predictive innovations in risk and compliance while investing in consulting services to keep them ahead of new regulations.

Unfortunately, rapid adoption and fast upstream development of open-source projects can sometimes prevent necessary and newly introduced security protections from being applied.

Containers are popular because they make it simple to bundle an application with all of its dependencies into a single image that can be deployed consistently and without requiring any change from development, test, QA, and then to production. Containers can be deployed on heterogeneous targets while providing the following benefits:

- Portability: The lightweight nature of containers can make them easy to move.
- Faster start: Containers can start and restart in seconds.
- Lower resources: Containers consume less CPU and memory.
- Management cost: Container management is streamlined with K8s.
- Hybrid multi architecture support: The lightweight nature of containers enables an application to be moved quickly between hybrid multi-cloud environments.

Using containers can provide flexibility and security benefits compared to traditional, monolithic software platforms. However, there are extra complications that are involved in maintaining everything securely, including the underlying infrastructure and microservices.

According to the National Security Agency (NSA),[1] the threats that are shown in Table 1-1 on page 5 represent the most likely sources of compromise for containers.

[1] https://media.defense.gov/2022/Aug/29/2003066362/-1/-1/0/CTR_KUBERNETES_HARDENING_GUIDANCE_1.2_20220829.PDF

Table 1-1 NSA identified threats

Supply chain		
Container or application level	**Container run time**	**Infrastructure**
A malicious third-party container or program might give cybercriminals access to the cluster.	Inadequate container separation might result from a flaw in the container run time.	Cybercriminals might get access to the cluster if systems that are employed as worker nodes or as part of the control plane are compromised.
Malicious threat actor		
Control plane	**Worker nodes**	**Containerized applications**
Cybercriminals commonly exploit unprotected control plane components with insufficient access constraints.	Worker nodes are present outside of the secured control plane and can be more accessible to online criminals.	Using an exposed application's internally accessible resources, an actor can switch from a pod that has already been compromised or increase their privileges within the cluster.
Insider threat		
Administrator	**User**	**Cloud service or infrastructure provider**
System or hypervisor administrators frequently have physical access to these items, which might be leveraged to undermine the container's environment.	Users of containerized applications might have access to containerized services. This degree of access might offer adequate tools to compromise the application or other containers' elements.	A container's environment might be compromised by using physical access to systems or hypervisors that control a K8s node.

By leveraging IBM Power security features, Red Hat OpenShift Container Platform can deliver hardened security across the entire stack, from chip to application, for cloud-native and containerized workloads.

1.4.2 IBM Power10 unique security features for containers

IBM Power10 servers protect sensitive data by leveraging the latest pervasive encryption capabilities across hybrid cloud deployments. The Power10 processor introduces full memory encryption at scale. Transparent memory encryption is designed to simplify encryption and support end-to-end security without impacting performance by leveraging hardware features for a seamless user experience.

Additionally, workloads on Power10 servers benefit from cryptographic algorithm acceleration hardware so that algorithms like AES, SHA2, and SHA3 to run faster on Power10 servers compared to other platforms.

To be prepared for the quantum era, IBM Power10 servers are built to efficiently support upcoming cryptography techniques such as quantum-safe cryptography and Fully Homomorphic Encryption (FHE).

According to the European Telecommunications Standards Institute (ETSI), “Quantum-safe cryptography refers to efforts to identify algorithms that are resistant to attacks by both classical and quantum computers to keep information assets secure even after a large-scale quantum computer has been built.”[2]

For more information about Quantum-Safe Cryptography, see this IBM Blog.

IBM Power servers offer the most secure workload isolation in cloud deployments, with integrity engineered into every layer of the system. All components of the stack are fully integrated and co-optimized, and they are provided from IBM as a single vendor, which makes the stack more secure.

IBM PowerVM®, which is the built-in hypervisor, has an outstanding track record. It has orders of magnitude fewer vulnerabilities than competitive hypervisors. Power10 servers have advanced firmware integrity with extra measures to isolate the CPU from service processors for better defense against attacks on management systems.

Power10 servers introduce innovations to address emerging threats, with extra features and enhancements to defend against application domain vulnerabilities such as return-oriented programming (ROP) attacks.

A global zero trust security strategy is essential, and container security starts with Linux security:

- Security in the Red Hat Enterprise Linux (RHEL) and CoreOS host applies to the container.
- IBM Power10 with Linux hosts (worker nodes) offer industry leading integrity and isolation.
- The correct configuration of SELinux and **`seccomp`** along with the usage of namespaces strengthens isolation.
- SELinux mitigates container runtime vulnerabilities.
- You must protect the host from container escape and the containers from each other.
- RHEL and CoreOS offer minimized attack surface.

IBM Enterprise Protected Containers (EPC) offer isolated, secure, and integrity protected Red Hat OpenShift containers that offer the same level of security as logical partitions (LPARs):

- EPC provides the same strength of isolation to containers that is afforded by virtual machines (VMs) or LPARs.
- EPC containers act as lightweight VMs, which use hybrid virtualization that is co-optimized in Power10 servers and PowerVM.
- EPC provides end-to-end protection of code and data, including when in use (confidential computing).
- EPC can simplify workload regulatory compliance, especially in multi-tenant environments.
- At the time of writing, PowerVM has no CVEs, and has an orders of magnitude lower number of CVEs than other hypervisors.

For more information, see *IBM Power Systems Cloud Security Guide: Protect IT Infrastructure In All Layers*, REDP-5659.

[2] https://www.etsi.org/technologies/quantum-safe-cryptography

2

Building blocks and IBM Power capabilities

This chapter introduces the security features that can be used to secure your cloud environment when it is running on IBM Power servers. We describe both hardware and software features to help you provide the security that is required to run your cloud-ready applications.

This chapter describes the following topics:

- IBM Power capabilities and features
- Storage
- Orchestrators and K8s
- Ingress Controller
- Container registry
- Red Hat OpenShift on IBM Power Virtual Server
- IBM Cloud Paks

2.1 IBM Power capabilities and features

Power servers are designed for security. Security is designed into the processor, hypervisor, operating system, communications, storage, and applications. Built-in cryptography hardware enables better scalability and an improved user experience.

Power10 servers include security features to help users comply with security-related regulatory requirements, such as:

- Identity and Access Management (IAM)
- Hardware and software encryption
- Communication security capabilities
- Extensive logging and reporting of security events

Cloud-native security complements the security that is provided as standard to virtual machines (VMs) by the IBM Power10 processor-based infrastructure.

Power10 security enables the following features:

- Cryptographic performance acceleration and main memory encryption

  IBM introduced transparent memory encryption, which is handled in the hardware to avoid performance degradation. This new memory encryption is a key security feature for containers, which are increasingly popular, particularly when it comes to cloud computing. There is support for quantum-safe cryptography and Fully Homomorphic Encryption (FHE). Power10 servers also have more crypto accelerators per core, which deliver faster AES crypto performance.
- Performance-enhanced, side-channel avoidance
- Protection against service processor vulnerabilities
- Defense against application vulnerabilities

Power servers incorporate the hardware and software that provide industry-leading defense against ransomware attacks by providing the following features:

- Prevention features:
  - Power servers provide industry-leading isolation and integrity that help prevent ransomware from being installed.
  - Host and firmware secure and trusted boot.
  - Guest OS secure boot.
  - Built-in OS runtime integrity: Linux Integrity Measurement Architecture (IMA).
  - A secure multi-tenant environment with orders of magnitude lower number of CVEs.
  - Simplified patching with IBM PowerSC.
  - Multi-factor authentication (MFA) with PowerSC MFA.
- Early detection:
  - Integrated security and compliance management with PowerSC makes it harder to misconfigure, and easier to detect anomalies.
  - Offerings such as IBM Security® QRadar® or Aqua Security enhance inherent security with early anomaly detection.

- Fast and efficient recovery:
  - Deploy data resiliency strategies with PowerHA and IBM Storage Safeguarded Copy.
  - Collaboration with IBM Storage and Security Services for fast detection and automated recovery of affected data.
- A multi-layered approach to security:
  - Base IBM Power platform integrity:
    - Secure and trusted boot.
    - Power10 enhanced CPU with Flexible Service Processor (FSP) and Baseboard Management Controller (BMC) isolation.
    - Power10 main memory encryption.
  - Workload security enablement:
    - Cryptographic algorithm acceleration.
    - Secure Key Storage (Platform Key Store).
    - Support for post-quantum cryptography (PQC) and FHE crypto algorithms.
  - End-to-end data encryption (bring-your-own-key (BYOK)):
    - Integration with IBM Hyper Protect Crypto Services (IBM HPCS) Cloud Key Mgmt Services.
    - Hybrid cloud enablement facilitates secure data transfer to and from an enterprise and the cloud.
  - Container Security Ecosystem: ISV: Aqua (Container Native Security).

For more information, see *IBM Power Systems Cloud Security Guide: Protect IT Infrastructure In All Layers*, REDP-5659.

2.1.1 IBM PowerVM hypervisor

The PowerVM hypervisor is a standard component of the system firmware, and it is considered the foundation for PowerVM. The PowerVM hypervisor provides an abstraction layer between the physical hardware resources and the logical partitions (LPARs) that use them. The PowerVM hypervisor can divide physical system resources into isolated LPARs. It can assign dedicated processors, memory, and I/O resources that can be dynamically reconfigured as needed to each LPAR. The PowerVM hypervisor can also assign shared processors to each LPAR.

Here are the main functions of the PowerVM hypervisor:

- Virtual Memory Management.
- Virtual Processor Management.
- Provides security and isolation between partitions.
- Provides virtual console support.

Starting with new Power10 Scale-Out Systems and Enterprise 9043-MRX (excluding 9080-HEX), all Power servers use enterprise Baseboard Management Controllers (eBMCs) instead of FSP. Power servers with eBMC use the Virtualization Management Interface (VMI), as shown in Figure 2-1.

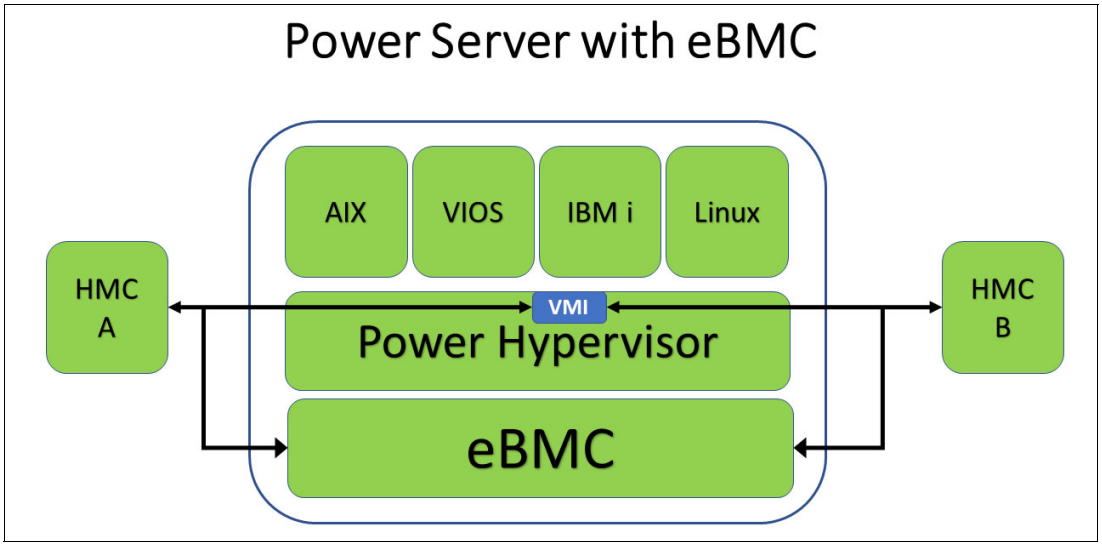


Figure 2-1 Power server with eBMC

Figure 2-2 shows a Power server that uses FSP.

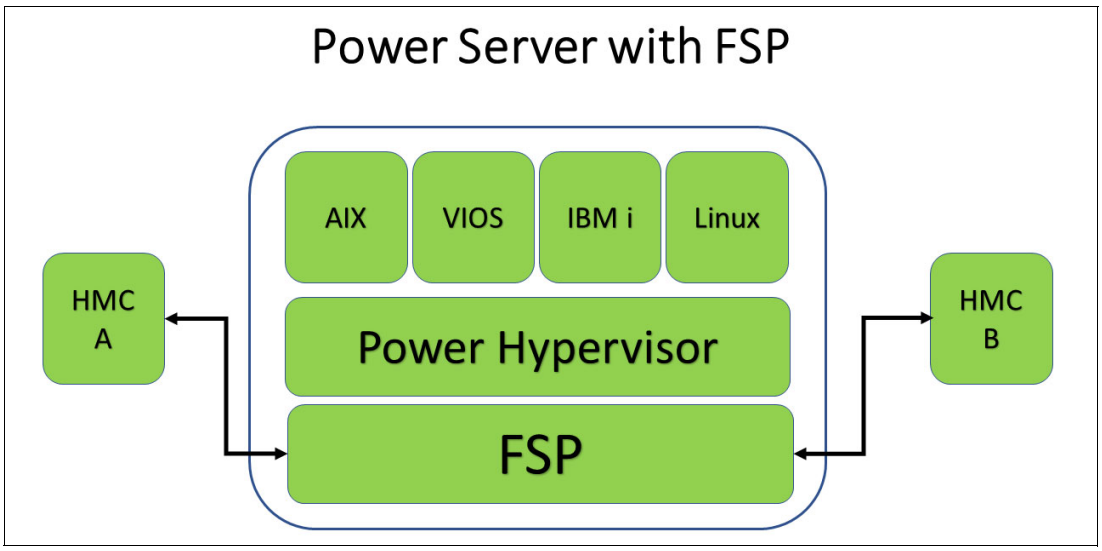


Figure 2-2 FSP-based Power server

For more information about how to configure VMI or eBMC and HMC, see IBM Power Community.

IBM has a corporate policy that products that are produced by IBM follow the IBM Security and Privacy by Design. The PowerVM and Power hardware teams put “security always” on the top of the list of priorities. Protection of client data is of paramount importance. For more information, including the target of evaluation (TOE) that is covered in Virtual I/O Server (VIOS) and PowerVM hypervisor, see PowerVM Common Criteria Certification.

The security benefits of PowerVM compared to other virtualization technologies are shown in Table 2-1 by the significantly lower number of CVEs that are reported.

Table 2-1 Virtualization CVE comparison

CVE type	PowerVM	VMWare ESX	Microsoft Hyper-V	KVM
Virtualization Technology CVEs	PowerVM: 9 VIOS: 53	448	176	193

Similarly, the security benefits of AIX and IBM i as shown by the significantly lower number of CVEs that are reported against them is shown in Table 2-2. Even for Linux, the number of CVEs that are reported is about 40% fewer than the number that is reported for Windows.

Table 2-2 OS-level CVE comparison

CVE type	IBM AIX	IBM i	Windows	Linux
Operating systems CVEs	378	IBM i: 35 IBM OS/400® 13	10191	6480

There are other advantages to using PowerVM when securing customer data and workloads:

- Enhanced defenses against return-oriented programming (ROP) attacks:
  - A new in-core hardware architecture with a low hardware footprint and a standard-based cryptographic solution to protect the integrity of the return stack.
  - Four new instructions in the Power ISA 3.1B.
  - The defenses are controlled by the Dynamic Execution Control Register (DEXCR), which provides per-thread mechanisms to control features with security or performance tradeoffs or speculation.
- Enhanced CPU isolation from service processors: Extend IBM POWER9 capabilities that limit the CPU resources that BMC and FSPs can access, such as combining the allowlist and blocklist built-in hypervisor approaches.
- Performance-enhanced side-channel avoidance: Enhanced handling of automatic thread isolation from speculation-based attacks.
- Secure and trusted boot for host and guest LPARs for the following operating systems:
  - AIX.
  - Linux that is available in Red Hat Enterprise Linux (RHEL) 8.5 with static keys.
- Using Trusted Platform Module (TPM) for the following features:
  - Root of Trust for Measurement (RTM) Measure (compute hashes) of firmware software components to create a hardware-based immutable record of what is loaded on a system. The record can be presented to remote systems for attestation.
  - Local Key Management and Storage keys can be generated, signed, and sealed for specific measurements or policies (allows controlled access to keys):
    - Physical TPM: Used by a physical host.
    - Virtual TPM Software emulation of TPM is offered by the hypervisor to guest partitions (LPARs).

2.2 Storage

Persistent storage for “stateful” containerized applications is difficult for the industry. Unlike monolithic applications. which reserve storage resources once, containers and microservices come in and out of existence, sometimes migrating between machines at breakneck speeds. The repeating cycle of coding and testing, which is followed by agile production deployment, further exacerbates the data storage challenges for containerized applications. Data services that are fashioned for traditional application architectures must serve a new, transient data paradigm.

By default, when a container is created from its master image, an ephemeral read/write layer is created that handles all written data. When the container stops, whether intentionally terminated, unintentionally terminated, or because the underlying pod failed, that read/write data layer disappears with the container. Any writes that are performed to the container are limited to that container’s lifetime. Even if a container is restarted by the orchestrator, the storage that is written to the ephemeral layer in the old container is lost. However, not all storage for containers must be ephemeral. The nature of some applications, for example, databases, need some persistent storage for work that is done by non-trivial containers.

Volumes and persistent volumes

The fundamental difference between a volume and a persistent volume (PV) is that a volume exists for the lifetime of the pod. If the pod persists, the volume also persists. When the pod ceases to exist, its volumes also cease to exist. Multiple volumes and classes of volume can be used by a pod simultaneously, but volumes are managed as non-persistent data that is bound to the lifecycle of the pod.

PVs are pieces of storage that are provisioned by the system administrator or are dynamically provisioned by using a storage class. PVs are defined to persist longer than the lifecycle of any pod.

Kubernetes (K8s) treats PVs as a cluster resource, much like it treats a node as a cluster resource. A cluster resource is available for a process to call and use, and it is maintained independent of any individual pod that uses the resource, which in this case is the PV.

Here is a couple of use cases for allocating persistent block storage:

- Database applications, such as IBM Db2 and MySQL
- Continuous integration and continuous delivery (CI/CD) products, such as Jenkins

A consideration for data volumes in a containerized environment is the access mode of the storage volume:

- ReadWriteOnce (RWO)

  The volume can be mounted as read/write by a single pod.
- ReadOnlyMany (ROX)

  The volume can be mounted as read-only by many pods.
- ReadWriteMany (RWX)

  The volume can be mounted as read/write by many pods.

PVs in K8s are allocated by a persistent volume claim (PVC), which defines the volume access mode. Although a volume may be mounted in multiple ways, it can be mounted in only one mode at any time.

Challenges for data storage for containers

Manageability is one significant challenge for clients running containers. Clients are looking for tools to manage their containerized applications and the required data. These tools should provide capabilities to deploy, manage, monitor, and scale applications while providing access to the data that is required by those applications.

Data protection is another challenge for clients. Containers can be running business-critical applications, and the data that is used by those applications needs robust backup and disaster recovery routines and should include contingency planning for protecting the state of the cluster, container image registries, and runtime information.

2.2.1 Container Storage Interface

The Container Storage Interface (CSI) driver was created to provide a vendor-neutral interface to block and file storage for use in a containerized environment. Before its introduction, if a vendor wanted to attach external storage to the container environment, they had to build code into the base K8s code, which was difficult to maintain and had to be done for each storage product. The CSI was designed with the objective of being an open specification for exposing block and file storage systems to container orchestration systems, K8s being one of them. The CSI is maintained on GitHub.

The fundamental benefit of the CSI driver is that it allows K8s to dynamically provision storage to bind to PVs for use by stateful containers. Otherwise, storage is allocated before the environment, volumes are created, and then claims are made by PVs to bind those volumes. The autonomy that CSI brings provides greater response, scalability, and management of the platform as a whole, including better usage of the underlying infrastructure.

In addition to dynamic provisioning, the CSI driver brings such capabilities as creating a snapshot of a volume, which can be attached to a new ReplicaSet. The CSI driver supports dynamic deprovisioning, and the ability to define thinly or thickly provisioned volumes.

All CSI drivers can perform the following tasks by using the defined CSI application programming interface (API):

- Dynamically provision or deprovision a volume.
- Enable local storage device mapping, for example, `lvm` or device mapper.
- Attach or detach a volume from a node.
- Mount or unmount a volume from a node.
- Consume block and mountable volumes (the latter for CSI file drivers).
- Create or delete a snapshot.
- Provision a volume from a snapshot.

IBM features the following written CSI driver families:

- The IBM block storage CSI driver, which is used by K8s for PVs, dynamic provisioning of block storage, and volume snapshots.[1]

  This driver supports the following storage systems:

  - IBM DS8000® family
  - IBM FlashSystem® A9000/R family
  - IBM Spectrum Virtualize based block storage, which includes IBM FlashSystem and SAN Volume Controllers

- IBM Spectrum Scale CSI driver for file-based storage.[2]

[1] https://www.ibm.com/docs/en/stg-block-csi-driver/1.10.0?topic=overview

For more information, see *IBM Storage for Red Hat OpenShift Blueprint*, REDP-5565 and *Using the IBM Block Storage CSI Driver in a Red Hat OpenShift Environment*, REDP-5613.

2.2.2 IBM Spectrum Fusion

IBM Spectrum Fusion™ is a container-native data services platform for Red Hat OpenShift. The solution helps bring applications to production faster by providing data services that are simple, reliable, consistent, and strategic. IBM Spectrum Fusion helps organizations to achieve the cloud-native agility and speed that they seek while mitigating the risks that are associated with the introduction of new technology.

As an enterprise data fabric, IBM Spectrum Fusion unlocks the ability to innovate and operate at full speed. From application development to data science to infrastructure modernization, IBM Spectrum Fusion helps organizations navigate cloud-native technologies with a simple, highly scalable and protected platform.

Figure 2-3 provides an overview of IBM Spectrum Fusion.

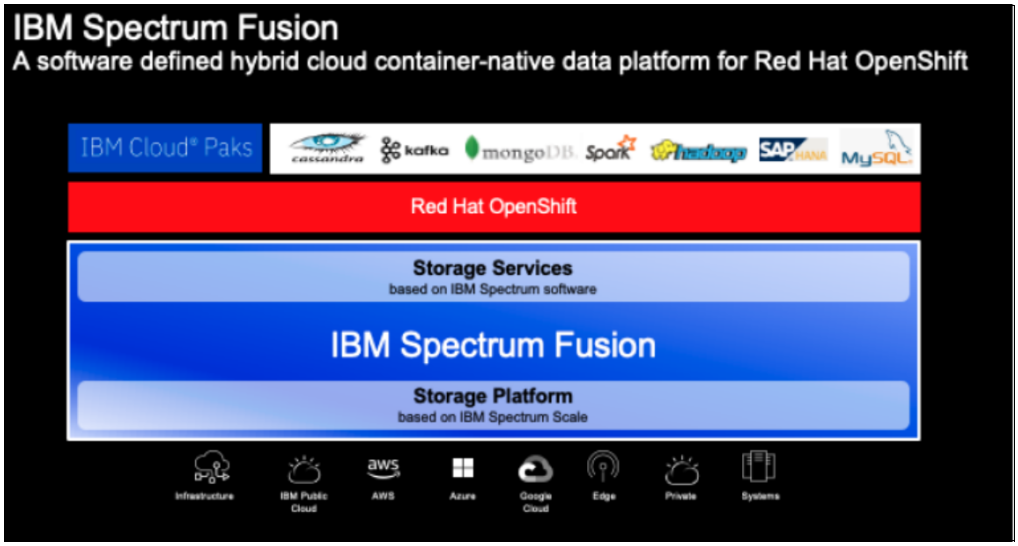


Figure 2-3 IBM Spectrum Fusion

IBM Spectrum Fusion delivers support for nearly all types of structured or unstructured data. IBM Spectrum Fusion has advanced data mobility features that help ensure that data is available on a global basis. When combined with advanced data discovery and cataloging, IBM Spectrum Fusion enables organizations to find the right data at the right time and present it anywhere globally.

IBM Spectrum Fusion is built on a market-leading technology that provides global access to data transparently to a container application. The application sees the data as another local file structure. The data can be physically placed in another data source up to thousands of miles away. This global data access includes S3 object data from the cloud or on-premises, Network File System (NFS) data from Dell/EMC, Netapp, or other vendors, and any IBM Spectrum Scale compatible storage system.

For more information about IBM Spectrum Fusion, see the IBM Spectrum Fusion website.

[2] https://www.ibm.com/docs/en/spectrum-scale-csi?topic=spectrum-scale-container-storage-interface-driver-26

2.2.3 Red Hat OpenShift Data Foundation

Red Hat OpenShift Data Foundation (previously known as Red Hat OpenShift Container Storage) is a software-defined storage orchestration platform for container environments that is provided by Red Hat. Red Hat now includes Red Hat OpenShift Data Foundation Essentials with Red Hat OpenShift Platform Plus. With this addition, Red Hat OpenShift Platform Plus provides an end-to-end solution with all the tools that organizations need. In addition to Red Hat OpenShift Container Platform, Red Hat OpenShift Platform Plus also includes Red Hat OpenShift Advanced Cluster Management for K8s, Red Hat OpenShift Advanced Cluster Security for K8s, Red Hat Quay container registry platform, and Red Hat OpenShift Data Foundation for persistent data services.

Red Hat OpenShift Data Foundation Essentials abstracts the details of the storage infrastructure while delivering data services that organizations need. Organizations can upgrade to Red Hat OpenShift Data Foundation Advanced to add more sophisticated data services functions.

Table 2-3 contrasts the capabilities that are provided by the two Red Hat OpenShift Data Foundation options.

Table 2-3 Red Hat OpenShift Data Foundation capabilities

Red Hat OpenShift Data Foundation Essentials	Red Hat OpenShift Data Foundation Advanced
► K8s RWO (block and file) ► K8s RWX (shared file and shared block) ► Object storage (S3-compatible) ► Internal mode storage (on-cluster) ► Volume snapshots ► Cluster-wide encryption ► Multicloud Object Gateway (MCG)	► External mode storage (shared cluster) ► Mixed usage patterns (off-cluster workloads) ► Volume-level encryption with BYOK support ► Metro disaster recovery ► Regional disaster recovery

Red Hat OpenShift Data Foundation uses a technology stack that consists of Red Hat Ceph Storage, Rook.io as a storage operator, and NooBaa as a storage gateway, behind which storage systems are knitted into a fabric design. Red Hat OpenShift Data Foundation uses CSI so that it can serve storage to platforms from pre-allocated storage and dynamically provision from storage subsystems that can use a CSI driver, such as the IBM FlashSystem family.

Red Hat OpenShift Data Foundation is packaged as an operator, and it is available through the Red Hat OpenShift Container Platform Service catalog to allow for deployment and management. Red Hat OpenShift Data Foundation can be deployed by using two methods: an internal storage cluster, and an external storage cluster.

The platform provides the following types of storage services, which are exposed through storage classes:

- Block storage: Primarily for database, logging, and monitoring workloads.
- Shared and distributed file: For CI/CD tools, messaging, and data aggregation workloads.
- Object storage: Provides a lightweight S3 API endpoint through NooBaa for abstraction of storage and retrieval from multiple object stores, which is ideal for cloud-native workloads or archival and backup data.

The Red Hat OpenShift Data Foundation platform uses the same stateful, declarative nature of K8s. It codifies administrative tasks and custom resources, which improves th automation of tasks and resources. Administrators can define the wanted state of the cluster, and Red Hat OpenShift Data Foundation operators can ensure that the cluster is in that state or approaching it while minimizing manual intervention.

For general-purpose persistent storage or dynamic provision requirements, Red Hat OpenShift Data Foundation is suitable for workloads like data science and data analytics, artificial intelligence (AI), machine learning, and Internet of Things workloads.

2.2.4 Enabling data encryption for IBM FlashSystem and IBM Spectrum Virtualize

To enable data encryption for block storage when creating Red Hat OpenShift Data Foundation Cluster for external IBM FlashSystem storage, complete the following steps:

1. In the Red Hat OpenShift Web Console, select **Operators** → **OperatorHub** to search for and install Red Hat OpenShift Data Foundation Operator, as shown in Figure 2-4.

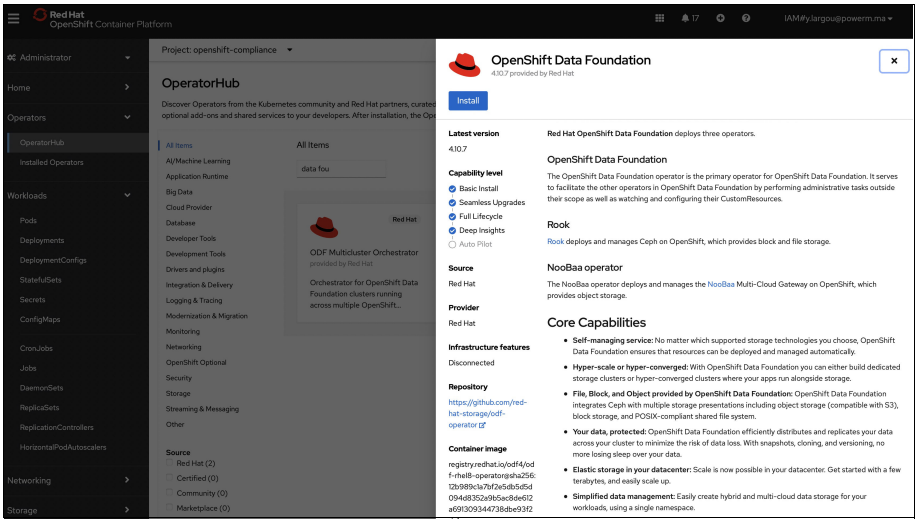


Figure 2-4 Installing Red Hat OpenShift Data Foundation Operator

2. The next window opens, as shown in Figure 2-5 on page 17. Select the update channel and other options and then click **Install**.

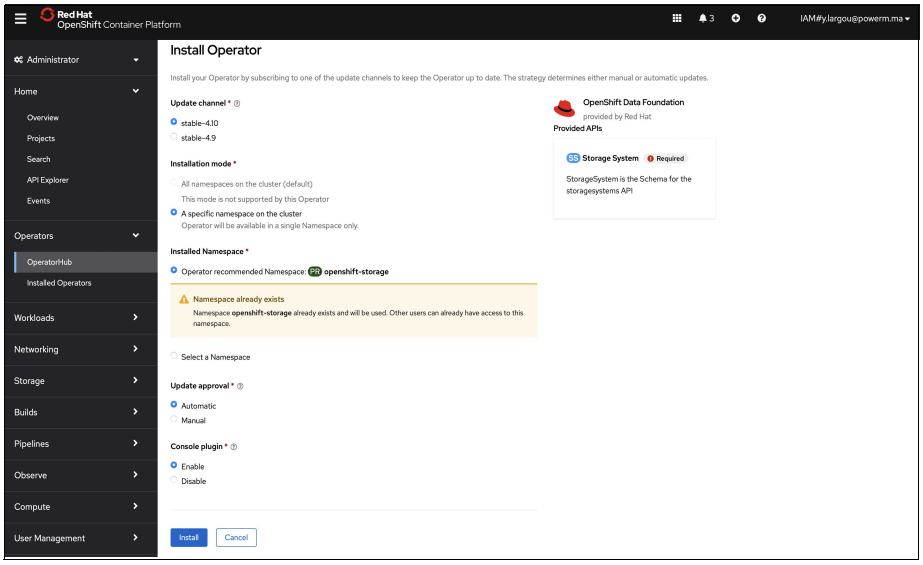


Figure 2-5 Red Hat OpenShift Data Foundation Operator settings

3. Check the Installed Operators, as shown in Figure 2-6.

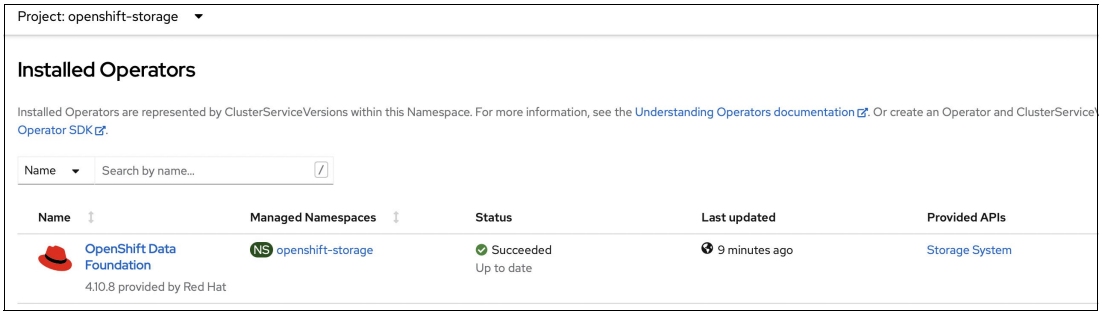


Figure 2-6 Checking the Red Hat OpenShift Data Foundation Operator installation

4. Click **Red Hat OpenShift Data Foundation** and then click **Create StorageSystem**, as shown in Figure 2-7.

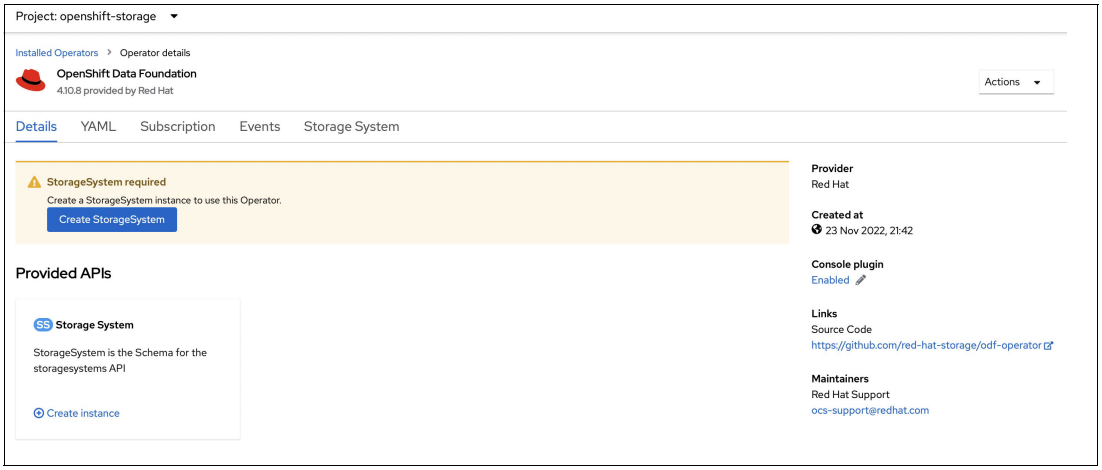


Figure 2-7 Create StorageSystem

5. In the Backing storage window, Select **Connect an external storage platform** from the available options and select **IBM FlashSystem Storage** from the Storage platform list, as shown in Figure 2-8.

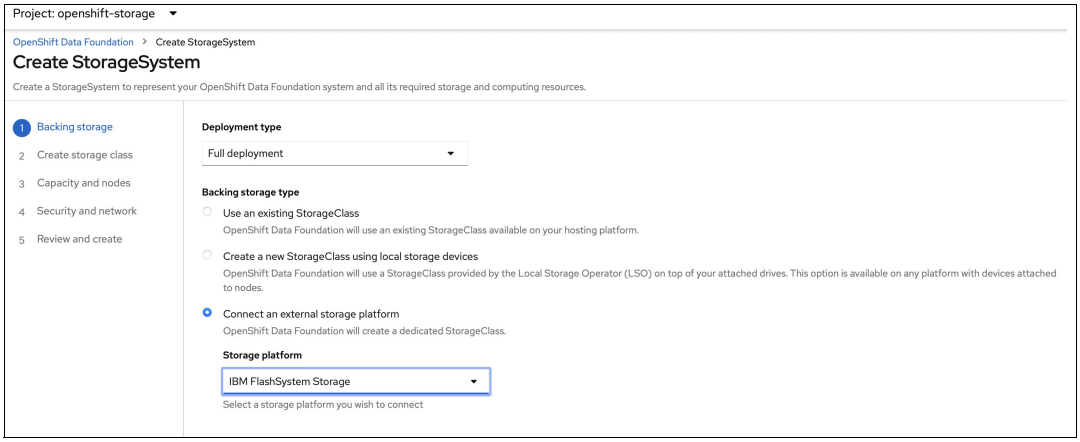


Figure 2-8 Connecting to external storage

6. In the Create storage class window, provide the name for the storage class, IP address, User name, Password, and Pool name of the IBM FlashSystem connection and select **thick** or **thin** for the Volume mode, as shown in Figure 2-9.

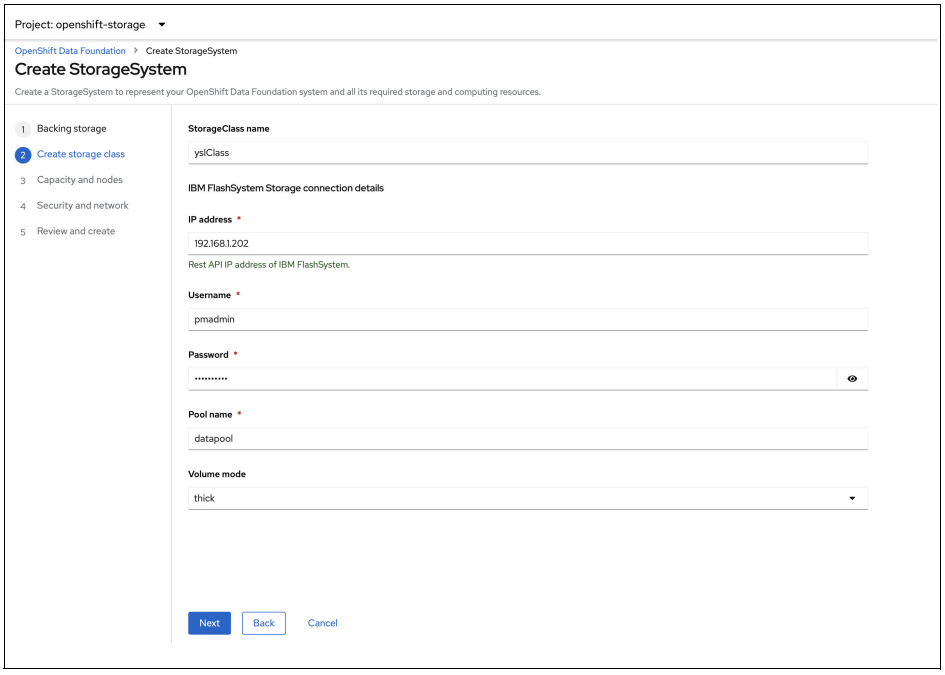


Figure 2-9 IBM FlashSystem information details

7. In the Capacity and nodes window, provide the necessary capacity details. Select at least three nodes in three different zones.
8. In the Security and network window, provide the necessary details, as shown in Figure 2-10 on page 19.

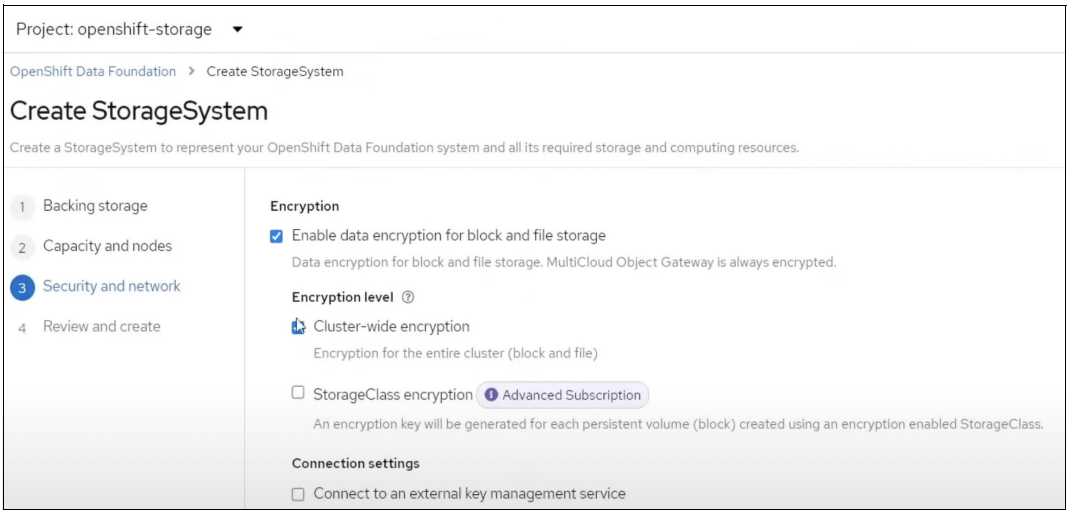


Figure 2-10 Red Hat OpenShift Data Foundation Operator security settings

a. To enable encryption, select **Enable data encryption for block and file storage**.

b. Choose any one or both encryption levels:

- **Cluster-wide encryption**
- **StorageClass encryption**

a. Select the **Connect to an external key management service** checkbox. This choice is optional for cluster-wide encryption.

i. Key Management Service Provider is set to **Vault** by default.

ii. Enter Vault Service Name, host Address of Vault server ('https://<hostname or ip>'), Port number, and Token.

iii. Expand **Advanced Settings** to enter more settings and certificate details based on your Vault configuration.

iv. Enter the Key Value secret path in the Backend Path that is dedicated and unique to Red Hat OpenShift Data Foundation.

v. Optional: Enter TLS Server Name and Vault Enterprise Namespace.

vi. Provide CA Certificate, Client Certificate, and Client Private Key by uploading the respective PEM encoded certificate file.

Note: Red Hat OpenShift Data Foundation Essentials supports device-level encryption. Red Hat OpenShift Data Foundation Advanced supports encryption at the PV level and supports volume-level encryption with BYOK.

For more information, see Chapter 2, “Deploy OpenShift Data Foundation using local storage devices”, of *Deploying OpenShift Data Foundation using IBM Power*.

2.2.5 IBM Spectrum Scale CSI driver security considerations

Here are the security best practices when using IBM Spectrum Scale CSI:

- Use separate IBM Spectrum Scale Cluster networks for metadata, data traffic, and administration.
- Ensure that only Red Hat OpenShift infrastructure nodes have access to the IBM Spectrum Scale GUI that uses the administration network. The following security parameters should be set for secure communication between the CSI plug-in and the IBM Spectrum Scale GUI:
  - **Secure SSL Mode**: Set to `true`.
  - **Certificate**: Specify a certificate authority (CA) certificate for the GUI server.
  - **User credential:** Specify IBM Spectrum Scale GUI “username” and “Password” as base64 encoded values.
- Ensure that adequate ownership is set for the PVs.
- Ensure that the pod security context is set properly.
- Use Federal Information Processing Standards (FIPS)-compliant secure data at rest that is supported by IBM Spectrum Scale by using key managers such as IBM Security Guardium® Key Lifecycle Manager.
- Encrypt all communications between the IBM Spectrum Scale clients and servers, which include data from containers or pods by setting the security mode to one of the following levels:[3]
  - **AUTHONLY**: The sending and receiving nodes authenticate each other with a TLS handshake and then close the TLS connection. Communication continues in the clear. The nodes do not encrypt transmitted data and do not check data integrity.
  - **Cipher**: The sending and receiving nodes authenticate each other with a TLS handshake. A TLS connection is established. The transmitted data is encrypted with the specified cipher and is checked for data integrity.

For more information, see *IBM Spectrum Scale CSI Driver for Container Persistent Storage*, REDP-5589.

2.3 Orchestrators and K8s

K8s is a container orchestrator platform, which orchestrates and manages the container lifecycle is an automated way, either on-premises or in the cloud.

In the era of digital transformation, enterprises are embracing modern and open-source technologies to overcome modern development issues. Due to the business demands for speed and agility in delivering products, developers adopted application modernization tools by using containerization to modernize their monolithic applications into microservices. In this transformation journey, K8s plays an important role because it offers various benefits for running and managing containerized services and applications in a cluster of nodes. K8s can be deployed across various deployment platforms, such as on-premises or in cloud, including private cloud, public cloud, and hybrid multicloud scenarios.

[3] https://www.ibm.com/docs/en/spectrum-scale/5.1.5?topic=cluster-security-mode

While enterprises are adopting K8s, architects, developers, Site Reliability Engineers (SREs), and administrators must be vigilant about one of the most important aspects of K8s, that is, its security. The K8s environment, whether it is on-premises or in the cloud, can be prone to attack if not properly hardened.

Here are some of the security loopholes that can be exploited by hackers to attack the K8s environment:

- Images in the registry are infected
- Unauthorized access to the apiserver
- Compromised cluster nodes
- Compromised containers
- Excessive or full access permission to service accounts or users
- Inappropriate usage of TLS and the firewall for the apiserver, load balancer, or Ingress Controller
- Public access to the cluster

Figure 2-11 shows some of the major attack vectors.

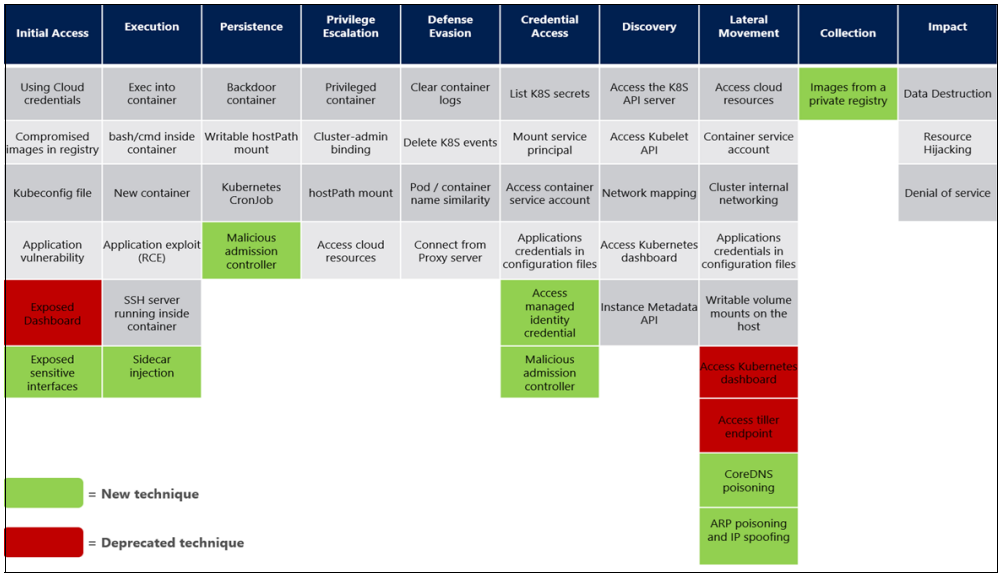


Figure 2-11 Security threat matrix

The security threat matrix that is shown in Figure 2-11 was published by Microsoft to define what it considers as the major attack vectors assembled into a threat matrix, which is discussed in this Microsoft Security Blog.

2.3.1 Security best practices for containers

K8s 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: Secure the K8s nodes. 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. For the benchmarks for the Kubernetes download, see CIS Benchmarks.
- Secure apiserver: Because all communication to the K8s containers and in the K8s cluster go through the API server, implement TLS for apiserver communication.
- Role-base access control (RBAC): Limit the access to the cluster with K8s RBAC. For more information about an RBAC setup, see 5.2, “RBAC setup for users and service accounts” on page 106.
- 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 K8s workloads in the cluster. Implement a service mesh, if appropriate.[4] Leverage side-car proxy and mutual Transport Layer Security (mTLS) for secure communication between microservices in the cluster. For more information, see this IBM Cloud document.
- 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 this Red Hat blog.
- 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 K8s up to date.
- Monitor: Set up monitoring and observability in your environment. For more information, see 7.1, “Monitoring containers and Red Hat OpenShift Container Storage security” on page 126.

2.4 Ingress Controller

Within a K8s or Red Hat OpenShift architecture, each service that represents a cluster of pod's has its own IP address to enable communication with other pods and services, but not external clients that are outside of the cluster. To enable HTTP/S access from clients, an HAProxy-based Ingress Controller should be used, which provides a rules-based highly available and load-balanced service with routing capability.

There are two aspects to the Ingress Controller:

- Ingress: Traffic (requests) inbound to a service from users.
- Egress: Traffic (responses) sent back from the service to the users.

Figure 2-12 on page 23 shows an example of a cluster with an Ingress Controller.

[4] https://www.techtarget.com/searchitoperations/definition/service-mesh

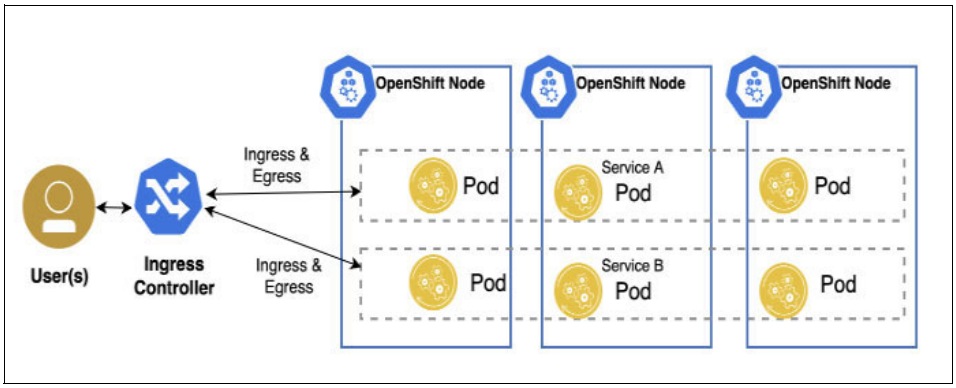


Figure 2-12 Ingress Controller with three worker nodes

The cluster has three worker nodes, and there are two services (service A and service B), each with their own IP address or DNS name that is configured to enable the Ingress Controller to provide bidirectional access to the application pod that was deployed across the three nodes. Traffic is routed to and from each multi-node pod within a service. The Ingress Controller is a critical component of the K8s and Red Hat OpenShift architecture because it enables and manages bidirectional traffic from users outside the K8s and Red Hat OpenShift architecture.

By default, the Ingress Controller replica is set to two pods and can be scaled as needed. The Ingress Controller is a reverse proxy. Without an Ingress Controller, the services (HTTP protocol only) would need to be exposed with the service type NodePort or LoadBalancer with an extra cluster external reverse proxy connecting to these node ports or load-balancers. Because the Ingress Controller can and should terminate TLS, it is a best practice to use it. Otherwise, each service that is exposed to other service types would need its own implementation of TLS termination.

2.5 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 securely 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 RBACs to manage who can pull and push each container image.
- Use private Red Hat OpenShift Container Platform registries such as Red Hat Quay, as described in 4.2.12, “Vulnerability scanners” on page 78.
- Use Portieris to enforce image security policies in IBM Cloud Container Registry. Portieris is a K8s admission controller that verifies your container images before you deploy them to your cluster in IBM Cloud Kubernetes Service. Use Portieris to enforce policies on image signatures and on vulnerabilities that are detected by Vulnerability Advisor. If an image does not meet your policy requirements, the resource that contains the pod is not deployed to your cluster.
- Integrate CI/CD pipelines and image registries with Red Hat Advanced Cluster Security for Kubernetes for continuous scanning and assurance, as described on 4.2.12, “Vulnerability scanners” on page 78.

2.6 Red Hat OpenShift on IBM Power Virtual Server

Enterprise users might discover that moving Power workloads out of a secure and “known” on-premises environment to an “unknown” virtual environment to be counterintuitive for security reasons. But, IBM Power Systems Virtual Server (IBM PowerVS) combines all the security capabilities of physical Power servers with robust IBM Cloud security capabilities, including IAM.

At the time of writing, IBM PowerVS is available in almost 15 IBM data centers around the world. IBM PowerVS fully supports mission-critical workload environments, such as:

- SAP HANA and traditional SAP workloads
- Custom AIX and IBM i applications
- Red Hat OpenShift
- IBM Cloud Pak

IBM PowerVS includes comprehensive governance compliance, including HIPAA, multiple SOC designations, General Data Protection Regulation (GDPR), and ISO 27K.

The IBM and Red Hat hybrid multicloud strategy is built on open standards that are hardened for the enterprise by combining Red Hat OpenShift Container Platform on the IBM IT infrastructure.

Existing Power customers can retain their data-intensive workloads running on-premises or in private clouds while leveraging the speed and flexibility of public cloud deployments. The move to an IBM hybrid cloud architecture is enabled by the following actions:

- Adopting Red Hat OpenShift as the open hybrid cloud platform that can orchestrate applications flexibly in any environment, such as Power servers, zSystems, x86 architectures, and across all clouds.
- Building modern scalable applications that can be deployed anywhere in the hybrid cloud with IBM Cloud Pak and Red Hat OpenShift.

The deployment of Red Hat OpenShift on IBM PowerVS is based on a Bring Your Own License model with a valid license entitlement for Red Hat OpenShift and Red Hat Linux on Power.

Here are best practices to use securely Red Hat OpenShift on IBM PowerVS.

- Use IBM Cloud Direct Link from the applications that are deployed on Red Hat OpenShift in IBM PowerVS to access IBM Cloud services. The IBM Cloud Direct Link service allows access to IBM Cloud resources by using a private network from the IBM PowerVS instance.
- To strengthen the security of cloud-based deployments, secure web applications with CA-signed certificates instead of self-signed certificates by replacing the default self-signed certificates with custom CA-signed certificates.
- Back up the Red Hat OpenShift cluster's etcd data regularly and store it in a secure location outside the Red Hat OpenShift Container Platform environment, such as IBM Cloud Object Storage with integration capabilities with IBM Cloud Key Management Services like IBM Key Protect and IBM HPCS.

2.7 IBM Cloud Paks

IBM Cloud Paks offer a simplified, enterprise-grade IBM technology stack in a containerized image. They are pre-certified and are built on Red Hat OpenShift. IBM Cloud Paks are production-ready and come bundled with management and governance tools that cater to the following services:

- Monitoring and logging
- Version upgrades and rollbacks
- Identity management
- Security and vulnerability scanning

Figure 2-13 shows some of the benefits of IBM Cloud Paks.

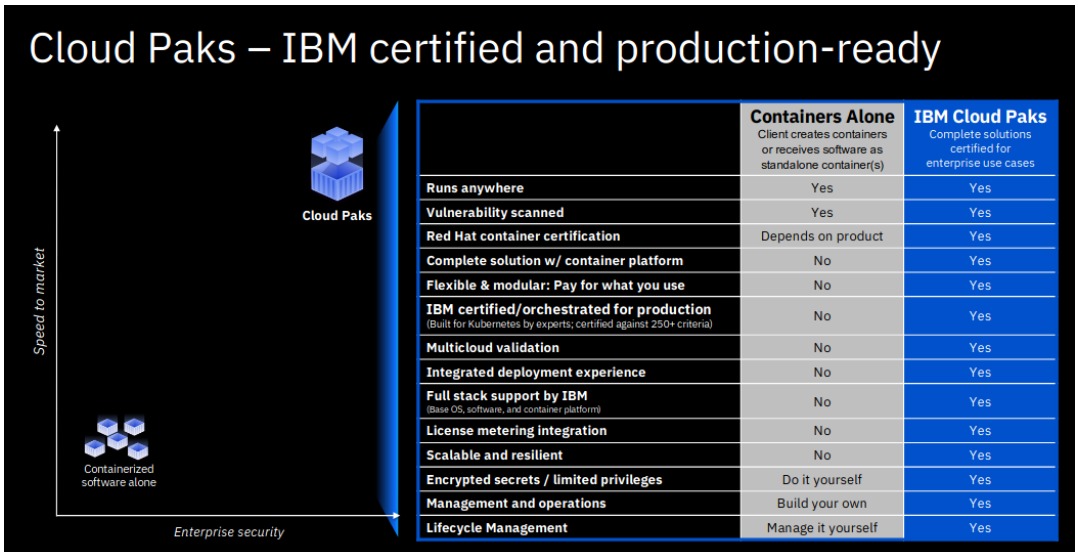


Figure 2-13 IBM Cloud Paks

IBM Cloud Paks provide a set of tools to help you build your cloud-native application environment. They also provide a high level of security for applications and data. Here are a few reasons why they do so:

- Built on Red Hat OpenShift: IBM Cloud Paks are built on top of the Red Hat OpenShift container orchestration platform, which provides a secure and robust foundation for running applications. Red Hat OpenShift includes built-in security features such as RBAC, network segmentation, and secrets management.
- Built-in security features: IBM Cloud Paks include various built-in security features, such as encryption, authentication, and access control. These features help to protect applications and data from unauthorized access and tampering.
- Compliance: IBM Cloud Paks help meet compliance requirements, such as those related to data privacy and security. IBM Cloud Paks include features that help to comply with regulations such as PCI-DSS, HIPAA, and SOC2.
- Built-in threat protection: IBM Cloud Paks include built-in threat protection features, such as firewall, intrusion detection and prevention systems, and security incident and event management (SIEM). These features are designed to protect applications and data from cyberthreats such as malware, hackers, and other types of malicious activities.
- Multi-cloud management: IBM Cloud Paks can be used to manage and secure applications across multiple clouds and on-premises environments so that you can leverage the security features that are provided by different clouds and use IBM Cloud Paks to centrally manage security policies and compliance requirements.

IBM Cloud Paks use cases

There are many IBM Cloud Paks that are available, each designed to meet a set of business requirements in your enterprise. Figure 2-14 gives an overview of some of the IBM Cloud Paks that are available and their use cases.

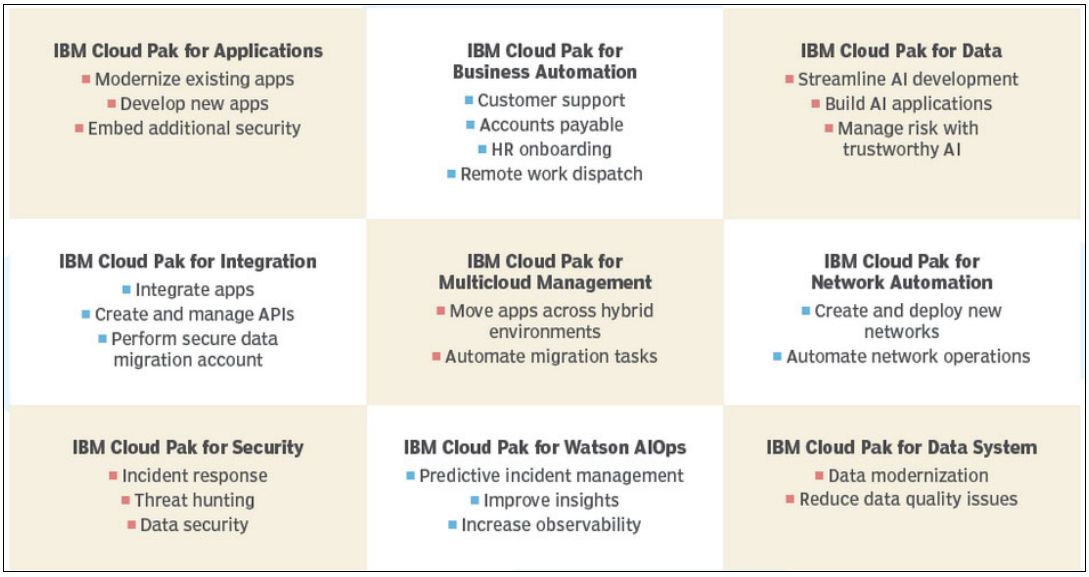


Figure 2-14 IBM Cloud Paks overview

2.7.1 IBM Cloud Pak for Applications

IBM Cloud Pak for Applications is built on the IBM WebSphere stack and Red Hat OpenShift. It is a containerized solution that provides developers with a choice of languages and frameworks to build cloud-native applications. The IBM Cloud Pak comes with best practice advice and guidance. Products and components that are included with IBM Cloud Pak for Applications are as follows:

- IBM WebSphere Application Server
- IBM Mobile Foundation
- Red Hat OpenShift Container Platform
- Red Hat Runtimes
- IBM Cloud Transformation Advisor

Figure 2-15 shows the benefits by using IBM Cloud Pak for Applications.

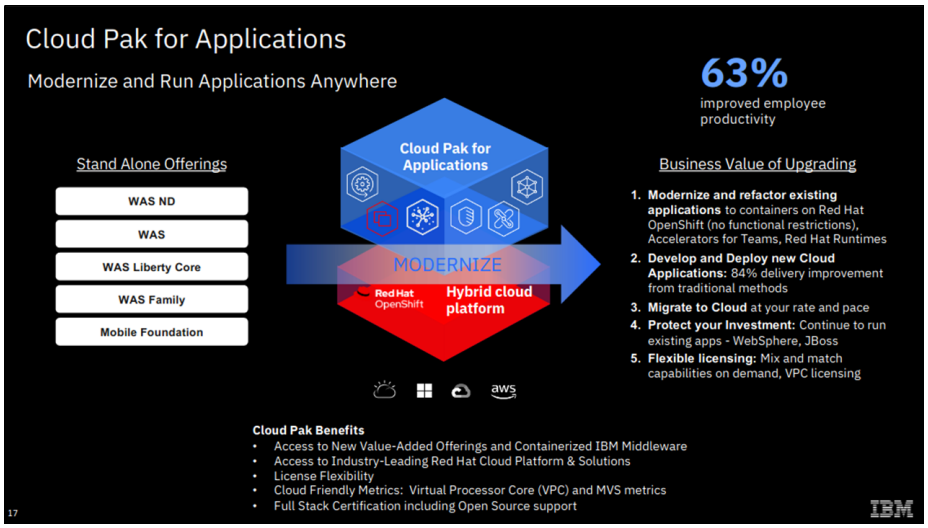


Figure 2-15 IBM Cloud Pak for Applications

Notes: IBM Cloud Pak for Applications can be installed on a supported version of Red Hat OpenShift Container Platform in any public or private cloud. It can be run on-premises behind a corporate firewall. Before installing the IBM Cloud Pak, you must have an IBM Cloud Pak License and configured a Red Hat OpenShift cluster.

For more information, see Getting started with IBM Cloud Pak for Applications.

3

Security framework and attack vectors

This chapter provides an overview of areas in your environment that can be used as entry points for entities, either internal or external, who want to instigate an attack on your applications and data. We describe a foundational framework that can be used to keep those attacks from being successful.

This chapter describes the following topics:

- Defining a threat
- Seven layer security model
- Assessing your security posture
- Layered defense approach
- Distributed application vulnerabilities
- Container vulnerabilities

3.1 Defining a threat

It is a common occurrence that different enterprises are being attacked, which results in business disruption and possibly data loss. As a result, you must act to avoid and thwart those attacks on your IT infrastructure.

According to the National Institute of Standards and Technology (NIST), a *threat* is defined as follows:[1]

“Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service. Also, the potential for a threat-source to successfully exploit a particular information system vulnerability.”

This chapter helps you understand areas of your IT infrastructure that are vulnerable and provide a framework to help you build in the proper protections in your applications and operations. Having a valid and complete inventory of your IT assets facilitates your planning process.

3.2 Seven layer security model

There have been many reports of data breaches and cyberattacks over several years. Security practices, countermeasures, tools, and methodologies are getting better and more sophisticated, but so are the techniques of cyberattackers. The attacks continue unabated even in 2023, the year that this book was written.

In October 2022, hackers released data from the Los Angeles Unified School District. In September 2022, Optus, an Australian telecoms company with 9+ million subscribers, suffered a data breach that affected 1.2 million of their customers. In August 2022, Greece's largest natural gas distributor, DESFA, suffered a limited scope data breach and system outage that was caused by a cyberattack. For more information about these cyberattacks, see Data Breaches That Have Happened in 2022 and 2023 So Far.

This sample is a tiny one of cyberattacks and data breaches. From the geographical distribution of these attacks, it is clear that the problem is worldwide. Additionally, with the paradigm shift following the COVID-19 pandemic where more business and activities were conducted digitally, there is a larger attack surface with the potential of being subject to malicious cyberactivity.

Understanding the 7 layers of security, as shown in Figure 3-1 on page 31, helps organizations to construct a multi-faceted approach when developing defense plans and measures to keep their systems safe. The following sections describe each of these layers.

Mission-critical assets

A *mission-critical asset* is anything without which a business cannot survive. The challenge is that each business must tailor this layer to their own business. What is critical for one business might not be critical for another business. When the assets at this layer are correctly identified, an organization can work backwards through the other six layers to build a security policy that comprehensively protects what they consider most precious.

[1] https://csrc.nist.gov/glossary/term/threat

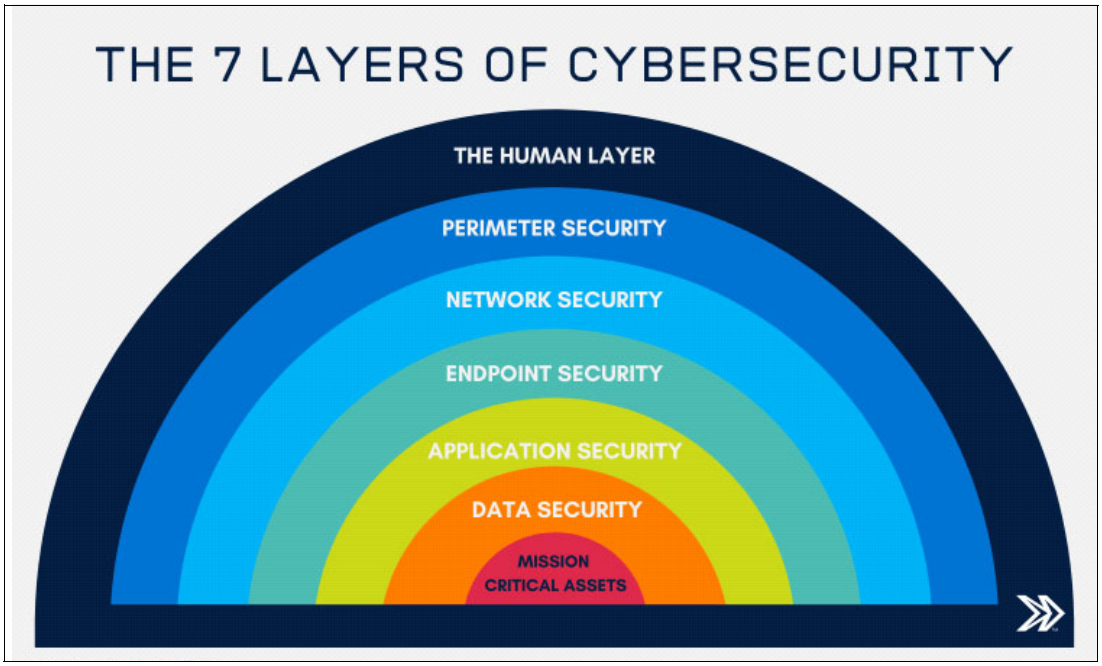


Figure 3-1 Seven layers of cybersecurity

Data security

This layer is about security controls to protect storage, transfer, backups, and data loss. The IBM Security Guardium Portfolio enables security teams to protect data at rest and in transit. For more information about IBM Security Guardium, see *IBM Power Systems Cloud Security Guide: Protect IT Infrastructure In All Layers*, REDP-5659 and IBM Security Guardium.

Application security

Web applications are the main target of hackers, and security at this level should clearly define whether an application is internally or externally facing. Security measures should account for these factors, and address access to the applications and how the applications access data.

Endpoint security

It is increasingly common for organizations to allow and encourage employees to install authentication apps on their personal mobile devices so that they can use them for multi-factor authentication (MFA) to corporate systems. Therefore, it is important that there are controls in place to protect the connections between user devices and enterprise systems. Security should be robust enough to make sure that user devices cannot be exploited to breach corporate systems and vice versa.

Network security

The goal here is to prevent unauthorized access to a business's network. Security should not stop after a legitimate user gains access to the network. Network security policies should continue to ensure that the legitimate user can access only what they are meant to access. Keeping current with security patches is vital in keeping networks protected. Offerings such as IBM Cloud Pak for Security[2] can form part of the arsenal in keeping networks secure.

[2] https://cloud.ibm.com/docs/cloud-pak-security?topic=cloud-pak-security-getting-started

Perimeter security

This outer layer of the network is the point at which all devices access company data. The outer layer does not necessarily stop at the office building, the data center, or even the city or country in which the business is located. With Internet of Things devices, such as railway wayside instruments, and other remote monitoring equipment, the perimeter is now global.

The first step in keeping this layer safe is to understand what comprises the perimeter and catalog and identify the groups of devices that connect to the network.

The human layer

In the latest Privacy Incident Benchmark Report (PIBR) for 2022 from RadarFirst,[3] it is made clear that unintentional human error is still the biggest cause of privacy data reaches. As shown in Figure 3-2, the report goes on to highlight that 95% of data privacy breaches are caused by human error. A further 3% of privacy breaches are caused by people that snoop to get unauthorized access to personal data but with no intent to cause harm to the enterprise. According to the report, only approximately 2% of all 55,000 assessments from 150 jurisdictions was malicious.

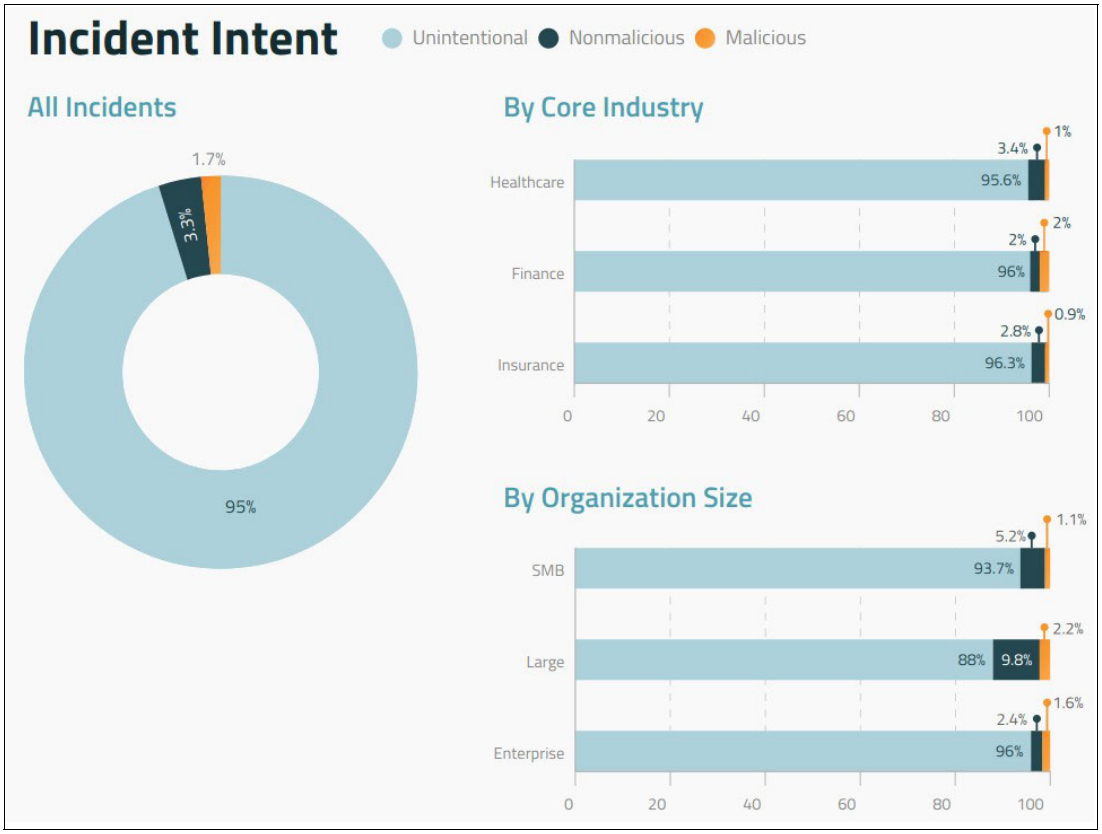


Figure 3-2 Privacy Incident Benchmark Report: human error impact

Education and training of personnel are some of the best ways to reduce the dangers that are posed by the human factor. In most medium and large organizations, training in cybersecurity is mandatory as part of the employee recertification process. It is common across organizations that employees and consultants take an annual re-certification on cybersecurity training.

[3] https://www.radarfirst.com/resources/2022-privacy-incident-benchmark-report/

Note: The PIBR is an annual publication by RadarFirst that collates and summarizes the risks of harm to individuals through the exposure of sensitive data. For more information, see The RadarFirst Story.

3.3 Assessing your security posture

Security posture is a measure of an organization's overall cybersecurity status. It is a measure of how vulnerable an organization is to cyberattacks or data breaches. Another important facet of security posture is how an organization reacts to cyberattacks and threats.

Figure 3-3 shows some of the various components of an organization's security posture. The following sections summarize what the diagram represents. For more information and a detailed explanation, see What is Security Posture.

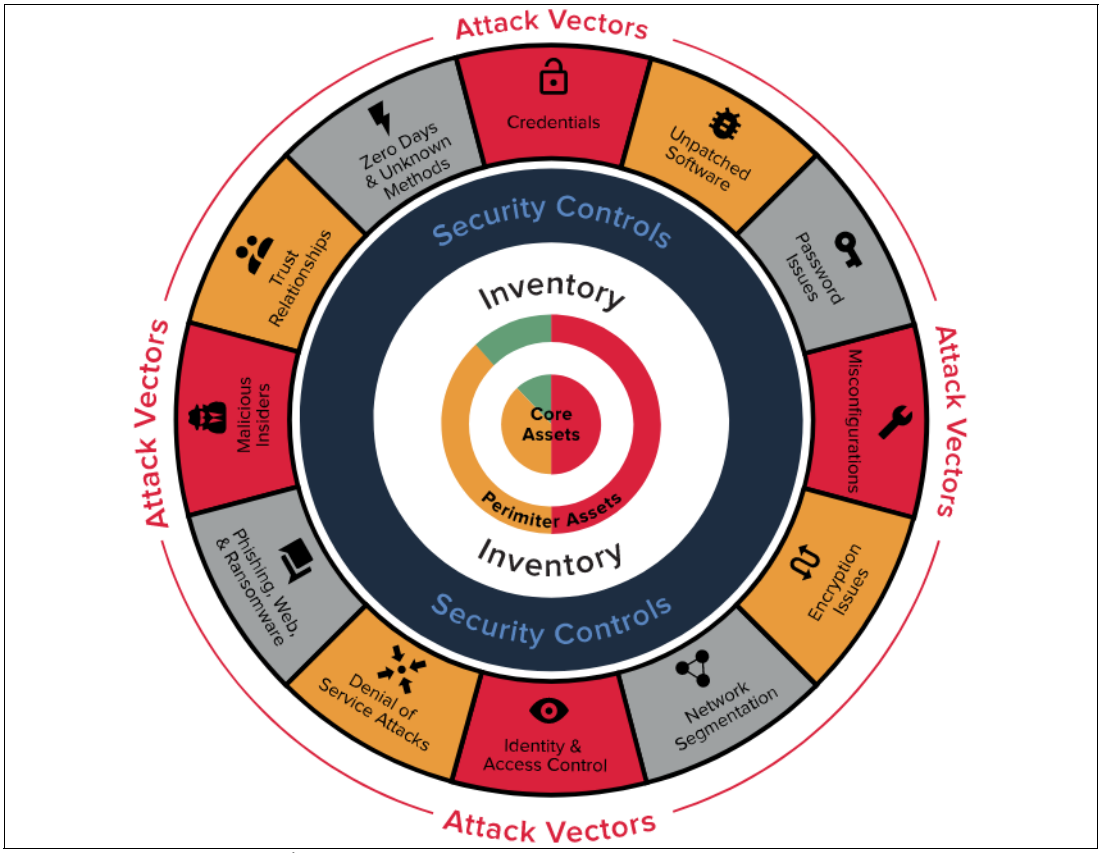


Figure 3-3 Attack vectors[4]

Inventory of assets

As mentioned in 3.2, "Seven layer security model" on page 30, at the core of establishing a security posture is having an accurate catalog of all IT assets and using it as a foundation for building a security posture.

[4] Source: https://www.balbix.com

Security controls

Security controls are about evaluating and understanding the efficacy of security controls that have been and will be implemented. Some controls prevent attacks, others detect attacks, and others are designed to help you recover from attacks.

Attack vectors

Attack vectors are the methods that attackers use to intrude into the network.

Attack surface

An attack surface is the amalgamation of asset inventories and attack vectors. The attack surface is a representation of all the ways attackers attempt to gain unauthorized access to the network.

3.4 Layered defense approach

Because IT security risks can occur at different levels, security measures must be put in place to provide a layered defense approach. Using a layered approach means that an attacker who gets through one layer of defense will be obstructed and blocked by the next layer.

In 3.2, “Seven layer security model” on page 30, we outlined the main components of a layered defense approach. System-level security is the last line of defense against attacks that originate from the internet. As such, be careful when building system-level security.

The IBM Power10 family of servers comes with a number of features, which help in building a strategy for a layered defense approach. Here are two of the most relevant features:

- Side-channel mitigation performance
- End-to-end encryption

Side-channel mitigation performance

Side-channel attacks (such as Spectre and Meltdown) allow unauthorized read access by malicious processes to the contents of protected kernel or host memory. CPU vendors introduced several features to protect against these kinds of attacks, but they can cause performance degradation in some cases. The Power10 processor is designed to improve side-channel mitigation performance.

End-to-end encryption

The Power10 processor supports Fully Homomorphic Encryption (FHE) and quantum-safe cryptography.

Note: For more information about how Power10 servers can be used as part of a multi-layered defense approach, see A multilayered approach to security with IBM Power.

3.5 Distributed application vulnerabilities

A *distributed application* in the context of Red Hat OpenShift is a software application that is designed to run on multiple regions and servers, connected over multiple networks, and deployed and managed by using the Red Hat OpenShift platform.

Red Hat OpenShift is a cloud-based platform for developing and deploying containerized applications. It provides several features and tools to help developers build, deploy, and manage distributed applications:

- Container orchestration: Red Hat OpenShift uses Kubernetes (K8s) to manage and orchestrate containers, which allows developers to easily scale and deploy their applications across multiple nodes in a cluster.
- Auto-scaling: Red Hat OpenShift automatically can scale applications based on demand, which helps ensure that applications have the resources that they need to meet user demand.
- Deployment pipelines: Red Hat OpenShift provides a deployment pipeline feature that allows developers to automate the build, test, and deployment process for their applications.
- Monitoring and logging: Red Hat OpenShift provides built-in monitoring and logging capabilities to help developers track the performance and health of their applications.
- Security: Red Hat OpenShift provides several security features, including network policies, pod security policies, and image scanning to help secure distributed applications.

By using Red Hat OpenShift, developers can easily build, deploy, and manage distributed applications in a secure and scalable manner. Distributed applications can be vulnerable to a range of security threats, including network-based attacks, data breaches, and malicious actors attempting to exploit vulnerabilities in the application itself. Common vulnerabilities in distributed applications include weak authentication and access controls, lack of input validation, and poor exception handling. For more information, see 3.5.4, “Common vulnerabilities affecting distributed applications” on page 37.

3.5.1 Security challenges in a microservices architecture

A microservices architecture provides a methodology to design and build software applications as a set of small, independent services that communicate with each other through well-defined interfaces. This approach has several benefits, including improved scalability, flexibility, and maintainability.

However, a microservices architecture also introduces some security challenges that must be addressed:

- Complexity: With microservices, it is more difficult to secure the entire system as a whole because it is divided into multiple, independent services, which can make it harder to identify and address vulnerabilities or threats.
- Communication: Microservices communicate with each other through application programming interfaces (APIs), which can increase the attack surface and make it easier for attackers to exploit vulnerabilities. It is important to secure the communication between microservices with measures such as encryption and authentication.
- Visibility: It can be challenging to monitor and detect threats or vulnerabilities in a microservices environment because of the distributed nature of the system.
- Access control: With microservices, it is important to implement granular access controls to ensure that only authorized users and services have access to specific resources.

To address these challenges, it is important to implement a robust security strategy that includes measures such as encryption, authentication, access control, and monitoring. It is also important to regularly test and assess the security of the microservices environment to ensure that it is adequately protected.

3.5.2 Understanding multi-region active-active architecture

A multi-region active-active architecture is a type of distributed application architecture that is designed to provide high availability and scalability by running in multiple regions simultaneously.

In this architecture, the application is deployed in multiple regions, and users can access the application from any of the regions. The application is configured to synchronize data between regions in real time, which allows users to access the same data regardless of which region from which they are accessing the application.

The benefits of a multi-region active-active architecture include the following ones:

- High availability: By running in multiple regions, the application is less likely to experience outages due to regional failures or maintenance.
- Scalability: The application can scale horizontally by adding more regions and vertically by adding more resources in each region.
- Improved performance: By running in multiple regions, the application can be closer to users, which can improve performance and reduce latency.
- Disaster recovery: If a region experiences a disaster, users can still access the application from other regions.

However, building and maintaining a multi-region active-active architecture can be complex because it requires coordinating data synchronization between regions and managing multiple instances of the application. It is typically more expensive than other types of architectures due to the extra infrastructure and resources that are required.

3.5.3 Requirements for a multi-region active-active architecture

There are several requirements that must be considered when building a multi-region active-active architecture:

- Data synchronization: One of the main challenges of a multi-region active-active architecture is ensuring that data is synchronized between regions in real time. This task requires a reliable and efficient data synchronization mechanism, such as a database replication or message queuing system.
- Network connectivity: To ensure that users can access the application from any region, the application must communicate with other regions over a reliable and high-bandwidth network.
- Load-balancing: To distribute traffic evenly across regions, the application should use a load-balancing mechanism, such as a global load-balancer or traffic manager to route users to the appropriate region.
- Disaster recovery: The application should be able to handle regional failures or disasters, such as data centers going offline or natural disasters. This task might require implementing backup systems and processes, such as redundant data centers or data replication across regions.

- Security: To ensure the security of the application and its data, the application should use secure communication channels and implement strong authentication and authorization mechanisms. It should follow general security best practices, such as using strong passwords and enforcing least privilege principles.
- Compliance: Depending on the industry and location of the application, it might be subject to various compliance requirements, such as data privacy regulations. Ensure that the application meets these requirements, which might require implementing extra controls and processes.

By considering these requirements and implementing appropriate mechanisms and controls, you can build a robust and scalable multi-region active-active architecture.

3.5.4 Common vulnerabilities affecting distributed applications

There are several common vulnerabilities that can affect distributed applications:

- Injection attacks: These attacks involve injecting malicious code or data into the application, which can be used to run unauthorized actions or access sensitive data. Examples include SQL injection and cross-site scripting (XSS) attacks.
- Broken authentication and authorization: Weak or improperly implemented authentication and authorization mechanisms can allow unauthorized users to gain access to the application or its data.
- Cross-site request forgery (CSRF): This vulnerability allows an attacker to trick a user into making a request to the application on the attacker's behalf, potentially allowing the attacker to run unauthorized actions. In the context of Red Hat OpenShift, a CSRF vulnerability might allow an attacker to perform actions on behalf of a user without the user's knowledge or consent. For example, an attacker might use a CSRF attack to create, modify, or delete resources within the Red Hat OpenShift environment.
- Insecure communication channels: If communication between distributed components is not encrypted, it can be intercepted and tampered with, potentially allowing an attacker to access sensitive data or run unauthorized actions.
- Improper error handling: If the application does not properly handle errors, it might reveal sensitive information, such as stack traces or database details, which can be used by an attacker to exploit the application.
- Lack of input validation: If the application does not properly validate input, it might be vulnerable to injection attacks or other types of malicious input.
- Lack of security patches: If the application and its components are not kept up to date with the latest security patches, it might be vulnerable to known vulnerabilities that are addressed in newer versions.

3.5.5 Best practices for securing distributed applications in Red Hat OpenShift

Here are best practices to secure distributed applications in Red Hat OpenShift:

- Use secure communication channels: Ensure that all communication between distributed components is encrypted to prevent eavesdropping and tampering:
  - Use Transport Layer Security (TLS) to encrypt communication between components. Red Hat OpenShift includes a built-in certificate authority (CA) to issue certificates to components and secure communication between them.
  - Enable mutual Transport Layer Security (mTLS) to require all components to present a valid certificate before they can communicate with each other.
  - Use network segmentation and firewall rules to restrict communication between components to only the necessary ports and protocols.
  - Use an external load balancer or Ingress Controller to terminate TLS connections and forward traffic to the appropriate component.
- Validate input: Validate all input to prevent malicious data from being processed by the application. This action includes sanitizing user input, verifying the authenticity of data that is received from other components, and verifying that data meets the expected format and constraints.
- Use secure authentication and authorization: Implement secure authentication and authorization mechanisms to ensure that only authorized users and components can access the application and its data. Red Hat OpenShift provides several authentication and authorization options, such as OAuth, OpenID Connect (OIDC), and LDAP, which can be used to secure your application.
- Implement proper error handling: Proper error handling can help prevent vulnerabilities by ensuring that the application gracefully handles unexpected input or errors and does not reveal sensitive information.
- Regularly update and patch: Keep all components of the distributed application up to date with the latest security patches to fix known vulnerabilities. Red Hat OpenShift provides automatic updates and patching capabilities to help you keep your application secure.
- Use security testing tools: Regularly use security testing tools, such as penetration testing and vulnerability scanners to identify and address vulnerabilities in the application.
- Enable role-based access control (RBAC): Define which users and groups can access specific components and resources within Red Hat OpenShift.
- Use Pod Security Policies: Define the security context for pods and containers in Red Hat OpenShift.
- Enable auditing and logging: Track and monitor access to components and resources within Red Hat OpenShift.
- Implement security best practices: Follow general security best practices, such as using strong passwords, disabling unnecessary services, and enforcing least privilege principles to help prevent vulnerabilities in the application.

In addition to these best practices, you also can use Red Hat OpenShift security features, such as network policies, pod security policies, and image scanning to further secure your application.

3.6 Container vulnerabilities

Container technology transformed dramatically the way that many organizations run their businesses. With containers, an enterprise (whether large or small) can now leverage the usage of containers so that the enterprise can become more agile and deliver solutions and benefits to their customers more quickly. For developers working within the enterprise, they can test their solutions end-to-end by deploying firewalls, LDAP instances, databases, and Enterprise Service Buses (ESBs) within containers running on their own laptops before deploying to a shared integration environment. This kind of agility and flexibility means that performance issues, bugs, and bottlenecks are identified and captured more quickly, which enables teams to deliver more robust solutions to the customer in a shortened period.

Figure 3-4 shows some of the main benefits of containerization.

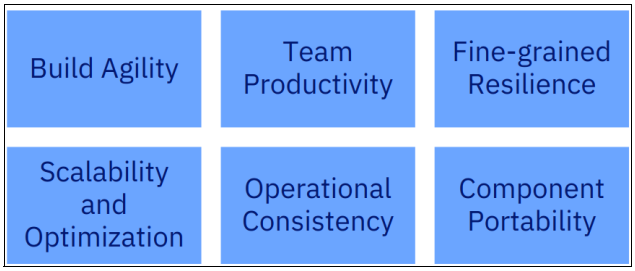


Figure 3-4 Containerization benefits

3.6.1 Recent security breaches

There are security risks and vulnerabilities that are associated with most new technologies, which does not mean that we should avoid containerization; it means that we must be aware of the risks and implement security best practices.

There have been several documented container and K8s attacks. One example[5] was how attackers exploited a misconfigured Docker API to run crypto mining software. The malware was designed to escape from the container to the host and then spread to other containers and hosts. Crypto currency mining requires the purchase of expensive hardware or GPUs, which consume a great deal of electricity. The creators of this malware effectively had unlimited use of electricity and expensive mining hardware to mine crypto currencies for profit.

[5] https://blog.aquasec.com/threat-alert-kinsing-malware-container-vulnerability

Figure 3-5 shows how this attack was done.

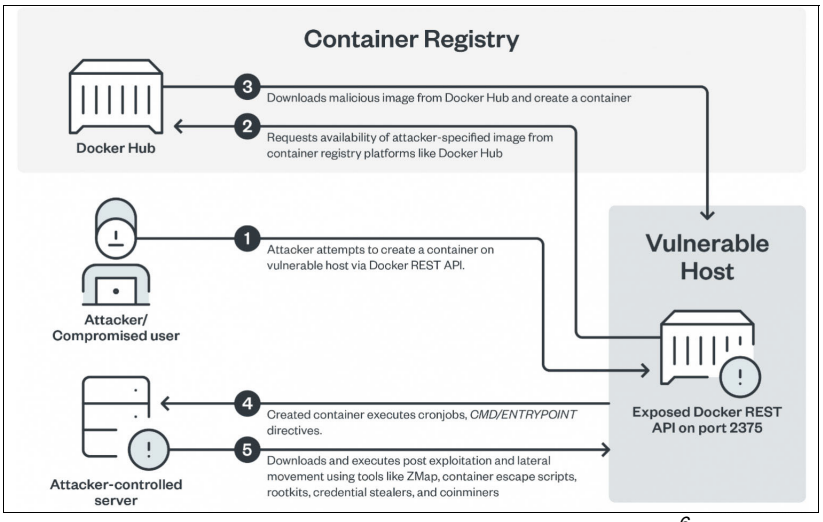


Figure 3-5 Hacker attack on a poorly configured Docker server[6]

3.6.2 Risks, vulnerabilities, and mitigation steps

Just as Docker and K8s became the standard for containerization and orchestration, there was a document that was published by NIST in September 2017[7] that laid out the major risks for container technologies and countermeasures for these risks.

The Docker and K8s partnership: In 2017 at DockerCon Europe 2017, Docker announced that they would support K8s. This announcement signaled that Docker was becoming a mature technology that provided enterprises with a choice of using either K8s or Swarm to orchestrate and coordinate containers and services.

Some of the risks, vulnerabilities, and associated countermeasures that were laid out in the publication are highlighted below:

- Image vulnerabilities: An image can be up to date with security patches at the time that the image is released but then falls behind as new vulnerabilities are discovered. Rather than using traditional vulnerability management tools, organizations should use container-specific vulnerability management tools. Traditional tools might not be able to detect accurately vulnerabilities within containers.
- Image configuration defects: An image might not have been correctly configured with the principle of “least privilege”, so a container with fully up-to-date security patches might still be vulnerable to an attack.
- Embedded malware: Malware might be purposefully, accidentally, or inadvertently added to a container image. Organizations must have measures and processes in place to ensure that images are continuously monitored for embedded malware.
- Unbounded network access from containers: The default configuration in most container run times is to allow containers to communicate with each other and the host. A compromised container might put other resources on the network at risk. Organizations must control and monitor network traffic that is sent from containers.

[6] https://www.bleepingcomputer.com/news/security/teamtnt-hackers-target-your-poorly-configured-docker-servers/

[7] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf

- Insecure container runtime configurations: A container running in privileged mode has access to all devices, resources, and containers that are running on the host. Organizations should have a robust automated process in place for ensuring compliance with security best practice standards.
- Large attack surface: General-purpose operating systems contain many libraries to allow them to be used for various applications. Container-specific operating systems are designed with fewer libraries and a smaller attack surface.

Interestingly, most of the countermeasures that were highlighted in 2017 are still the measures that are currently proposed by the Open Web Application Security Project (OWASP) in their OWASP Docker Security Cheat Sheet Rules:

- Rule #0 - Keep Host and Docker up to date.
- Rule #1 - Do not expose the Docker daemon socket.
- Rule #2 - Configure a container to use an unprivileged user.
- Rule #3 - Grant only specific capabilities that are needed by a container.
- Rule #4 - Run Docker images with the **`no-new-privileges`** flag to prevent an escalation of privileges.
- Rule #5 - Disable inter-container communication.
- Rule #6 - Use Linux Security Module.
- Rule #7 - Limit machine resources.
- Rule #8 - Run containers with a read-only file system.
- Rule #9 - Use tools to detect container vulnerabilities, secrets in images, and misconfiguration in K8s and Docker.
- Rule #10 - Set the Docker daemon with a logging level of at least `info`.
- Rule #11 - Use Lint on the Dockerfile[8] at build time (Lint, or a linter, is a static code analysis tool that is used to flag programming errors, bugs, stylistic errors, and suspicious constructs).

Tip: The OWASP Cheat Sheet Series was created to provide a concise collection of high-value information about specific application security topics. These cheat sheets were created by various application security professionals who have expertise in specific topics.

[8] https://en.wikipedia.org/wiki/Lint_(software)

4

Designing and implementing Red Hat OpenShift with security first

Containerized applications ensure consistency of development, testing, and deployment across physical servers, virtual machines (VMs), and public and private clouds.

This chapter describes the key elements to securing a Red Hat OpenShift environment. These elements encompass each layer of the container's solution stack from before you deploy and run your container to the lifecycle of the containerized application after it is placed into production.

This chapter describes the following topics:

- Approach to making Red Hat OpenShift secure by design
- Securing Red Hat OpenShift building blocks

4.1 Approach to making Red Hat OpenShift secure by design

This section describes the key elements to securing a Red Hat OpenShift environment. These elements encompass each layer of the container's solution stack starting before you deploy and run your container and through the lifecycle of the containerized application after it is placed into production.

This section describes the best practices for securing containerized applications. The following environments are addressed:

- Container Host OS, IBM PowerVM Hypervisor, and multi-tenancy
- Red Hat OpenShift trusted sources
- Red Hat OpenShift secure container orchestration
- Red Hat OpenShift deployment on IBM Power Systems Virtual Server
- Red Hat OpenShift build process security
- Red Hat OpenShift deployment process security
- Network isolation and API endpoint security
- Security consideration for federation of containerized applications

4.1.1 Container Host OS, IBM PowerVM Hypervisor, and multi-tenancy

Secure the shared OS that hosts all the containers and containerized applications:

- The host operating system kernel should be able to secure the host kernel from container vulnerabilities.
- The host operating system should be able to isolate containers and keep them from interacting with each other.

Red Hat Enterprise Linux (RHEL) and Red Hat OpenShift are positioned to secure the Linux operating system through Red Hat exclusive Security Enhanced Linux (SELinux):

- Red Hat SELinux isolates containers from the host kernel, and containers from each other.
- Administrators should enforce mandatory access controls (MACs) for every user, application, process, or file.
- Control groups (cgroups) should place limitations on the resources that a container may consume from the host system.
- Namespaces provide a level of abstraction inside a container to make an application appear as though it is running its own operating system inside the container with its own dedicated allocation of resources from the global pool.

Cloud-native security complements the security that is provided as standard to VMs by the Power10 infrastructure. IBM Power10 processors enable the following features:

- Cryptographic performance acceleration
- Main memory encryption
- Performance-enhanced side-channel avoidance
- Protection against service processor vulnerabilities
- Defense against application vulnerabilities (Return-Oriented Programming (ROP) Protection)

4.1.2 Red Hat OpenShift trusted sources

Ensure that third-party container registry images will not compromise an existing infrastructure or contaminate other containers that are running in the same environment:

- Check whether the application layer of the container has known vulnerabilities.
- Check how frequently the container image is updated.
- Check the author's container image updates.

Use Red Hat packaged certified containers for the following actions:

- Use the Red Hat Quay registry to validate certified-secure container images, which are ready for use with Red Hat OpenShift.
- Red Hat Container Health Index scores and rates any container for security and vulnerabilities.
- Red Hat Security Advisories alert you to any newly discovered vulnerabilities in certified container images, with pointers to the updated images so that you can secure your application layer.
- You can use your own container scanning tools to check for vulnerabilities. You also can leverage thee RHEL and Red Hat OpenShift pluggable application programming interface (API) to integrate scanners such as OpenSCAP[1] with your continuous integration and continuous delivery (CI/CD) pipeline.

4.1.3 Red Hat OpenShift secure container orchestration

Modern microservices-based applications are made possible because of orchestration services like Kubernetes (K8s), which handle the complexities of deploying multiple containerized applications across distributed hosts or nodes. Ensure that the installation, deployment, hardening, and operations of orchestrators follow best practices.

Red Hat OpenShift is designed around K8s to deliver container orchestration, scheduling automation, and application management at the scale that is needed for the enterprise.

Red Hat OpenShift adds enterprise security options that often are missing from open-source K8s distributions:

- Access to the master nodes uses Transport Layer Security (TLS).
- Apiserver access is based on X.509 certificates or OAuth access tokens.
- The keystore `etcd` is no longer exposed directly to the cluster. Using encryption provides an extra layer of security. Red Hat OpenShift apiserver and K8s apiserver resources, Secrets, routes, OAuth access, and authorize tokens also should be encrypted.
- Runs on Red Hat exclusive Security Enhanced Linux (SELinux).

4.1.4 Red Hat OpenShift deployment on IBM Power Systems Virtual Server

IBM Power servers help clients respond faster to business demands, protect data from core to cloud, and streamline insights and automation while maximizing reliability in a sustainable way. IBM Power Systems Virtual Server (IBM PowerVS) can modernize applications and infrastructure with a frictionless hybrid cloud experience to provide the agility that companies need.

[1] https://www.open-scap.org/

The IBM and Red Hat hybrid multi-cloud strategy is built on open standards that are hardened for the enterprise. This strategy combines Red Hat OpenShift Container Platform with the IBM infrastructure.

By deploying Red Hat OpenShift on IBM PowerVS in IBM Cloud, you gain the following benefits:

- IBM Cloud, which is the most open and secure public cloud for business.
- Red Hat OpenShift, which is the most secure K8s platform for operationalizing container workloads remotely or as a hosted service.
- IBM Power, which is the most reliable mainstream server platform for innovation. You can use it to get to market faster with comprehensive end-to-end security at every layer of the stack.
- The highest level of encryption, which is Federal Information Processing Standards (FIPS) 140-2 Level 4.
- Isolation for cloud-native Red Hat OpenShift Kubernetes Service (ROKS) and containers on IBM Power servers.
- Integrated infrastructure as a service (IaaS) and platform as a service (PaaS) enhanced availability service-level agreements (SLAs) with a high availability of 99.99%.
- Security leadership:
  - Highest compliance for data encryption
  - Configurable so that even IBM cannot see customer data
  - Edge-to-cloud threat management with IBM security integration

4.1.5 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 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)[2] 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 assemble automatically 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.[3]

Ultimately, this approach of securing the software build process allows operations teams to manage base images, architects to manage middleware and software that needed by the application layer, and developers to focus on writing better code.

[2] https://docs.openshift.com/container-platform/3.11/creating_images/s2i.html
[3] https://www.ibm.com/docs/en/rbd/9.7?topic=application-rational-appscan

4.1.6 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)[4] 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:

- Running of privileged containers.
- Capabilities that a running container may request.
- Allow or deny access to volumes.
- Container user ID.
- Security Enhanced Linux (SELinux) context of the container.

4.1.7 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)[5] 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 (OIDC)-based authentication.

4.1.8 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 K8s 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 K8s pods (groups of containers) have consistent IP addresses and port ranges that are assigned to them.

[4] https://docs.openshift.com/container-platform/4.10/authentication/managing-security-context-constraints.html#security-context-constraints-about_configuring-internal-oauth

[5] https://docs.openshift.com/container-platform/3.11/architecture/networking/sdn.html

4.2 Securing Red Hat OpenShift building blocks

In this section, we describe best practices to secure all Red Hat OpenShift environment building blocks.

4.2.1 Hardware

IBM Power10 servers protect sensitive data by leveraging the latest pervasive encryption capabilities across hybrid cloud deployments. These enhancements introduce full memory encryption at scale, and provide end-to-end encryption without affecting performance.

Also, workloads on Power10 servers benefit from cryptographic algorithm acceleration so that algorithms like AES, SHA2, and SHA3 run faster on Power10 servers than they did on POWER9 processor-based systems (on a per-core basis). This performance acceleration allows features like Volume Encryption to be turned on with low performance impact.

To be prepared for the quantum era, Power10 servers are built to support efficiently upcoming cryptography techniques such as Quantum-safe Cryptography and Fully Homomorphic Encryption (FHE). Quantum-safe Cryptography refers to the efforts to identify algorithms that are resistant to attacks by both classical and quantum computers in preparation for the time when large-scale quantum computers are built.

Power10 servers offer also secure workload isolation in cloud deployments with integrity that is engineered into every layer of the system. All components of the stack are fully integrated and co-optimized, and they are provided from IBM as a single vendor, which makes it more secure.

IBM PowerVM, the built-in hypervisor, has an outstanding track record, with orders of magnitude fewer vulnerabilities than competitive x86 hypervisors. Power10 servers have advanced firmware integrity with extra measures to isolate the CPU from service processors for better defense against attacks on management systems.

Power10 servers introduce innovations to address emerging threats, with extra features and enhancements to defend against application domain vulnerabilities, such as ROP attacks (a security exploit technique that is used by attackers to run code on a target system).

Figure 4-1 on page 49 shows how cloud-native security complements the security that is provided as standard to logical partitions (LPARs) by the Power10 infrastructure, which provides more depth of defense across the infrastructure, VMs, and workloads.

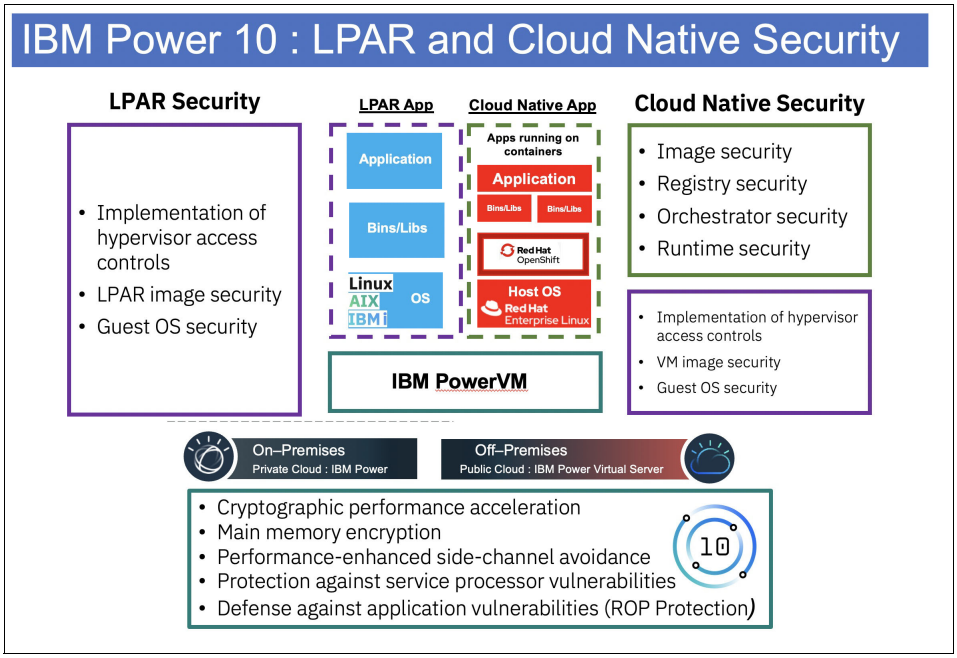


Figure 4-1 IBM Power10 LPAR and cloud-native security

4.2.2 Networking

As we describe networking in K8s, it can be applied to Red Hat OpenShift because it is an implementation of K8s.

A K8s cluster is composed of a master node and several worker nodes, each of which are virtual or physical machines, which all have their own IP addresses. When applications are deployed on this cluster in pods, each pod is assigned an IP address. Each pod can be running different applications. For example, a pod might be running a database server while another pod is running a web server, as shown in Figure 4-2.

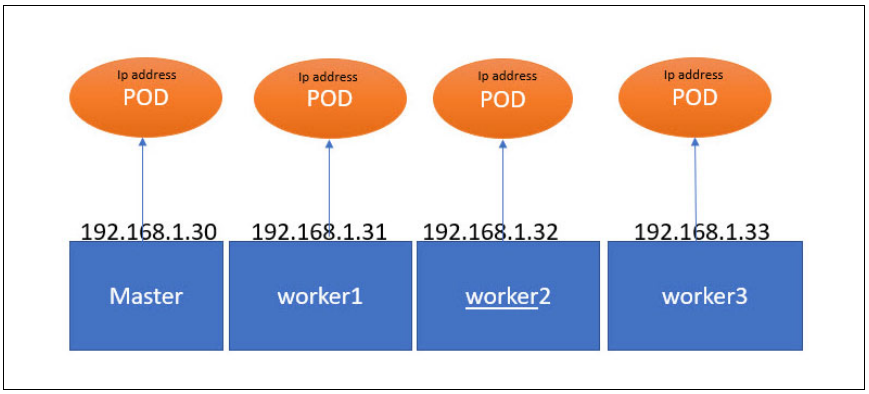


Figure 4-2 Example network configuration

The pods must communicate with each other, so they must be on a network that is configured for that communication, and each pod must have a unique IP address. In Red Hat OpenShift, the SDN is responsible for assigning and making sure that each pod has a unique IP address and ensure that routes are in place to direct traffic between worker nodes. The Red Hat OpenShift SDN creates a virtual network that spans across multiple nodes in the cluster. This virtual network is called an *overlay network*, and it is created by using the Open vSwitch standard.[6]

Note: Open vSwitch is a distributed virtual switch that is used to interconnect VMs in a hypervisor. Some of it is features include VLAN tagging, trunking, LACP, and port mirroring.

In K8s, the smallest unit is a *pod*. One pod always contains at least one main container. Every pod has a unique IP address, and the IP address is reachable from all other pods in the K8S cluster.

One challenge in a distributed infrastructure with multiple servers is allocating ports to services and applications running on servers without conflicts because a port can be allocated only once on a single host. Every container in a pod shares the network namespace, including the IP address and network ports. Inside a pod, the containers that belong to the pod can communicate with each other by using localhost. When containers in the pod communicate with entities outside the pod, they must coordinate how they use the shared network resources, like ports. Giving each pod its own IP address means that pods can be treated like physical hosts or VMs in terms of port allocation, networking, naming, service discovery, load-balancing, application configuration, and migration.

Red Hat OpenShift Container Platform uses an SDN to implement connectivity. The Red Hat OpenShift SDN separates the network into a control plane and a data plane.

The SDN meets five requirements:

- Manages network traffic and resources as software so that they can be programmed by the application owner.
- Communicates among containers running within the same project.
- Communicates among pods within and beyond project boundaries.
- Manages network communication from a pod to a service.
- Manages network communication from an external network to a service.

Red Hat OpenShift network resources include the following items:

- Services provide load-balancing to replicated pods in an application, which are essential in providing access to applications. Services connect to endpoints, which are individual pod IP addresses.
- Ingress is a K8s resource that exposes services to external users:
  - Ingress adds URLs, load-balancing, and access rules.
  - Ingress is not used this way in Red Hat OpenShift.
- Red Hat OpenShift routes are an alternative to Ingress.

  To facilitate multiple services, such as front-end and back-end services while using multiple pods, use environment variables for usernames, service IP addresses, and more, so that front-end pods can communicate with back-end services.

[6] https://docs.openshift.com/container-platform/4.11/welcome/index.html

The network is managed by the Red Hat OpenShift Cluster Network Operator (CNO). CNO deploys and manages the cluster network components on a Red Hat OpenShift Container Platform cluster, including the Container Network Interface (CNI) default network provider plug-in that is selected for the cluster during installation.

The DNS operator implements CoreDNS. The internal CoreDNS server is used by pods for DNS resolution, which means that we do not need to set the DNS name server on our pods because it is automatically set to the IP address of CoreDNS.

Services are K8s API resources that are used in Red Hat OpenShift:

- Services are used as a load-balancer that provides access to a group of pods that is addressed by using a label as the selector.
- Services are needed for pod access because pods are added and removed dynamically.
- Services use labels and selectors to address dynamically pods.
- If you use **`oc new-app`**, a service resource is automatically added to expose access to the application.

There are different types of services, for example, ClusterIP, NodePort, LoadBalancer, and ExternalName.

We should also understand how network policies work.

- By default, there are no restrictions for network traffic in K8s.
- Pods always can communicate, even if they are in other namespaces.
- To limit communication, use network policies.
- If there is no match in a policy, traffic is denied.
- If no network policy is used, all traffic is allowed.

There are three different identifiers that can be used in the network policies:

- Pods (podSelector): The pod cannot block access to itself.
- Namespace (namespaceSelector): Grants access to a specific namespace.
- IP blocks (ipBlock): Notices that traffic to and from the node where a pod is running is always allowed.

When defining a pod or namespace-based network policy, a selector label is used to specify what traffic is allowed.

4.2.3 Hyperconverged infrastructure and cloud

National Institute of Standards and Technology (NIST) SP 800-145, what was written by Peter Mell and Tim Grance (2011),[7] defines cloud computing as:

“A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources.”

[7] https://csrc.nist.gov/publications/detail/sp/800-145/final

Cloud computing is a key enabler of digital transformation that provides developers with the solutions and tools that they need to create modern applications and rapidly respond to business changes. Red Hat OpenShift plays a vital role by being a cloud-native container orchestrator platform that enables the main cloud essential characteristics that are defined by NIST:

- On-demand self-service: With Red Hat OpenShift, you can provision quickly resources by using the console, APIs, or the Red Hat OpenShift command-line interface (CLI).
- Broad network access: Red Hat OpenShift leverages K8s network capabilities.
- Resource pooling: Red Hat OpenShift enables the usage of a broad spectrum of hardware, so you can have different types and models of worker nodes, for example, you can build a regular pool of common workers and another pool of specialized GPU enabled-hardware worker pool for specific workloads. You also can use different types of storage. Combined with the flexibility of the Power platform, you obtain a plethora of available options.
- Rapid elasticity: Red Hat OpenShift enables workloads to increase or decrease quickly their capabilities on-demand. Pods can be configured to answer quickly to peaks in demand and increase their replicas, and decommission those pods after demand returns to a normal state. You also can use Red Hat OpenShift Serverless to provide a serverless solution, where your app remains with zero container running and scales only to 1 or more when necessary, freeing resources for other applications.
- Measured service: Red Hat OpenShift provides full observability of the resources being used, with native dashboards showing the consumption of CPU, memory, network, disk, and much more.

Now, let us explore, in a hands-on approach, how Red Hat OpenShift addresses these characteristics.

Important: These next steps assume that you have administrator access to your Red Hat OpenShift environment on your Power platform.

Note: We show only how Red Hat OpenShift fulfills the cloud-computing characteristics. We provide a hands-on example for each one of those characteristics, but be aware that the platform has many other features that also are examples on how such characteristics are met.

On-demand self-service

This section describes how a developer can easily deploy an application by using Red Hat OpenShift in a self-service fashion without provisioning an infrastructure or help from the operations team.

In our example, the developer deploys a sample *quarkus* hello world application by completing the following steps:

1. In the Developer perspective, create a project that is named “devproject”, as shown in Figure 4-3.

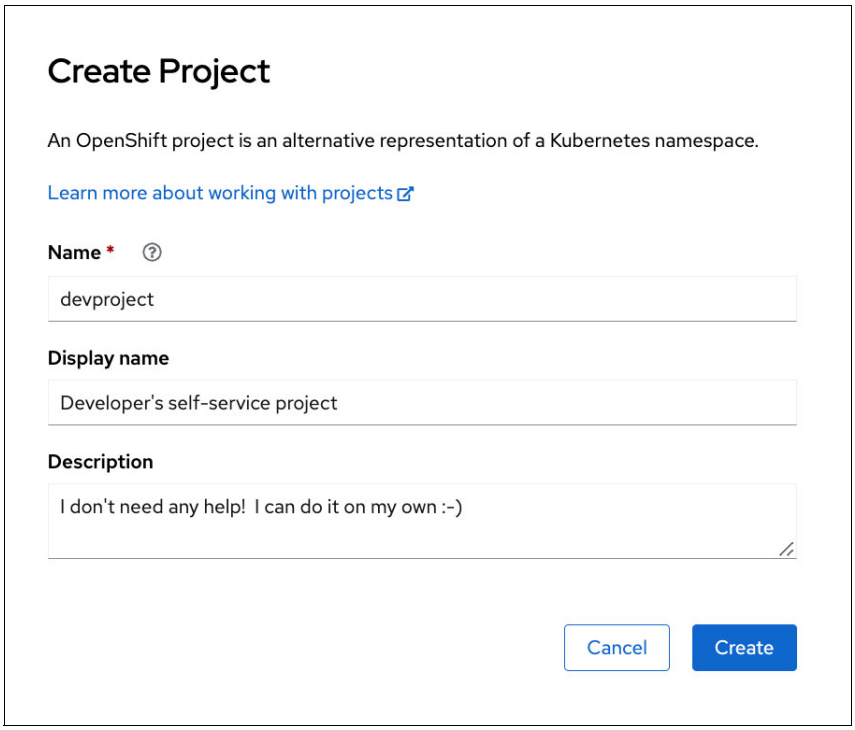


Figure 4-3 Create Project

2. Select **Basic Quarkus** in the “Create applications using samples” category, as shown in Figure 4-4.

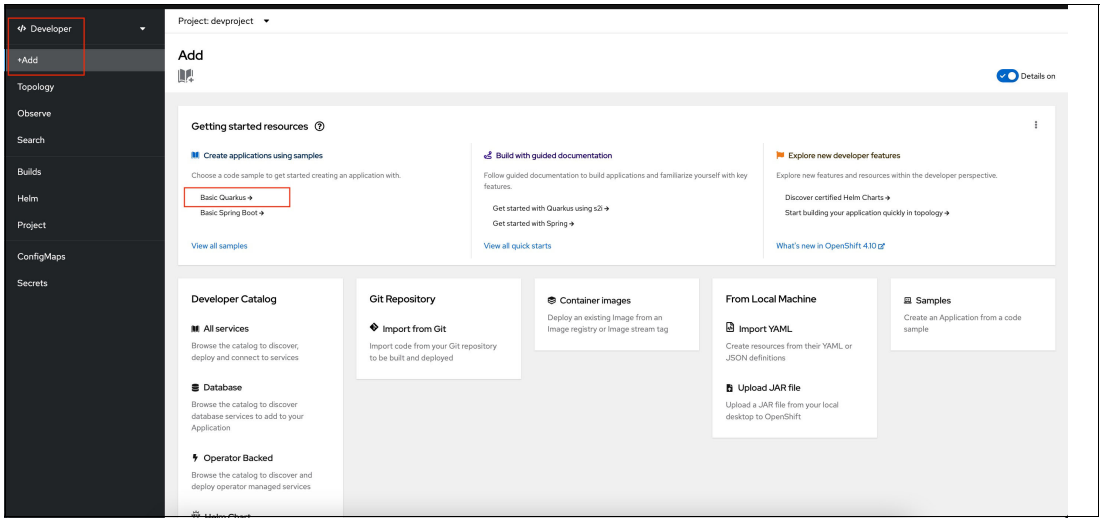


Figure 4-4 Creating an application: step 1

3. Use the suggested code-with-quarkus name, and click **Create**, as shown in Figure 4-5.

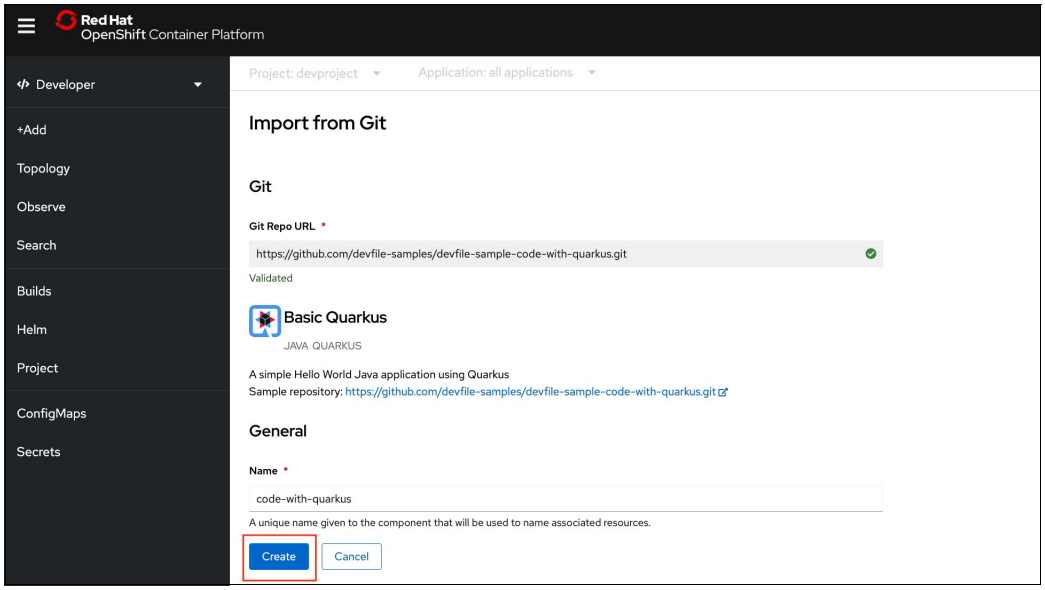


Figure 4-5 Creating an application: step 2

4. The developer provisioned the app and it is ready to go. Click **Open URL** and see the app, as shown in Figure 4-6 and Figure 4-7.

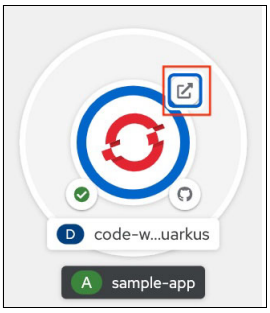


Figure 4-6 Quarkus Hello world application

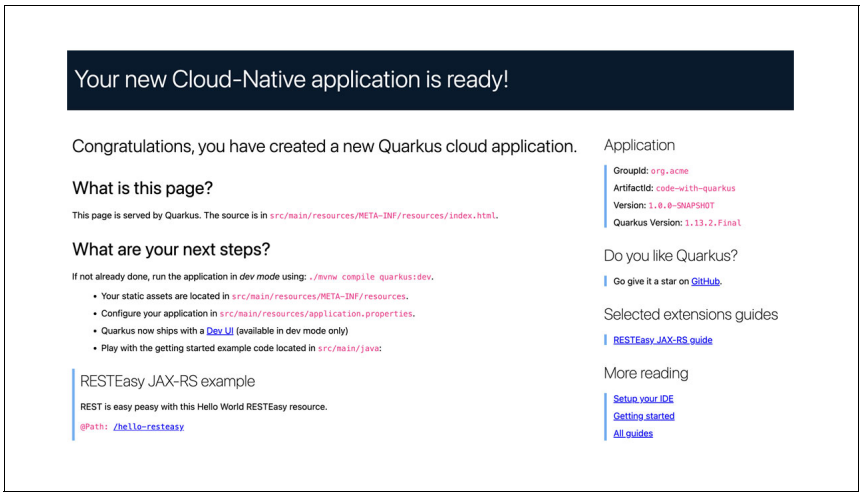


Figure 4-7 Hello world

Resource pooling

Red Hat OpenShift is about resource pooling. Your cluster hosts different workloads from different people that share the underlying infrastructure, like CPUs, memory, and storage.

To allocate storage for the cloud environment, complete the following steps:

1. In the Administrator perspective, select **Storage** → **PersistentVolumeClaims**, as shown in Figure 4-8.

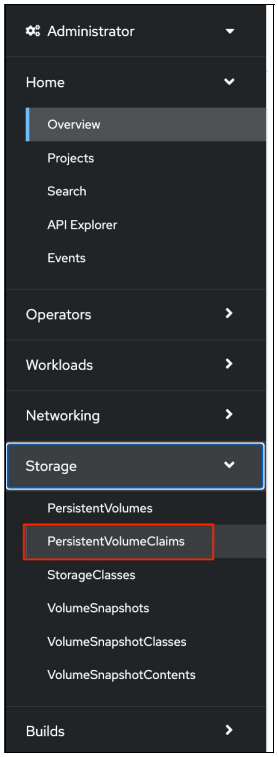


Figure 4-8 Creating a persistent volume claim

2. Create a persistent volume claim (PVC). You have a persistent volume (PV) available or can provision one. In our example, we are using IBM Cloud, so we leverage resource pooling even more: IBM Cloud natively provides (among others) a storage class that is called `ibm-vpc-block-10iops-tier`, which we use to provision a block storage PVC to our project. Select **Create PersistentVolumeClaim**, as shown in Figure 4-9.

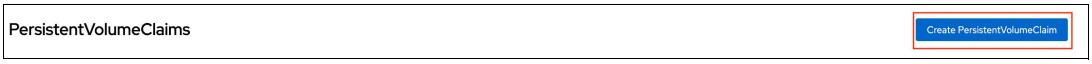


Figure 4-9 PersistentVolumeClaims

3. Complete the Claim name and size. In the volume mode, select **Block**, and then select **Create**, as shown in Figure 4-10.

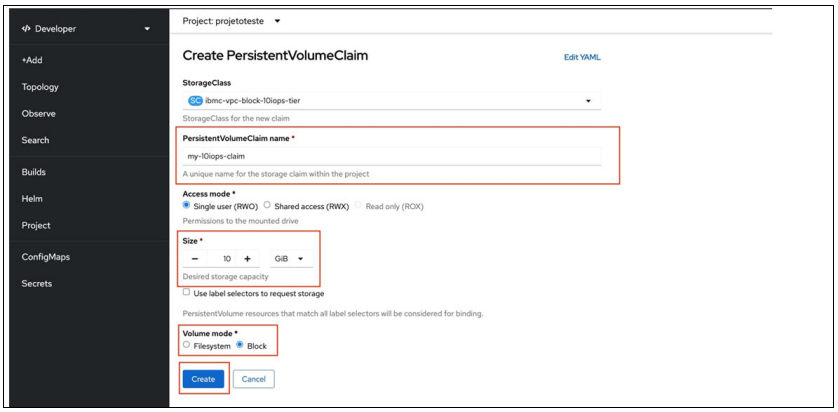


Figure 4-10 PVC details

4. Select **Events**. You see the details of the creation of the PVC, as shown in Figure 4-11.

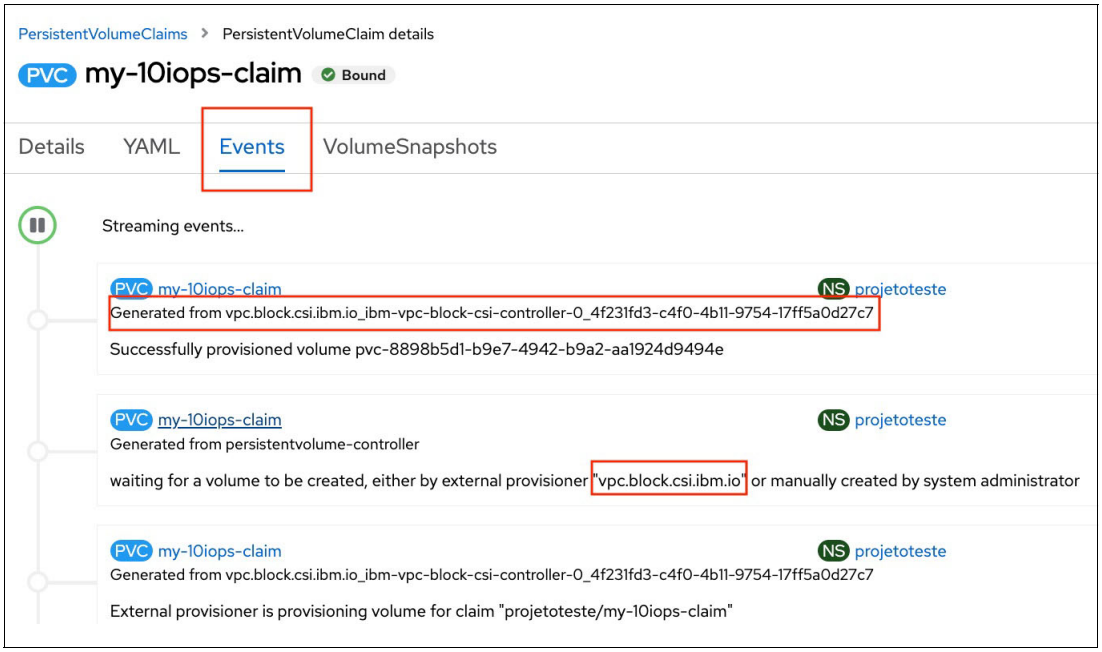


Figure 4-11 PVC results

An external provider that is named `vpc.block.csi.ibm.io` was requested to create the volume, which was successful. The PVC was created on top of this volume with the following name:

`vpc.block.csi.ibm.io_ibm-vpc-block-csi-controller-0_4f231fd3-c4f0-4b11-9754-17ff5a0d27c7`

5. Repeat the process. Create a PV that will be used for another workload, and then repeat steps 2 on page 56 and 3 on page 57 to create a PVC that is named `my-second-volume`. Make sure to select the same storage class and Block volume mode, as shown in Figure 4-12.

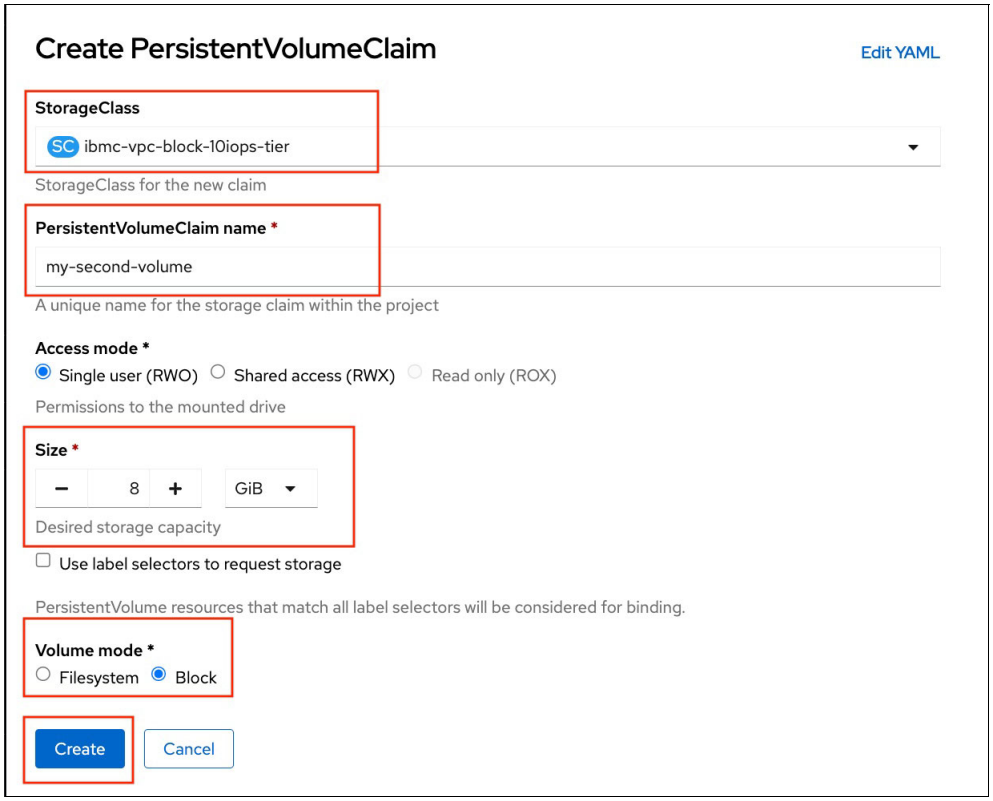


Figure 4-12 Second PVC request

6. Go to the **Events** menu. Figure 4-13 shows the results.

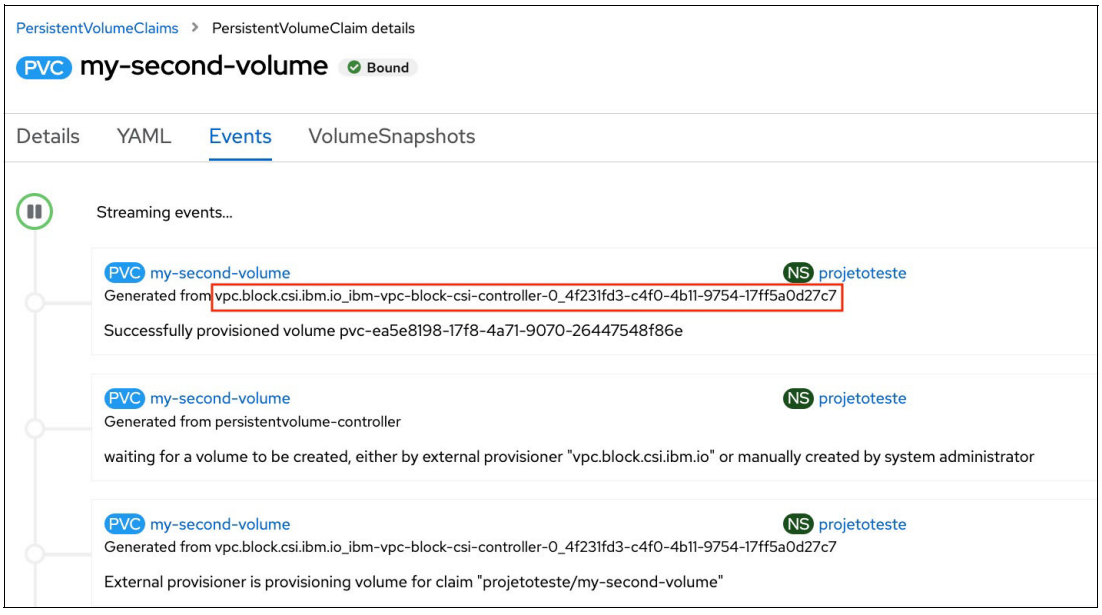


Figure 4-13 Second PVC request results

7. You see resource pooling in action, that is, the new PVC was created on top of the same PV we had before:

   ```
   vpc.block.csi.ibm.io_ibm-vpc-block-csi-controller-0_4f231fd3-c4f0-4b11-9754-17ff5a0d27c7
   ```

Rapid elasticity

To explore quickly the rapid elasticity characteristic, go to the sample quarkus application that we deployed in “On-demand self-service” on page 53, and then complete the following steps:

1. Go to `devproject` and click the deployment icon, as shown in Figure 4-14.

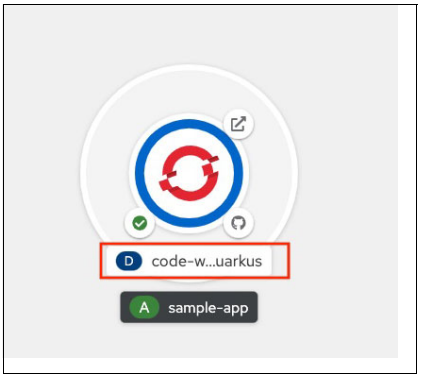


Figure 4-14 Deployment example

2. The app is running in one pod. You can increase or decrease capacity by clicking the arrows. Scale to 10, as shown in Figure 4-15.

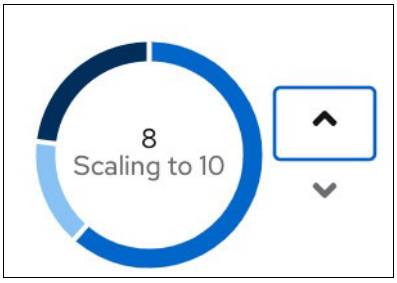


Figure 4-15 Scale pods

Red Hat OpenShift ensures that traffic to your app is balanced between those new instances. This example is a simple one for exploring the topic. In real-world scenarios, you have triggers that are linked to specific conditions (like CPU usage, as shown in Figure 4-16) that automatically scale your pods.

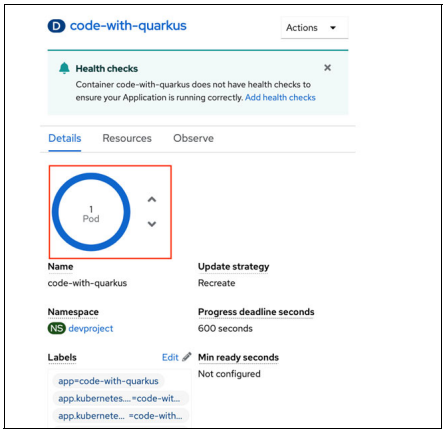


Figure 4-16 Health conditions

4.2.4 Supported operating systems and hypervisors

Red Hat OpenShift supports many operating systems for use as the base operating system for containers. Some of the supported operating systems include the following ones:

- RHEL

  RHEL is a widely used enterprise-grade Linux distribution that is supported by Red Hat OpenShift. It is designed for mission-critical workloads and optimized for performance and security. To secure RHEL, see *Red Hat Enterprise Linux 8 Security hardening guide* and *IBM Power Systems Cloud Security Guide: Protect IT Infrastructure In All Layers*, REDP-5659.

- SUSE Linux Enterprise Server

  SUSE Linux Enterprise Server is a widely used enterprise-grade Linux distribution that is supported by Red Hat OpenShift. It is designed for mission-critical workloads and optimized for performance and reliability. To secure SUSE Linux Enterprise Server, see *SUSE Linux Enterprise Server 12 SP4 Security Guide*.

- CentOS

  CentOS is a community-supported distribution of RHEL that is supported by Red Hat OpenShift. It is based on the same source code as RHEL, and designed to be fully compatible with it.

- Fedora

  Fedora is a community-supported Linux distribution that is sponsored by Red Hat. It is known for its fast release cycle and often used as a platform for testing new technologies.

- Oracle Linux

  Oracle Linux is a Linux distribution that is optimized for running Oracle applications that is supported by Red Hat OpenShift.

Red Hat OpenShift on Power servers can be deployed only on the PowerVM hypervisor. Here are some best practices to secure a Virtual I/O Server (VIOS):

- Set the security level to specify the security hardening rules for your VIOS system by using the **viosecure** command, as shown in Example 4-1.

Example 4-1 Securing VIOS

```
# viosecure -level high -apply
```

- Enable the VIOS firewall to control IP address activity by using the **viosecure** command with the **-firewall** option, as shown in Example 4-2.

Example 4-2 VIOS firewall

```
# viosecure -firewall allow | deny -port number
```

- You can configure a Kerberos client on the VIOS to enhance security in communications across the internet, as shown in Example 4-3.

Example 4-3 Configuring a Kerberos client

```
# mkkrb5clnt -c KDC_server -r realm_name \ -s Kerberos_server -d Kerberos_client
```

- Use role-based access control (RBAC) to define roles for users in the VIOS, as described at Using role-based access control with the Virtual I/O Server.

Note:

- Starting with RHEL 8.4, using KVM virtualization on IBM power hardware is deprecated. As a result, currently certified IBM Power servers with KVM are still supported by RHEL 8. The KVM virtualization function was removed with the release of RHEL 9 for the IBM Power Architecture®. In addition, future IBM Power servers will not be certified to use KVM virtualization technology with any RHEL release.
- If you are using KVM with IBM POWER8® or IBM POWER9 systems on RHEL 8, you may continue to do so until 31 May 2029.

For more information, see Deprecation of KVM on Red Hat Enterprise Linux for IBM Power.

4.2.5 Red Hat OpenShift operators

A Red Hat OpenShift operator is a method of packaging, deploying, and managing a K8s application that provides autonomous management by exposing a configuration natively through K8s objects to automate repeatable tasks.[8]

Here are security best practices for using Red Hat OpenShift operators:

- Use carefully cluster-scope and namespace-scope permissions. Use namespace-scope permissions by using RBAC, as shown in 5.2, “RBAC setup for users and service accounts” on page 106.
- Avoid deploying operators in a shared namespace, especially one that allows non-privileged users access.
- Check RBAC roles that can leverage themselves to gain extra privileges when performing code reviews.

[8] https://www.redhat.com/en/technologies/cloud-computing/openshift/what-are-openshift-operators

- Use the Container Security Operator, which provides vulnerability reporting for images that are added to selected namespaces.
- Use security context for containers to control all privileges of the processes running inside the container.
- Use pod security policies that enable the administrator to configure policies to enforce security on every container running on the cluster.

4.2.6 Cloud-native applications

There are several approaches that can be taken to scan vulnerabilities on a cloud-native application:

- Use a vulnerability scanner: There are many tools that are available that can scan a cloud-native application for vulnerabilities, such as Aqua Security's Cloud Native Security Platform.[9] These tools can scan container images, K8s clusters, and other cloud-native infrastructure for known vulnerabilities.
- Use a static code analysis tool: Static code analysis tools can analyze the source code of an application and identify potential vulnerabilities.
- Use a DAST tool: DAST tools can test an application by sending it various inputs and analyzing the responses. DAST is a type of security testing that involves evaluating the security of an application by interacting with it in a dynamic manner, simulating real-world attacks and attempting to exploit vulnerabilities. During a DAST test, the application is ran and tested in a live environment, typically by using a tool that sends various inputs to the application and analyzes the responses. The tool looks for signs of vulnerabilities, such as cross-site scripting (XSS) attacks, SQL injection attacks, and other types of security issues.
- Use a penetration testing tool: Penetration testing tools, such as Kali Linux,[10] can be used to simulate attacks against a cloud-native application and identify vulnerabilities.
- Perform manual code reviews: Manually reviewing the code of a cloud-native application also can help identify vulnerabilities. This task can be done by a team of security experts or by using automated tools to help with the review.

4.2.7 Ingress Controller

There are several best practices to secure the Red Hat OpenShift Ingress Controller:

- Use a TLS security profile to define which TLS ciphers are required by various Red Hat OpenShift Container Platform components.
- Red Hat OpenShift platform components use *routes* for communication and must trust other components' certificates to interact with them. If you are using a public key infrastructure (PKI), you should configure it so that its privately signed certificate authority (CA) certificates are recognized across the entire cluster.
- Configure the Ingress Controller to enable access logs and forward them to a custom syslog endpoint.
- Configure the Ingress Controller to use a custom certificate with a certificate and key pair in PEM-encoded files (signed by a trusted CA or by a PKI).
- Use the Ingress Controller to implement access controls that are based on a client IP address or hostname.

[9] https://www.aquasec.com
[10] https://www.kali.org

- Use rate limiting to prevent denial of service attacks against the Ingress Controller.
- Monitor the Ingress Controller for unusual traffic patterns or other signs of potential security issues.
- Use the Red Hat OpenShift router or a third-party load balancer to distribute traffic to the Ingress Controller and provide extra security features.

For more information, see Setting the Ingress Controller maximum connections.

4.2.8 Storage back end

Red Hat OpenShift supports the following types of storage:

- PVs: Volumes that can be used by applications to store data. They are persistent, which means that the data that is stored on them is not lost when the pod or container that is using the volume is deleted or restarted.
- Ephemeral volumes: Volumes that are created and destroyed along with the pod or container that is using them. They are not persistent, which means that the data that is stored on them is lost when the pod or container is deleted or restarted.
- ConfigMaps: Key-value pairs that can be used to store configuration data for applications. They are stored in `etcd`, which is the cluster-wide configuration store, and they can be accessed by pods through the apiserver.
- Secrets: Sensitive data, such as passwords or API keys that are stored in `etcd` and can be accessed by pods through the apiserver. They are encrypted at rest and can be accessed only by pods with the appropriate permissions.
- Downward API volumes: Special volumes that allow pods to access certain attributes of their own configuration, such as environment variables or labels, through a special file system.
- EmptyDir volumes: Temporary volumes that are created when a pod is scheduled to a node and deleted when the pod is terminated. They are useful for storing files that do not need to be persisted across pod restarts.

Here are some best practices for securing PVs in Red Hat OpenShift:

- Use encrypted PVs for sensitive data. Many storage back ends support encryption at rest. Enable this feature to protect your data from unauthorized access. For more information, see 2.2, “Storage” on page 12.
- Use ephemeral volumes only for non-sensitive data. Use them for data that does not need to be persistent.
- Use EmptyDir volumes. Use these volumes only for non-sensitive data that does not need to be persistent.
- Enable encryption for `etcd` to protect data at rest and in transit. For more information, see 6.4, “Data at rest encryption” on page 117 and 4.2.13, “Enhanced data resilience and security by using IBM Spectrum Protect Plus” on page 93.
- Use access controls to limit access to PVs. Use RBACs to limit access to volumes to only those users and pods who need it.

4.2.9 Secret management systems

Secret management is a feature that enables administrators to securely store, manage, and use sensitive data, such as passwords, API keys, and encryption keys. Secrets are stored in a secure location within the Red Hat OpenShift environment and are encrypted at rest.

Secrets can be used by applications, pods, and other resources within the Red Hat OpenShift environment to access sensitive data or services. For example, an application might use a secret to access a database or to authenticate with an external API.

Secrets can be created and managed through the Red Hat OpenShift web console or by using the Red Hat OpenShift CLI. Administrators can control access to secrets through RBAC, ensuring that only authorized users or resources can access them.

Red Hat OpenShift provides several built-in secret types, including generic secrets, TLS secrets, and Docker registry secrets. Administrators can create custom secret types as needed.

Using secret management can help organizations ensure the security and integrity of sensitive data within their Red Hat OpenShift environment. It is an important part of a comprehensive security strategy for any Red Hat OpenShift deployment.

Here are some best practices for secret management:

- Use the Red Hat OpenShift web console or CLI to manage secrets rather than storing them in source code or configuration files.
- Regularly review and revoke access for any users who no longer need it.
- Use access controls to limit access to secrets to only those users who need them.
- Regularly rotate secrets, such as passwords and encryption keys to reduce the risk of compromise.
- Implement security monitoring and alerting to detect and respond to potential security breaches.
- Conduct regular security audits to identify and address vulnerabilities in your secret management system.

For more information, see 6.2, “Central secrets management: Single source of truth” on page 114.

4.2.10 Code repository

A *code repository* is where all your computer source assets are stored. With a good code repository solution, you can unlock collaboration and innovation by enabling features like code sharing, code reuse, and automation.

Code repositories are not something new: in The earliest days of the internet, people used software archives that were available through FTP or websites like SourceForge.[11]

One main driving force of the mass adoption of code repositories was the creation of *Git*. Git was written by Linus Torvalds in 2005 to improve the development of the Linux kernel. Git is open-source software and was widely adopted. Many commercial companies built commercial solutions based on Git, like GitHub[12] or GitLab,[13] and today it is hard to find a DevOps toolchain without some Git-based components.

[11] https://sourceforge.net/
[12] https://github.com/

Code repositories can be public or private. Public repositories are the place where open-source code and freely available software are published. By enabling the whole internet to see, download, change, and improve your software, developers foster innovation and obtain the required help to keep their creations updated and relevant. Public code repositories are mined by researchers to study how to improve bug-finding techniques, improve AI, and so on. Private repositories are available for companies, organizations, and individuals who want to leverage Git capabilities but keep access to their intellectual property restricted.

Red Hat OpenShift works well with both scenarios. You can connect your Red Hat OpenShift environment with any Git-based code repository solution, public or private, and use it as the source of the deployment of your applications. In this publication, we explore the usage of IBM Cloud Git Repos and Issue Tracking, but other solutions work as well by following a similar approach.

Git Repos and Issue Tracking is the IBM Cloud solution to store and manage your code. It also provides protection by using granular permissions, and enables collaboration and tracking features. An IBM Cloud Git Repo is inside a construct that is called a *Toolchain*, which is a software-defined space that integrates your DevOps solutions and pipelines.

Complete the following steps:

1. Create the toolchain by going to the IBM Cloud Toolchain creation page and selecting **Build your own toolchain**, as shown in Figure 4-17.

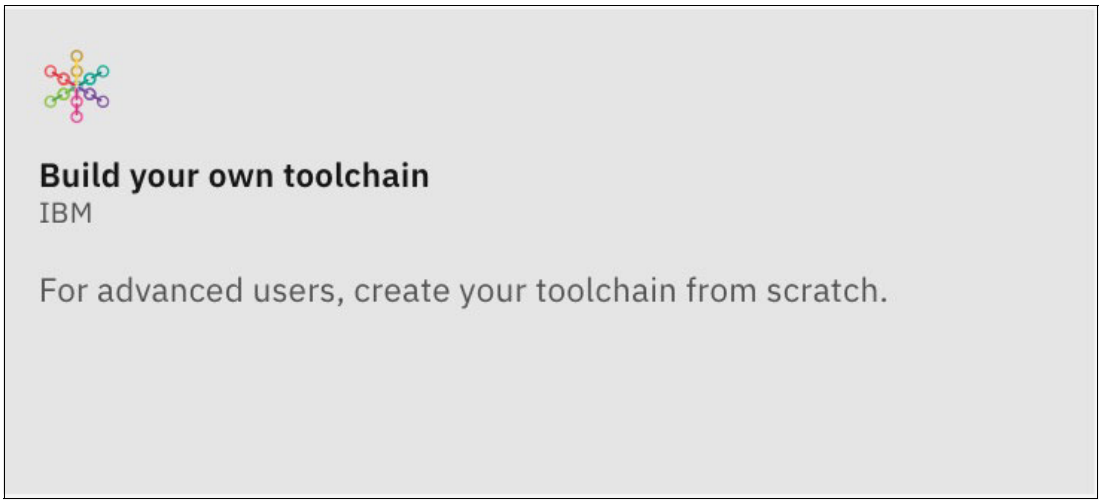


Figure 4-17 Building your own toolchain

[13] https://about.gitlab.com/

2. In the “Build your toolchain” page, give your toolchain a unique name, and select the IBM Cloud region where it will be and its resource group. Click **Create**, as shown in Figure 4-18.

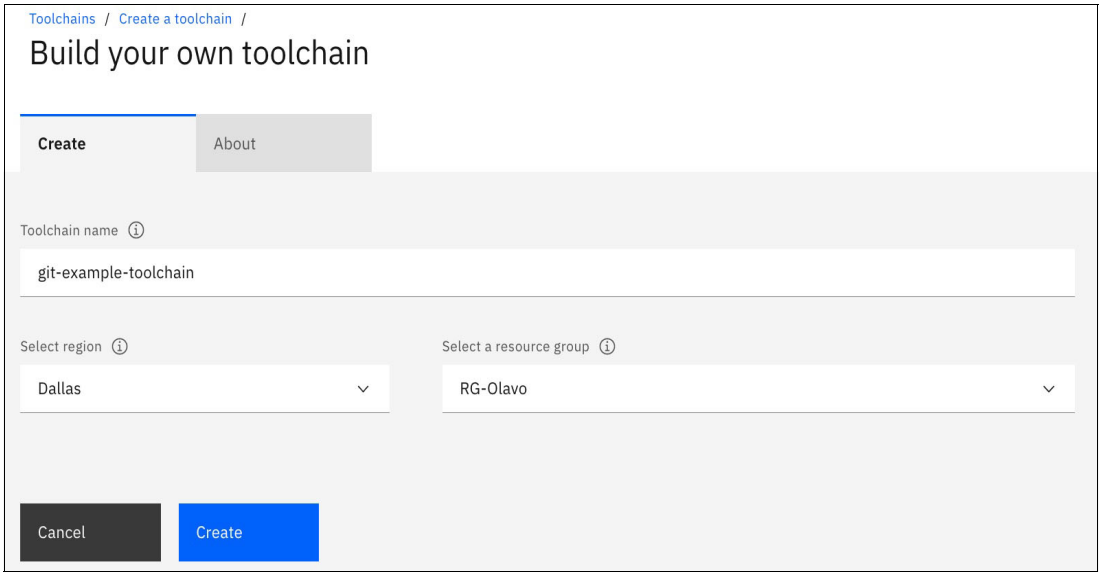


Figure 4-18 Building your own toolchain

3. The toolchain is created. Add your code repository by clicking **Add**, as shown in Figure 4-19.

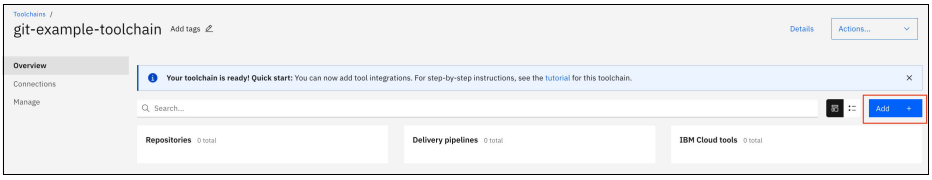


Figure 4-19 Adding your toolchain

4. The “Add tool integration” window shows many different artifacts that you can use inside your toolchain solution. In this example, click **Git Repos and Issue Tracking**,” as shown in Figure 4-20 on page 67.

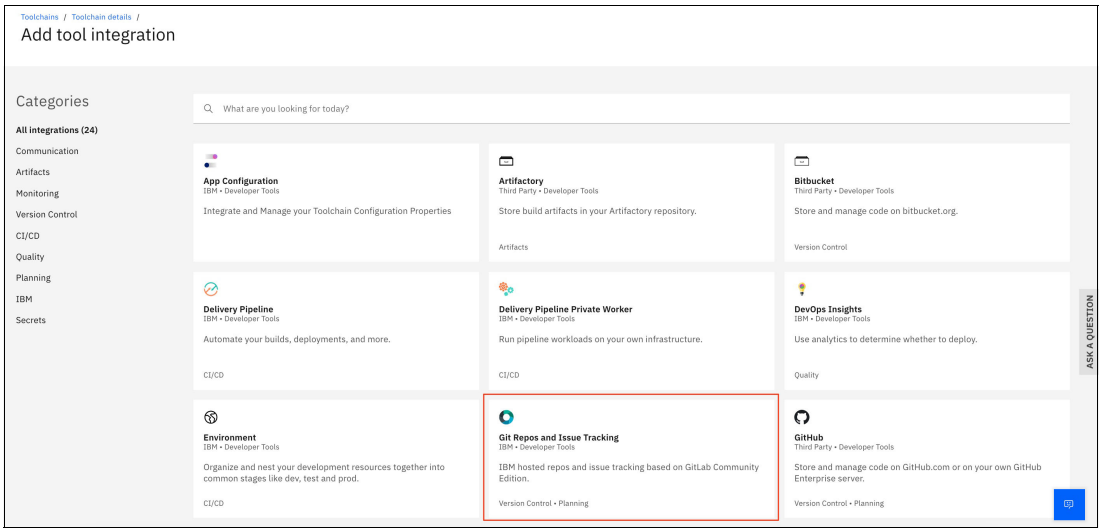


Figure 4-20 Git Repos and Issue Tracking

5. Now, in the “Configure Git Repos and Issue Tracking” window, you create a repository. You have many options. If required, you can, for example, create a repository that is based on an existing one by forking or cloning the repository. Because you are in a “green field” and are doing a fresh deployment, click **New** and give your repository a unique name.

> **Note:** Regarding Issue Tracking, the DevOps for IBM Cloud based their code repository solution on the GitLab open core, and they use the Issues features so that developers and managers have the visibility and flexibility to manage their repositories.

6. Select **Make this repository private** to use a private repository.
7. Complete the fields, and then select **Create Integration**, as shown in Figure 4-21.

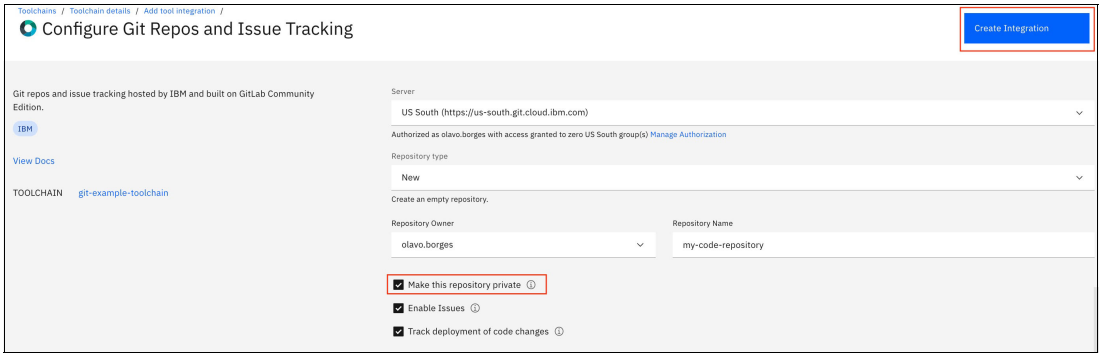


Figure 4-21 Create Integration

You return to your toolchain dashboard. You see that the repository was created. Click the new URL to access the repository, as shown in Figure 4-22.

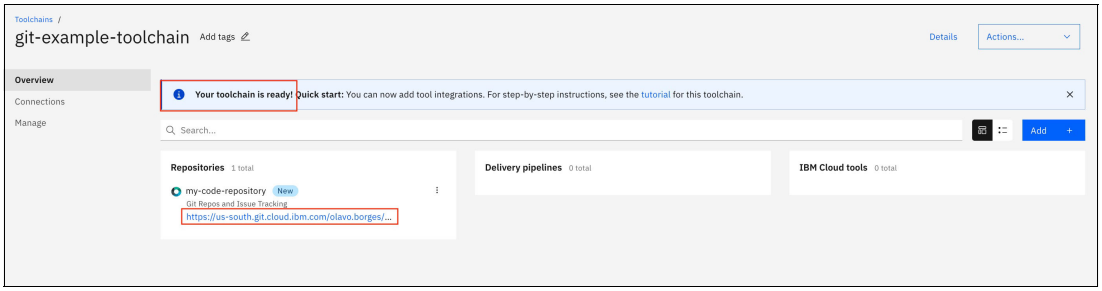


Figure 4-22 Repository successfully created

Now, create two simple files in your repository that you use to provision a Node.js demonstration application by completing the following steps:

1. Create a file that is named `app.js` with the content that is shown in Example 4-4.

Example 4-4 Creating the app.js content

```
http.createServer(function (request, response) {
    response.writeHead(418);See
    response.end('Hello, I\'m a teapot\n');
}).listen(8080);
```

2. Create a file that is named `package.json` with the content that is shown in Example 4-5.

Example 4-5 Creating the package.json content

```
{
  "name": "app",
  "version": "1.0.0",
  "description": "my test app",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "author": "reader",
  "license": "MIT"
}
```

3. The repository is empty. Click **Upload**, as shown in Figure 4-23 on page 69.

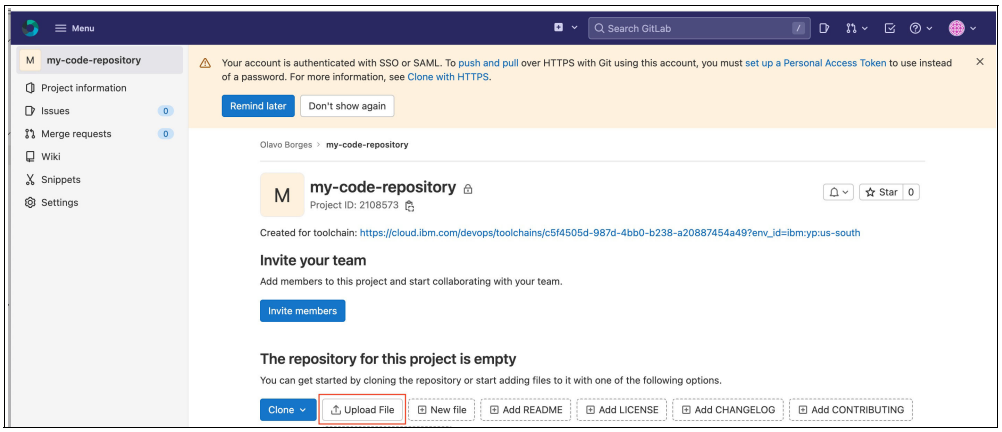


Figure 4-23 Uploading objects

4. Click **Upload** and select `app.js`, as shown in Figure 4-24.

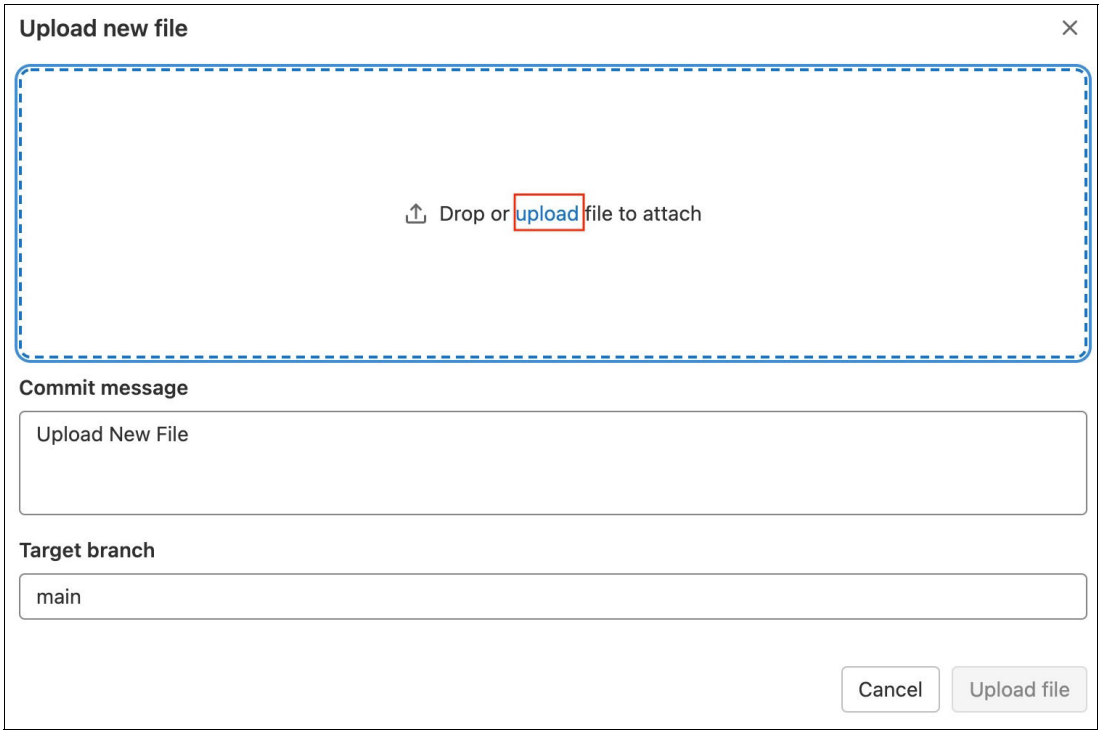


Figure 4-24 Choosing the files to upload

5. Leave the default message and target branch and click **Upload File**, as shown in Figure 4-25.

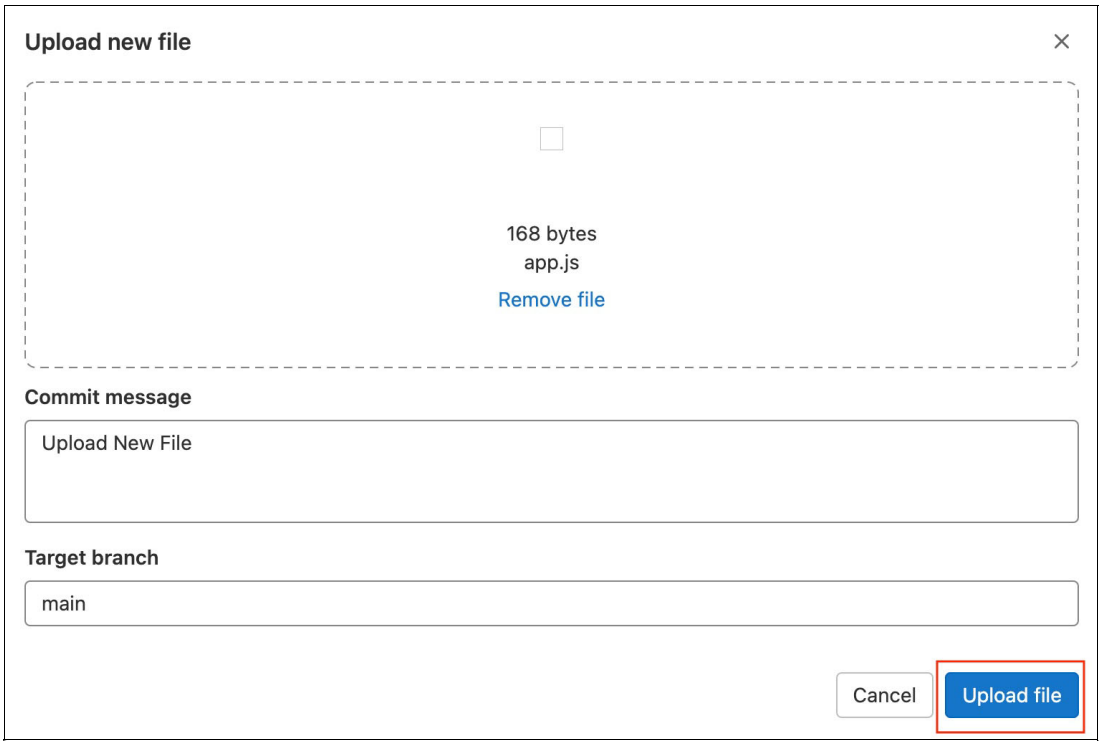


Figure 4-25 Uploading the files

6. Repeat steps 4 on page 69 and 5 to upload `package.json`.
7. Because the code repository is private, create a Personal Access Token so that your Red Hat OpenShift environment can access the repository. Select **Settings** → **Access Tokens**, as shown in Figure 4-26 on page 71.

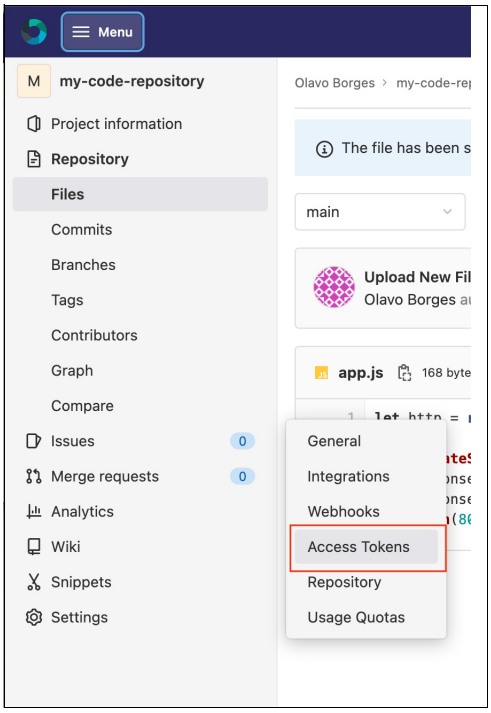


Figure 4-26 Access token creation

8. Define a unique name for your token and an expiration date for it.

   For the role, select “Developer”. Because you need only read access to the repository, select only the read_repository scope. Click **Create project access token**, as shown in Figure 4-27.

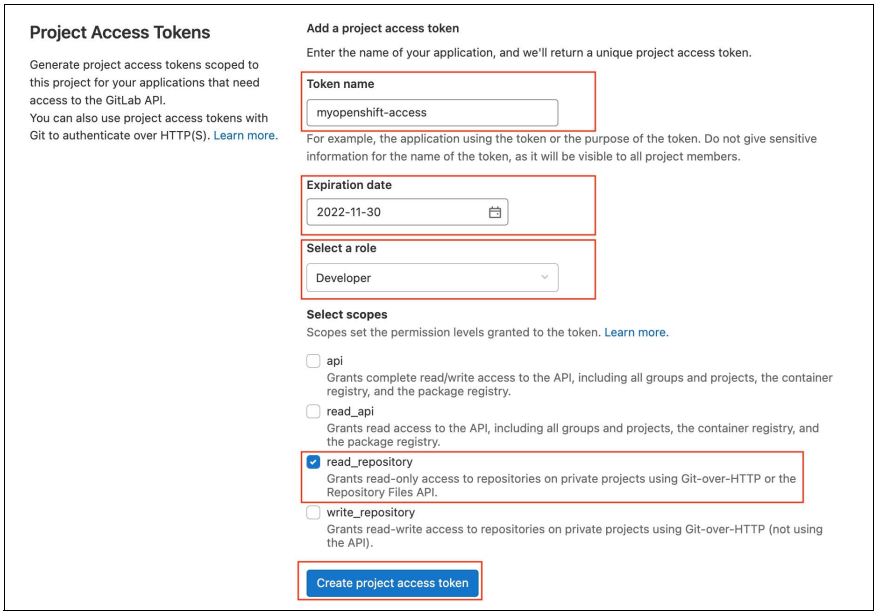


Figure 4-27 Project Access Token definition

Important: Now, you can see your new project access token. Because it is the *only time* that you see it, copy it and store securely. If you do not copy it and store it, you will not be able to use it, and must delete it and create a new one.

9. Copy the project access token so that you can use it later, as shown in Figure 4-28.

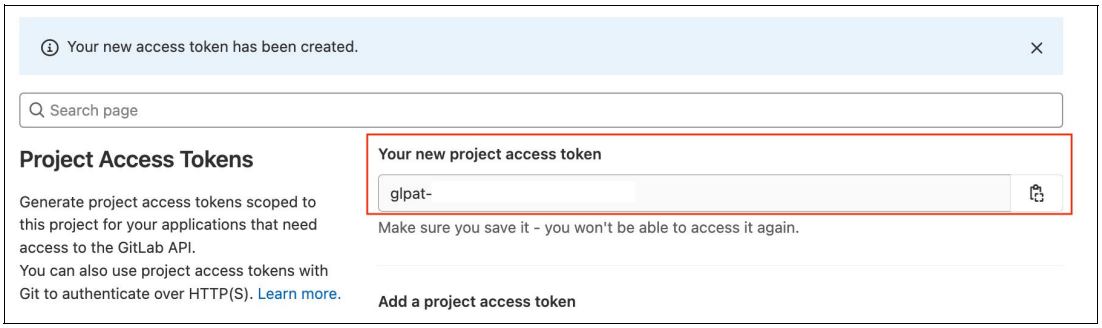


Figure 4-28 Copying the access token

10. Copy your repository URL to use it later, as shown in Figure 4-29. Click **Clone**, and then click **Clone with HTTPS**. Save the copied text.

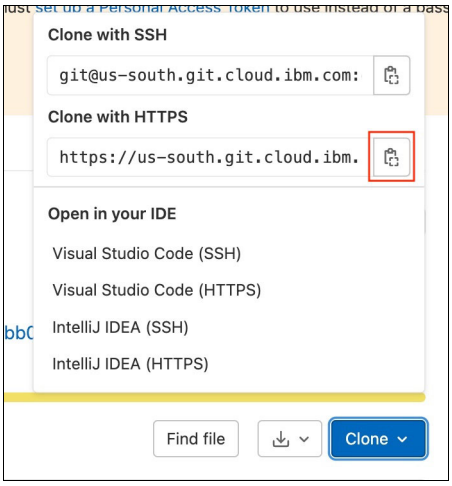


Figure 4-29 Cloning with HTTPS

11. Go to your Red Hat OpenShift environment and configure the secret, which you use to access your Git repository.

Note: The next steps assume that you are logged in to your Red Hat OpenShift environment and have the proper permissions to create a project.

12. Create a project. In the Developer perspective, create a project that is named *"myproject"*, as shown in Figure 4-30 on page 73. Click **Create**.

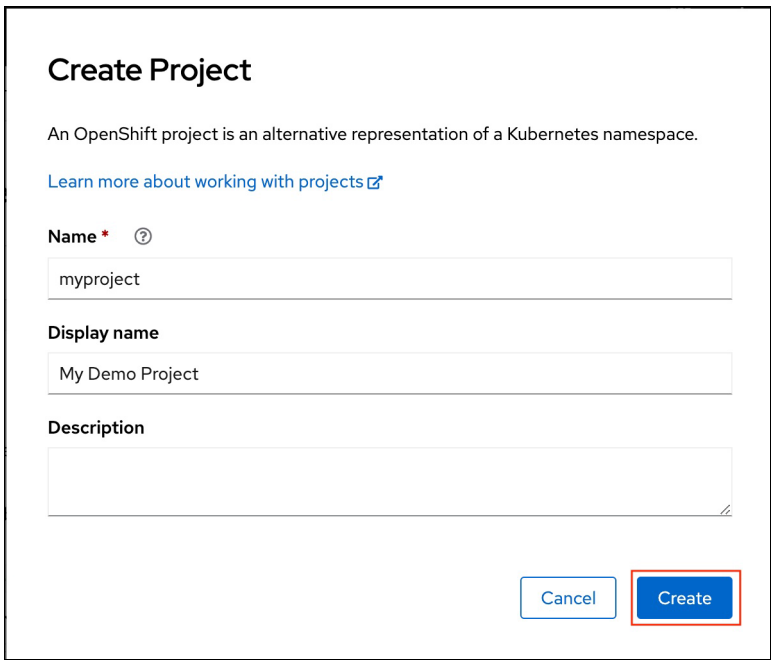


Figure 4-30 Create Project

13. In the Secrets window, and create a source secret, as shown in Figure 4-31.

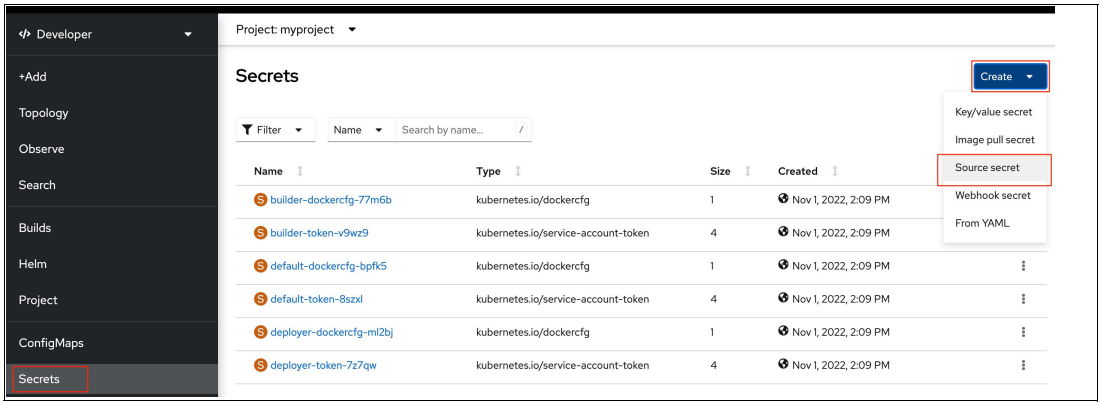


Figure 4-31 Creating a secret

14. Enter a unique name for the secret. Make sure that **Basic authentication** is selected, and for the username, enter the name of the token that you created. The token value itself is entered in the password field. Click **Create**, as shown in Figure 4-32.

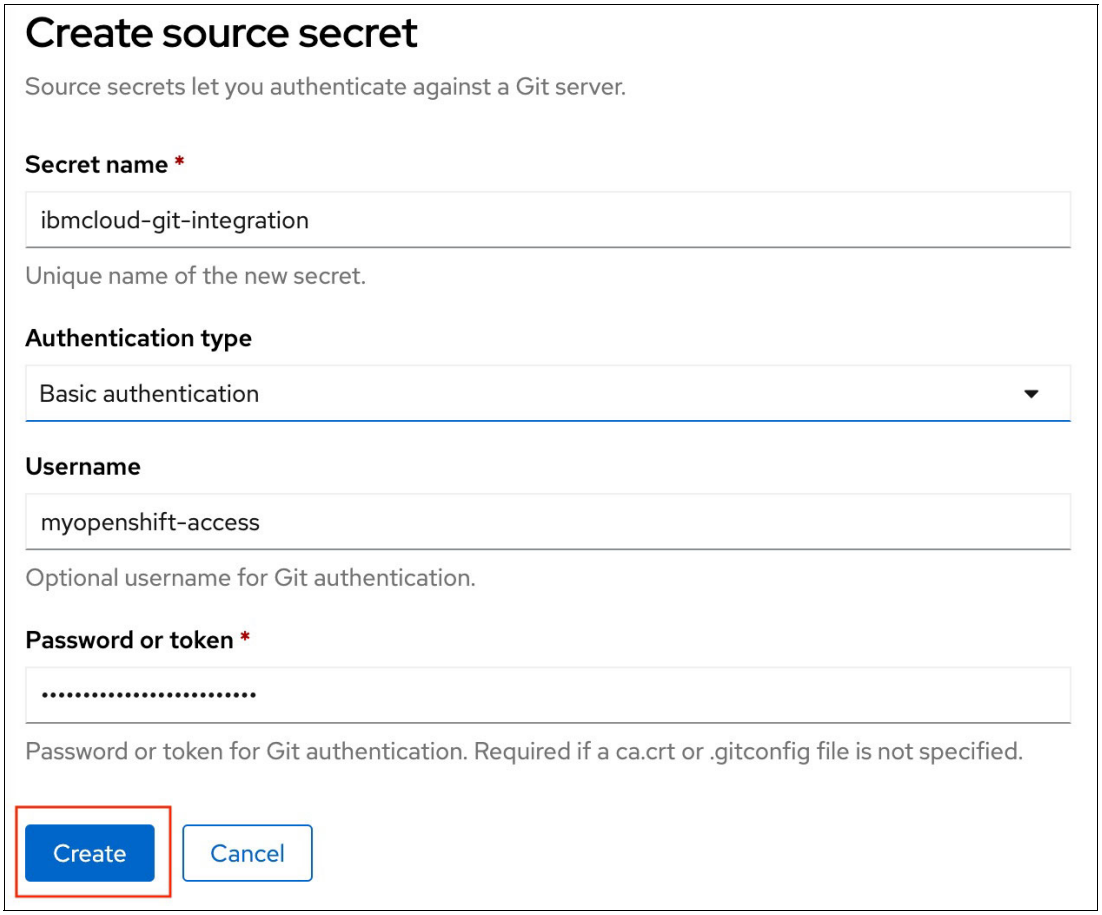


Figure 4-32 Create source secret

15. Use your code repository as the source of a new application that you will create inside your new project. Click **+Add**, and then click **Import from Git** inside the Git Repository pane, as shown in Figure 4-33.

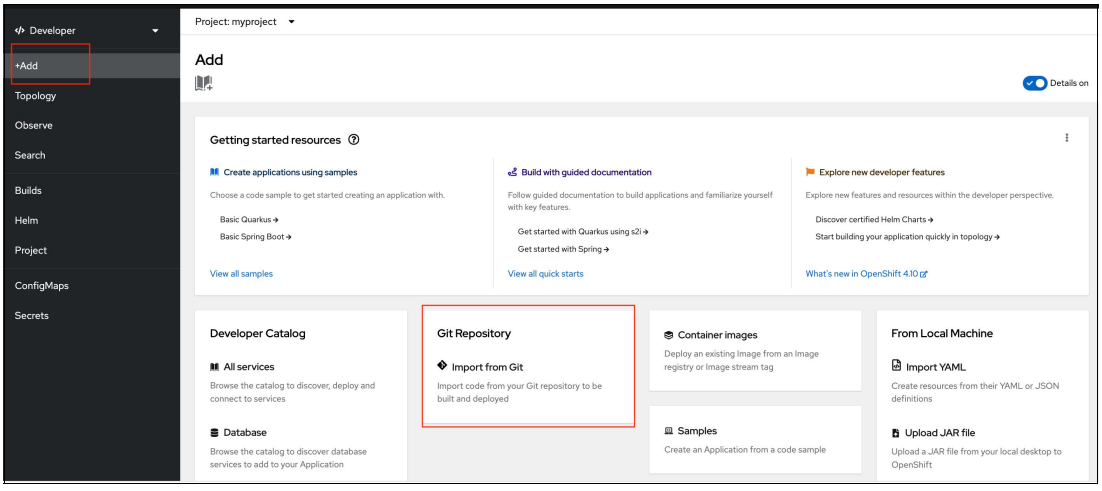


Figure 4-33 Adding a project

16. In the Git Repo URL field, type the name of your code repository. Because IBM Cloud code repositories are based on GitLab, you must select **GitLab** in the Git type field.

    Your repository is read. The wizard detects that the repository must use a Node.js builder image. Leave everything else as default and click **Create**, as shown in Figure 4-34.

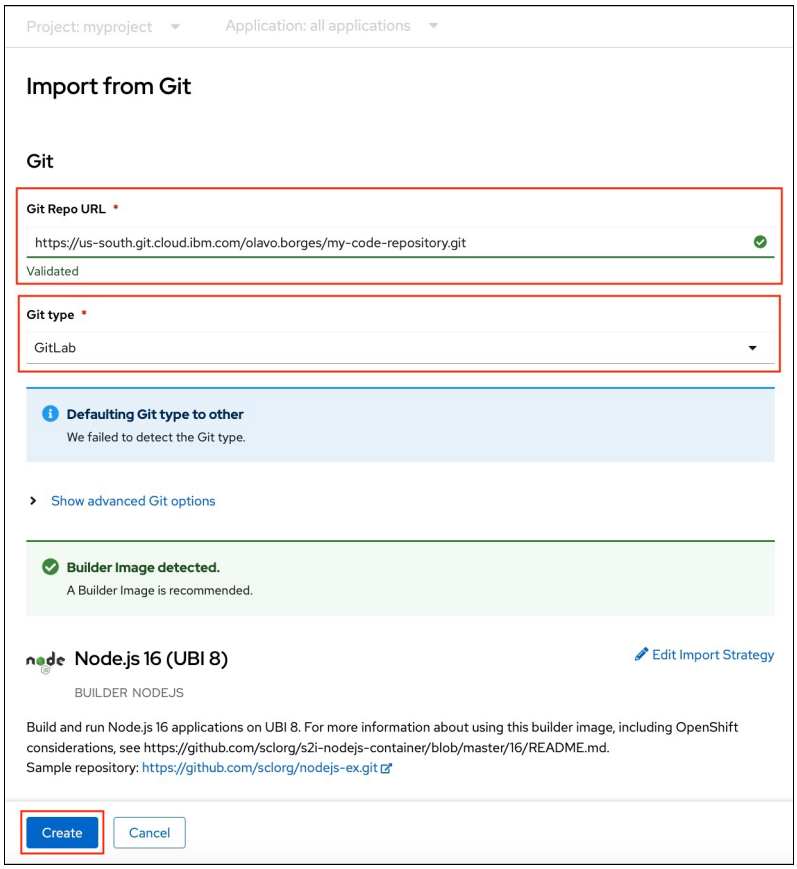


Figure 4-34 Import from Git

17. Your new app is created from your code repository. Click the expand icon to open your new teapot app, which is shown in Figure 4-35.

Figure 4-35 Application

4.2.11 Container registry

In a container orchestration platform like Red Hat OpenShift, container registries play an important role. They are responsible both for storing and managing the containers in your environment. Although you can build your own container registry, there is a plethora of ready-to-use options that are available for use in all major cloud providers. These options offer container registry as a service, and they often include extra services like high availability and vulnerability scanning. These options integrate nicely with Red Hat OpenShift.

Red Hat OpenShift comes with an internal container registry that is managed by the Infrastructure Operator. When you are building a secure enterprise-grade infrastructure, you must leverage solutions that go beyond what the internal container registry delivers.

IBM Cloud Container Registry provides a ready-to-use container platform where you can store private and public images; define access policies so only authorized systems and applications can access the images; and manage the security of the images by using Vulnerability Advisor. IBM Cloud Container Registry also provides high availability by leveraging the IBM Cloud worldwide infrastructure so that you can connect your central image repository with as many Red Hat OpenShift clusters as you have, whether they are on-premises in your Power environment or spread across various cloud providers or other infrastructures.

Creating your first IBM Cloud Container Registry

The following procedure assumes that you have an IBM Cloud account. If you do not have an account, create one at `https://cloud.ibm.com`.

Complete the following steps:

1. Log in to your IBM Cloud account, and in the upper right (Figure 4-36), select the IBM Cloud Shell icon to open a shell session. With IBM Cloud Shell, you can manage your environment with the already configured IBM CLI tools.

Figure 4-36 IBM Cloud Shell

2. To interact with the container registry service, first configure the container registry region. In this example, we use `Dallas (us-south)`. Run the following command:

   ```
   ibmcloud crregion-set us-south
   ```

   The results are shown in Figure 4-37.

```
olavoborges@cloudshell:~$ ibmcloud cr region-set us-south
The region is set to 'us-south', the registry is 'us.icr.io'.
```

Figure 4-37 Cloud region set results

3. Create a namespace. In this example, we name it ‘`myfirst-ns`’. We use the default resource group. Run the following command:

   ```
   ibmcloud cr namespace-add myfirst-ns
   ```

   The results are shown in Figure 4-38 on page 77.

```
Successfully added namespace 'myfirst-ns'

OK
```

Figure 4-38 Namespace added results

4. Push your custom image to the container registry. However, because you are experimenting, first download an image that you can use for learning purposes and use it as your “custom image”.

   In this example, we use the default Apache HTTP container image from Docker Hub for this task. To download it locally, use the following command:

   ```
   docker pull httpd
   ```

   The results are shown in Figure 4-39.

```
olavoborges@cloudshell:~$ docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
a603fa5e3b41: Pull complete
4691bd33efec: Pull complete
ff7b0b8c417a: Pull complete
9df1012343c7: Pull complete
b1c114085b25: Pull complete
Digest: sha256:f2e89def4c032b02c83e162c1819ccfcbd4ea6bdbc5ff784bbc68cba940a9046
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
```

Figure 4-39 Docker image pull results

5. To create a repository, first tag your image and then push it:
   a. First, tag your image. Run the following command:

      ```
      docker tag httpd us.icr.io/myfirst-ns/my-repo
      ```

   b. Log in to the container registry by running the following command:

      ```
      ibmcloud cr login
      ```

The results are shown in Figure 4-40.

```
olavoborges@cloudshell:~$ ibmcloud cr login
Logging 'docker' in to 'us.icr.io'...
Logged in to 'us.icr.io'.

OK
```

Figure 4-40 Container registry login results

   c. Push the image by running the following command:

      ```
      docker push us.icr.io/myfirst-ns/my-repo
      ```

The results are shown in Figure 4-41.

```
dlavoborges@cloudshell:~$ docker push us.icr.io/myfirst-ns/my-repo
Using default tag: latest
The push refers to repository [us.icr.io/myfirst-ns/my-repo]
a4a57da7ddfc: Pushed
a0b242781abd: Pushed
29657939e55a: Pushed
7603afd8f9aa: Pushed
ec4a38999118: Pushed
latest: digest: sha256:f2f8e87c61c78701277b59757b6e43d531571e4426b512f5248729811508d2de size: 1366
```

Figure 4-41 Pushing the image to the registry results

6. The image is now inside your new repository and ready to use. You can browse the menu and see the many options that the solution provides.

   If you go to the **Images** menu, for example, you see that the image was scanned by the IBM Cloud Vulnerability Advisor and that no security issues were found, as shown in Figure 4-42.

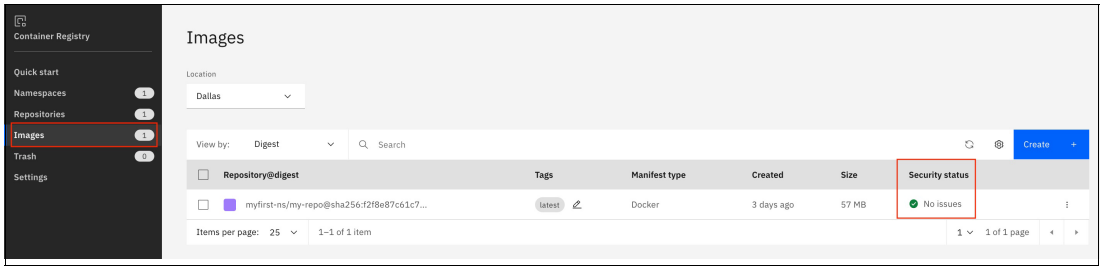


Figure 4-42 Container registry results

4.2.12 Vulnerability scanners

Red Hat OpenShift vulnerability scanning is an important process to identify and remediate security gaps in Red Hat OpenShift deployments. The process involves updating Red Hat OpenShift itself when vulnerabilities are discovered in the open-source project; scanning container images and open-source elements within them for vulnerabilities; and ensuring that the Red Hat OpenShift configuration meets best practices and compliance requirements.

Here are the key elements of Red Hat OpenShift vulnerability scanning:

- Vulnerabilities in Red Hat OpenShift itself
- Container image scanning

Exploring Vulnerability Advisor

Vulnerability Advisor is one of the best features of IBM Cloud Container Registry. By using it, you can scan your images before making them available to your environments so that any issues are reported beforehand. You also can integrate it into your DevOps pipeline so that detected security vulnerabilities in images are reported, which triggers automated tasks or blocks the deployment.

Vulnerability Advisor offers a dashboard to learn more about your image's security posture, as shown in Figure 4-43 on page 79.

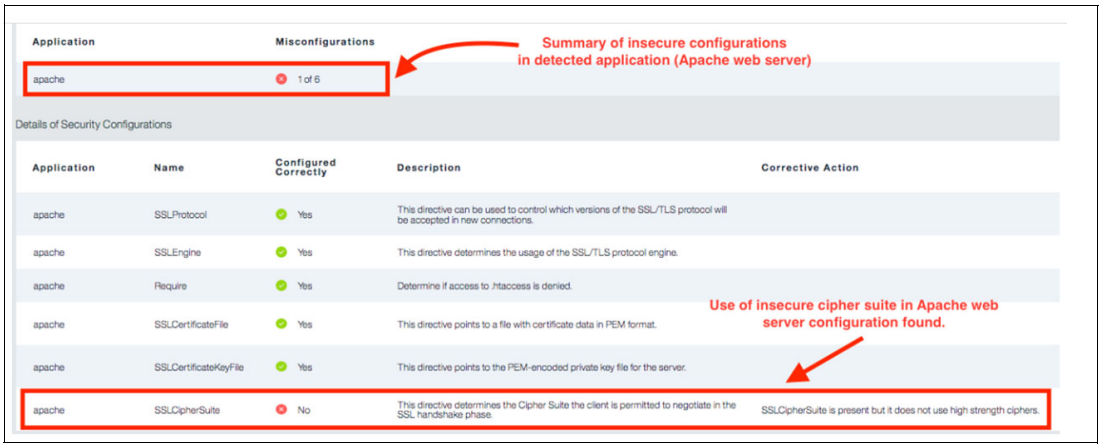


Figure 4-43 Vulnerability Advisor dashboard

As an enterprise-grade solution, you can use IBM Cloud Container Registry to customize policies, define exceptions, and configure the solution to work for your specific needs.

Scanning pods for vulnerabilities by using Red Hat Quay

Red Hat Quay is a private container registry that stores, builds, and deploys container images. It analyzes your images for security vulnerabilities to identify potential issues and mitigate security risks.

Red Hat Quay Container Security Operator provides access to vulnerability scan results from Red Hat OpenShift Container Platform for container images that are used in active pods on the cluster.

Red Hat Quay Container Security Operator performs the following functions:

- Watches containers that are associated with pods on all or specified namespaces.
- Queries the container registry where the containers came from for vulnerability information (if an image's registry is running image scanning).
- Exposes vulnerabilities through the Manifestation object in the K8s API.

For more information, see Security and compliance overview.

Scanning pod images with the Container Security Operator

Complete the following steps:

1. Check that the Red Hat Quay Container Security Operator is installed by selecting **Operators** → **Installed Operators**, as shown in Figure 4-44.

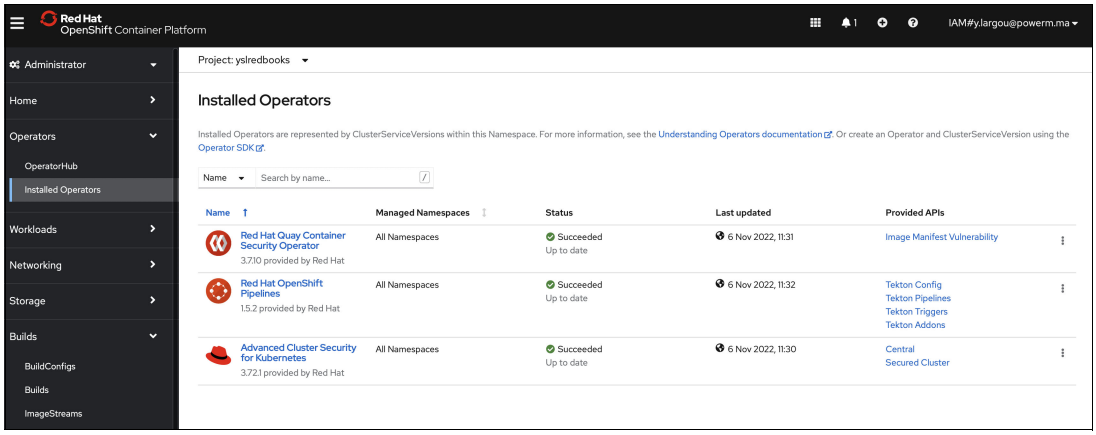


Figure 4-44 Installed Operators page

2. Under the Red Hat OpenShift Dashboard, there is a link to Image Security under the status section that lists the number of vulnerabilities that were found, as shown in Figure 4-45. To see more details, select the link to the vulnerability.

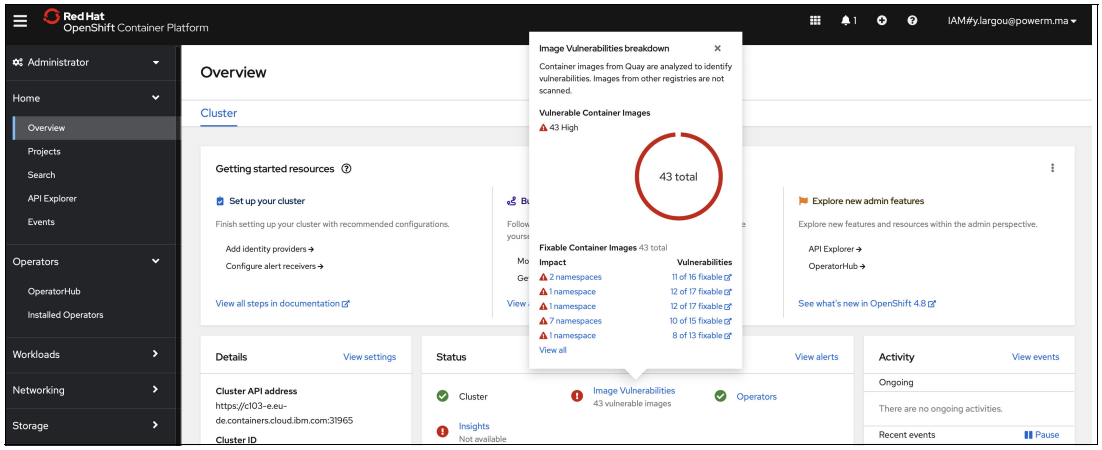


Figure 4-45 Vulnerabilities list page

Figure 4-46 on page 81 shows an example of detected vulnerabilities.

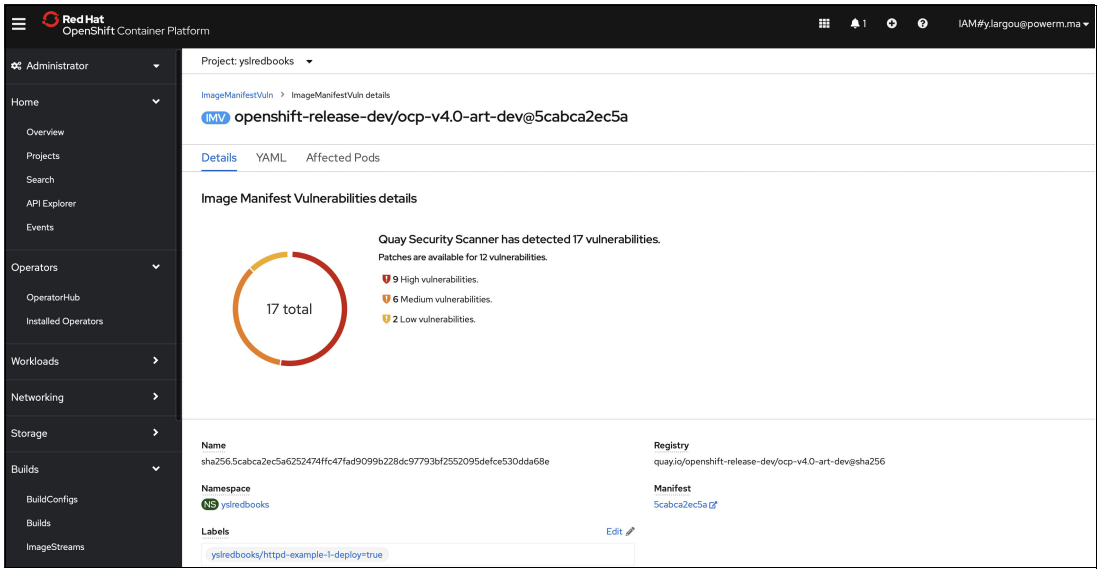


Figure 4-46 Example of a detected vulnerability

Querying image vulnerabilities by using the CLI

To display information about vulnerabilities that are detected by the Red Hat Quay Container Security Operator by using the CLI, run the following commands:

1. Connect to IBM Cloud Shell and use the *oc* command, as shown in Example 4-6.

Example 4-6 Connecting to IBM Cloud Shell and checking the oc version

```
Welcome to IBM Cloud Shell!
Image version: 1.0.66

Note: Your Cloud Shell session is running in Frankfurt (eu-de). Your workspace
includes 500 MB of temporary storage. This session will close after an hour of
inactivity. If you don't have any active sessions for an hour or you reach the
50-hour weekly usage limit, your workspace data is removed. To track your usage,
go to Usage quota in the Cloud Shell menu.

Tip: Enter 'ibmcloud' to use the IBM Cloud CLI. The Frankfurt (eu-de) region is
targeted by default. You can switch the region by running 'ibmcloud target -r
<region-name>'.

y_largou@cloudshell:~$ oc version
Client Version: 4.8.24
```

2. Log in with your token, as shown in Example 4-7.

Example 4-7 Logging in with a token

```
y_largou@cloudshell:~$ oc login
--token=sha256~muHF9j9O-BP3DIE7d1-OpZQVAHdYt54HKnBvYcKUrQY
--server=https://c108-e.eu-gb.containers.cloud.ibm.com:32759
Logged in to "https://c108-e.eu-gb.containers.cloud.ibm.com:32759" as
"IAM#y.largou@powerm.ma" using the token provided.
```

```
You have access to 65 projects. The list has been suppressed. You can list all
projects with 'oc projects'

Using project "default".
Welcome! See 'oc help' to get started.
```

3. Display the query for the detected container image vulnerabilities, as shown in Example 4-8.

Example 4-8 Querying a detected container image CVE

```
y_largou@cloudshell:~$ oc get vuln --all-namespaces
NAMESPACE                                          NAME AGE
openshift-cluster-node-tuning-operator
sha256.6f27078a90cb735b853447f75b0f33753c9a1412b76f73bdf86dad3e9cf72500   6h39m
openshift-cluster-samples-operator
sha256.ca2d1349a605d575b4c0c1ba9b5a7f77f806fea0d38cd5e319fce21716209f69   6h39m
openshift-cluster-storage-operator
sha256.1fd0804e7b69d5b5ba0fb5551a7fafd0624fc67b9e0ed3f78b5e7070cef0cf33   6h40m
openshift-cluster-storage-operator
sha256.3a9f0b56c58b5418fb920c69b7e9ffcb83ff569b6a60fdc51a65dd3b8ea000b5   6h40m
openshift-cluster-storage-operator
sha256.9cbd1a970a20c02c130117ea066fafd418404313c2bd84ff5ff0352d7aa44597   6h39m
openshift-cluster-storage-operator
sha256.c90f3233bf4d6a62c6276cd2dbff1671f9a07f20928b6c756b768434233e89db   6h39m
openshift-console-operator
sha256.0ef47bb656bc815321ab553b55c8a491e728a2699f5f6308db7de097055d3bc9   6h39m
openshift-console
sha256.2a8fba9dda24eb4a6ddd3ed7904291d6a8194f04443795d5f96c73c5ad633d6a   6h39m
openshift-console
sha256.6b296eb495c0bc44be45901cd86840bf71f97769f755a612de9d68356bc17202   6h40m
openshift-dns-operator
sha256.68f4064ea9725887f1adc61ee3c54a03816ac0d6e216288676b6f94d7560911d   6h39m
openshift-dns-operator
sha256.eb4689ba4b82e603bcd43ecddb2ad492358e1eb8cc773b52674684ef25d05eaa   6h39m
openshift-dns
sha256.2d3851b378f0ac7f9d65ef5b5773aede9e0fe1e31904d6154a36c45b57851697   6h39m
openshift-dns
sha256.4abe7f48249b4c580ec22bfe9d1612b4994c1dedfc081e74a7ca454bd1165a23   6h39m
openshift-dns
sha256.eb4689ba4b82e603bcd43ecddb2ad492358e1eb8cc773b52674684ef25d05eaa   6h39m
openshift-image-registry
sha256.911753c860d418b743eb83d7abe06690ecba448bf151bd1e84a5030cb56613b1   6h39m
openshift-image-registry
sha256.e25da438ab61fe32a4adc7a265ea66492c825e3ba5e296f568cc38d86d7c43d4   6h39m
openshift-ingress-canary
sha256.0858da96a5ef69714910e4564e37d30b137f942ac8845a49e9eb62f796f4c6f0   6h39m
openshift-ingress-operator
sha256.0858da96a5ef69714910e4564e37d30b137f942ac8845a49e9eb62f796f4c6f0   6h39m
openshift-ingress-operator
sha256.eb4689ba4b82e603bcd43ecddb2ad492358e1eb8cc773b52674684ef25d05eaa   6h39m
openshift-ingress
sha256.4f322f7c459883f44d1a3bd7dc03f634d3b0d63198cea47aa41d851c9887836e   6h39m
openshift-kube-proxy
sha256.e489cc0498d7a68daeb91437c8979fdb8d60d3a02eeff4073984bfd129fc4f91   6h39m
```

```
openshift-kube-proxy
sha256.eb4689ba4b82e603bcd43ecddb2ad492358e1eb8cc773b52674684ef25d05eaa   6h39m
openshift-kube-storage-version-migrator-operator
sha256.905a15539183385d346fea04cdd1ec4333ee335cef97b762a2dd40bc47c933b3   6h39m
openshift-kube-storage-version-migrator
sha256.0fcc6b95130e47c5144e5df7f398754c6e665d100f084dec3a4ba976d14d054d   6h39m
openshift-marketplace
sha256.22f1d6d0d3a5c0e53d705bf8f57e0d8f8bacd4b44ab3694a17a4e5c4a6b77836   6h39m
openshift-monitoring
sha256.09c45bb7f897b44e599f7f79959ca67b2b48145810bb1054dcca9c28e2e086a1   6h39m
openshift-monitoring
sha256.1929247501d11208a9bf36b1716f71b631df21a5209980152c6a5a18039e1b8b   6h39m
openshift-monitoring
sha256.25d4752df835bf37ff44e9c39d52a9df2d33f86bb552257c5a999bb77f1273f1   6h39m
openshift-monitoring
sha256.4ea395a3e84d2dadb00a499f253fb06aaeb5f8124e40ef8ed4bdcf8cc68b1a04   6h39m
openshift-monitoring
sha256.5649cb30106601e6932770c3c5df40dd09bc72f33adcf805a5d50e3bdd43b288   6h39m
openshift-monitoring
sha256.73718270eaa0ec5e70d6ca4abceba819ff43a05e0d6c01ce2ae861bf51d9b0b4   6h39m
openshift-monitoring
sha256.81217df9a0c4d235ce24294c89f982fcb07bc50bb03ae57c8734e09185873f54   6h39m
openshift-monitoring
sha256.82182248234d42228906e959e81134c094f9d134f4be9ea74d75667151970891   6h39m
openshift-monitoring
sha256.85ae72cd6081a1bd0877d9ad8a07a6cdae885a2509db1b99d53b865e8a6d9d4e   6h39m
openshift-monitoring
sha256.b2b092b5086f113550569c2844501ee52b3312a75c849f526c6e53d3a1f621c8   6h39m
openshift-monitoring
sha256.b9871fcd6fb02452c273c09d4d67032bd598fec2201cb95cf6616ac3d0a4192a   6h39m
openshift-monitoring
sha256.d33fd2ec339842e439169928b1331d165fc6866fa06f3198bcaea9561540e2ea   6h39m
openshift-monitoring
sha256.d3c3fa01291702621902ad24d2d63b945704952315718f9771165ba56d3c1290   6h39m
openshift-monitoring
sha256.eb4689ba4b82e603bcd43ecddb2ad492358e1eb8cc773b52674684ef25d05eaa   6h39m
openshift-monitoring
sha256.fb0d7eb7232992902c296531423c54fc4eaa54ad39ee12b3f697f4fcee2058af   6h39m
openshift-multus
sha256.24f33c4258d6177a3bdb1a7c859b1c199a7afffa4e04f5c7dc21149b9c2dc9d8   6h39m
openshift-multus
sha256.5096e4451e496e8eadee2a190928a00dc816d348e4a9126aaaf6425e6b56ed36   6h39m
openshift-multus
sha256.6f5422a05b49d68451debb485ec429351e78c8b200a44eded7b3d739a7ce3693   6h40m
openshift-multus
sha256.eb4689ba4b82e603bcd43ecddb2ad492358e1eb8cc773b52674684ef25d05eaa   6h40m
openshift-network-diagnostics
sha256.130ff7ce1fd91a71929d40832f8045c233070b0518a51c311f16a46126127383   6h39m
openshift-network-operator
sha256.130ff7ce1fd91a71929d40832f8045c233070b0518a51c311f16a46126127383   6h39m
openshift-operator-lifecycle-manager
sha256.acf9b47d66445709e9a3061c1b12f83e9104cf0e4ec4c3e2cb1c460a8430bcdc   6h39m
openshift-service-ca-operator
sha256.c2b9c99b65c949e9599fd5e61c4ac0c6a0ef9faa27811a97908028fdcebd3be4   6h39m
```

```
openshift-service-ca
sha256.c2b9c99b65c949e9599fd5e61c4ac0c6a0ef9faa27811a97908028fdcebd3be4   6h39m
y_largou@cloudshell:~$
```

4. Display the details for a specific vulnerability by using the command that is shown in Example 4-9.

Example 4-9 Displaying the details for a specific VCE

```
y_largou@cloudshell:~$ oc describe vuln --namespace openshift-dns
sha256.2d3851b378f0ac7f9d65ef5b5773aede9e0fe1e31904d6154a36c45b57851697
Name:         sha256.2d3851b378f0ac7f9d65ef5b5773aede9e0fe1e31904d6154a36c45b57851697
Namespace:    openshift-dns
Labels:       openshift-dns/dns-default-5ghqv=true
              openshift-dns/dns-default-6t6xd=true
              openshift-dns/dns-default-rtm92=true
Annotations:  <none>
API Version:  secscan.quay.redhat.com/v1alpha1
Kind:         ImageManifestVuln
Metadata:
  Creation Timestamp:  2022-11-01T16:56:32Z
  Generation:          7
  Managed Fields:
    API Version:  secscan.quay.redhat.com/v1alpha1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:labels:
          .:
          f:openshift-dns/dns-default-5ghqv:
          f:openshift-dns/dns-default-6t6xd:
          f:openshift-dns/dns-default-rtm92:
      f:spec:
        .:
        f:features:
        f:image:
        f:manifest:
    Manager:      security-labeller
    Operation:    Update
    Time:         2022-11-01T16:56:57Z
    API Version:  secscan.quay.redhat.com/v1alpha1
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        .:
        f:affectedPods:
          .:
          f:openshift-dns/dns-default-5ghqv:
          f:openshift-dns/dns-default-6t6xd:
          f:openshift-dns/dns-default-rtm92:
        f:fixableCount:
        f:highCount:
        f:highestSeverity:
        f:lastUpdate:
        f:lowCount:
        f:mediumCount:
```

```
    Manager:         security-labeller
    Operation:       Update
    Subresource:     status
    Time:            2022-11-01T16:56:57Z
  Resource Version:  246734
  UID:               ecfa3294-d2b8-408b-b451-e25cf456e32c
Spec:
  Features:
    Name:     bind-libs-lite
    Version:  32:9.11.26-4.el8_4
    Vulnerabilities:
      Description:  The Berkeley Internet Name Domain (BIND) is an implementation of the Domain
Name System (DNS) protocols. BIND includes a DNS server (named); a resolver library (routines
for applications to use when interfacing with DNS); and tools for verifying that the DNS server
is operating correctly.

Security Fix(es):

* bind: memory leak in ECDSA DNSSEC verification code (CVE-2022-38177)

* bind: memory leaks in EdDSA DNSSEC verification code (CVE-2022-38178)

For more details about the security issue(s), including the impact, a CVSS score,
acknowledgments, and other related information, refer to the CVE page(s) listed in the
References section.
      Fixedby:         32:9.11.26-4.el8_4.1
      Link:            https://access.redhat.com/errata/RHSA-2022:6779
https://access.redhat.com/security/cve/CVE-2022-38177
https://access.redhat.com/security/cve/CVE-2022-38178
      Metadata:        {"UpdatedBy": "RHEL8-rhel-8.4-eus", "RepoName":
"cpe:/a:redhat:rhel_eus:8.4::appstream", "RepoLink": null, "DistroName": "Red Hat Enterprise
Linux Server", "DistroVersion": "8", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H", "Score": 7.5}}}
      Name:            RHSA-2022:6779: bind security update (Important)
      Namespace Name:  RHEL8-rhel-8.4-eus
      Severity:        High
    Name:              python3-bind
    Version:           32:9.11.26-4.el8_4
    Vulnerabilities:
      Description:  The Berkeley Internet Name Domain (BIND) is an implementation of the Domain
Name System (DNS) protocols. BIND includes a DNS server (named); a resolver library (routines
for applications to use when interfacing with DNS); and tools for verifying that the DNS server
is operating correctly.

Security Fix(es):

* bind: memory leak in ECDSA DNSSEC verification code (CVE-2022-38177)

* bind: memory leaks in EdDSA DNSSEC verification code (CVE-2022-38178)

For more details about the security issue(s), including the impact, a CVSS score,
acknowledgments, and other related information, refer to the CVE page(s) listed in the
References section.
      Fixedby:         32:9.11.26-4.el8_4.1
```

```
      Link:            https://access.redhat.com/errata/RHSA-2022:6779
https://access.redhat.com/security/cve/CVE-2022-38177
https://access.redhat.com/security/cve/CVE-2022-38178
      Metadata:        {"UpdatedBy": "RHEL8-rhel-8.4-eus", "RepoName":
"cpe:/a:redhat:rhel_eus:8.4::appstream", "RepoLink": null, "DistroName": "Red Hat Enterprise
Linux Server", "DistroVersion": "8", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H", "Score": 7.5}}}
      Name:            RHSA-2022:6779: bind security update (Important)
      Namespace Name:  RHEL8-rhel-8.4-eus
      Severity:        High
    Name:              openshift4/ose-coredns
    Version:           v4.10.0-202208241855.p0.g3ec1ee7.assembly.stream
    Vulnerabilities:
      Description:     Non-random values for ticket_age_add in session tickets in crypto/tls
before Go 1.17.11 and Go 1.18.3 allow an attacker that can observe TLS handshakes to correlate
successive connections by comparing ticket ages during session resumption.
      Fixedby:         v4.11.0-202208031306.p0.g7fe212f.assembly.stream
      Link:            https://access.redhat.com/errata/RHSA-2022:6103
https://access.redhat.com/security/cve/CVE-2022-30629
      Metadata:        {"UpdatedBy": "rhel-container-updater", "RepoName": "Red Hat Container
Catalog", "RepoLink": "https://catalog.redhat.com/software/containers/explore", "DistroName":
"", "DistroVersion": "", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N", "Score": 3.1}}}
      Name:            RHSA-2022:6103
      Namespace Name:  rhel-container-updater
      Severity:        Low
    Name:              bind-libs
    Version:           32:9.11.26-4.el8_4
    Vulnerabilities:
      Description:  The Berkeley Internet Name Domain (BIND) is an implementation of the Domain
Name System (DNS) protocols. BIND includes a DNS server (named); a resolver library (routines
for applications to use when interfacing with DNS); and tools for verifying that the DNS server
is operating correctly.

Security Fix(es):

* bind: memory leak in ECDSA DNSSEC verification code (CVE-2022-38177)

* bind: memory leaks in EdDSA DNSSEC verification code (CVE-2022-38178)

For more details about the security issue(s), including the impact, a CVSS score,
acknowledgments, and other related information, refer to the CVE page(s) listed in the
References section.
      Fixedby:         32:9.11.26-4.el8_4.1
      Link:            https://access.redhat.com/errata/RHSA-2022:6779
https://access.redhat.com/security/cve/CVE-2022-38177
https://access.redhat.com/security/cve/CVE-2022-38178
      Metadata:        {"UpdatedBy": "RHEL8-rhel-8.4-eus", "RepoName":
"cpe:/a:redhat:rhel_eus:8.4::appstream", "RepoLink": null, "DistroName": "Red Hat Enterprise
Linux Server", "DistroVersion": "8", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H", "Score": 7.5}}}
      Name:            RHSA-2022:6779: bind security update (Important)
      Namespace Name:  RHEL8-rhel-8.4-eus
      Severity:        High
    Name:              pip
```

```
    Version:           9.0.3
    Vulnerabilities:
      Description:     The pip package before 19.2 for Python allows Directory Traversal when a
URL is given in an installation command because a Content-Disposition header can have ../ in a
file name, as demonstrated by overwriting the /root/.ssh/authorized_keys file. This occurs in
_download_http_url in _internal/download.py.
      Metadata:        {"UpdatedBy": "pyupio", "RepoName": "pypi", "RepoLink":
"https://pypi.org/simple", "DistroName": "", "DistroVersion": "", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N", "Score": 7.5}}}
      Name:            pyup.io-38765 (CVE-2019-20916)
      Namespace Name:  pyupio
      Severity:        High
      Description:     Pip 21.1 updates its dependency 'urllib3' to v1.26.4 due to security
issues.
      Metadata:        {"UpdatedBy": "pyupio", "RepoName": "pypi", "RepoLink":
"https://pypi.org/simple", "DistroName": "", "DistroVersion": "", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N", "Score": 6.5}}}
      Name:            pyup.io-40291 (CVE-2021-28363)
      Namespace Name:  pyupio
      Severity:        Medium
      Description:     A flaw was found in python-pip in the way that it handled Unicode
separators in Git references. A remote attacker could possibly use this issue to install a
different revision on a repository. The highest threat from this vulnerability is to data
integrity. This is fixed in python-pip version 21.1.
      Metadata:        {"UpdatedBy": "pyupio", "RepoName": "pypi", "RepoLink":
"https://pypi.org/simple", "DistroName": "", "DistroVersion": "", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:N/I:H/A:N", "Score": 5.7}}}
      Name:            pyup.io-42559 (CVE-2021-3572)
      Namespace Name:  pyupio
      Severity:        Medium
    Name:              bind-utils
    Version:           32:9.11.26-4.el8_4
    Vulnerabilities:
      Description:  The Berkeley Internet Name Domain (BIND) is an implementation of the Domain
Name System (DNS) protocols. BIND includes a DNS server (named); a resolver library (routines
for applications to use when interfacing with DNS); and tools for verifying that the DNS server
is operating correctly.

Security Fix(es):

* bind: memory leak in ECDSA DNSSEC verification code (CVE-2022-38177)

* bind: memory leaks in EdDSA DNSSEC verification code (CVE-2022-38178)

For more details about the security issue(s), including the impact, a CVSS score,
acknowledgments, and other related information, refer to the CVE page(s) listed in the
References section.
      Fixedby:         32:9.11.26-4.el8_4.1
      Link:            https://access.redhat.com/errata/RHSA-2022:6779
https://access.redhat.com/security/cve/CVE-2022-38177
https://access.redhat.com/security/cve/CVE-2022-38178
      Metadata:        {"UpdatedBy": "RHEL8-rhel-8.4-eus", "RepoName":
"cpe:/a:redhat:rhel_eus:8.4::appstream", "RepoLink": null, "DistroName": "Red Hat Enterprise
Linux Server", "DistroVersion": "8", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H", "Score": 7.5}}}
```

```
      Name:            RHSA-2022:6779: bind security update (Important)
      Namespace Name:  RHEL8-rhel-8.4-eus
      Severity:        High
    Name:              urllib3
    Version:           1.24.2
    Vulnerabilities:
      Description:     urllib3 before 1.25.9 allows CRLF injection if the attacker controls the
HTTP request method, as demonstrated by inserting CR and LF control characters in the first
argument of putrequest(). NOTE: this is similar to CVE-2020-26116.
      Metadata:        {"UpdatedBy": "pyupio", "RepoName": "pypi", "RepoLink":
"https://pypi.org/simple", "DistroName": "", "DistroVersion": "", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:N", "Score": 7.2}}}
      Name:            pyup.io-38834 (CVE-2020-26137)
      Namespace Name:  pyupio
      Severity:        High
      Description:     Urllib3 1.26.5 includes a fix for CVE-2021-33503: An issue was discovered
in urllib3 before 1.26.5. When provided with a URL containing many @ characters in the authority
component, the authority regular expression exhibits catastrophic backtracking, causing a denial
of service if a URL were passed as a parameter or redirected to through an HTTP redirect.
https://github.com/advisories/GHSA-q2q7-5pp4-w6pg
      Metadata:        {"UpdatedBy": "pyupio", "RepoName": "pypi", "RepoLink":
"https://pypi.org/simple", "DistroName": "", "DistroVersion": "", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H", "Score": 7.5}}}
      Name:            pyup.io-43975 (CVE-2021-33503)
      Namespace Name:  pyupio
      Severity:        High
    Name:              bind-license
    Version:           32:9.11.26-4.el8_4
    Vulnerabilities:
      Description:  The Berkeley Internet Name Domain (BIND) is an implementation of the Domain
Name System (DNS) protocols. BIND includes a DNS server (named); a resolver library (routines
for applications to use when interfacing with DNS); and tools for verifying that the DNS server
is operating correctly.

Security Fix(es):

* bind: memory leak in ECDSA DNSSEC verification code (CVE-2022-38177)

* bind: memory leaks in EdDSA DNSSEC verification code (CVE-2022-38178)

For more details about the security issue(s), including the impact, a CVSS score,
acknowledgments, and other related information, refer to the CVE page(s) listed in the
References section.
      Fixedby:         32:9.11.26-4.el8_4.1
      Link:            https://access.redhat.com/errata/RHSA-2022:6779
https://access.redhat.com/security/cve/CVE-2022-38177
https://access.redhat.com/security/cve/CVE-2022-38178
      Metadata:        {"UpdatedBy": "RHEL8-rhel-8.4-eus", "RepoName":
"cpe:/a:redhat:rhel_eus:8.4::appstream", "RepoLink": null, "DistroName": "Red Hat Enterprise
Linux Server", "DistroVersion": "8", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H", "Score": 7.5}}}
      Name:            RHSA-2022:6779: bind security update (Important)
      Namespace Name:  RHEL8-rhel-8.4-eus
      Severity:        High
    Name:              expat
```

```
    Version:           2.2.5-4.el8_4.3
    Vulnerabilities:
      Description:  Expat is a C library for parsing XML documents.

Security Fix(es):

* expat: a use-after-free in the doContent function in xmlparse.c (CVE-2022-40674)

For more details about the security issue(s), including the impact, a CVSS score,
acknowledgments, and other related information, refer to the CVE page(s) listed in the
References section.
      Fixedby:         0:2.2.5-4.el8_4.4
      Link:            https://access.redhat.com/errata/RHSA-2022:6831
https://access.redhat.com/security/cve/CVE-2022-40674
      Metadata:        {"UpdatedBy": "RHEL8-rhel-8.4-eus", "RepoName":
"cpe:/a:redhat:rhel_eus:8.4::appstream", "RepoLink": null, "DistroName": "Red Hat Enterprise
Linux Server", "DistroVersion": "8", "NVD": {"CVSSv3": {"Vectors":
"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H", "Score": 9.8}}}
      Name:            RHSA-2022:6831: expat security update (Important)
      Namespace Name:  RHEL8-rhel-8.4-eus
      Severity:        High
  Image:               quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256
  Manifest:            sha256:2d3851b378f0ac7f9d65ef5b5773aede9e0fe1e31904d6154a36c45b57851697
Status:
  Affected Pods:
    openshift-dns/dns-default-5ghqv:
      cri-o://5005f4a4aa27685d2c4cb4d018f0475abb801967f6779a413a766e27a4f46dd8
    openshift-dns/dns-default-6t6xd:
      cri-o://5598633bb47c26fe5945c8807029238d97a4243fd2d9e49d337115abad82a3b0
    openshift-dns/dns-default-rtm92:
      cri-o://8fc2ac16f021cb339b862fbaa7fa83d898bbcc61cbb752aba2961da7574135d3
  Fixable Count:     7
  High Count:        9
  Highest Severity:  High
  Last Update:       2022-11-01 23:26:48.135845459 +0000 UTC
  Low Count:         1
  Medium Count:      2
Events:              <none>
y_largou@cloudshell:~$
```

Using Red Hat Advanced Cluster Security for Kubernetes

Building on the foundation of security that Red Hat OpenShift provides, Red Hat Advanced Cluster Security for K8s helps to automate DevOps so the developers can mitigate security issues such as image vulnerabilities early in the container lifecycle; identify workload misconfiguration like excessive access permissions to further reduce the attack surface and risk profile; and help security teams controlling runtime security to detect and respond to threats, such as unauthorized access and privilege escalation.

For more information, see Red Hat Advanced Cluster Security for Kubernetes.

In the following use case, we add extra tasks to an existing pipeline by using the pipeline builder to run a full vulnerability image scan task.

Complete the following steps:

1. Verify that the Red Hat Advanced Cluster Security for Kubernetes Operator is installed by selecting **Operators** → **Installed Operators**, as shown in Figure 4-47.

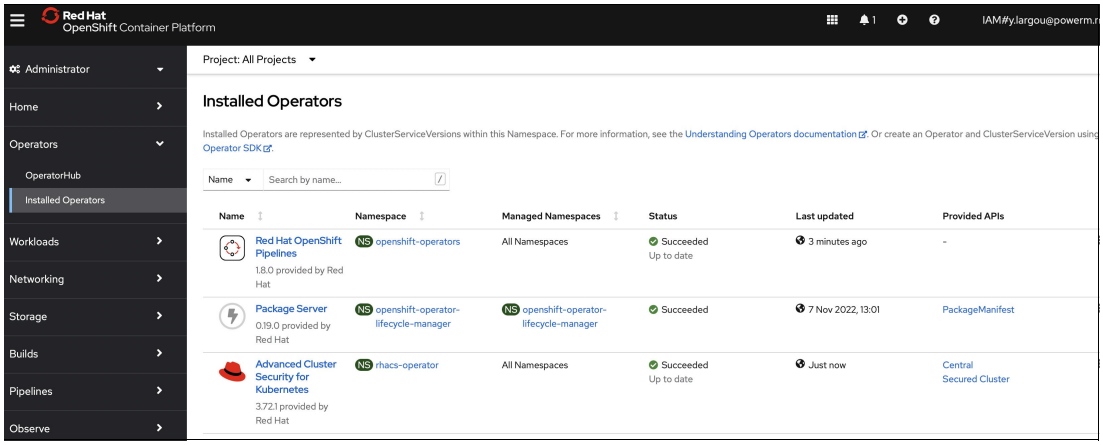


Figure 4-47 Verifying Installed Operators

2. Ensure that the Red Hat Advanced Cluster Security for Kubernetes Operator Central and Secured Cluster custom resources are configured as described in the Red Hat documentation.[14]

3. In the web console, go to **Pipelines** → **Pipelines**, as shown in Figure 4-48.

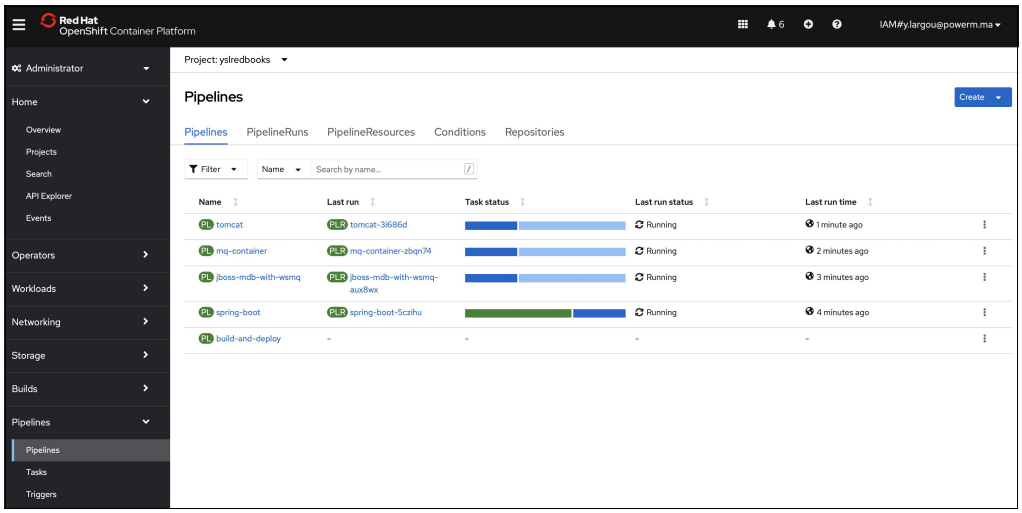


Figure 4-48 Pipelines page

4. Select the spring-boot pipeline, and then select **Actions** → **Edit Pipeline**, as shown in Figure 4-49 on page 91.

[14] https://docs.openshift.com/acs/3.72/installing/install-ocp-operator.html

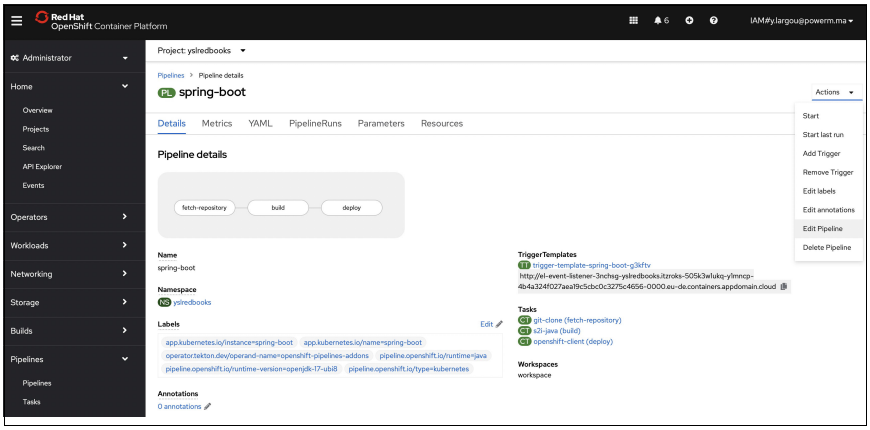


Figure 4-49 Selecting a pipeline

5. Add a sequential task after the build task, as shown in Figure 4-50.

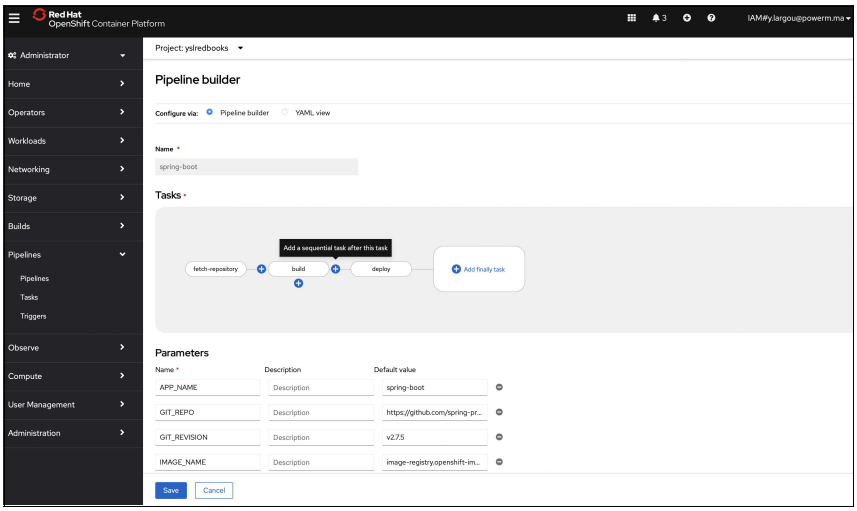


Figure 4-50 Adding a sequential task

6. Select the rhacs-image-scan task, click **Add**, as shown in Figure 4-51, and then click **Save**.

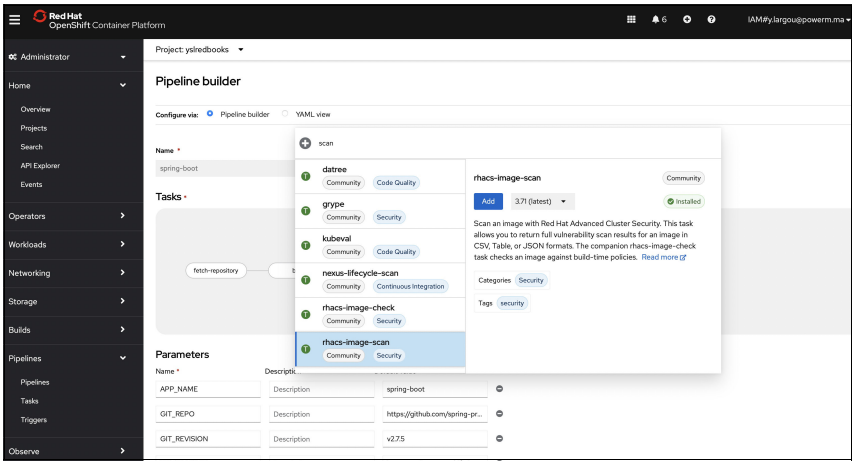


Figure 4-51 Adding the rhacs-image-scan task

This task returns full vulnerability scan results for an image in JSON, CSV, or Pretty format.

Add the following information when you create this task:

- rox_central_endpoint
- rox_api_token
- image

7. Select **Actions** → **Start**, as shown in Figure 4-52.

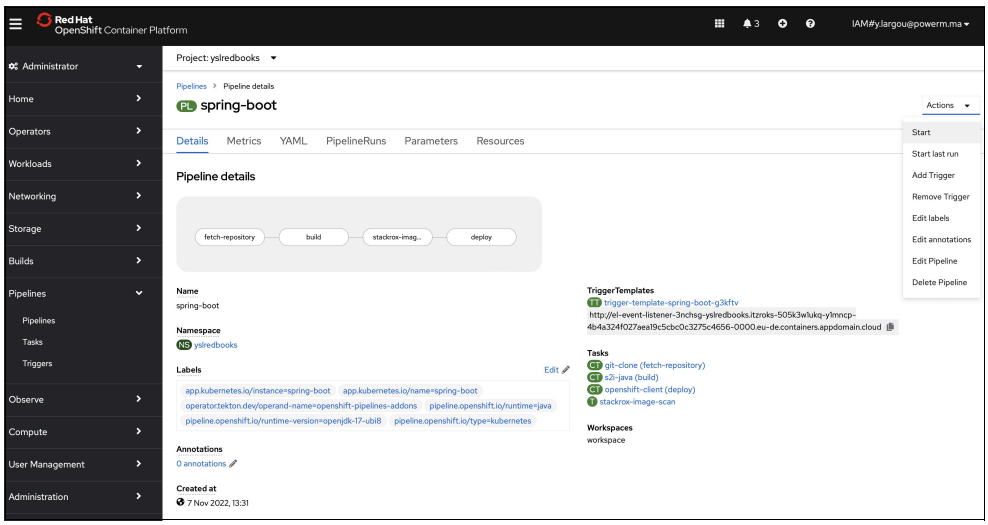


Figure 4-52 Starting the pipeline

8. Monitor the pipeline, as shown in Figure 4-53.

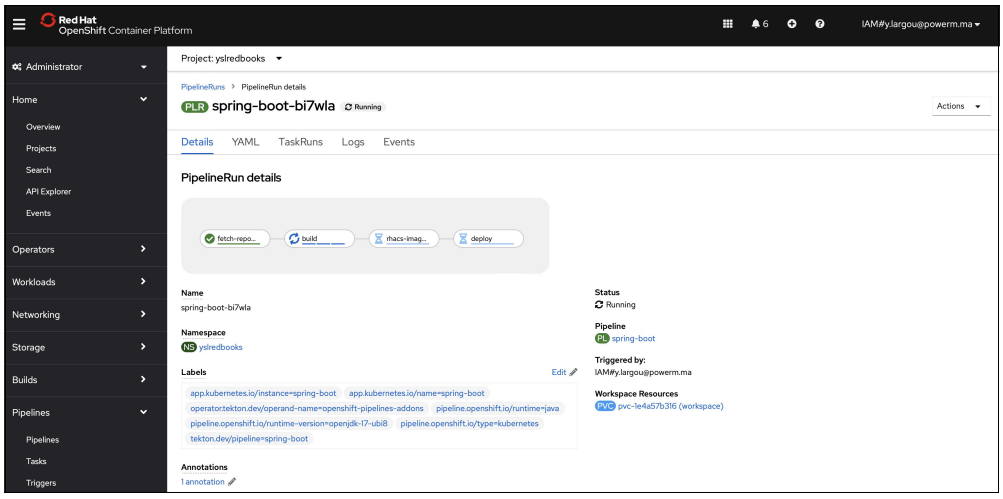


Figure 4-53 Pipeline run details

You can check Policy violation by severity, images, and deployments at most risk by accessing the Red Hat Advanced Cluster Security For Kubernetes dashboard, as shown in Figure 4-54 on page 93.

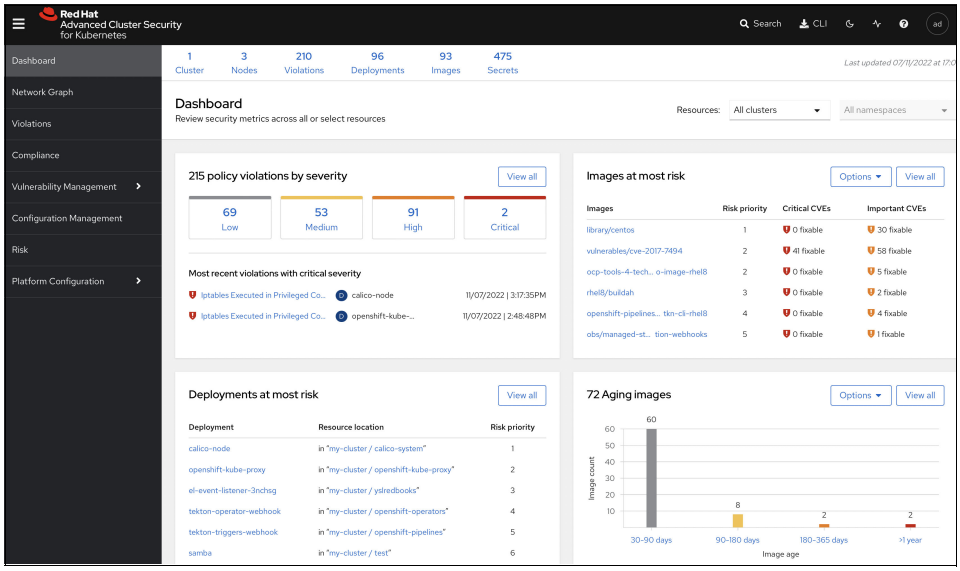


Figure 4-54 Red Hat Advanced Cluster Security For Kubernetes dashboard

4.2.13 Enhanced data resilience and security by using IBM Spectrum Protect Plus

Data protection requirements have evolved from standard data backup and recovery solutions to a new data resilience approach that provides continuous access to data and applications while protecting critical information from a system or application failure, human error, a security breach, or a catastrophic disaster. Organizations are struggling with the cost and complexity of protecting their data as they embrace the new digital transformation, manage massive data growth, and tackle the requirements of always-on services. Moreover, VMs and containerized workloads are more prevalent. Thus, modern data resilience solutions must be designed to explicitly operate both on-premises and in the cloud, and these solutions are essential.

IBM Spectrum Protect Plus is a data resilience solution that provides data protection, recovery, replication, and reuse for VMs, databases, applications, file systems, software as a service (SaaS) workloads, containers, and cloud environments.

IBM Spectrum Protect Plus 10.1.10 delivers support across multiple container-based environments:

- K8s and Container Storage Interface (CSI) snapshots
- Red Hat OpenShift Container Platform
- Red Hat OpenShift Data Foundation
- Red Hat OpenShift Virtualization (providing VM support in a Red Hat environment)
- Red Hat OpenShift both on-premises and in the cloud

IBM Spectrum Protect Plus protects PVs along with `etcd` *d*ata and metadata in the container and uses in-place storage snap copies for instant recovery of containers. When leveraging IBM Spectrum Scale, you can create application and crash-consistent backup copies of the data.

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

Here are best practices to secure container backups when using IBM Spectrum Protect Plus:

- Enable data at rest encryption to protect all sensitive data, including the copy backup data and Container Backup Support secrets.
- Ensure that secrets are encrypted when stored in the cluster `etcd` database.
- Deploy an IBM Spectrum Protect Plus vSnap server to enable encryption.
- Create backup requests that specify encryption-enabled SLAs so that data can be directed to a vSnap server for encryption if the vSnap server is enabled for encryption of data at rest.
- Starting from IBM Spectrum Protect Plus 10.1.9, container backups can go directly to object storage or cloud storage without requiring an IBM Spectrum Protect Plus vSnap server.

  Note: The only supported object store types are IBM Cloud Object Store, Microsoft Azure Blob Storage, and AWS S3.

  Backups on object storage are encrypted by default, and the encryption password is stored as an attribute of the SLA policy.
- Verify that the Container Backup Support installation files signature that is included with the installation package against the suitable signature and certificates.

Using an IBM Spectrum Protect Plus SLA policy to encrypt backups to IBM Cloud Object Storage

In this scenario, we use an IBM Spectrum Protect Plus SLA policy to encrypt backups to IBM Cloud Object Storage. The Red Hat OpenShift cluster is deployed on IBM PowerVS.

Complete the following steps:

1. Before configuring IBM Spectrum Protect Plus, gather information, such as IBM Cloud Object Storage endpoints, the access key, the secret key, and the certificate, from the IBM Cloud console.
2. Log in to the IBM Cloud console by using your credentials, select **Resource List** → **Storage**, and choose the cloud storage object, as shown in Figure 4-55.

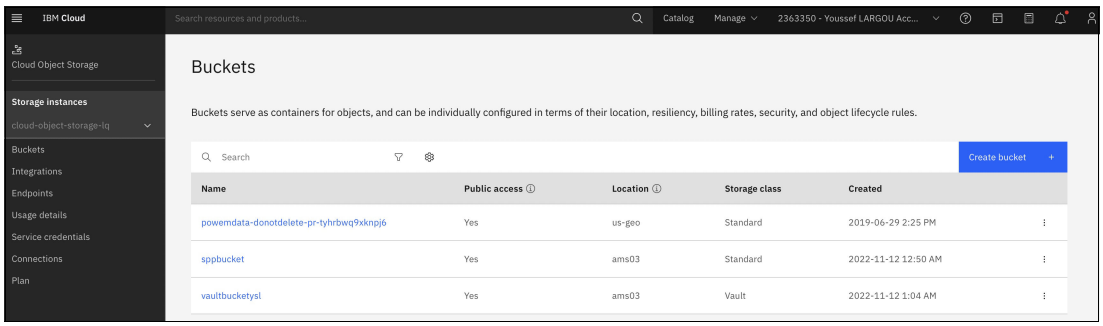


Figure 4-55 Selecting the cloud storage object

3. Go to **Service Credentials** and select `access_key_id` and `secret_access_key`, as shown in Figure 4-56 on page 95.

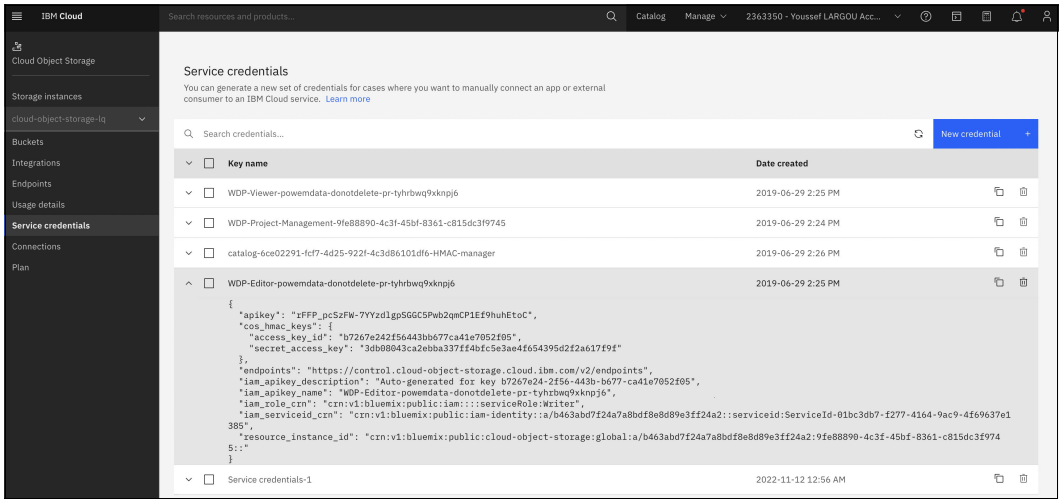


Figure 4-56 Selecting the access and secret keys

4. Verify that the IBM Spectrum Protect Plus Operator is installed, as shown in Figure 4-57.

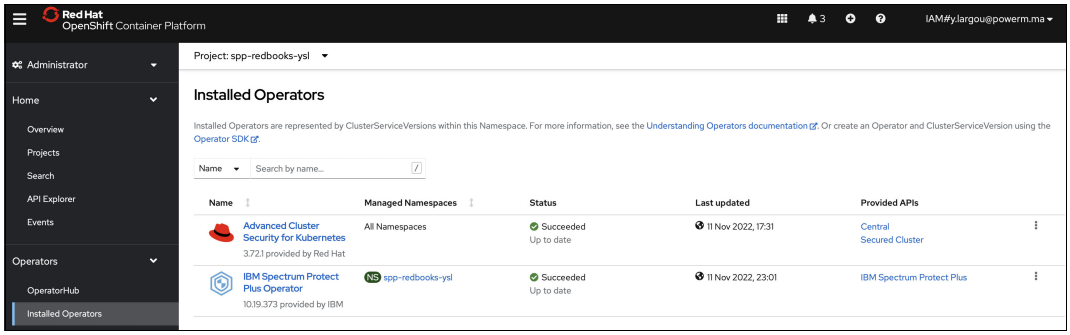


Figure 4-57 Verifying the IBM Spectrum Protect Plus Operator installation

5. Verify the Containers and pods topology, as shown in Figure 4-58.

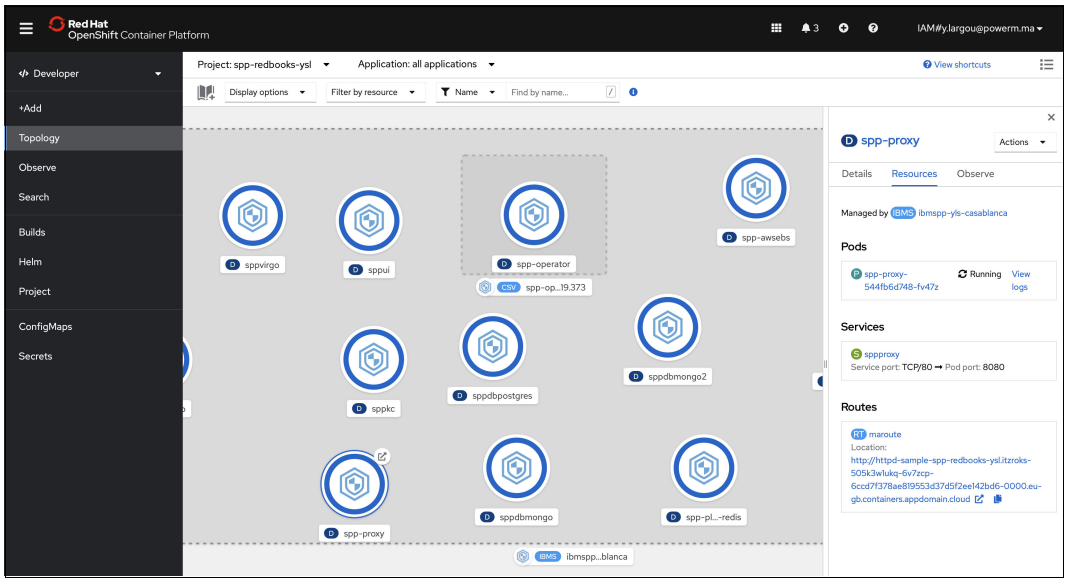


Figure 4-58 Pods topology

6. Verify the PVs and PVCs that define the data locations, as shown in Example 4-10.

Example 4-10 Verifying the PVs and PVCs

```
y_largou@cloudshell:~$ oc project spp-powertst-ysl
Already on project "spp-powertst-ysl" on server "https://c108-e.eu-gb.containers.cloud.ibm.com:31539".

y_largou@cloudshell:~$ oc get pv |grep -E "NAME|spp"
NAME                           CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLASS     REASON   AGE
pvc-03eb59a1-1391-47cf-8e97-ea02d323141a   20Gi       RWO            Delete           Bound
spp-powertst-ysl/postgresql                                                    18h
pvc-08b215d3-2ae3-4c43-b652-93af2aeece50   50Gi       RWO            Delete           Bound
spp-powertst-ysl/sppdbmongo-spp-powertst-ysl-claim                             13h
pvc-16b6718f-142c-4e19-909e-38ec9cfd2258   20Gi       RWO            Delete           Bound
spp-powertst-ysl/sppvadp-spp-powertst-ysl-claim                                13h
pvc-30f4dc30-6ed0-4964-8766-9a2e2fb28027   100Gi      RWO            Delete           Bound
spp-powertst-ysl/sppdbmongo2-spp-powertst-ysl-claim                            13h
pvc-8c41f60d-3fd9-47f2-b938-d81eb0b1fc5c   150Gi      RWO            Delete           Bound
spp-powertst-ysl/virgo-lucene-spp-powertst-ysl-claim                           13h
pvc-ca48d7fe-1c5f-4860-93bb-153c2f7cab03   20Gi       RWO            Delete           Bound
spp-powertst-ysl/sppnodejs-spp-powertst-ysl-claim                              13h
pvc-cd12f108-7939-4361-afac-acd1a67114fb   20Gi       RWO            Delete           Bound
spp-powertst-ysl/sppdbpostgres-spp-powertst-ysl-claim                          13h
pvc-f0a45b12-5443-44e1-bc82-0c209a7bc123   20Gi       RWO            Delete           Bound
spp-powertst-ysl/spp-log-spp-powertst-ysl-claim
```

7. Validate that the services were created and are running for an IBM Spectrum Protect Plus server deployment. The services are shown in Example 4-11.

Example 4-11 List of IBM Spectrum Protect Plus services

```
y_largou@cloudshell:~$ oc get services | grep spp
spp                    ClusterIP   172.21.191.7     <none>        8082/TCP,5672/TCP,5671/TCP   13h
spp-operator-metrics   ClusterIP   172.21.114.34    <none>        8686/TCP,8383/TCP            13h
sppmanager             ClusterIP   172.21.133.8     <none>        80/TCP                       13h
sppproxy               ClusterIP   172.21.226.140   <none>        80/TCP                       13h
```

8. Verify the routes that access the containerized application from outside the cluster. An example output is shown in Example 4-12.

Example 4-12 Verifying the routes

```
y_largou@cloudshell:~$ oc get routes
NAME          HOST/PORT
PATH   SERVICES             PORT   TERMINATION   WILDCARD
devfile-sample-git
devfile-sample-git-spp-powertst-ysl.itzroks-505k3w1ukq-6v7zcp-6ccd7f378ae819553d37d5f2ee142bd6-0
000.eu-gb.containers.appdomain.cloud          devfile-sample-git   3001                 None
maroute
httpd-sample-spp-powertst-ysl.itzroks-505k3w1ukq-6v7zcp-6ccd7f378ae819553d37d5f2ee142bd6-0000.eu
-gb.containers.appdomain.cloud                sppproxy             8080                 None
```

9. Login to IBM Spectrum Protect Plus Admin console by using the following URL, as shown in Figure 4-59 on page 97:

   `httpd-sample-spp-powertst-ysl.itzroks-505k3w1ukq-6v7zcp-6ccd7f378ae819553d37d5f2ee142bd6-0000.eu-gb.containers.appdomain.cloud`

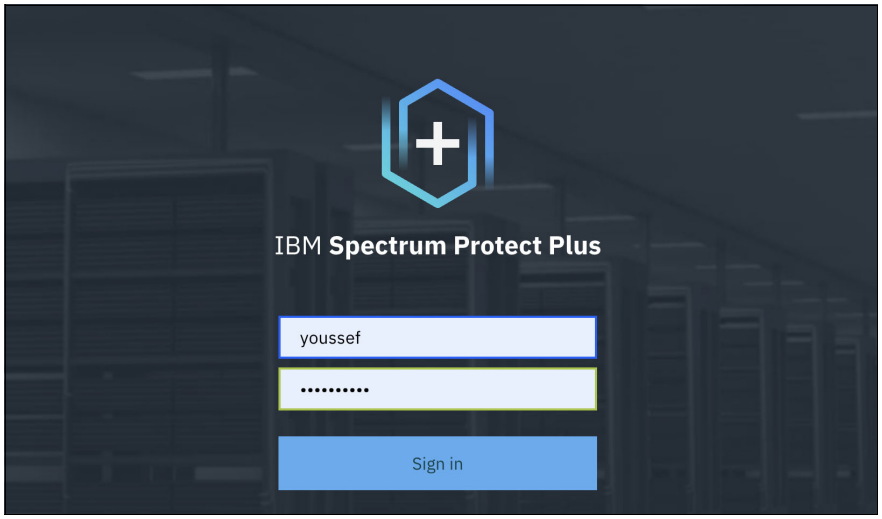


Figure 4-59 IBM Spectrum Protect Plus login page

10. Add cloud storage by selecting **System Configuration** → **Storage**, as shown in Figure 4-60.

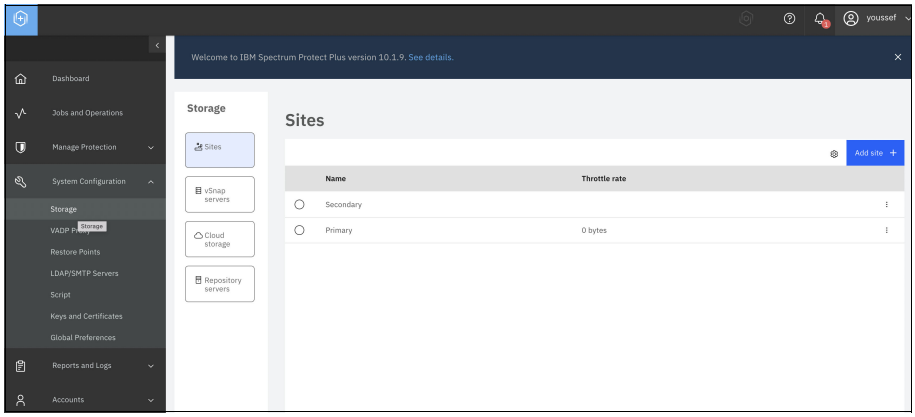


Figure 4-60 Adding cloud storage

11. Select **IBM Cloud Object Storage**, as shown in Figure 4-61.

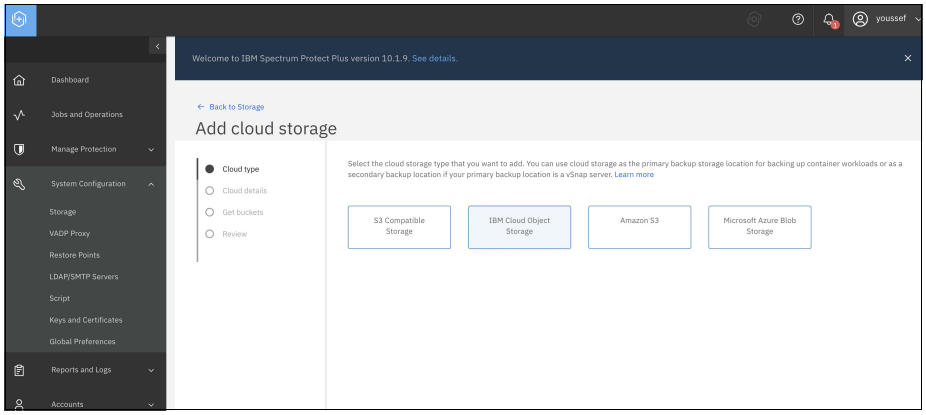


Figure 4-61 Selecting IBM Cloud Object Storage

12. Specify the IBM Cloud Object name and key name, and then provide the access and secret keys from step 3 on page 94, as shown in Figure 4-62.

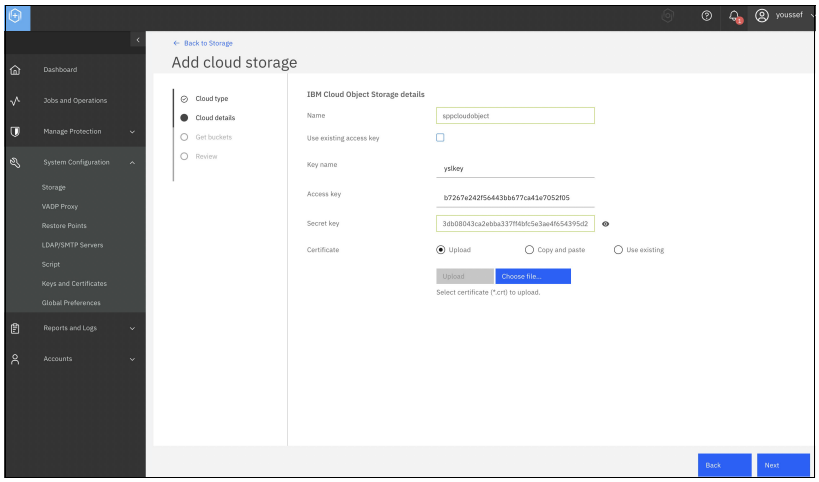


Figure 4-62 Creating a cloud object

13. Enter the endpoint to update the buckets, and choose the backup storage bucket and additional copy buckets, as shown in Figure 4-63.

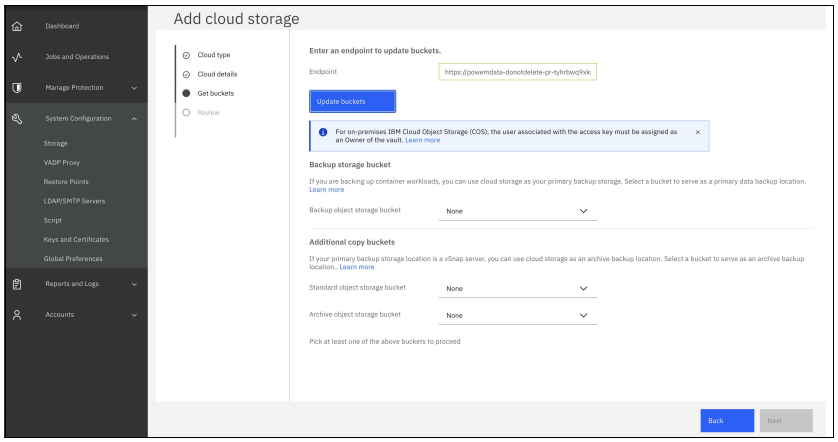


Figure 4-63 Choosing the backup storage buckets

14. Create an SLA policy by completing the following steps:

    a. In the navigation pane, select **Manage Protection** → **Policy Overview**, as shown in Figure 4-64 on page 99.

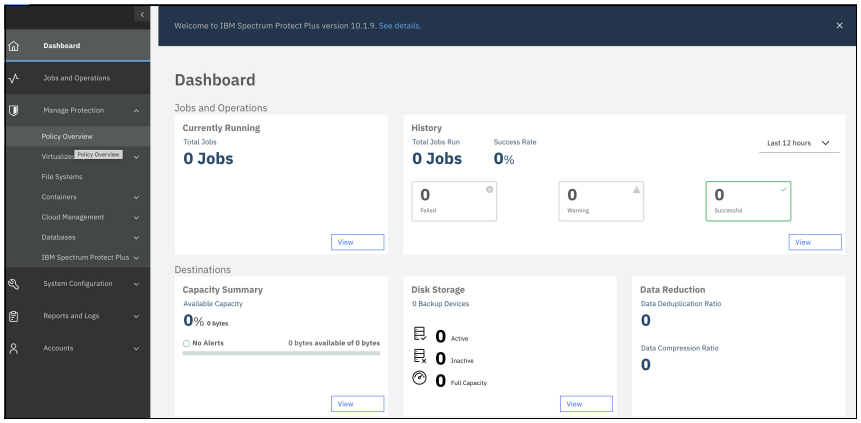


Figure 4-64 IBM Spectrum Protect Plus Policy Overview window

b. Click **Add SLA Policy**.

c. The New SLA Policy pane opens. Complete the following steps:

- In the Name field, enter a name, for example, `redbookgold`.
- Select **Kubernetes** → **OpenShift**.

  The SLA policy options for K8s or Red Hat OpenShift clusters are displayed.
- In the Snapshot Protection section, set the following options for snapshot operations: Retention, Disable Schedule, Repeats, Start Time, and Snapshot Prefix.
- In the Backup Policy section, set the following options for copy backup operations to cloud storage: Backup Storage, Retention, Disable Schedule, Repeats Start Time. For Target Site, select **Object Storage**, and for Target, select **sppcloudobject**.

d. Complete the following fields:

- Use existing backup encryption passphrase.
- Backup encryption passphrase name.
- Backup encryption passphrase.

e. Click **Save**. The SLA policy that you created is displayed in the table in the SLA Policies pane.

With this configuration, you can back up the IBM Spectrum Protect Plus Server Catalog and restore data that has been backed up to IBM Cloud Object Storage.

5

Authentication and authorization

A significant part of security management is understanding who is accessing your environment and controlling what that user can do. You must securely validate and authenticate the user, and limit that user to only functions to which they are authorized.

This chapter describes the following topics:

- Understanding authentication
- RBAC setup for users and service accounts

5.1 Understanding authentication

For users to interact with Red Hat OpenShift Container Platform, first they must authenticate to the cluster. The authentication layer identifies the user that is associated with requests to the Red Hat OpenShift Container Platform application programming interface (API). The authorization layer uses information about the requesting user to determine whether the request is allowed. The cluster administrator is responsible for configuring authentication for Red Hat OpenShift Container Platform.

5.1.1 Users

A *user* in Red Hat OpenShift Container Platform is an entity that can make requests to the Red Hat OpenShift Container Platform API. A Red Hat OpenShift Container Platform User object represents an actor that can be granted permissions in the system by adding roles to them or to their groups. Typically, this object represents the account of a developer or administrator that is interacting with Red Hat OpenShift Container Platform.

Several types of users can exist:

- Regular users
- System users
- Service accounts

Regular users

Regular users are how the most interactive Red Hat OpenShift Container Platform users are represented. Regular users are created automatically in the system on first login, or they can be created by using the API. Regular users are represented with the User object.

An example of a regular user is `joe alice`.

System users

Many *system users* are created automatically when the infrastructure is defined, mainly for enabling the infrastructure to interact with the API securely. They include a cluster administrator (with access to everything), a per-node user, users for use by routers and registries, and various others. Finally, there is an anonymous system user that is used by default for unauthenticated requests.

An example of a system user is `system:admin system:openshift-registry system:node:node1.example.com`.

Service accounts

Service accounts are special system users that are associated with projects. Some of these users are created automatically when the project is created. Project administrators can create more of these users to define access to the contents of each project. Service accounts are represented with the ServiceAccount object.

An example of service account is `system:serviceaccount:default:deployer system:serviceaccount:foo:builder`.

5.1.2 Groups

A user can be assigned to one or more *groups*, each of which represent a certain set of users. Groups are useful when managing authorization policies to grant permissions to multiple users at once, for example, granting access to objects within a project versus granting them to users individually.

In addition to explicitly defined groups, there are also *system groups*, or *virtual groups*, that are automatically provisioned by the cluster.

The default virtual groups that are shown in Table 5-1 are important.

Table 5-1 Virtual groups defined in Red Hat OpenShift

Virtual group	Description
`system:authenticated`	Automatically associated with all authenticated users.
`system:authenticated:oauth`	Automatically associated with all users that are authenticated with an OAuth access token.
`system:unauthenticated`	Automatically associated with all unauthenticated users.

5.1.3 API authentication

Requests to the Red Hat OpenShift Container Platform API are authenticated by using the following methods:

- OAuth access tokens
- X.509 client certificates

OAuth access tokens

- Obtained from the Red Hat OpenShift Container Platform OAuth server by using the `<namespace_route>/oauth/authorize` and `<namespace_route>/oauth/token` endpoints.
- Sent as an authorization.
- Sent as a websocket subprotocol header in the form `base64url.bearer.authorization.k8s.io.<base64url-encoded-token>` for websocket requests.

X.509 client certificates

- Requires an HTTPS connection to the apiserver.
- Verified by the apiserver against a trusted certificate authority (CA) bundle.
- The apiserver creates and distributes certificates to controllers to authenticate themselves.

Any request with an invalid access token or an invalid certificate is rejected by the authentication layer with a 401 error.

If no access token or certificate is presented, the authentication layer assigns the `system:anonymous` virtual user and the `system:unauthenticated` virtual group to the request. With these designations, the authorization layer can determine which requests, if any, an anonymous user might make.

5.1.4 Red Hat OpenShift Container Platform OAuth server

The Red Hat OpenShift Container Platform master includes a built-in OAuth server. Users obtain OAuth access tokens to authenticate themselves to the API.

When a person requests a new OAuth token, the OAuth server uses the configured identity provider to determine the identity of the person making the request. Then, it determines what user to which the identity maps, creates an access token for that user, and returns the token for use.

Every request for an OAuth token must specify the OAuth client that will receive and use the token. The following OAuth clients are automatically created when starting the Red Hat OpenShift Container Platform API:

- The openshift-browser-client client
- The openshift-challenging-client client

The openshift-browser-client client

Requests tokens at `<namespace_route>/oauth/token/request` with a user-agent that can handle interactive logins. `<namespace_route>` refers to the namespace route, which you can find by running the following command:

```
oc get route oauth-openshift -n openshift-authentication -o json | jq .spec.host
```

The openshift-challenging-client client

Requests tokens with a user-agent that can handle WWW-Authenticate challenges.

You can configure default options for the internal OAuth server's token duration. The default setting is 24 hours, after which existing sessions expire. If the default time is insufficient, then it can be modified (for more information, see Configuring the internal OAuth server). You can set an inactivity timeout for tokens. By default, no inactivity timeout is set.

5.1.5 Defining more identity providers

The Red Hat OpenShift Container Platform master includes a built-in OAuth server. Developers and administrators obtain OAuth access tokens to authenticate themselves to the API. As an administrator, you can configure OAuth to specify an identity provider after you install your cluster.

You can configure the following types of identity providers:

- htpasswd
- Keystone
- LDAP
- Basic authentication
- Request header
- GitHub or GitHub Enterprise
- GitLab
- Google
- OpenID Connect

htpasswd

Configure the htpasswd identity provider to validate usernames and passwords against a flat file that is generated by using htpasswd.

Keystone

Configure the keystone identity provider to integrate your Red Hat OpenShift Container Platform cluster with Keystone to enable shared authentication with an OpenStack Keystone v3 server that is configured to store users in an internal database.

LDAP

Configure the LDAP identity provider to validate usernames and passwords against an LDAPv3 server by using simple bind authentication.

Basic authentication

Configure a basic-authentication identity provider for users to log in to Red Hat OpenShift Container Platform with credentials that are validated against a remote identity provider. Basic authentication is a generic back-end integration mechanism.

Request header

Configure a request-header identity provider to identify users from request header values, such as X-Remote-User. A request header typically is used with an authenticating proxy, which sets the request header value.

GitHub or GitHub Enterprise

Configure a GitHub identity provider to validate usernames and passwords against GitHub or the GitHub Enterprise OAuth authentication server.

GitLab

Configure a GitLab identity provider to use any GitLab instance as an identity provider.

Google

Configure a Google identity provider by using Google OpenID Connect (OIDC) integration.

OpenID Connect

Configure an OIDC identity provider to integrate with an OIDC identity provider by using an Authorization Code Flow.

When an identity provider is defined, you can use role-based access control (RBAC) to define and apply permissions. For more information about RBAC, see 5.2, “RBAC setup for users and service accounts” on page 106.

5.1.6 Authentication metrics for Prometheus

Red Hat OpenShift Container Platform captures the following Prometheus system metrics during authentication attempts:

- `openshift_auth_basic_password_count` counts the number of **`oc`** login username and password attempts.
- `openshift_auth_basic_password_count_result` counts the number of **`oc`** login username and password attempts by result, success, or error.
- `openshift_auth_form_password_count` counts the number of web console login attempts.

- `openshift_auth_form_password_count_result` counts the number of web console login attempts by result, success, or error.
- `openshift_auth_password_total` counts the total number of **oc** login and web console login attempts.

5.2 RBAC setup for users and service accounts

To grant users the minimum required permissions, use RBAC objects.

Here are the main components of RBAC:

- Subjects: Users, administrators, processes, and processes in a pod
- Resources: Pod, services, Node, namespace, persistent volumes (PVs), secrets, Ingress, persistent volumes claims (PVCs), and deployment
- Verbs: Get, create, list, and delete

Red Hat OpenShift RBAC Hierarchy is composed of the following objects, as shown in Figure 5-1.

- Role: Contains a list of rules, each of which is built from a verb and an API resource. The rule specifies a list of operations that may be performed on a specific resource.
- RoleBinding: Creates the association between a “Subject” and a “Role” that specifies the permissions themselves.
- ClusterRoles: Roles that are cluster-scoped associated with any user in the cluster. They are created once.
- ClusterRoleBinding: Binds the ClusterRole to the Subject in the entire cluster.

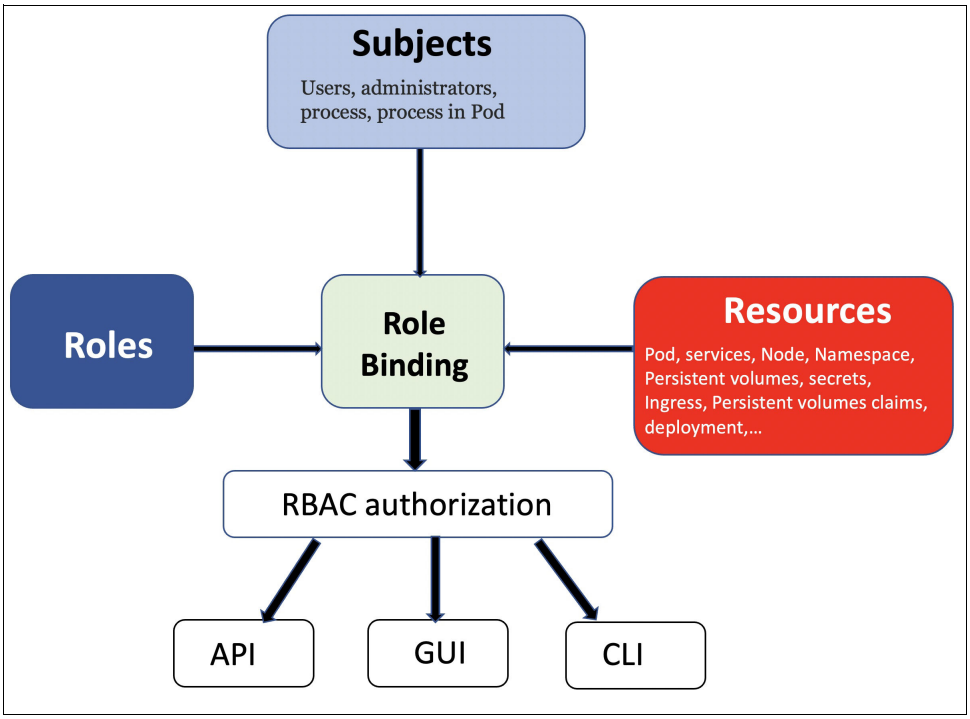


Figure 5-1 RBAC objects

To manage fewer roles with more associations, define ClusterRoles globally and associate them to subjects locally by using RoleBinding.

To create Roles and RoleBinding by using the Red Hat OpenShift GUI, complete the following steps:

1. Select **User Management** → **Role**, and then select **Create Role**, as shown in Figure 5-2.

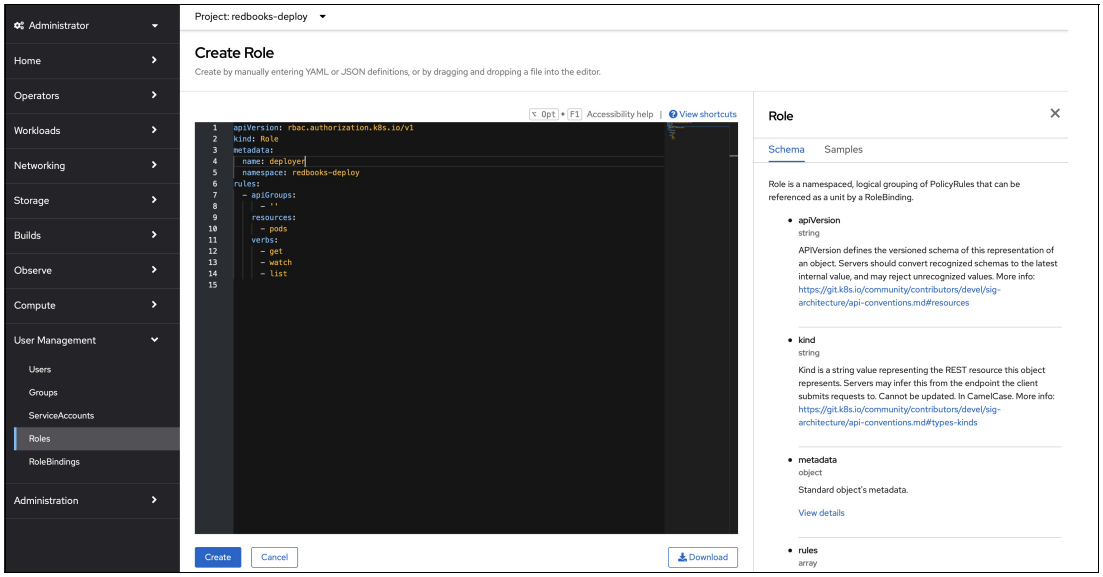


Figure 5-2 Create Role

2. Go to RoleBindings and click **Create binding**, as shown in Figure 5-3.

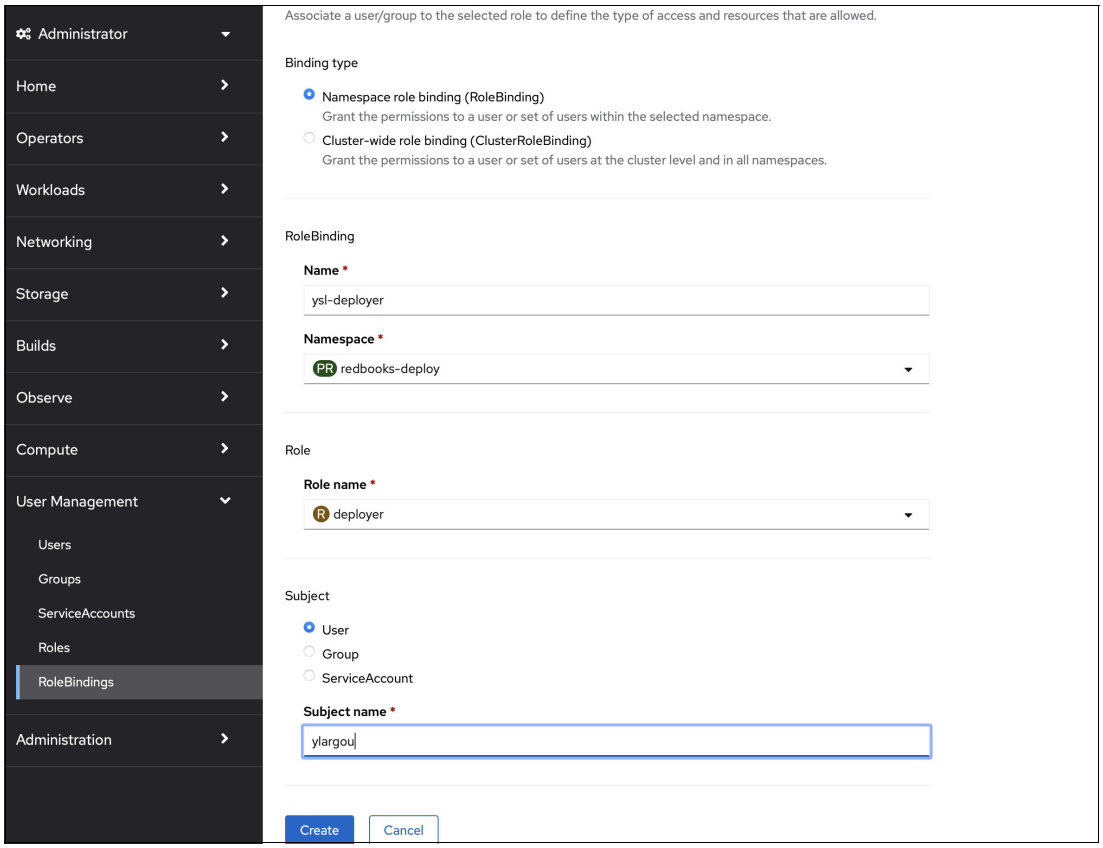


Figure 5-3 Create RoleBindings

Red Hat OpenShift Container Platform includes the following default cluster roles that you can bind to users and groups cluster-wide or locally:[1]

- admin
- basic-user
- cluster-admin
- cluster-status
- cluster-reader
- edit
- self-provisioner
- view

[1] https://docs.openshift.com/container-platform/4.11/authentication/using-rbac.html

Using RBAC to apply permission examples

Complete the following steps:

1. To view local roles and bindings, run the command that is shown in Example 5-1.

Example 5-1 Viewing local roles and bindings for a project sample

```
$ oc describe rolebinding.rbac -n powervm-rmc
Name:         system:deployers
Labels:       <none>
Annotations:  openshift.io/description:
                Allows deploymentconfigs in this namespace to rollout pods in this namespace.  It is
auto-managed by a controller; remove subjects to disa...
Role:
  Kind:  ClusterRole
  Name:  system:deployer
Subjects:
  Kind            Name      Namespace
  ----            ----      ---------
  ServiceAccount  deployer  powervm-rmc

Name:         system:image-builders
Labels:       <none>
Annotations:  openshift.io/description:
                Allows builds in this namespace to push images to this namespace.  It is auto-managed
by a controller; remove subjects to disable.
Role:
  Kind:  ClusterRole
  Name:  system:image-builder
Subjects:
  Kind            Name     Namespace
  ----            ----     ---------
  ServiceAccount  builder  powervm-rmc

Name:         system:image-pullers
Labels:       <none>
Annotations:  openshift.io/description:
                Allows all pods in this namespace to pull images from this namespace.  It is
auto-managed by a controller; remove subjects to disable.
Role:
  Kind:  ClusterRole
  Name:  system:image-puller
Subjects:
  Kind   Name                                Namespace
  ----   ----                                ---------
  Group  system:serviceaccounts:powervm-rmc

Name:         system:openshift:scc:privileged
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  system:openshift:scc:privileged
Subjects:
  Kind            Name         Namespace
  ----            ----         ---------
  ServiceAccount  powervm-rmc  powervm-rmc
```

2. To manage the roles and bindings, run the command that is shown in Example 5-2.

Example 5-2 Adding a role to a user for specific project

```
$ oc adm policy add-role-to-user admin y.largou -n powervm-rmc
clusterrole.rbac.authorization.k8s.io/admin added: "ylargou"
```

3. Verify local roles and bindings by running the command that is shown in Example 5-3 (after adding user `ylargou` to the admins RoleBinding).

Example 5-3 Verifying local roles and binding

```
./oc describe rolebinding.rbac -n powervm-rmc
Name:         admin
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  admin
Subjects:
  Kind  Name     Namespace
  ----  ----     ---------
  User ylargou
Name:         system:deployers
Labels:       <none>
Annotations:  openshift.io/description:
                Allows deploymentconfigs in this namespace to rollout pods in this namespace.  It is
auto-managed by a controller; remove subjects to disa...
Role:
  Kind:  ClusterRole
  Name:  system:deployer
Subjects:
  Kind            Name      Namespace
  ----            ----      ---------
  ServiceAccount  deployer  powervm-rmc

Name:         system:image-builders
Labels:       <none>
Annotations:  openshift.io/description:
                Allows builds in this namespace to push images to this namespace.  It is auto-managed
by a controller; remove subjects to disable.
Role:
  Kind:  ClusterRole
  Name:  system:image-builder
Subjects:
  Kind            Name     Namespace
  ----            ----     ---------
  ServiceAccount  builder  powervm-rmc

Name:         system:image-pullers
Labels:       <none>
Annotations:  openshift.io/description:
                Allows all pods in this namespace to pull images from this namespace.  It is
auto-managed by a controller; remove subjects to disable.
Role:
  Kind:  ClusterRole
  Name:  system:image-puller
Subjects:
  Kind   Name                                Namespace
  ----   ----                                ---------
  Group  system:serviceaccounts:powervm-rmc

Name:         system:openshift:scc:privileged
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  system:openshift:scc:privileged
Subjects:
  Kind            Name         Namespace
  ----            ----         ---------
  ServiceAccount  powervm-rmc  powervm-rmc
```

4. To create a cluster role, run the command that is shown in Example 5-4.

Example 5-4 Creating a cluster role sample

```
$ oc create clusterrole powertstviewonly --verb=get --resource=pod
clusterrole.rbac.authorization.k8s.io/powertstviewonly created
```

5. To create a cluster admin, run the command that is shown in Example 5-5.

Example 5-5 Creating a cluster admin

```
$ oc adm policy add-cluster-role-to-user cluster-admin wlargou
```

6. To create a local role for a project and then bind it to a user, complete the following steps:

   a. Create a local role by running the command that is shown in Example 5-6, where `powertst-role` is the local role name and `powertst-deploy` is the project name.

Example 5-6 Creating a local role

```
$ oc create role powertst-role --verb=get --resource=pod -n powertst-deploy
role.rbac.authorization.k8s.io/powertst-role created
```

   b. Bind a local role to a user by running the command that is shown in Example 5-7.

Example 5-7 Binding a local role to a user

```
$ oc adm policy add-role-to-user powertst-role ylargou --role-namespace=powertst-deploy -n
powertst-deploy
role.rbac.authorization.k8s.io/powertst-role added: "ylargou"
```

7. To check the accessibility to a resource, run the command that is shown in Example 5-8.

Example 5-8 Checking the accessibility to a resource

```
$ oc adm policy who-can get pod

resourceaccessreviewresponse.authorization.openshift.io/<unknown>

Namespace: default
Verb:      get
Resource:  pods

Users:  ylargou
        system:admin
        system:kube-scheduler
        system:serviceaccount:default:deployer
        system:serviceaccount:kube-system:deployment-controller
        system:serviceaccount:kube-system:endpoint-controller
        system:serviceaccount:kube-system:endpointslice-controller
        system:serviceaccount:kube-system:ephemeral-volume-controller
        system:serviceaccount:kube-system:generic-garbage-collector
        system:serviceaccount:kube-system:namespace-controller
        system:serviceaccount:kube-system:persistent-volume-binder
        system:serviceaccount:kube-system:pvc-protection-controller
        system:serviceaccount:kube-system:statefulset-controller
        system:serviceaccount:openshift-apiserver-operator:openshift-apiserver-operator
        system:serviceaccount:openshift-apiserver:openshift-apiserver-sa
        system:serviceaccount:openshift-authentication-operator:authentication-operator
        system:serviceaccount:openshift-authentication:oauth-openshift
        system:serviceaccount:openshift-cluster-node-tuning-operator:cluster-node-tuning-operator
        system:serviceaccount:openshift-cluster-storage-operator:cluster-storage-operator
        system:serviceaccount:openshift-cluster-storage-operator:csi-snapshot-controller-operator
        system:serviceaccount:openshift-cluster-version:default
        system:serviceaccount:openshift-config-operator:openshift-config-operator
```

```
system:serviceaccount:openshift-controller-manager-operator:openshift-controller-manager-operator
        system:serviceaccount:openshift-controller-manager:openshift-controller-manager-sa
        system:serviceaccount:openshift-dns-operator:dns-operator
        system:serviceaccount:openshift-etcd-operator:etcd-operator
        system:serviceaccount:openshift-etcd:installer-sa
        system:serviceaccount:openshift-image-registry:cluster-image-registry-operator
        system:serviceaccount:openshift-image-registry:pruner
        system:serviceaccount:openshift-infra:build-controller
        system:serviceaccount:openshift-infra:default-rolebindings-controller
        system:serviceaccount:openshift-infra:deployer-controller
        system:serviceaccount:openshift-infra:pv-recycler-controller
        system:serviceaccount:openshift-infra:template-instance-controller
        system:serviceaccount:openshift-infra:template-instance-finalizer-controller
        system:serviceaccount:openshift-ingress-operator:ingress-operator
        system:serviceaccount:openshift-insights:gather
        system:serviceaccount:openshift-kube-apiserver-operator:kube-apiserver-operator
        system:serviceaccount:openshift-kube-apiserver:installer-sa
        system:serviceaccount:openshift-kube-apiserver:localhost-recovery-client

system:serviceaccount:openshift-kube-controller-manager-operator:kube-controller-manager-operator
        system:serviceaccount:openshift-kube-controller-manager:installer-sa
        system:serviceaccount:openshift-kube-controller-manager:localhost-recovery-client
        system:serviceaccount:openshift-kube-scheduler-operator:openshift-kube-scheduler-operator
        system:serviceaccount:openshift-kube-scheduler:installer-sa
        system:serviceaccount:openshift-kube-scheduler:localhost-recovery-client
        system:serviceaccount:openshift-kube-scheduler:openshift-kube-scheduler-sa

system:serviceaccount:openshift-kube-storage-version-migrator-operator:kube-storage-version-migrator-o
perator
        system:serviceaccount:openshift-kube-storage-version-migrator:kube-storage-version-migrator-sa
        system:serviceaccount:openshift-machine-api:cluster-autoscaler
        system:serviceaccount:openshift-machine-api:machine-api-controllers
        system:serviceaccount:openshift-machine-config-operator:default
        system:serviceaccount:openshift-machine-config-operator:machine-config-daemon
        system:serviceaccount:openshift-monitoring:cluster-monitoring-operator
        system:serviceaccount:openshift-monitoring:prometheus-adapter
        system:serviceaccount:openshift-monitoring:prometheus-k8s
        system:serviceaccount:openshift-multus:metrics-daemon-sa
        system:serviceaccount:openshift-multus:multus
        system:serviceaccount:openshift-network-diagnostics:network-diagnostics
        system:serviceaccount:openshift-network-operator:default
        system:serviceaccount:openshift-oauth-apiserver:oauth-apiserver-sa
        system:serviceaccount:openshift-operator-lifecycle-manager:olm-operator-serviceaccount
        system:serviceaccount:openshift-ovn-kubernetes:ovn-kubernetes-controller
        system:serviceaccount:openshift-ovn-kubernetes:ovn-kubernetes-node
        system:serviceaccount:openshift-service-ca-operator:service-ca-operator
        system:serviceaccount:openshift-storage:ibm-spectrum-scale-csi-operator
        system:serviceaccount:openshift-storage:ibm-spectrum-scale-csi-resizer
        system:serviceaccount:openshift-storage:ocs-operator
        system:serviceaccount:openshift-storage:rook-ceph-system
Groups: system:cluster-admins
        system:cluster-readers
system:masters
```

6

Data and application security

In this chapter, we explore the different aspects of data and application security in the Red Hat OpenShift environment.

This chapter describes the following topics:

- Credential rotation for application to application communication
- Central secrets management: Single source of truth
- Container security considerations
- Data at rest encryption

6.1 Credential rotation for application to application communication

As a best practice, rotate credentials to safeguard them, and to mitigate damage that is done from credentials leakage. Leaked credentials can lead to costly breaches, loss of data, and loss of trust.

Rotate credentials at the following times (no downtime is needed):

- All credentials are rotated at least annually, and it is better to rotate them every 90 days.
- In an emergency such as a malicious attack, the credentials should be rotated within 4 hours.
- When someone leaves the organization, rotate the credential within 24 hours of separation.

Rotating credentials involves the following steps:

1. Generate new credentials.
2. Distribute the new credentials to the applications.
3. Ensure that the applications use the new keys.
4. Validate that the applications are working as expected.
5. Delete or disable the old keys.

The rotation of credentials is complex and should not be done manually. To help with rotation, consider the following solutions:

- Hashicorp Vault, which is no-charge, open-source product that enables automatic rotation. It helps integrate applications with Hashicorp Vault and improves secret management.
- IBM Cloud Secrets Manager helps to automatically rotate credentials. You also can use it to schedule rotation. IBM Cloud Secrets Manager also helps to restore secrets that are accidentally replaced.

Here are best practices for automatic secrets rotation:

- Define a rotation strategy and determine the frequency of the rotation.
- Set up alerts for the expiring secrets according to the determined frequency.
- Plan and enable automatic secret rotation by using services like IBM Cloud Secrets Manager.
- Avoid application outages by locking the secrets to avoid accidental deletion.

6.2 Central secrets management: Single source of truth

Secrets management refers to the tools and methods that are used to manage the digital authentication credentials. Secrets are used to unlock protected resources or data. Secrets can be of any type, such as passwords, certificates, application programming interface (API) keys, Secure Shell or Secure Socket Shell (SSH) keys, and encryption keys. Secrets mismanagement is one of the biggest cybersecurity concerns.

Here are the best practices for secrets management:

- Centralize the secrets.
- Use access-control-based secret access.
- Use end-to-end encryption to protect secrets when they are transmitted over the network.
- Avoid storing secrets in environment variables.
- Never write secrets to disk or any persistent storage.
- Monitor and audit activity for secrets.

Mature, production-grade systems should maintain centralized secrets, and centralized secrets management should address the following challenges:

- Management of secrets, including creation, secure storage, rotation, and access control
- Single source of secrets for both humans and machines
- Encryption management
- Providing governance for accessing the secrets

To help with secrets management, consider the following solutions:

- Hashicorp Vault is a no-charge, open-source product that centrally manages and enforces access to secrets and systems by using trusted sources of application and user identity.
- IBM Cloud Secrets Manager is a centralized secret manager that can dynamically create secrets and lease them to applications while access is controlled from a single location.

  IBM Cloud Secrets Manager helps with the following tasks:

  - Data isolation
  - Implementing the principle of least privilege
  - Encrypting the backups of secrets
  - Lifecycle management of secrets
  - Building your own public key infrastructure (PKI) system

For application secrets that need a higher level of control that relies on highly secure, customer controlled cryptographic hardware, use the IBM Key Protect for IBM Cloud service, which helps you provision and store encrypted keys for applications across IBM Cloud services so that you can see and manage data encryption and the entire key lifecycle from one central location.

6.3 Container security considerations

For security, there are the standard regulatory bodies, such as National Institute of Standards and Technology (NIST), PCI-DSS, TSA, and HIPAA, which define the security requirements. Based on their specific directives, each organization designs its security requirements.

In a container, there are various layers. Security must be implemented in each layer to make it secure.

Consider the following tasks when you are setting the criteria for platform and application-level security in containers:

- Integrate with Global Authentication:
  - Use Security Assertion Markup Language (SAML), OpenID Connect (OIDC), or OAuth.
  - Apply multi-factor authentication (MFA).
  - Use native- and federation-based authorization to evaluate user attributes, groups, and roles to grant user-specific access to applications.

- Enforce and manage least privilege for all users:
  - Use standardized user roles and accounts or subscription-level policies to enforce access control.
  - Avoid using local accounts or long-lived credentials.
- Use Infrastructure as Code (IaC) and DevOps processes that include security testing and scanning:
  - Deploy all changes in system integration and the production environment by using IaC.
  - All IaC deployments should be scanned for security weakness, errors, vulnerabilities, and overly permissive policies.
- Adopt native encryption for data at rest:
  - All databases, file storage, block storage, message queuing, and other systems that store data at rest must be encrypted according to security policies by using the standard key management or Hardware Security Module (HSM) services.
  - Encryption keys must not be shared among accounts, subscriptions, different applications, or across environments.
  - A single key may be used to encrypt components of the same application in the same environment.
- Adopt native encryption in transit for all traffic:
  - All traffic in the cloud, API traffic, and traffic between on-premises and cloud must be encrypted with TLS 1.2.
  - Certificates for encrypted traffic in the cloud must use the standard certificate management service. No manual certificate management should be done.
  - Certificates must not be shared among accounts, subscriptions, or different applications, or across environments.
  - A common certificate may be used for components of the same application in the same environment.
  - Self-signed certificates should not be used.
- Structure the usage of cloud components into a segmented network model:
  - Traffic should not be permitted between accounts or subscriptions except as needed for cross-application functions.
  - All inter-application traffic must be private and not over the internet.
  - For components and services that support micro-segmentation, apply the principle “Deny All, Permit by Exception”.
- Log application and platform actions and ensure integration with Security Incident and Event Manager (SIEM):
  - Platform-level and service-level logging must be used, and these log contents must be sent to the SIEM.
  - Application-level logging must be enabled for high-risk transactions and activities, and these events should be sent to the SIEM.

- Ensure compliance with standard cloud platform architectures and controls:
  - All accounts, services, platforms, and application configurations must adhere to the Center for Internet Security (CIS) benchmarks.
  - All services and accounts must be tagged and have security controls that are enforced based on environmental needs and data classification.
  - Use only approved and standardized configurations.

6.4 Data at rest encryption

Data encryption is a pillar of application security. It controls which entities have access to the data. Data can be at rest, in use, and in transit. To protect against data leakage, the data must be encrypted at every stage.

This section focuses on data at rest.

To enable encryption for Red Hat OpenShift Container Platform and Kubernetes (K8s) components, you must know which components contain data that is necessary to encrypt. There are two major components to consider:

- One component is the application persistence layer, which stores the application data.
- The other component is the Red Hat OpenShift and Kubernetes API Server, which has the application configuration.

The application configuration also can contain data with sensitive information. It can be a definition of a pod, a deployment, a secret, a configmap, and other items.

6.4.1 Application persistence layer

Application data can be stored on different back ends. Here are some storage back ends:

- Databases (relational, NoSQL, TSDB, and others)
- Key-value stores (`etcd`, Redis, and others)
- Object storages (Ceph, MinIO, AWS S3, Azure Blob Storage, Google Cloud Object Storage, and others)
- File shares (Network File System (NFS), CIFS, and others)
- Block devices (disks)

Databases and key-value stores are applications that store the data as files on a block device or a file share. The major storage back ends that are used by pods are shown in the following list:

- Block devices
- File shares
- Object storage

Red Hat OpenShift Data Foundation Advanced provides these three types of storage. Depending on the environment where the *Red Hat OpenShift Container Platform* cluster is installed and which data storage back ends you are using, the data encryption might not be available, and if it is available, it might have different methods of implementation. This section focuses on running Red Hat OpenShift Data Foundation on IBM Power.

Figure 6-1 shows the encryption options of Red Hat OpenShift Data Foundation.

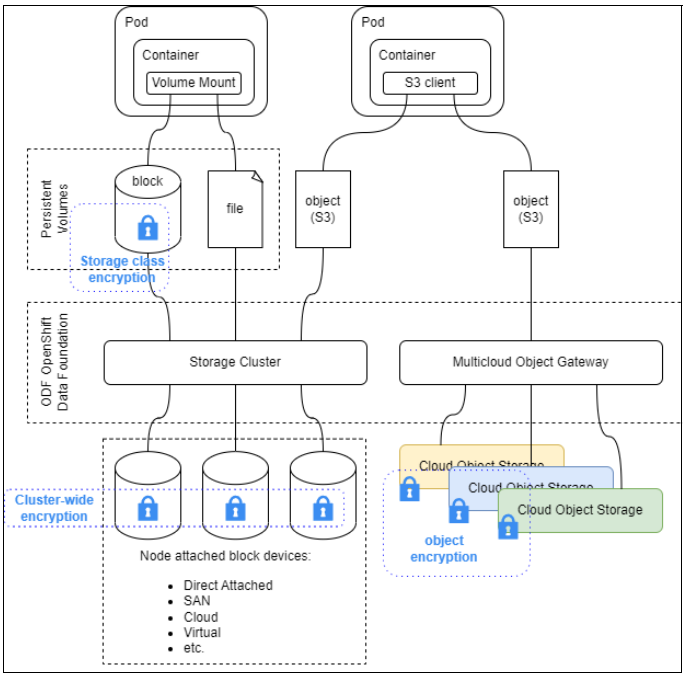


Figure 6-1 Storage encryption

Object encryption

The *Multicloud Object Gateway* (MCG) is a layer between the application and the object storage where the data is stored. MCG provides the standard S3 API for your applications and can store data on various on-premises and cloud object storage, for example, Ceph Object Storage, MinIO, Google Cloud Object Storage, Azure Blob, or AWS S3. MCG also provides extra features like an *object bucket claim* and an *object bucket*, which can be compared with a *physical volume claim* (PVC) and a *physical volume* (PV).

For more information about these capabilities, see this Red Hat document.

All stored objects are encrypted by default. The encryption keys are stored in a Key Management System (KMS). Creating an object bucket claim leads to a new object bucket and the creation of a new object bucket account with its own credentials. Every object bucket is accessible only by its dedicated account.

Cluster-wide encryption

Red Hat OpenShift Data Foundation Advanced creates a storage cluster. The underlying technology is Ceph that is operated by the *Rook-Ceph operator*. The storage cluster provides block, file, and object storage.

Cluster-wide encryption can be enabled at the time of the deployment of the storage cluster, and encryption cannot be enabled on an existing storage cluster after it is deployed. All data that is stored on the cluster, whether through block, file, or object protocol, is transparently encrypted on the underlying disks. The encryption keys can be stored on a KMS.

For more information, see the following documentation:

- Data encryption options (Planning your deployment)
- Enabling cluster-wide encryption with KMS (Deploying Red Hat OpenShift Data Foundation by using IBM Power)

Storage class encryption

The Red Hat OpenShift Data Foundation Advanced storage cluster also provides storage-class encryption. This encryption option is available only for persistent volumes (PVs) that are based on block devices with ReadWriteOnce (RWO) access. Shared PVs with ReadWriteMany (rwx) access cannot be encrypted with that option. This type of encryption ensures tenant isolation because the encryption keys can be accessed only within a project. The encryption keys are stored on a KMS.

It is *not* a best practice to use both cluster-wide and storage class encryption concurrently because this configuration doubles the encryption and consumes compute resources. Consider both options at planning time to avoid this situation.

For more information, see the following documentation:

- Data encryption options (Planning your deployment)
- Storage class for persistent volume encryption (Managing and allocating storage resources)

Key Management System

Each type of encryption that is described in 6.4.1, “Application persistence layer” on page 117 provides for the usage of a KMS. For cluster-wide encryption, KMS is optional, but for storage-class and object encryption, KMS is mandatory. The KMS that is supported by Red Hat OpenShift Data Foundation is Hashicorp Vault.

Whether it is optional or mandatory, it is a best practice to use a KMS that runs outside of the Red Hat OpenShift Container Platform cluster to ensure that the encryption keys are not stored inside the Red Hat OpenShift Container Platform cluster, which can prevent the encryption keys from getting lost. A KMS also provides key rotation, which increases security.

Figure 6-2 shows a best practice for a KMS implementation.

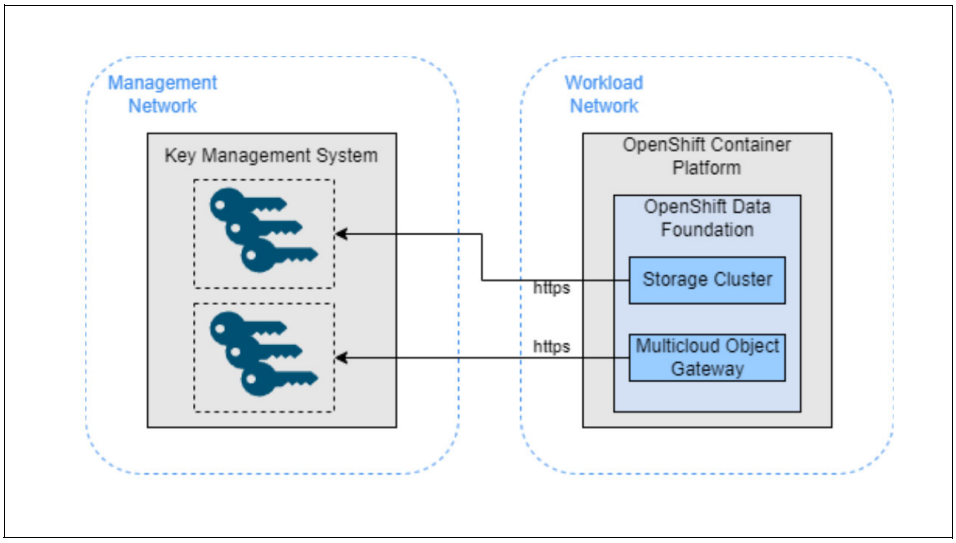


Figure 6-2 Key Management System

Application backup

Red Hat OpenShift Data Foundation Advanced provides encrypted block, file, and object storage for applications, but there is an extra layer of data storage to consider, which is the backup of your application data. To prevent data loss at the backup layer, store this data encrypted.

Red Hat OpenShift Platform provides Red Hat OpenShift APIs for Data Protection Operator, which leverages *Velero* and its APIs for making backups of your application data, which includes all related K8s resource objects, PVs, and internal images. The backups of the PVs are done by snapshots by using the standard Container Storage Interface (CSI) that is provided by Red Hat OpenShift Data Foundation Advanced. The PV snapshots are encrypted the same way as the source PVs. Restoration of PV snapshots is done by using the same storage class as the source PVs, and the snapshots are automatically encrypted in the same way. Every other resource except the PV is backed up to an object storage. One solution is to use a bucket that is defined with the MCG because this data is also encrypted by default. It is also possible to back up the PVs to object storage. Here, the same rule applies when using the MCG as the backup location, that is, all data is encrypted by default.

If your applications have their own backup solution, consider using the MCG as the backup location.

For more information, see the following documentation:

- Installing and configuring Red Hat OpenShift APIs for Data Protection with MCG
- Installing and configuring Red Hat OpenShift APIs for Data Protection with Red Hat OpenShift Data Foundation
- Velero Documentation

6.4.2 Red Hat OpenShift and Kubernetes API Server

Red Hat OpenShift and Kubernetes API Server holds the application configuration, which can be any API object. These objects are created by applying the declarative YAML files against Red Hat OpenShift and Kubernetes API Server. Red Hat OpenShift and Kubernetes API Server stores its data in the highly available key-value store `etcd`, which persists its data on a local volume on the host on which it runs.

Figure 6-3 shows the Red Hat OpenShift and Kubernetes API Server storage functions.

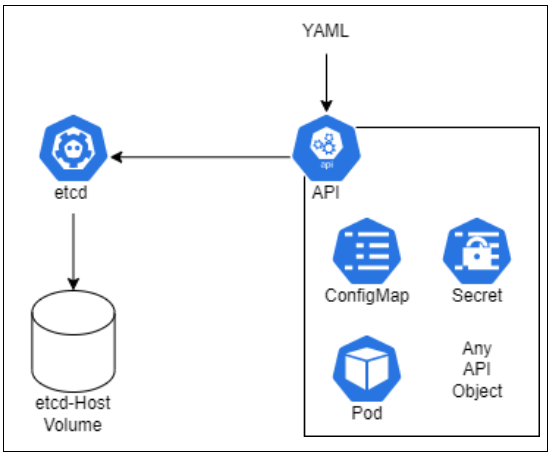


Figure 6-3 Red Hat OpenShift and Kubernetes API Server storage

The data that is stored in `etcd` is not encrypted by default. To perform a disaster recovery, as a best practice, run repetitive snapshots of `etcd` and make a backup of the snapshots to a secure storage that is different from the host `etcd` on which it is running.

To keep the data containing sensitive information (for example, usernames and passwords) secure even at the backup location, encrypt it so that if an unauthorized person gains access to the backups, the sensitive information cannot be read.

You can enable `etcd` encryption in Red Hat OpenShift Container Platform. For more information, see Encrypting etcd data.

The weekly rotated encryption keys can be found in the namespaces `openshift-apiserver` and `openshift-kube-apiserver` as a secret with the name `encryption-config`. They are necessary for disaster recovery, and should also be stored securely on a repetitive basis.

Important: The `etcd` encryption encrypts the apiserver resources: Secrets, Config Maps, Routes, OAuth Access Tokens, and OAuth Authorize Tokens. All other resources are not encrypted. When you create resources like deployments, follow best practices and never use sensitive information in their declarations. Instead, refer to related Secrets, for example, with volumes from a Secret with the '`volumes[].secret.secretName`' notation or environment variables with the '`env[].valueFrom.secretKeyRef.{name,key}`' notation.

6.4.3 IBM Security Guardium for File and Database Encryption

IBM Security Guardium for File and Database Encryption is an enterprise-grade solution that protects data by encrypting files (including database files) and folders by predefining policies for who and which processes can access the data.

Implementation of encryption and decryption and key management is transparent and automated. In addition, IBM Security Guardium for File and Database Encryption supports scheduled automatic key rotation, helping with compliance to *cryptoperiod* recommendations that are mandated by NIST and other security guidelines.

IBM Security Guardium for Container Data Encryption is a file and database encryption extension that enforces traditional transparent encryption to on container environments. For more information about IBM Security Guardium for Container Data Encryption, see 6.4.4, "IBM Security Guardium for Container Data Encryption" on page 122.

Figure 6-4 shows how IBM Security Guardium for File and Database Encryption provides basic image-level protection by securing and controlling access to container images and instances.

From a container perspective, IBM Security Guardium for File and Database Encryption provides the following basic image-level protection:

- Encrypts containers.
- A policy restricts container access and usage to Red Hat OpenShift environments.
- Restricts usage of containers to only authorized (signed) environment instances.
- Restricts access to data resources that are used by containers to the container environment.
- Requires no operational impact on Red Hat OpenShift environments.
- There is no need to change container images.
- Reports unauthorized access attempts.

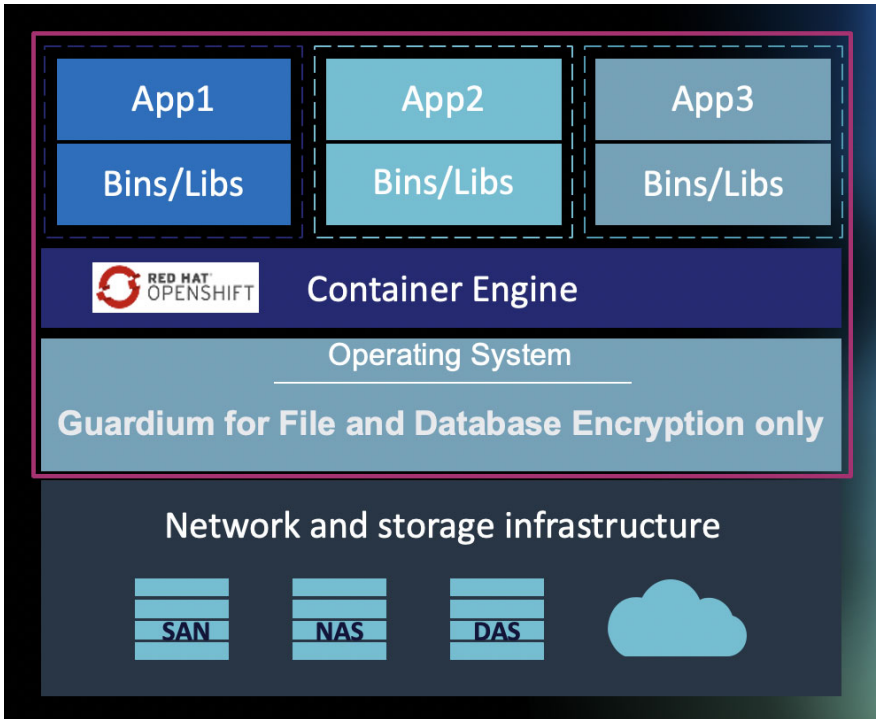


Figure 6-4 IBM Security Guardium for File and Database Encryption

6.4.4 IBM Security Guardium for Container Data Encryption

IBM Security Guardium for Container Data Encryption protects databases and unstructured files on Linux, UNIX, and Windows (LUW) platforms, and it provides security for container environments.

As modern application and data store architectures embrace container-based platforms, data protection policies must enforce container security. Certain containers might require more restrictive access policies than others, and IBM Security Guardium for Container Data Encryption helps achieve and enforce policies for those containers.

The IBM Security Guardium for Container Data Encryption extension to IBM Security Guardium for File and Database Encryption delivers container-aware data protection and encryption capabilities for granular data access controls and data access logging in containerized environments for both Red Hat OpenShift hosts and images.

Figure 6-5 shows how IBM Security Guardium for Container Data Encryption extends the security controls of IBM Security Guardium for File and Database Encryption.

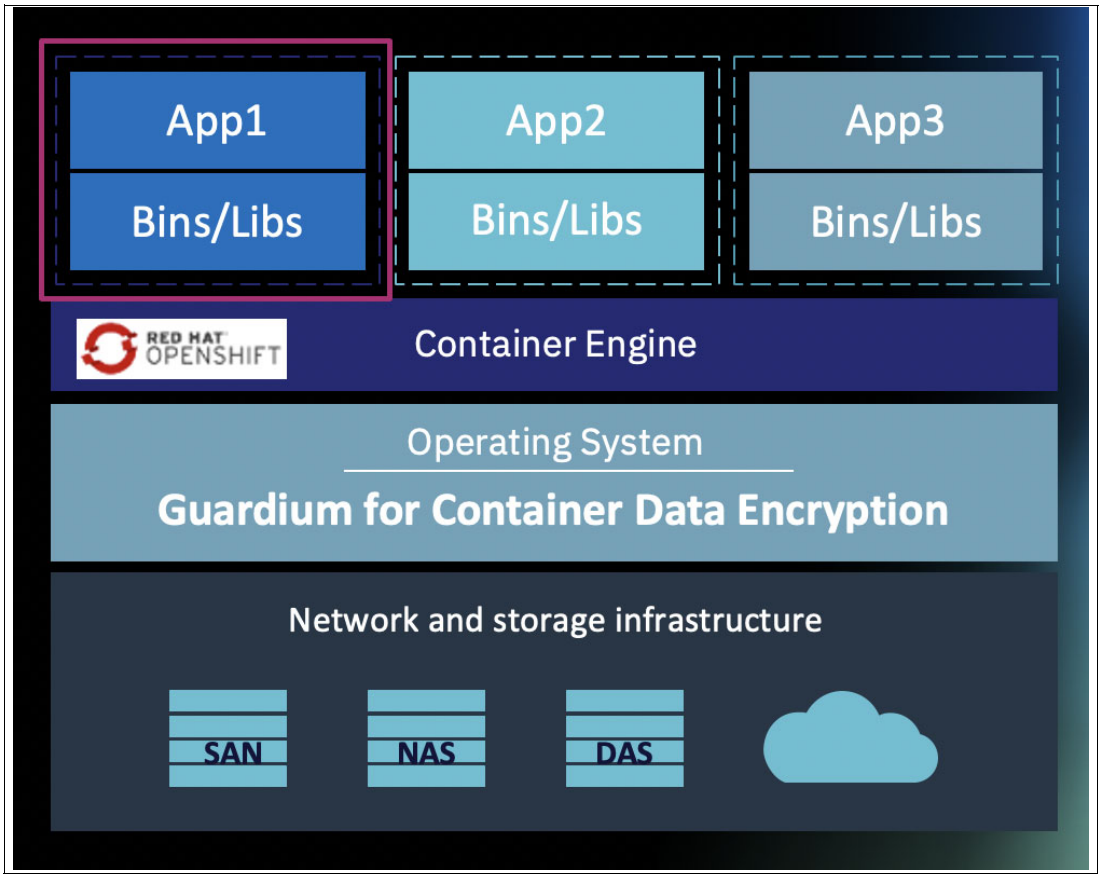


Figure 6-5 IBM Security Guardium for Container Data Encryption

This solution enables security teams to modify encryption, access controls, and data access audit logging on a per-container basis, both the data inside the containers and to external storage that is accessible from the container. This solution secures container volumes; protects against root, privileged, or unauthorized user access within containers; and prevent privilege escalation attacks from other containers.

Users can isolate data access between containers and establish granular access policies that are based on specific users, process, and resource sets.

IBM Security Guardium for Container Data Encryption extends the security controls of IBM Security Guardium for File and Database Encryption by providing the following features:

- Highly granular security controls that work within Red Hat OpenShift containers.
- Meeting compliance regulations for encryption, access control, and container-level access auditing.
- Per-container encryption, access control, and security intelligence.

- Encrypts data that is generated by applications and stored locally in containers or on linked external storage.
- Adds container-specific, fine-grained access control for internal container users (container administrators, users, processes, and resource sets) to existing systems.
- Requires no container changes.
- Provides protection from access by root, privileged, or unauthorized users inside containers.
- Provides data protection against privilege escalation attacks from other containers.
- Provides access isolation between containers.

Here are some use cases to encrypt Red Hat OpenShift containers, data, and images by using IBM Security Guardium Container Data Encryption:

- Encrypt a Red Hat OpenShift container with an empty GuardPoint. Any new data that is written into the GuardPoint is encrypted with an encryption key that is specified in the GDE Policy.
- Encrypt a Red Hat OpenShift image with an empty GuardPoint.
- Encrypt data in a Red Hat OpenShift image with existing data to protect a directory in a Red Hat image that already contains data. All Red Hat OpenShift containers that are started from this guarded Red Hat OpenShift image are applied with the same GDE policy with encrypted data.

For more information, see the IBM Guardium GDE documentation:

- Product Documentation for IBM Guardium Data Encryption
- *IBM Guardium Data Encryption Administrator's Guide Release v.4.0.0.2*

7

Logging and monitoring

Logging and monitoring enable earlier detection of vulnerabilities in Red Hat OpenShift Container Platform because you can understand the context of security incidents during an active investigation and postmortem analysis; pro-actively monitor security-related activities; and confirm the effectiveness and integrity of the existing security configuration.

This chapter describes the following topics:

- Monitoring containers and Red Hat OpenShift Container Storage security
- Audit logs
- Red Hat OpenShift File Integrity Operator monitoring

7.1 Monitoring containers and Red Hat OpenShift Container Storage security

An important aspect of managing a containerized environment is monitoring what is occurring in the containers. Container monitoring involves tracking and measuring various key performance indicators (KPIs) of a containerized environment. Monitoring containers is a continuous process to ensure that decoupled applications (and often, a microservices environment) are performing at their best.

7.1.1 Challenges of monitoring containers

- Containers are designed to be provisioned and terminated quickly. As such, it can be a challenge to track changes in environments where dozens of containers (and their instances) can be continuously provisioned and terminated.
- Because containers are temporary, their metrics, logs, and other data disappear immediately after they close. You must collect the data before the containers terminate and store it in a central location for analysis.
- Containers share resources such as memory, CPU, and operating systems, so it can be challenging to measure container performance.
- Many traditional monitoring tools are often inadequate when used to monitor containerized environments.

7.1.2 How to effectively monitor containers

You can effectively monitor containers in the following ways:

- Monitoring the entire stack
- Granular visibility
- Contextualized alerting

Monitoring the entire stack

The whole stack must be monitored to get full application visibility. Monitoring must cover containers, clusters, networking, and inter-container communications.

Granular visibility

Multiple levels of granularity are required to get a complete picture. Drilling down by degrees of granularity help to pinpoint where exactly the problems are.

Contextualized alerting

In a containerized environment, an alert in one container might be related to its interaction with another container. Pay careful attention when creating alerts to ensure that the alert contains relevant context information.

7.1.3 Benefits of monitoring containers

Monitoring containers provide the following benefits:

- Determine the cause of a problem, solve it, and catalog the “lessons learned” for future reference.
- Analyze how containerized applications use cloud resources and how to apportion costs.
- Helps organizations to plan future computing resource requirements by using historical monitoring data.

7.1.4 Red Hat OpenShift Container Platform Monitoring

Red Hat OpenShift Container Monitoring Platform[1] overcomes most of the challenges that are outlined in 7.1.2, “How to effectively monitor containers” on page 126. It includes a pre-configured and automatically updating monitoring stack that provides monitoring for core components. Red Hat OpenShift Container Monitoring Platform is based on Prometheus, Grafana, and Alert Manager, and has intergrated best practices. Figure 7-1 shows the components that are installed by the Red Hat OpenShift Container Monitoring Platform.

- *Prometheus* is used as a back end to store time-series data. It is an open-source solution that is the *de facto* standard for cloud-native architecture monitoring.
- *Alertmanager* is used to handle alarms and send notifications.
- *Grafana* is used for displaying data in graph format.

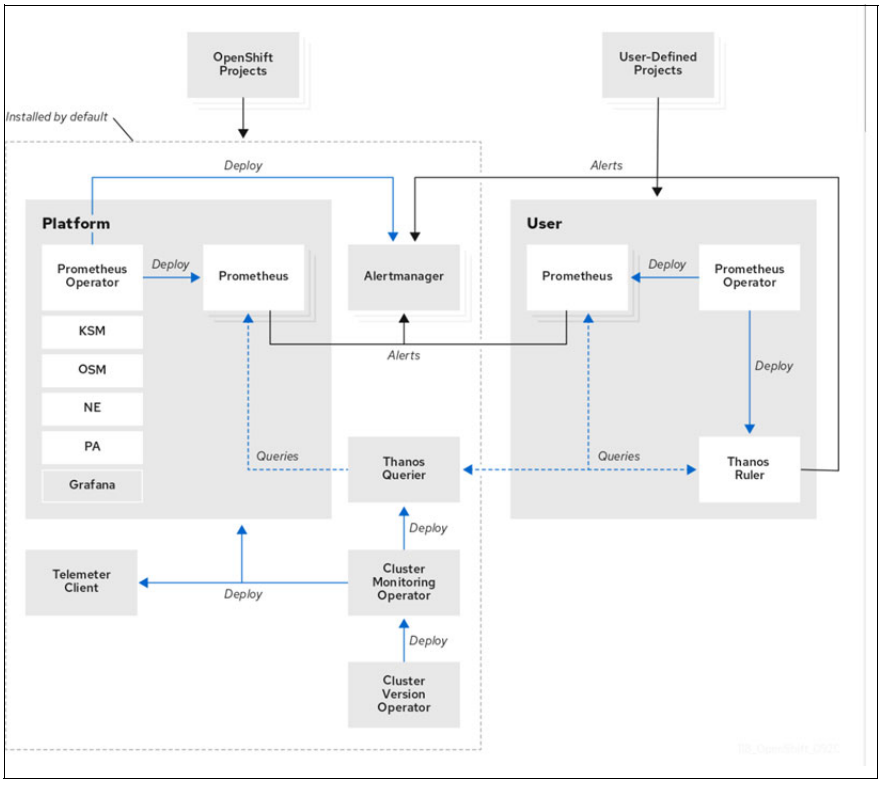


Figure 7-1 Red Hat OpenShift Container Platform Monitoring components installed by default

[1] https://access.redhat.com/documentation/en-us/openshift_container_platform/4.11/html/monitoring/monitoring-overview

A set of alerts are included by default that immediately notify administrators about issues with a cluster. Default dashboards in the Red Hat OpenShift Container Platform web console include visual representations of cluster metrics to help you to quickly understand the state of your cluster. With the Red Hat OpenShift Container Platform web console, you can view and manage metrics, alerts, and review monitoring dashboards.

In the Observe section of Red Hat OpenShift Container Platform web console, you can access and manage monitoring features, such as metrics, alerts, monitoring dashboards, and metrics targets.

After installing Red Hat OpenShift Container Platform, cluster administrators can optionally enable monitoring for user-defined projects. By using this feature, cluster administrators, developers, and other users can specify how services and pods are monitored in their own projects.

7.1.5 Observability and application performance monitoring with IBM Instana

As cloud-native applications continue to grow in scale and complexity, companies rely on application performance monitoring (APM) observability to provide constant visibility into the health of the app and its infrastructure. APM observability provides the key capabilities that are required by highly distributed and scalable cloud-native and hybrid apps to provide optimum performance and resiliency. APM observability must scale with cloud-native apps to ensure that all the components and their dependencies are visible always.

IBM Instana® is a tool that can be used to elevate the observability and APM functions that are provided by the default Red Hat OpenShift container monitoring tools. Instana is an automated system and APM service. It allows visualization of performance through graphs that are generated from machine learning algorithms. Instana increases application performance and reliability though deep observability and applied intelligence to what is being observed. It excels in cloud-based microservice architectures, enabling development teams to iterate more quickly and get in front of issues before they impact customers.

Figure 7-2 provides an overview of the IBM Instana Enterprise Observability Platform.

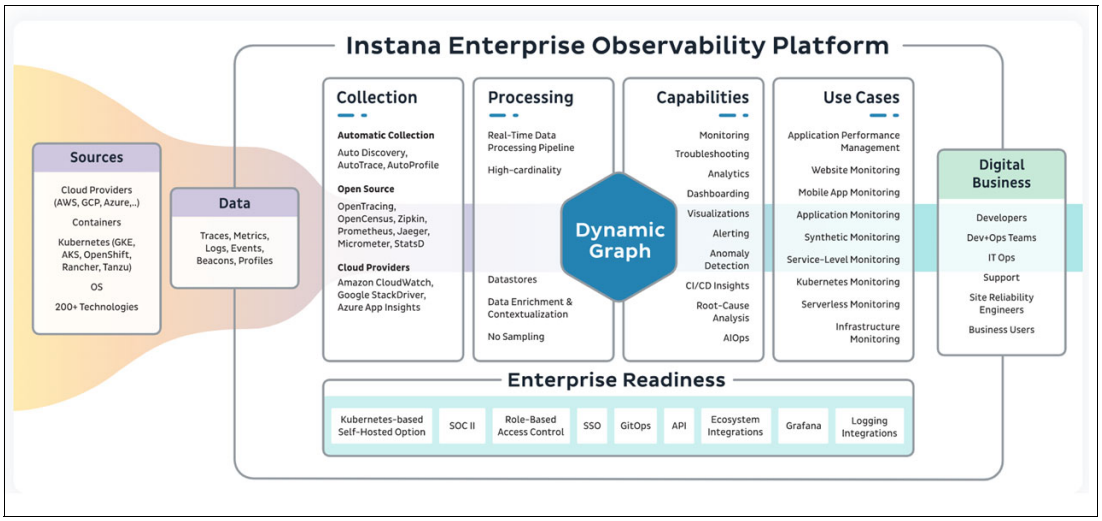


Figure 7-2 IBM Instana Enterprise Observability Platform

For more information about Instana, see Instana.

7.2 Audit logs

Red Hat OpenShift Container Platform Audit provides a chronological record of security-relevant events that documents the sequence of activities that affect every component of the Red Hat OpenShift Cluster.

The following Red Hat OpenShift Container Platform Audit logs can be viewed:

- The Red Hat OpenShift application programming interface (API) server logs, by running the command that is shown in Example 7-1.

Example 7-1 Viewing the apiserver logs

```
$ oc adm node-logs --role=master --path=openshift-apiserver/
```

- The Kubernetes apiserver logs, by running the command that is shown in Example 7-2.

Example 7-2 Viewing the Kubernetes apiserver logs

```
$ oc adm node-logs --role=master --path=kube-apiserver/
```

- The Red Hat OpenShift OAuth apiserver logs, by running the command that is shown in Example 7-3.

Example 7-3 Viewing the OAuth apiserver logs

```
$ oc adm node-logs --role=master --path=oauth-apiserver/
```

7.2.1 Logging operator

The logging subsystem aggregates the following logs from the Red Hat OpenShift cluster:

- Node system audit logs
- Application container logs
- Infrastructure logs

The logging subsystem components include a collector (implemented by Fluentd) that collects logs from Red Hat OpenShift infrastructure, a secured log store (implemented by Elasticsearch), and a visualization UI (implemented by Kibana).

The installation of the logging subsystem for Red Hat OpenShift is done by deploying the following operators:

- Red Hat OpenShift Elasticsearch Operator
- Red Hat OpenShift Logging Operator

7.2.2 Installing the logging subsystem for Red Hat OpenShift

To install the logging subsystem for Red Hat OpenShift, complete the following steps:

1. Configure persistent storage for Elasticsearch with the required space and performance that is described in Configuring persistent storage for the log store.
2. Ensure that the **`vm.max_map_count`** setting is at least `262144` on all nodes.

   If you must change the **`vm_max_count`** setting, see Red Hat OpenShift Container Platform configuration for foundational services.

3. In the Red Hat OpenShift Container Platform web console, select **Operators** → **OperatorHub**.
4. Select and install Red Hat OpenShift Elasticsearch Operator, as shown in Figure 7-3.

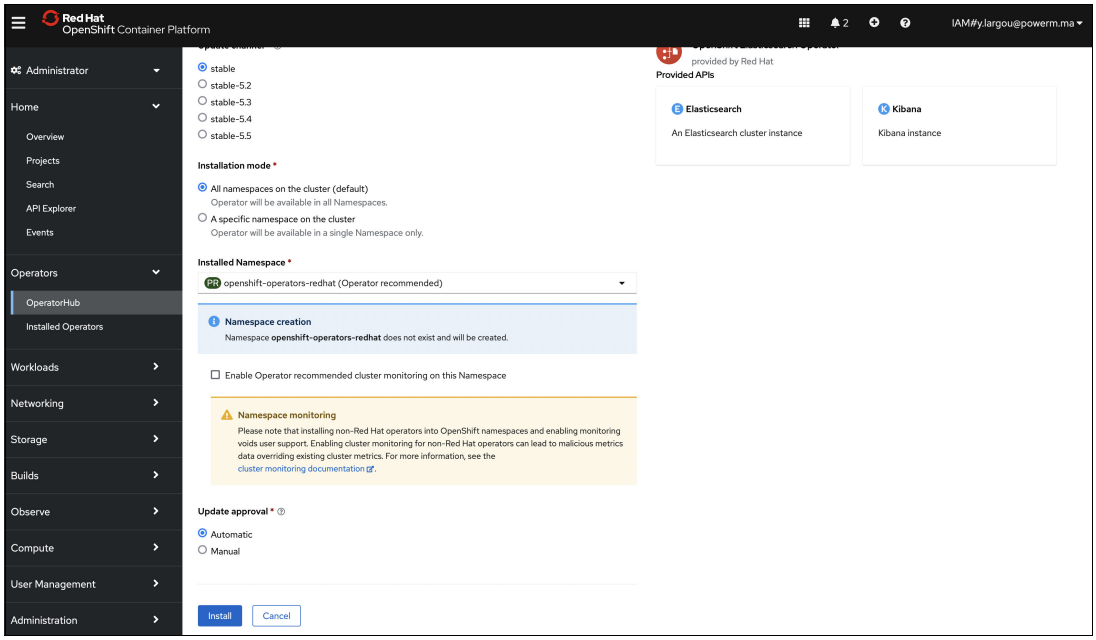


Figure 7-3 Installing Red Hat OpenShift Elasticsearch Operator

5. Select and install Red Hat OpenShift Logging Operator (select **Enable Operator recommended cluster monitoring on this Namespace**), as shown in Figure 7-4.

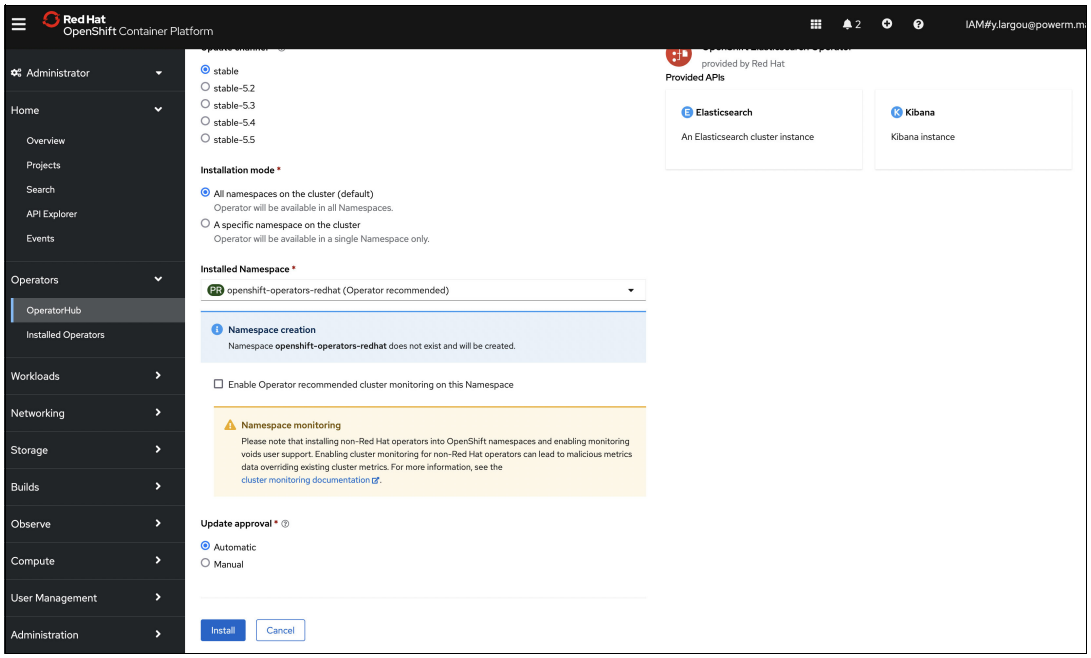


Figure 7-4 Installing Red Hat OpenShift Logging Operator

6. Ensure that Red Hat OpenShift Logging and Red Hat OpenShift Elasticsearch are listed in the Installed Operators with a Status of Succeeded, as shown in Figure 7-5 on page 131.

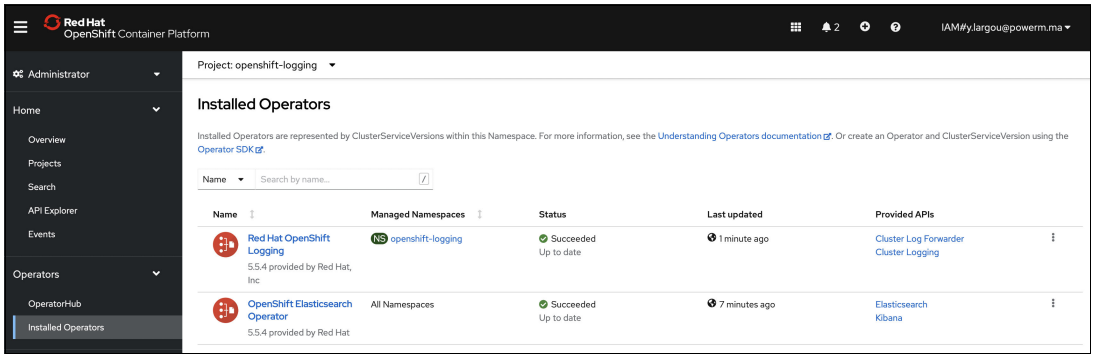


Figure 7-5 Verifying the installed operators

7. Create an Red Hat OpenShift Logging instance:

   a. Select **Administration** → **Custom Resource Definitions**, go to the Custom Resource Definitions page, and click **ClusterLogging**. On the ClusterLogging page, click **Create ClusterLogging**.

   b. In the YAML field, replace the code with the code that is shown in Example 7-4. Click **Create**.

Example 7-4 Sample YAML file for a ClusterLogging instance

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    retentionPolicy:
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: "storage-block-gold"
        size: 200G
      resources:
          limits:
            memory: "16Gi"
          requests:
            memory: "16Gi"
      proxy:
        resources:
          limits:
            memory: 256Mi
          requests:
            memory: 256Mi
```

```
    redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana"
  kibana:
    replicas: 1
collection:
  logs:
    type: "fluentd"
    fluentd: {}
```

8. Verify the pods, as shown in Figure 7-6.

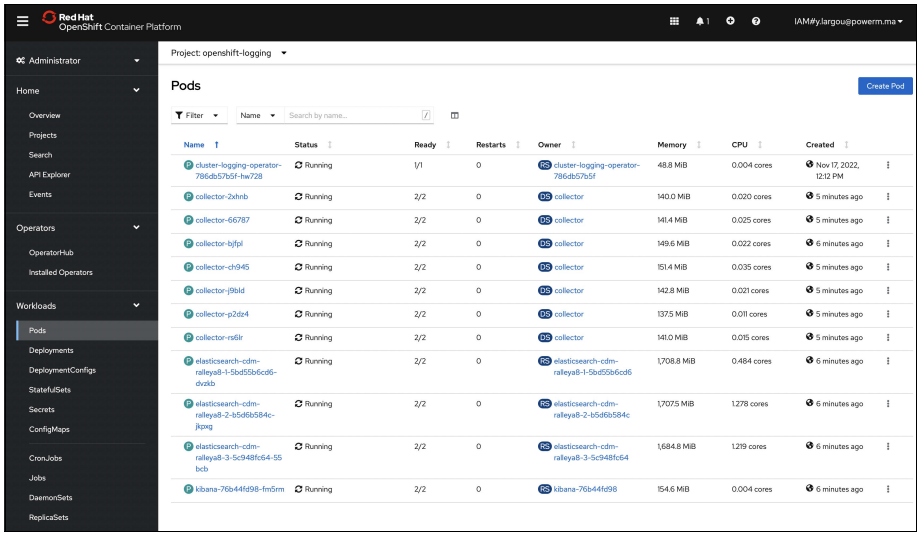


Figure 7-6 Verifying the pods

9. Verify the persistent volumes (PVs), as shown in Figure 7-7.

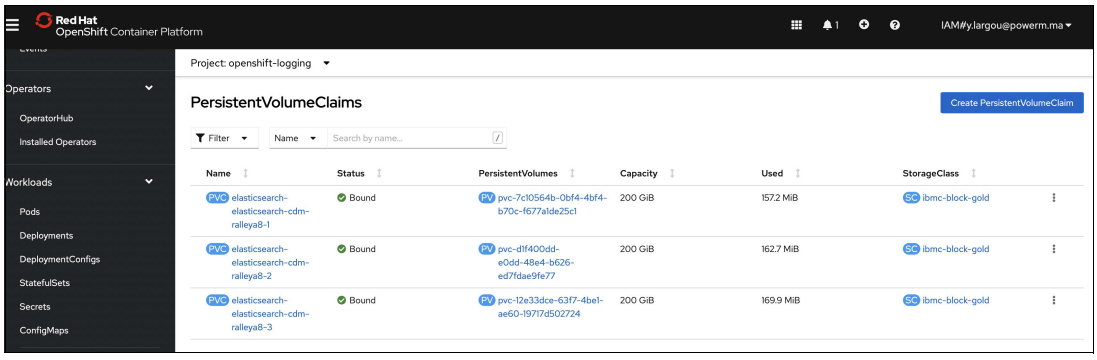


Figure 7-7 Verifying the persistent volumes

7.2.3 Using the logging subsystem for Red Hat OpenShift

To explore and visualize log data, you can use Kibana. To do so, complete the following steps:

1. Select **Developer** → **Topology**, and click the **kibana** pod, as shown in Figure 7-8.

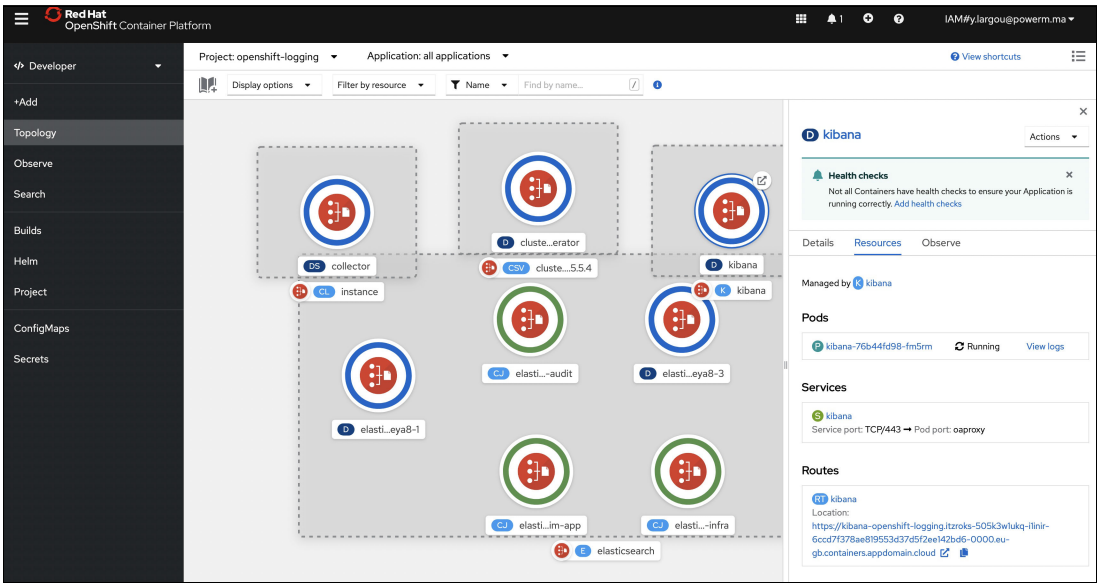


Figure 7-8 Pod topology

2. Create the index and use Kibana, as shown in Figure 7-9.

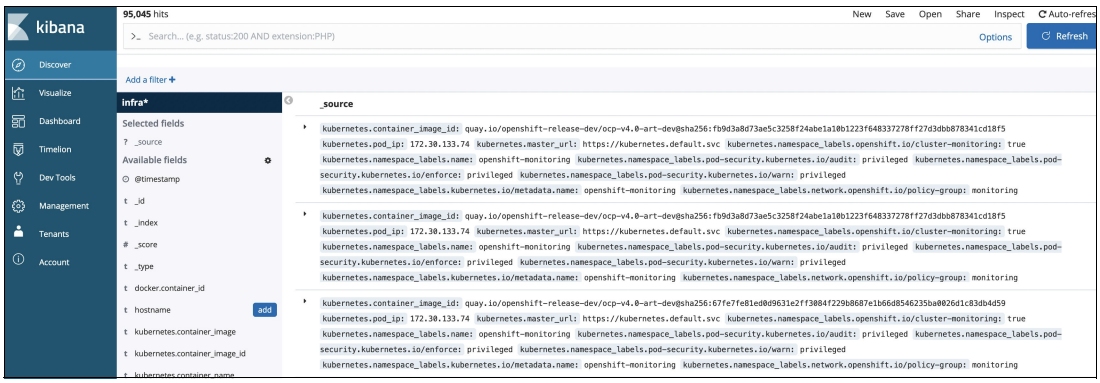


Figure 7-9 Using Kibana

7.3 Red Hat OpenShift File Integrity Operator monitoring

Red Hat OpenShift File Integrity Operator continuously examines the cluster nodes' file integrity. On each node, it starts a daemon set that runs privileged Advanced Intrusion Detection Environment (AIDE) containers and provides a record of files that were modified since the DaemonSet initially ran.

Red Hat OpenShift File Integrity Operator creates a database that is based on regular expression rules that are in a configuration file. When the database is initialized, it can be used to verify the integrity of files. It has several message digest algorithms for checking file integrity, and file attributes also can be checked for inconsistencies.

For more information, see the integrity operator release notes.

7.3.1 Installing Red Hat OpenShift File Integrity Operator

To install Red Hat OpenShift File Integrity Operator, complete the following steps:

1. Under the Red Hat OpenShift Dashboard (select **Operators** → **OperatorHub**), search for `File Integrity Operator`, as shown in Figure 7-10.

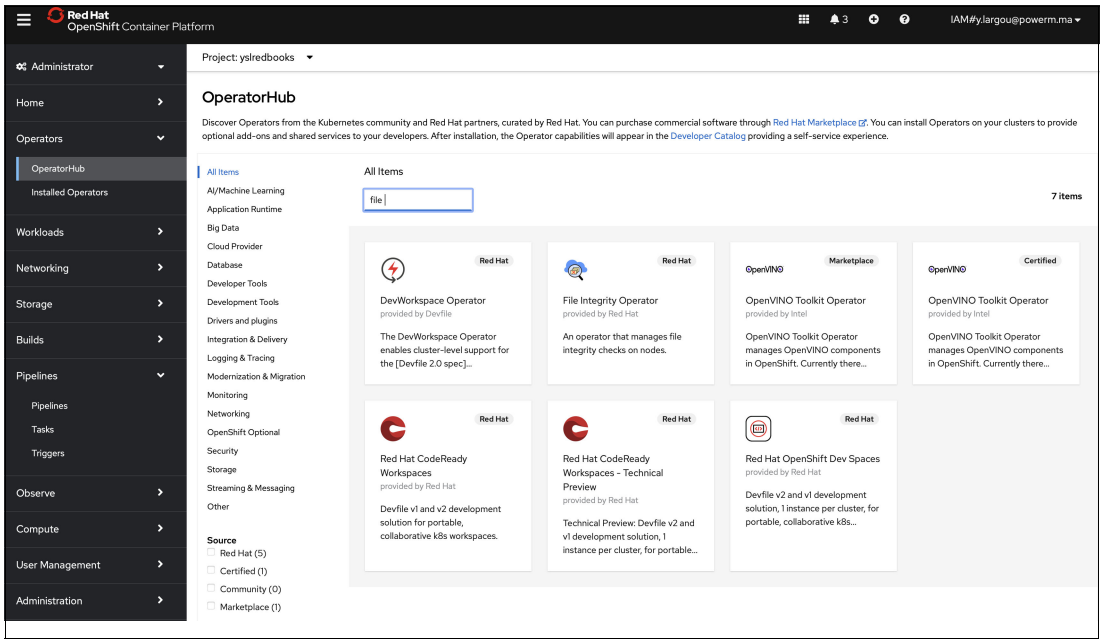


Figure 7-10 OperatorHub page

2. Install Red Hat OpenShift File Integrity Operator with the default parameters, as shown in Figure 7-11.

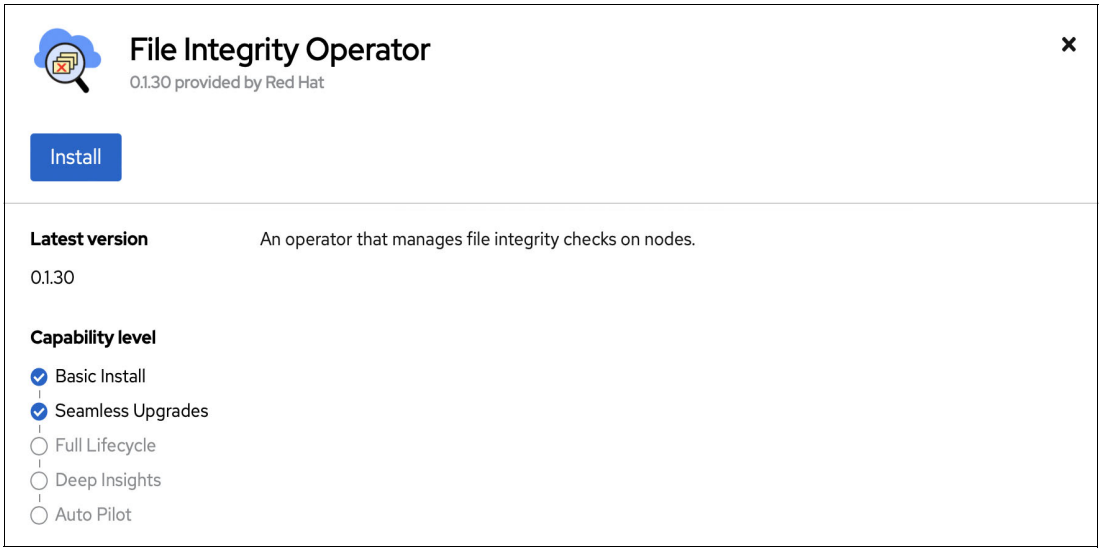


Figure 7-11 Installing Red Hat OpenShift File Integrity Operator

3. Under the Red Hat OpenShift Dashboard (select **Operators** → **Installed Operators**), check that Red Hat OpenShift File Integrity Operator is correctly installed, as shown in Figure 7-12.

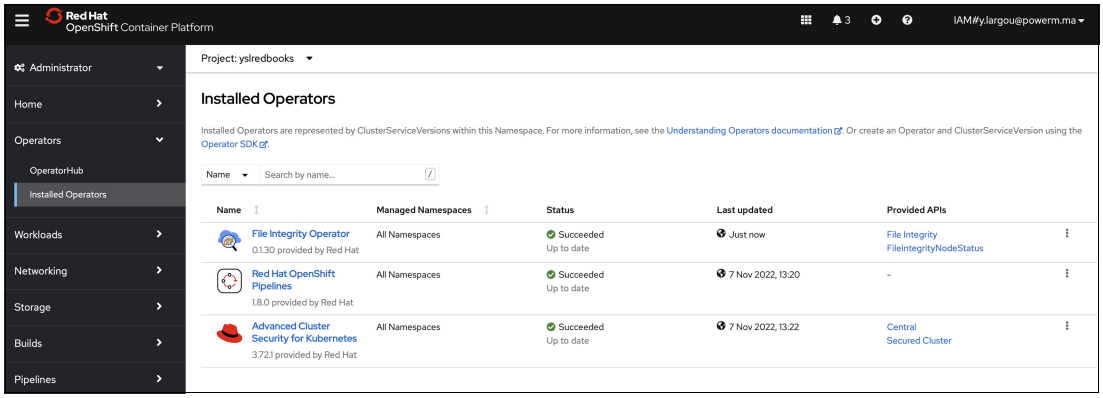


Figure 7-12 Verifying the installed operators

7.3.2 Configuring Red Hat OpenShift File Integrity Operator

To configure Red Hat OpenShift File Integrity Operator, complete the following steps:

1. Under the Red Hat OpenShift Dashboard (select **Operators** → **Installed Operators**), select **File Integrity Operator** and click **Create Instance**, as shown in Figure 7-13.

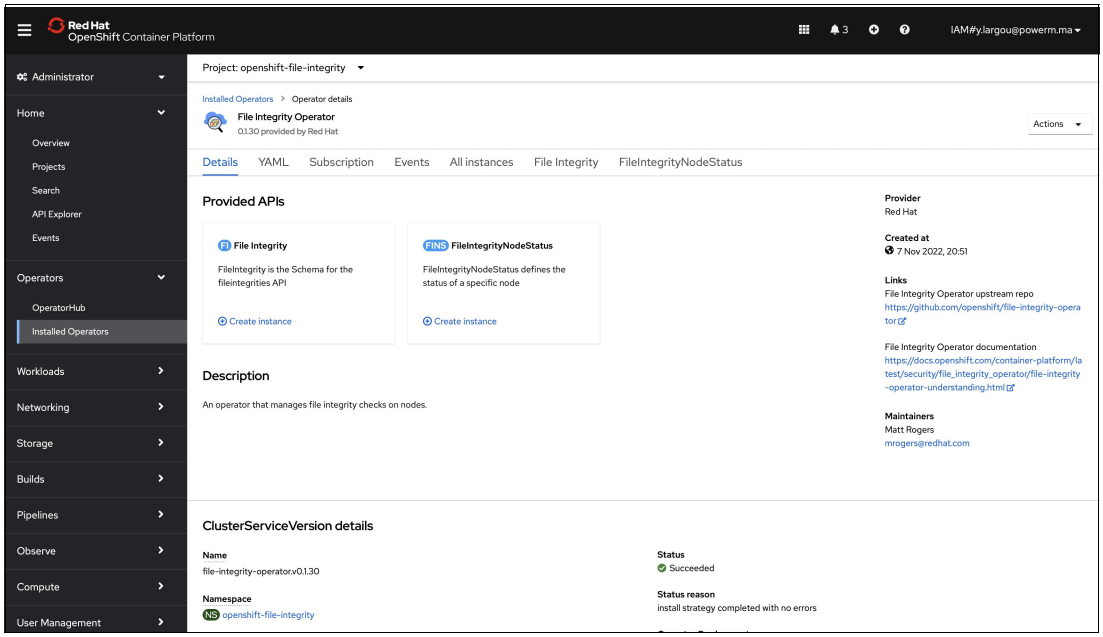


Figure 7-13 File Integrity Operator Page

2. Create a FileIntegrity CR that is named worker-fileintegrity.yaml to enable scans on worker nodes, as shown in Example 7-5.

Example 7-5 The worker-fileintegrity.yaml file

```
apiVersion: fileintegrity.openshift.io/v1alpha1
kind: FileIntegrity
metadata:
  name: worker-fileintegrity
  namespace: openshift-file-integrity
spec:
  config:
    gracePeriod: 900
    maxBackups: 5
  tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/master
      operator: Exists
  debug: true
```

3. Apply the YAML file to the openshift-file-integrity namespace, as shown in Example 7-6.

Example 7-6 Applying the YAML file to the openshift-file-integrity namespace

```
y_largou@cloudshell:~$ oc apply -f worker-fileintegrity.yaml -n
openshift-file-integrity
```

4. Under the File Integrity column, check that the FileIntegrity instance is correctly configured, as shown in Figure 7-14.

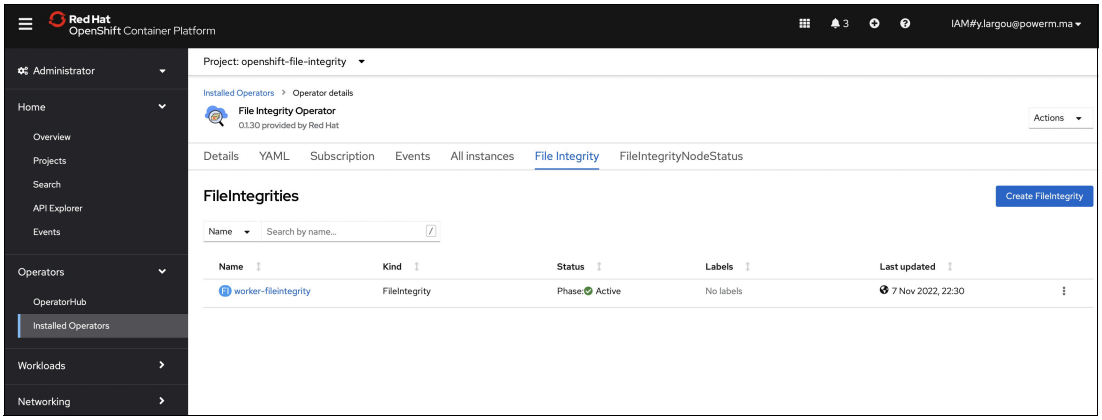


Figure 7-14 The FileIntegrities page

5. Under the FileIntegrityNodeStatus column, check that the status of FileIntegrityNode, as shown in Figure 7-15 on page 137.

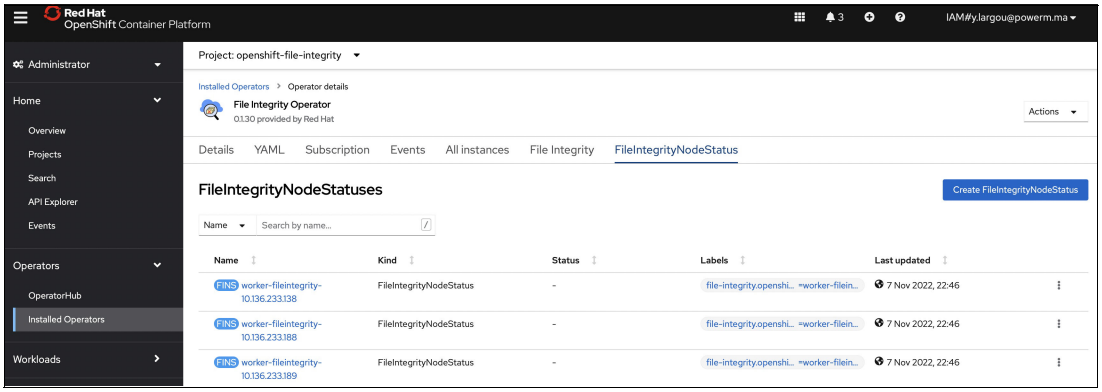


Figure 7-15 The FileIntegrityNodeStatus page

When a node scan fails, an event is created with the add, changed, removed, and configmap information. To get the list of events, run the command that is shown in Example 7-7.

Example 7-7 Applying the YAML file to the openshift-file-integrity namespace

```
y_largou@cloudshell:~$ oc get events --field-selector reason=FileIntegrityStatus -n
openshift-file-integrity
LAST SEEN   TYPE     REASON                OBJECT                                  MESSAGE
16m         Normal   FileIntegrityStatus   fileintegrity/worker-fileintegrity      Pending
16m         Normal   FileIntegrityStatus   fileintegrity/worker-fileintegrity      Initializing
16m         Normal   FileIntegrityStatus   fileintegrity/worker-fileintegrity      Active

y_largou@cloudshell:~$  oc get events --field-selector reason=NodeIntegrityStatus -n
openshift-file-integrity
LAST SEEN   TYPE     REASON                OBJECT                                  MESSAGE
57s         Normal   NodeIntegrityStatus   fileintegrity/worker-fileintegrity      no changes to node
10.136.233.189
52s         Normal   NodeIntegrityStatus   fileintegrity/worker-fileintegrity      no changes to node
10.136.233.138
44s         Normal   NodeIntegrityStatus   fileintegrity/worker-fileintegrity      no changes to node
10.136.233.188
```

8

Compliance and regulation

Regulations about how an enterprise handles data and security are present worldwide. These regulations can be complex and subject to change over time based on actions by different governing bodies.

This chapter describes some of the methods and tools that can be used to ensure that your enterprise is in compliance with those regulations.

This chapter describes the following topics:

- Regulations and compliance
- IBM Cloud Security and Compliance Center
- OpenSCAP for Red Hat OpenShift
- Red Hat OpenShift Compliance Operator
- Red Hat OpenShift Machine Config Operator
- IBM Hyper Protect Crypto Services

8.1 Regulations and compliance

Today, there is a huge focus on data privacy, and new regulations are being enacted around the world that affect the way that enterprises handle privacy and security. These regulations drive how private and public organizations must handle and protect personal data. There are other critical regulations in place to guide the same considerations for health data, military information, and other data such as financial data. An enterprise's compliance posture in terms of these various regulations can be the difference between success and failure.

8.1.1 Introduction

The number of standards and regulations with which a company must comply is huge and variable, depending on the industry. Some of these standards and regulations are common like General Data Protection Regulation (GDPR) in Europe and Lei Geral de Proteção de Dados (LGPD) in Brazil. They aim to protect personal data by defining how companies handle data and how they must comply with the preferences regarding your personal data. Although this situation might not seem to be directly related to security, it must be a consideration for organizations that collect and store personal data. The organization must protect the data and ensure that the data is not used for purposes to which the data owner did not agree to or for criminal purposes.

There is not a line separating compliance and security requirements because sometimes they can be thought of as the same thing, and in fact are complementary:

- Security encompasses technical requirements and technologies that are used to protect either an application or its data from external and internal threats.
- Compliance is related to standards and rules to which the infrastructure and application must comply.

In general, security practices must be implemented to help comply with a certain standard, such as GDPR. Compliance with the standard might dictate where a certain type of data must reside physically, and the architecture in place must ensure that the rule is met while providing the users with the capabilities that are expected from that application. These needs are met by using security techniques that involve data stores, data management, and encryption.

8.1.2 Security and compliance in the cloud

Security is one of the top concerns in cloud computing adoption. To help customers feel more comfortable with moving to the cloud, IBM recently announced IBM Cloud for Financial Services®, which is designed to provide an environment that is ready to host the most critical workloads in the financial world on IBM Cloud.

IBM Cloud for Financial Services is designed to help clients mitigate risk and accelerate cloud adoption for even their most sensitive workloads. Security and controls are built in to the platform to enable financial institutions to automate their security and compliance posture and demonstrate their regulatory compliance posture, and for their clients to simplify their risk management.

Cloud computing has different service and deployment models, which means security and compliance must adapt to those different perspectives by matching the requirements of the scenario to be adopted. As is true with many IT environments, there is no one-size-fits-all solution. This paper is focused on infrastructure as a service (IaaS) and platform as a service (PaaS) by using public cloud, private cloud, and hybrid cloud for deployment models.

8.1.3 Infrastructure as a service

On this service model, the cloud provider owns the hardware, networking, and storage, and they provide the virtualization techniques that are required to enable multiple customers to share an environment. In a private cloud, the scenario is the same, and the only difference is the customers, which in this case are going to be different departments within the same organization.

The common security challenges for IaaS normally are around the following items:

- Environment misconfiguration: Misconfiguration results in virtual machines (VMs) or logical partitions (LPARs) that might have some exposure to other external users who were not originally included by the cloud consumer.
- Data encryption: Considering how data flows from and to the cloud environment means that you must protect the data. The data may be encrypted at the source and flow to the cloud encrypted, or it may be encrypted in the cloud. Most of modern solutions today rely on encryption mechanisms for data transmission, like HTTPS and Secure FTP. Most cloud database services offer encryption at rest, which is required by different standards dealing with sensitive personal information.

8.1.4 Platform as a service

On this service model, the cloud provider owns the infrastructure as in IaaS, but they also own the databases and run times. This model is targeted at software developers so that they can spend most of their time developing new solutions instead of maintaining the environment. There are pros and cons of this scenario because security must be part of the development lifecycle to ensure that delivered solutions in production comply with applicable standards, and that the company is secured against malicious threats and bad users.

Some best practices to be considered are as follows:

- Check the cloud provider's ability to offer the security mechanisms that are required by the company by using benchmarks, external audits, and contract reviews by legal SMEs.
- *Threat modeling*, in a simplified view, consists of analyzing and documenting potential threats and deciding how to deal with them to avoid any exposure. This task can be done for applications, infrastructure, networking, and even business processes, so it is not exclusive to PaaS solutions, but it is important to be done here because it adds the security and compliance mindset to the development team while giving them the freedom of not having to handle cloud infrastructure, databases, and runtime configurations. Thus, it is critical that the solution is developed to be secure in its design.
- Software vulnerabilities must be identified and re-mediated as soon as possible.

8.1.5 Private cloud

All of a private cloud infrastructure is used by a single consumer. A private cloud can be owned by a cloud provider and physically located outside the consumer's building. It is private because the cloud is not accessible by anyone else but the consumer that contracted for it.

There are some security advantages because the cloud environment is isolated from external access (integration points must be carefully exposed). The consumer has greater control over the resources, which can help with compliance and security requirements, for example, data encryption demands.

8.1.6 Public cloud

A public cloud is the opposite of a private cloud. The infrastructure is shared among different organizations, which are called cloud consumers. The environment is accessible through the internet, which brings some concerns and more security requirements, for example, the need to intercept a distributed denial-of-service (DDoS) attack.

8.1.7 Hybrid cloud

Hybrid comes from the fact that a software solution often encompasses components that are hosted in more than one cloud environment. There are mechanisms to offer the cloud consumer a view of all their components that are hosted in different environments, which means security is one aspect to be monitored. Security challenges are centered on the data exchange among different cloud environments and integration points among the components working together.

8.1.8 Compliance posture

The compliance posture of an organization consists of the technologies, manual and automated procedures, applicable regulations, and training that is offered to its employees to ensure that everyone that is involved in the product development and management is focused on keeping the infrastructure, application, and the data safe from malicious intruders. Again, it is worth mentioning that “compliance” here is referring to the combination of external and internal regulations plus security requirements.

A breakdown of a compliance posture includes the following items:

- Preparation:
  - Threat-modeling or similar methods can be used to assess the security and regulatory risks that are involved for a new or changed infrastructure, or a software component.
  - Select a training roadmap for all IT personnel, including, for example, Open Web Application Security Project (OWASP) concepts and references.
  - Map the tools and frameworks that are used to develop and maintain the IT department, like programming languages, core libraries, build platforms, and dependencies.
  - Add security checks to the continuous integration and continuous delivery (CI/CD) pipelines.

- Protect:
  - Define the approved repositories for source code.
  - Protect the development and build environments to prevent access by people without a valid business need.
  - Use a static code scan and an open-source vulnerabilities scan.
  - Automatically check for secrets or passwords in the source code and keep them from being saved in the repository.
  - Perform security tests before putting a new release into production.
  - Whenever possible, digitally sign the container images.
  - Foster a culture of removing unused code and libraries from software in production.

This list depends on the security requirements of different levels of software and infrastructure. Section 3.2, “Seven layer security model” on page 30 covers seven layers of cybersecurity, which identifies the most important aspects of a security posture. Chapter 4, “Designing and implementing Red Hat OpenShift with security first” on page 43 covers specific aspects that are related to Red Hat OpenShift.

There are many ways for an enterprise to manage security and compliance. For more information about how IBM handles its IT security for internal operations, see IBM Enterprise IT Security.

8.2 IBM Cloud Security and Compliance Center

Compliance and security requirements span multiple layers of software and hardware, including the hardware itself and physical location of the servers. The number of variables is significant and complex. Cloud providers invest in tools and processes to automate and help security and compliance leaders make sure that the compliance posture is implemented. Companies might rely on some components running on-premises, some components in public clouds with different providers, and some components in a private cloud.

With IBM Cloud Security and Compliance Center, you can manage a hybrid cloud by converting your compliance posture definitions to goals, which are assessed during automated scans of all the IT resources[1] and then shown in a dashboard. Detailed reports also can be downloaded and used, for example, audit purposes. For more information, see Getting started with IBM Cloud Security and Compliance Center.

8.2.1 How IBM Cloud Security and Compliance Center works

There must be a collector to run the assessments, and that collector depends on other components to work. The scope that is analyzed, for example, covers the IBM Cloud Account as the target, the goals that represent the security and compliance requirements, and the credentials that are required to complete the scan. When the setup is complete, a schedule is defined for running the assessments, and then the results are available from a dashboard.

[1] Available for IBM Cloud, Amazon Web Services, Microsoft Azure, Google Cloud Platform (GCP), and on-premises environments.

Figure 8-1, from IBM Cloud documentation,[2] shows how these elements work together.

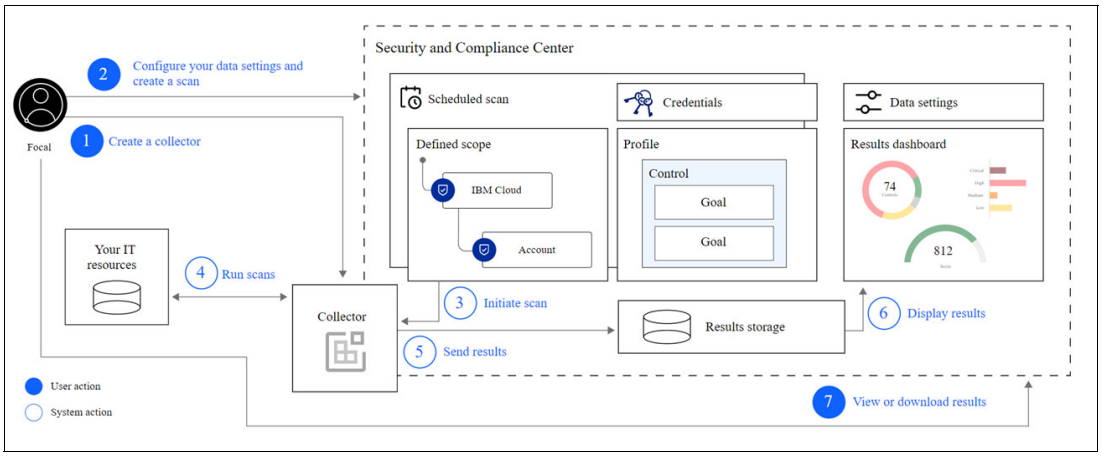


Figure 8-1 IBM Security and Compliance Center

The collector is a module that is installed on an infrastructure that has access to the IT resources to are analyzed by following the schedule that is defined by the user in IBM Cloud Security and Compliance Center.

When it is time, the collector receives a signal to trigger the new assessment against the scope by using the credentials that are provided and the security and compliance requirements that are determined as goals belonging to a profile. The results are consolidated in a dashboard.

There are two types of collectors:

- The IBM managed collector is installed on the IBM infrastructure as a Universal Base Image (UBI),[3] which means that IBM is responsible for the lifecycle of the collector. There are some limitations, such as one collector per account, and no access to on-premises environments.
- There is an alternative collection option that is called a customer-managed collector. It can be installed as an UBI or as an Ubuntu image.

Note: The Ubuntu image is not compliant with Federal Information Processing Standards (FIPS).

Managing collectors provides more information about the following items:

- Collectors and how data is collected and analyzed.
- More considerations for IBM managed collectors versus customer-managed collectors.
- How communication works between collectors and IBM Cloud Security and Compliance Center and between collectors and the IT resources that are analyzed.

[2] https://cloud.ibm.com/docs-content/v1/content/a72a51ea6aec7e72e86cea5d12415b10061aee44/security-compliance/images/posture.svg
[3] https://www.redhat.com/en/blog/introducing-red-hat-universal-base-image

8.2.2 Connecting Red Hat OpenShift Compliance Operator

Red Hat OpenShift Compliance Operator, when deployed to a Red Hat OpenShift on IBM Cloud cluster, can be integrated in to IBM Cloud Security and Compliance Center so that the scan results are available in a central location along with the results of other clusters that are configured in the IBM Cloud Security and Compliance Center.

Figure 8-2 is taken from IBM Cloud documentation[4] and shows the steps that required to implement the integration.

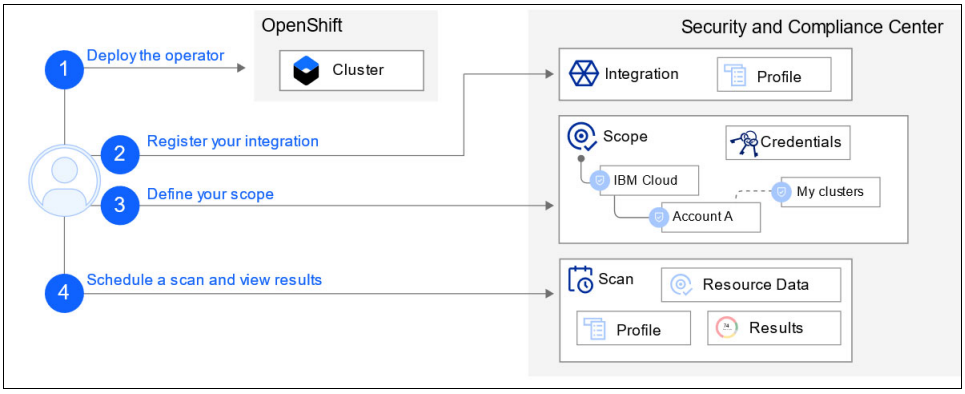


Figure 8-2 Red Hat OpenShift integration for IBM Cloud Security and Compliance

For more information about how to install and configure Red Hat OpenShift Compliance Operator, see 8.3, “OpenSCAP for Red Hat OpenShift” on page 145. When Red Hat OpenShift Compliance Operator is ready to use, the next step is to register it as a new integration in IBM Cloud Security and Compliance Center so that you can create a scan the Red Hat OpenShift cluster as the target. For more information about instructions to complete this setup, see IBM Cloud documentation.

8.3 OpenSCAP for Red Hat OpenShift

Security Content Automation Protocol (SCAP) is a line of specifications that is managed by the National Institute of Standards and Technology (NIST) for maintaining systems security.

SCAP is implemented by the OpenSCAP application, and it is available for Red Hat OpenShift as an operator. Red Hat OpenShift Compliance Operator keeps the cluster compliant with the required security benchmarks and provides remediations for the issues that are found.

For more information about OpenSCAP, see OpenSCAP.

Red Hat OpenShift uses core OS by default, which has an immutable Red Hat Enterprise Linux (RHEL) kernel that is SELinux Enforced. To help secure containerized workloads, Red Hat also provides a UBI, which is an OCI-compliant secure foundation for cloud-native microservices.

[4] https://cloud.ibm.com/docs-content/v1/content/a72a51ea6aec7e72e86cea5d12415b10061aee44/security-compliance/images/osco.svg

Scan each cloud-native application for vulnerabilities by using Red Hat Quay. Figure 8-3 illustrates how Red Hat Quay flags vulnerable containers before being deployed.

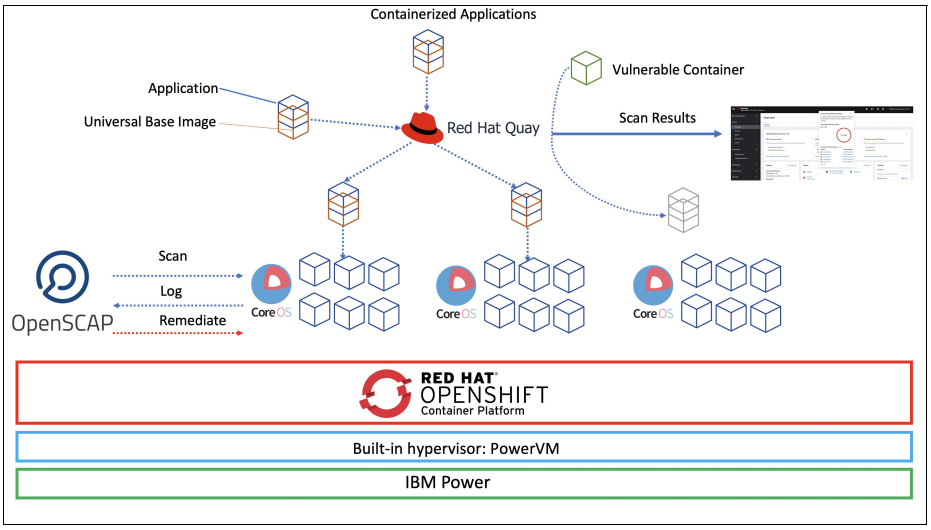


Figure 8-3 OpenSCAP and Red Hat OpenShift security highlights

OpenSCAP is available in a Git Repo as an Operator, and it is used to audit, log, and re-mediate the Red Hat OpenShift Infrastructure.

Configuring and using Red Hat OpenShift Compliance Operator

To use OpenSCAP for infrastructure scanning and remediation, complete the following steps:

1. Connect to IBM Cloud Shell and connect to Red Hat OpenShift Cluster on IBM Power Systems Virtual Server (IBM PowerVS), as shown in Example 8-1.

Example 8-1 Verifying the cluster version

```
Welcome to IBM Cloud Shell!
Image version: 1.0.67

Note: Your Cloud Shell session is running in Frankfurt (eu-de). Your workspace includes 500 MB of
temporary storage. This session will close after an hour of inactivity. If you don't have any active
sessions for an hour or you reach the 50-hour weekly usage limit, your workspace data is removed. To
track your usage, go to Usage quota in the Cloud Shell menu.

Tip: Enter 'ibmcloud' to use the IBM Cloud CLI. The Frankfurt (eu-de) region is targeted by default.
You can switch the region by running 'ibmcloud target -r <region-name>'.

y_largou@cloudshell:~$ oc login --token=sha256~8_46cNOPgOd84vS6c3MBSbjeFeqHe-wxh6M9n1pTOEM
--server=https://c102-e.eu-de.containers.cloud.ibm.com:30907
Logged into "https://c102-e.eu-de.containers.cloud.ibm.com:30907" as "IAM#y.largou@powerm.ma" using
the token provided.

You have access to 70 projects, the list has been suppressed. You can list all projects with 'oc
projects'

Using project "default".

Welcome! See 'oc help' to get started.
```

2. Clone The OpenSCAP Git Repository, as shown in Example 8-2.

Example 8-2 Cloning the OpenSCAP Git Repository

```
y_largou@cloudshell:~$ mkdir openscap
y_largou@cloudshell:~$ cd openscap
y_largou@cloudshell:~/openscap$ git clone https://github.com/openshift/compliance-operator
Cloning into 'compliance-operator'...
remote: Enumerating objects: 22817, done.
remote: Total 22817 (delta 0), reused 0 (delta 0), pack-reused 22817
Receiving objects: 100% (22817/22817), 30.46 MiB | 9.57 MiB/s, done.
Resolving deltas: 100% (11928/11928), done.
Updating files: 100% (4877/4877), done.
y_largou@cloudshell:~/openscap$
```

3. Create the `openshift-compliance` namespace, as shown in Example 8-3.

Example 8-3 Creating the namespace

```
y_largou@cloudshell:~/openscap$ cd openscap/compliance-operator/deploy
y_largou@cloudshell:~/openscap/compliance-operator/deploy$ oc create -f ns.yaml
namespace/openshift-compliance created
```

4. Create an OpenSCAP operator, as shown in Example 8-4.

Example 8-4 Creating an OpenSCAP operator

```
y_largou@cloudshell:~/openscap$ cd openscap/compliance-operator/deploy/crd
y_largou@cloudshell:~/openscap/compliance-operator/deploy/crds$ for i in $(ls -1 *crd.yaml); do oc
create -f $i ; done

customresourcedefinition.apiextensions.k8s.io/compliancecheckresults.compliance.openshift.io created
customresourcedefinition.apiextensions.k8s.io/complianceremediations.compliance.openshift.io created
customresourcedefinition.apiextensions.k8s.io/compliancescans.compliance.openshift.io created
customresourcedefinition.apiextensions.k8s.io/compliancesuites.compliance.openshift.io created
customresourcedefinition.apiextensions.k8s.io/profilebundles.compliance.openshift.io created
customresourcedefinition.apiextensions.k8s.io/profiles.compliance.openshift.io created
customresourcedefinition.apiextensions.k8s.io/rules.compliance.openshift.io created
customresourcedefinition.apiextensions.k8s.io/scansettingbindings.compliance.openshift.io created
customresourcedefinition.apiextensions.k8s.io/scansettings.compliance.openshift.io created
customresourcedefinition.apiextensions.k8s.io/tailoredprofiles.compliance.openshift.io created
customresourcedefinition.apiextensions.k8s.io/variables.compliance.openshift.io created
y_largou@cloudshell:~/openscap/compliance-operator/deploy/crds$
```

5. Deploy the OpenSCAP operator (ignore the latest error because the namespace `openshift-compliance` already exists). Example 8-5 shows the results.

Example 8-5 Deploying the OpenSCAP operator

```
y_largou@cloudshell:~/openscap/compliance-operator$ oc create -f deploy/
deployment.apps/compliance-operator created
role.rbac.authorization.k8s.io/compliance-operator created
clusterrole.rbac.authorization.k8s.io/compliance-operator created
role.rbac.authorization.k8s.io/resultscollector created
role.rbac.authorization.k8s.io/api-resource-collector created
role.rbac.authorization.k8s.io/resultserver created
role.rbac.authorization.k8s.io/remediation-aggregator created
clusterrole.rbac.authorization.k8s.io/remediation-aggregator created
role.rbac.authorization.k8s.io/rerunner created
role.rbac.authorization.k8s.io/profileparser created
clusterrole.rbac.authorization.k8s.io/api-resource-collector created
rolebinding.rbac.authorization.k8s.io/compliance-operator created
clusterrolebinding.rbac.authorization.k8s.io/compliance-operator created
rolebinding.rbac.authorization.k8s.io/resultscollector created
```

```
rolebinding.rbac.authorization.k8s.io/remediation-aggregator created
clusterrolebinding.rbac.authorization.k8s.io/remediation-aggregator created
clusterrolebinding.rbac.authorization.k8s.io/api-resource-collector created
rolebinding.rbac.authorization.k8s.io/api-resource-collector created
rolebinding.rbac.authorization.k8s.io/rerunner created
rolebinding.rbac.authorization.k8s.io/profileparser created
rolebinding.rbac.authorization.k8s.io/resultserver created
serviceaccount/compliance-operator created
serviceaccount/resultscollector created
serviceaccount/remediation-aggregator created
serviceaccount/rerunner created
serviceaccount/api-resource-collector created
serviceaccount/profileparser created
serviceaccount/resultserver created
Error from server (AlreadyExists): error when creating "deploy/ns.yaml": namespaces
"openshift-compliance" already exists
```

6. Change default project to `openshift-compliance`, as shown in Example 8-6.

Example 8-6 Changing the project

```
y_largou@cloudshell:~/openscap/compliance-operator$ oc project openshift-compliance
Now using project "openshift-compliance" on server
"https://c102-e.eu-de.containers.cloud.ibm.com:30907".
```

7. Select a scan YAML file from the `crds` directory, as shown in Example 8-7.

Example 8-7 Verifying the cluster version

```
Welcome to IBM Cloud Shell!
Image version: 1.0.67
y_largou@cloudshell:~/openscap/compliance-operator/deploy/crds$ ls
compliance.openshift.io_compliancecheckresults_crd.yaml compliance.openshift.io_rules_crd.yaml
compliance.openshift.io_v1alpha1_compliancesuite_cr.yaml
compliance.openshift.io_complianceremediations_crd.yaml
compliance.openshift.io_scansettingbindings_crd.yaml
compliance.openshift.io_v1alpha1_profilebundle_cr.yaml
compliance.openshift.io_compliancescans_crd.yaml   compliance.openshift.io_scansettings_crd.yaml
compliance.openshift.io_v1alpha1_scansettingbinding_cr.yaml
compliance.openshift.io_compliancesuites_crd.yaml
compliance.openshift.io_tailoredprofiles_crd.yaml
compliance.openshift.io_v1alpha1_scansetting_cr.yaml
compliance.openshift.io_profilebundles_crd.yaml
compliance.openshift.io_v1alpha1_compliancescan_node_cr.yaml
compliance.openshift.io_v1alpha1_tailoredprofile_cr.yaml
compliance.openshift.io_profiles_crd.yaml
compliance.openshift.io_v1alpha1_compliancescan_platform_cr.yaml
compliance.openshift.io_variables_crd.yaml
```

In this example, we use the `compliance.openshift.io_v1alpha1_compliancesuite_cr.yaml` file, as shown in Example 8-8.

Example 8-8 The compliance.openshift.io_v1alpha1_compliancesuite_cr.yaml file

```
apiVersion: compliance.openshift.io/v1alpha1
kind: ComplianceSuite
metadata:
  name: example-compliancesuite
spec:
  autoApplyRemediations: false
  schedule: "0 1 * * *"
  scans:
    - name: workers-scan
```

```
    profile: xccdf_org.ssgproject.content_profile_moderate
    content: ssg-rhcos4-ds.xml
    contentImage: quay.io/compliance-operator/compliance-operator-content:latest
    nodeSelector:
      node-role.kubernetes.io/worker: ""
  - name: platform-scan
    scanType: Platform
    profile: xccdf_org.ssgproject.content_profile_moderate
    content: ssg-ocp4-ds.xml
    contentImage: quay.io/compliance-operator/compliance-operator-content:latest
```

8. Start the scan, as shown in Example 8-9.

Example 8-9 Starting the scan

```
y_largou@cloudshell:~/openscap/compliance-operator/deploy/crds$ oc create -f
compliance.openshift.io_v1alpha1_compliancesuite_cr.yaml
compliancesuite.compliance.openshift.io/example-compliancesuite created
```

9. Monitor the cluster scan, as shown in Example 8-10.

Example 8-10 Monitoring the cluster scan

```
y_largou@cloudshell:~/openscap/compliance-operator/deploy/crds$  oc get pods -w
NAME                                           READY   STATUS              RESTARTS        AGE
compliance-operator-76cd8fbdd-gmwck            1/1     Running             1 (2m43s ago)   2m53s
example                                        1/1     Running             0               16h
httpd-example-1-build                          0/1     Completed           0               16h
httpd-example-1-deploy                         0/1     Completed           0               16h
httpd-example-1-n4m6c                          1/1     Running             0               16h
ocp4-default-pp-784f4d456-6d8xn                1/1     Running             0               2m33s
ocp4-openshift-compliance-pp-7c58985c75-5gmbl  1/1     Running             0               2m33s
platform-scan-api-checks-pod                   0/2     Init:1/2            4 (58s ago)     2m23s
platform-scan-rs-785c7d7b66-j672x              0/1     ContainerCreating   0               2m23s
rhcos4-default-pp-54d894695d-xjt4n             1/1     Running             0               2m33s
rhcos4-openshift-compliance-pp-bd74cd8d7-x6g4z 1/1     Running             0               2m33s
spring-boot-954f64c46-dwxdj                    0/1     ImagePullBackOff    0               16h
workers-scan-10.136.233.138-pod                1/2     NotReady            0               2m23s
workers-scan-10.136.233.188-pod                1/2     NotReady            0               2m23s
workers-scan-10.136.233.189-pod                1/2     NotReady            0               2m23s
workers-scan-rs-548667f98-xcj25                0/1     ContainerCreating   0               2m23s
platform-scan-rs-785c7d7b66-j672x              0/1     ContainerCreating   0               2m28s
platform-scan-rs-785c7d7b66-j672x              0/1     ContainerCreating   0               2m28s
platform-scan-rs-785c7d7b66-j672x              1/1     Running             0               2m30s
workers-scan-rs-548667f98-xcj25                0/1     ContainerCreating   0               2m41s
workers-scan-rs-548667f98-xcj25                0/1     ContainerCreating   0               2m41s
platform-scan-api-checks-pod                   0/2     Init:CrashLoopBackOff   4 (15s ago)     2m43s
workers-scan-rs-548667f98-xcj25                1/1     Running             0               2m43s
workers-scan-10.136.233.138-pod                0/2     Completed           0               2m46s
workers-scan-10.136.233.188-pod                0/2     Completed           0               2m47s
workers-scan-10.136.233.138-pod                0/2     Completed           0               2m47s
spring-boot-954f64c46-dwxdj                    0/1     ImagePullBackOff    0               16h
workers-scan-10.136.233.188-pod                0/2     Completed           0               2m48s
workers-scan-10.136.233.138-pod                0/2     Completed           0               2m48s
workers-scan-10.136.233.188-pod                0/2     Completed           0               2m49s
workers-scan-10.136.233.189-pod                0/2     Completed           0               3m9s
workers-scan-10.136.233.189-pod                0/2     Completed           0               3m11s
workers-scan-10.136.233.189-pod                0/2     Completed           0               3m13s
aggregator-pod-workers-scan                    0/1     Pending             0               0s
aggregator-pod-workers-scan                    0/1     Pending             0               0s
aggregator-pod-workers-scan                    0/1     Init:0/1            0               0s
aggregator-pod-workers-scan                    0/1     Init:0/1            0               1s
aggregator-pod-workers-scan                    0/1     Init:0/1            0               1s
aggregator-pod-workers-scan                    0/1     PodInitializing     0               3s
aggregator-pod-workers-scan                    1/1     Running             0               4s
aggregator-pod-workers-scan                    0/1     Completed           0               18s
aggregator-pod-workers-scan                    0/1     Completed           0               19s
aggregator-pod-workers-scan                    0/1     Completed           0               20s
workers-scan-10.136.233.189-pod                0/2     Terminating         0               3m41s
workers-scan-10.136.233.189-pod                0/2     Terminating         0               3m41s
workers-scan-10.136.233.138-pod                0/2     Terminating         0               3m41s
workers-scan-10.136.233.138-pod                0/2     Terminating         0               3m41s
workers-scan-10.136.233.188-pod                0/2     Terminating         0               3m41s
workers-scan-10.136.233.188-pod                0/2     Terminating         0               3m41s
```

```
aggregator-pod-workers-scan                 0/1     Terminating            0              25s
aggregator-pod-workers-scan                 0/1     Terminating            0              25s
workers-scan-rs-548667f98-xcj25             1/1     Terminating            0              3m41s
workers-scan-rs-548667f98-xcj25             1/1     Terminating            0              3m41s
workers-scan-rs-548667f98-xcj25             0/1     Terminating            0              3m42s
workers-scan-rs-548667f98-xcj25             0/1     Terminating            0              3m42s
workers-scan-rs-548667f98-xcj25             0/1     Terminating            0              3m42s
aggregator-pod-workers-scan                 0/1     Pending                0              0s
aggregator-pod-workers-scan                 0/1     Pending                0              0s
aggregator-pod-workers-scan                 0/1     Init:0/1               0              0s
aggregator-pod-workers-scan                 0/1     Init:0/1               0              1s
aggregator-pod-workers-scan                 0/1     Init:0/1               0              1s
aggregator-pod-workers-scan                 0/1     PodInitializing        0              3s
aggregator-pod-workers-scan                 1/1     Running                0              4s
aggregator-pod-workers-scan                 0/1     Completed              0              7s
aggregator-pod-workers-scan                 0/1     Completed              0              8s
aggregator-pod-workers-scan                 0/1     Completed              0              9s
aggregator-pod-workers-scan                 0/1     Terminating            0              10s
aggregator-pod-workers-scan                 0/1     Terminating            0              10s
platform-scan-api-checks-pod                0/2     Init:1/2               5 (93s ago)    4m1s
```

10. Check the Compliance Remediations that were found, as shown in Example 8-11.

Example 8-11 Checking the compliance remediations that were found

```
y_largou@cloudshell:~/openscap/compliance-operator$ oc get -n openshift-compliance
complianceremediations
NAME                                                        STATE
workers-scan-auditd-name-format                             NotApplied
workers-scan-coredump-disable-backtraces                    NotApplied
workers-scan-coredump-disable-storage                       NotApplied
workers-scan-disable-ctrlaltdel-burstaction                 NotApplied
workers-scan-disable-users-coredumps                        NotApplied
workers-scan-grub2-audit-argument                           NotApplied
workers-scan-grub2-audit-backlog-limit-argument             NotApplied
workers-scan-grub2-page-poison-argument                     NotApplied
workers-scan-no-direct-root-logins                          NotApplied
```

11. To apply a remediation, edit that object and set its **Apply** attribute to `true`, as shown in Example 8-12.

Example 8-12 Applying remediation

```
y_largou@cloudshell:~/openscap/compliance-operator/deploy/crds$ oc edit -n
complianceremediation/workers-scan-no-direct-root-logins
```

12. Monitor the node status (after the nodes restart, run another suite to ensure that the remediation fixed the issue) by running the command in Example 8-13.

Example 8-13 Monitoring the node status

```
y_largou@cloudshell:~/openscap$oc get nodes -w
```

13. List the persistent volumes (PVs) that store the OpenSCAP logs, as shown in Example 8-14.

Example 8-14 Listing the persistent volumes

```
y_largou@cloudshell:~$ oc get pv
NAME                       CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS     REASON   AGE
pvc-0dfa5c03-abd9-4409-9539-70a2119a9582   20Gi       RWO            Delete           Bound
openshift-compliance/platform-scan                                          36m
pvc-129f02fe-1020-428c-8375-f82a7def30ad   20Gi       RWO            Delete           Bound
yslpowertst/pvc-2e9e0ecd58                                                  43h
pvc-15d63eb9-7ef6-469a-aeba-d08f22d41988   20Gi       RWO            Delete           Bound
yslpowertst/pvc-667273726e                                                  43h
```

```
pvc-2a034385-8dc8-4006-815f-c42f9720b695   20Gi       RWO            Delete           Bound
yslpowertst/pvc-1e4a57b316                                                          47h
pvc-38ef8c05-c94e-4528-9162-9ab98acfcd1d   100Gi      RWO            Delete           Bound
yslpowertst/stackrox-db                                                             47h
pvc-4ef86b8d-3736-4f9d-8dba-78bbf34f39c2   20Gi       RWO            Delete           Bound
openshift-compliance/pvc-4a4d6ceaa9                                                 16h
pvc-66ca0be0-bd1d-4316-b192-2e31b4398e92   20Gi       RWO            Delete           Bound
openshift-compliance/pvc-f2a04d3894                                                 16h
pvc-9eac44e9-b521-48a2-a6d2-4fe825eceb1c   20Gi       RWO            Delete           Bound
yslpowertst/pvc-faf4bdc7d4                                                          43h
pvc-e6c44da5-a91e-40ab-bb95-8759ed788540   20Gi       RWO            Delete           Bound
openshift-compliance/workers-scan                                                   37m
pvc-fdcf9d08-23cb-40ab-95ed-f428bdb649c1   100Gi      RWX            Delete           Bound
openshift-image-registry/image-registry-storage   ibmc-file-gold           47h

y_largou@cloudshell:~$ oc get pvc
NAME                          STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS      AGE
example-compliancescan-node   Bound    pvc-c25bf5ff-8563-488a-b97a-4a2c3a12f69f   20Gi       RWO
ibmc-block-gold   8m52s
platform-scan                 Bound    pvc-0dfa5c03-abd9-4409-9539-70a2119a9582   20Gi       RWO
ibmc-block-gold   51m
pvc-4a4d6ceaa9                Bound    pvc-4ef86b8d-3736-4f9d-8dba-78bbf34f39c2   20Gi       RWO
ibmc-block-gold   17h
pvc-f2a04d3894                Bound    pvc-66ca0be0-bd1d-4316-b192-2e31b4398e92   20Gi       RWO
ibmc-block-gold   17h
workers-scan                  Bound    pvc-e6c44da5-a91e-40ab-bb95-8759ed788540   20Gi       RWO
ibmc-block-gold   51m

y_largou@cloudshell:~$ oc get pvc/workers-scan
NAME           STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS      AGE
workers-scan   Bound    pvc-e6c44da5-a91e-40ab-bb95-8759ed788540   20Gi       RWO
ibmc-block-gold   52m
```

14. Start a pod that mounts the PV, as shown in Figure 8-4.

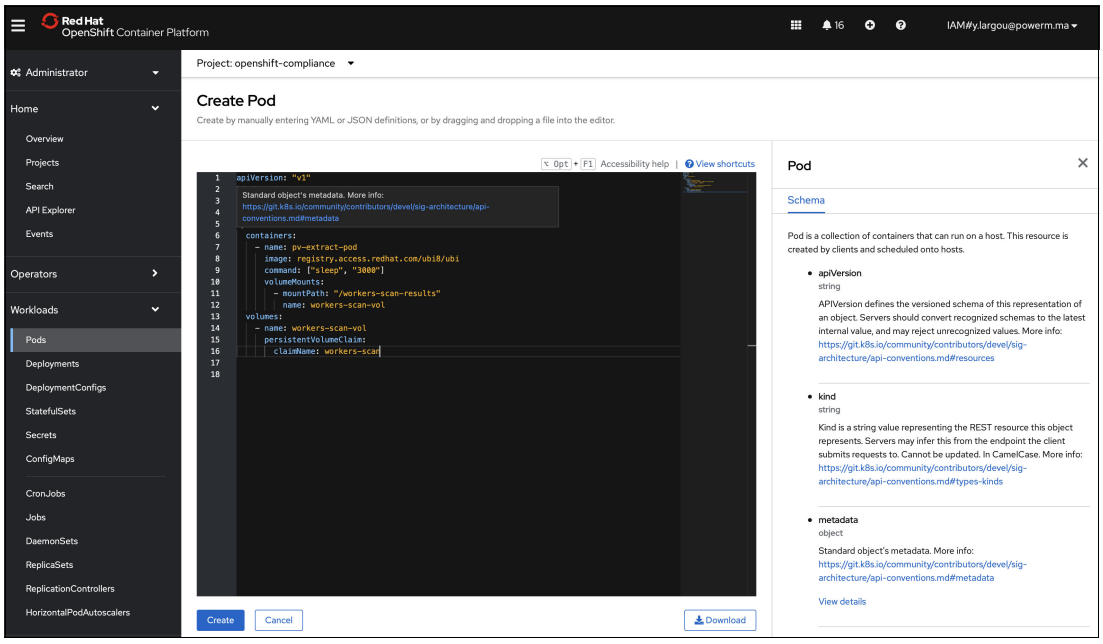


Figure 8-4 Running a pod

15. Extract the files to get the raw Assessment Results Format (ARF) files, as shown in Example 8-15.

Example 8-15 Extracting the raw ARF files

```
y_largou@cloudshell:~$ oc exec pods/pv-extract -- ls /workers-scan-results/0
workers-scan-10.136.233.138-pod.xml.bzip2
workers-scan-10.136.233.188-pod.xml.bzip2
workers-scan-10.136.233.189-pod.xml.bzip2
y_largou@cloudshell:~$  bunzip2 -c  workers-scan-10.136.233.138-pod.xml.bzip2 >
workers-scan-10.136.233.138-pod.xml
```

16. Extract the XCCDF results from the files, as shown in Example 8-16.

Example 8-16 Extracting the XCCDF results

```
y_largou@cloudshell:~$ oc get cm -l=compliance.openshift.io/scan-name=masters-scan
NAME                              DATA   AGE
workers-scan-10.136.233.138-pod   1      45m
workers-scan-10.136.233.138-pod   1      44m
workers-scan-10.136.233.138-pod   1      46m


$ oc extract cm/workers-scan-10.136.233.138-pod
```

8.4 Red Hat OpenShift Compliance Operator

With Red Hat OpenShift Compliance Operator, an administrator can run compliance scans and provide remediations for found anomalies in a Red Hat OpenShift cluster and the worker machines (nodes) running the cluster. Red Hat OpenShift Compliance Operator leverages OpenSCAP and community-based compliance content that is developed in the ComplianceAsCode/content project. This content project provides a bundle of security policies, 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.

8.4.1 Installing the Red Hat OpenShift Compliance Operator

To install the Red Hat OpenShift Compliance Operator, complete the following steps.

Prerequisites: You must have cluster admin privileges.

1. In the Red Hat OpenShift Container Platform web console, select **Operators** → **OperatorHub**.
2. Search for the Red Hat OpenShift Compliance Operator, as shown in Figure 8-5 on page 153.

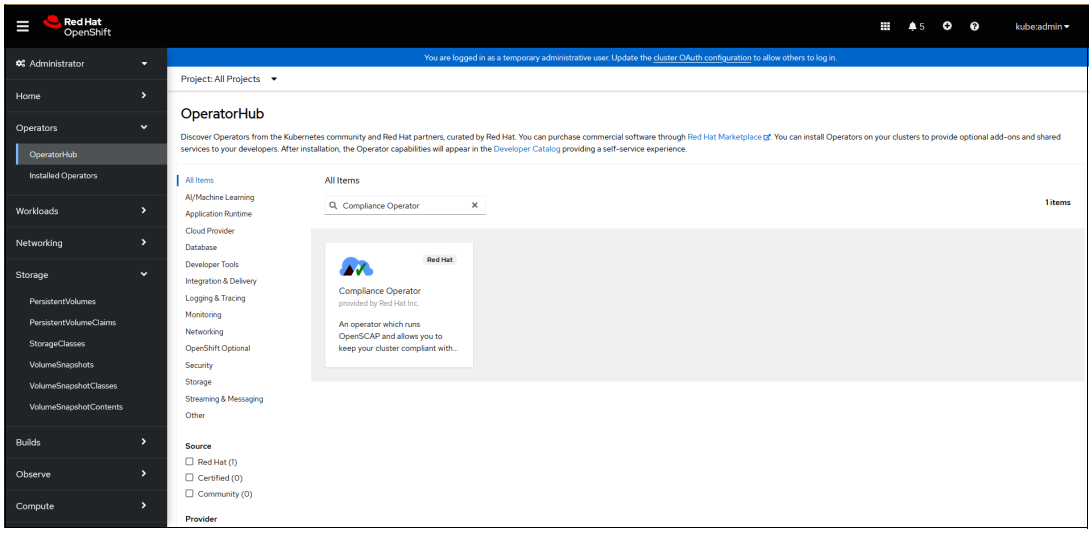


Figure 8-5 Searching for the Red Hat OpenShift Compliance Operator

3. Click **Compliance Operator**, and then click **Install**, as shown in Figure 8-6.

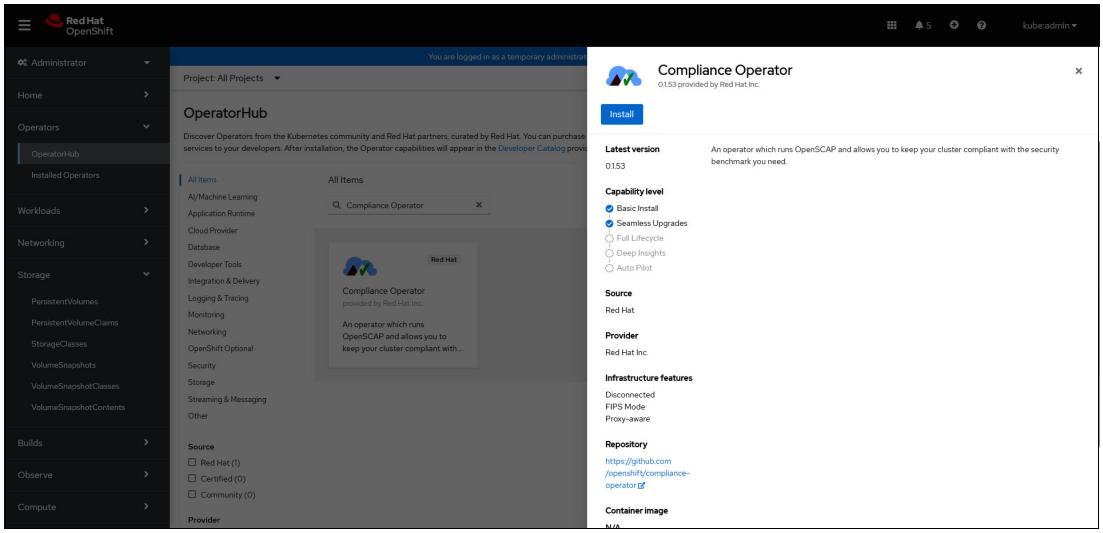


Figure 8-6 Red Hat OpenShift Compliance Operator Installation page

Note: Keep the default selection of Installation mode and namespace to ensure that the operator is installed to the `openshift-compliance` namespace.

4. Click **Install**, as shown in Figure 8-7.

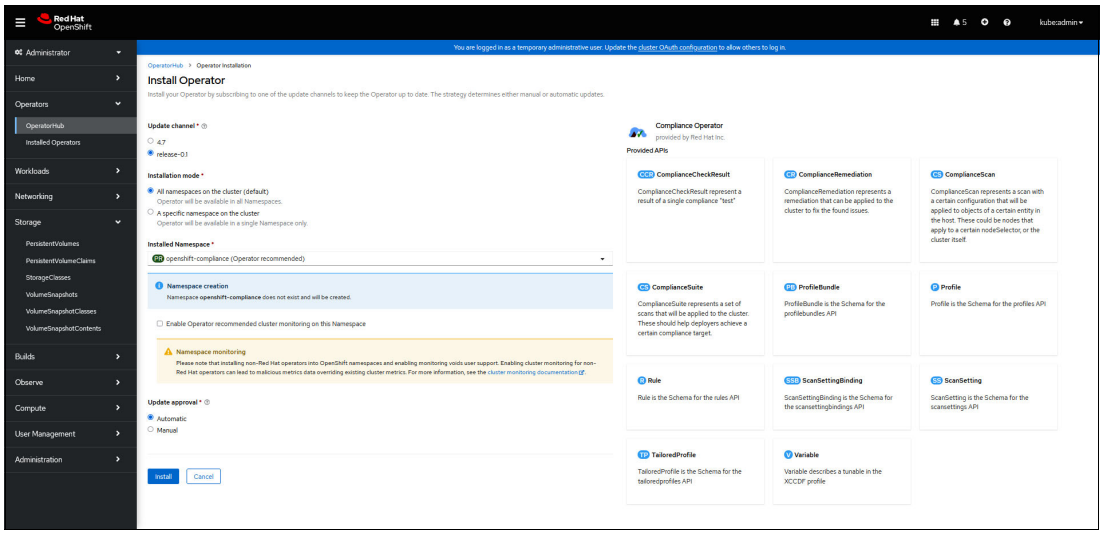


Figure 8-7 Installing Red Hat OpenShift Compliance Operator

5. To confirm that the installation is successful, complete the following steps:
   a. Select **Operators** → **Installed Operators**.
   b. Check that the Red Hat OpenShift Compliance Operator is installed in the `openshift-compliance` namespace and that its status is `Succeeded`, as shown in Figure 8-8.

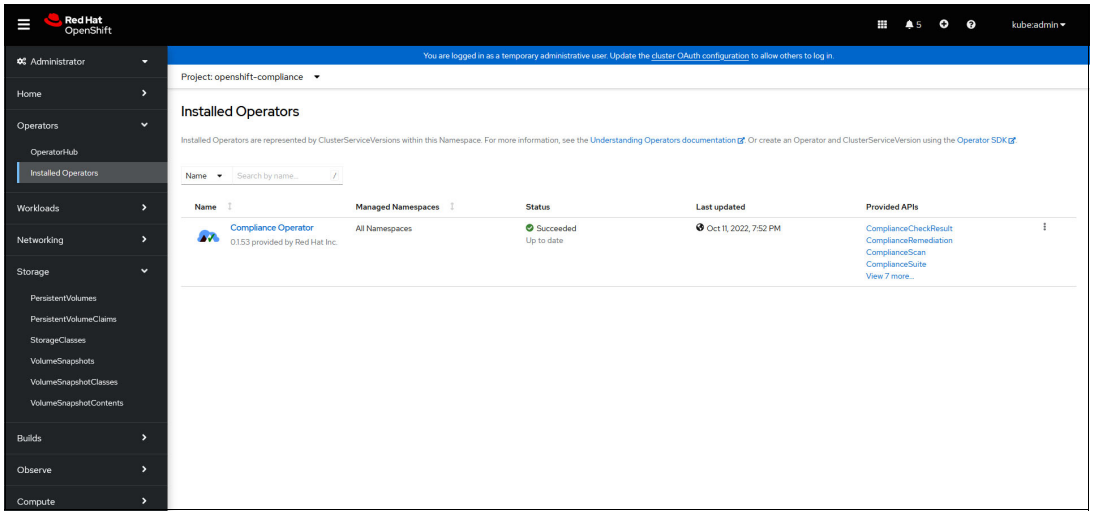


Figure 8-8 Verifying the Red Hat OpenShift Compliance Operator installation

Running Red Hat OpenShift Compliance Operator scans

After installation, Red Hat OpenShift Compliance Operator creates default ScanSetting objects with default settings for your convenience. You can run a compliance scan by using the default CIS profiles, as shown in Figure 8-9 on page 155.

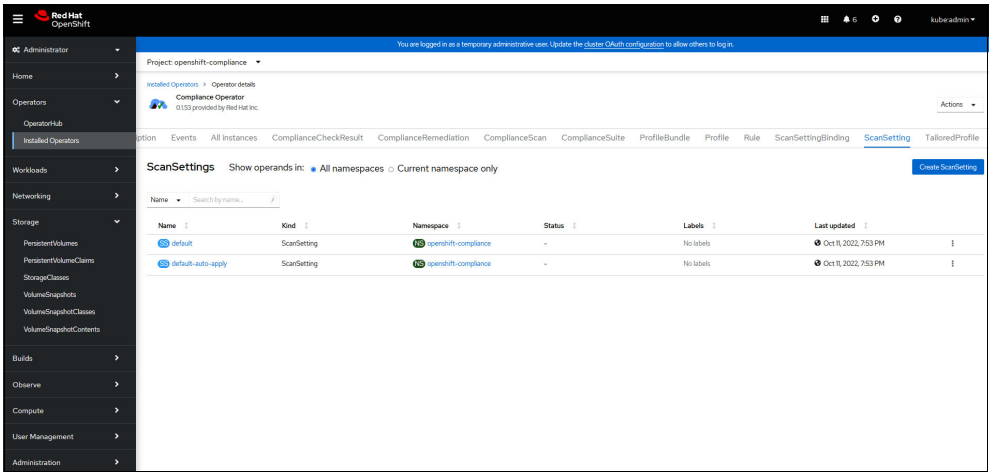


Figure 8-9 Verifying the ScanSetting objects

Complete the following steps:

1. To start the scan, create a ScanSettingBinding object that binds to the default ScanSetting object and cis and cis-node profiles, as shown in Figure 8-10.

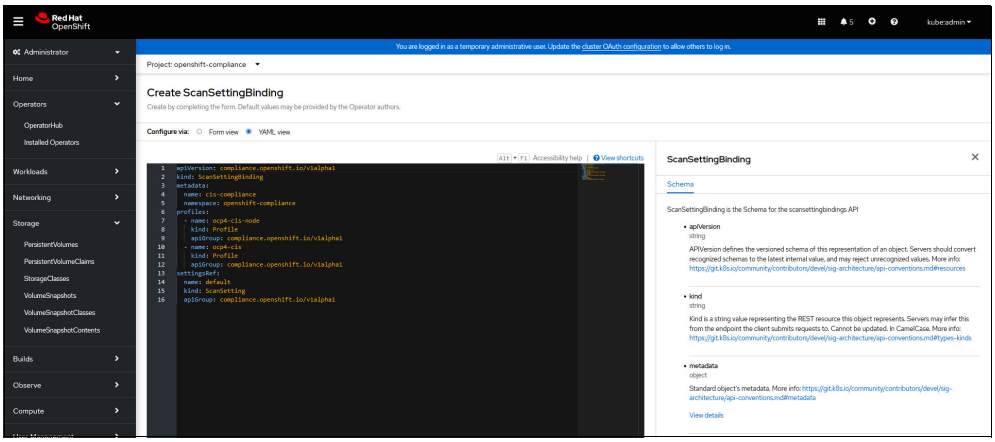


Figure 8-10 Creating a scan

2. After creating the ScanSettingBinding object, check that ScanSettingBinding object status is `Condition:Ready`, as shown in Figure 8-11.

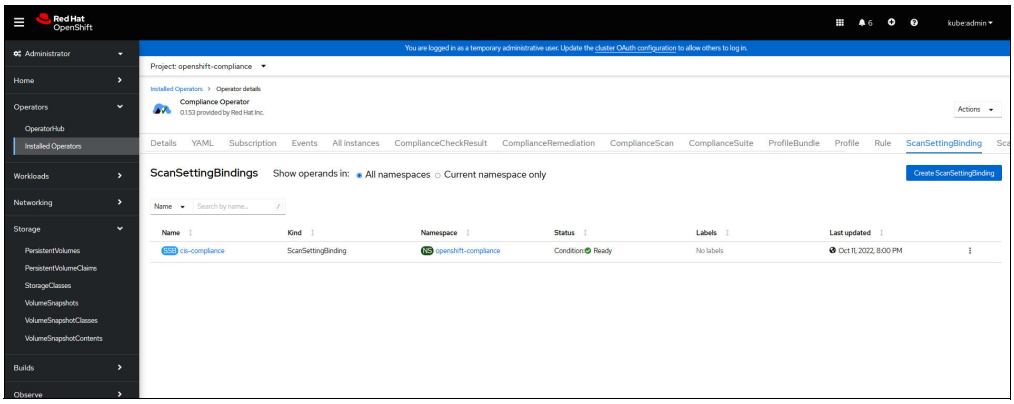


Figure 8-11 Checking the ScanSettingBinding object status

3. When the ScanSettingBinding object is in the Ready state, it automatically generates the ComplianceSuite object with the same name as the ScanSettingBinding object, as shown in Figure 8-12.

.

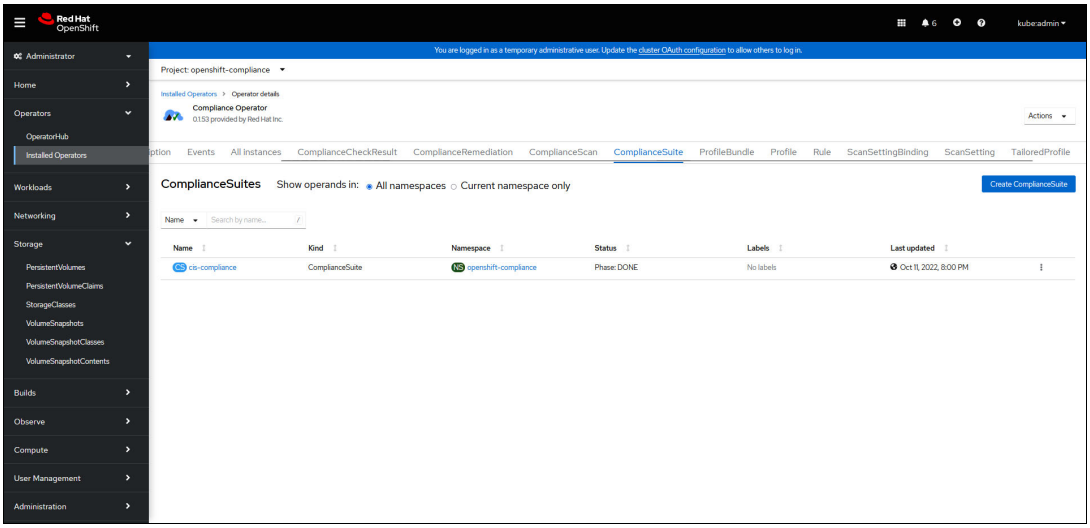


Figure 8-12 Verifying the ComplianceSuite object creation

4. The ComplianceSuite object automatically generates the ComplianceScans objects, which are based on the Compliance Suite definition, as shown in Figure 8-13.

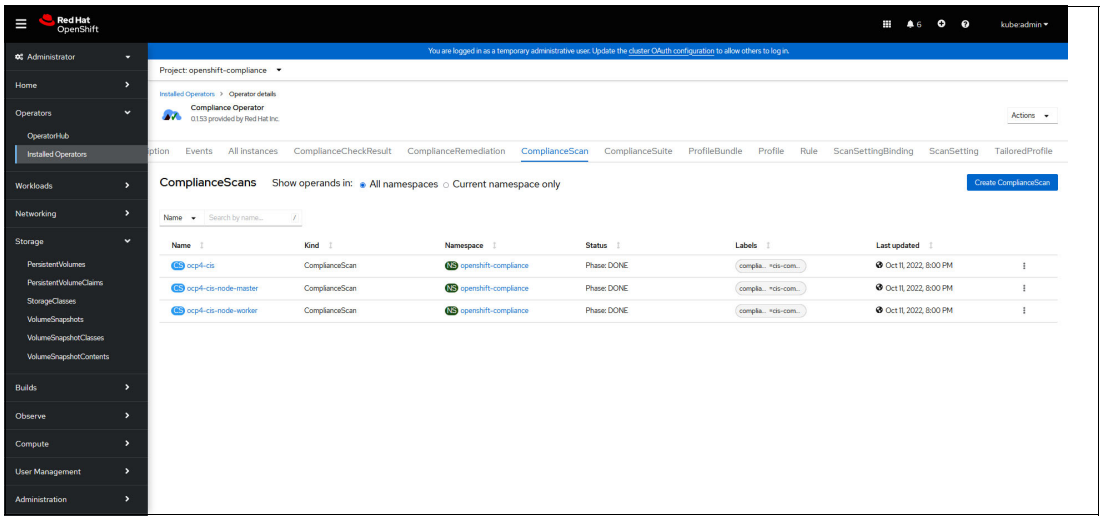


Figure 8-13 Verifying the ComplianceScans objects creation

5. After a successful compliance scan run, Red Hat OpenShift Compliance Operator generates the following object types:

   – ComplianceCheckResult

     Represents the state of the scan results against each Rule object in the scan profile, as shown in Figure 8-14 on page 157.

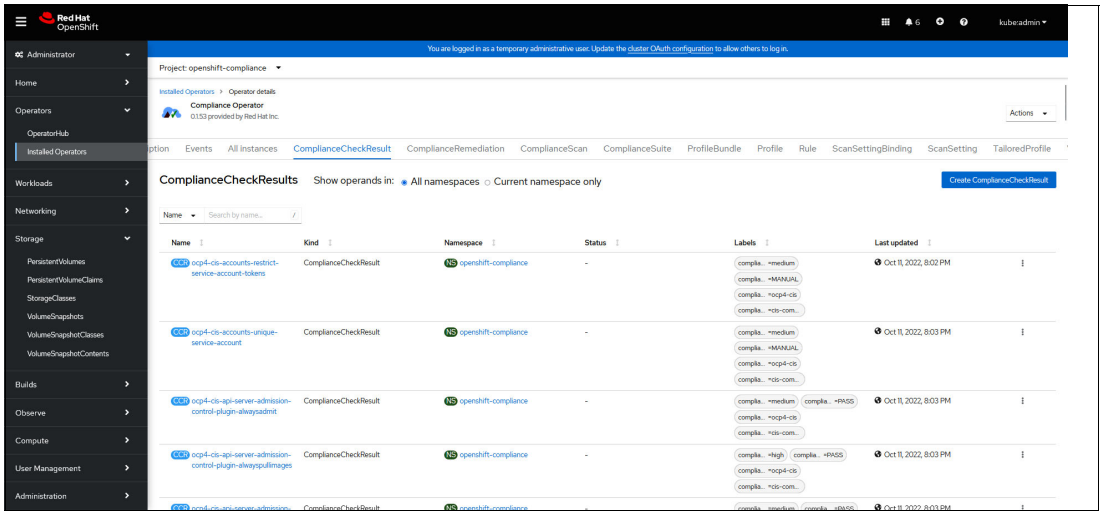


Figure 8-14 Checking the scan results state

– ComplianceRemediation

These objects show how to apply the fix for the rule. The apply value shows whether a fix should be applied, and the object value indicates what is expected to be applied on the node or cluster, as shown in Figure 8-15.

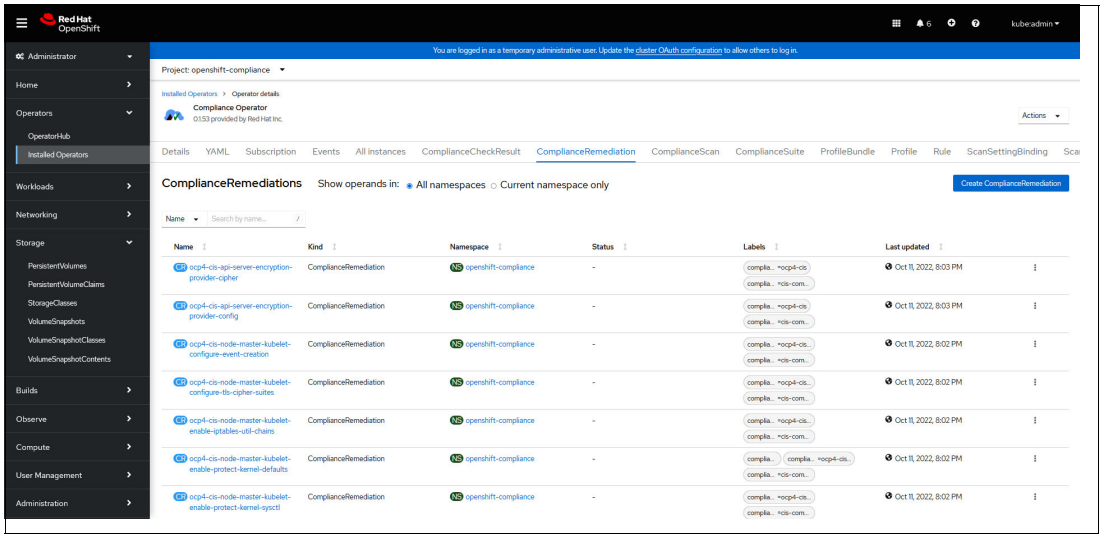


Figure 8-15 ComplianceRemediation page

Reviewing ComplianceRemediation

When you create a custom MachineConfigPool, add a label to MachineConfigPool so that the machineConfigPoolSelector that is present in KubeletConfig can match the label with MachineConfigPool.

8.5 Red Hat OpenShift Machine Config Operator

Red Hat OpenShift Machine Config Operator is a Kubernetes (K8s) operator that provides an automated way to manage the configuration of the operating system on nodes in a cluster. You use it to create and manage MachineConfig objects, which specify the wanted state of the operating system configuration on a node.

You can use Red Hat OpenShift Machine Config Operator for the following tasks:

- Updating the kernel or other system packages on nodes in the cluster
- Changing system-level configuration options
- Applying patches or security updates to the operating system
- Managing the configuration of cloud-init or other initialization systems

8.5.1 Applying remediation when using customized machine config pools

To accomplish this task, you need the following information:

- The failed compliance results, which are shown in Example 8-17.

Example 8-17 Compliance results

```
root@api.powercsi.ibm.com ~]# oc get ccr -lcompliance.openshift.io/check-status=FAIL
NAME                                                                STATUS   SEVERITY
ocp4-cis-api-server-encryption-provider-cipher                      FAIL     medium
ocp4-cis-api-server-encryption-provider-config                      FAIL     medium
ocp4-cis-audit-log-forwarding-enabled                               FAIL     medium
ocp4-cis-configure-network-policies-namespaces                      FAIL     high
ocp4-cis-idp-is-configured                                          FAIL     medium
ocp4-cis-kubeadmin-removed                                          FAIL     medium
ocp4-cis-node-master-kubelet-configure-event-creation               FAIL     medium
ocp4-cis-node-master-kubelet-configure-tls-cipher-suites            FAIL     medium
ocp4-cis-node-master-kubelet-enable-iptables-util-chains            FAIL     medium
ocp4-cis-node-master-kubelet-enable-protect-kernel-defaults         FAIL     medium
ocp4-cis-node-master-kubelet-enable-protect-kernel-sysctl           FAIL     medium
ocp4-cis-node-master-kubelet-enable-streaming-connections           FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-hard-imagefs-available    FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-hard-imagefs-inodesfree   FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-hard-memory-available     FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-hard-nodefs-available     FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-hard-nodefs-inodesfree    FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-soft-imagefs-available    FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-soft-imagefs-inodesfree   FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-soft-memory-available     FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-soft-nodefs-available     FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-soft-nodefs-inodesfree    FAIL     medium
ocp4-cis-node-worker-kubelet-configure-event-creation               FAIL     medium
ocp4-cis-node-worker-kubelet-configure-tls-cipher-suites            FAIL     medium
ocp4-cis-node-worker-kubelet-enable-iptables-util-chains            FAIL     medium
ocp4-cis-node-worker-kubelet-enable-protect-kernel-defaults         FAIL     medium
ocp4-cis-node-worker-kubelet-enable-protect-kernel-sysctl           FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-hard-imagefs-available    FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-hard-imagefs-inodesfree   FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-hard-memory-available     FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-hard-nodefs-available     FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-hard-nodefs-inodesfree    FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-soft-imagefs-available    FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-soft-imagefs-inodesfree   FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-soft-memory-available     FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-soft-nodefs-available     FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-soft-nodefs-inodesfree    FAIL     medium
ocp4-cis-scc-limit-container-allowed-capabilities                   FAIL     medium
```

- The current kubeletconfigs machineconfig objects, which are shown in Example 8-18.

Example 8-18 The kubeletconfigs machineconfig objects

```
[root@api.powercsi5.cp.fyre.ibm.com ~]# oc get kubeletconfigs.machineconfiguration.openshift.io
NAME                                          AGE
01-master-ibm-spectrum-scale-increase-pid-limit   24h
01-worker-ibm-spectrum-scale-increase-pid-limit   14h
```

- Example 8-19 shows a kubeletconfig machineconfig object that helps to fix some of the compliance failures.

Example 8-19 Example of kubeletconfigs machineconfig

```
[root@api.powercsi5.cp.fyre.ibm.com ~]# cat complaince.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: KubeletConfig
metadata:
  name: cis-hardening-workerpool
spec:
  machineConfigPoolSelector:
    matchLabels:
      pools.operator.machineconfiguration.openshift.io/worker: ""
  kubeletConfig:
    eventRecordQPS: 5
    tlsCipherSuites:
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    protectKernelDefaults: false
    evictionSoftGracePeriod:
      memory.available:  "5m"
      nodefs.available:  "5m"
      nodefs.inodesFree: "5m"
      imagefs.available: "5m"
    evictionHard:
      memory.available:  "100Mi"
      nodefs.available:  "10%"
      nodefs.inodesFree: "5%"
      imagefs.available: "15%"
    evictionSoft:
      memory.available:  "100Mi"
      nodefs.available:  "10%"
      nodefs.inodesFree: "5%"
      imagefs.available: "15%"
[root@api.powercsi5.cp.fyre.ibm.com ~]# cat complaincemaster.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: KubeletConfig
metadata:
  name: cis-hardening-masterpool
spec:
  machineConfigPoolSelector:
    matchLabels:
      pools.operator.machineconfiguration.openshift.io/master: ""
  kubeletConfig:
    eventRecordQPS: 5
    tlsCipherSuites:
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    protectKernelDefaults: false
    evictionSoftGracePeriod:
      memory.available:  "5m"
```

```
    nodefs.available:  "5m"
    nodefs.inodesFree: "5m"
    imagefs.available: "5m"
  evictionHard:
    memory.available:  "100Mi"
    nodefs.available:  "10%"
    nodefs.inodesFree: "5%"
    imagefs.available: "15%"
  evictionSoft:
    memory.available:  "100Mi"
    nodefs.available:  "10%"
    nodefs.inodesFree: "5%"
    imagefs.available: "15%"
```

Apply the above kubelet machineconfigs in Example 8-19 on page 159 and perform a successful compliance rerun. You can see that the results of the previous failed compliance rule show as passed (they are not listed in Example 8-20).

Example 8-20 Passed results

```
[root@api.powercsi5.cp.fyre.ibm.com ~]# oc get ccr-l
compliance.openshift.io/check-status=FAIL
NAME                                                                 STATUS   SEVERITY
ocp4-cis-api-server-encryption-provider-cipher                       FAIL     medium
ocp4-cis-api-server-encryption-provider-config                       FAIL     medium
ocp4-cis-audit-log-forwarding-enabled                                FAIL     medium
ocp4-cis-configure-network-policies-namespaces                       FAIL     high
ocp4-cis-idp-is-configured                                           FAIL     medium
ocp4-cis-kubeadmin-removed                                           FAIL     medium
ocp4-cis-node-master-kubelet-enable-iptables-util-chains             FAIL     medium
ocp4-cis-node-master-kubelet-enable-protect-kernel-defaults          FAIL     medium
ocp4-cis-node-master-kubelet-enable-protect-kernel-sysctl            FAIL     medium
ocp4-cis-node-master-kubelet-enable-streaming-connections            FAIL     medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-hard-imagefs-inodesfree FAIL medium
ocp4-cis-node-master-kubelet-eviction-thresholds-set-soft-imagefs-inodesfree FAIL medium
ocp4-cis-node-worker-kubelet-enable-iptables-util-chains             FAIL     medium
ocp4-cis-node-worker-kubelet-enable-protect-kernel-defaults          FAIL     medium
ocp4-cis-node-worker-kubelet-enable-protect-kernel-sysctl            FAIL     medium
ocp4-cis-node-worker-kubelet-enable-streaming-connections            FAIL     medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-hard-imagefs-inodesfree FAIL medium
ocp4-cis-node-worker-kubelet-eviction-thresholds-set-soft-imagefs-inodesfree FAIL medium
ocp4-cis-scc-limit-container-allowed-capabilities                    FAIL     medium
```

8.6 IBM Hyper Protect Crypto Services

For public key cryptography, X509 certificates have been the standard since it was first issued in 1988.[5] With the advent of the internet, the popularity of x509 certificates increased exponentially. The certificates are used for securing websites, Internet of Things objects, software, and container orchestrators like K8s and Red Hat OpenShift.

Speaking from experience with our customers, we see on an average that each enterprise deals with hundreds if not thousands of certificates. For a security or system administration staff, rotating them and keeping them up to date is a significant challenge.

To help with this problem, IBM created the IBM Hyper Protect Crypto Services[6] (IBM HPCS) offering to help enterprises manage their keys in a multicloud environment.

[5] https://en.wikipedia.org/wiki/X.509#History_and_usage

8.6.1 Universal Key Orchestrator

There might be billions of certificates in use. Even at the enterprise level, there are hundreds or even thousands of keys to maintain. The certificates are helpful in securing communication channels, but they require maintenance. Generally, certificates expire every 2 - 3 years, the old certificates and their associated keys are rotated, and new certificates and keys are issued.

The Universal Key Orchestrator (UKO) provides a central control plane for managing certificates regardless of their location. It is a software as a service (SaaS) application that, when configured, can store the certificates securely, rotate them at a frequency of the customer's choice, and integrate them into all major cloud providers, such IBM Cloud, AWS, GCP, and Microsoft Azure.

UKO is built on top of IBM HPCS. IBM HPCS is the only single tenant keep-your-own-key (KYOK) encryption technology that uses an FIPS 140-2 Level 4 hardware Hardware Security Module (HSM) device that is single tenant. With KYOK, a customer manages the key vaults with their own key that is accessible only by the customer. IBM HPCS is the only product that is NIST certified for FIPS 140-2 Level 4. For more information, see Announcing Multicloud Key Management with IBM Cloud Hyper Protect Crypto Services.

8.6.2 IBM HPCS with Unified Key Orchestrator

IBM HPCS is a single-tenant, regional service that supports complete tenant-based workload isolation. It is formed by a dedicated key management service that is based on HSM on IBM Cloud. IBM HPCS is used to perform cryptographic operations and orchestrate all keys from a multi-cloud environment. Unified Key Orchestrator provides the only cloud-native, single point of control of encryption keys across hybrid multi-cloud environments, including on-premises environments.

Hardware Security Module

An HSM provides secure key storage and cryptographic operations within a tamper-resistant hardware device for sensitive data. It uses all key material without exposing it outside the cryptographic boundary of the hardware. Many IBM Cloud services support data encryption by using customer-managed keys, also known as bring-your-own-key (BYOK). An example of BYOK is using IBM Key Protect to deliver encryption keys from internal solutions to environments on the cloud. IBM Key Protect is a multi-tenant solution that uses an FIPS 140-2 Level 3 HSM that you can integrate with multiple services running on IBM Cloud, such as database, storage, container, and computing services.

IBM HPCS features KYOK encryption capabilities that ensure full control of the entire key hierarchy where no IBM Cloud administrators have access to your keys.

8.6.3 Use cases and scenarios

Here are some illustrative use cases and scenarios:

- Pervasively protecting data at rest in the cloud
- Using Universal Key Orchestrator for multicloud key orchestration
- Using IBM HPCS for Public Key Cryptography Standards #11 HSMs

[6] https://www.ibm.com/cloud/hyper-protect-crypto

Pervasively protecting data at rest in the cloud

One of the most common problems we must deal with is encrypting cloud data at the highest security level with your own keys. IBM HPCS uses the same key-provider application programming interface (API) as IBM Key Protect to provide a consistent envelope encryption and file system encryption approach to adopting IBM Cloud services. Encryption keys that are generated by IBM HPCS can be used to provide application record-level or field-level encryption to avoid insider threats such as database administrator access. You can benefit from the cryptographic capabilities of IBM HPCS built on top of a FIPS 140-2 Level 4 Certified HSM for both your new and existing workloads. Figure 8-16 shows data at rest encryption.

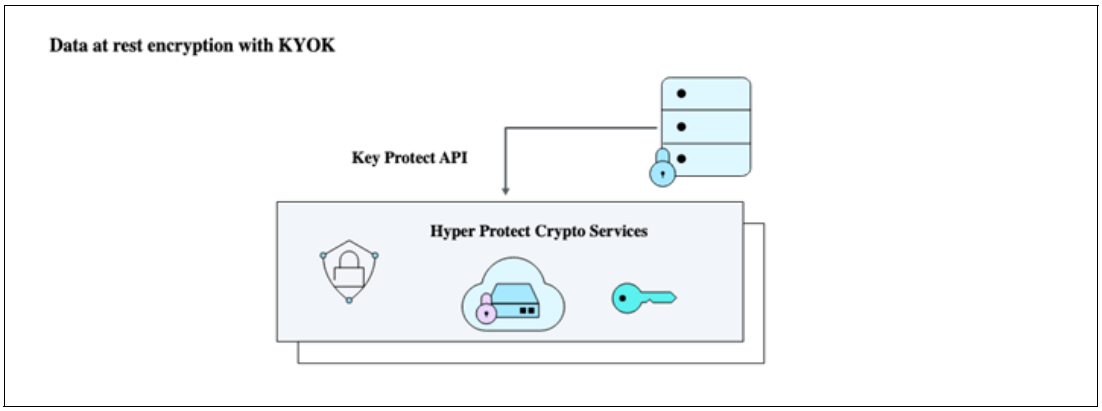


Figure 8-16 Data at rest encryption with KYOK

Organizations that use VMware Solutions on IBM Cloud to process and store personal data require the highest level of security. With the Key Management Interoperability Protocol for VMware component, the VMware environment can store and use keys from IBM HPCS. IBM HPCS extends the family of key management services in the IBM Cloud toward single-tenant instances with dedicated hardware secret control, as shown in Figure 8-17.

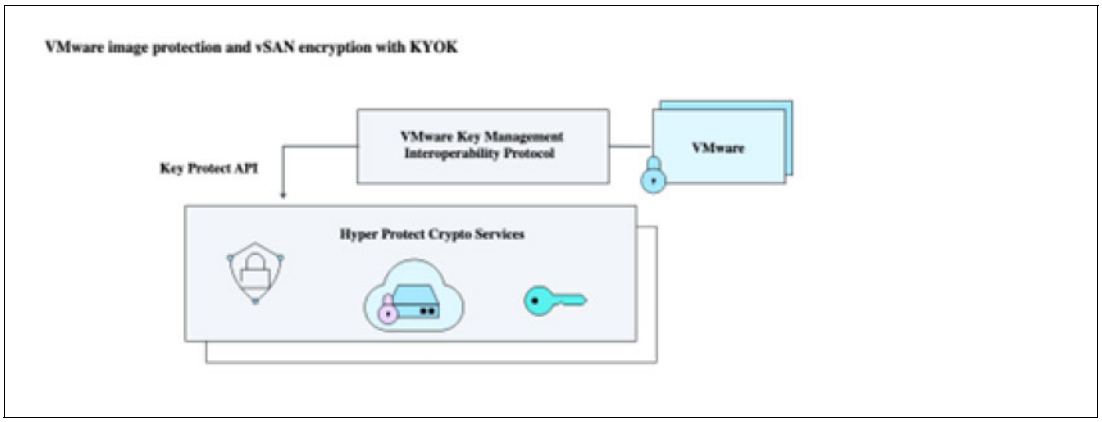


Figure 8-17 VMware image protection

Using Universal Key Orchestrator for multicloud key orchestration

UKO can be used to securely create and manage keys and internal keystores across multiple environments, such as different cloud providers. Figure 8-18 shows how UKO can be used.

Doing this integration brings multiple functions, such as centralized key management, which use Identity and Access Management (IAM) to provide control access to the vault, granting access to keys and keystore that are assigned to the vault. Also, user interfaces can be used to create, manage, and delete cryptographic keys so that key lifecycle management can be fully audited. This approach avoids reinstalling keys if you deploy services on a different environment is needed because all keys are backed up and easy to recover if a unrecoverable error occurs.

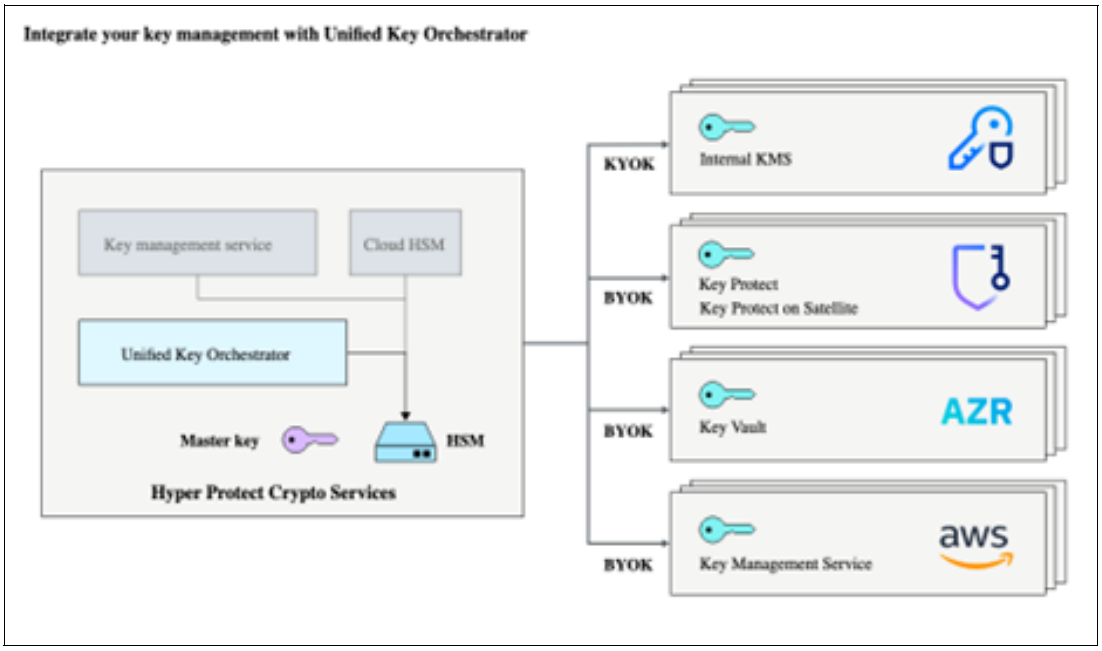


Figure 8-18 Key management

Using IBM HPCS for Public Key Cryptography Standards #11 HSMs

IBM HPCS provides the Public Key Cryptography Standards (PKCS) #11 API that is defined as one of the PKCS. All cryptographic operations are run in an HSM in the cloud. Application programmers can design and develop applications with a standard PKCS #11 API to request encryption or sign the application data without programmers needing to become encryption experts. Applications can use the IBM HPCS PKCS #11 library to modernize business process and use a digital workflow with private data and digital reviews, approvals, and signatures that are secure and trustworthy.

The IBM HPCS PKCS #11 library can be used to encrypt data between clouds with a wide list of cryptographic operations: signing, signature validation, message authentication codes, and more advanced encryption schemes, as shown in Figure 8-19.

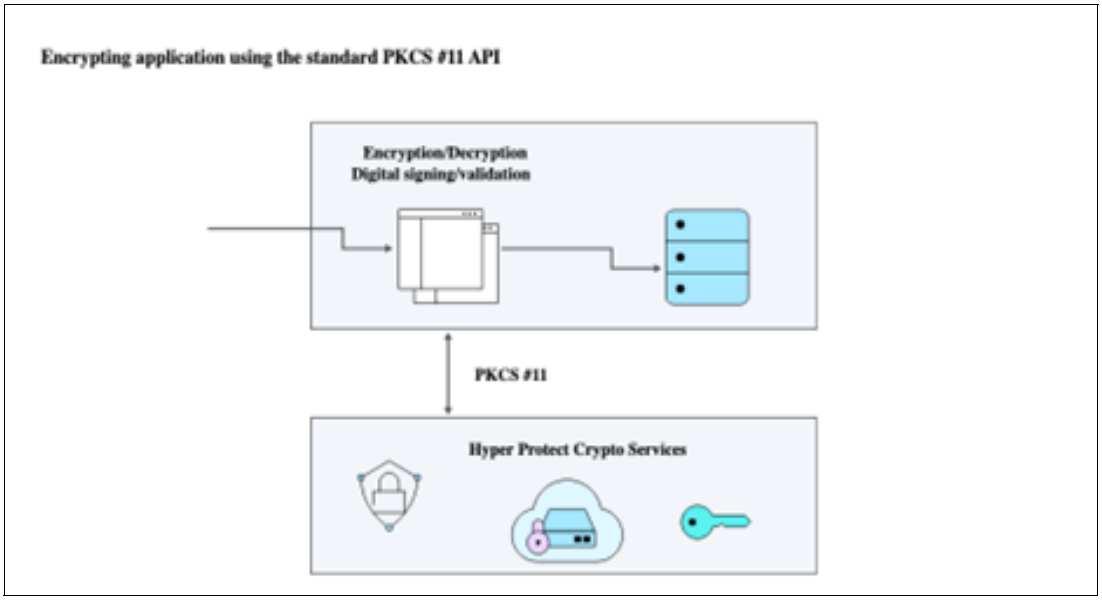


Figure 8-19 Encryption applications

The library also can be used for database encryption, for example, with IBM HPCS, an Oracle or Db2 database can be encrypted by using Transparent Data Encryption (TDE). Using TDE, sensitive data on database storage media can be encrypted like table spaces and files. The database system automatically and transparently encrypts and decrypts data when it is used by authorized users and applications.

Database users and applications do not need to be aware of implementing or adapting TDE. TDE uses a two-tiered key hierarchy that is composed of both a TDE data encryption key to encrypt data and a TDE master encryption key for encrypting and decrypting a data encryption key, as shown in Figure 8-20.

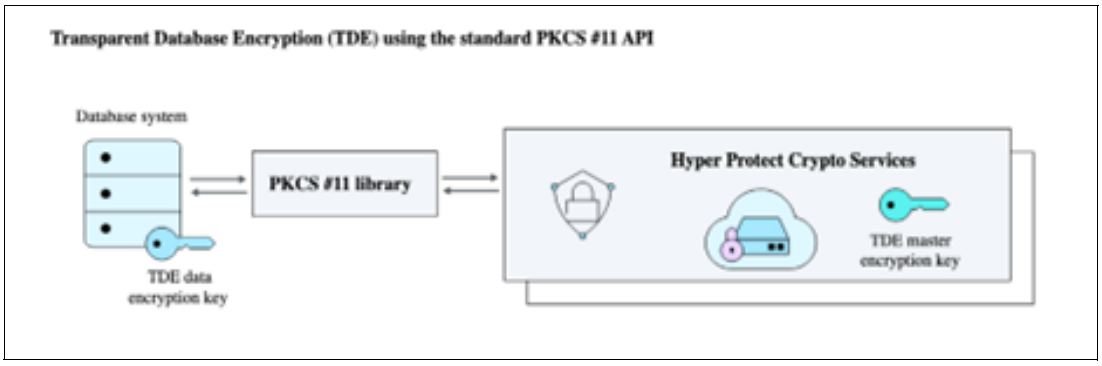


Figure 8-20 Database encryption

In the context of web servers, with Transport Layer Security (TLS) and Secure Sockets Layer (SSL), a website establishes its identity. IBM HPCS provides a way to offload cryptographic operations that are done during a TLS handshake to establish a secure connection to the web server while it keeps the private key securely stored in the dedicated HSM. Offloading to IBM HPCS enables data in transit protection for web, API, and mobile transactions by using the standard PKCS #11 API.

Figure 8-21 shows how TLS and SSL can be managed.

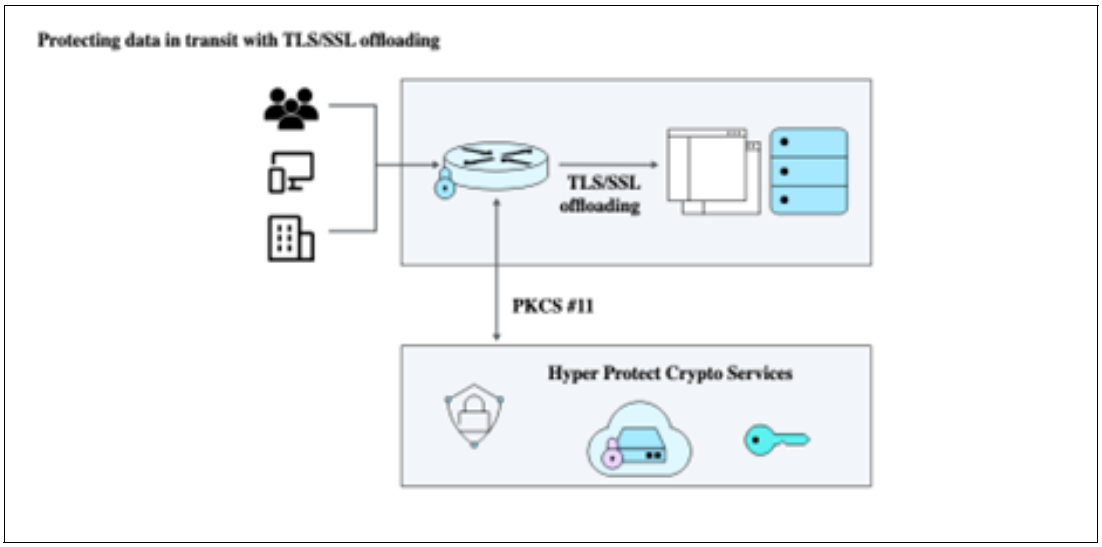


Figure 8-21 TLS/SSL management

9

Security Site Reliability Engineer

Enterprise customers have been adopting a DevOps culture over the past decade. The Site Reliability Engineer (SRE) function employs a data-driven approach to balance new feature development and customer experiences. In a fast-paced, cloud-native development environment with automated continuous integration and continuous delivery (CI/CD) processes, SREs play a key role in determining whether to deploy a new feature to production or halt deployment to ensure that the existing software in production continues to perform well according to service contracts and obligations.

This chapter describes the following topics:

- Introducing the Site Reliability Engineer
- Security scoring
- Service levels to apply to security
- Security runbooks

9.1 Introducing the Site Reliability Engineer

Over the past few years, there has been a surge of data breaches, security lapses, and problems in the security of the software supply chain. There is now a renewed focus on DevSecOps, which integrates security controls and best practices into the DevOps workflow. DevSecOps means building security into application development from end to end. This integration into the pipeline requires a new organizational mindset and new tools. DevOps teams should automate security to protect the overall environment and data, and the continuous integration/continuous delivery process, which is a goal that includes the security of microservices in containers.

One of the new roles in DevSecOps is the SRE. The SRE creates a bridge between development and IT operations by taking on the tasks that are done typically by operations. Instead, such tasks are given to SREs, who use automation tools to solve problems by creating scalable and reliable software systems. In this paper, we present examples of container security, code scanning and signing tools, and practices to enable the SRE function and ensure proper security in the containerized environment. The rest of this chapter revisits some fundamental concepts because they can be applied to security.

In the mainstream SRE practice, the operations teams working together with development should teams define the service-level indicators (SLIs), service-level objectives (SLOs), service-level agreements (SLAs), and Error Budgets that are used to manage their application environment. We can organize the same concepts as a security theme for use in your SRE practice.

For more information about the basics of SRE, see What is site reliability engineering (SRE)?

For more information about how SREs can benefit from Artificial Intelligence for IT Operations (AIOps), see An SRE journey to AIOps.

9.2 Security scoring

A service's security footprint is the sum of its parts that are spread across software code; its dependencies (open-source or commercial libraries); and its testing methods, deployment methods, and runtime environment. This section describes many of those components.

9.2.1 Security scoring example

Red Hat OpenShift Compliance Operator on Red Hat OpenShift runs scans and produces compliance reports. You can obtain the reports by running a pod to mount the volumes and extract the reports. Figure 9-1 shows an example of a security scoring output.

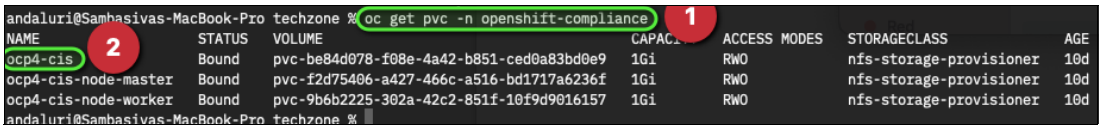


Figure 9-1 Security scoring output

These results were created by using the YAML file that is shown in Example 9-1.

Example 9-1 Running security scoring reports

```
cat <<EOF | kubectl apply -f -
apiVersion: "v1"
kind: Pod
metadata:
  name: pv-extract
spec:
  containers:
    - name: pv-extract-pod
      image: registry.access.redhat.com/ubi8/ubi
      command: ["sleep", "3000"]
      volumeMounts:
        - mountPath: "/workers-scan-results"
          name: workers-scan-vol
  volumes:
    - name: workers-scan-vol
      persistentVolumeClaim:
        claimName: ocp4-cis
EOF
```

To gather these results, complete the following tasks:

1. Find the persistent volume claims (PVCs) that are created in the `openshift-compliance` project (`openshift-compliance` is the default project name that Red Hat OpenShift Compliance Operator uses). In this example, we use `ocp4-cis` as an example PVC that we want to mount into a dummy pod that is named `pv-extract`.
2. When the pod runs, you can copy and extract the files by using the command that is shown in Example 9-2.

Example 9-2 Finding persistent volume claims

```
oc -n openshift-compliance cp pv-extract:workers-scan-results .
bunzip2 <arf output files>
```

The files contain reports in Assessment Results Format (ARF),[1] which is an open-source specification that is based on XML that is compressed in a `bzip2` format. This data can be consumed by other tools that can render the report in a user-readable format, such as HTML. One example of such a tool is `oscap-report`.

Example 9-3 shows an example about how to install the tool and convert the reports to HTML.

Example 9-3 Installing the OSCAP report

```
pip install openscap-report
oscap-report < ocp4-cis-api-checks-pod.xml.bzip2.out > ocp4-report.html
```

[1] For more information about ARF, see ARF.

Figure 9-2 shows an example HTML report that shows the results of the Center for Internet Security (CIS) rules that scanned a demonstration cluster. The results provide information about whether the scan passed or failed, reports on the severity of any issues, and provides the compliance score.

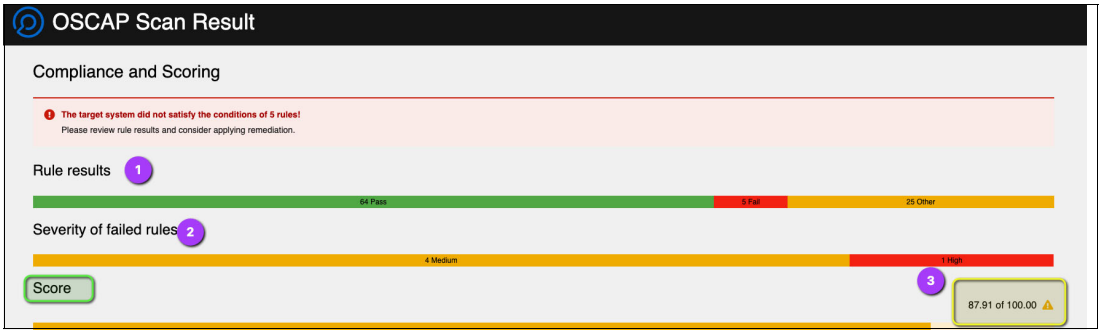


Figure 9-2 OSCAP scan result

In one of the failed rule scans, as shown in Figure 9-3, you can see the rule `reference-id`, the related references in CIS, and other specifications along with a clear rationale for the rule and its importance.

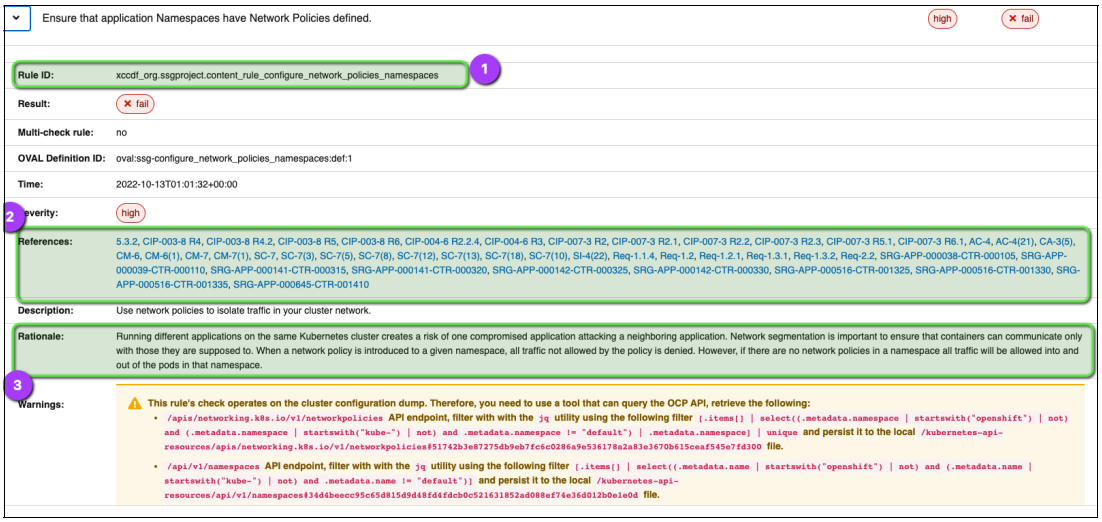


Figure 9-3 Details for a failed scan

9.2.2 Security scoring in IBM Cloud Security and Compliance Center

At the time of writing, IBM Cloud Security and Compliance Center offers a collector tool that can aggregate ARF reports from on-premises or IBM Cloud resources into a single dashboard. Tanium, an IBM Business Partner, offers tools for collecting the data from AWS, Google Cloud Platform (GCP), and Azure, and showing those results in a dashboard in IBM Cloud Security and Compliance Center.

The Open Source Security Foundation (OpenSSF) provides thought leadership and projects for increasing security for open-source projects. One of their projects, Security Score Cards, provides tools and integration (for example, GitHub) to scan the code and publish a score card for open-source projects. OpenSSF publishes a list of 864+ open-source projects that have at least a passing score (for more information, see BadgeApp). Commonly used tools such as curl have a Gold score, and Node.js have a passing score. This list helps developers to choose appropriate projects in the open-source ecosystem to improve their security posture.

For the scope of this paper, we explore how you can use the metrics from the ARF reports and incorporate those metrics into SRE practice. For a specific implementation, see Chapter 10, “Aqua” on page 173.

9.3 Service levels to apply to security

There are many concepts in service-level management that can be applied directly to security.

Security focused SLIs

A SLI is a quantitative measurement of a service level, such as latency or availability. For a security focused SLI, we might measure the number of vulnerabilities in a container, or measure a compliance score of a cluster or node in an Red Hat OpenShift environment.

Security focused SLOs

A SLO is a target for an SLI. For example, a service that serves http requests, *95% percentile latency less than 5 ms*, as the target or objective that the SRE wants to achieve. For a security focused SLO, this target might be the percentage of high-risk vulnerabilities that must be less than 5% for the service to be deemed secure. This target might vary based on the industry vertical. Customers in regulated industries such as financial services or health care might have more stringent requirements for compliance or security targets.

Security focused SLA

SLAs are guarantees for a certain level of service beyond which the service provider provides a discount or payout when SLAs are breached. If there is a security breach that results in data loss, it might cost a business $4.35 million dollars as of 2022.[2] Hence, enterprises have a rationale to institute internal SLAs for security issues such that risk mitigation measures might be deployed to set the security within the limits that are set by the security SLA standards.

[2] https://newsroom.ibm.com/2022-07-27-IBM-Report-Consumers-Pay-the-Price-as-Data-Breach-Costs-Reach-All-Time-High

9.4 Security runbooks

Runbooks are a tool that helps the operations team define a set of repeatable instructions that are run to solve an issue when it occurs. These instructions originate from the corrective actions that were deployed previously. Security runbooks address a set of pretested steps that can be followed even by an entry level engineer.

However, runbooks are not an answer if an issue repeats in the same way multiple times in each period, which might be due to a bug or an architectural or design flaw.

Although runbooks themselves are an excellent tool, as a best practice, automate the runbook execution when an event occurs. By automating the runbook, you get a faster and more consistent approach to security issues when they are discovered.

For more information about a runbook automation architecture pattern that implements this principle, see Runbooks - IBM Cloud Architecture Center.

10

Aqua

This chapter provides an overview of Aqua, which is a tool for securing your workload that is running on Red Hat OpenShift on IBM Power. *Aqua* is a security tool from an IBM Business Partner that runs on IBM Power servers and provides security from development to a full container run time.

This chapter describes the following topics:

- Cloud-Native Application Protection Platform
- Aqua for cloud-native application protection
- Container security lifecycle and risk areas
- Container security lifecycle
- The Cloud-Native Application Protection Platform
- Aqua support for Red Hat OpenShift on IBM Power

10.1 Cloud-Native Application Protection Platform

If you are embarking on a journey to a hybrid cloud and you have successfully tested Red Hat OpenShift on Power servers, you are thinking about how you are going to secure this new environment that is so different from the traditional paradigm. In the new paradigm, development and operations are combining to create a fast-paced software delivery platform (DevOps) that decreases time-to-market for new features.

To make DevOps work, new frameworks are used, and open-source packages are integrated into the code. Concerns about the security of the code itself, as a part of the inherent cloud-native environment, are forcing companies to change the way that they implement security. This approach contrasts with the traditional approach for applications, where the security for the software was provided by the software vendors through errata and security bulletins.

10.2 Aqua for cloud-native application protection

Cloud-native application protection requires shifting security from patching deployed applications to patching and fixing security issues before the software is packaged, and maintaining that security through deployment and run time.

There are three major parts of the platform where you ensure security:

1. Infrastructure
2. Cloud-native build
3. Workload run time

Aqua can help address all three parts with a unique solution that works for Red Hat OpenShift on Power servers, and other Kubernetes (K8s) clusters and different cloud providers.

10.3 Container security lifecycle and risk areas

Regarding cloud-native applications, you want options. Although lowering your total coast of operation (TCO) on an on-premises Power infrastructure makes sense for continuous workloads, the public cloud might make sense for workloads that are seasonal or for a new project fast start.

With this hybrid approach and possibly multiple cloud providers, you need to span across clouds and K8s platforms.

Aqua can help you in this process in the following ways:

1. Manage Kubernetes security.
2. Manage the cloud security posture.
3. Manage hybrid and multi-cloud environments.
4. Help demonstrate compliance.

Because you are focusing on Red Hat OpenShift on Power servers, start looking into the features that can help you keep the Red Hat OpenShift infrastructure secure by applying K8s security techniques.

Aqua Security delivers a secure container platform by providing a comprehensive set of security features for securing the entire container lifecycle from build to run time. These features include the following ones:

- Image scanning: Aqua Security can scan container images for vulnerabilities and malware, both before and after deployment. This task can be done both on-demand and as part of the continuous integration and continuous delivery (CI/CD) pipeline. Aqua uses both open-source scanning tools and its own proprietary engine to scan images for known vulnerabilities, malware, and other security issues.
- Runtime protection: Aqua Security can protect running containers from threats such as privilege escalation, network attacks, and malicious processes. This task can be done by using runtime-protection policies such as network security rules, access controls, and process-level isolation.
- Compliance and governance: Aqua Security can enforce compliance policies and provide detailed auditing and reporting to meet regulatory and compliance requirements.
- Centralized management: The Aqua Security platform can be managed centrally, which provides a unified view of security across multiple clusters and namespaces. This task can be done by using the Aqua web-based console, application programming interfaces (APIs), and integrations with external security incident and event managers (SIEMs).
- Secrets Management: Aqua Security can protect and manage secrets, credentials, and sensitive data in the container environment by providing secure storage, encryption, and access controls.

Aqua Security integrates with Red Hat OpenShift by installing an *Aqua Enforcer* container on each node in the cluster. This container communicates with the *Aqua Security Control Plane* to enforce security policies and provide visibility into the security status of the cluster.

Also, Aqua Security offers the Cloud-Native Application Protection Platform (CNAPP), which is built to work natively with Red Hat OpenShift and provide a security-focused, API-driven, and automation-friendly platform for automating and streamlining the application security process.

Aqua Security augments Red Hat OpenShift native security controls to ensure compliance and visibility over container workloads, as shown in Figure 10-1.

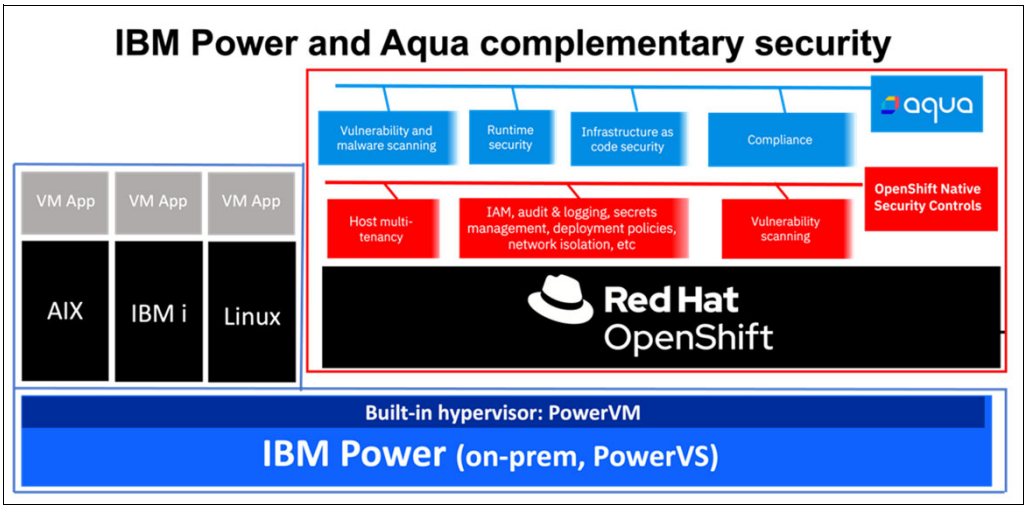


Figure 10-1 Augmented Red Hat OpenShift security with Aqua

10.4 Container security lifecycle

Security for containers must be built in initially and then maintained throughout the lifecycle of the container. Here are the main stages of the container security lifecycle:

- Development and Build: This stage is focused on ensuring that the container images that are used to run the application are secure and free of vulnerabilities or malware. This stage can include practices such as writing secure code, testing for vulnerabilities, and scanning images for known vulnerabilities. The container images are built from the code and prepared for deployment. This task can include practices such as verifying the authenticity of the base images that are used to build the container images, and ensuring that only trusted images are used.
- Deployment: This stage is focused on deploying the container images and ensuring that they are running securely in the production environment. This stage can include practices such as network segmentation, runtime protection, and access controls.
- Run: During this stage, the containerized applications are monitored, updated, and managed throughout their lifecycle. This stage can include practices such as vulnerability management, monitoring, and compliance.

Figure 10-2 illustrates the container security lifecycle and highlights major risk areas.

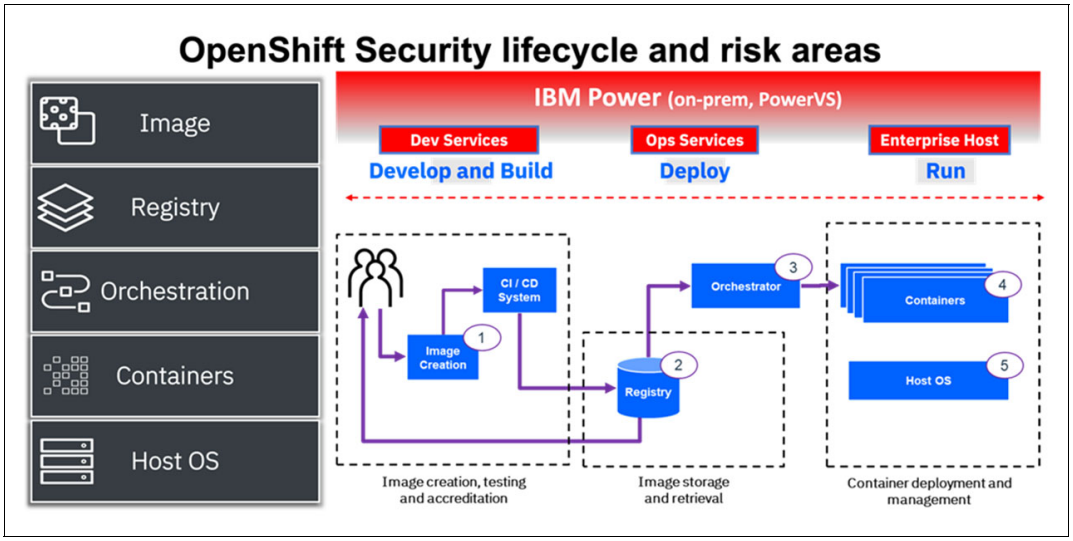


Figure 10-2 Container security lifecycle with risk areas

Risks from the container image, the registry, the orchestration process, the container itself, and from the host operating system introduce a new set of threat vectors, as shown in Table 10-1.

Table 10-1 Threat vectors in container environments

Container risk areas	Threat vectors
Images	► Image vulnerabilities ► Configuration defects ► Embedded malware ► Embedded clear text secrets ► Untrusted images
Registry	► Insecure connections to registries ► Stale images in registries ► Insufficient authentication and authorization restrictions
Orchestration	► Unbounded administrative access ► Unauthorized access ► Poorly separated inter-container network traffic ► Mixing of workload sensitivity levels and orchestrator node trust
Container	► Vulnerabilities within the runtime software ► Unbounded network access from containers ► Insecure container runtime configurations ► App vulnerabilities ► Rogue containers
Host operating system	► Large attack surface ► Shared kernel ► Host operating system component vulnerabilities ► Improper user access rights and host operating system file system tampering

10.5 The Cloud-Native Application Protection Platform

CNAPP is a security solution that provides comprehensive security for cloud-native applications and the infrastructure on which they run. CNAPP works natively with cloud-native technologies such as K8s, Red Hat OpenShift, and Docker, and provides security throughout the entire application lifecycle, from development to production.

CNAPP provides several features that help secure the containerized applications lifecycle:

- Image scanning and vulnerability management
- Network segmentation
- Runtime protection
- Compliance and governance
- Secrets management
- Centralized management and visibility
- Automation and integration with the DevOps pipeline

CNAPP also helps to secure the runtime environment of containers by implementing security policies on the orchestration platform to monitor the containers and alert on potential threats. Also, the solution can integrate with other tools that are used in the development pipeline, such as CI/CD tools, which enable security to be built in to the development process rather than being an afterthought.

CNAPP solutions can help organizations to address security concerns that are associated with the deployment and operation of cloud-native applications to reduce the attack surface, mitigate risks, and improve overall security posture, without impacting the speed or flexibility of the development process.

Aqua addresses the unique set of security risks that are introduced when container environments are created, whether by using containers to build applications from the ground up or porting existing monolithic apps.

Aqua secures the full container stack and lifecycle:

Trust code: As developers pull together source code and base images to build their containers, Aqua scanning tools check for misconfiguration, vulnerabilities, or malware to ensure the integrity of the CI/CD pipeline.

Harden infrastructure: Aqua checks the deployment environment to ensure that there is no misconfiguration and that the environment complies with any industry or territory regulations, and then categorizes any issues that are detected for resolution.

Protect workloads: When the application is deployed and running in the production environment, Aqua continues to monitor for malicious or unexpected activity.

Figure 10-3 shows how Aqua provides a comprehensive set of tools to support CNAPP so that you regain visibility and reduce the risk to your business.

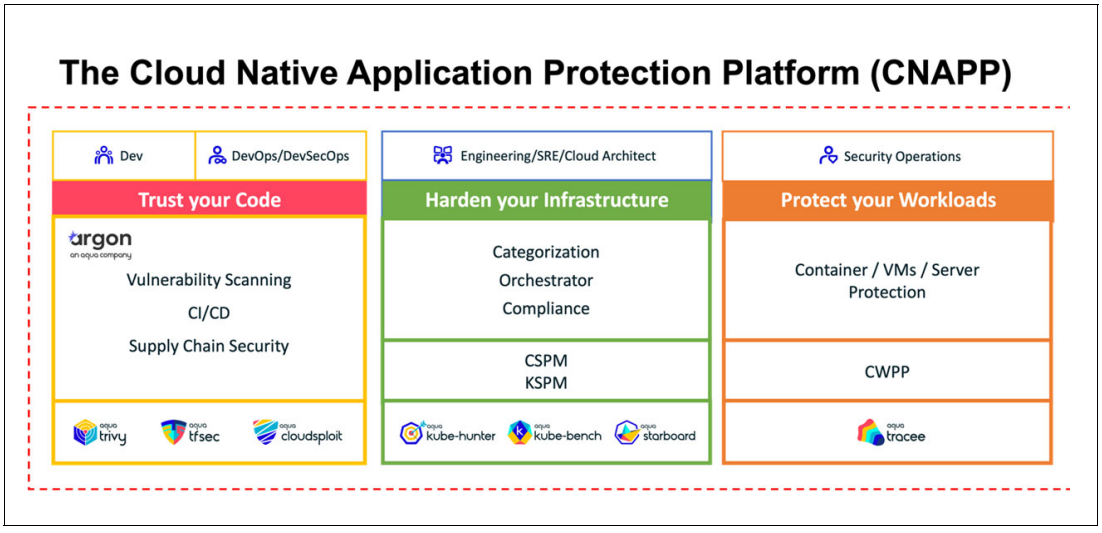


Figure 10-3 Cloud-Native Application Protection Platform

10.6 Aqua support for Red Hat OpenShift on IBM Power

The Aqua security platform is the recommended, integrated cloud-native security solution to run on the Red Hat OpenShift on IBM Power combined platform. It provides a consistent view of risk across each part of the lifecycle.

Here are Aqua supported features for Red Hat OpenShift on IBM Power:

- Scan for vulnerabilities: With this feature, you can scan images on Red Hat OpenShift on IBM Power with integration to a Red Hat OpenShift Container Platform registry.
- Aqua Trivy is a supported vulnerability scanner for container images that can scan for vulnerabilities in the operating system packages and libraries in a container image and also in the application itself. It can be integrated with a CI/CD pipeline to detect vulnerabilities early in the development process. For more information, see 10.6.2, “Scanning for vulnerabilities by using Aqua Trivy and Starboard” on page 181.
- Workload assurance through *Aqua Starboard* by using auto-discovery of Red Hat OpenShift resources and evaluating workloads for security risks. Aqua Starboard is a runtime security solution that provides real-time security visibility and protection for Red Hat OpenShift applications. It provides an agent that runs within the host and monitors all the containers running on the host, and it detects and mitigates security threats in real time. It also can identify malicious activities like privilege escalation and network attacks, and report them to a central management console. Aqua Starboard also integrates with *Aqua Trivy* to provide a comprehensive security solution for Red Hat OpenShift applications. For more information, see 10.6.2, “Scanning for vulnerabilities by using Aqua Trivy and Starboard” on page 181.
- Risk posture management: *Aqua Risk Explorer* displays risk insights and enables vulnerability management prioritization.
- Runtime protection through *Aqua KubeEnforcer*: Use the Admission controller to validate a workload configuration, and block non-compliant workloads and unregistered images. For more information, see Aqua KubeEnforcer.
- Center for Internet Security (CIS) benchmark compliance checks: Use CIS K8s benchmark checks through kube-bench, which is validated on Red Hat OpenShift on IBM Power.
- Drift prevention for safeguarding against misuse or abuse of resources with container immutability enforcement.
- Scan Red Hat OpenShift hosts running on IBM Power Architecture for malware and vulnerabilities.
- Enforce network segmentation to restrict blast radius and protect against IP addresses and DNS domains with bad reputations and crypto-mining attacks.

10.6.1 Installing Aqua Security operator

To install Aqua Security Operator by using Red Hat OpenShift OperatorHub, complete the following steps. For more information, see Deploy Red Hat OpenShift Operator. Ensure that you have a valid Aqua Security license.

1. Create a project for Aqua by running the command that is shown in Example 10-1.

Example 10-1 Creating an Aqua project

```
oc new-project aqua
```

2. Create the secret for the Aqua Database password by running the command that is shown in Example 10-2.

Example 10-2 Creating a secret for Aqua

```
oc create secret generic aqua-database-password
-from-literal=db-password=<password> -n aqua
```

3. Install Aqua from Red Hat OpenShift OperatorHub, as shown in Figure 10-4.

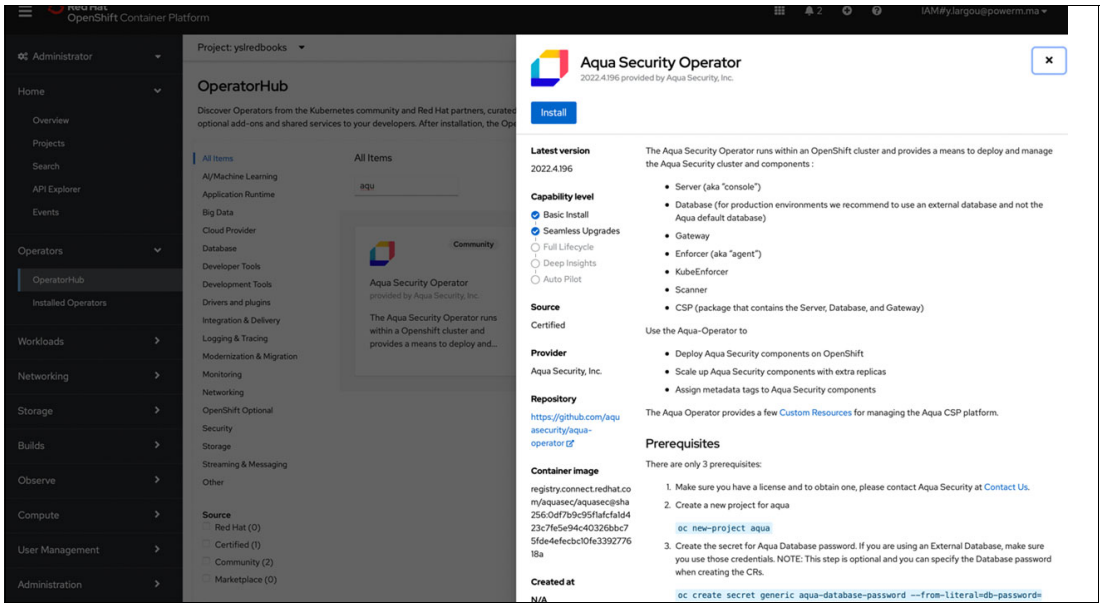


Figure 10-4 Installing Aqua Security Operator

4. Verify the Aqua Operator applications, as shown in Figure 10-5.

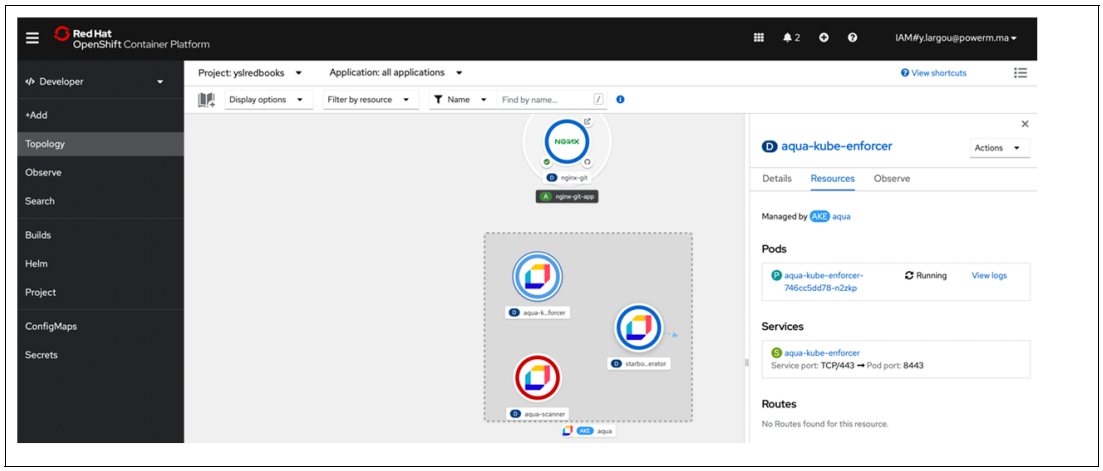


Figure 10-5 Verifying the Aqua Security Operator applications

Note: To obtain an Aqua license, contact Aqua Security Services and bring IBM Security Services into the conversation for implementation and managed security services.

10.6.2 Scanning for vulnerabilities by using Aqua Trivy and Starboard

You can scan for vulnerabilities by using the following tools:

- Using Aqua Trivy
- Using Aqua Starboard

Using Aqua Trivy

In this scenario, you scan a container image to detect vulnerabilities and misconfiguration in a GitLab runner operator on an IBM Power server. For more information, see the documentation at Aqua Trivy.

Complete the following steps:

1. Install and build Aqua Trivy by running the commands that are shown in Example 10-3.

Example 10-3 Installing and building Aqua Trivy

```
$ git clone https://github.com/aquasecurity/trivy.git

Cloning into 'trivy'...
remote: Enumerating objects: 62358, done.
remote: Counting objects: 100% (524/524), done.
remote: Compressing objects: 100% (225/225), done.
remote: Total 62358 (delta 209), reused 429 (delta 165), pack-reused 61834
Receiving objects: 100% (62358/62358), 793.41 MiB | 16.99 MiB/s, done.
Resolving deltas: 100% (32003/32003), done.
Updating files: 100% (1326/1326), done.

$ cd trivy

$ docker build -t quay.io/snehakpersistent/trivy:ppc64le .
```

2. Scan the container image by running the command that is shown in Example 10-4.

Example 10-4 Scanning a container image to detect vulnerabilities in a GitLab runner operator

```
$ trivy image
registry.gitlab.com/skanekar1/gitlab-runner-operator/gitlab-runner-operator:linux-ppc64le-v0.0.1-25940
2c6

2023-01-15T12:21:11.433+0100INFONeed to update DB
2023-01-15T12:21:11.434+0100INFODB Repository: ghcr.io/aquasecurity/trivy-db
2023-01-15T12:21:11.434+0100INFODownloading DB...
36.06 MiB / 36.06 MiB
[----------------------------------------------------------------------------------------------------]
100.00% 6.32 MiB p/s 5.9s
2023-01-15T12:21:18.683+0100INFOVulnerability scanning is enabled
2023-01-15T12:21:18.683+0100INFOSecret scanning is enabled
2023-01-15T12:21:18.683+0100INFOIf your scanning is slow, try '--security-checks vuln' to disable
secret scanning
2023-01-15T12:21:18.683+0100INFOSee also
https://aquasecurity.github.io/trivy/v0.36/docs/secret/scanning/#recommendation for faster secret
detection
2023-01-15T12:21:28.664+0100INFODetected OS: Red Hat
2023-01-15T12:21:28.665+0100INFODetecting RHEL/CentOS vulnerabilities...
2023-01-15T12:21:28.685+0100INFONumber of language-specific files: 1
2023-01-15T12:21:28.685+0100INFODetecting gobinary vulnerabilities...

registry.gitlab.com/skanekar1/gitlab-runner-operator/gitlab-runner-operator:linux-ppc64le-v0.0.1-25940
2c6 (Red Hat 8.4)
```

```
Total: 189 (UNKNOWN: 0, LOW: 55, MEDIUM: 123, HIGH: 11, CRITICAL: 0)
manager (gobinary)

Total: 14 (UNKNOWN: 0, LOW: 0, MEDIUM: 4, HIGH: 10, CRITICAL: 0)
```

Part of the output from the Aqua Trivy scan is a list of known vulnerabilities for container contents. An example of this output is shown in Table 10-2 and Table 10-3 on page 183.

Table 10-2 Vulnerabilities discovered

Library	Vulnerability	Severity	Installed version	Fixed version	Title
bzip2-libs	CVE-2019-12900	Low		1.0.6-26.el8	bzip2: out-of-bounds write in function BZ2_decompress https://avd.aquasec.com/nvd/cve-2019-12900
curl	CVE-2021-22876	Medium	7.61.1-18.el8	7.61.1-22.el8	curl: Leak of authentication credentials in URL via automatic Referer https://avd.aquasec.com/nvd/cve-2021-22876
	CVE-2021-22922			7.61.1-18.el8_4.1	curl: Content not matching hash in Metalink is not being discarded https://avd.aquasec.com/nvd/cve-2021-22922
	CVE-2021-22923				curl: Metalink download sends credentials https://avd.aquasec.com/nvd/cve-2021-22923
	CVE-2021-22924				curl: Bad connection reuse due to flawed path name checks https://avd.aquasec.com/nvd/cve-2021-22924
	CVE-2021-22946			7.61.1-18.el8_4.2	curl: Requirement to use TLS not properly enforced for IMAP,POP3, and... https://avd.aquasec.com/nvd/cve-2021-22946
	CVE-2021-22947				curl: Server responses received before STARTTLS processed after TLS handshake https://avd.aquasec.com/nvd/cve-2021-22947
	CVE-2022-22576			7.61.1-22.el8_6.3	curl: OAUTH2 bearer bypass in connection re-use https://avd.aquasec.com/nvd/cve-2022-22576
	CVE-2022-27774				curl: credential leak on redirect https://avd.aquasec.com/nvd/cve-2022-27774

Library	Vulnerability	Severity	Installed version	Fixed version	Title
curl	CVE-2022-27776	Medium	7.61.1-18.el8	7.61.1-22.el8_6.3	curl: auth/cookie leak on redirect https://avd.aquasec.com/nvd/cve-2022-27776
	CVE-2022-27782				curl: TLS and SSH connection too eager reuse https://avd.aquasec.com/nvd/cve-2022-27782
	CVE-2022-32206			7.61.1-22.el8_6.4	curl: HTTP compression denial of service https://avd.aquasec.com/nvd/cve-2022-32206
	CVE-2022-32208				curl: FTP-KRB bad message verification https://avd.aquasec.com/nvd/cve-2022-32208
xz-libs	CVE-2022-1271	High	5.2.4-3.el8	5.2.4-4.el8_6	gzip: arbitrary-file-write vulnerability https://avd.aquasec.com/nvd/cve-2022-1271
zlib	CVE-2018-25032		1.2.11-17.el8	1.2.11-18.el8_5	zlib: A flaw found in zlib when compressing (not decompressing) certain inputs... https://avd.aquasec.com/nvd/cve-2018-25032
	CVE-2022-37434	Medium		1.2.11-19.el8_6	zlib: heap-based buffer over-read and overflow in inflate() in inflate.c via a... https://avd.aquasec.com/nvd/cve-2022-37434

Table 10-3 Additional vulnerabilities discovered

Library	Vulnerability	Severity	Installed version	Fixed version	Title
github.com/gogo/protobuf	CVE-2021-3121	High	v1.3.1	1.3.2	gogo/protobuf: plugin/unmarshal/unmarshal.go lacks certain index validation https://avd.aquasec.com/nvd/cve-2021-3121
github.com/prometheus/client_golang	CVE-2022-21698		v1.0.0	1.11.1	prometheus/client_golang: Denial of service using InstrumentHandlerCounter
k8s.io/client-go	CVE-2020-8565	Medium	v0.18.6	0.20.0-alpha.2	K8s: Incomplete fix for CVE-2019-11250 allows for token leak in logs when... https://avd.aquasec.com/nvd/cve-2020-8565

Using Aqua Starboard

In this scenario, you use the Vulnerability Scanner to generate vulnerability reports on the `ngnix` pod by using Aqua Starboard. For more information, see Aqua Starboard.

Complete the following steps:

1. Install and build Aqua Starboard by running the commands that are shown in Example 10-5.

Example 10-5 Installing and building Aqua Starboard

```
$ git clone https://github.com/snehakpersistent/starboard.git

Cloning into 'starboard'...
remote: Enumerating objects: 7214, done.
remote: Total 7214 (delta 0), reused 0 (delta 0), pack-reused 7214
Receiving objects: 100% (7214/7214), 60.62 MiB | 14.79 MiB/s, done.
Resolving deltas: 100% (4168/4168), done.

$ cd starboard

$ make

CGO_ENABLED=0 go build -o ./bin/starboard ./cmd/starboard/main.go
go: downloading k8s.io/client-go v0.24.4
go: downloading k8s.io/klog/v2 v2.70.1
go: downloading github.com/spf13/cobra v1.5.0
go: downloading github.com/spf13/pflag v1.0.5
go: downloading k8s.io/api v0.24.4
go: downloading k8s.io/apiextensions-apiserver v0.24.2
go: downloading k8s.io/apimachinery v0.24.4
go: downloading k8s.io/cli-runtime v0.24.1
go: downloading k8s.io/utils v0.0.0-20220706174534-f6158b442e7c
go: downloading sigs.k8s.io/controller-runtime v0.12.3
go: downloading github.com/google/go-containerregistry v0.11.0
go: downloading github.com/go-logr/logr v1.2.3
go: downloading github.com/emirpasic/gods v1.18.1
go: downloading github.com/google/uuid v1.3.0
go: downloading github.com/davecgh/go-spew v1.1.1
go: downloading github.com/hashicorp/go-version v1.5.0
go: downloading github.com/gogo/protobuf v1.3.2
go: downloading github.com/google/gofuzz v1.2.0
go: downloading github.com/liggitt/tabwriter v0.0.0-20181228230101-89fcab3d43de
go: downloading sigs.k8s.io/yaml v1.3.0
go: downloading github.com/evanphx/json-patch v5.6.0+incompatible
go: downloading sigs.k8s.io/structured-merge-diff/v4 v4.2.1
go: downloading github.com/gorhill/cronexpr v0.0.0-20180427100037-88b0669f7d75
go: downloading github.com/opencontainers/go-digest v1.0.0
go: downloading github.com/caarlos0/env/v6 v6.10.0
go: downloading github.com/open-policy-agent/opa v0.44.0
go: downloading github.com/golang/protobuf v1.5.2
go: downloading github.com/google/gnostic v0.6.9
go: downloading golang.org/x/net v0.0.0-20220906165146-f3363e06e74c
go: downloading golang.org/x/time v0.0.0-20220609170525-579cf78fd858
go: downloading gopkg.in/inf.v0 v0.9.1
go: downloading github.com/valyala/quicktemplate v1.7.0
go: downloading github.com/Azure/go-autorest/autorest v0.11.24
go: downloading github.com/Azure/go-autorest v14.2.0+incompatible
go: downloading github.com/Azure/go-autorest/autorest/adal v0.9.18
go: downloading golang.org/x/oauth2 v0.0.0-20220718184931-c8730f7fcb92
go: downloading sigs.k8s.io/json v0.0.0-20220525155127-227cbc7cc124
go: downloading gopkg.in/yaml.v2 v2.4.0
go: downloading github.com/pkg/errors v0.9.1
go: downloading golang.org/x/text v0.3.8
go: downloading sigs.k8s.io/kustomize/api v0.11.4
go: downloading sigs.k8s.io/kustomize/kyaml v0.13.6
go: downloading github.com/gregjones/httpcache v0.0.0-20190611155906-901d90724c79
go: downloading github.com/peterbourgon/diskv v2.0.1+incompatible
go: downloading github.com/imdario/mergo v0.3.13
go: downloading golang.org/x/term v0.0.0-20220526004731-065cf7ba2467
go: downloading github.com/prometheus/client_golang v1.13.0
go: downloading k8s.io/kube-openapi v0.0.0-20220627174259-011e075b9cb8
go: downloading github.com/json-iterator/go v1.1.12
go: downloading github.com/OneOfOne/xxhash v1.2.8
go: downloading gomodules.xyz/jsonpatch/v2 v2.2.0
go: downloading google.golang.org/protobuf v1.28.1
go: downloading gopkg.in/yaml.v3 v3.0.1
go: downloading github.com/google/go-cmp v0.5.8
go: downloading github.com/valyala/bytebufferpool v1.0.0
```

```
go: downloading github.com/Azure/go-autorest/logger v0.2.1
go: downloading github.com/Azure/go-autorest/tracing v0.6.0
go: downloading github.com/Azure/go-autorest/autorest/date v0.3.0
go: downloading github.com/golang-jwt/jwt/v4 v4.2.0
go: downloading golang.org/x/crypto v0.0.0-20220622213112-05595931fe9d
go: downloading cloud.google.com/go v0.99.0
go: downloading github.com/google/btree v1.0.1
go: downloading golang.org/x/sys v0.0.0-20220728004956-3c1f35247d10
go: downloading k8s.io/component-base v0.24.2
go: downloading github.com/prometheus/client_model v0.2.0
go: downloading github.com/prometheus/common v0.37.0
go: downloading github.com/golang/groupcache v0.0.0-20210331224755-41bb18bfe9da
go: downloading github.com/modern-go/concurrent v0.0.0-20180306012644-bacd9c7ef1dd
go: downloading github.com/modern-go/reflect2 v1.0.2
go: downloading github.com/xeipuuv/gojsonreference v0.0.0-20180127040603-bd5ef7bd5415
go: downloading github.com/rcrowley/go-metrics v0.0.0-20201227073835-cf1acfcdf475
go: downloading github.com/ghodss/yaml v1.0.0
go: downloading github.com/gobwas/glob v0.2.3
go: downloading github.com/tchap/go-patricia/v2 v2.3.1
go: downloading github.com/yashtewari/glob-intersection v0.1.0
go: downloading github.com/munnerz/goautoneg v0.0.0-20191010083416-a7dc8b61c822
go: downloading github.com/beorn7/perks v1.0.1
go: downloading github.com/cespare/xxhash/v2 v2.1.2
go: downloading github.com/prometheus/procfs v0.8.0
go: downloading github.com/matttproud/golang_protobuf_extensions v1.0.2-0.20181231171920-c182affec369
go: downloading github.com/fsnotify/fsnotify v1.5.4
go: downloading github.com/xeipuuv/gojsonpointer v0.0.0-20190905194746-02993c407bfb
go: downloading github.com/agnivade/levenshtein v1.1.1
go: downloading github.com/emicklei/go-restful/v3 v3.8.0
go: downloading github.com/go-openapi/swag v0.21.1
go: downloading github.com/go-openapi/jsonreference v0.20.0
go: downloading github.com/google/shlex v0.0.0-20191202100458-e7afc7fbc510
go: downloading github.com/monochromegane/go-gitignore v0.0.0-20200626010858-205db1a8cc00
go: downloading github.com/stretchr/testify v1.8.0
go: downloading github.com/xlab/treeprint v0.0.0-20181112141820-a009c3971eca
go: downloading github.com/go-errors/errors v1.0.1
go: downloading github.com/mailru/easyjson v0.7.7
go: downloading github.com/go-openapi/jsonpointer v0.19.5
go: downloading github.com/josharian/intern v1.0.0
go: downloading go.starlark.net v0.0.0-20200306205701-8dd3e2ee1dd5
go: downloading github.com/pmezard/go-difflib v1.0.0

CGO_ENABLED=0 GOOS=linux go build -o ./bin/starboard-operator ./cmd/starboard-operator/main.go
go: downloading github.com/go-logr/zapr v1.2.3
go: downloading go.uber.org/zap v1.21.0
go: downloading go.uber.org/atomic v1.9.0
go: downloading go.uber.org/multierr v1.8.0
CGO_ENABLED=0 GOOS=linux go build -o ./bin/starboard-scanner-aqua ./cmd/scanner-aqua/main.go
```

$ cd bin

$ ./starboard init -v 3

```
I0115 15:29:02.427038    4704 installer.go:410] Creating CRD "vulnerabilityreports.aquasecurity.github.io"
I0115 15:29:02.478507    4704 installer.go:410] Creating CRD "clustervulnerabilityreports.aquasecurity.github.io"
I0115 15:29:02.516438    4704 installer.go:410] Creating CRD "ciskubebenchreports.aquasecurity.github.io"
I0115 15:29:02.548759    4704 installer.go:410] Creating CRD "kubehunterreports.aquasecurity.github.io"
I0115 15:29:02.590130    4704 installer.go:404] Updating CRD "configauditreports.aquasecurity.github.io"
I0115 15:29:02.625052    4704 installer.go:404] Updating CRD "clusterconfigauditreports.aquasecurity.github.io"
I0115 15:29:02.818666    4704 request.go:533] Waited for 192.558459ms due to client-side throttling, not priority and
fairness, request:
PUT:https://c100-e.eu-de.containers.cloud.ibm.com:30428/apis/apiextensions.k8s.io/v1/customresourcedefinitions/cluste
rconfigauditreports.aquasecurity.github.io
I0115 15:29:03.018043    4704 request.go:533] Waited for 169.457821ms due to client-side throttling, not priority and
fairness, request:
GET:https://c100-e.eu-de.containers.cloud.ibm.com:30428/apis/apiextensions.k8s.io/v1/customresourcedefinitions/cluste
rcompliancereports.aquasecurity.github.io
I0115 15:29:03.032517    4704 installer.go:410] Creating CRD "clustercompliancereports.aquasecurity.github.io"
(….)
I0115 15:29:13.457135    4704 installer.go:427] Creating compliance spec "nsa"
I0115 15:29:13.516531    4704 installer.go:342] Creating Namespace "starboard"
I0115 15:29:13.931599    4704 installer.go:357] Creating ServiceAccount "starboard/starboard"
I0115 15:29:13.975981    4704 installer.go:374] Creating ClusterRole "starboard"
I0115 15:29:14.029941    4704 installer.go:392] Creating ClusterRoleBinding "starboard"

Starboard is an Aqua Security open source project.
Learn about our open source work and portfolio on https://www.aquasec.com/products/open-source-projects/.
```

2. Use the Vulnerability Scanner to generate vulnerability reports on the `ngnix` pod by using Aqua Starboard. Run the commands that are shown in Example 10-6.

Example 10-6 Generating vulnerability reports on the ngnix image

```
$ kubectl get deployment --namespace yslpowertst

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
aqua-kube-enforcer  0/1     1            0           66m
aqua-operator       1/1     1            1           5h31m
aqua-scanner        0/1     1            0           66m
nginx-git           1/1     1            1           30m
starboard-operator  1/1     1            1           66m

# Generate vulnerability reports:

$ ./starboard scan vulnerabilityreports deployment/nginx-git  --namespace yslpowertst -v 3

# Retrieve the vulnerability report:

$ ./starboard get vulnerabilities deployment/nginx-git  -o yaml --namespace yslpowertst -v 3
```

Glossary

alerting rules Alerting rules contain a set of conditions that outline a state within a cluster. Alerts are triggered when those conditions are true. An alerting rule can be assigned a severity that defines how the alerts are routed.

Alertmanager Handles alerts that are received from Prometheus. Alertmanager also is responsible for sending the alerts to external notification systems.

continuous integration and continuous deployment (CI/CD) A CI/CD pipeline is a set of automated processes that manage the flow of code changes from development to production.

cloud-native Refers to an application that is designed to reside in the cloud from the start. Cloud-native involves cloud technologies like microservices, container orchestrators, and auto scaling.

Cluster Monitoring Operator (CMO) A central component of the monitoring stack. It deploys and manages Prometheus instances, such as the Thanos Querier, the Telemeter Client, and metrics targets to ensure that they are up to date. The CMO is deployed by the Cluster Version Operator (CVO).

Cloud-Native Application Protection Platform (CNAPP) A set of security tools and services to protect applications running in cloud environments, such as a public cloud or private cloud infrastructure.

configmap Provides a way to inject configuration data into pods. You can reference the data that is stored in a config map in a volume of type ConfigMap. Applications running in a pod can use this data.

Container Storage Interface (CSI) A plug-in for Kubernetes (K8s) and other container orchestrators that allow storage suppliers to expose their products to containerized applications as persistent storage.

container A container is a lightweight and executable image that includes software and all its dependencies. Containers virtualize the operating system. As a result, you can run containers anywhere from a data center to a public or private cloud, or a developer's laptop.

custom resource (CR) An extension of the K8s application programming interface (API). You can create custom resources.

common vulnerabilities and exposures (CVEs) The database of publicly disclosed information on security issues. All organizations use CVEs to identify and track vulnerabilities.

distributed application A software application that is designed to run on multiple regions and servers that are connected over multiple networks. Distributed applications can provide scalability and high availability for an application.

Elasticsearch A distributed, no-charge, and open search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured.

enterprise Baseboard Management Controller (eBMC) A specialized service processor that monitors the physical state of the system by using sensors. A system administrator or service representative can communicate with the eBMC through an independent connection. The eBMC is used instead of the Flexible Service Processor (FSP) starting with IBM Power10 servers.

etcd The key-value store for Red Hat OpenShift Container Platform, which stores the state of all resource objects.

Flexible Service Processor (FSP) An always-on management processor that helps to manage a server out-of-band. The FSP is the external face of a Power server that provides various platform management interfaces.

Fluentd Gathers logs from nodes and feeds them to Elasticsearch.

Kibana A no-charge and open front-end application that sits on top of the Elastic Stack that provides search and data visualization capabilities for data that is indexed in Elasticsearch. Commonly known as the charting tool for the Elastic Stack (previously referred to as the ELK Stack after Elasticsearch, Logstash, and Kibana), Kibana also acts as the user interface for monitoring, managing, and securing an Elastic Stack cluster

kubelets Runs on nodes and reads the container manifests. Ensures that the defined containers started and are running.

Kubernetes API server Validates and configures data for the API objects.

labels Key-value pairs that you can use to organize and select subsets of objects, such as a pod.

Logstash An open-source data collection engine with real-time pipelining capabilities. Logstash can dynamically unify data from disparate sources and normalize the data into destinations of your choice.

metering A general-purpose data analysis tool that enables you to write reports to process data from different data sources.

node A worker machine in the Red Hat OpenShift Container Platform cluster. A node is either a virtual machine (VM) or a physical machine.

observability In K8s, the continuous process of using the metrics, events, logs, and trace data that a K8s system generates to identify, understand, and optimize its health and performance.

operator The preferred method of packaging, deploying, and managing a K8s application in an Red Hat OpenShift Container Platform cluster. An operator takes human operational knowledge and encodes it into software that is packaged and shared with customers.

persistent storage Stores the data even after the device is shut down. K8s uses persistent volumes to store the application data.

persistent volume claim (PVC) You can use a PVC to mount a persistent volume into a pod. You can access the storage without knowing the details of the cloud environment.

Pod The pod is the smallest logical unit in K8s. A pod is composed of one or more containers that run in a worker node.

Prometheus The monitoring system on which the Red Hat OpenShift Container Platform monitoring stack is based. Prometheus is a time-series database and a rule evaluation engine for metrics. Prometheus sends alerts to Alertmanager for processing.

security posture A measure of an organization's overall cybersecurity status. It also is a measure of how vulnerable an organization is to cyberattacks or data breaches.

service-level agreement (SLA) A commitment between a service provider and a customer. Aspects of the service (quality, availability, and responsibilities) are agreed between the service provider and the service user.

service-level indicator (SLI) A specific metric that helps companies measure some aspect of the level of services to their customers. SLIs are a smaller subsection of service-level objectives (SLOs), which are part of an SLA that impacts overall service reliability.

service-level objective (SLO) An agreement within an SLA about a specific metric like uptime or response time. So, if the SLA is the formal agreement between you and your customer, SLOs are the individual promises that you are making to that customer.

storage class Provides a way for administrators to describe the "classes" of storage that they offer. Different classes might configmap to quality of service (QoS) levels, backup policies, or arbitrary policies that are determined by the cluster administrators.

Thanos Ruler A rule evaluation engine for Prometheus that is deployed as a separate process. In Red Hat OpenShift Container Platform, Thanos Ruler provides rule and alerting evaluation for the monitoring of user-defined projects.

Trusted Platform Module (TPM) Enables remote attestation of the code stack on a running system. The chain of trust firmware records the hash of the loaded firmware and stores the records in the network of processor TPMs.

web console A user interface (UI) to manage Red Hat OpenShift Container Platform.

YAML A data serialization language that often is used for writing configuration files. YAML is a popular programming language because it is human-readable and easy to understand. Because of its flexibility and accessibility, YAML is used by the Ansible automation tool to create automation processes.

Abbreviations and acronyms

ACSC	Australian Cyber Security Centre
AI	artificial intelligence
AIDE	Advanced Intrusion Detection Environment
AIOps	Artificial Intelligence for IT Operations
API	application programming interface
APM	application performance monitoring
ARF	Assessment Results Format
BIND	Berkeley Internet Name Domain
BYOK	bring-your-own-key
CA	certificate authority
CEng	Chartered Engineer
CI/CD	continuous integration and continuous delivery
CIS	Center for Internet Security
CLI	command-line interface
CNAPP	Cloud Native Application Protection Platform
CNI	Container Network Interface
CNO	Cluster Network Operator
CSI	Container Storage Interface
CSRF	cross-site request forgery
DAST	Dynamic Application Security Testing
DDoS	distributed denial-of-service
DEXCR	Dynamic Execution Control Register
DNS	Domain Name System
eBMC	enterprise Baseboard Management Controller
EPC	Enterprise Protected Containers
ESB	Enterprise Service Bus
ETSI	European Telecommunications Standards Institute
FHE	Fully Homomorphic Encryption
FIPS	Federal Information Processing Standards
FSP	Flexible Service Processor
GCP	Google Cloud Platform
GDPR	General Data Protection Regulation
HPC	high-performance computing
HPCS	IBM Hyper Protect Crypto Services
HSM	Hardware Security Module
IaaS	infrastructure as a service
IaC	Infrastructure as Code
IAM	Identity and Access Management
IBM	International Business Machines Corporation
IBM PowerVS	IBM Power Systems Virtual Server
IMA	Integrity Measurement Architecture
K8s	Kubernetes
KMS	Key Management System
KPI	key performance indicator
KYOK	keep your own key
LGPD	Lei Geral de Proteção de Dados
LPAR	logical partition
LUW	Linux, UNIX, and Windows
MAC	mandatory access control
MCG	Multicloud Object Gateway
MFA	multi-factor authentication
mTLS	mutual Transport Layer Security
NFS	Network File System
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OIDC	OpenID Connect
OpenSSF	Open Source Security Foundation
OWASP	Open Web Application Security Project
PaaS	platform as a service
PIBR	Privacy Incident Benchmark Report
PKCS	Public Key Cryptography Standards
PKI	public key infrastructure
PMP	Project Management Professional
PQC	post-quantum cryptography
PV	persistent volume
PVC	persistent volume claim
RBAC	role-based access control
RHEL	Red Hat Enterprise Linux
ROKS	Red Hat OpenShift Kubernetes Service
ROP	return-oriented programming
ROX	ReadOnlyMany

RWO	ReadWriteOnce
RWX	ReadWriteMany
S2I	Source-2-Image
SaaS	software as a service
SAML	Security Assertion Markup Language
SAST	Static Application Security Testing
SCAP	Security Content Automation Protocol
SCC	Security Context Constraints
SDN	software-defined networking
SIEM	Security Incident and Event Manager
SLA	service-level agreement
SLI	service-level indicator
SLO	service-level objective
SRE	Site Reliability Engineer
SSH	Secure Socket Shell
SSL	Secure Sockets Layer
TDE	Transparent Data Encryption
TLS	Transport Layer Security
TOE	target of evaluation
TPM	Trusted Platform Module
UBI	Universal Base Image
UKO	Universal Key Orchestrator
VIOS	Virtual I/O Server
VM	virtual machine
VMI	Virtualization Management Interface

Related publications

The publications that are listed in this section are considered suitable for a more detailed description of the topics that are covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topics in this document. Some publications that are referenced in this list might be available in softcopy only.

- *IBM Power Systems Cloud Security Guide: Protect IT Infrastructure In All Layers*, REDP-5659
- *IBM Spectrum Protect Plus: Protecting Red Hat OpenShift Containerized Environments*, REDP-5636
- *IBM Spectrum Scale CSI Driver for Container Persistent Storage*, REDP-5589
- *IBM Storage for Red Hat OpenShift Blueprint*, REDP-5565
- *Using the IBM Block Storage CSI Driver in a Red Hat OpenShift Environment*, REDP-5613

You can search for, view, download, or order these documents and other Redbooks, Redpapers, web docs, drafts, and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- Getting started with Security and Compliance Center

  https://cloud.ibm.com/docs/security-compliance?topic=security-compliance-getting-started

- Installing and configuring the Red Hat OpenShift API for Data Protection with Multicloud Object Gateway

  https://docs.openshift.com/container-platform/4.11/backup_and_restore/application_backup_and_restore/installing/installing-oadp-mcg.html

- Installing and configuring the Red Hat OpenShift API for Data Protection with Red Hat OpenShift Data Foundation

  https://docs.openshift.com/container-platform/4.11/backup_and_restore/application_backup_and_restore/installing/installing-oadp-ocs.html

- Kubernetes (K8s) Documentation / Concepts / Security

  https://kubernetes.io/docs/concepts/security/

- Red Hat OpenShift Data Foundation: Data Encryption Options

  https://access.redhat.com/documentation/en-us/red_hat_openshift_data_foundation/4.11/html/planning_your_deployment/security-considerations_rhodf#data-encryption-options_rhodf
- Red Hat OpenShift Data Foundation: Enabling Cluster-Wide Encryption

  https://access.redhat.com/documentation/en-us/red_hat_openshift_data_foundation/4.11/html/planning_your_deployment/security-considerations_rhodf#storage_class_encryption
- Red Hat OpenShift Data Foundation: Managing Hybrid and Multicloud Resources

  https://access.redhat.com/documentation/en-us/red_hat_openshift_data_foundation/4.11/html/managing_hybrid_and_multicloud_resources/index
- Red Hat OpenShift Data Foundation: Persistent Volume Encryption

  https://access.redhat.com/documentation/en-us/red_hat_openshift_data_foundation/4.11/html/managing_and_allocating_storage_resources/storage-classes_rhodf#storage-class-for-persistent-volume-encryption_rhodf
- Red Hat OpenShift Container Platform: Encrypting etcd data

  https://docs.openshift.com/container-platform/4.11/security/encrypting-etcd.html
- Security for Red Hat OpenShift on IBM Cloud

  https://cloud.ibm.com/docs/openshift?topic=openshift-security
- Velero Documentation

  https://velero.io/docs/v1.9/

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

REDP-5690-00

ISBN 0738461075

Printed in U.S.A.