Red Hat OpenShift Container Platform on IBM Z and IBM LinuxONE

Reference Architecture

May 2022

Table of Contents

Notices

Trademarks

Executive summary

Organizations across the globe continue to rapidly develop innovative software applications in hybrid and multi-cloud environments to achieve competitive advantages and ensure customer satisfaction. Many of these applications are deployed in a private cloud due to security and compliance, data affinity, and performance requirements. The IT organizations responsible for operating the private cloud value simplicity, agility, flexibility, security, and cost efficiency, as these features reduce their own barriers to innovation as part of their overall hybrid and multi-cloud strategy.

The IBM Z and IBM LinuxONE platforms are very well suited for these cloud environments, due to the capability of non-disruptive vertical and horizontal scalability on demand, in the most securable platform and inheriting the reliability, stability and availability of the mainframe. In co-location with traditional workloads, they can enable data gravity in secure computing environments and fulfil the business and IT requirements of today.

This reference architecture showcases a prescriptive, pre-validated, private cloud solution from Red Hat® and IBM that provides IT as a Service (ITaaS), and the rapid provisioning and lifecycle management of containerized apps, virtual machines (VMs), and associated application and infrastructure services for cloud users such as software developers, data scientists, and solution architects. Red Hat OpenShift® Container Platform (RHOCP) and IBM z/VM are the key architecture components of this solution.

The following key characteristics highlight the value of Red Hat OpenShift Container Platform:

- An enterprise Kubernetes platform for container workloads
- Enables seamless Kubernetes deployments on any cloud or on-premises environments
- Integrated and automated installation
- Seamless platform and application updates
- Auto-scaling of resources and services
- Ability to run enterprise workloads with enterprise CI/CD services, across multiple deployments

Chapter 1. About this document

Container-based applications provide organizations with a new paradigm for application development and deployment. Kubernetes and containerized applications allow organizations to streamline their DevOps environment and rapidly deploy, develop, and maintain both stateless and stateful applications.

Red Hat OpenShift Container Platform (RHOCP) is Red Hat's enterprise Kubernetes distribution. It provides developers, IT organizations, and business leaders with a hybrid cloud application platform for developing and deploying new containerized applications on a secure platform. RHOCP also offers scalable resources that requires minimal configuration and management. Developer can create and deploy applications by delivering a consistent environment throughout the lifecycle of the application including development, deployment, and maintenance. The self-service capabilities and minimal maintenance requirements reduces the burden on IT organizations associated with deploying and maintaining application platforms.

This document provides an overview of the options available for implementing Red Hat OpenShift Container Platform (RHOCP) on IBM Z and LinuxONE. In addition, this document reviews topologies and configurations, as well as implementation choices with their pros and cons.

The reference architecture represents the best practice for installing and configuring RHOCP on IBM Z and IBM LinuxONE. The information provided by this document is intended to help organizations deploy RHOCP on IBM Z and IBM LinuxONE quickly in a reliable and supported configuration. Important design considerations and key integrations between the products are provided as part of this document. This document should not be considered an implementation guide. Official Red Hat and IBM documentation exists for all the components included within this reference architecture and should be reviewed prior to deployment.

This document is intended for Systems Integrators, IT Architects, Systems Administrators, Application Developers. It is assumed that the reader has previous experience with Kubernetes, Red Hat OpenShift Container Platform, IBM Z and IBM LinuxONE, and Red Hat® Enterprise Linux®.

For more information regarding about Red Hat OpenShift Container Platform visit: Red Hat OpenShift Container Platform Overview

Chapter 2. Value proposition

Before we dive into the architecture and key technical considerations of a deployment of Red Hat OpenShift Container Platform (RHOCP), it is essential to understand the potential benefits and values for the business.

2.1 Benefits of Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform (RHOCP) is an enterprise Kubernetes-based platform to manage and run containerized applications.

- **Faster innovation and time-to-value**: Well-defined recommendations and best practices for designing and operating RHOCP lead to a standardized architecture.
- **Simplification**: A consistent set of APIs is defined for developers and administrators. Kubernetes Operators dramatically simplify a solution architecture and required skills to operate it.
- **Safer deployments**: RHOCP automates and secures the orchestration and life-cycle management of applications within its cluster.
- **Pre-validated patterns**: Best practices for the RHOCP solution stack are validated by Red Hat and IBM, resulting in a highly prescriptive and future-oriented solution.
- **Cost efficient**: RHOCP is based on open source technologies that are fully supported by Red Hat. It includes a high degree of operational automation beside the functional capabilities, leading to cost efficient IT environments.

2.2 Benefits of running Red Hat OpenShift Container Platform on IBM Z and IBM LinuxONE

Specially, when running RHOCP on IBM Z and IBM LinuxONE, customers can take full advantage of the platform capabilities and characteristics:

- **Non-disruptively growth**: vertical and horizontal scalability help to accommodate substantial increase of workload on demand.
- Highest **scalability** for millions of containers and thousands of Linux guests in one physical machine have been proven. https://newsroom.ibm.com/2019-09-12-IBM-Unveils-z15-With-Industry-First-Data-Privacy-Capabilities
- **Containerized workload** within a state of the art microservices architecture, while accessing traditional transactional z/OS services and databases.
- Take advantage of **advanced security** and confidential cloud computing, including FIPS 140-2 Level 4 certification with the IBM Z Cryptographic capabilities. https://www.ibm.com/security/cryptocards/hsms
- **Multi tenancy** with full LPAR isolation (EAL5+) allows administrators to share a single hardware securely. Even virtual machines on the same hardware offer EAL4 certification.

2.3 Target use cases

This reference architecture is valuable for cross industry enterprises that want to deploy a private cloud solution with programmable Infrastructure as a Service (IaaS), Containers as a Service (CaaS), and Platform as a Service (PaaS) capabilities.

2.3.1 Data gravity

A key use case in which RHOCP can take advantage of the IBM Z and IBM LinuxONE platform is the co-location of containerized applications with traditional workloads (for example datalakes, databases, transactional systems, or other traditional workloads running in Linux on IBM Z or z/OS). In such a scenario the applications can be located close to the data to optimize latency, response time, deployment, security, service and cost. In combination with Oracle databases as an example, the resulting synergy can dramatically improve the Total Cost of Ownership (TCO).

Functional Description:

- Bring cloud-native applications closer to systems of record and enterprise databases
  - Integrate REST services and data in z/OS with RHOCP through z/OS connect
  - Integrate RHOCP applications with core enterprise databases (for example PostgreSQL, DB2, Oracle)
  - Achieve latency advantages
  - Build secure cloud solutions
- Digitalization & modernization
  - Introduce microservices & containers
- Benefit from the security features provided by the IBM Z and IBM LinuxONE platform
- Dynamic, granular resource allocation & scalability
- Enable high flexibility through resource sharing
- Make use of internal network capabilities using a load balancer to connect z/OS to RHOCP
- Automate z/OS tasks via IBM Cloud Broker & Ansible® in z/OS
  - Enable CI/CD and life-cycle management
  - Total Cost of Ownership reduction

2.3.2 Consolidation and TCO Reduction

When consolidating a RHOCP environment from x86 platforms to IBM Z and IBM LinuxONE, customers can achieve economic and operational advantages. Most of the 3-dimension scalability (vertical, horizontal and combined) results in an high flexibility without the need for new hardware footprint. This is a key advantage for dynamic workloads and unpredicted growth.

In summary these key benefits can be achieved, when implementing this use case:

- Fewer physical resources needed

- Less operational endpoints to manage
- Less security endpoints to control
- Economic advantages due to reduced number of subscriptions needed
- Disaster Recovery (DR) with Capacity Backup Upgrade (CBU) to save license costs

2.3.3 Business continuity

For business continuity consider these two different aspects:

- High availability (HA)
- Disaster recovery (DR)

For high availability (HA), IBM Z and IBM LinuxONE provides an internal network with significantly more reliability based on a higher degree of redundancy in the hardware. Because virtualization happens within a single hardware environment, the networking traffic is more predictable with less latency compared to x86 environments. In an environment with z/VM you can take advantage of the management of z/VM Single System Image (SSI) and Live Guest Relocation (LGR) can reduce operational costs (requires shared ECKD dasd).

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

In IBM Z the hardware consolidation leads to the fact that Capacity and Operational analytics & analysis is simplified. This is because all results are consolidated on the single hardware machine for the entire RHOCP environment versus having a distributed environment with many physical servers. As a result, capacity planning is more predictable.

2.3.4 Vertical Solutions

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

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

Additional industries which can benefit include:

- Media
- Telecommunications
- Education
- Transportation

- Pharmaceutical & medical products, healthcare
- Hospitality & travel
- Manufacturing
- Utilities and energy
- Electronics
- Chemicals
- Consumer products
- Metals & natural resources
- Construction & engineering
- Food & beverage processing

Chapter 3. Planning and architecturing overview

In this chapter you will learn architectural choices and understand how RHOCP can be deployed and used in typical environments on IBM Z and IBM LinuxONE.

3.1 Reference architecture high level design

RHOCP is designed to facilitate the orchestration, deployment, and management of running containerized applications. Therefore, the key architectural challenges of your deployment are to consider how you can ensure that your applications adhere to best practices of a cloud native container workload. For greenfield application development this might be an easy task to design your solutions as microservices and deploy them as containers, which can interact with each other through light-weight protocols and use well defined APIs. For monolithic applications the packaging as containerized workloads will most likely be more of a challenge and require planning of transformation efforts.

When planning to deploy a RHOCP cluster it is important to understand the IBM Z and IBM LinuxONE architecture and decide if you want to plan for Proof of Concept (PoC) or Proof of Technology (PoT) environments or rather production-like environments for larger workloads.

When deploying an RHOCP environment on IBM Z and IBM LinuxONE, you have to consider that the machine needs to be partitioned first, which represents a hardware level virtualization allowing you to share resources across those partitions like processor/core capacity, I/O and network, but still remaining multi-tenant due to the highest isolation certification in industry *Evaluation. Assurance Level 5+* (EAL5+). The resulting Logical Partitions (LPARs) are the base for building different workload environments that can run side by side on the same hardware machine. The LPARs can reside on one or multiple physical machines.

3.1.1 Typical RHOCP Deployments and Topologies

Depending on the use case, you can choose from different topologies and deployment options for RHOCP:

1) Deploying RHOCP as the only cloud environment on IBM Z and IBM LinuxONE

2) Deploying RHOCP in conjunction with another, non-containerized workload that you also run as back-end in a Linux environment on IBM Z and IBM LinuxONE hardware

3) Deploying RHOCP in co-location with a traditional operating and transactional environment and services such as z/OS, z/VSE, or z/TPF on IBM Z hardware.

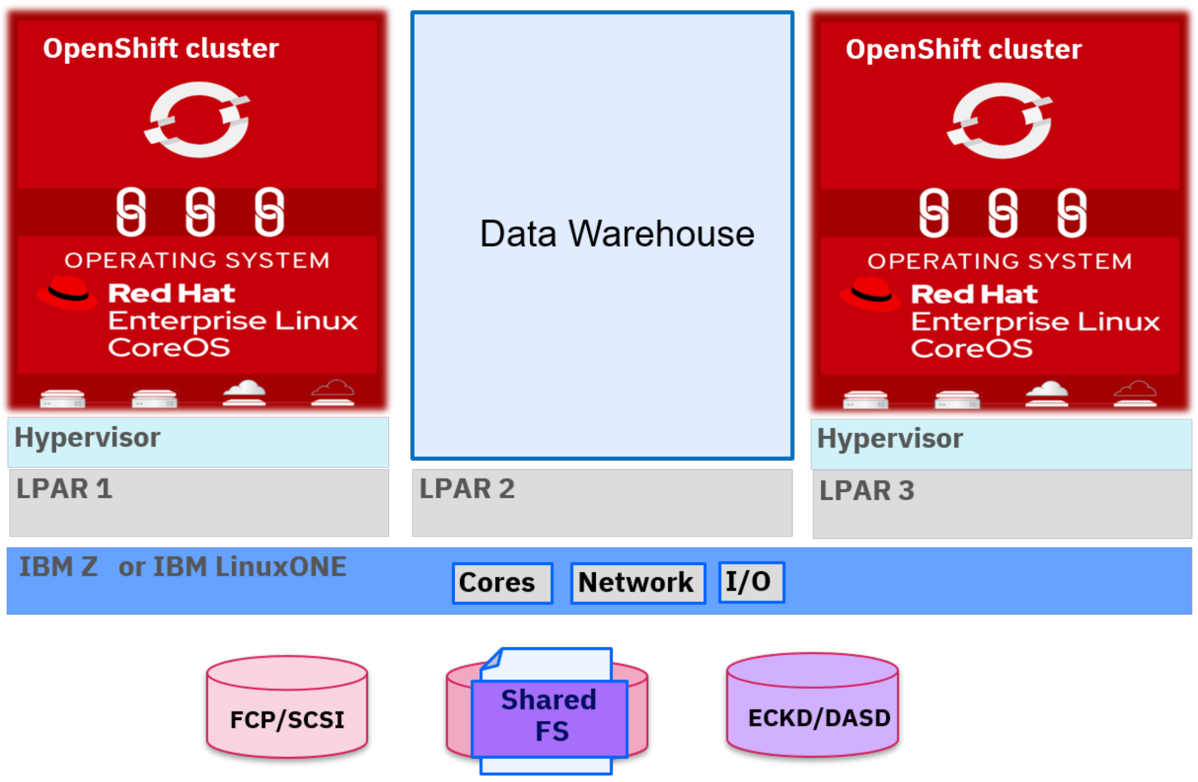


Figure 3.1-1: RHOCP Topologies on IBM Z and LinuxONE

Depending on the planned deployment, you can opt for different resource sharing topologies for core capacity, I/O and network. IBM Z and LinuxONE provides specialized components, which can be shared between the LPARs:

- The capacity core for running Linux and RHOCP is called Integrated Facility for Linux processor (IFL) and corresponds as component to a distributed core, but has characteristics, which enables for more workload per unit.
- The I/O attachments use FICON cards, which can be configured for traditional DASDs disks or for Fibre Channel SCSI, or FCP protocol. IBM Z and IBM LinuxONE machines have dedicated processors to handle I/O and are independent from the application.
- The network can use machine internal network capabilities called HiperSockets networks (HS) and network cards called Opensystem Adapters (OSA) or RoCE cards (Remote Direct Memory Access over Converged Ethernet cards). Multiple isolated internal networks can be defined and the ports on the network cards can be shared.

The deployment of an RHOCP environment can be realized in one or more LPARs.

The concept of an RHOCP cluster is to provide service with high availability in mind and therefore, the RHOCP cluster needs to be deployed with three control plane nodes and a number of compute nodes depending on the planned workloads and container applications. The minimum number of compute nodes is two, additional compute nodes can be added to the cluster as required.

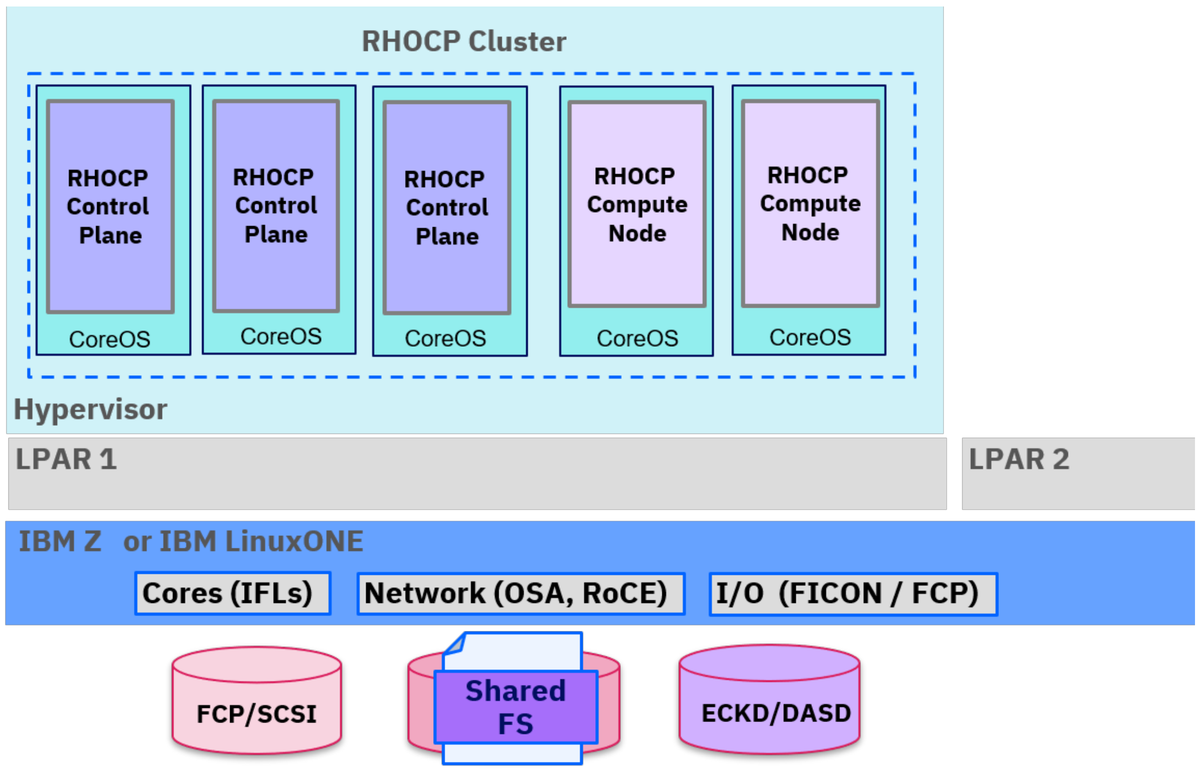


Figure 3.1-2: RHOCP Virtual machines in a Cluster

With Red Hat OpenShift 4.9 a new deployment model was made available with converged Control Planes and Compute Nodes in a total of 3 Nodes per cluster, as shown in the picture below. This can be a very helpfull for Proof of technologies, Development or Test environments.

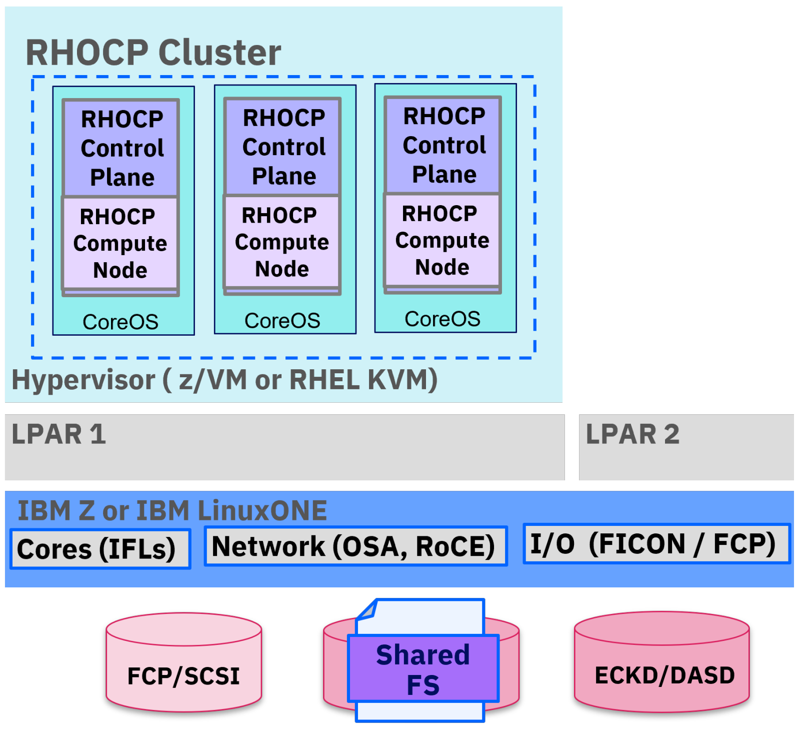


Figure 3.1-3: The RHOCP 3-Node cluster

A RHOCP cluster on IBM Z and IBM LinuxONE is deployed in a virtualized environment, which means there is a requirement for a hypervisor in the RHOCP LPAR. On IBM Z and IBM LinuxONE, the preferred hypervisors are:

- IBM z/VM Version 7.1 or later
- Red Hat Enterprise Linux 8.3 KVM (RHEL KVM) or later

For a RHOCP cluster, Red Hat Enterprise Linux CoreOS (RHCOS) is the host operating system for all nodes, control plane, and compute nodes. The RHCOS operating system is delivered as one package with the Red Hat OpenShift components. RHCOS is a self-managing, secure, and immutable container host, which is optimized for performance. RHCOS is versioned with Red Hat OpenShift. The big advantage of the use of RHCOS is that due to the fact that it is immutable, the platform can ensure security and does not need any active management by an administrator.

Running RHOCP under Red Hat Enterprise Linux on IBM Z and IBM LinuxONE is not supported.

RHOCP in Version 4 nodes run on Red Hat Enterprise Linux CoreOS (RHCOS).

NOTE The only supported operating system for RHOCP on IBM Z and IBM LinuxONE is currently RHCOS. For more information, see CoreOS has joined the Red Hat family.

To install a RHOCP cluster you can decide for different storage to be used. The installation of the hypervisor and the RHOCP guests is implemented on local storage disks.

The local storage can be FICON-attached DASD storage using Extended Count Key Data (ECKD) format or it can be Fibre Channel (FCP) attached storage using SCSI block device format. For most applications local (non-shared) storage can be sufficient.

For the RHOCP internal container image registry it is mandatory to have a shared storage with read-write-many (rwm) capabilities. For shared storage (rwm) you can use:

- NFS-mounted storage
- IBM Cloud Native Storage Access (CNSA) - a CSI-based cross cluster mount capability to attach a RHOCP cluster to an IBM Spectrum Scale clustered file system. The IBM Spectrum Sclae Cluster provides shared file system storage capabilities for container applications and can be shared between different hardware architectures.
- Red Hat OpenShift Data Foundation Storage: This option is a shared storage approach based on Ceph and the corresponding CephFS File System. Red Hat OpenShift Data Foundation Storage provides block, file and object storage via different interfaces.

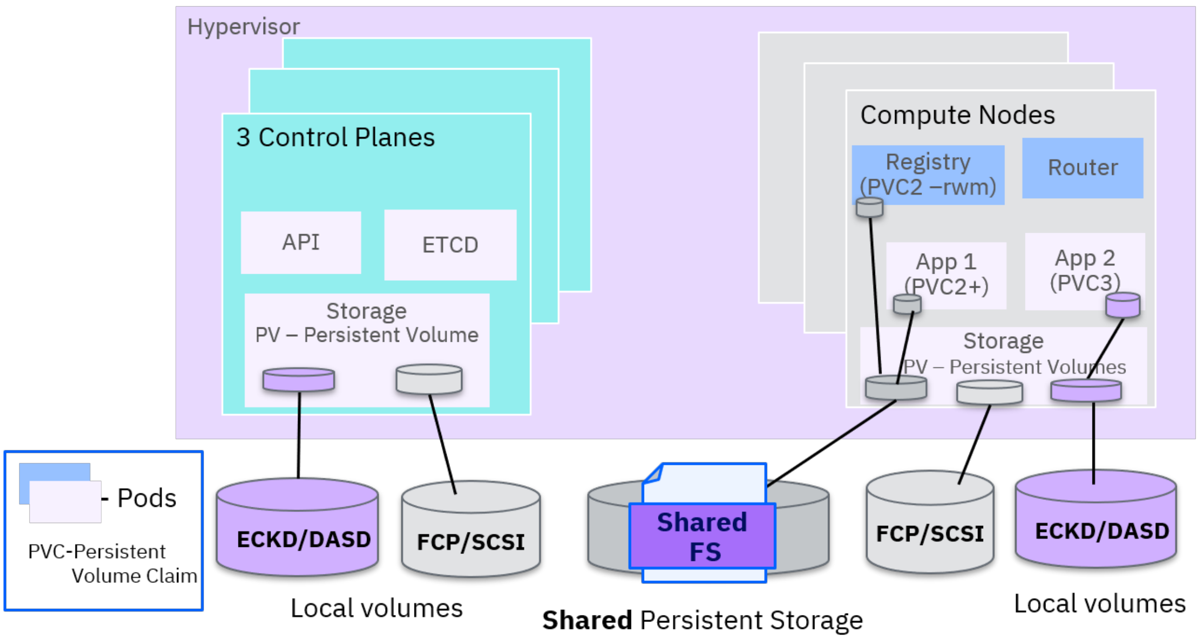


Figure 3.1-4: Topology with persistence storage

3.1.2 PoT and PoC Environments

For a Proof of Technology or Proof of Concept project, a RHOCP Cluster setup in its minimal configuration can be a 3 Node CLuster or as depicted in the following diagram, the resource requirements are thre same:

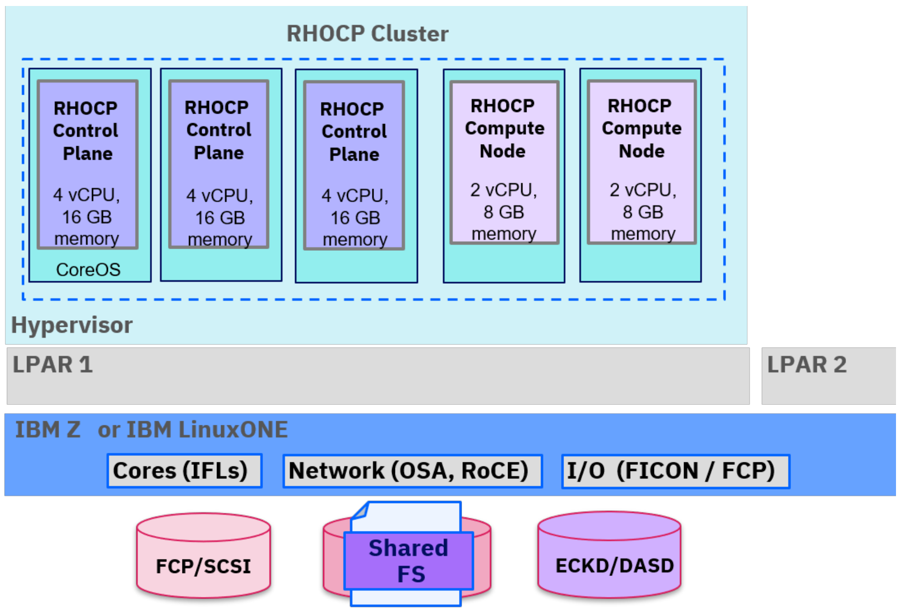


Figure 3.1-5: Minimum RHOCP Cluster setup for PoC

The minimum system requirements for z/VM guests of an RHOCP cluster are:

- Hypervisor
  - z/VM hypervisor 7.1
    - EAV Function - the Extended Address Volume (EAV) function is needed to support the required higher capacity DASDs

        - HyperPAV recommended - Parallel Access Volume PAV is highly recommended for disk performance
    - RHEL 8.3 KVM
        - Multipathing is recommended for local disk performance
- Control plane nodes
    - 4 vCPU (vIFLs)
    - 16 GB RAM
- Compute nodes
    - 2 vCPU (vIFLs)
    - 8 GB RAM (workload dependent)
- Networking options
    - OSA, RoCE
    - the internal Hipersockets network capabilities
    - in a z/VM based implementation, z/VM VSWITCH is an additional option
    - in a KVM based implementation, the BOND device enables additional bandwith and failover option
- Storage
    - RHOCP control plane, 120 GB each
    - RHOCP compute, 120 GB (workload dependent)
    - Clustered file system storage
        - NFS, greater than 100 GB
        - IBM Spectrum Scale via IBM Cloud Native Storage Access (CNSA)
    - Storage for monitoring/logging has to be added

For further information the following RHOCP resource is available Installing a cluster on IBM Z and LinuxONE.

NOTE Be aware that the PoC environment, in particular with the specified minimal hardware setup, is limited in terms of capacity and performance for CPU intense and network intense workload.

3.1.3 Production Environments

Due to a design with High Availability in mind by design, in a production environment a RHOCP cluster needs to be deployed with three control plane nodes or control planes spread across multiple LPARs in one or multiple physical machines.

RHOCP provides powerful orchestration capabilities for the container workload you are deploying. In order to make that happen seamlessly, it is essential to provide sufficient resources to your cluster. For production environments it is highly recommended to provide at least the preferred or recommended hardware and system requirements mentioned below.

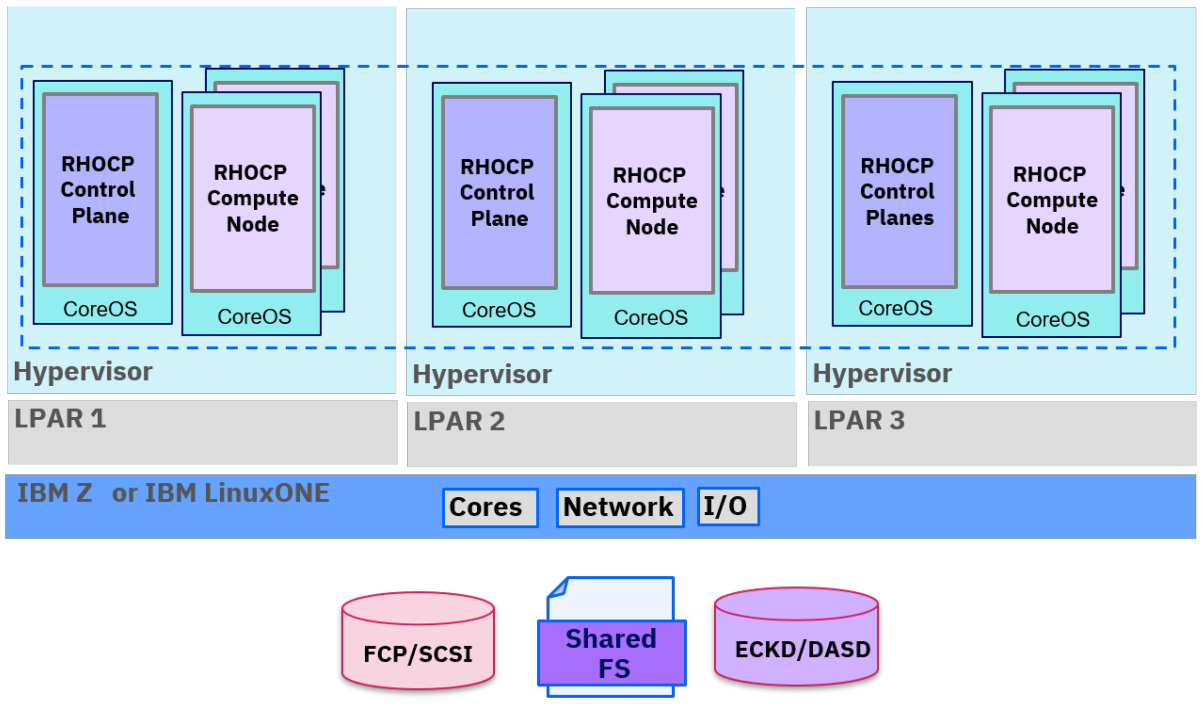


Figure 3.1-6: Production like RHOCP Cluster Setup

Hardware requirements

- 3 LPARs with a total of at least 6 IFLs that support SMT2
- OSA or RoCE network adapters in HA configuration
- HiperSockets, which are attached to a node
    - In a z/VM environment the HiperSockets network can use the z/VM VSWITCH bridge to enable the communication to a z/VM VSWITCH (see Chapter 4.2)
    - In a RHEL KVM environment you can use bridged MacVTap network to implement Hipersockets (see *3.5.5 Defining the MacVTap network* in http://www.redbooks.ibm.com/redbooks/pdfs/sg248463.pdf)

Operating system requirements

- Instances of hypervisors on each LPAR

On your hypervisor instances, set up:

- 3 guest virtual machines for RHOCP control plane machines, one per hypervisor
- At least 6 guest virtual machines for RHOCP compute machines, distributed across the hypervisor instances

Disk storage for the hypervisor guest virtual machines

- FICON attached disk storage (DASDs).
    - For z/VM hypervisor, these can be z/VM minidisks, fullpack minidisks, or dedicated DASDs. To reach the minimum required DASD size for Red Hat Enterprise Linux CoreOS (RHCOS)

installations, you need extended address volumes (EAVs) in z/VM. It is highly recommended to use Parallel Access Volume access (HyperPAV) and High Performance FICON (zHPF) to ensure optimal performance.

- FCP-attached disk storage
    - For FCP/SCSI attached storage it is highly recommended to use Multipathing

Storage / Main Memory

- 16 GB for RHOCP control plane machines
- 8 GB for RHOCP compute machines (workload dependent)
- 16 GB for the temporary RHOCP bootstrap machine

NOTE Please be aware that the PoC environment, in particular with the specified minimal hardware setup, is limited in terms of capacity and performance for CPU intense and network intense workload.

For latest and complete listing of prerequisites refer to https://docs.openshift.com/container-platform/latest/installing/installing_ibm_z/installing-ibm-z.html#preferred-ibm-z-system-requirements_installing-ibm-z

It is also strongly advisable to monitor production systems carefully in terms of performance and resources consumption and make sure that RHOCP does not encounter bottlenecks in available hardware.

To ensure appropriate sizing for production workloads refer to the documentation. https://docs.openshift.com/container-platform/latest/scalability_and_performance/recommended-host-practices.html

3.1.4 Considerations for Business continuity via High Availability and Disaster Recovery

In addition to the setup of a full production environment, it is strongly advisable to consider aspects of **high availability** and **disaster recovery**.

- High availability ensures availability through redundancy of resources to ensure continuous operation and make sure that workloads can be processed continuously without disruptions, with unplanned or planned outages, even when software is being restarted or updated (e.g. individual containers are being replaced with new versions of your applications). High availability includes applications (containers), middleware, operating system / virtualization / hypervisor, as well as infrastructure hardware availbility.
- Disaster recovery adds the need for a consideration that an entire datacenter is affected and cannot operate. This situation requires the redundancy of the entire environment in the disaster recovery datacenter to ensure business continuity. Typically this is implemented by a second and more often even a third data center.

The corresponding detailed design considerations are listed in a separate section of this document.

3.2 Integrating Red Hat OpenShift Container Platform in the enterprise

In a real customer environment RHOCP is not an isolated deployment. Integration and interaction with other workloads distributed across an IT landscape is therefore a key aspect of each RHOCP architecture. The various networking capabilities and options on IBM Z and IBM LinuxONE will be outlined in detail in this reference architecture document. For RHOCP environment, you require administrative access to the DNS configuration to configure the required domains used by the RHOCP clusters. A default network policy mode is implemented for each RHOCP internal SDN (Software Defined Network).

Key components of the networking integration concepts are:

- **Load balancer**: The load balancer ensures that the workload is spread across the available compute nodes. In PoC environments the load balancer can be provided as a software component within your RHOCP installation. For production purposes it is recommended to implement this as a dedicated component or hardware.
- **DNS**: As RHOCP has patterns on naming of servers and workload apps, the DNS is a key part of your deployment. You need to have administrative access to a DNS zone to manage the required RHOCP entries.
- **DHCP**: Depending whether the implementation uses dynamic IP addresses for the RHOCP nodes, it is required to have a DHCP service. If you consider using static IP addresses, it is not required to have or use a DHCP service.
- **Network security**: All network traffic can be secured and can be configured in RHOCP. Authorization can be realized via internal RHOCP security or external LDAP integrated security.
- **Network and storage**: Planning for increased network traffic to shared storage, has to be considered and eventually isolated from the application network traffic. To achieve this, you can consider using multi Nic (Network Interface) support for different networks in the RHOCP Nodes. Details are outlined in chapter 4.2 networking options.

NOTE You can have multiple RHOCP cluster and traditional workload in one physical machine and the external load balancers take care of workload distribution.

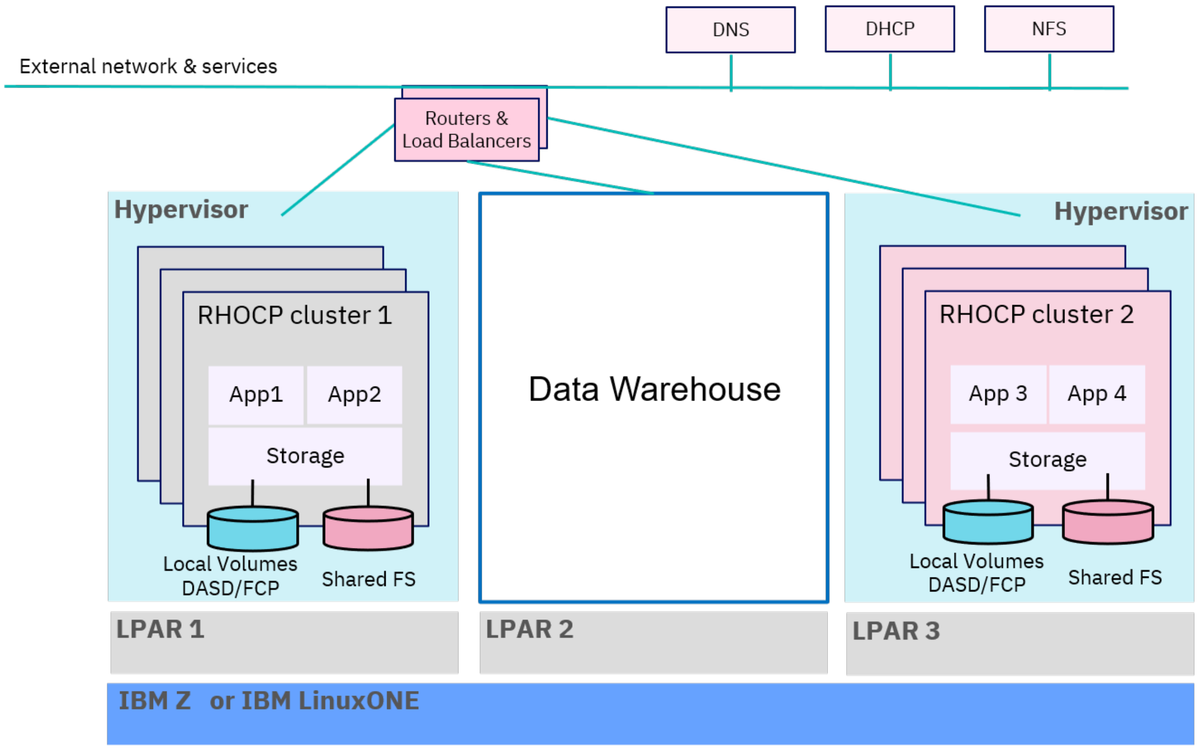


Figure 3.2-1: Reference Topology of network components, for RHOCP integration in the enterprise

The following chapter will detail the architectural design considerations for your solution based on RHOCP.

Chapter 4. Design considerations

This section describes in detail how this reference architecture addresses key design considerations. The recommendations in this reference architecture were developed in conjunction with Red Hat field consultants, engineers, and product teams.

4.1 Virtualization and hypervisors

RHOCP on IBM Z and IBM LinuxONE is deployed in a virtualized environment and offers different configuration options:

- Using z/VM as hypervisor

The most common virtualization of RHOCP on IBM Z and IBM LinuxONE takes advantage of the z/VM hypervisor. This virtualization offers the security certification EAL 4+, which is the highest level of certification and represents the highest level of isolation in virtualized environments. As z/VM is a well-established hypervisor for the mainframe, it is a proven technology and commonly used by many customers.

- Using a KVM-based deployment

An alternative approach for virtualization is a KVM-based deployment, using Red Hat Enterprise Linux KVM 8.3 or later. KVM is based on Open Source technology which implements components that turn a Linux system into an hypervisor.

4.1.1 Using z/VM

For more details on installing the z/VM hypervisor in an IBM Z and IBM LinuxONE environment, please refer to the documentation: https://www.vm.ibm.com/install/

Red Hat OpenShift Container Platform (RHOCP) can be installed in a single Logical Partition (LPAR) or on multiple LPARs. A virtualization deployment with z/VM in those LPARs requires planning from a capacity, security, and Service Level Agreement (SLA) perspective, regarding of the estimated workload that will run in each LPAR.

You can install z/VM in a cluster, called Single System Image (SSI) cluster, or in standalone mode. In an SSI cluster the nodes can be in one or multiple hardware machines. The advantage in SSI is that you can manage the hypervisor from each node and can use Life Guest Relocation (LGR) for planned outages or maintenance in some nodes.

The z/VM deployment consists of:

- Installing z/VM in the LPAR
- Defining the z/VM guest entries by creating z/VM directory entries, which will act as RHOCP Node configurations

The installation of z/VM is very well documented in the IBM Redbooks publication that contains a series of books that is called The Virtualization Cookbook for IBM Z Systems. These volumes cover the practical steps from planning, installation, and customization of the z/VM hypervisor.

It is recommended that you use Volume 1 of this series because IBM z/VM is the base "layer" when you install Red Hat OpenShift Container Platform on IBM Z and IBM LinuxONE. Volume 1 starts with an introduction, discusses planning, then describes z/VM installation, configuration, hardening, automation, and servicing. It adopts a cookbook format that provides a concise, repeatable set of procedures for installing and configuring z/VM by also using the Single System Image (SSI) clustering feature.

Volumes 2 can be of use if you want to install and customize your own Red Hat Enterprise Linux (RHEL) virtual servers on IBM Z and IBM LinuxONE hardware virtualized with IBM z/VM. The cookbook format describes the steps for installing and customizing Red Hat Enterprise Linux. RHEL servers can run side by side with RHOCP on the same machine.

You can find the redbooks under: http://www.redbooks.ibm.com/

The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM, SG24-8147-01

4.1.2 Using RHEL KVM

Red Hat OpenShift Container Platform (RHOCP) can be installed in a single Logical Partition (LPAR) or on multiple LPARs. The Virtualization with KVM in those LPARs requires planning from a capacity, security, and Service Level Agreement (SLA) perspective, in consideration of the estimated workload that will run in each LPAR.

You can install KVM as a cluster deployment, or in standalone mode. In an KVM cluster, the nodes can be in one or multiple hardware machines. The advantage in a cluster, is that you can manage the hypervisor from each node and can use live migration capabilities for planned outages or maintenance in some nodes.

The KVM-based deployment consists of:

- Installing KVM in the LPAR
- Defining the KVM guest entries, which will act as RHOCP node configurations

The installation of RHEL KVM is very well documented in the IBM Redbooks publication that contains a series of books called The Virtualization Cookbooks for IBM Z systems. These volumes cover the practical steps from planning, installation, and customization of the RHEL KVM hypervisor.

It is recommended that you use Volume 5 of this series for a KVM-based virtualization for building a Red Hat OpenShift Container Platform on IBM Z and IBM LinuxONE. Volume 5 covers introduction, planning, KVM installation, configuration, networking and life cycle. It adopts a cookbook format that provides a concise, repeatable set of procedures for installing and configuring KVM.

You can find the redbooks under: http://www.redbooks.ibm.com/

The Virtualization Cookbook for IBM Z Systems Volume 5: KVM, SG24-8463-00

4.2 Networking

In an RHOCP environment, planning of the network is very important given the fact that with IBM Z and IBM LinuxONE you have different choices and combinations for the RHOCP network. Each RHOCP node can have one or more network interfaces. The network design is dependent on the isolation requirements and integration architecture with other workloads.

Depending on the implementation and hypervisor, you can use the following network options:

- HiperSockets networks, which are internal IBM Z and LinuxONE networks. They do not require any network cards. You can define multiple instances in one machine.
- OSA (Open System Adapter) cards that can be shared and at the same time can be bounded for enhancing the network bandwidth and build an HA network.
- RoCE cards which are like the OSA cards, being able to be shared as well and run different protocols.
- Depending on the hypervisor you can use virtual switches in combination with the network capabilities of IBM Z and LinuxONE mentioned above.

A good reference for network configuration options can be found here:

https://developer.ibm.com/tutorials/understanding-network-definitions-for-openshift-4-on-ibm-z-and-linuxone/

4.2.1 RHOCP internal networking

RHOCP uses an internal SDN (Software Defined Network) to communicate between the nodes, manage the cluster and distribute workloads to the pods in the nodes. It can take advantage of the network capabilities in IBM Z and LinuxONE and includes an ingress load balancer. Included in RHOCP internal network capabilities is IPsec encryption support, which provides encryption of cross node pod-to-pod traffic on the cluster network. For secured communications, Openshift Service Mesh is enabling that transparently for the microservices and application interconnections. It can take advantage of the network capabilities and acceleration in IBM Z and LinuxONE and also includes an ingress load balancer. Let’s see the structure of these internal network capabilities.

- There are two options for the internal network in a RHOCP cluster for the default Container Network Interface (CNI) network provider, OpenShift SDN and OVN-Kubernetes. The newer one, the Open Virtual Network (OVN), is an Open vSwitch-based software-defined networking (SDN) solution. OVN enables to programmatically connect groups of guest instances into private L2 and L3 networks. The Cluster Network Operator deploys the OpenShift SDN default CNI network provider plug-in that you selected during cluster installation, by using a daemon set.

For more information, see About Red Hat OpenShift SDN.

- The next level of the internal network topology in a RHOCP cluster, is the pod interconnectivity in the node and accross the nodes. Each pod has one, but can have multiple network interfaces, which are connected to one or more network interfaces (NICs) of the node. To enable and manage multiple network interfaces in a pod, the Multus software component is available. Multus is the open source component that enables Kubernetes pods to attach to multiple networks.

For more information regardng Multus, see About Red Hat OpenShift Multus.

- The RHOCP internal ingress load balancer is controlled by the control plane nodes and is responsible for distributing the workload to the different compute nodes and routes, which represent the entry point to the various services running in different nodes and pods.

4.2.2 RHOCP load balancer

Per default RHOCP will establish an internal load balancer in each cluster. It is mandatory to have an external load balancer, especially during setup and installation of the RHOCP cluster. For each RHOCP cluster in general it is highly recommended to have at least one external load balancer that will distribute the workload among the API servers in the control plane nodes and the application requests running on the compute nodes.

External load balancers can be implemented in software or specialized hardware such as:

- Software load balancers:
  - HAProxy
  - NGINX
- Hardware load balancers:
  - F5
  - IBM Datapower
- Another option is to use DNS round-robin in absence of a load balancer, with a DNS server called *authoritative nameserver*.

RHOCP with its built-in DNS resolves the names of the services and associated routes to be reached via the service DNS as well as the service IP/port. For communications from outside the cluster RHOCP provides methods using an Ingress Controller with services and routes for apps running in different Pods in the cluster. For details of getting traffic into the cluster, refer to: https://docs.openshift.com/container-platform/latest/networking/configuring_ingress_cluster_traffic/configuring-ingress-cluster-traffic-ingress-controller.html

The Ingress Operator implements the ingresscontroller API and is the component responsible for enabling external access to Red Hat OpenShift Container Platform cluster services. The Ingress Operator makes this possible by deploying and managing one or more HAProxy-based Ingress Controllers to handle routing. You can use the Ingress Operator to route traffic by specifying Red Hat OpenShift Container Platform Routes and Kubernetes Ingress resources. Scale an Ingress Controller to meeting routing performance or availability requirements such as the requirement to increase throughput. https://docs.openshift.com/container-platform/latest/networking/ingress-operator.html#nw-ingress-controller-configuration_configuring-ingress

For more details see:

https://docs.openshift.com/container-platform/latest/networking/ingress-operator.html

4.2.3 Request flow in RHOCP through DNS and load balancers

Essential paths of the information flow include:

- Incoming request through an external load balancer passes the request:
  - To a control plane node, if it is a command for an API server port
  - To a compute node with a route pod, if it is a request for an application port
  - The route pod uses the HAProxy based Ingres Controller to route between pods
- The built-in internal DNS, converts the services by name
- Compute plane forwards DNS queries to internal services via Ingress Controllers and routes
- The Software Defined Networking (SDN) in a RHOCP cluster network enables pod-to-pod communication
- RHOCP follows the Kubernetes Container Networking Interface (CNI) plug-in model

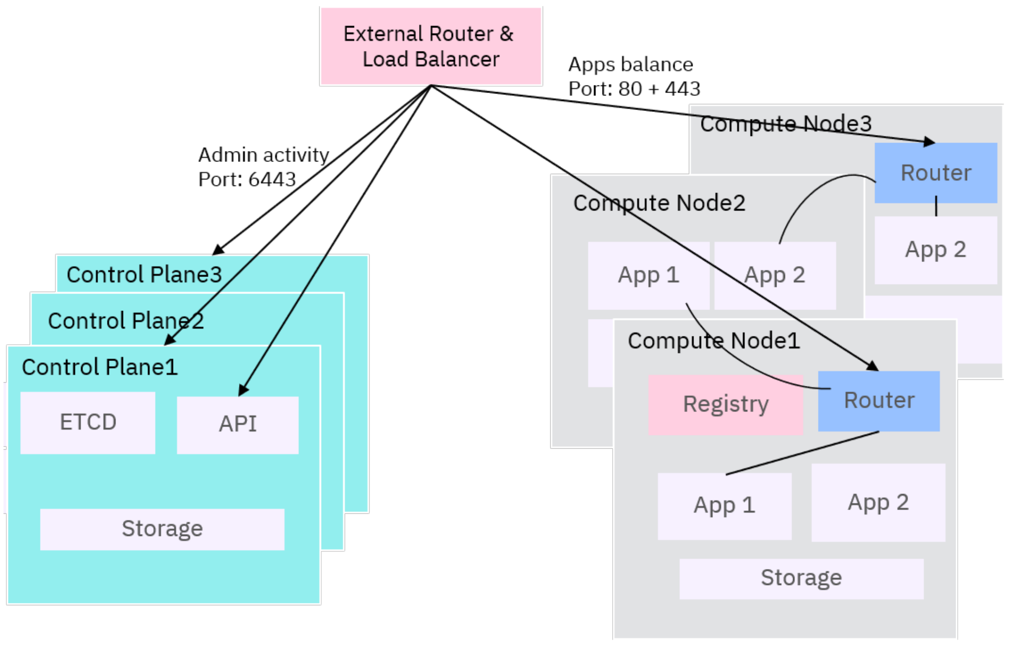


Figure 4.2-3-1: Load balancer and routing

Additional references:

https://docs.openshift.com/container-platform/latest/installing/installing_bare_metal/installing-bare-metal-network-customizations.html#installation-network-user-infra_installing-bare-metal-network-customizations

4.2.4 Networking options with Red Hat OpenShift Container Platform

For attaching RHOCP to networking interfaces, you have different options. Depending on the use case, each option has different advantages. It is important to classify the workload and estimate if RHOCP has many incoming requests from the network, or rather has high communication demand with an external database. Based on the workload you can chose or combine different network topologies for the best performance.

- Connect control plane and compute nodes using one or multiple virtual switches. Connect to the external load balancer for outside communication using OSA cards.
    - Benefit: Virtual switches allow for very fast changes of the RHOCP node configuration and capability to extend a cluster with additional nodes.

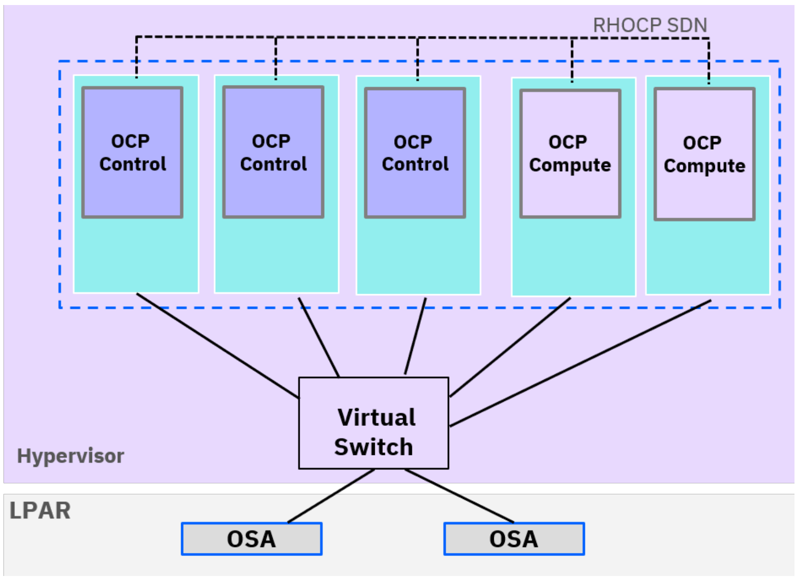


Figure 4.2.4-1: Network Topology Options with a Virtual Switch

- Connect control plane and compute nodes as well as load balancer using OSA or RoCE cards.
    - Benefit: Direct attached OSA or RoCE cards provide very fast communication and ensure high availability and enhanced bandwidth.

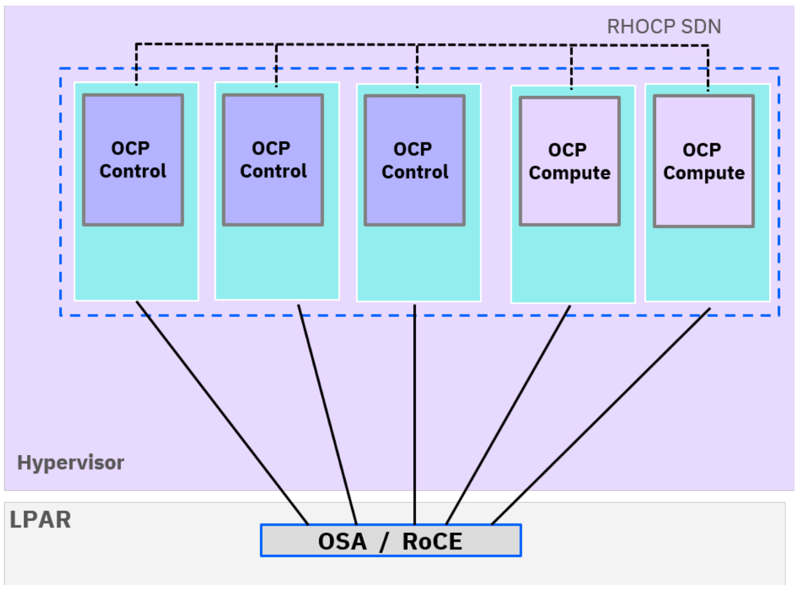


Figure 4.2.4-2: Network Topology Options With OSA

- Make use of multi-NIC support to connect the RHOCP nodes to multiple, different networks. You can plan and define multiple network interfaces for the nodes during installation time or after the installation. For the installation process, the number of network interfaces per node is just limited by the length of the parmline (896 bytes).
    - Benefit: Network traffic separation and isolation for different types of workloads as well as

network isolation or balancing purposes.

- If you want to add an additional network interface to a node in a RHOCP cluster after the installation, you can use the day-two operation capability with the nmstate-operator and via a single .yaml config file define additional network interfaces for several control or compute nodes. The resulted configuration persists after node reboot and these interfaces will be integrated in the communication with the API server in the control planes for monitoring their healthy.

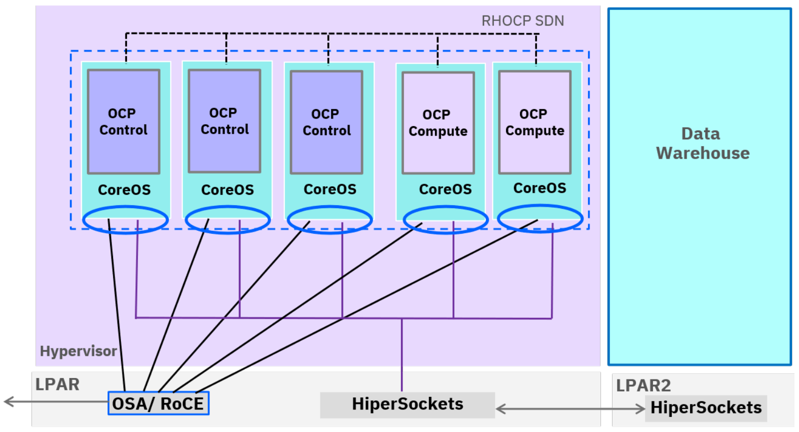


Figure 4.2.4-3: Network Topology Options with multi-NIC

4.2.5. Networking in a z/VM hypervisor based implementation

In an environment using the z/VM hypervisor the z/VM VSWITCH can be used to build virtual networks in the hypervisor, connected to an OSA card for external communication or even connected to HiperSockets using the HiperSockets Bridge feature.

- Connect control plane and compute nodes using the z/VM VSWITCH and then connect them with the load balancer via an OSA card.
    - Benefit: Increased flexibility when adding and extending RHOCP Nodes along with internal network isolation by using multiple VSWITCHES.

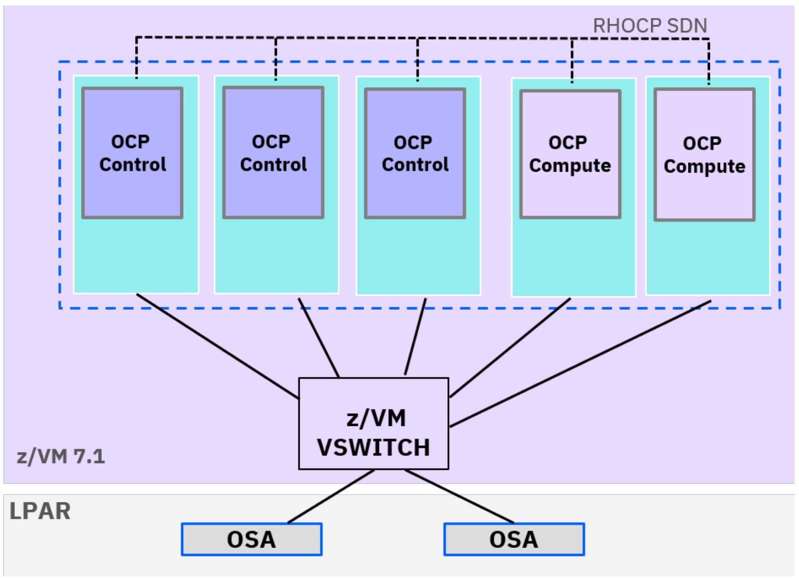


Figure 4.2.5-1: Network Topology Options with z/VM VSWITCH

- Connect control plane and compute nodes in a HiperSockets network and the HiperSockets Bridge to connect with the z/VM VSWITCH that provides a communication path to the external network via an OSA card.
    - Benefit: Added flexibility in a failover and relocation scenario when some nodes are relocated to another physical machine.

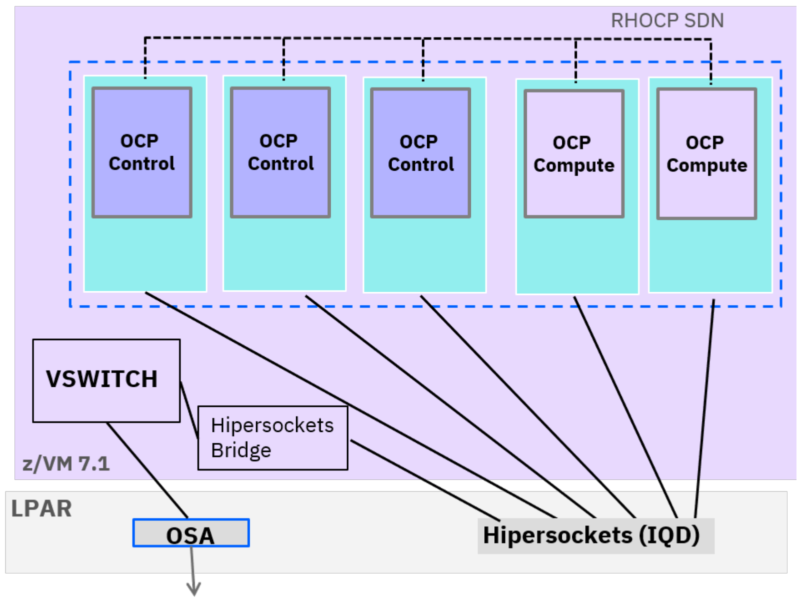


Figure 4.2.5-2: z/VM VSWITCH Network Topology Option with HiperSockets

4.2.6. Networking in a KVM hypervisor-based implementation

In an environment using the KVM hypervisor the use of MacVTap enables the build of virtual bridged networks in the hypervisor.

You have different options to build the network topology with RHOCP:

- connect to an OSA or RoCE card for external communication as shown in **Figure 4.2.6-1:** Network Topology Options With OSA / RoCE
- Make use of network channel bonding support to connect the nodes to multiple network cards.
    - Benefit: enhanced network bandwidth and balancing purposes.

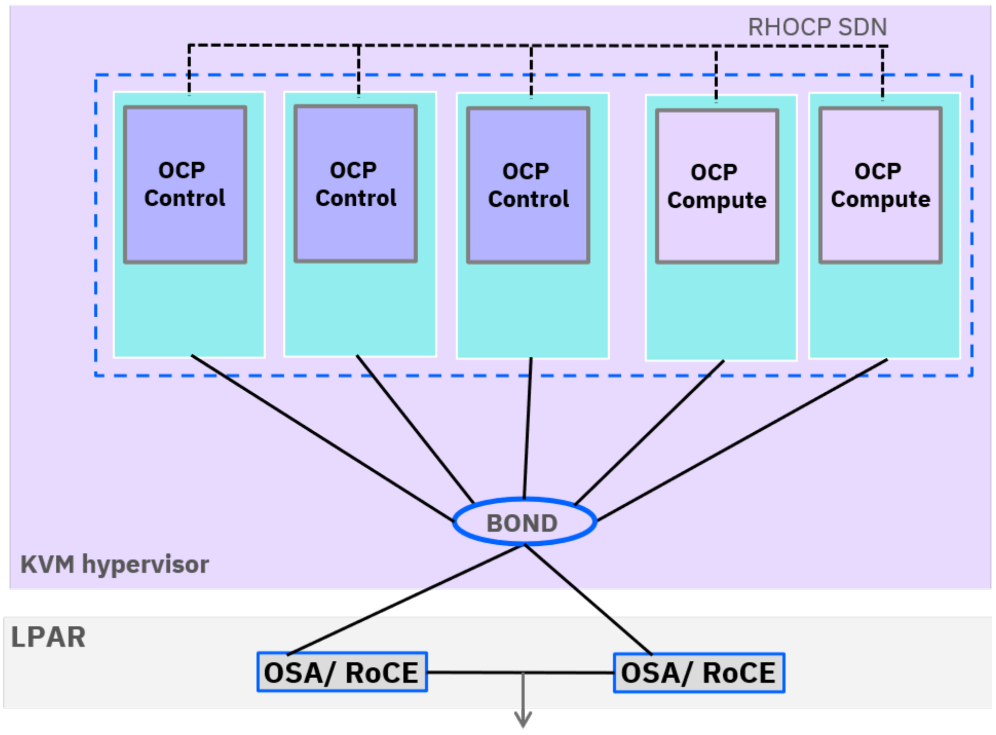


Figure 4.2.6-2: Network Topology Options with bonding

- Make use of the Linux bridging network capabilities to connect the nodes for example to a Hipersockets network
  - Benefit: internal network without external routers and switches

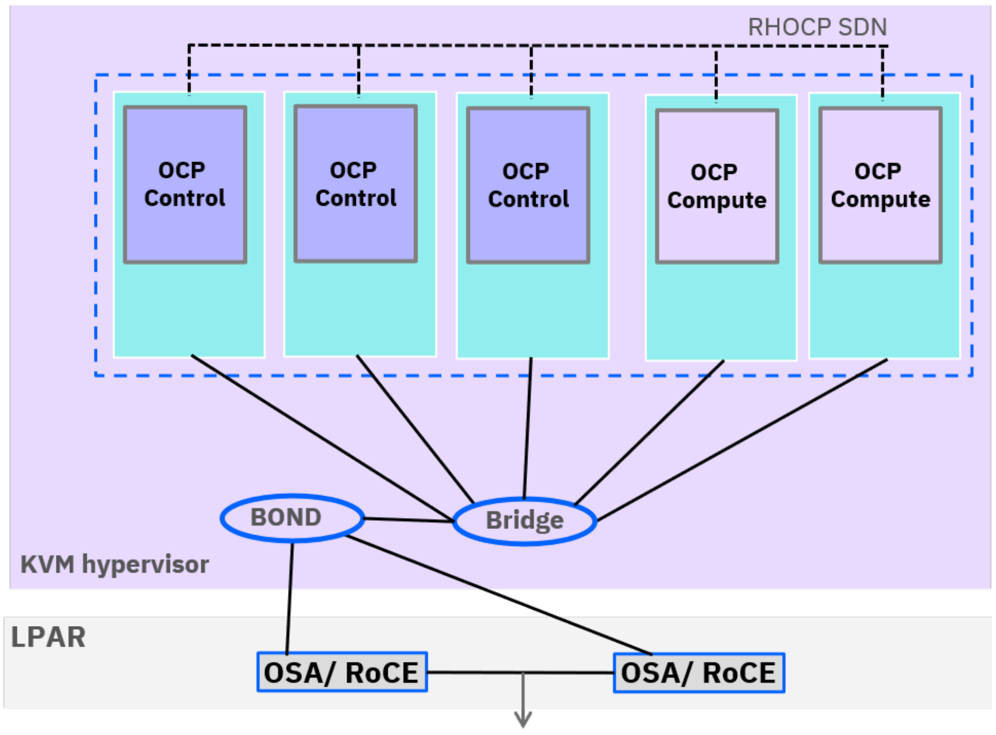


Figure 4.2.6-3: Network Topology Options with bridging

4.2.7 Server Time Protocol

Server Time Protocol (STP) is a server-wide facility that is implemented in the Licensed Internal

Code (LIC) of the IBM® Z systems. The STP functionality was introduced in a previous generation of IBM Z systems and it provides improved time synchronization in a sysplex or non-sysplex configuration.

In z/Architecture, the Store Clock (STCK) and the Store Clock Extended (STCKE) instructions provide a means by which programs can both establish time-of-day and unambiguously determine the ordering of serialized events, such as updates to a database, a log file, or another data structure. The architecture requires that the TOD clock resolution be sufficient to ensure that every value stored by a STCK or STCKE instruction is unique. Consecutive STCK or STCKE instructions executed, possibly on different CPUs in the same server, must always produce increasing values. Thus, the time stamps can be used to reconstruct, recover, or in many different ways ensure the ordering of these serialized updates to shared data.

STP is a message-based protocol, like the industry standard Network Time Protocol (NTP). STP allows a collection of IBM Z servers to maintain time synchronization with each other using a time value known as Coordinated Server Time (CST). The network of servers is known as a Coordinated Timing Network (CTN). The mainframe's Hardware Management Console (HMC) plays a critical role with STP CTNs: The HMC can initialize Coordinated Server Time (CST) manually or initialize CST to an external time source. The HMC also sets the time zone, daylight savings time, and leap seconds offsets. It also performs the time adjustments when needed.

For a RHOCP cluster running on a IBM Z or LinuxONE the time synchronization configuration can be skipped because all the VMs are running on the same IBM Z system and as mentioned previously are managed by the STP. To get more information about the IBM Z Server Time Protocol, refer to the IBM Z Server Time Protocol Guide RedBook.

4.2.8 Summary of the Options

- Options for load balancer

As external load balancer you can use specialized hardware like F5 or IBM Datapower, or software based load balancers like NGINX or HAProxy.

- Options for Networking

Options	suitable usage scenario
HiperSockets	network in the box, internal communication
z/VM VSWITCH	high flexibility and SDN
OSA	for fast communication inside-out, VSWITCH compatible
RoCE	for fast communication inside-out
MacVTab	network flexibility in a KVM environment

Networking in Red Hat OpenShift Container Platform (RHOCP) is a topic that requires good planning. For additional information refer to the links below:

Red Hat OpenShift Container Platform - Networking

4.3 DNS

The RHOCP Version 4 installation program greatly simplifies the DNS requirements seen in previous RHOCP installations. All internal DNS resolution for the cluster, certificate validation, and bootstrapping is provided through a self-hosted, installer-controlled solution that uses the mDNS plugin for CoreDNS. This solution was developed to perform DNS lookups based on discoverable information from mDNS. This plugin will resolve both the etcd-NNN records, as well as the `_etcd-server-ssl._tcp.SRV` record. It is also able to resolve the name of the nodes.

With this solution, you do not need to add the IP addresses of the control plane nodes, compute nodes, and the bootstrap node, either manually or dynamically, to any form of public DNS. The RHOCP installation is entirely self-contained in this respect.

This reference architecture considers the use of the RHOCP installation program parameter `externalDNS` to allow the installer-built subnets to offer external DNS resolution to their instances.

4.3.1 DNS setup

The RHOCP on z/VM deployment has the following DNS requirements, both before and after installation:

- The installation host must be able to resolve the RHOCP API address.
- The bootstrap node must be able to resolve external, public domains.

More details can be found in the Installation Redpiece: http://www.redbooks.ibm.com/abstracts/redp5605.html

4.3.1.1 RHOCP API DNS

The RHOCP API floating IP address needs to be in place for installing the cluster, and to ensure the `api.<cluster name>.<base domain>.` address space resolves to it.

4.3.1.2 Application DNS

A wildcard entry is required in your DNS for this IP to resolve the following naming structure:

`*.apps.<cluster name>.<base domain>.`

4.3.1.3 Bootstrap node

The bootstrap node must be able to resolve external domain names directly. The RHOCP 4 installation program uses this name resolution to connect externally and retrieve the containers required to stand up the bootstrap cluster used to instantiate the production cluster. No other RHOCP nodes require this external resolution.

4.4 Storage

Many containerized workloads, including several core RHOCP services, require a persistent storage solution to maintain data persistence when pods are deleted or restarted. The variety of storage types, like file, block, and object storage enable options that you need to consider depending on the implementation and workload.

RHOCP uses the Kubernetes Persistent Volume (PV) framework to allow cluster administrators to provision persistent storage for a cluster. The Kubernetes persistent volume (PV) framework allows RHOCP users to request persistent volumes without any knowledge of the underlying storage. The users make use of Persistent Volume Claims (PVCs) to request PV resources, without having specific knowledge of the underlying physical storage infrastructure. PV resources on their own are not scoped to any single project. They can be shared across the entire RHOCP cluster and claimed from any project. After a PV is bound to a PVC, that PV cannot be bound to any additional PVCs. Consider that PVs represent a piece of existing storage in the cluster that was either statically provisioned by the cluster administrator or dynamically provisioned.

Storage options include

- Block Storage (used to store data files on Storage Area Networks (SANs) or cloud-based storage environments)
- File Storage (as file-level or file-based storage, is normally associated with Network Attached Storage (NAS) technology)
- Object Storage (breaks data files up into pieces called objects and stores those objects in a single repository, which can be spread out across multiple networked systems)

Differentiators are the supported attachment modes and data handling characteristics: ReadWriteOnce, ReadOnlyMany, ReadWriteMany, which have influence on the requirements for shared or non-shared data across RHOCP nodes.

For many applications, non-shared storage options are perfectly suitable and sufficient. Nevertheless, for some use cases you will require shared storage. One of those use cases is for databases running in RHOCP with entities in different nodes, accessing the same data.

The RHOCP internal container image registry requires ReadWriteMany type of storage and therefore shared storage is required as well.

Persistent Volumes can be made available to RHOCP using different attachment type plg.ins or Operators. The most relevant for an environment on IBM Z and IBM LinuxONE are:

- Local volume
- iSCSI
- HostPath
- Network File System (NFS)
- IBM Cloud Native Storage Access (CNSA) - a cross cluster mount capability to attach a RHOCP cluster to an IBM Spectrum Scale cluster
- Red Hat OpenShift Data Foundation (formerly known as OpenShift Container Storage)

- Local Storage Operator
- Container Storage Interface (CSI) attached Storage

More details can be found in the following sections as well as the product documentation:

https://docs.openshift.com/container-platform/latest/storage/understanding-persistent-storage.html

4.4.1 Local Storage Operator

The Local Storage Operator is not installed in Red Hat OpenShift Container Platform by default. You need to install and configure this operator to enable local volumes in your cluster. This also means during installation of RHOCP, you cannot make use of the Local Storage Operator. Details can be found in the product documentation:

https://docs.openshift.com/container-platform/latest/storage/persistent_storage/persistent-storage-local.html

Depending on the used Hypervisor, the architecture of leveraging local storage using the local storage operator can slightly vary. The following two images show the detailed components of the stack for KVM and z/VM.

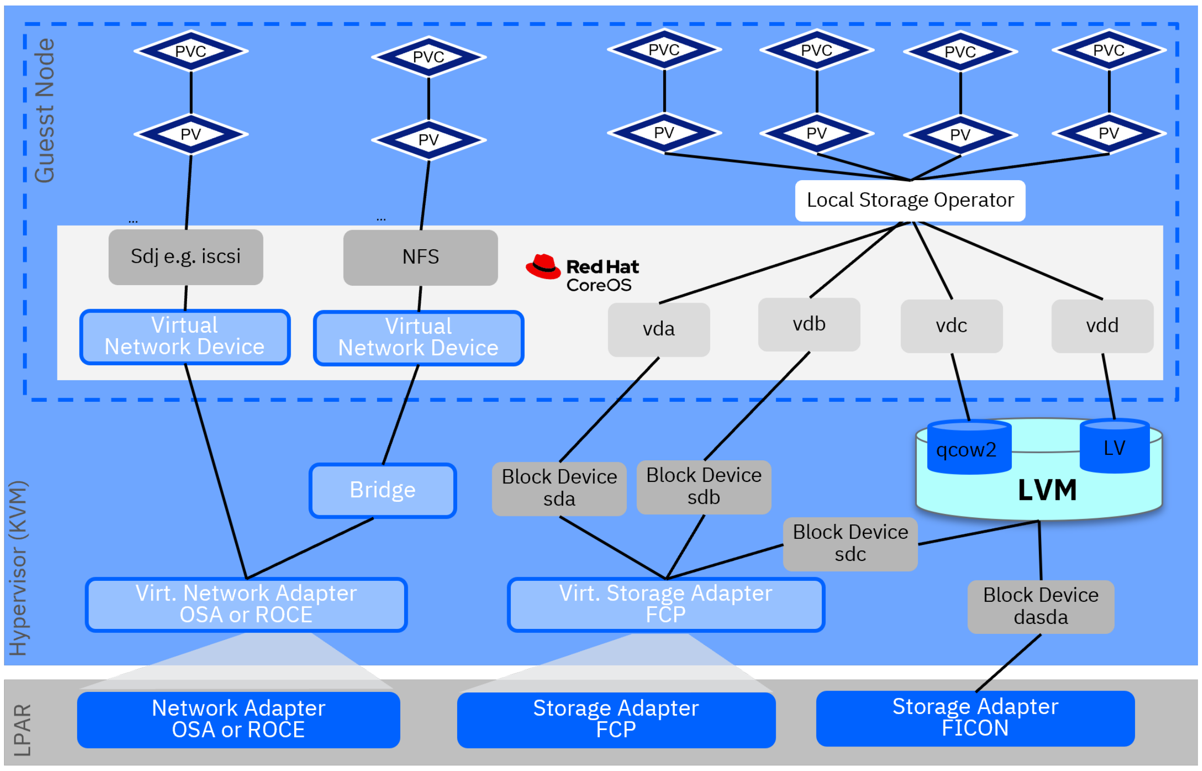


Figure 4.4-1: Storage and Persistence in RHOCP based on KVM

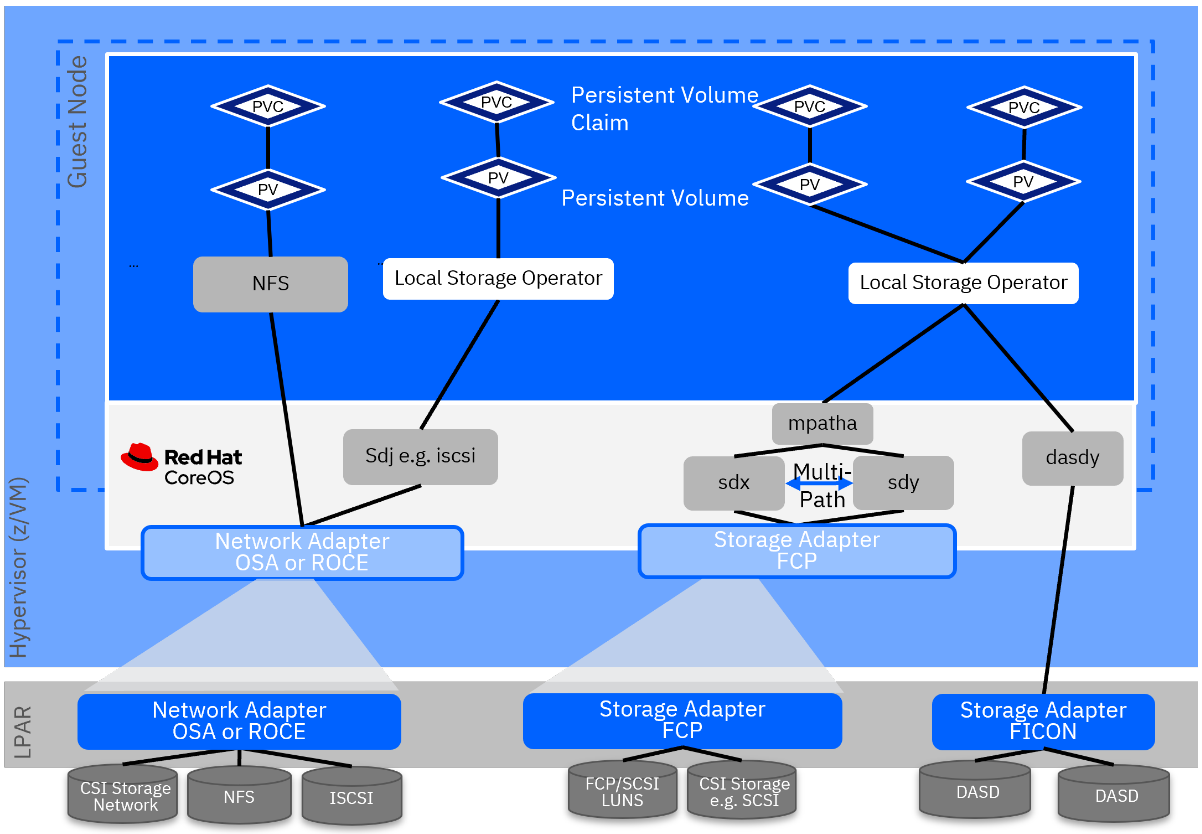


Figure 4.4-2: Storage and Persistence in RHOCP based on z/VM

4.4.2 Local Storage Volume

Red Hat OpenShift Container Platform can be provisioned with persistent storage by using local volumes using the Local Storage Operator.

This is the easiest way to provide disks to the control plane nodes as there is no additional configuration of RHOCP. All the instance volumes are stored directly on the disk local to the Compute service.

Local persistent volumes (PV) allow you to access local storage devices, by using the standard PVC (Persistent Volume Claim) interface.

Local volumes can be used without manually scheduling pods to nodes, because the system is aware of the PV volume.

Local volumes are still subject to the availability of the underlying node and are not suitable for all applications.

Local volumes can only be used as a statically created Persistent Volume.

For many applications, local storage options are perfectly suitable and sufficient. Nevertheless, for some use cases you will require shared storage. One of those use cases is setting up the RHOCP image registry. A typical choice for providing storage for RHOCP image registry is to use NFS.

Prerequisites:

- The Local Storage Operator is installed.

- Local disks are attached to the Red Hat OpenShift Container Platform nodes.

4.4.3 HostPath

A hostPathvolume in an Red Hat OpenShift Container Platform cluster: mounts a file or directory from the host node's filesystem into your pod. Most pods will not need a hostPathvolume, but it does offer a quick option for testing should an application require it.

The cluster administrator must configure pods to run as privileged. This grants access to pods running on the same node. Red Hat OpenShift Container Platform supports hostPath mounting for development and testing on a single-node cluster.

A host path volume is provisioned statically.

In a production cluster, you would not use hostPath.

4.4.4 NFS

Red Hat OpenShift Container Platform clusters can be provisioned with persistent shared storage using shared storage in NFS servers. Persistent Volumes (PVs) and Persistent Volume Claims (PVCs) provide a convenient method for sharing a volume across a project, because the NFS attached PV is of type RedWriteMany.

While the NFS-specific information contained in a PV definition could also be defined directly in a Pod definition, doing so does not create the volume as a distinct cluster resource, making the volume more susceptible to conflicts.

Provisioning

Storage must exist in the underlying infrastructure before it can be mounted as a volume in Red Hat OpenShift Container Platform.

Requirements to provision NFS volumes:

- NFS servers with storage paths defined
- Export the paths

Further documentation on the use of NFS as persistent storage can be found in the Chapter 5.2: Adding Persistent Storage with NFS.

- RHOCP - Configuring the registry for bare metal: https://docs.openshift.com/container-platform/latest/registry/configuring_registry_storage/configuring-registry-storage-baremetal.html
- Adding Persistent Storage with NFS: https://www.linkedin.com/pulse/adding-persistent-storage-your-openshift-image-registry-miranda/
- Adding Dynamic Persistent Storage with NFS: https://www.linkedin.com/pulse/persistent-storage-part-ii-dynamic-nfs-provisioning-43-filipe-miranda/

4.4.5 Container Storage Interface (CSI)

The Container Storage Interface (CSI) allows Red Hat OpenShift Container Platform to consume storage from storage back ends that implement the CSI interface as persistent storage. Storage Vendors prove those CSI drivers for their own storage solutions.

- IBM provides CSI drivers to attach enterprise storage:
  - IBM block storage CSI driver, supporting IBM Spectrum Virtualize and IBM Spectrum Accelerate product portfolios
  - IBM Spectrum Scale CSI driver, supporting IBM Spectrum Scale and IBM Elastic Storage® Server

https://www.ibm.com/support/knowledgecenter/SSRQ8T/landing/IBM_block_storage_CSI_driver_welcome_page.html

4.4.6 Red Hat OpenShift Data Foundation

Container workloads are stateless by definition. This fundamental assumption makes their life cycle so easy and efficient to orchestrate. A container is easily portable and can be scaled up and down at ease. A container can be restarted and be moved between servers at any time. But this implies, that any kind of permanent persistence needs to be provided separately.

Adding persistence to containers to make them stateful is exactly the focus of Red Hat® OpenShift® Data Foundation, formerly known as Red Hat OpenShift Container Storage.

OpenShift Data Foundation provides software-defined storage for containerized applications. It manages storage resources for the application containers to work with and ensures that persistence is strictly decoupled from the orchestration of containers. Engineered as the data and storage services platform for RHOCP, OpenShift Data Foundation helps teams develop and deploy applications quickly and efficiently across hybrid cloud and multicloud container deployments.

Based on OpenShift Container Platform and OpenShift Data Foundation, an enterprise can achieve a unified access to data services across different applications and infrastructure environments. This includes:

- Federate underlying storage infrastructure to single abstracted repository
- Separate storage provisioning from storage consumption
- Deal with different kinds of data and storage classes

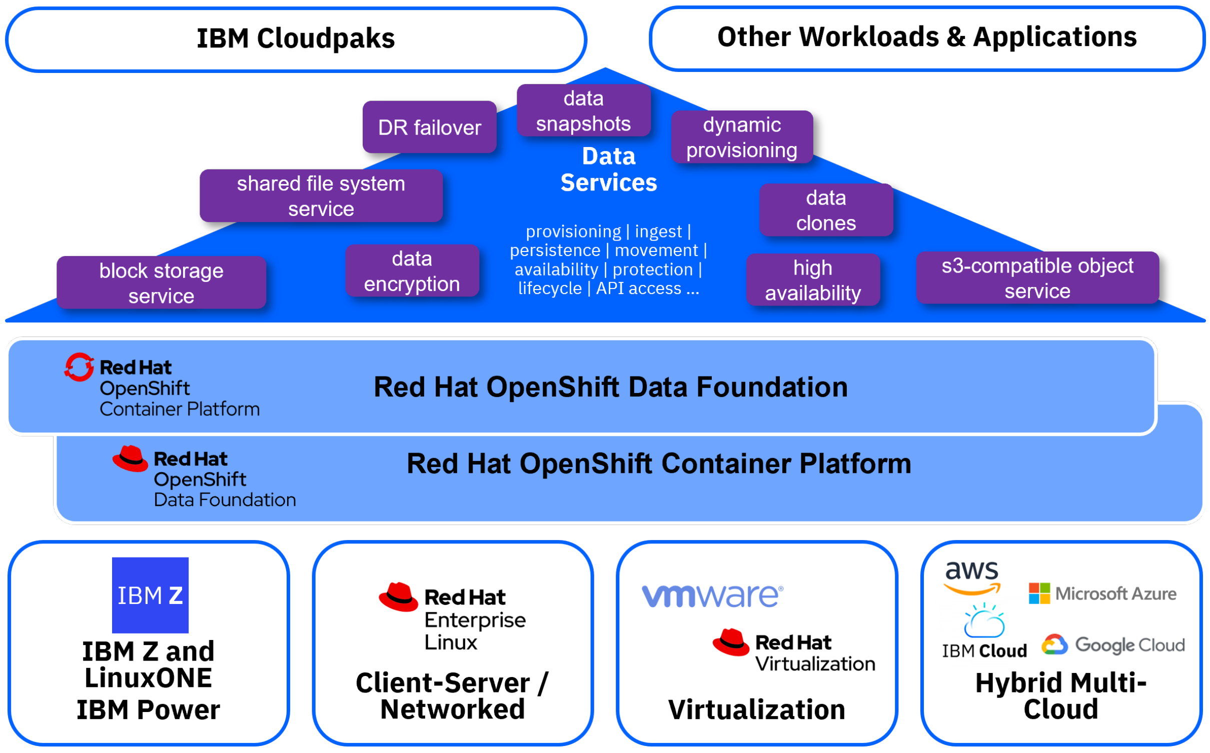

IBM Cloudpaks
Other Workloads & Applications
DR failover
data snapshots
dynamic provisioning
Data Services
shared file system service
data clones
provisioning | ingest | persistence | movement | availability | protection | lifecycle | API access ...
block storage service
data encryption
high availability
s3-compatible object service
Red Hat OpenShift Container Platform
Red Hat OpenShift Data Foundation
Red Hat OpenShift Data Foundation
Red Hat OpenShift Container Platform
IBM Z
IBM Z and LinuxONE IBM Power
Red Hat Enterprise Linux
Client-Server / Networked
vmware
Red Hat Virtualization
Virtualization
aws
Microsoft Azure
IBM Cloud
Google Cloud
Hybrid Multi-Cloud

Figure 4.4-3: Enterprise-ready approach to provide and scale storage

NOTE The **documentation** for OpenShift Data Foundation on IBM Z and LinuxONE can be found here: Red Hat OpenShift Data Foundation Documentation

NOTE A separate **reference architecture** document describing Red Hat OpenShift Data Foundation in more detail can be downloaded here.

4.4.6.1 Key components of OpenShift Data Foundation

Red Hat OpenShift Data Foundation is based on a technology stack including Rook, Ceph®, and NooBaa.

- Ceph is the core storage platform of OpenShift Data Foundation. Ceph is based on RADOS (Reliable Autonomic Distributed Object Store), which by itself is an open-source object storage service and integral part of Ceph distributed storage system.
- Rook is a storage orchestrator for Kubernetes and coordinates the services provided by OpenShift Data Foundation. Rook has operators to support different storage backends.
- NooBaa (Multi Cloud Object Gateway) provides consistent S3 endpoints across different backend infrastructure; like AWS, Azure, GCP, Bare Metal, VMware, or OpenStack.

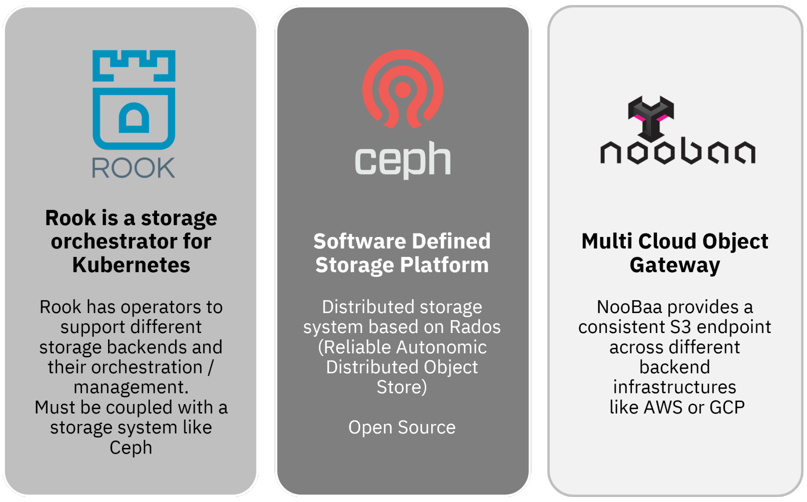

ROOK
Rook is a storage orchestrator for Kubernetes
Rook has operators to support different storage backends and their orchestration / management.
Must be coupled with a storage system like Ceph
ceph
Software Defined Storage Platform
Distributed storage system based on Rados (Reliable Autonomic Distributed Object Store)
Open Source
noobaa
Multi Cloud Object Gateway
NooBaa provides a consistent S3 endpoint across different backend infrastructures like AWS or GCP

Figure 4.4-4: Key Technologies used by OpenShift Data Foundation

OpenShift Data Foundation is deployed inside RHOCP using operators. In conjunction with the Local Storage Operator and the Rook-Ceph operator the installation creates a containerized Red Hat Ceph Storage cluster running on RHOCP.

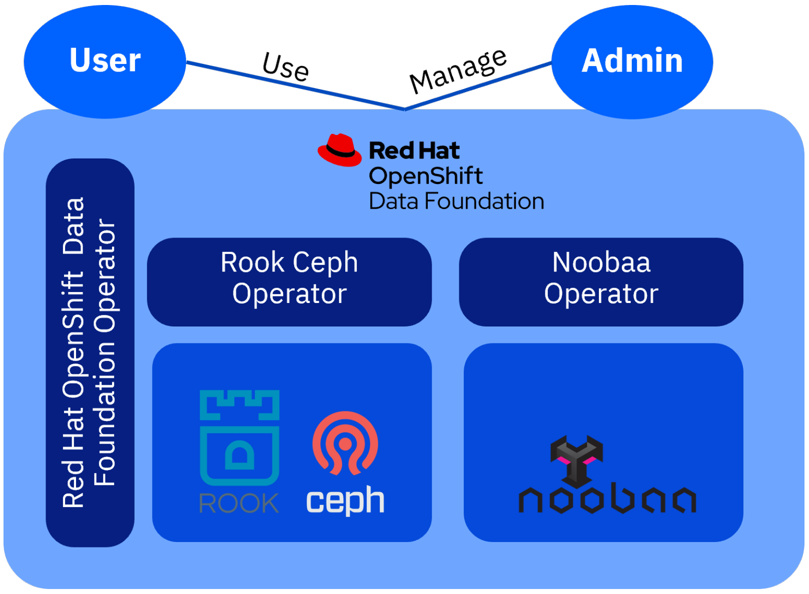

User
Use
Manage
Admin
Red Hat OpenShift Data Foundation
Red Hat OpenShift Data Foundation Operator
Rook Ceph Operator
Noobaa Operator
ROOK
ceph
noobaa

Figure 4.4-5: OpenShift Data Foundation related Operators

4.4.6.2 Storage classes

OpenShift Data Foundation supports a variety of storage types; like file, block, and object storage, which gives developers flexibility when implementating workload. Differentiators are the supported attachment modes and data handling characteristics:

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)

Those storage classes have influence on the requirements for shared or non-shared data across RHOCP nodes.

- Block Storage: Mainly appropriate for RWO access modes. However, RWX might be appropriate if the application can maintain data consistency and integrity. Suitable for databases and systems of record.
- File Storage (Shared and distributed file system): Appropriate for both RWO and RWX access modes, as the underlying file system is designed for multiple threads and multi-tenancy. Suitable for messaging, data aggregation, workloads machine learning, and deep learning.
- Multicloud object storage accessed via a lightweight S3 API endpoint. Appropriate for retrieval of data from multiple cloud object stores. Suitable for images, non-binary files, documents, snapshots, or backups.

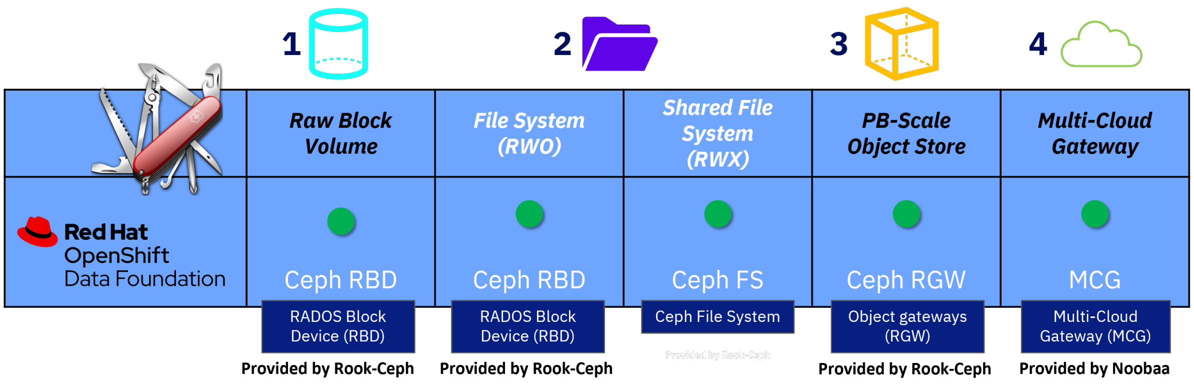


Figure 4.4-6: Storage Classes supported by OpenShift Data Foundation

4.4.6.3 Encryption

Red Hat OpenShift Data Foundation allows encryption of the stored data (data at rest) using the common Linux Unified Key Setup (LUKS2). Two levels of granularity are possible:

- OSD level: an entire storage device is encrypted
- PV level: a specific persistent volume (used by a application) is encrypted individually

For an **OSD level encryption**, the keys used for the encryption process can be stored either internally or externally.

- Internal key management is done within the OpenShift cluster. For this purpose the key is stored as a name-value-pair inside the OpenShift etcd database.
- External key management can be provided by a 3rd party key store, such as Hashicorp vault.

An external key management solution is preferred because it ensures a clear separation between the keys and the protected data.

For a **PV level encryption**, external key management is mandatory. In this case it is not possible to store keys internally inside the etcd database. The PV level encryption allows scoping the encryption process to specific persistent volumes used by an application. This allows an instance to protect one application (or tenant) from another. OpenShift Data Foundation has solved the separation between the persistent volumes, by creating a set of storage classes for each PV/tenant and assigning keys specifically to a storage class.

NOTE When storing keys internally, it is important to secure and backup the etcd database correctly!

4.4.6.4 Deployment topologies

OpenShift Data Foundation is can be deployed in different modes:

- **Internal mode**: OpenShift Data Foundation is installed within a single RHOCP cluster. Highly scalable, enterprise grade storage, which is fully integrated into the RHOCP lifecycle, monitoring, and management.
    - Application pods and OpenShift Data Foundation pods can be scheduled on the **same nodes** (compute nodes). In this case, compute and storage infrastructure scale together within the same cluster. This setup is optimized for simplicity of management.
    - As an alternative, application pods and OpenShift Data Foundation pods can be scheduled on **different nodes** (e.g. infrastructure nodes). This implies that compute hosts and storage hosts scale independently and a more balanced deployment can be achieved.
- **External mode**: Application pods run on one or more OpenShift clusters, while OpenShift Data Foundation storage is provided from an external Red Hat Ceph Storage cluster. This implies, that lifecycle, monitoring, and management of RHOCP and OpenShift Data Foundation are decoupled. Compute and storage infrastructure scales independently in different clusters. Optimized for scale and performance (on-premises only). Typically the external OpenShift Data Foundation is deployed as a Ceph storage environment on x86 bare metal hardware.

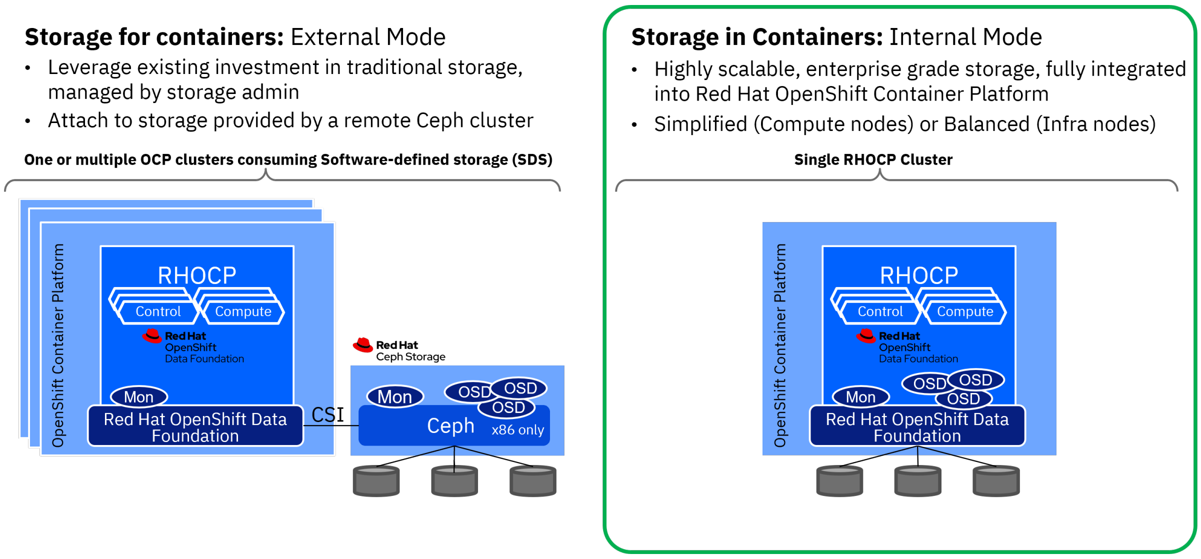

Storage for containers: External Mode
Leverage existing investment in traditional storage, managed by storage admin
Attach to storage provided by a remote Ceph cluster
One or multiple OCP clusters consuming Software-defined storage (SDS)
OpenShift Container Platform
RHOCP
Control
Compute
Red Hat OpenShift Data Foundation
Mon
Red Hat OpenShift Data Foundation
CSI
Red Hat Ceph Storage
Mon
OSD
OSD
OSD
Ceph
x86 only
Storage in Containers: Internal Mode
Highly scalable, enterprise grade storage, fully integrated into Red Hat OpenShift Container Platform
Simplified (Compute nodes) or Balanced (Infra nodes)
Single RHOCP Cluster
OpenShift Container Platform
RHOCP
Control
Compute
Red Hat OpenShift Data Foundation
Mon
OSD
OSD
OSD
Red Hat OpenShift Data Foundation

Figure 4.4-7: Internal and external Mode

4.4.6.5 Comparing OpenShift Data Foundation with IBM Spectrum Scale and NFS

Table 1. Comparing OpenShift Data Foundation with IBM Spectrum Scale and NFS

	OpenShift Data Foundation	IBM Spectrum Scale	NFS Storage
Key Value Proposition	• Cluster based on CEPH • Tightly integrated into Red Hat OpenShift Container Platform • Special developed for RHOCP	• Cluster based on General Parallel File System (GPFS) • Can share data between different architectures (x86, Power, IBM Z) • OCP attached via CSI API	• Popular and common usage with same API for all HW architectures • Not highly performant • Great for tests & simple cases
Additional Aspects	• Software defined storage Rook in SAN • Can federate RHOCP storage across local and cloud environments via Noobaa • SAN using FCP/SCSI Storage	• Can also host Db2, Oracle … • Data Tiering included • Backup & HA / DR functions • Can use SCSI + ECKD disks • RHOCP attaches to an existing cluster	• Transparent access to data from different architectures • Not highly securable • No tiering or auto scaling • Not recommended for production
Scale	• Min of 3 storage nodes • Min 1 disk • Maximum of 3 X 500 PVs • Max of 81 TiB (3 x 27) storage	• Highly scalable to Petabytes • Implementation variations for scalability and performance	• Can scale based on the NFS limits
Storage Classes	• General purpose file, block or object Cloud native storage • Multicloud object store via MCG gateway (Noobaa)	• General purpose file storage • Can be used for RHOCP or other data / solutions	• General purpose file storage

4.4.7 IBM Spectrum Scale Container Native Storage Access (CNSA)

What if you could use one storage solution for all your data, like databases and containers and have shared file systems. Share the same data in different environments and IT architectures, use automatic storage tiering with the fastest storage device for the most frequent accessed data and your backup and secure archive would be automated in the tiering process. A comprehensive storage software family from IBM, the IBM Spectrum Storage™ Suite is available and has the capability to change the economics of storage on-premise and in hybrid Multicloud environments.

One of the fundamental components is IBM Spectrum Scale®, which is platform-independent, software defined storage. IBM Spectrum Scale for Linux on IBM Z implements a clustered file system and has many features beyond common shared data access. It includes data replication, policy-based storage management, and multi-site operations. It provides superior capabilities of resiliency, scalability, and high-performance for data and file management - built upon IBM General Parallel File System (GPFS™). IBM Spectrum Scale implements high availability and reliability with no single point of failure.

It can run on IBM Z, IBM LinuxONE, IBM Power Systems, and on x86 machines. The attached storage can be FICON attached DASD or FCP/SCSI attached storage from IBM and/or other vendors. IBM Spectrum Scale offers native, high-performance, and scalable access to file and object data via almost all standard storage protocols, including OpenStack®, Swift, Amazon S3, CIFS, NFS, HDFS, and Red Hat® OpenShift Container Platform via the Containers Storage Interface (CSI).

You can implement the storage cluster next to your RHOCP cluster and take advantage of the reduced latency, by using internal networks and the new *direct access* feature gives direc access to disks and reduces the network communication for just IBM spectrum Scale control information.

To install an IBM Spectrum Scale cluster, you find the detailed documentation at Getting started with IBM Spectrum Scale for IBM Z

The picture below illustrates the architecture of a RHOCP cluster using the Cloud Native Storage Access (CNSA) components to connect to a Spectrum Scale storage cluster. To plan, setup, and install the environment, you can find more details in IBM Spectrum Scale for Red Hat OpenShift Container Platform on IBM Z and IBM LinuxONE

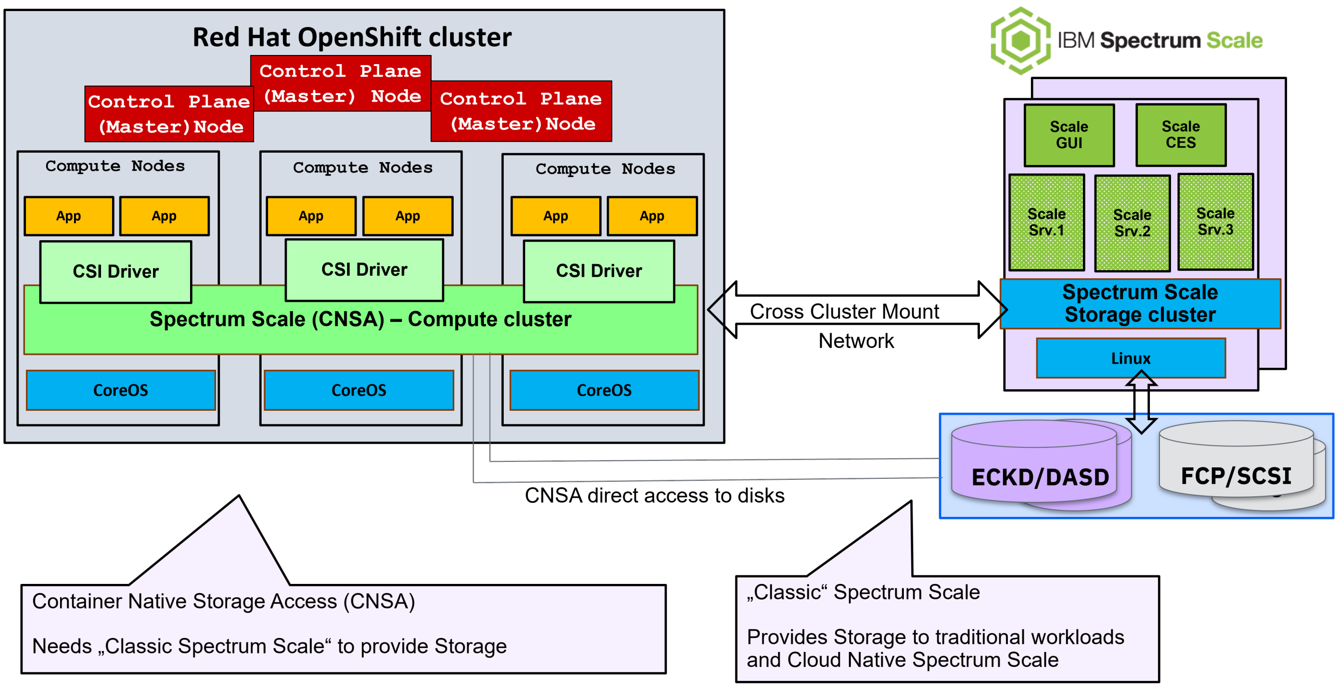

Red Hat OpenShift cluster
Control Plane (Master) Node
Control Plane (Master) Node
Control Plane (Master) Node
Compute Nodes
Compute Nodes
Compute Nodes
App
CSI Driver
Spectrum Scale (CNSA) – Compute cluster
CoreOS
Cross Cluster Mount
Network
IBM Spectrum Scale
Scale GUI
Scale CES
Scale Srv.1
Scale Srv.2
Scale Srv.3
Spectrum Scale Storage cluster
Linux
ECKD/DASD
FCP/SCSI
CNSA direct access to disks
Container Native Storage Access (CNSA)
Needs „Classic Spectrum Scale“ to provide Storage
„Classic“ Spectrum Scale
Provides Storage to traditional workloads and Cloud Native Spectrum Scale

Figure 4.4.7-1: Spectrum Scale Cluster attached to RHOCP

4.4.8 Summary of the Options

Table 2. Options for Storage

Options	**Suitable usage scenario**
Local Storage Operator	Operator used to create PVs from Local Volumes
Local Volume	local attached persistent volumes, statically created, ReadWriteOnce (without sharing across nodes)
iSCSI	ReadWriteOnce, ReadOnlyMany, manually provisioned
Fibre Channel	ReadWriteOnce, ReadOnlyMany, manually provisioned
HostPath	Quick options for testing, ReadWriteOnce
NFS	network attached persistent shared storage, ReadWriteMany
IBM Spectrum Scale CNSA	Shared, Clustered, multi-purpose network attached storage option, CSI attached to a RHOCP cluster and implemented with IBM Spectrum Scale Container Native Storage Access (CNSA)
Using Container Storage Interface (CSI) attached storage	ISV provided driver for storage attachments
Red Hat OpenShift Data Foundation	Versatile multi-purpose storage access. ReadWriteOnce, ReadWriteMany

4.5 Implementing high availability

High availability (HA) and Disaster Recovery (DR) are requirements for production deployments. A crucial consideration is removing single points of failure on the various layers of the environment. The goal is to avoid planned and unplanned outages and ensure continuous operation. The considerations for HA/DR include the following:

- All layers of the environment to be considered
- The solution architecture integrates HA and DR concepts
- Application development and DevOps (e.g. ensure that new services can be deployed without downtime aka rolling updates)
- The operational procedures are well defined and consider HA and DR

The following diagram shows an example of a highly available RHOCP deployment. IBM virtualization and the IBM Z and IBM LinuxONE platform offers characteristics, which make it a perfect match for high availability (e.g. redundancy of hardware elements). Red Hat OpenShift Container Platform (RHOCP) adds to this with its cluster-based implementation, the ability for fail-over of workloads within the same cluster. In the simplest setup for HA, there is a possibility to unify HA with disaster recovery.

For pure DR scenarios, an outage will occur when restarting applications or when failing over from Active to the Standby datacenter. In preparation to DR scenarios, consider disk replication and automation for the failover between datacenters.

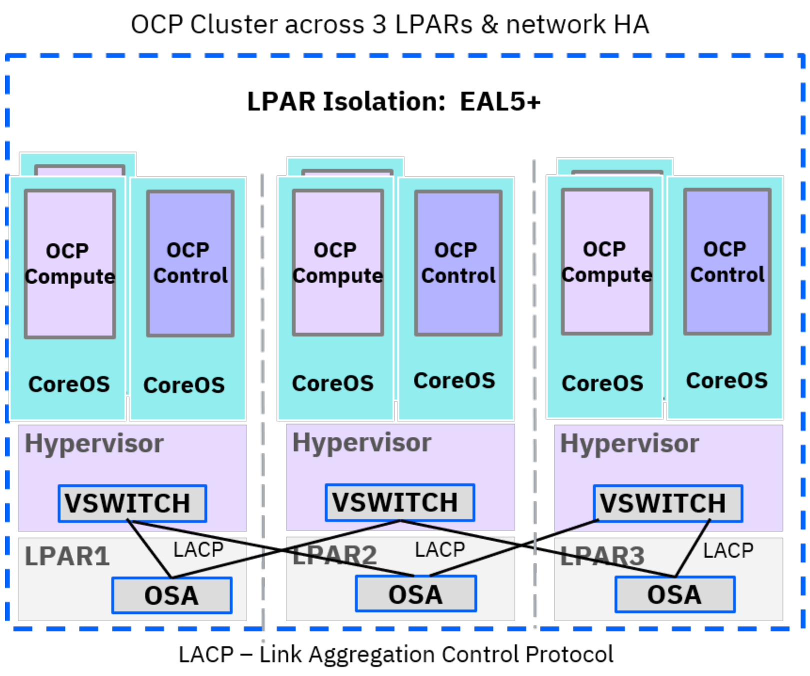


Figure 4.5-1: RHOCP Architecture for High Availability

As soon as disaster recovery is needed, a second site needs to be established. In case of a disaster the secondary site will take over and run the entire workload. To enable that, the Capacity Upgrade Feature (CBU) is available for IBM Z and IBM LinuxONE. It allows to increase the capacity in the DR datacenter in case of a disaster.

Key challenges to consider are:

- Replication of the application data between the sites (synchronously or asynchronously).
    - Storage must be replicated either by the application or databases or via the storage subsystem.
    - Coordinated backups of the application and data is essential.
- Network latency between the sites if the data between the sites is managed synchronously (which implies that synchronous active-active DR setups are only possible if the datacenters are in close geographical proximity).
    - Red Hat OpenShift requires a low latency network across its control planes to synchronously replicate state.
- Operational considerations, e.g. a plan with steps to switch from a production environment to a DR datacenter.

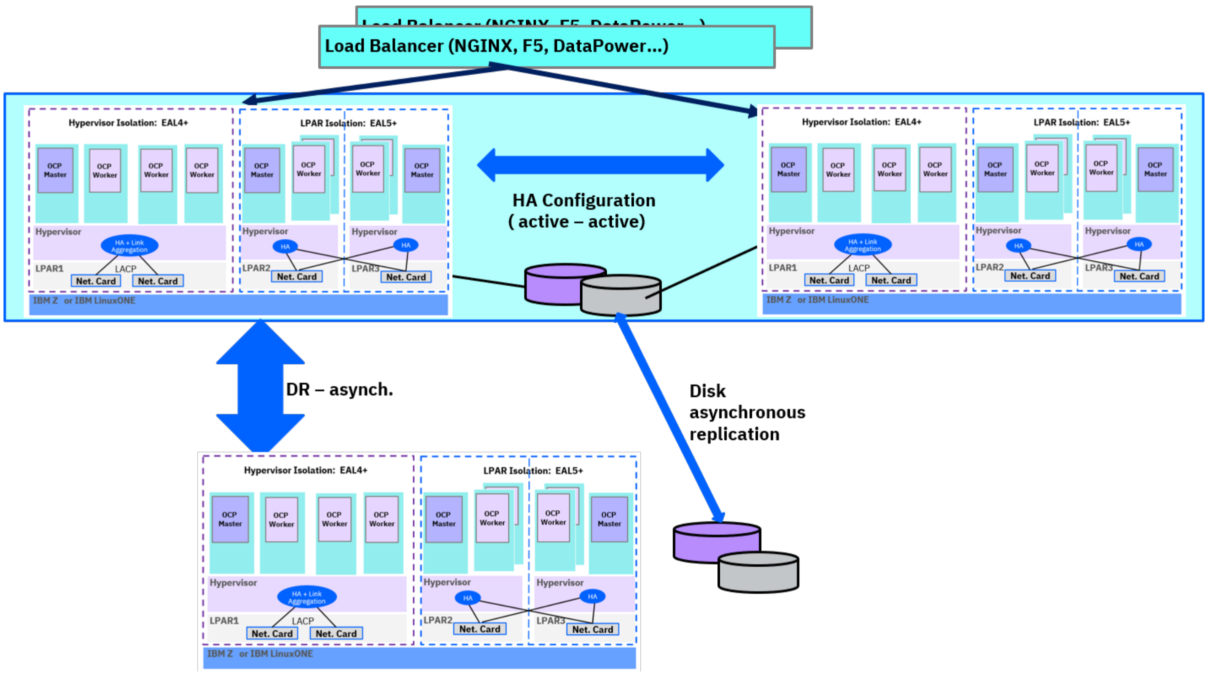


Figure 4.5-2: RHOCP Architecture for high availability and disaster recovery

4.5.1 RHOCP HA

RHOCP is also deployed for HA. The control plane manages the RHOCP cluster. The `etcd` state database are running on the control plane nodes. As `etcd` requires a minimum of three nodes for HA, you must deploy three control plane nodes.

RHOCP hosts critical control plane components on three control plane nodes for HA. In RHOCP 4.x, the various infrastructure services, such as router pods, container image registry, metrics and monitoring services, and cluster aggregated logging, no longer run on separate infrastructure nodes by default. Instead, infrastructure services now run on the compute nodes. You can deploy the infrastructure services to separate infrastructure nodes on Day 2. For more information, see Creating infrastructure MachineSets.

Compute nodes are where the actual application workloads run. The control planes manage the compute nodes. You should ensure that you have enough compute nodes to handle failures due to excessive load and lost infrastructure.

For a working HA setup, these constraints need to be considered:

- The number of (n/2)+1 control plane nodes must always be active in order to keep the cluster running.
- In order to ensure data consistency in the storage, the different datacenters/systems need to be close enough for synchronous communication. An active/active setup is supported in this case.
- Different data centers which are further apart in terms of geography can be combined to a DR/HA solution as a active/passive setup. Data storage needs an asynchronous disk replication.

NOTE At the moment of writing this document Red Hat OpenShift Platform 4.x requires that exactly 3 master nodes are present. Scaling master nodes is currently not supported as it means scaling etcd nodes. For more information, see Is it possible to scale master / etcd nodes in Red Hat OpenShift Platform 4.x?.

4.5.2 Storage HA

Storage HA can be implemented through replication or mirroring or by defining storage clusters across the environments to be HA. The HA scenarios can even be implemented between datacenters if they are in a distance that allows synchronous replication. IBM Spectrum Scale and Red Hat OpenShift Data Foundation, when available for IBM Z and IBM LinuxONE, are enabled to provide several HA options, when dealing with storage on IBM Z and IBM LinuxONE, even across multiple different architectures.

4.5.3 Hardware HA

IBM Z and IBM LinuxONE features all the aspects of reliability and resilience, eliminating most single points of failure at the hardware level. For more information on reliability, availability, and serviceability of IBM Z refer to: https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zmainframe/zconc_RAS.htm

Hardware fault tolerance of IBM Z and IBM LinuxONE include the following:

- RAID on server internal hard disks
- Redundant server power supplies connected to different power sources
- Bonded network interfaces connected to redundant network switches
- Extensive self-checking a self-recovery capability

4.6 Deployment options

You can deploy Red Hat OpenShift Container Platform (RHOCP) on IBM Z and IBM LinuxONE with different network and internet connections.

- Deploying RHOCP with a direct internet connection
- Deploying RHOCP using a corporate proxy service
- Deploying RHOCP in a restricted network environment

4.6.1 Deploying RHOCP with a direct Internet connection

The RHOCP installation program can take advantage of direct Internet access and an active subscription to Red Hat products. The installation processes are not self-contained and require access to external sources to retrieve core assets such as:

- RHOCP containers
- RHOCP images
- RHOCP programs

Therefore, an active, working DNS resolution is required.

4.6.2. Deploying RHOCP using a corporate proxy service

A RHOCP user-provisioned infrastructure deployment supports the implementation of a HTTP or HTTPS cluster-wide proxy at both install time, by using the configuration file, and after an install, by using the cluster-wide egress proxy. For more information on setting up a cluster-wide proxy, see Configuring the cluster-wide proxy in the *Red Hat OpenShift Container Platform Networking* guide.

4.6.3 Deploying RHOCP in a restricted network environment

In many customer environments, the RHOCP cluster needs to be strictly disconnected from the Internet for security reasons. In this case, an air-gapped installation can be applied.

In Red Hat OpenShift Container Platform 4, you can perform an installation that does not require an active connection to the internet to obtain software components.

To start a restricted network installation, upfront you have to create a registry that mirrors the contents of the Red Hat OpenShift Container Platform registry and contains the installation media. You can create and mirror this registry on a separate server, which can access both the internet and later your closed network, or by using other methods that meet your restrictions.

In a restricted network installation you are not able to apply Updates over the internet, you need to use the same method as for installation. For details, see Link below: https://docs.openshift.com/container-platform/latest/updating/updating-restricted-network-cluster.html

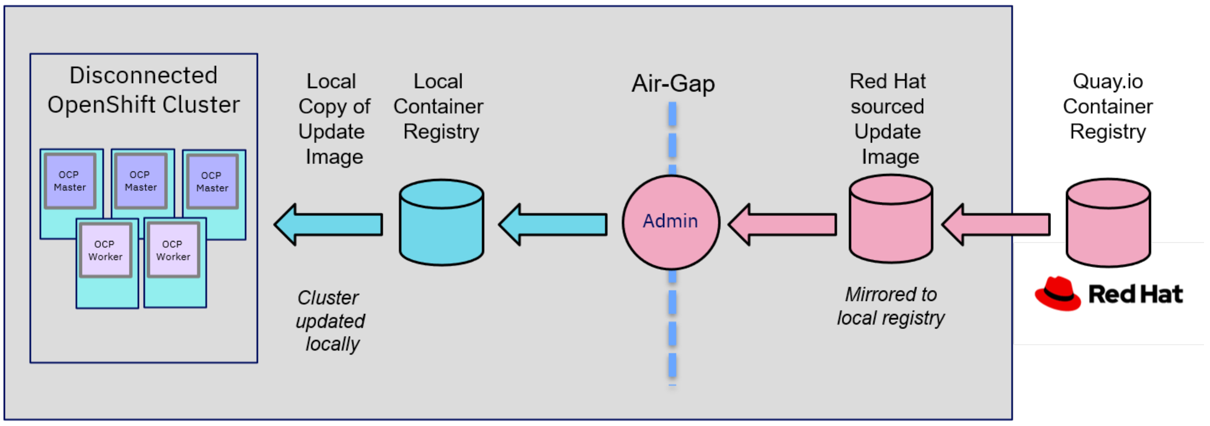

Disconnected OpenShift Cluster
OCP Master
OCP Master
OCP Master
OCP Worker
OCP Worker
Local Copy of Update Image
Local Container Registry
Air-Gap
Admin
Red Hat sourced Update Image
Quay.io Container Registry
Red Hat
Cluster updated locally
Mirrored to local registry

Figure 4.6-1: Air Gapped Installation

Air-gapped installation is a very common setup in enterprise customer environments.

Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The ClusterVersion status includes an unable to retrieve available updates error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required ImageStreamTags.

Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The kube-controller-manager only approves the kubelet client CSRs. The machine-approver cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

See additional resources:

- See Bridging a HiperSockets LAN with a z/VM Virtual Switch in the IBM Knowledge Center.
- See Scaling HyperPAV alias devices on Linux guests on z/VM for performance optimization.
- See KVM setup, networking and Hipersockets setup

Multipathing is now supported during initial deployment and early boot for FCP-attached storage devices when attached to clusters that are created using OpenShift Container Platform 4.7 or later. For installations on FCP-type disks, complete the tasks described in the Link and see Step 4 of the details for different nodes to enable Multipathing.

4.7 Resource considerations

4.7.1 Disk and I/O consideration

It is essential that you ensure fast, redundant disk is available to your Red Hat OpenShift Container Platform (RHOCP) installation. To make this possible on RHOCP on IBM Z and LinuxONE, you need to consider the minimum disk requirements and how you are going to provide the disk to the RHOCP nodes.

4.7.2 Network consideration

To install RHOCP on IBM Z, the hypervisor requires at minimum one virtual NIC in layer 2 mode, attached to:

- A virtual switch in the hypervisor attached to a network card or
- Definitions for direct-attached OSA or RoCE network adapter to the RHOCP nodes

You can have one or more network interfaces per RHOCP node.

4.7.2.1 Network options using the z/VM hypervisor

With z/VM, you can take advantage of the z/VM VSWITCH, which means you need:. - A z/VM VSWITCH set up if RHOCP is not direct attached to OSA or RoCE. For a preferred setup, use OSA link aggregation with z/VM VSWITCH.

4.7.2.2 Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in initramfs during boot to fetch Ignition config from the Machine Config Server.

During the initial boot, the machines require either a DHCP server or that static IP addresses be set on each host in the cluster in order to establish a network connection, which allows them to download their Ignition config files.

It is recommended to use the DHCP server to manage the machines for the cluster long-term. Ensure that the DHCP server is configured to provide persistent IP addresses, hostnames, the default gateway and the DNS servers for the cluster machines.

The Kubernetes API server, which runs on each Control Plane node after a successful cluster installation, must be able to resolve the node names of the cluster machines.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

The user-provisioned infrastructure method creates all RHOCP internal networks required. This means that modifying those networks with special settings is not possible unless done manually after the installation.

4.7.3 Limitations

As the control plane virtual machines run the etcd key-value store, they need to meet the known resource requirements. Fast disks are the most critical for etcd stability and performance.

Minimum disk requirement	50 sequential IOPS, for example, a 7200 RPM disk.
Minimum disk requirement for heavily loaded clusters	500 sequential IOPS, for example, a typical local SSD or a high performance virtualized block device.

For more information on the etcd disk requirements, see Hardware recommendations - Disks.

For this reference architecture, all control plane nodes need access to fast disk to ensure stability and guarantee supportability. Contact your Red Hat account team for assistance on reviewing your disk performance needs.

Installation

This reference architecture is intentionally opinionated and prescriptive. You can customize your initial deployment by using the `install-config.yaml` file. Manual configuration changes to RHCOS itself are not supported.

Nevertheless additional infrastructure customization beyond the install-config.yaml can occur after deployment using Machine Config Operators (MCO). MCO can be used to change machine configuration settings in a day 2 operation. For example, this essentially is required to install an IBM Spectrum® Scale Container Native Storage Access (CNSA) because it requires CoreOS extensions managed through MCO and modifications to kernel parameters.

Details can be found in this documentation: https://docs.openshift.com/container-platform/latest/post_installation_configuration/machine-configuration-tasks.html

4.8 Installation methods and tooling

Red Hat OpenShift Container Platform (RHOCP) has a new installation program for the 4.x stream. It features a streamlined interface and simplified installation process allowing a faster, easier, and more precise installation. For more information, see the Red Hat OpenShift Container Platform Installing guide

RHOCP offers two different kinds of installation:

- User-provisioned infrastructure
- Installer-provisioned infrastructure

As customers on IBM Z typically have strongly customized system topologies with many individual infrastructure choices, The RHOCP installation program focuses on the **user-provisioned infrastructure** installation: Administrators are responsible for preparing, creating and managing their own underlying infrastructure for clusters. This approach allows greater customization prior to installing RHOCP.

An installed RHOCP cluster has the following characteristics:

- Highly available infrastructure with no single points of failure by default.
- A deep integration between RHOCP and the underlying operating system, Red Hat Enterprise Linux CoreOS (RHCOS), that provides “appliance-like” integration.
- Administrators maintain control over what updates are applied, and when.

4.8.1 The main steps to install a RHOCP cluster in a z/VM environment are:

- Prepare your z/VM environment (User directories, storage, network)
- Prepare a Bootstrap (temporary) server
- Access the installation images and files
  - Setup an FTP or HTTP server with internet access to acccess the RHOCP registry
  - For restricted network install, setup a separate server to mirror the registry with installation images first, before the installation process
- Prepare the RHOCP pre-req. Services: DNS, NFS, load balancer, DHCP
- Download RHOCP product code for IBM Z and IBM LinuxONE from cloud.redhat.com:
  - openshift-installer, RHCOS image
- Run the openshift-installer to define and create ignition files for the RHOCP cluster
- Save the ignition files on the FTP / HTTP server
- Copy kernel image, parmfile, coreos-installer image to your z/VM guests
- Adjust the parmfile for the z/VM guest and specify the ignition file for the bootstrap, master, and compute nodes
- Punch the installation files into the z/VM virtual readers

- Boot (IPL from z/VM virtual reader) the CoreOS-installer on each node, to install the bootstrap, master, and compute nodes

4.8.2 The main steps to install a RHOCP cluster in a KVM environment are:

- As prerequisite, you need to install a RHEL 8.3 KVM or later environment
- Prepare your KVM environment (User entries, storage, network)
- Prepare an installer VM and bootstrap VM (temporary) server
- For restricted network install, set up a separate server to mirror the registry with installation images before the installation process
- Prepare the RHOCP pre-req. Services: DNS, NFS, load balancer, DHCP
- Download RHCOS QCOW2 images and installer for IBM Z and IBM LinuxONE from mirror.openshift.com, prior to cluster creation
- Run the openshift-installer on installer VM to define and create Ignition files for the RHOCP cluster
  - Cluster has to be created within 24 hours after ignition file creation
  - The certificated will expire if the cluster is not created within 24 hours after the ignition files have been generated.
- Send the ignition files to the KVM host.
- Start bootstrap, all control plane nodes and all compute nodes with short delay
  - In case that it did not happen automatically, allow compute nodes to be added to the cluster
- Remove the bootstrap or transform it into a compute node
- Allow the compute nodes to be added to the cluster by approving any pending CSRs
- Cluster should run for 24 hours for first certificate rotation process to complete

A detailed description is provided via the links below.

NOTE	Currently you can build a user-provisioned infrastructure (UPI) cluster, the installer-provisioned infrastructure (IPI) cluster is on the roadmap for an IBM Z and IBM LinuxONE environment.

This reference architecture features the user-provisioned infrastructure method for installing RHOCP with z/VM or KVM. This results in an architecture that is highly available, fully tested, and entirely supported, suitable for production today.

For the minimum and preferred prerequisites please check the full documentation:

- https://docs.openshift.com/container-platform/latest/installing/installing_ibm_z/installing-ibm-z.html
- https://developer.ibm.com/tutorials/install-red-hat-openshift-on-ibmz-linuxone/
- https://www.openshift.com/blog/installing-ocp-in-a-mainframe-z-series
- https://www.openshift.com/blog/red-hat-openshift-installation-process-experiences-on-ibm-z-

linuxone

- https://docs.openshift.com/container-platform/latest/installing/installing_ibm_z/installing-ibm-z-kvm.html#installation-infrastructure-user-infra_installing-ibm-z-kvm

If you need to add additional compute nodes after you had completed the installation process, you can check the instructions in the Chapter 5.6: Adding a new compute node.

4.9 Performance considerations

4.9.1 Best practices for performance

For a RHOCP environment it is very important to understand the workload characteristics from different aspects.

- workload, which is compute (CPU) intensive
- workload, which is memory intensive
- workload, which has a high network traffic inside the cluster due to the application distribution in the pods (Apps working with a database in RHOCP)
- workload, which requires to serve a high number of requests from outside the RHOCP cluster (e.g. Web or Mobile Apps)

Based on observations in several RHOCP environments, there is a collection of best practices considerations for:

- CPU-intensive workloads and network-intensive workloads

The performance test results were obtained in a controlled environment. The measured differences in performance, latency, throughput, etc. might be different in other scenarios and production environments, but the best practices apply.

All details can be found in the documentation and further publications:

Recommended host practices documentation

OpenShift on IBM Z performance experiences and best practices

Chapter 5. Operational considerations

This chapter focuses on aspects of "Day 2" operations like validation, security, persistence, backup and performance tuning. This chapter enables you to become productive quickly. It also helps to make your solution reliable and secure right from the beginning.

5.1 Post installation validation

Some validations steps can be executed to ensure that the installation of Red Hat OpenShift Container Platform has been successful. The following tasks are a good routine to follow after the installation of Red Hat OpenShift Container Platform:

- Confirm that all the cluster components are online checking the cluster operators
- Check the cluster version

You can check the technical details in Chapter 8.2 Validations.

5.1.2 Troubleshooting

If something went wrong during the installation and the previous command shows and error, the following actions can be taken:

- Check the status of the pods in all namespaces (Chapter 8.3 Checking the status of the pods)
- Check the logs for a Pod that is listed in the output of the previous command and is not running (Chapter 8.3: Checking the logs of a pod)
- Gather debug information (Chapter 8.3: Gathering debug information)

See also how to generate SOSREPORT within RHOCP 4 nodes without SSH.

5.2 Adding Persistent Storage with NFS

After installing the cluster, by default, all the container images pulled from the remote registries are stored locally in memory, that means if the registry pod is restarted all these images will be lost. Since Red Hat OpenShift Container Platform is running on IBM Z and IBM LinuxONE, you need to make use of persistent storage to store the container images on a persistent volume.

Persistent storage can be local storage that is attached directly to the hypervisor and made available to the cluster, or network attached storage such as NFS or IBM Spectrum Scale.

The following sections show how to add persistent storage to the RHOCP image registry, first using the RHOCP user interface (UI) and later also using the RHOCP CLI.

WARNING

Testing shows issues with using the NFS server on RHEL as storage backend for core services. This includes the Red Hat OpenShift Container Registry and Quay, Prometheus for monitoring storage, and Elasticsearch for logging storage. Therefore, using RHEL NFS to back PVs used by core services is not recommended. Other NFS implementations on the marketplace might not have these issues. Contact the individual NFS implementation vendor for more information on any testing that was possibly completed against these Red Hat OpenShift Container Platform core components.

5.2.1 Preparing the NFS server

The first thing to do is to set up a NFS server in the network. For testing purposes, the NFS server using Red Hat Enterprise Linux can be used, alternatively, you can use any other Linux distribution or appliance as well as use a storage device that provides NFS shares.

An NFS server is required to be installed and configured with firewall rules enabled. Refer to the official documentation for detailed technical instructions on setting up and configuring NFS: official documentation.

The steps and requirements are as following:

- The minimum requirements for the RHOCP image registry is 100Gi, for this example a disk volume of 100Gi has been added and mounted to the NFS server under the directory /mnt/ocp4-registry.
- To make it permanent make sure that is configured in the /etc/fstab
- The file /etc/exports contains a table of local physical file systems on an NFS server that are accessible to NFS clients

5.2.2 Adding persistent storage with NFS using the UI

Once the NFS server is configured the RHOCP User Interface or the RHOCP CLI can be used to create the persistent volumes. The following steps summarize the process:

- Create a Persistent Volume using the RHOCP web console Chapter 8.4: Persistent Volume example

- Create a Persistent Volume Claim using the RHOCP web console (make sure the project "openshift-image-registry" is selected) Chapter 8.3: Persistent Volume Claim example
- The last thing to do is to edit the operator to use the new NFS configuration. Chapter 8.3: Editing the image registry operator

5.3 Adding dynamic persistent storage with NFS

After using NFS to provide persistent storage in the previous chapter, this section will show how to set up NFS and RHOCP 4 to provide a dynamic allocation of persistent volumes to workloads running on the cluster. This approach uses a Storage Class and explains how to configure RHOCP to use a default storage class. Next a simple application will be deployed to demonstrate the dynamic allocation of storage to the deployment. The official documentation defines what a storage class is and how it relates to dynamic provisioning with the following words:

> "The StorageClass resource object describes and classifies storage that can be requested, as well as provides a means for passing parameters for dynamically provisioned storage on demand. StorageClass objects can also serve as a management mechanism for controlling different levels of storage and access to the storage. Cluster Administrators (cluster-admin) or Storage Administrators (storage-admin) define and create the StorageClass objects that users can request without needing any intimate knowledge about the underlying storage volume sources.
>
> The Red Hat OpenShift Container Platform persistent volume framework enables this functionality and allows administrators to provision a cluster with persistent storage. The framework also gives users a way to request those resources without having any knowledge of the underlying infrastructure."

The following steps are showing how to add dynamic persistent storage with NFS (Chapter 8.3: Dynamic persistent storage using NFS):

- The first thing to do is to define the NFS directories for each persistent storage definition
- Later, the directories created must be added to the NFS exports configuration
- The storage class needs to be created and defined as the default one
- Now the PersistentVolumes that will use the NFS directories created on the first point can be created
- Lastly, a test application to use the dynamic storage can be deployed

NOTE For applications that require persistent storage during the deployment, such as databases for example, the allocation of the persistent storage should happen during the deployment using this configuration.

For further details and technical details the following documentation can be checked.

5.4 Backup and recovery

Backup and Recovery is a consideration, which is essential for production use.

You must ensure the backup of the crucial etcd database.

- The etcd database is the key-value store for RHOCP container Platform, which persists the state of all resource objects.
- Do not take an etcd backup before the first certificate rotation completes, which occurs 24 hours after installation, otherwise the backup will contain expired certificates.
- It is recommended to take etcd backups during non-peak usage hours, as it is a blocking action.
- It is highly recommended that you monitor Prometheus for etcd metrics and defragment it when needed before etcd raises a clusterwide alarm that puts the cluster into a maintenance mode

Some more details can be found here:

https://docs.openshift.com/container-platform/latest/scalability_and_performance/recommended-host-practices.html

https://docs.openshift.com/container-platform/latest/backup_and_restore/control_plane_backup_and_restore/backing-up-etcd.html

Beyond the etcd database, there are the backup strategies for your virtualization layer (e.g. z/VM or KVM).

In addition, there is helpful background information on backup/recovery for z/VM in this presentation:

https://www.ibm.com/downloads/cas/LN4RXLJ3

5.5 Security and identities

When accessing RHOCP you need to authenticate with the cluster. During the installation process, an initial user identity is created (*kubeadmin*) to get started quickly. One of the first tasks after the installation is to connect RHOCP to a user directory listed the actual users.

For this purpose, RHOCP needs to be connected to an identity provider, that contains the actual users.

Supported Identity providers are

- **HTPasswd**: A flat list of names and passwords
- **Keystone**: OpenStack Keystone v3 server with an internal database
- **LDAP**: LDAPv3 server using simple bind authentication.
- **Basic authentication**: Remote identity provider using basic authentication as a generic backend integration mechanism.
- **Request header**: identify users from request header values, such as X-Remote-User.
- **GitHub** or **GitHub Enterprise**: Validate user names and passwords against GitHub or GitHub Enterprise's OAuth authentication server.
- **GitLab**: Use GitLab.com or any other GitLab instance as an identity provider.
- **Google**: Use Google identity provider using Google's OpenID Connect integration.
- **OpenID Connect**: oidc identity provider to integrate with an OpenID Connect identity provider using an Authorization Code Flow.

More details are described in the following documentation: https://docs.openshift.com/container-platform/latest/authentication/understanding-identity-provider.html

Once an identity provider is successfully configured, the default .kubeadmin. identity is no longer needed and should be removed in productions environments. But keep in mind: It's not possible to recreate the kubeadmin later again without a full re-install. Prior to removing the kubeadmin account ensure a user account has assigned the cluster-admin role.

RHOCP provides its own internal CA

- Certificates are used to provide secure connections to
  - Control plane(APIs) and computes
  - Ingress controller and registry
  - etcd
- Certificate rotation is automated
- Optionally configure external endpoints to use custom certificates

Users can be granted specific permissions as *regular users*, *_system users*, or as *service account*. Users can also be assigned to groups, which makes assigning of permissions easier, as each permission will be mapped to all users within a specific group.

This role-based access control of RHOCP allows to precisely define which permissions are granted to which set of users

- Project scope & cluster scope available
- Matches request attributes (verb, object, etc)
- If no roles match, request is denied (deny by default)
- Operator- and user-level roles are defined by default
- Custom roles are supported

More details are described in the product documentation https://docs.openshift.com/container-platform/latest/authentication/understanding-authentication.html

- If RHOCP security can also be managed in a z/VM and z/OS manner using the LDAP extension for RACF (Resource Access Control Facility).
  - How to use that with z/VM can be found in the z/VM documentation: https://www.ibm.com/docs/en/zvm/7.1
  - How to integrate z/OS RACF with LDAP can be found here: https://www.ibm.com/support/knowledgecenter/SSLTBW_2.2.0/com.ibm.zos.v2r2.icha700/ldap.htm

For highest encryption, the IBM Z platform uses IBM® Crypto Express cards for IBM Z® and IBM LinuxONE which are Crypto Express Accelerator Cards (CEX) that can enable encryption of data using a master key, which is never leaving the CEX card and is never in memory, therefore it enables the highest security in an environment. With Red Hat OpenShift, you can take advantage of these crypto capabilities, and access the crypto functions directly from a container. To enable the capability and for further details, you can use the description of the Kubernetes device plug-in for IBM Crypto Express (CEX) cards.

```
provides containerized applications access to IBM Crypto Express cards.
```

The *Red Hat OpenShift Security Guide* provides a comprehensive and detailed look into the many challenges related to security in the cloud. This guide helps in the triaging of security trade-offs and risk, policy enforcement, reporting, and the validation of system configuration. The guide is available to download from the Red Hat Customer Portal.

5.6 Adding a new compute node

After installing RHOCP on IBM Z it is possible to add more compute nodes to the cluster. The process described below assumes that you already have an existing RHOCP cluster running for more than 24-hours. In case your cluster is younger than 24-hours a compute node has to be deployed in the same way as during the installation process (for example creating RHCOS machines during installation on bare metal) but using an updated worker.ign file. This method consists on extracting the digital certificate from the running RHOCP cluster and used it to replace the one existing in the current worker.ign file.

5.6.1 Preparing the infrastructure

There are a few things to update in the existing infrastructure like DNS and loadbalancer as well as in the hipervisor, to add the VM definition for the new compute node.

- First the DNS Service configuration needs to be updated: The zone record needs to include the *A* pointers to the new node, the compute and the etcd, for example (compute2 has been added), an example is included in the Chapter 8.5: Adding a new node
- Later, the new compute node has to be added to the load balancer: In this example (HAProxy configuration example) HAProxy is used as a load balancer.
- A new virtual machine needs to be created: In this example (New VM guest example) the VM uses zFCP and not ECKD for the installation of Red Hat CoreOS.
- The next step is to transfer the initramfs, kernel, the parameter file (PARM file example) and the worker.exec to the new compute node virtual machine disk, it can be done using FTP or using the application VMUR (s390utils-base-2 rpm) if using a RHEL guest as a helper on the same LPAR like the VM guests. If installing a cluster across multiple LPARS, a RHEL guest must exist on each LPAR to upload the boot files to the VM guests on that LPAR.

5.6.2 Creating a new compute node based on an existing compute ignition file from the previous RHOCP install process

The following steps define how to start the process to install RHCOS to the new compute VM:

- Using a 3270 terminal emulator, the WORKER EXEC file has to be executed
- The node must be started to get installed with the new ignition file
- The new CSR must be approved

To get the detailed commands on how to add a new compute the following documentation is available:

- Adding compute nodes to the OCP 4 UPI cluster existing 24+ hours
- How to Add a new compute node on an Existing Red Hat OpenShift 4 Cluster on IBM Z and LinuxONE
- Installing OCP in a Mainframe z-series

5.7 RHOCP container catalog access for s390x

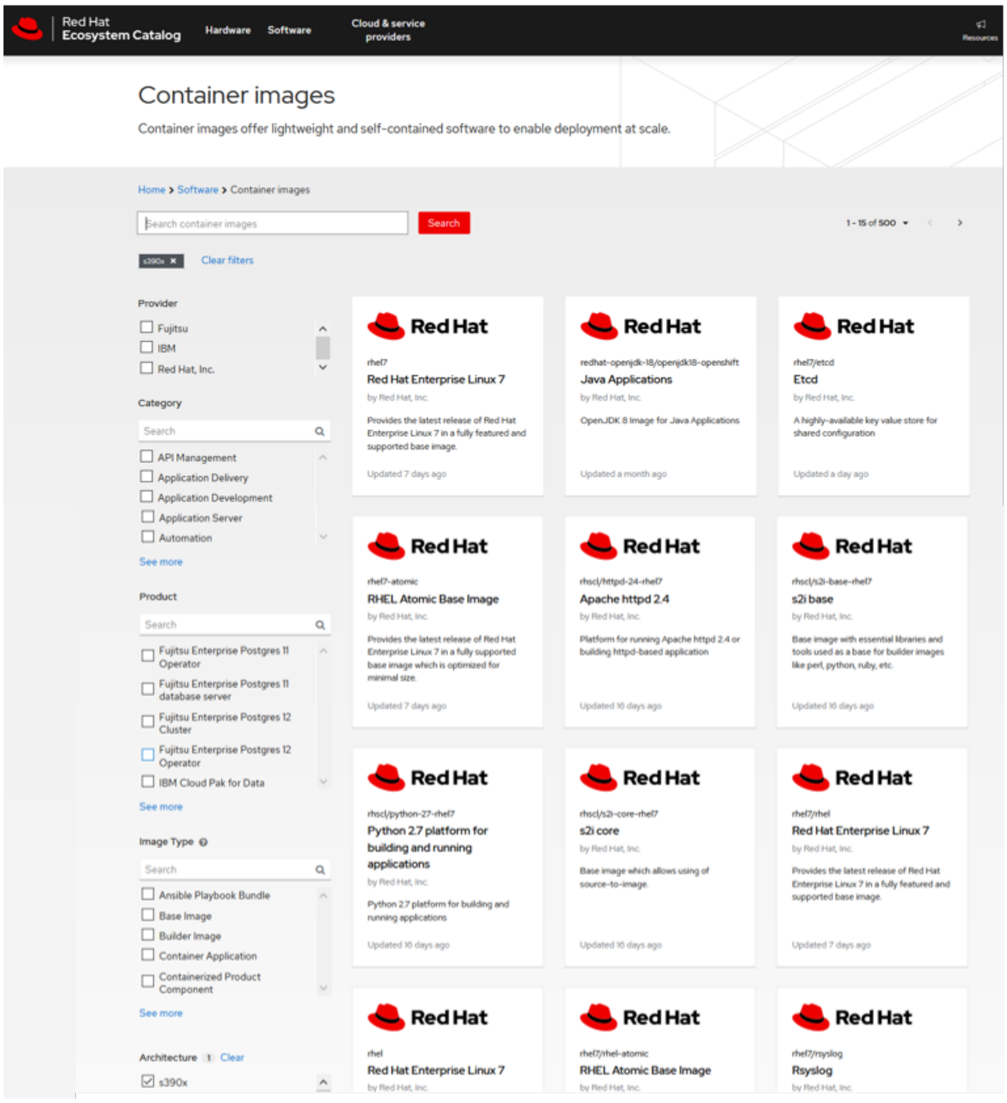
This chapter explains how to access the Red Hat Container Image Catalog and leverage hundreds of curated container images to run on OpenShift running on IBM Z.

One of the advantages of using the Red Hat Container Catalog is that this repository is curated by Red Hat and it receives constant updates to remediate vulnerabilities. This example demonstrates one way of using the container images from the Red Hat Container Catalog, for other methods please consult the Red Hat OpenShift official documentation.

To browse the Red Hat Container Catalog for s390x access the link below:

https://catalog.redhat.com

Applying two filters from the Red Hat Catalog, the option s390x on Architecture and the option Red Hat, Inc on Provider will narrow the results as shown in the figure 5.7-1 below:



Container images

Container images offer lightweight and self-contained software to enable deployment at scale.

rhel7
Red Hat Enterprise Linux 7
by Red Hat, Inc.
Provides the latest release of Red Hat Enterprise Linux 7 in a fully featured and supported base image.
Updated 7 days ago

redhat-openjdk-18/openjdk18-openshift
Java Applications
by Red Hat, Inc.
OpenJDK 8 Image for Java Applications
Updated a month ago

rhel7/etcd
Etcd
by Red Hat, Inc.
A highly-available key value store for shared configuration
Updated a day ago

rhel7-atomic
RHEL Atomic Base Image
by Red Hat, Inc.
Provides the latest release of Red Hat Enterprise Linux 7 in a fully supported base image which is optimized for minimal size.
Updated 7 days ago

rhscl/httpd-24-rhel7
Apache httpd 2.4
by Red Hat, Inc.
Platform for running Apache httpd 2.4 or building httpd-based application
Updated 16 days ago

rhscl/s2i-base-rhel7
s2i base
by Red Hat, Inc.
Base image with essential libraries and tools used as a base for builder images like perl, python, ruby, etc.
Updated 16 days ago

rhscl/python-27-rhel7
Python 2.7 platform for building and running applications
by Red Hat, Inc.
Python 2.7 platform for building and running applications
Updated 16 days ago

rhscl/s2i-core-rhel7
s2i core
by Red Hat, Inc.
Base image which allows using of source-to-image.
Updated 16 days ago

rhel7/rhel
Red Hat Enterprise Linux 7
by Red Hat, Inc.
Provides the latest release of Red Hat Enterprise Linux 7 in a fully featured and supported base image.
Updated 7 days ago

rhel
Red Hat Enterprise Linux 7
by Red Hat, Inc.

rhel7/rhel-atomic
RHEL Atomic Base Image
by Red Hat, Inc.

rhel7/rsyslog
Rsyslog
by Red Hat, Inc.

Figure 5.7-1: Red Hat Catalog filtered by s390x architecture and Red Hat

To access these container images on RHOCP on IBM Z or LinuxONE the following steps must be completed:

1. Create an account secret so that RHOCP will be able to access the Red Hat Container Catalog and download container images as image streams (Chapter 8.7.1: Registering a service account).
2. Create a project, assign the account secret to it, import the container image and create an image stream for it (Chapter 8.7.2: Creating a project, importing the Account Secret and selecting a container image to use).
3. Deploy a Pod using the image stream that contains the imported container image from the Red Hat Container Catalog (Chapter 8.7.3: Deploying a pod using the container image).

The Chapter 8.7: RHOCP container catalog access for s390x shows the technical details for enabling the access to the Red Hat Container Catalog.

5.8 Infrastructure nodes for boosting performance on RHOCP

To offload basic housekeeping tasks from your compute nodes, dedicated infrastructure nodes can be introduced. This will boost the performance of the compute nodes. As infrastructure nodes do not need licensing it will also help to optimize TCO.

To isolate and offload basic housekeeping tasks from your compute nodes, you can introduce dedicated infrastructure nodes. The following Red Hat OpenShift Container Platform components are infrastructure components:

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

The isolation of such components can boost the performance of the compute nodes. You can build the infrastructure nodes out of compute nodes.

Any node that runs any other container, pod, or component is a compute node that your subscription must cover.

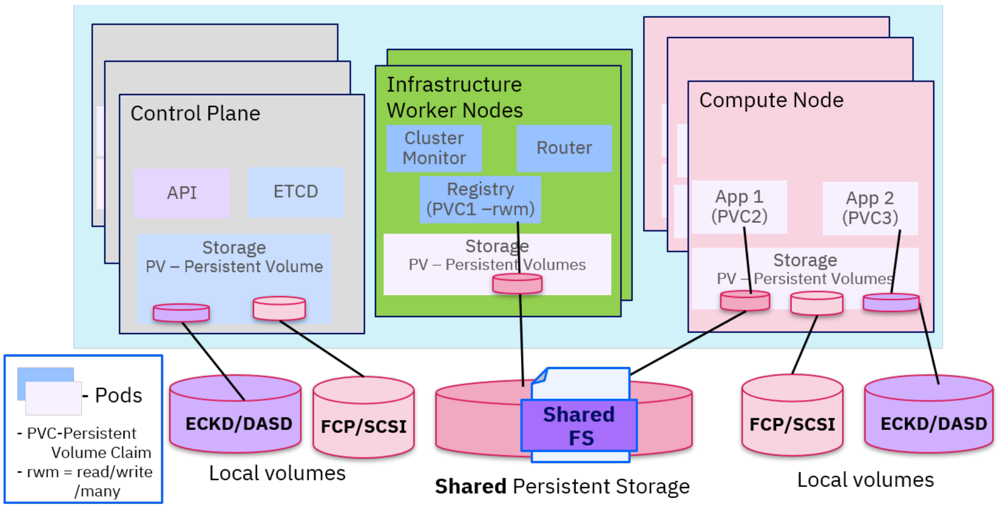


Figure 5.8-1: RHOCP with infrastructure nodes

For more information about infrastructure nodes and how to build them can be found here: https://docs.openshift.com/container-platform/latest/machine_management/creating-infrastructure-machinesets.html

5.8.1 RHOCP infrastructure components

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

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

Any node that runs any other container, pod, or component is a compute node that your subscription must cover.

In a production deployment, to hold infrastructure components at least three MachineSets are necessary . Both the logging aggregation solution and OpenShift Service Mesh deploy Elasticsearch, and Elasticsearch requires three instances that are installed on different nodes. For high availability, these nodes should be placed to different availability zones. Since you need different MachineSets for each availability zone, it is necessary to have at least three MachineSets.

5.8.2 Moving resources to infrastructure MachineSets

Some of the infrastructure resources are deployed in your cluster by default. You can move them to the infrastructure MachineSets that you created.

- **Moving the router** The router Pod can be deployed to a different MachineSet. By default, the Pod is displayed to a compute node.
- **Moving the default registry** The registry Operator must be configured to deploy its pods to different nodes.
- **Moving the monitoring solution** By default, the Prometheus Cluster Monitoring stack, which contains Prometheus, Grafana, and AlertManager, is deployed to provide cluster monitoring. It is managed by the Cluster Monitoring Operator. To move its components to different machines, a custom ConfigMap needs to be created.
- **Moving the cluster logging resources** The Cluster Logging Operator can be configured to deploy the pods for any or all the Cluster Logging components, Elasticsearch, Kibana, and Curator to different nodes. You cannot move the Cluster Logging Operator pod from its installed location. For example, the Elasticsearch pods can be moved to a separate node because of high CPU, memory, and disk requirements.

Once infrastructure nodes are added to your cluster configuration, it’s recommended to create a Machine Config Pool to the infra nodes as they were added after the RHOCP cluster creation. Otherwise, when the RHOCP cluster is upgraded, the compute nodes with the infra label applied won’t get upgraded. This is the expected behavior.

Refer to the official documentation for detailed technical instructions: https://docs.openshift.com/

container-platform/latest/machine_management/creating-infrastructure-machinesets.html

A good reading on this topic can be found here:

https://www.linkedin.com/pulse/boosting-performance-using-infrastructure-nodes-your-cluster-miranda/

5.9 Logging and monitoring

The Red Hat OpenShift Container Platform has built in capabilities for Monitoring and Logging. That means you can monitor in real-time the behavior of the cluster resources and application and create alerts if resource consumption exceed limits. For application behavior and operational purposes, RHOCP has the capability to monitor system and application Logs, and display them or forward them to an external Logging tool to be analyzed.

5.9.1 Monitoring in RHOCP

To learn more about monitoring, the different components and how to deploy and configure it you can check Understanding cluster logging

Red Hat OpenShift Container Platform includes a pre-configured, pre-installed, and self-updating monitoring stack that is based on the Prometheus open source project and its wider eco-system. It provides monitoring of cluster components and includes a set of alerts to immediately notify the cluster administrator about any occurring problems and a set of Grafana dashboards. The cluster monitoring stack is only supported for monitoring Red Hat OpenShift Container Platform clusters.

NOTE To ensure compatibility with future Red Hat OpenShift Container Platform updates, configuring only the specified monitoring stack options is supported.

For more information for understanding the monitoring stack you can check the official Red Hat OpenShift Container Platform

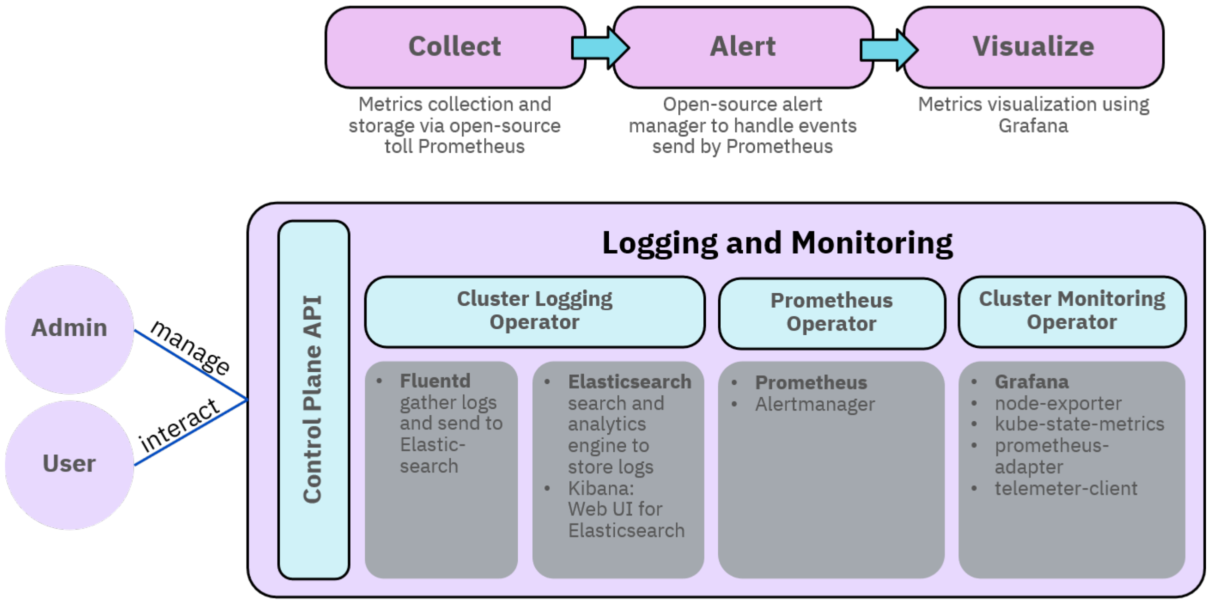

Collect
Metrics collection and storage via open-source toll Prometheus
Alert
Open-source alert manager to handle events send by Prometheus
Visualize
Metrics visualization using Grafana
Logging and Monitoring
Admin
User
manage
interact
Control Plane API
Cluster Logging Operator
Prometheus Operator
Cluster Monitoring Operator
Fluentd gather logs and send to Elastic-search
Elasticsearch search and analytics engine to store logs
Kibana: Web UI for Elasticsearch
Prometheus
Alertmanager
Grafana
node-exporter
kube-state-metrics
prometheus-adapter
telemeter-client

Figure 5.9-1: Monitoring Collection Alerting and Visualization of Metrics

5.9.2 Logging in RHOCP

As a cluster administrator, you can deploy cluster logging to aggregate all the logs from your Red Hat OpenShift Container Platform cluster, such as node system audit logs, application container logs, and infrastructure logs. Cluster logging aggregates these logs from throughout your cluster and stores them in a default log store. You can use those logs in different scenarios:

- View logs for a resource
- Use Kibana to visualize the log data
- Push those data to an outside Logging aggregation tool such as Splunk

Cluster logging aggregates the following types of logs:

- Application - Container logs generated by user applications running in the cluster, except infrastructure container applications.
- Infrastructure - Logs generated by infrastructure components running in the cluster and Red Hat OpenShift Container Platform nodes, such as journal logs. Infrastructure components are pods that run in the openshift*, kube*, or default projects.
- Audit - Logs generated by the node audit system (auditd), which are stored in the /var/log/audit/audit.log file, and the audit logs from the Kubernetes apiserver and the Red Hat OpenShift API Server.

NOTE Because the internal Red Hat OpenShift Container Platform Elasticsearch log store does not provide secure storage for audit logs, audit logs are not stored in the internal Elasticsearch instance by default. If you want to send the audit logs to the internal log store, for example to view the audit logs in Kibana, you must use the Log Forward API as described in Forward audit logs to the log store.

5.9.1 Sizing estimates for Monitoring and Logging

The monitoring stack imposes additional resource requirements. Consult the computing resources recommendations in Scaling the Cluster Monitoring Operator and verify that you have sufficient resources.

Number of Nodes	Number of Pods	Prometheus storage growth per day	Prometheus storage growth per 15 days	RAM Space (per scale size)	Network (per tsdb chunk)
50	1800	6.3 GB	94 GB	6 GB	16 MB
100	3600	13 GB	195 GB	10 GB	26 MB
150	5400	19 GB	283 GB	12 GB	36 MB
200	7200	25 GB	375 GB	14 GB	46 MB

Table 5.9-2: *Prometheus Database storage requirements based on number of nodes/pods in the cluster*

Approximately 20 percent of the expected size was added as overhead to ensure that the storage requirements do not exceed the calculated value.

The above calculation is for the default Red Hat OpenShift Container Platform Cluster Monitoring Operator.

NOTE CPU utilization has minor impact. The ratio is approximately 1 core out of 40 per 50 nodes and 1800 pods.

Recommendations for Red Hat OpenShift Container Platform

- Use at least three infrastructure (infra) nodes.
- Use at least three openshift-container-storage nodes with fast storage drives.

For technical details about the procedure the following documentation can be checked https://docs.openshift.com/container-platform/latest/scalability_and_performance/scaling-cluster-monitoring-operator.html#scaling-cluster-monitoring-operator

5.9.2 Configuration of Monitoring and Logging

The supported way of configuring Red Hat OpenShift Container Platform Monitoring is indicated in the official documentation. Do not use other configurations, as they are unsupported. Configuration paradigms might change across Prometheus releases, and such cases can only be handled gracefully if all configuration possibilities are controlled. If you use configurations other than those described in this section, your changes will disappear because the cluster-monitoring-operator reconciles any differences. The operator reverses everything to the defined state by default and by design.

Explicitly unsupported cases include:

- Creating additional ServiceMonitor objects in the openshift-* namespaces. This extends the targets the cluster monitoring Prometheus instance scrapes, which can cause collisions and load differences that cannot be accounted for. These factors might make the Prometheus setup unstable.
- Creating unexpected ConfigMap objects or PrometheusRule objects. This causes the cluster monitoring Prometheus instance to include additional alerting and recording rules.
- Modifying resources of the stack. The Prometheus Monitoring Stack ensures its resources are always in the state it expects them to be. If they are modified, the stack will reset them.
- Using resources of the stack for your purposes. The resources created by the Prometheus Cluster Monitoring stack are not meant to be used by any other resources, as there are no guarantees about their backward compatibility.
- Stopping the Cluster Monitoring Operator from reconciling the monitoring stack.
- Adding new alerting rules.
- Modifying the monitoring stack Grafana instance.

5.9.3 Configuring persistent storage

Running cluster monitoring with persistent storage means that your metrics are stored to a persistent volume (PV) and can survive a pod being restarted or recreated. This is ideal if you require your metrics or alerting data to be guarded from data loss. For production environments, it is highly recommended to configure persistent storage. Because of the high IO demands, it is advantageous to use local storage.

You can get more information about the Recommended configurable storage technology

Prerequisites

- Dedicate sufficient local persistent storage to ensure that the disk does not become full. How much storage you need depends on the number of pods. For information on system requirements for persistent storage, see Prometheus database storage requirements.
- Make sure you have a persistent volume (PV) ready to be claimed by the persistent volume claim (PVC), one PV for each replica. Because Prometheus has two replicas and Alertmanager has three replicas, you need five PVs to support the entire monitoring stack. The PVs should be available from the Local Storage Operator. This does not apply if you enable dynamically provisioned storage.
- Use the block type of storage.
- Configure local persistent storage.

By default, the Prometheus Cluster Monitoring stack configures the retention time for Prometheus data to be 15 days. You may want to modify the retention time to change how soon the data is deleted.

5.9.4 Alert manager

The Prometheus Alertmanager is a component that manages incoming alerts, including:

- Alert silencing
- Alert inhibition
- Alert aggregation
- Reliable deduplication of alerts
- Grouping alerts
- Sending grouped alerts as notifications through receivers such as email, PagerDuty, and HipChat

Red Hat OpenShift Container Platform monitoring ships with the Watchdog alert, which fires continuously. Alertmanager repeatedly sends notifications for the Watchdog alert to the notification provider, for example, to PagerDuty. The provider is usually configured to notify the administrator when it stops receiving the Watchdog alert. This mechanism helps ensure continuous operation of Prometheus as well as continuous communication between Alertmanager and the notification provider.

Red Hat OpenShift Container Platform Cluster Monitoring by default ships with a set of pre-defined alerting rules. Note that:

- The default alerting rules are used specifically for the Red Hat OpenShift Container Platform cluster and nothing else. For example, you get alerts for a persistent volume in the cluster, but you do not get them for persistent volume in your custom namespace.
- Currently you cannot add custom alerting rules.
- Some alerting rules have identical names. This is intentional. They send alerts about the same event with different thresholds, with different severity, or both.
- With the inhibition rules, the lower severity is inhibited when the higher severity is firing.

For technical details and how to configure the monitoring stack the following link can be checked: https://docs.openshift.com/container-platform/latest/monitoring/configuring-the-monitoring-stack.html

5.9.5 Monitoring your own services

You can use RHOCP monitoring for your own services in addition to monitoring the cluster. This way, you do not need to use an additional monitoring solution. This helps keeping monitoring centralized. Additionally, you can extend the access to the metrics of your services beyond cluster administrators. This enables developers and arbitrary users to access these metrics. You can also export custom application metrics for the horizontal pod autoscaler.

NOTE Opting into monitoring your own services is mutually exclusive with either a custom installation of Prometheus Operator or installing Prometheus Operator using Operator Lifecycle Manager (OLM).

NOTE Monitoring your own services is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

To get the technical details on how to enabling monitoring of your own services you can check enabling monitoring for user-defined projects.

Chapter 6. Application considerations

This chapter explains how RHOCP fits into a modern build and deployment pipeline for agile continuous delivery. We will describe how you can build a typical CI/CD pipeline and coordinate development, deployment, and daily operations of the applications within a RHOCP cluster.

6.1 Source, binaries, images and containers

RHOCP helps you setup a consistent build and deployment pipeline. It facilitates the collaboration of involved teams and establishes an agile DevOps process in your organization. The underlying artifacts include the source code, as well as the app binaries, which get turned into deployable container images. The RHOCP runtime deploys the images and the artifacts around it.

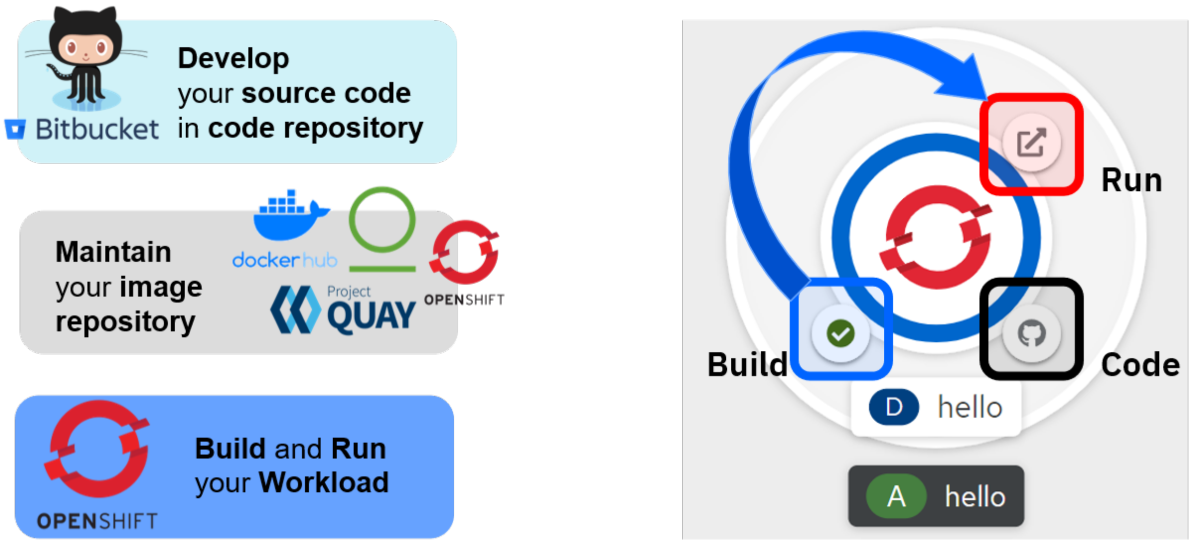

Develop your source code in code repository
Bitbucket
Maintain your image repository
docker hub
Project QUAY
OPENSHIFT
Build and Run your Workload
OPENSHIFT
Run
Build
Code
D hello
A hello

Figure 6.1-1: Build and Deploy Container Images

Let’s have a look at the key steps of the delivery pipeline:

- First, you will need a repository for your source code. GitHub is the most popular tool involved in CI/CD solutions including RHOCP.
- Second, you will need a place where the container images are stored. Remember, in RHOCP deployments, applications are represented as containers, which can be orchestrated in a standardized fashion. RHOCP provides an internal image repository, which typically pulls the container images from an external registry when you use and share containerized software in different environments.
- In the third step, the RHOCP takes care of executing and scaling the containers, which has been created from the source code during this build and deploy process.

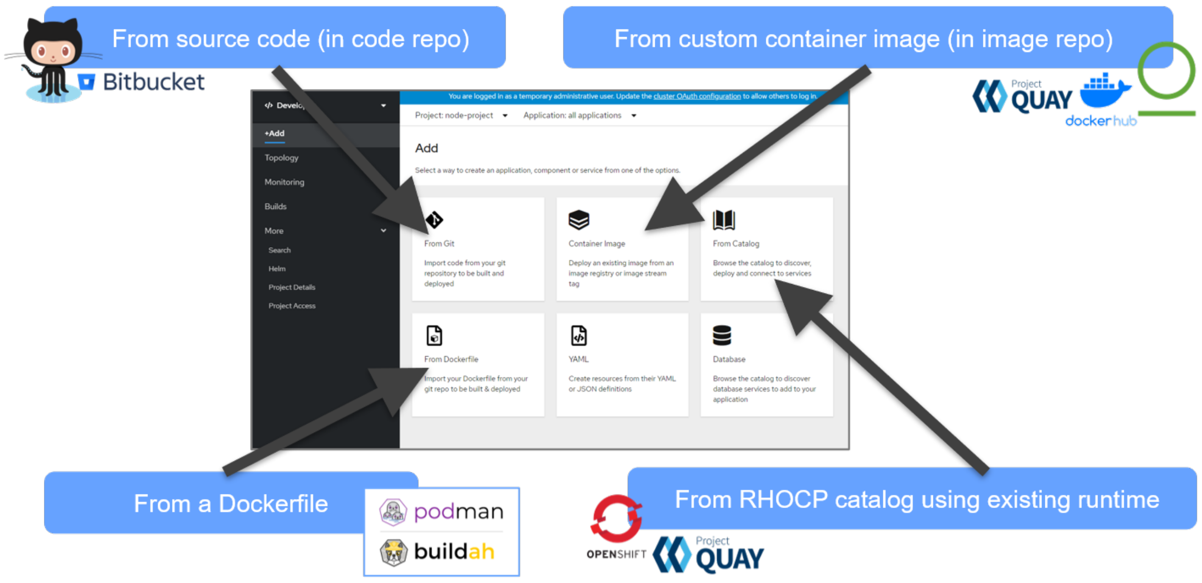
6.2 Red Hat OpenShift Container Platform Web Console

The web console exposes the capabilities, which a DevOps team needs to create a project and deploy source code or container images. In the web console you can start builds and monitor the progress while your application's container instances are being created, deployed, and started to run as workloads in the RHOCP cluster. In addition, the web console allows an organization to easily scale up and down the container instances based on workload demands.

6.3 Simple build and deployment procedures

In this section we will describe four different approaches how to turn your code into a running workload:

- First, you can create a running application by selecting a prepared container workload from the catalog in RHOCP
- Second, you can fetch your own container image from an external image repository, which you maintain for your organization
- Third, you create your application from a podman container definition, for example described by a Dockerfile.
- Finally, you can create the container images and workload directly from source code



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Select a way to create an application, component or service from one of the options.

Import code from your git repository to be built and deployed

Deploy an existing image from an image registry or image stream tag

Browse the catalog to discover, deploy and connect to services

Import your Dockerfile from your git repo to be built & deployed

Create resources from their YAML or JSON definitions

Browse the catalog to discover database services to add to your application

Figure 6.3-1: Creating Applications

The following section will describe each of these options in more detail.

6.3.1 Create an application from the catalog with RHOCP runtimes (base images)

When selecting a prepared base runtime you simply select the starting point of your choice. RHOCP will deploy a corresponding app and run it as workload in the cluster. A number of such RHOCP runtimes exist, e.g. nginx, nodejs, Jboxx, Vert.x, and Spring boot. In order for these runtimes to be available they must first be provisioned by the RHOCP administrator before they will be available in the catalog.

To provision runtimes to the catalog of your RHOCP cluster you need to

- Provide a secret to authenticate to the repository from which you pull the images to your cluster

```
$ oc create secret docker-registry redhat-connect-sso --docker
-server=registry.redhat.io --docker-username=<username> --docker-password=<token>
--docker-email=unused
$ oc secrets link default redhat-connect-sso --for=pull
```

- Create the image in your cluster as shown in the following samples:

```
$ oc create -f <reference to the imagestrean json description> -n openshift
```

- Use the set of base runtimes, which the RHOCP platform provides includes Jboss, vert.x, Node.js, Nginx, PHP and more.
  - https://catalog.redhat.com/software/containers/search?p=1&architecture=s390x
  - https://access.redhat.com/products/Red_Hat_Enterprise_Linux/Developer (Red Hat Software Collections)

Please also refer to this article describing RHOCP Container Catalog Access for s390x:

https://www.linkedin.com/pulse/red-hat-container-catalog-access-270-images-s390x-from-filipe-miranda/

6.3.2 Create an application from custom container images

A very common setup is to establish a container image repository in your organization. This is the place where the output of a build process stores produced container images. RHOCP can pull the images from there and deploy them as workload in the cluster. RHOCP includes an internal image registry. Jfrog Artifactory, DockerHub, Quay are examples of examples of external container image registries, which can be used in combination with RHOCP. In order to establish authentication between RHOCP and an external image registry you will need to create a secret, which is stored in the details of the project description.

6.3.3 Create an application from a Dockerfile (using Podman)

Another option is to use the description of a container, represented as a Dockerfile, and stored on Github. From the provided Dockerfile, RHOCP triggers the building of a container images, stores the image in RHOCP internal image registry and deploys that application workload in the cluster.

6.3.4 Create an application from source code directly

The RHOCP platform also allow to link the source code directly with its execution as workload in the cluster. During app creation you can specify a corresponding Github repository. If the repo is private, you need to specify a secret to give RHOCP access the code. During deployment, RHOCP builds the code and create container images which are stored in the internal image registry of RHOCP. RHOCP then deploys the containers into the cluster. After code updates, you can trigger new builds directly from the web console. The running containers will be replaced by new containers reflecting the updated code one after the other. With the help of webhooks you can also configure a notification mechanism by Github to trigger a fresh build in RHOCP automatically.

6.4 Development best practices

In the samples we went through so far, the focus was on understanding the basic principles, when dealing with code and containers.

For completeness, the full architecture of an elaborate RHOCP pipeline is depicted in the chart below. Most notable is the very clear split between development, test and production environments. Between each of the environments quality gates need to be in place. The external and internal RHOCP image registry is the binding link between the different environments.

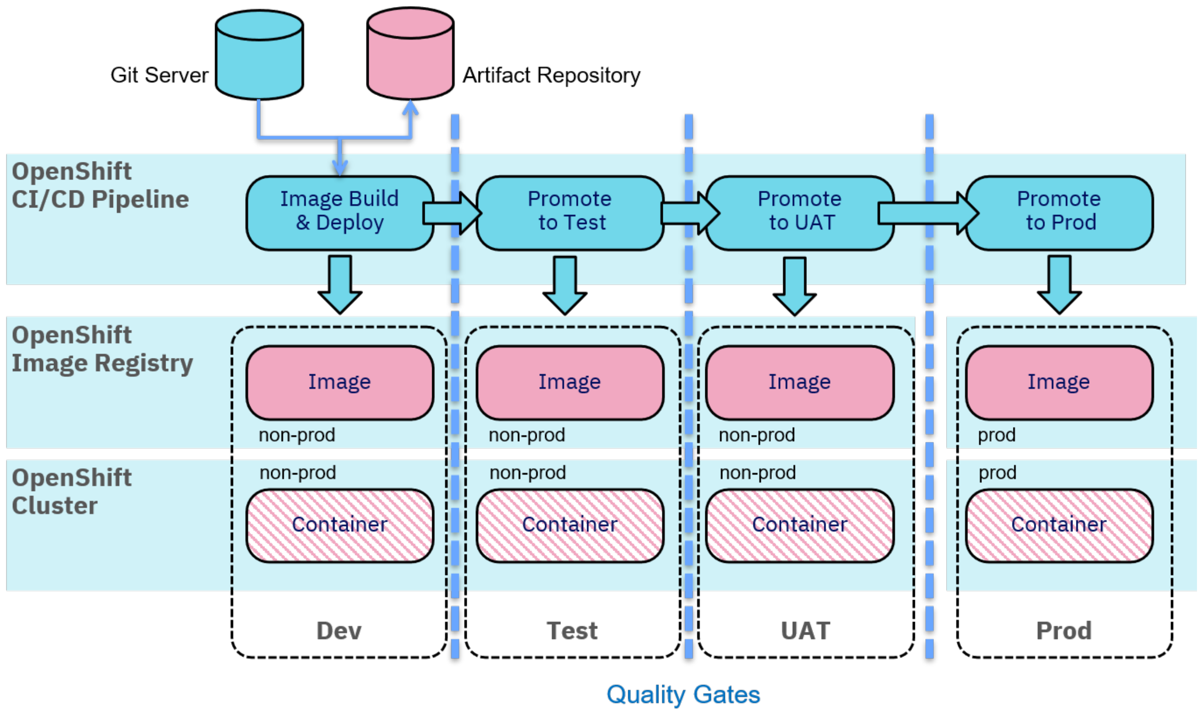


Figure 6.4-1: Build and Deploy Pipeline

Integration with existing CI/CD pipelines for build and deployment is a major focus of future releases of RHOCP.

Chapter 7. Summary

With Red Hat OpenShift Container Platform (RHOCP), organizations have access to a comprehensive and prescriptive experience for their Kubernetes based container infrastructure. This document helps to ensure that your solution meets best-practices and recommendations. The Red Hat Quality Engineering teams (QE) have tested and validated the consideration described in this document. This ensures organizations seeking to operationalize this reference architecture quickly can be assured that all options represented are both fully tested as well as fully supported by Red Hat.

In chapter 3 a solid understanding of the overarching concepts and architectural principles of RHOCP has been sketched out. This is the essential context technical leadership needs to have to guide real-life implementation projects towards a successful solution.

In chapter 4 the key design considerations have been detailed. This enables the technical team to make the correct decisions on networking, storage and installation options based on a solid understanding of the RHOCP concepts. This also helps to avoid mistakes which need cumbersome mitigation in later stages of a project.

Chapter 5 focuses on aspects of "Day 2" operations like security, persistence, backup, and performance tuning. This chapter enables you to become productive quickly. It also helps to make your solution reliable and secure right from the beginning.

Chapter 6 outlines the actual development process. At this time, your actual application workload will move on your prepared container infrastructure to deliver the actual value. This chapter shows how you can build CI/CD pipelines for an agile and efficient DevOps implementation.

As RHOCP evolves in the future and provides new capabilities, this document will be updated on a regular basis to include the latest best practices and consideration.

With the provided information of this document we encourage you to proceed with your RHOCP based project and deliver successful solutions to your customers.

Chapter 8. Appendix

This section includes some examples of commands and code to be used.

8.1 Monitoring during the installation

While RHOCP is being installed the cluster progress can be monitored with the following command:

```
$ openshift-install --dir=<installation_directory> wait-for install-complete ①
INFO Waiting up to 30m0s for the cluster to initialize...
```

① For <installation_directory>, specify the path to the directory that you stored the installation files in.

8.2 Validations post installation: checking cluster operators, cluster version and nodes

To confirm that all the cluster components are online, the cluster operators can be checked with the following command:

```
$ oc get clusteroperators
NAME                                       VERSION   AVAILABLE   PROGRESSING
DEGRADED   SINCE
cloud-credential                           4.4.20    True        False         False
37m
dns                                        4.4.20    True        False         False
6m57s
etcd                                       4.4.20    True        False         False
5m52s
...
```

Each line shows the *name*, *version*, and if *available*, *progressing* or *degraded* status. Also, the *since* column indicates how long the operator has been in the *Available*, *Progressing*, or *Degraded* status. The conditions to meet for successful installation are having the *Available* column set to true and the *Degraded* column to false. In the case any of the operators have the *Progressing* value set to True, it means that this operator didn't finish with the installation process.

The following command can be executed to determine if the installation process has completed successfully:

```
$ oc get clusterversion
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.4.20    True        False         12m     Cluster version is 4.4.20
```

This command shows the version of the installation and the actual status of the installation or upgrade.

If an error condition occurs the following command will return an error like the example below:

```
$ oc get clusterversion
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version             False       True          40m     Unable to apply 4.4.20: the
control plane is down or not responding
```

Also the nodes can be checked with the following command:

```
$ oc get nodes
NAME                                                 STATUS   ROLES               AGE
VERSION
rarch-master-0.rarch.redhat.com                      Ready    master              38m
v1.17.1
rarch-master-1.rarch.redhat.com                      Ready    master              38m
v1.17.1
rarch-master-2.rarch.redhat.com                      Ready    master              38m
v1.17.1
rarch-worker-0.rarch.redhat.com                      Ready    worker              38m
v1.17.1
rarch-worker-1.rarch.redhat.com                      Ready    worker              38m
v1.17.1
rarch-worker-2.rarch.redhat.com                      Ready    worker              38m
v1.17.1
```

In the case there is a node missing on the previous list output it could be necessary to check the pending Certificate Signing Requests (CSR). To check for pending CSRs the following can be executed:

```
$ oc get csr
```

And the CSRs can be approved with the following command:

```
$ oc get csr -o name | xargs -n 1 oc adm certificate approve
```

For a healthy node the *STATUS* column has to be *Ready*, otherwise the node details can be checked with the following command:

```
$ oc describe node rarch-master-0.rarch.redhat.com
```

The following checks are usefull after the installation:

- Check that all servers have joined the cluster → shows failures, because of pending certificates, which must be approved manually.
- The image-registry does not come up because no shared persistent storage was provided for the image registry during installation → temporary fix: attach localHost storage and add shared persistent storage later.
- By default, application containers can also run on masters → this can be changed by changing the scheduler configuration.
- In some installations the authentication operator does not start properly.
  - This prevents the console operator to work properly and login at the console is not possible → this can be fixed by deleting the ingress controller.

8.3 Troubleshooting commands

Check the status of the pods in all namespaces

If an error occurred during the installation and the oc get cluster version command returns and error, then the following command can be executed to determine the status of pods in all namespaces:

```
$ oc get pods --all-namespaces

NAMESPACE                         NAME
READY   STATUS      RESTARTS   AGE
openshift-apiserver-operator      openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running     1          9m
openshift-apiserver               apiserver-67b9g                                 1/1
Running     0          3m
openshift-apiserver               apiserver-ljcmx                                 1/1
Running     0          1m
openshift-apiserver               apiserver-z25h4                                 1/1
Running     0          2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

Checking the logs of a pod

The logs for a Pod that is listed in the output of the previous command can be checked by using the following command:

```
$ oc logs <pod_name> -n <namespace>
```

Gathering debug information

Debugging information that might help to troubleshoot and debug certain issues with an Red Hat OpenShift Container Platform installation on IBM Z can be gathered.

```
$ oc debug node/<nodename>
chroot /host
toolbox
dbginfo.sh
```

The scp command can be used to retrieve the data from the RHOCP node.

8.4 Persistent storage examples

On platforms that do not provide shareable object storage, the RHOCP Image Registry Operator bootstraps itself as Removed. This allows openshift-installer to complete installations on these platform types. After installation, you must edit the Image Registry Operator configuration to switch the managementState from Removed to Managed.

The following example shows how to define a NFS based Persistent Volume to use it for the registry.

Persistent Volume Example

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: ocp-registry
spec:
  capacity:
    storage: 100Gi
  accessModes:
  - ReadWriteMany
  nfs:
    path: /mnt/ocp4-registry ①
    server: 172.17.0.2 ②
```

① The nfs path points to the directory where the 100Gi device has been mounted

② The nfs server is the IP address of the NFS server in this example

```
$ oc apply -f registry_pv.yaml -n openshift-image-registry
```

Persistent Volume Claim example (optional)

The following example shows how to create a Persistent Volume Claim with *ReadWriteMany* access mode and 100Gi of storage.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: registry-claim ①
  namespace: openshift-image-registry ②
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
```

① This PVC is called registry-claim

② This PVC is created in the namespace openshift-image-registry

Use the *oc apply* command to create the persistent volume as shown below:

```
$ oc apply -f registry_pv.yaml -n openshift-image-registry
```

Starting with RHOCP 4.3 there is no need to create the PVC. The registry operator will take care of creating the required PVC. The only requirements are to edit the operator as described earlier and shown below.

Editing the image registry operator to use the persistent storage

```
$ oc edit configs.imageregistry.operator.openshift.io
```

The section storage needs to be replaced with the following:

```
...
storage:
  pvc:
    claim:
```

If you want to control the claim, you can create it manually and use it like the following example. The section "storage:" has to be replaced with the following:

```
...
storage:
  pvc:
    claim: registry-claim
```

The ManagementState has to be "Managed":

```
  managementState: Managed ①
```

① By default the *managementState* is se to *Removed* and must be changed to *Managed* as shown in the example above.

Once that is done the image registry operator will automatically apply the configuration and it can be verified by typing:

```
$ oc get co
```

The image-registry cluster operator should look like:

```
$ image-registry                                 4.2.29    True        False         False
15m ①
```

① For this example Red Hat OpenShift 4.2.29 was used, the reader may have a different version

Dynamic persistent storage using NFS

The first step is to define the NFS directories for each persistent storage definition:

```
$ for i in {0..20};do mkdir /mnt/nfs/share$i -p; chmod 777 /mnt/nfs/share$i; done
```

Next, add the new directories to the NFS server exports configuration:

```
/mnt/nfs *(rw,no_subtree_check,sync,no_wdelay,insecure,no_root_squash)
```

WARNING Do not use "*" after the directory path, restrict it to appropriate subnet in production.

The NFS service must be restarted so the changes will take affect:

```
# systemctl restart nfs-server
```

After preparing the new NFS resources a new StorageClass is going to be defined in RHOCP (save the below code as sc.yaml):

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: nfs
  provisioner: no-provisioning
  parameters:
```

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

And now the new NFS storage class must be patched to indicated that it is the default storage class as shown in the command below:

```
$ oc patch storageclass nfs -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class": "true"}}}'
```

A file with the definition of the Persistent Storage Volumes is created. This file includes 20 volumes ReadWriteMany. It may be adjusted as necessary to meet the needs of each case:

```
$ for i in {01..20}; do \
cat <<EOF >> volumes.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
   name: vol${i}
spec:
   capacity:
      storage: 20Gi
   accessModes:
      - ReadWriteMany
   persistentVolumeReclaimPolicy: Recycle
   storageClassName: nfs
   nfs:
      path: /mnt/nfs/share${i}
      server: <nfs_server_IP_address>
---
EOF
done
```

The previous code generates a file saved as *volumes.yaml* that can be applied with the following *oc command*:

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

To verify the creation of the persistent volumes the following command can be executed:

```
$ oc get pv

NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
image-registry-pv   100Gi      RWX            Retain           Bound       openshift-
image-registry/image-registry-storage   nfs                     2d21h
vol01               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol02               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol03               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol04               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol05               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol06               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol07               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol08               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol09               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol10               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol11               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol12               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol13               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol14               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol15               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol16               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol17               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol18               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol19               20Gi       RWX            Recycle          Available
nfs                     2d21h
vol20               20Gi       RWX            Recycle          Available
nfs                     2d21h
```

The default storage class can be verified with the following command *oc command*:

```
$ oc get storageclass
NAME            PROVISIONER       AGE
nfs (default)   no-provisioning   2d21h
```

8.5 Adding a new node

The following examples show how to configure the DNS and a HAProxy load balancer for adding a new node. Additionaly the definition of a new VM is being created.

DNS configuration example

```
...

compute0    IN   A   <compute0-IP-address>
compute1    IN   A   <compute1-IP-address>
compute2    IN   A   <compute2-IP-address>

etcd-0.<cluster_name>  IN   A   <master0-IP-address>
etcd-1.<cluster_name>  IN   A   <master1-IP-address>
etcd-2.<clsuter_name>  IN   A   <master2-IP-address>

...
```

In addition, the compute2 entry has to be included into the Reverse Zone Records file:

```
...

<XX>       IN      PTR     master2.<domain>.
<XX>       IN      PTR     master3.<domain>.
<XX>       IN      PTR     compute0.<domain>.
<XX>       IN      PTR     compute1.<domain>.
<XX>       IN      PTR     compute2.<domain>.
<XX>       IN      PTR     workstation.<domain>.

...
```

Where *<XX>* for each record will be the last octet of their IP addresses.

HAProxy load balancer configuration example

In this example HAProxy is used as a load balancer, the configuration has to be modified as shown below:

```
...

listen ingress-http

    bind *:80
    mode tcp

    server worker0 <compute0-IP-address>:80 check
    server worker1 <compute1-IP-address>:80 check
    server worker2 <compute2-IP-address>:80 check ①

listen ingress-https

    bind *:443
    mode tcp

    server worker0 <compute0-IP-address>:443 check
    server worker1 <compute1-IP-address>:443 check
    server worker2 <compute2-IP-address>:443 check ②

...
```

① Includes the new compute node to listen as a ingress-http

② Includes the new compute node to listen as a ingress-https

New VM is being created

The new VM uses zFCP and not ECKD for the installation of Red Hat CoreOS.

- You do not need to have a base 191 disk in z/VM.
- To ipl RHCOS, the WWPN port name and LUN are needed.

```
 USER worker2 worker2 8G 16G G
  INCLUDE LNXDFLT
  OPTION LNKNOPAS APPLMON
  COMMAND DEFINE STORAGE 8G STANDBY 8G
  LOADDEV PORTNAME 5005073603481CAA
  LOADDEV LUN 4010400F00000000
  DEDICATE 010A A000
```

An example of the LNXDFLT could be:

```
PROFILE LNXDFLT
  IPL CMS
  COMMAND SET RUN ON
  CPU 00 BASE
  CPU 01
  CPU 02
  CPU 03
  MACHINE ESA 6
  NICDEF 500 TYPE QDIO LAN SYSTEM VSW1
  NICDEF 700 TYPE QDIO LAN SYSTEM VSW1
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  CONSOLE 009 3215 T
  LINK MAINT 0190 0190 RR
  LINK MAINT 019D 019D RR
  LINK MAINT 019E 019E RR
  LINK MAINT 0402 0402 RR
  LINK MAINT 0401 0401 RR
  LINK LNXMAINT 192 191 RR
  LINK TCPMAINT 592 592 RR
```

NOTE The provided examples are for been used as templates, the user may adjust them according to its installations standards.

Content of the Compute PARM file An example of the content of the compute parm file looks like the following:

```
rd.neednet=1 coreos.inst=yes
coreos.inst.install_dev=<sda> ①
coreos.inst.image_url=http://<http-server>/rhcos-4.2.18.raw.gz
coreos.inst.ignition_url=http://<http-server>/worker.ign
vlan=eth0.<1100>:<enc1e00>
ip=<bootstrap-IP-address>::<gateway>:<netmask>:<worker2-hostname>:eth0.<1100>:off
nameserver=<nameserver-IP-address>
rd.znet=qeth,<0.0.1f00>,<0.0.1f01>,<0.0.1f02>,layer2=1,portno=0
cio_ignore=all,!condev
rd.zfcp=<0.0.010A>,<0x5005076810171FA3>,<0x0000000000000000> ②
zfcp.allow_lun_scan=0 ③
```

① The Red Hat CoreOS will recognize FCP devices as SCSI devices, so sda has to be indicated

② This is the order of the parameters to properly specify the FCP device. Currently Red Hat CoreOS only supports single path. Support for multipath is planned for a future release.

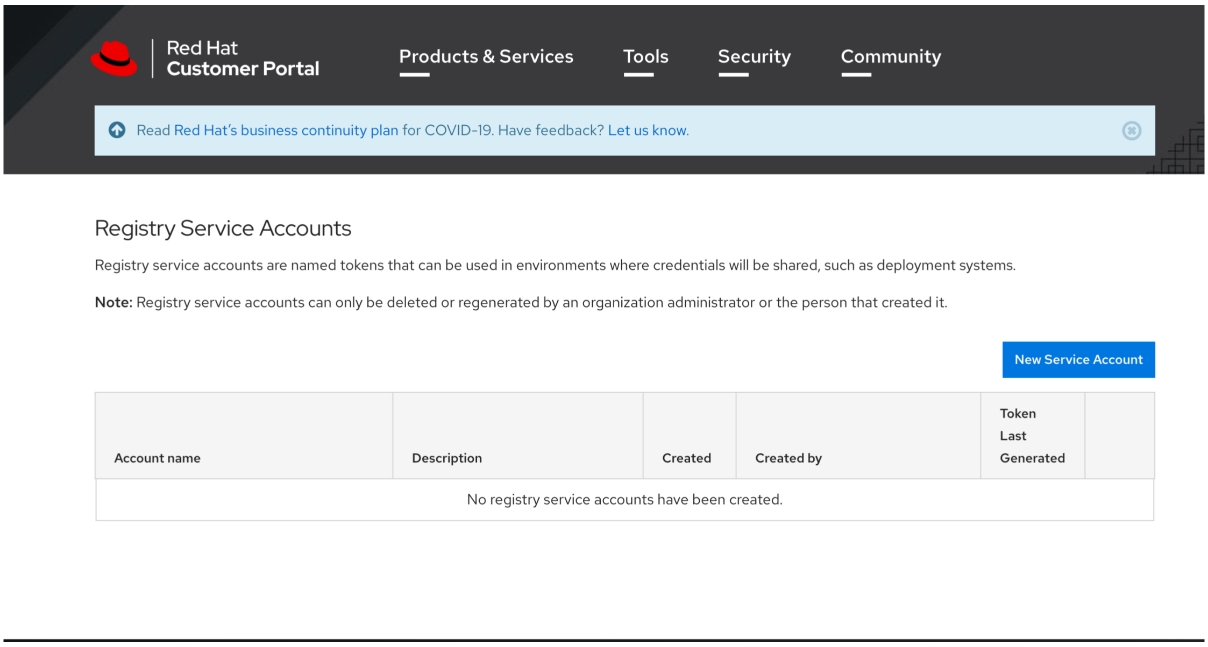
③ This parameter prevents the scanning of LUNs for the specified FCP device. This is a requirement as per in the Red Hat official documentation.

8.7 RHOCP container catalog access for s390x

8.7.1 Step 1: Registering a service account

The following URL is used to register the service account, the Red Hat credentials are requested. Then the option "New Service Account" has to be selected:

Then the option "New Service Account" has to be selected:


Red Hat Customer Portal
Products & Services
Tools
Security
Community
Read Red Hat's business continuity plan for COVID-19. Have feedback? Let us know.

Registry Service Accounts

Registry service accounts are named tokens that can be used in environments where credentials will be shared, such as deployment systems.

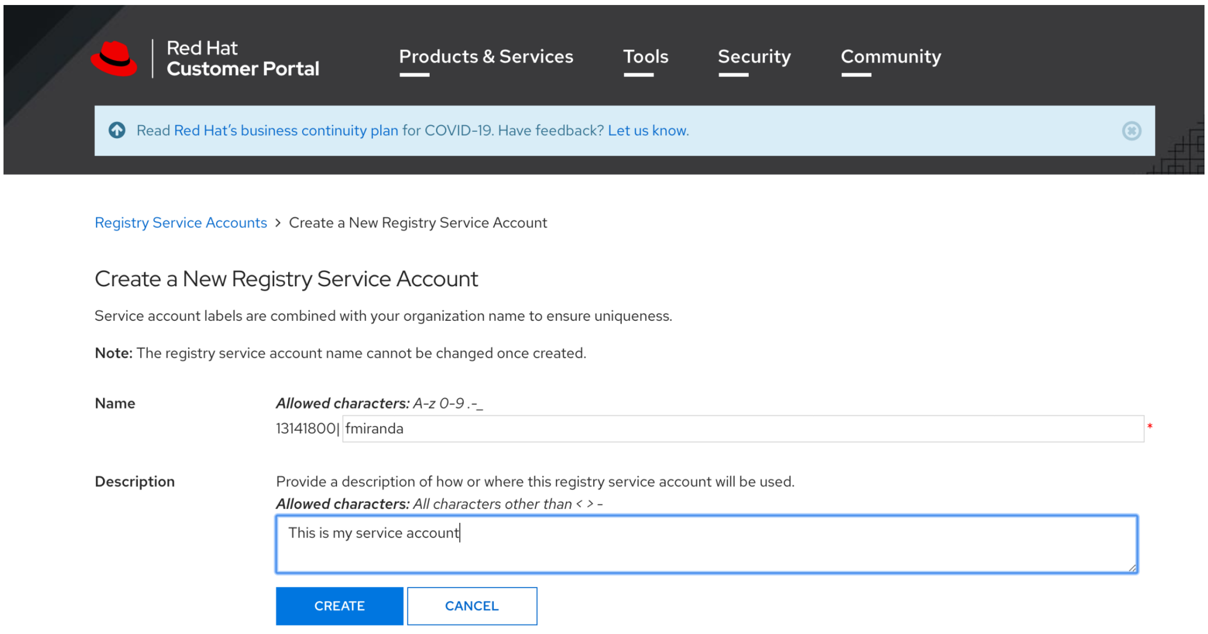
Note: Registry service accounts can only be deleted or regenerated by an organization administrator or the person that created it.

New Service Account

Account name	Description	Created	Created by	Token Last Generated	
No registry service accounts have been created.					

Figure 8.7-1: New Service Account

The Name and Description input fields have to be completed:


Red Hat Customer Portal
Products & Services
Tools
Security
Community
Read Red Hat's business continuity plan for COVID-19. Have feedback? Let us know.

Registry Service Accounts > Create a New Registry Service Account

Create a New Registry Service Account

Service account labels are combined with your organization name to ensure uniqueness.

Note: The registry service account name cannot be changed once created.

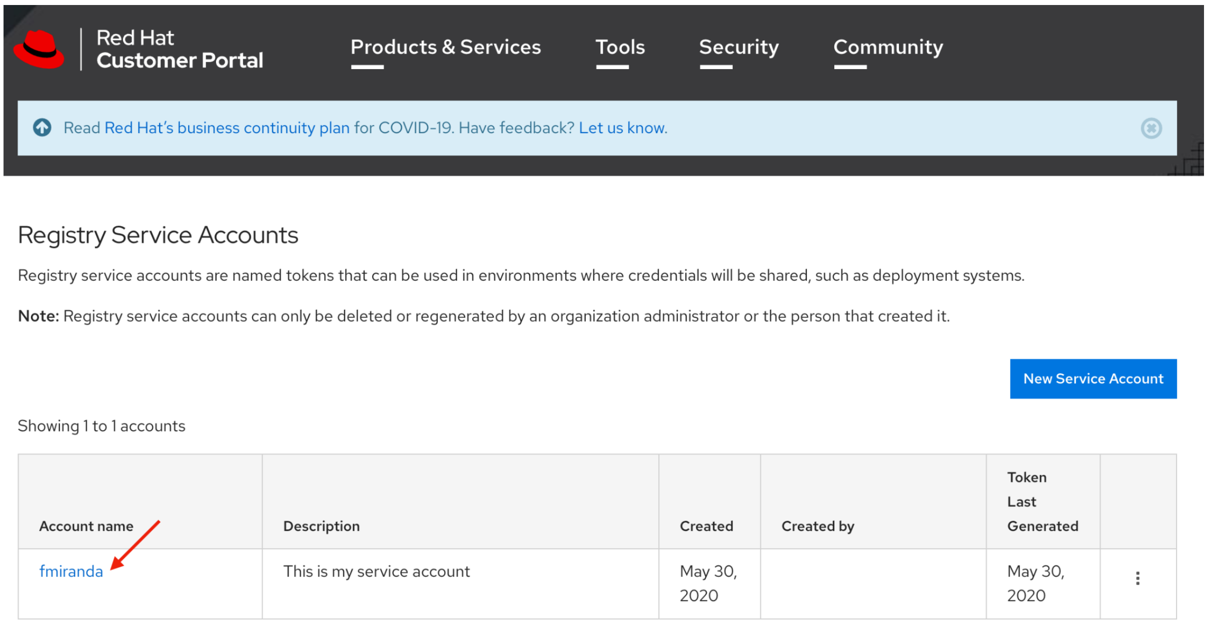
Name — ***Allowed characters:*** *A-z 0-9 .-_*
13141800| fmiranda *

Description — Provide a description of how or where this registry service account will be used.
Allowed characters: *All characters other than < > -*
This is my service account

CREATE CANCEL

Figure 8.7-2: Input fields to be filled

The next step is to choose the recently created account:



Red Hat Customer Portal
Products & Services
Tools
Security
Community
Read Red Hat's business continuity plan for COVID-19. Have feedback? Let us know.

Registry Service Accounts

Registry service accounts are named tokens that can be used in environments where credentials will be shared, such as deployment systems.

Note: Registry service accounts can only be deleted or regenerated by an organization administrator or the person that created it.

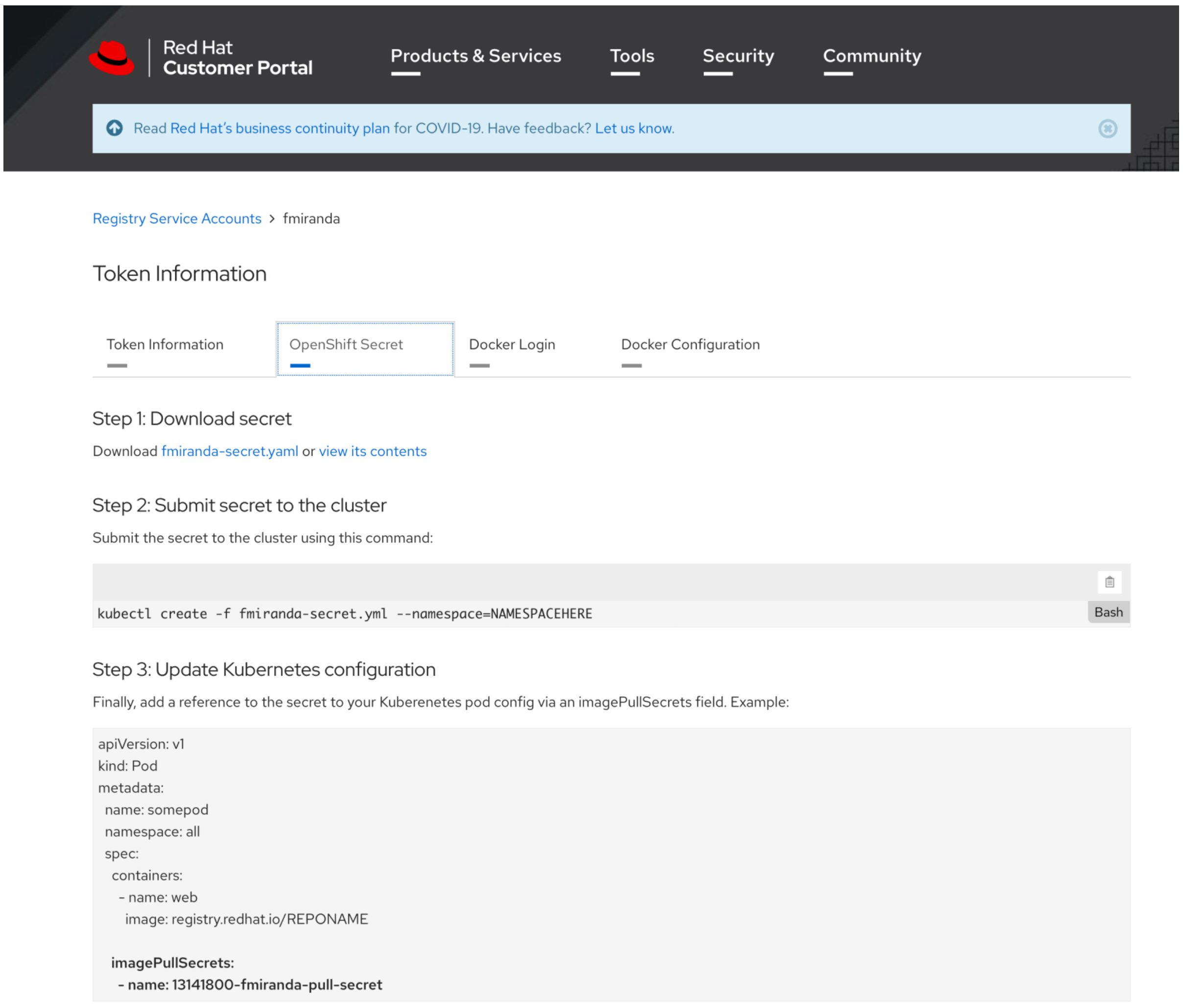
New Service Account

Showing 1 to 1 accounts

Account name	Description	Created	Created by	Token Last Generated	
fmiranda	This is my service account	May 30, 2020		May 30, 2020	⋮

Figure 8.7-3: Choose the created account

The Red Hat **OpenShift** tab has to be selected:



Token Information

Step 1: Download secret

Download fmiranda-secret.yaml or view its contents

Step 2: Submit secret to the cluster

Submit the secret to the cluster using this command:

```
kubectl create -f fmiranda-secret.yml --namespace=NAMESPACEHERE
```

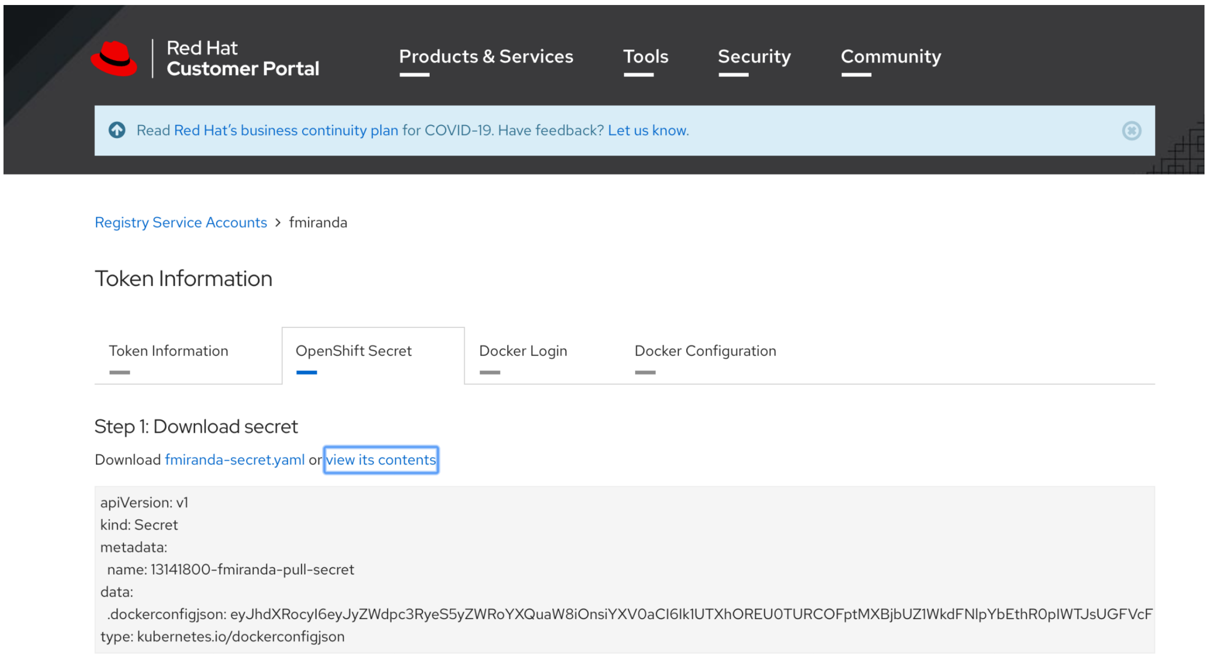
Step 3: Update Kubernetes configuration

Finally, add a reference to the secret to your Kuberenetes pod config via an imagePullSecrets field. Example:

```
apiVersion: v1
kind: Pod
metadata:
  name: somepod
  namespace: all
spec:
  containers:
    - name: web
      image: registry.redhat.io/REPONAME

  imagePullSecrets:
    - name: 13141800-fmiranda-pull-secret
```

Figure 8.7-4: Select the Red Hat OpenShift tab

Select the "View its Contents":



Step 1: Download secret

Download fmiranda-secret.yaml or view its contents

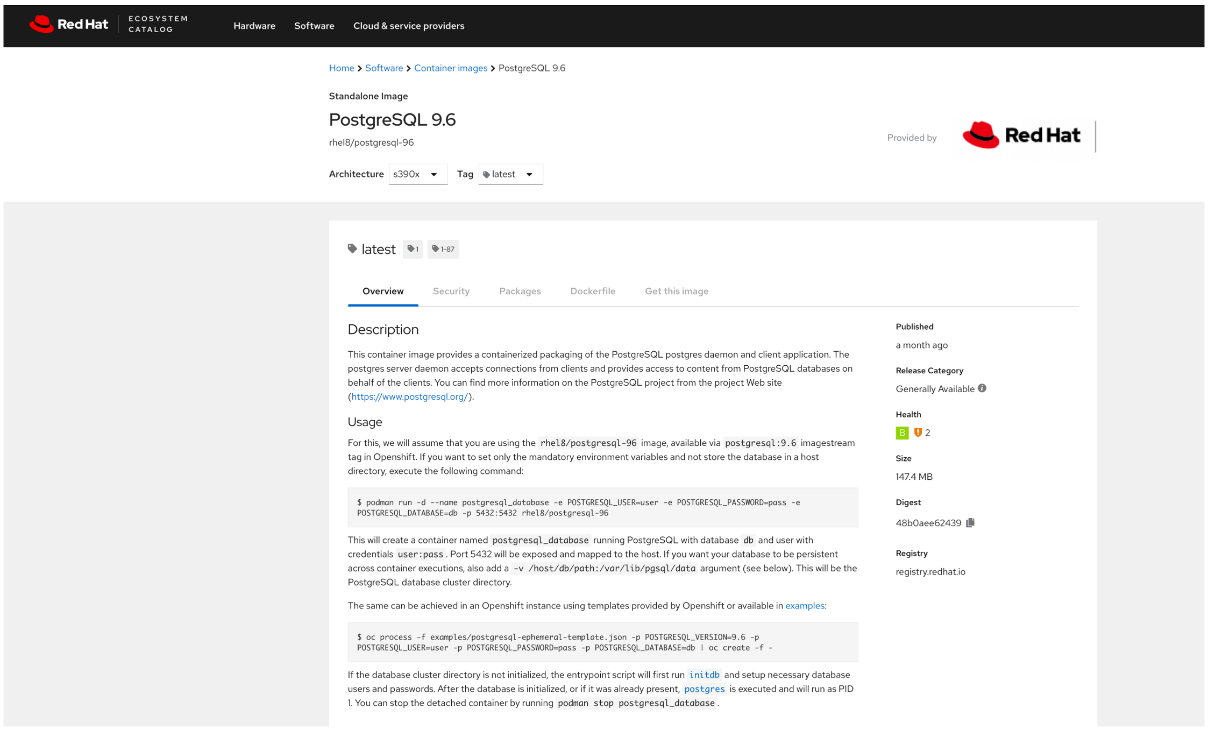
```
apiVersion: v1
kind: Secret
metadata:
  name: 13141800-fmiranda-pull-secret
data:
  .dockerconfigjson: eyJhdXRocyI6eyJyZWdpc3RyeS5yZWRoYXQuaW8iOnsiYXV0aCI6Ik1UTXhOREU0TURCOFptMXBjbUZ1WkdFNlpYbEthR0pIWTJsUGFVcF
type: kubernetes.io/dockerconfigjson
```

Figure 8.7-5: View content

The figure above shows the secret definition for the Registry Service Account recently created. Uploading this secret definition to a namespace (project) on RHOCP enables that project to access and download any content from the Red Hat Container Catalog. This process is shown on Step 2.

8.7.2 Step 2: Creating a project, importing the account secret and selecting a container image to use

To test the catalog a popular container image can be selected from the available container images, in this example the image of PostgreSQL9.6 is chosen:


Red Hat ECOSYSTEM CATALOG
Hardware
Software
Cloud & service providers
Home > Software > Container images > PostgreSQL 9.6
Provided by Red Hat
latest
1
1-87
Overview
Security
Packages
Dockerfile
Get this image

Standalone Image

PostgreSQL 9.6

rhel8/postgresql-96

Architecture s390x Tag latest

Description

This container image provides a containerized packaging of the PostgreSQL postgres daemon and client application. The postgres server daemon accepts connections from clients and provides access to content from PostgreSQL databases on behalf of the clients. You can find more information on the PostgreSQL project from the project Web site (https://www.postgresql.org/).

Usage

For this, we will assume that you are using the `rhel8/postgresql-96` image, available via `postgresql:9.6` imagestream tag in Openshift. If you want to set only the mandatory environment variables and not store the database in a host directory, execute the following command:

```
$ podman run -d --name postgresql_database -e POSTGRESQL_USER=user -e POSTGRESQL_PASSWORD=pass -e
POSTGRESQL_DATABASE=db -p 5432:5432 rhel8/postgresql-96
```

This will create a container named `postgresql_database` running PostgreSQL with database `db` and user with credentials `user:pass`. Port 5432 will be exposed and mapped to the host. If you want your database to be persistent across container executions, also add a `-v /host/db/path:/var/lib/pgsql/data` argument (see below). This will be the PostgreSQL database cluster directory.

The same can be achieved in an Openshift instance using templates provided by Openshift or available in examples:

```
$ oc process -f examples/postgresql-ephemeral-template.json -p POSTGRESQL_VERSION=9.6 -p
POSTGRESQL_USER=user -p POSTGRESQL_PASSWORD=pass -p POSTGRESQL_DATABASE=db | oc create -f -
```

If the database cluster directory is not initialized, the entrypoint script will first run `initdb` and setup necessary database users and passwords. After the database is initialized, or if it was already present, `postgres` is executed and will run as PID 1. You can stop the detached container by running `podman stop postgresql_database`.

Published
a month ago

Release Category
Generally Available

Health
B 2

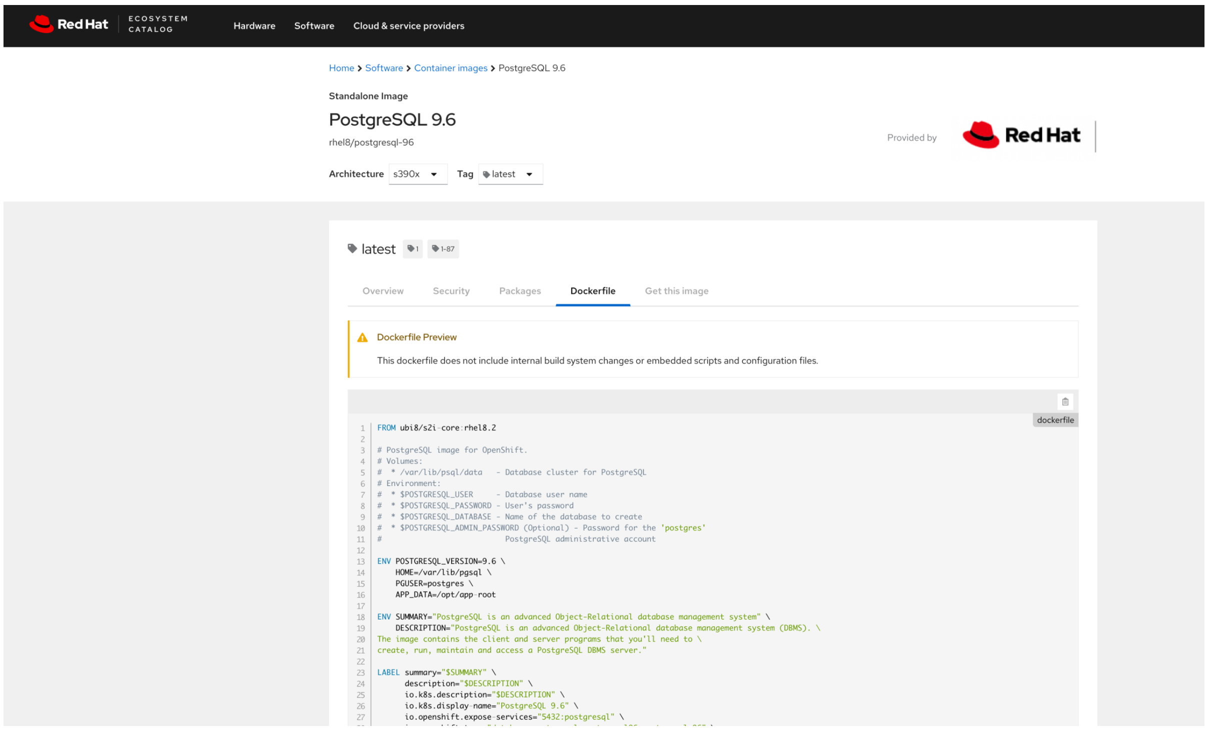
Size
147.4 MB

Digest
48b0aee62439

Registry
registry.redhat.io

Figure 8.7-6: Select a popular container image

The PostgreSQL container definition requires a few variables to be set during deployment:


Red Hat ECOSYSTEM CATALOG
Hardware
Software
Cloud & service providers
Home > Software > Container images > PostgreSQL 9.6
Provided by Red Hat
Overview
Security
Packages
Dockerfile
Get this image
dockerfile

Standalone Image

PostgreSQL 9.6

rhel8/postgresql-96

Architecture s390x Tag latest

latest 1 1-87

Dockerfile Preview

This dockerfile does not include internal build system changes or embedded scripts and configuration files.

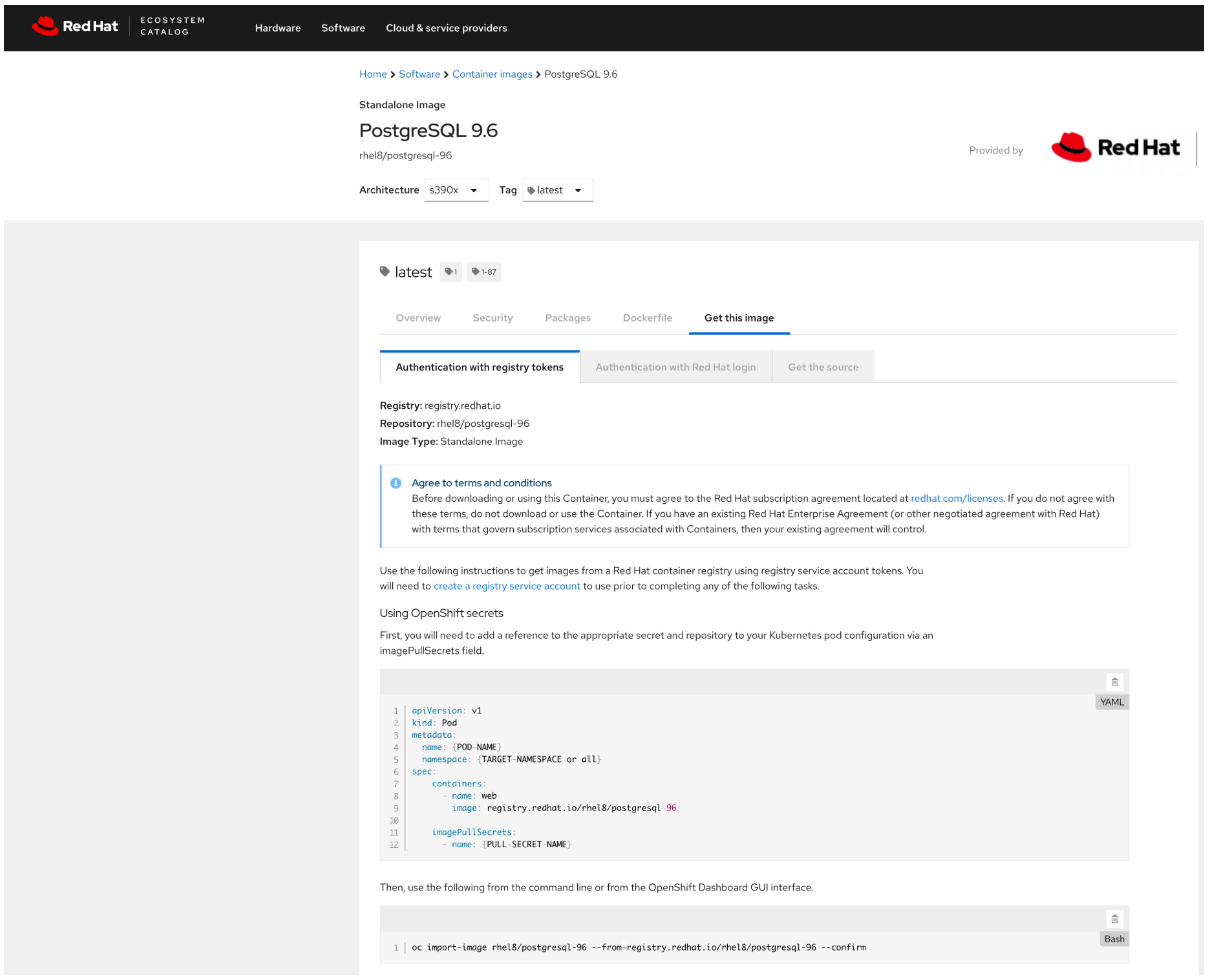
```
FROM ubi8/s2i-core:rhel8.2

# PostgreSQL image for OpenShift.
# Volumes:
#  * /var/lib/psql/data   - Database cluster for PostgreSQL
# Environment:
#  * $POSTGRESQL_USER     - Database user name
#  * $POSTGRESQL_PASSWORD - User's password
#  * $POSTGRESQL_DATABASE - Name of the database to create
#  * $POSTGRESQL_ADMIN_PASSWORD (Optional) - Password for the 'postgres'
#                           PostgreSQL administrative account

ENV POSTGRESQL_VERSION=9.6 \
    HOME=/var/lib/pgsql \
    PGUSER=postgres \
    APP_DATA=/opt/app-root

ENV SUMMARY="PostgreSQL is an advanced Object-Relational database management system" \
    DESCRIPTION="PostgreSQL is an advanced Object-Relational database management system (DBMS). \
The image contains the client and server programs that you'll need to \
create, run, maintain and access a PostgreSQL DBMS server."

LABEL summary="$SUMMARY" \
      description="$DESCRIPTION" \
      io.k8s.description="$DESCRIPTION" \
      io.k8s.display-name="PostgreSQL 9.6" \
      io.openshift.expose-services="5432:postgresql" \
```

Figure 8.7-7: PostgreSQL image variables

To download this container image, the following command line has to be used from a client machine that has access to the RHOCP cluster and has been previously logged in:

```
$ oc import-image rhel8/postgresql-96 --from=registry.redhat.io/rhel8/postgresql-96
--confirm
```

This information can be found by selecting the tab "Get this Image":


Red Hat ECOSYSTEM CATALOG
Hardware
Software
Cloud & service providers
Home > Software > Container images > PostgreSQL 9.6

Standalone Image

PostgreSQL 9.6

rhel8/postgresql-96

Provided by Red Hat
Architecture s390x
Tag latest
latest 1 1-87
Overview
Security
Packages
Dockerfile
Get this image
Authentication with registry tokens
Authentication with Red Hat login
Get the source

Registry: registry.redhat.io
Repository: rhel8/postgresql-96
Image Type: Standalone Image

Agree to terms and conditions
Before downloading or using this Container, you must agree to the Red Hat subscription agreement located at redhat.com/licenses. If you do not agree with these terms, do not download or use the Container. If you have an existing Red Hat Enterprise Agreement (or other negotiated agreement with Red Hat) with terms that govern subscription services associated with Containers, then your existing agreement will control.

Use the following instructions to get images from a Red Hat container registry using registry service account tokens. You will need to create a registry service account to use prior to completing any of the following tasks.

Using OpenShift secrets

First, you will need to add a reference to the appropriate secret and repository to your Kubernetes pod configuration via an imagePullSecrets field.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: {POD-NAME}
  namespace: {TARGET-NAMESPACE or all}
spec:
    containers:
      - name: web
        image: registry.redhat.io/rhel8/postgresql-96

    imagePullSecrets:
      - name: {PULL-SECRET-NAME}
```

Then, use the following from the command line or from the OpenShift Dashboard GUI interface.

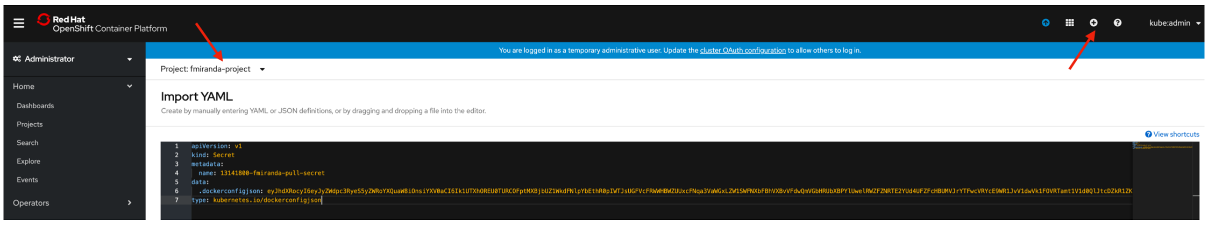
```bash
oc import-image rhel8/postgresql-96 --from=registry.redhat.io/rhel8/postgresql-96 --confirm
```

Figure 8.7-8: PostgreSQL image command

Before downloading the PostgreSQL container image, a project needs to be created, which will use the Service Account Secret created on Step 1:

```
$ oc new-project fmiranda-project
```

Once the project has been created, login to the RHOCP web console and the + (plus) sign at the top right corner has to be selected. It is important to verify that the project created has been selected and the Account Secret created previously from Step 1 can be pasted as shown in the figure below:



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

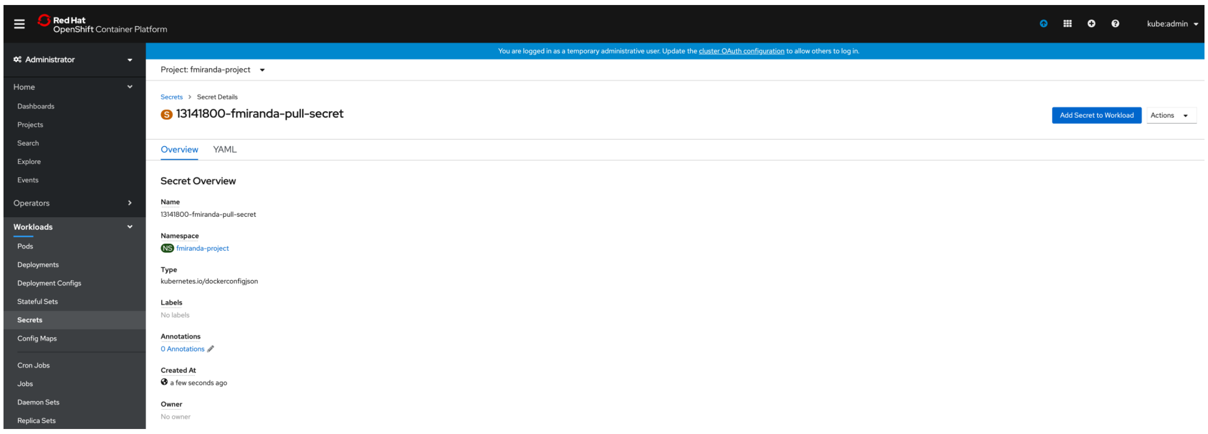
Import YAML

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: 13141800-fmiranda-pull-secret
data:
  .dockerconfigjson: eyJhdXRocyI6eyJyZWdpc3RyeS5yZWRoYXQuaW8iOnsiYXV0aCI6Ik1UTXhOREU0TURCOFptMXBjbUZ1WkdFNlpYbEthR0pIWTJsUGFVcFRWWHBWZUUxcFNqa3VaWGxLZW1SWFNXbFBhVXBvVFdwQmVGbHRUbXBPYlUwelRWZFZNRTE2YUd4UFZFcHBUMVJrYTFwcVRYcE9WR1JvV1dwVk1FOVRamt1V1d0QlJtcDZkR1ZK
type: kubernetes.io/dockerconfigjson
```

Figure 8.7-9: Import secret YAML in the project

The installed Account Secret can be verified selecting *Workloads* on the left menu, then *Secrets*, and then the correct project:



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: fmiranda-project

13141800-fmiranda-pull-secret

Secret Overview

Name
13141800-fmiranda-pull-secret

Namespace
NS fmiranda-project

Type
kubernetes.io/dockerconfigjson

Labels
No labels

Annotations
0 Annotations

Created At
a few seconds ago

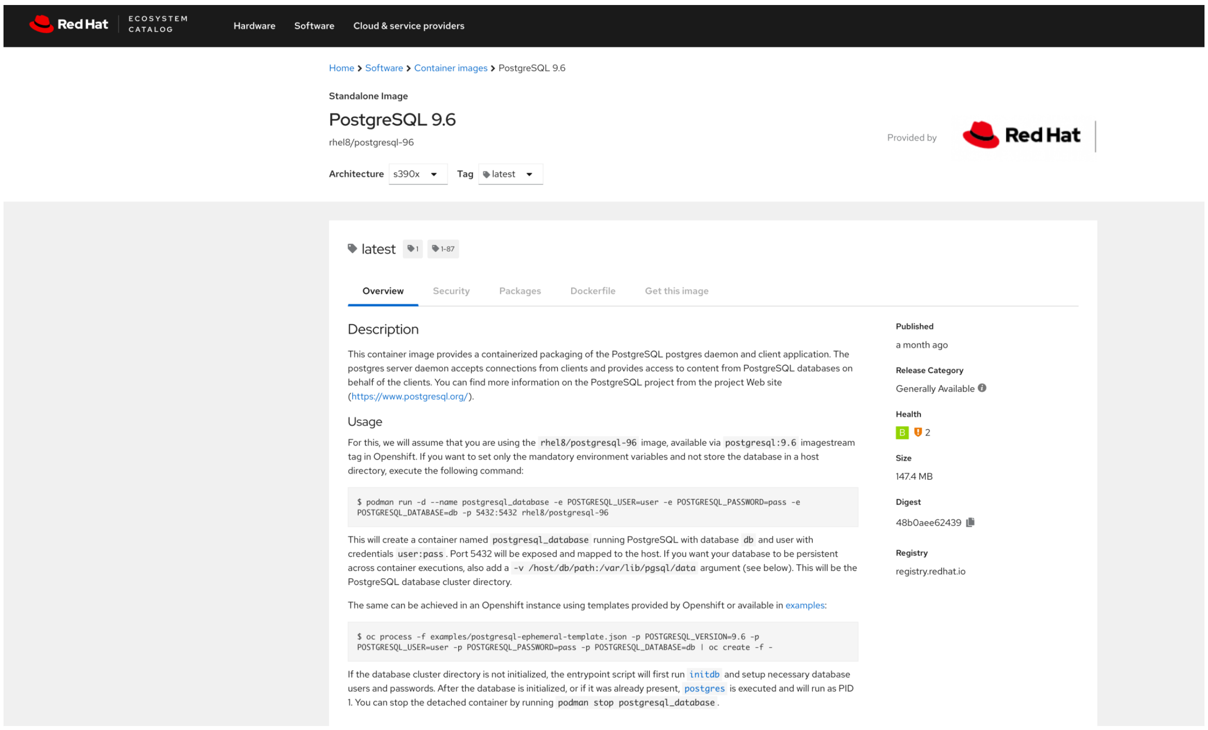
Owner
No owner

Figure 8.7-10: Verify the secret

Once the above steps are completed the container image from the Red Hat Container Catalog can be downloaded as an image stream by using the oc import-image command as shown below:

```
$ oc import-image rhel8/postgresql-96 --from=registry.redhat.io/rhel8/postgresql-96
--confirm

imagestream.image.openshift.io/postgresql-96 imported

Name:             postgresql-96
Namespace:        fmiranda-project
Created:          Less than a second ago
Labels:           <none>
Annotations:          openshift.io/image.dockerRepositoryCheck=2020-05-30T19:27:52Z
Image Repository:     image-registry.openshift-image-registry.svc:5000/fmiranda-
project/postgresql-96
Image Lookup:         local=false
Unique Images:        1
Tags:             1

latest
  tagged from registry.redhat.io/rhel8/postgresql-96

<output removed>
```

To test the catalog a popular container image can be selected from the available container images, in this example the image of PostgreSQL9.6 is chosen:


Red Hat ECOSYSTEM CATALOG
Hardware
Software
Cloud & service providers
Home > Software > Container images > PostgreSQL 9.6
Overview
Security
Packages
Dockerfile
Get this image

Standalone Image

PostgreSQL 9.6

rhel8/postgresql-96

Provided by Red Hat

Architecture s390x Tag latest

latest 1 1-87

Description

This container image provides a containerized packaging of the PostgreSQL postgres daemon and client application. The postgres server daemon accepts connections from clients and provides access to content from PostgreSQL databases on behalf of the clients. You can find more information on the PostgreSQL project from the project Web site (https://www.postgresql.org/).

Usage

For this, we will assume that you are using the `rhel8/postgresql-96` image, available via `postgresql:9.6` imagestream tag in Openshift. If you want to set only the mandatory environment variables and not store the database in a host directory, execute the following command:

```
$ podman run -d --name postgresql_database -e POSTGRESQL_USER=user -e POSTGRESQL_PASSWORD=pass -e
POSTGRESQL_DATABASE=db -p 5432:5432 rhel8/postgresql-96
```

This will create a container named `postgresql_database` running PostgreSQL with database `db` and user with credentials `user:pass`. Port 5432 will be exposed and mapped to the host. If you want your database to be persistent across container executions, also add a `-v /host/db/path:/var/lib/pgsql/data` argument (see below). This will be the PostgreSQL database cluster directory.

The same can be achieved in an Openshift instance using templates provided by Openshift or available in examples:

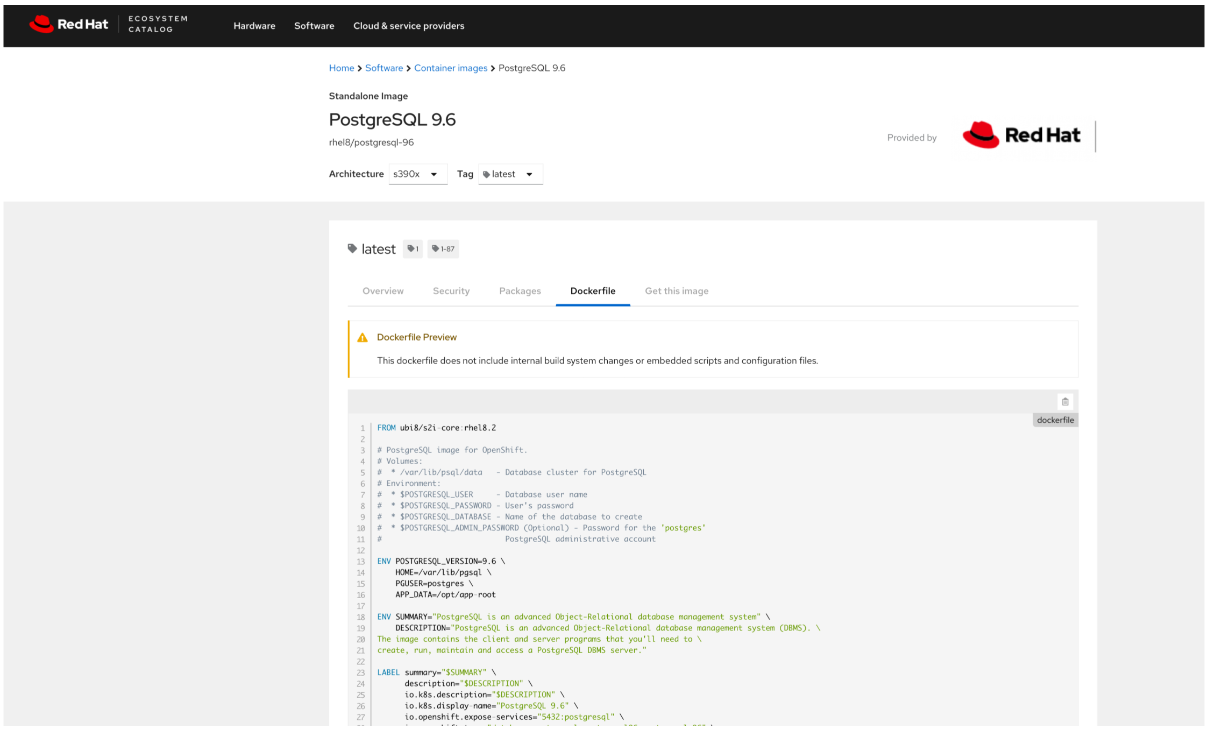
```
$ oc process -f examples/postgresql-ephemeral-template.json -p POSTGRESQL_VERSION=9.6 -p
POSTGRESQL_USER=user -p POSTGRESQL_PASSWORD=pass -p POSTGRESQL_DATABASE=db | oc create -f -
```

If the database cluster directory is not initialized, the entrypoint script will first run `initdb` and setup necessary database users and passwords. After the database is initialized, or if it was already present, `postgres` is executed and will run as PID 1. You can stop the detached container by running `podman stop postgresql_database`.

Published: a month ago
Release Category: Generally Available
Health: B 2
Size: 147.4 MB
Digest: 48b0aee62439
Registry: registry.redhat.io

Figure 8.7-11: Select a popular container image

The PostgreSQL container definition requires a few variables to be set during deployment:


Red Hat ECOSYSTEM CATALOG
Hardware
Software
Cloud & service providers
Home > Software > Container images > PostgreSQL 9.6
Overview
Security
Packages
Dockerfile
Get this image
dockerfile

Standalone Image

PostgreSQL 9.6

rhel8/postgresql-96

Provided by Red Hat

Architecture s390x Tag latest

latest 1 1-87

Dockerfile Preview

This dockerfile does not include internal build system changes or embedded scripts and configuration files.

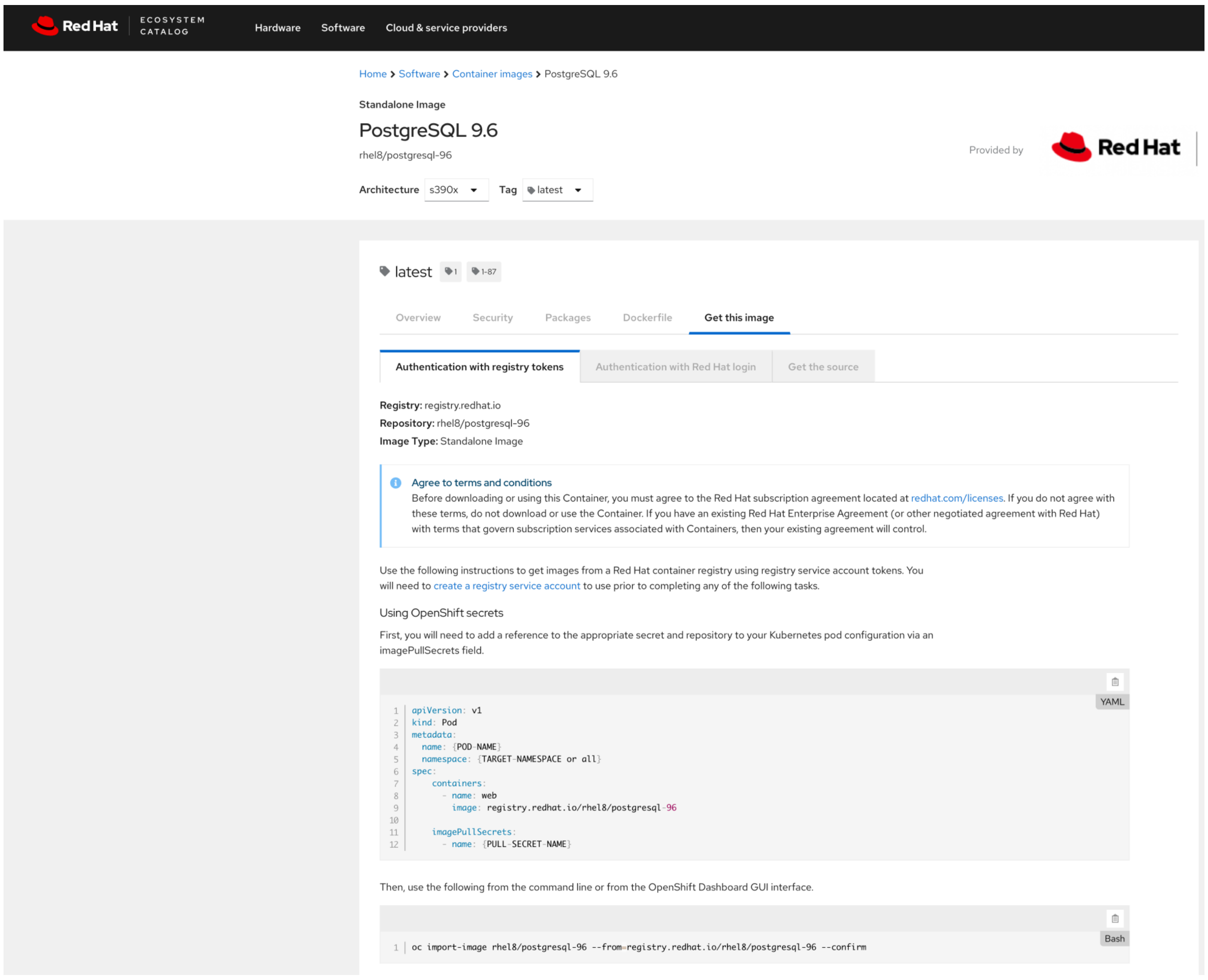
```
FROM ubi8/s2i-core:rhel8.2

# PostgreSQL image for OpenShift.
# Volumes:
#  * /var/lib/psql/data   - Database cluster for PostgreSQL
# Environment:
#  * $POSTGRESQL_USER     - Database user name
#  * $POSTGRESQL_PASSWORD - User's password
#  * $POSTGRESQL_DATABASE - Name of the database to create
#  * $POSTGRESQL_ADMIN_PASSWORD (Optional) - Password for the 'postgres'
#                           PostgreSQL administrative account

ENV POSTGRESQL_VERSION=9.6 \
    HOME=/var/lib/pgsql \
    PGUSER=postgres \
    APP_DATA=/opt/app-root

ENV SUMMARY="PostgreSQL is an advanced Object-Relational database management system" \
    DESCRIPTION="PostgreSQL is an advanced Object-Relational database management system (DBMS). \
The image contains the client and server programs that you'll need to \
create, run, maintain and access a PostgreSQL DBMS server."

LABEL summary="$SUMMARY" \
      description="$DESCRIPTION" \
      io.k8s.description="$DESCRIPTION" \
      io.k8s.display-name="PostgreSQL 9.6" \
      io.openshift.expose-services="5432:postgresql" \
```

Figure 8.7-12: PostgreSQL image variables

To download this container image, the following command line has to be used from a client machine that has access to the RHOCP cluster and has been previously logged in:

```
$ oc import-image rhel8/postgresql-96 --from=registry.redhat.io/rhel8/postgresql-96
--confirm
```

This information can be found by selecting the tab "Get this Image":


Red Hat ECOSYSTEM CATALOG Hardware Software Cloud & service providers

Home > Software > Container images > PostgreSQL 9.6

Standalone Image

PostgreSQL 9.6

rhel8/postgresql-96

Provided by Red Hat

Architecture s390x Tag latest

latest 1 1-87

Overview Security Packages Dockerfile Get this image

Authentication with registry tokens | Authentication with Red Hat login | Get the source

Registry: registry.redhat.io
Repository: rhel8/postgresql-96
Image Type: Standalone Image

Agree to terms and conditions
Before downloading or using this Container, you must agree to the Red Hat subscription agreement located at redhat.com/licenses. If you do not agree with these terms, do not download or use the Container. If you have an existing Red Hat Enterprise Agreement (or other negotiated agreement with Red Hat) with terms that govern subscription services associated with Containers, then your existing agreement will control.

Use the following instructions to get images from a Red Hat container registry using registry service account tokens. You will need to create a registry service account to use prior to completing any of the following tasks.

Using OpenShift secrets

First, you will need to add a reference to the appropriate secret and repository to your Kubernetes pod configuration via an imagePullSecrets field.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: {POD-NAME}
  namespace: {TARGET-NAMESPACE or all}
spec:
    containers:
      - name: web
        image: registry.redhat.io/rhel8/postgresql-96

    imagePullSecrets:
      - name: {PULL-SECRET-NAME}
```

Then, use the following from the command line or from the OpenShift Dashboard GUI interface.

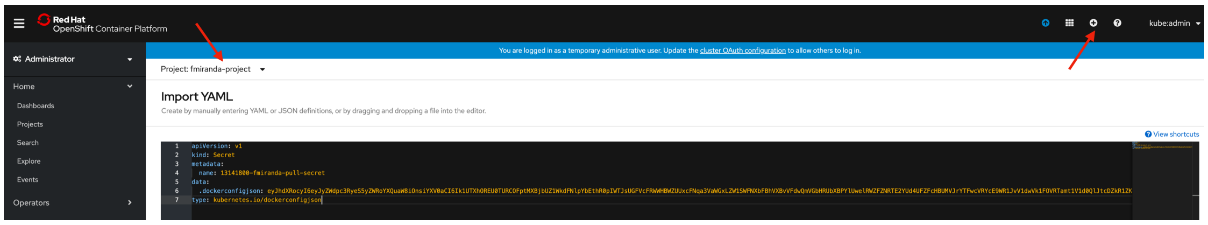
```bash
oc import-image rhel8/postgresql-96 --from=registry.redhat.io/rhel8/postgresql-96 --confirm
```

Figure 8.7-13: PostgreSQL image command

Before downloading the PostgreSQL container image, a project needs to be created, which will use the Service Account Secret created on Step 1:

```
$ oc new-project fmiranda-project
```

Once the project has been created, login to the RHOCP web console and the + (plus) sign at the top right corner has to be selected. It is important to verify that the project created has been selected and the Account Secret created previously from Step 1 can be pasted as shown in the figure below:



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: fmiranda-project

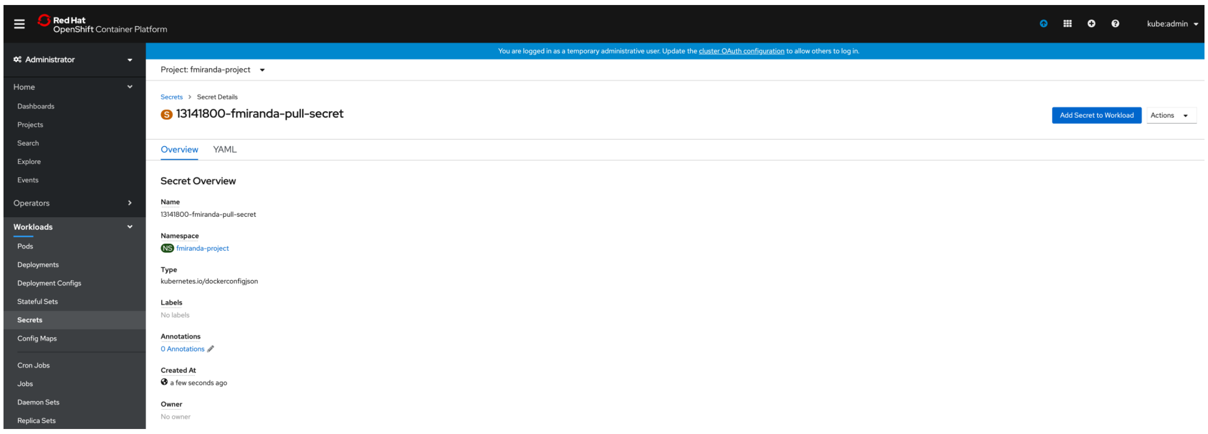
Import YAML

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: 13141800-fmiranda-pull-secret
data:
  .dockerconfigjson: eyJhdXRocyI6eyJyZWdpc3RyeS5yZWRoYXQuaW8iOnsiYXV0aCI6Ik1UTXhOREU0TURCOFptMXBjbUZ1WkdFNlpYbEthR0pIWTJsUGFVcFRWWHBWZUUxcFNqa3VaWGxLZW1SWFNXbFBhVXBvVFdwQmVGbHRUbXBPYlUwelRWZFZNRTE2YUd4UFZFcHBUMVJrYTFwcVRYcE9WR1JvV1dwVk1FOVRamt1V1d0QlJtcDZkR1ZK
type: kubernetes.io/dockerconfigjson
```

Figure 8.7-14: Import secret YAML in the project

The installed Account Secret can be verified selecting *Workloads* on the left menu, then *Secrets*, and then the correct project:



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: fmiranda-project

13141800-fmiranda-pull-secret

Secret Overview

Name
13141800-fmiranda-pull-secret

Namespace
NS fmiranda-project

Type
kubernetes.io/dockerconfigjson

Labels
No labels

Annotations
0 Annotations

Created At
a few seconds ago

Owner
No owner

Figure 8.7-15: Verify the secret

Once the above steps are completed the container image from the Red Hat Container Catalog can be downloaded as an image stream by using the oc import-image command as shown below:

```
$ oc import-image rhel8/postgresql-96 --from=registry.redhat.io/rhel8/postgresql-96
--confirm

imagestream.image.openshift.io/postgresql-96 imported

Name:           postgresql-96
Namespace:      fmiranda-project
Created:        Less than a second ago
Labels:         <none>
Annotations:        openshift.io/image.dockerRepositoryCheck=2020-05-30T19:27:52Z
Image Repository:   image-registry.openshift-image-registry.svc:5000/fmiranda-
project/postgresql-96
Image Lookup:       local=false
Unique Images:      1
Tags:           1

latest
  tagged from registry.redhat.io/rhel8/postgresql-96

<output removed>
```

8.7.3 Step 3: Deploying a pod using the container image imported from the Red Hat Container Catalog

The first thing to do is to switch to the Developer profile:

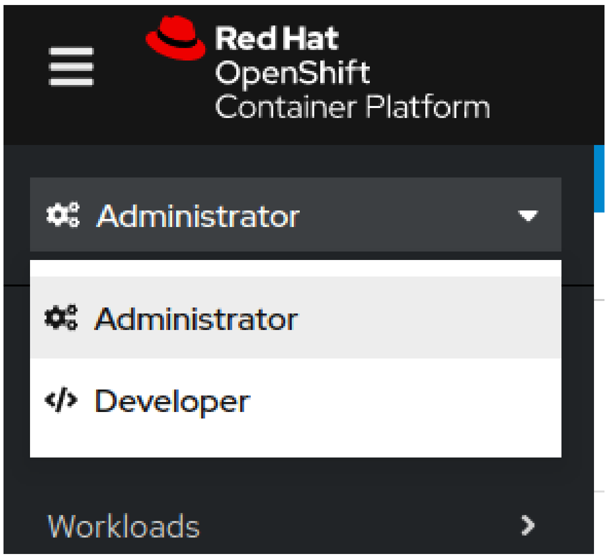
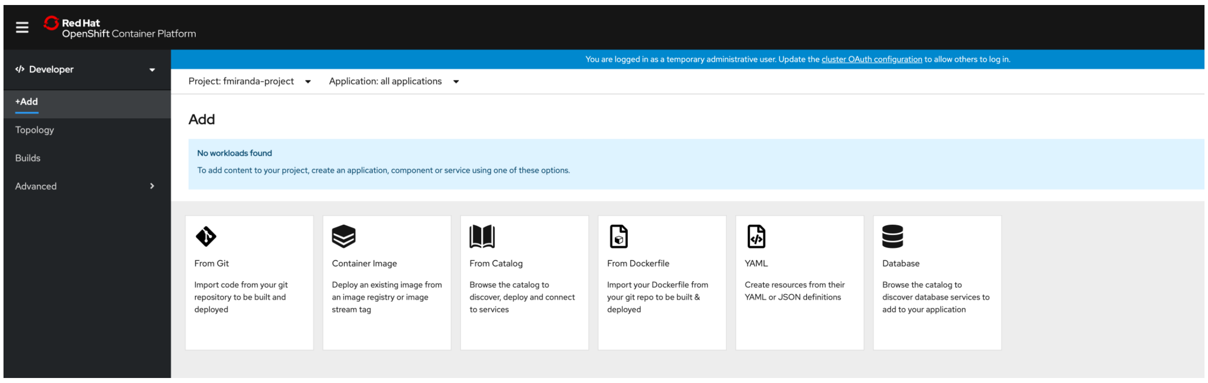


Figure 8.7-16: Switch to the developer mode

The +Add option has to be selected and also the Container Image:


Red Hat
OpenShift Container Platform
Developer
+Add
Topology
Builds
Advanced
Project: fmiranda-project
Application: all applications
Add

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

No workloads found

To add content to your project, create an application, component or service using one of these options.

From Git

Import code from your git repository to be built and deployed

Container Image

Deploy an existing image from an image registry or image stream tag

From Catalog

Browse the catalog to discover, deploy and connect to services

From Dockerfile

Import your Dockerfile from your git repo to be built & deployed

YAML

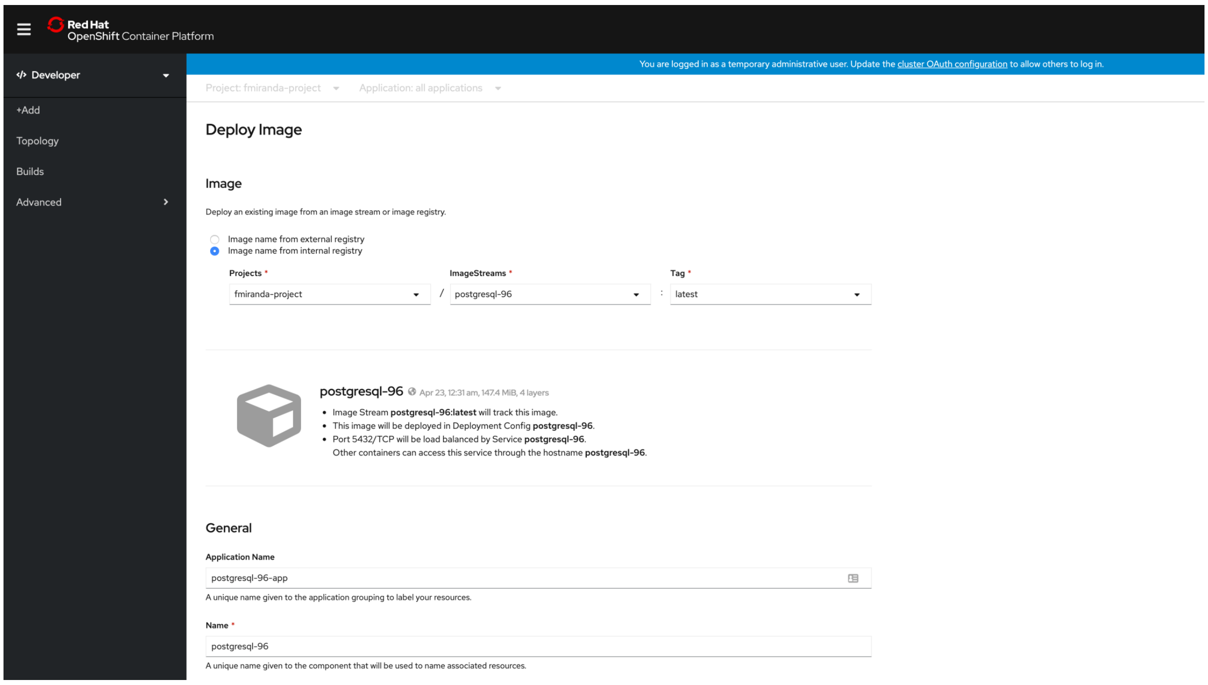
Create resources from their YAML or JSON definitions

Database

Browse the catalog to discover database services to add to your application

Figure 8.7-17: Add an application

The Image name has to be selected from the Internal Registry (Container image option), the correct project from Projects has to be also selected, and lastly the container image (postgresql-96) from ImageStreams and the version from Tag:



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: fmiranda-project   Application: all applications

Deploy Image

Image

Deploy an existing image from an image stream or image registry.

Image name from external registry
[✓ Image name from internal registry]

Projects *		ImageStreams *		Tag *
fmiranda-project	/	postgresql-96	:	latest

postgresql-96 Apr 23, 12:31 am, 147.4 MiB, 4 layers

- Image Stream **postgresql-96:latest** will track this image.
- This image will be deployed in Deployment Config **postgresql-96**.
- Port 5432/TCP will be load balanced by Service **postgresql-96**.
  Other containers can access this service through the hostname **postgresql-96**.

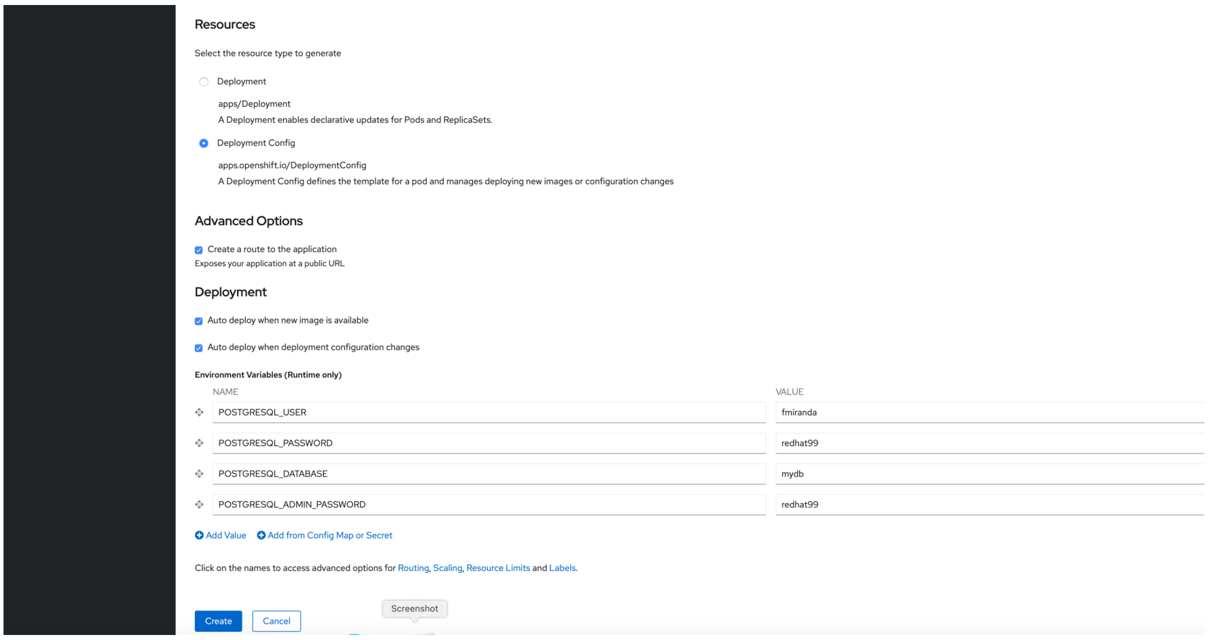
General

Application Name: postgresql-96-app
A unique name given to the application grouping to label your resources.

Name *: postgresql-96
A unique name given to the component that will be used to name associated resources.

Figure 8.7-18: Deploy an application

Now on Resources, the Deployment Config can be selected (for more flexibility, such as adding a persistent storage for example), then Deployment (opens a few options, including the Environment Variables). PostgreSQL requires a few environment variables to be defined, these variables are defined on the file used by the container image:



Resources

Select the resource type to generate

Deployment
apps/Deployment
A Deployment enables declarative updates for Pods and ReplicaSets.

[✓ Deployment Config]
apps.openshift.io/DeploymentConfig
A Deployment Config defines the template for a pod and manages deploying new images or configuration changes

Advanced Options

[✓] Create a route to the application
Exposes your application at a public URL

Deployment

[✓] Auto deploy when new image is available

[✓] Auto deploy when deployment configuration changes

Environment Variables (Runtime only)

NAME	VALUE
POSTGRESQL_USER	fmiranda
POSTGRESQL_PASSWORD	redhat99
POSTGRESQL_DATABASE	mydb
POSTGRESQL_ADMIN_PASSWORD	redhat99

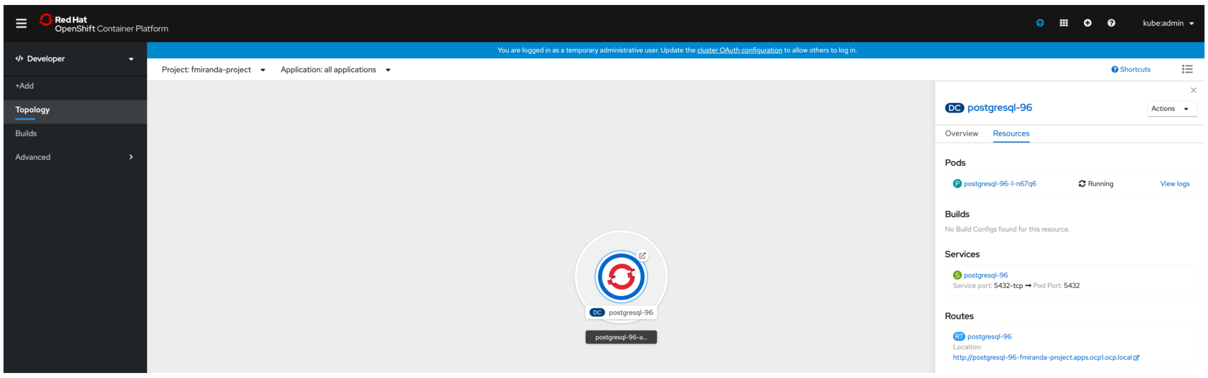
Add Value   Add from Config Map or Secret

Click on the names to access advanced options for Routing, Scaling, Resource Limits and Labels.

Create   Cancel

Figure 8.7-19: Deployment config of the image

Once all the environment variables are defined, the Create button can be selected, this will deploy a pod using the container image imported from the Red Hat Container Catalog:



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Pods

postgresql-96-1-n67q6 Running View logs

Builds

No Build Configs found for this resource.

Services

postgresql-96
Service port: 5432-tcp → Pod Port: 5432

Routes

postgresql-96
Location:
http://postgresql-96-fmiranda-project.apps.ocp1.ocp.local

Figure 8.7-20: Deployment the container

Now the application is being deployed with the image downloaded from the Red Hat Container Catalog using a Secret associated to the Registry Service account.

Chapter 9. References

Red Hat OpenShift Container Platform 4 Documentation

- Red Hat OpenShift Container Platform Documentation
- Red Hat Software Portal
- Upstream installer code and documentation
- IBM Knowledge Center
- Red Hat OpenShift Data Foundation Documentation

Installation related references

- Installing the infrastructure for a User Provisioned Infrastructure (UPI) deployment:
    - Installing IBM z/VM
    - Installing RHEL KVM
- Blog by Pedro Ibáñez Requena: Installing Red Hat OpenShift in a Mainframe z-series
- Blog by Filipe Miranda: Red Hat OpenShift Installation Process Experiences on IBM Z/LinuxONE
- Red Hat Ansible Playbooks for installing Red Hat OpenShift on Z
- Red Hat OpenShift Container Platform Installation

Getting Started with Red Hat OpenShift

- Get started with Red Hat OpenShift
- Interactive Learning Portal
- Red Hat OpenShift Blog

Developing Applications

- Red Hat Software Collections
- Steve Martinelli, Sai Vennam: A quick lab to explore the Red Hat OpenShift development environment
- Red Hat OpenShift Pipelines: Cloud-native CI/CD on Red Hat OpenShift

LinuxONE and System Z

- Blog by Eberhard Pasch
- Linux on Z blog

Chapter 10. Contributors

The following individuals contributed to the production of this reference architecture. Feel free to send us any feedback about the document.

Pedro Ibáñez Requena	**Project and content lead**	pedro@redhat.com
Thomas Stober	**Project and content lead**	tstober@de.ibm.com
Wilhelm Mild	**Project and content lead**	wilhelm.mild@de.ibm.com
Ken Bell	**Project and content lead**	kbell@redhat.com

The following individuals contributed with content, reviews or feedback for this reference architecture.

Filipe Miranda	**Content contributor**	fmiranda@ibm.com
Hendrik Brueckner	**Content reviewer**	brueckner@de.ibm.com
Murthy Garimella	**Content reviewer**	mgarimel@redhat.com
Andy McCrae	**Content reviewer**	amccrae@redhat.com
Silke Niemann	**Content reviewer**	sniem@de.ibm.com