Red Hat OpenStack Services on OpenShift 18.0

Planning your deployment

Planning a Red Hat OpenStack Services on OpenShift environment on a Red Hat OpenShift Container Platform cluster

Last Updated: 2024-09-12

Red Hat OpenStack Services on OpenShift 18.0 Planning your deployment

Planning a Red Hat OpenStack Services on OpenShift environment on a Red Hat OpenShift Container Platform cluster

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

Plan your Red Hat OpenStack Services on OpenShift control plane and data plane.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION 4

CHAPTER 1. RED HAT OPENSTACK SERVICES ON OPENSHIFT OVERVIEW 5
1.1. RHOSO SERVICES AND OPERATORS 5
1.2. FEATURES OF A RHOSO ENVIRONMENT 8
1.3. SUPPORTED TOPOLOGIES FOR A RHOSO ENVIRONMENT 9
1.3.1. Compact topology 9
1.3.2. Dedicated nodes topology 10

CHAPTER 2. PLANNING YOUR DEPLOYMENT 12
2.1. HOW TO DEPLOY THE CLOUD INFRASTRUCTURE 12
2.2. CUSTOM RESOURCE DEFINITIONS (CRDS) 13
2.2.1. CRD naming conventions 14

CHAPTER 3. SYSTEM REQUIREMENTS 15
3.1. RED HAT OPENSHIFT CONTAINER PLATFORM CLUSTER REQUIREMENTS 15
3.2. DATA PLANE NODE REQUIREMENTS 17
3.3. COMPUTE NODE REQUIREMENTS 17

CHAPTER 4. PLANNING YOUR NETWORKS 19
4.1. DEFAULT PHYSICAL NETWORKS 19
4.2. RHOSO NETWORK ISOLATION 19
4.3. NICS 19
4.3.1. Scaling considerations 20
4.3.2. Troubleshooting Compute hostname change detection 20
4.4. PLANNING FOR IRONIC AND BAREMETAL PROVISIONING NETWORKS 20
4.5. STORAGE NETWORK PLANNING CONSIDERATIONS 21
4.6. NETWORK FUNCTIONS VIRTUALIZATION (NFV) 21
4.7. ADDITIONAL RESOURCES FOR RHOSO NETWORK PLANNING 21

CHAPTER 5. FEDERAL INFORMATION PROCESSING STANDARD ON RED HAT OPENSTACK SERVICES ON OPENSHIFT 22
5.1. PREPARING TO INSTALL A FIPS ENABLED RED HAT OPENSTACK SERVICES ON OPENSHIFT CONTROL PLANE 22
5.2. VERIFICATION OF FIPS STATUS 22

CHAPTER 6. PLANNING STORAGE AND SHARED FILE SYSTEMS 24
6.1. SUPPORTED STORAGE FEATURES AND TOPOLOGIES 24
6.2. STORAGE TECHNOLOGIES 26
6.2.1. Red Hat Ceph Storage 26
6.2.2. Block storage (cinder) 27
6.2.3. Images (glance) 27
6.2.4. Object Storage (swift) 27
6.2.5. Shared File Systems (manila) 27
6.3. STORAGE NETWORKS 28
6.3.1. Planning networking for the Block Storage service 28
6.3.2. Planning networking for the Shared File Systems service 29
6.3.2.1. Setting DHSS to true 29
6.3.2.2. Setting DHSS to false 29
6.3.2.3. Ensuring network connectivity to the share 30
6.4. SCALABILITY AND BACK-END STORAGE 30
6.5. STORAGE ACCESSIBILITY AND ADMINISTRATION 30
6.6. STORAGE SECURITY 31

6.7. STORAGE REDUNDANCY AND DISASTER RECOVERY 31
6.8. MANAGING THE STORAGE SOLUTION 32
6.9. SIZING RED HAT OPENSHIFT STORAGE 32
6.9.1. Image service considerations 32
6.9.2. Object Storage service considerations 33
CHAPTER 7. INTEGRATION 35
CHAPTER 8. SUBSCRIPTIONS 36

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

Providing documentation feedback in Jira

Use the Create Issue form to provide feedback on the documentation for Red Hat OpenStack Services on OpenShift (RHOSO) or earlier releases of Red Hat OpenStack Platform (RHOSP). When you create an issue for RHOSO or RHOSP documents, the issue is recorded in the RHOSO Jira project, where you can track the progress of your feedback.

To complete the Create Issue form, ensure that you are logged in to Jira. If you do not have a Red Hat Jira account, you can create an account at https://issues.redhat.com.

1. Click the following link to open a **Create Issue** page: Create Issue

2. Complete the **Summary** and **Description** fields. In the **Description** field, include the documentation URL, chapter or section number, and a detailed description of the issue. Do not modify any other fields in the form.

3. Click **Create**.

CHAPTER 1. RED HAT OPENSTACK SERVICES ON OPENSHIFT OVERVIEW

Red Hat OpenStack Services on OpenShift (RHOSO) provides the foundation to build a private or public Infrastructure-as-a-Service (IaaS) cloud on top of Red Hat Enterprise Linux. It is a scalable, fault-tolerant platform for the development of cloud-enabled workloads.

The RHOSO control plane is hosted and managed as a workload on a Red Hat OpenShift Container Platform (RHOCP) cluster. The RHOSO data plane consists of external Red Hat Enterprise Linux (RHEL) nodes, managed with Red Hat Ansible Automation Platform, that host RHOSO workloads. The data plane nodes can be Compute nodes, Storage nodes, Networker nodes, or other types of nodes.

The RHOSO IaaS cloud is implemented by a collection of interacting services that control its computing, storage, and networking resources. You can manage the cloud with a web-based interface to control, provision, and automate RHOSO resources. Additionally, an extensive API controls the RHOSO infrastructure and this API is also available to end users of the cloud.

NOTE

RHOSO only supports RHOCP master and worker nodes with processors based on a 64-bit x86 hardware architecture.

1.1. RHOSO SERVICES AND OPERATORS

The Red Hat OpenStack Services on OpenShift (RHOSO) IaaS services are implemented as a collection of Operators running on a Red Hat OpenShift Container Platform (RHOCP) cluster. These Operators manage the compute, storage, networking, and other services for your RHOSO cloud.

IMPORTANT

Red Hat recommends using the Red Hat OpenShift Container Platform (RHOCP) OperatorHub to obtain all Operators.

The OpenStack Operator (**openstack-operator**) installs all the service Operators detailed in the Services table, and is the interface that you use to manage those Operators. The OpenStack Operator also installs and manages the following Operators:

openstack-baremetal-operator

Used by the OpenStack Operator to optionally provision bare-metal nodes.

openstack-ansibleee-operator

Used by the OpenStack Operator to execute Ansible to deploy, configure, and orchestrate software on the data plane nodes. Ansible uses SSH to communicate with the data plane nodes.

For more information on the functionality of each service, see the service-specific documentation on the Red Hat OpenStack Services on OpenShift 18.0 documentation portal.

Table 1.1. Services

Service	Operator	Default	Description

Service	Operator	Default	Description
Bare Metal Provisioning (ironic)	**ironic-operator**	Disabled	Supports physical machines for a variety of hardware vendors with hardware-specific drivers. Bare Metal Provisioning integrates with the Compute service to provision physical machines in the same way that virtual machines are provisioned, and provides a solution for the bare-metal-to-trusted-project use case.
Block Storage (cinder)	**cinder-operator**	Enabled	Provides and manages persistent block storage volumes for virtual machine instances.
Compute (nova)	**nova-operator**	Enabled	Provides management of the provisioning of compute resources, such as Virtual Machines, through the **libvirt** driver or physical servers through the **ironic** driver.
Dashboard (horizon)	**horizon-operator**	Disabled	Provides a browser-based GUI dashboard for creating and managing cloud resources and user access. The Dashboard service provides Project, Admin, and Settings dashboards by default. You can configure the dashboard to interface with other products such as billing, monitoring, and additional management tools.
DNS (designate)	**designate-operator**	Enabled	Provides DNS-as-a-Service (DNSaaS) that manages DNS records and zones in the cloud. You can deploy BIND instances to contain DNS records, or you can integrate the DNS service into an existing BIND infrastructure. Can also be integrated with the RHOSO Networking service (neutron) to automatically create records for virtual machine instances, network ports, and floating IPs.
Identity (keystone)	**keystone-operator**	Enabled	Provides user authentication and authorization to all RHOSO services and for managing users, projects, and roles. Supports multiple authentication mechanisms, including username and password credentials, token-based systems, and AWS-style log-ins.
Image (glance)	**glance-operator**	Enabled	Registry service for storing resources such as virtual machine images and volume snapshots. Cloud users can add new images or take a snapshot of an existing instance for immediate storage. You can use the snapshots for backup or as templates for new instances.

Service	Operator	Default	Description
Key Management (barbican)	**barbican-operator**	Enabled	Provides secure storage, provisioning and management of secrets such as passwords, encryption keys, and X.509 Certificates. This includes keying material such as Symmetric Keys, Asymmetric Keys, Certificates, and raw binary data.
Load-balancing (octavia)	**octavia-operator**	Disabled	Provides Load Balancing-as-a-Service (LBaaS) for the cloud that supports multiple provider drivers. The reference provider driver (Amphora provider driver) is an open-source, scalable, and highly available load balancing provider. It accomplishes its delivery of load balancing services by managing a fleet of virtual machines, collectively known as amphorae, which it creates on demand.
MariaDB	**mariadb-operator**	Enabled	Provides methods to deploy and manage MariaDB Galera clusters.
Memcached	**infra-operator**	Enabled	Provides methods for managing infrastructure.
Networking (neutron)	**neutron-operator**	Enabled	Provides Networking-as-a-Service (NaaS) through software-defined networking (SDN) in virtual compute environments. Handles the creation and management of a virtual networking infrastructure in the cloud, which includes networks, subnets, and routers.
Object Storage (swift)	**swift-operator**	Enabled	Provides efficient and durable storage of large amounts of data, including static entities such as videos, images, email messages, files, or instance images. Objects are stored as binaries on the underlying file system with metadata stored in the extended attributes of each file.
OVN	**ovn-operator**	Enabled	Provides methods to deploy and manage OVNs.
Orchestration (heat)	**heat-operator**	Disabled	Template-based orchestration engine that supports automatic creation of resource stacks. Provides templates to create and manage cloud resources such as storage, networking, instances, or applications. You can use the templates to create stacks, which are collections of resources.

Service	Operator	Default	Description
Placement (placement)	**placement-operator**	Enabled	Provides methods to install and manage an OpenStack Placement installation.
Telemetry (ceilometer, prometheus)	**telemetry-operator**	Enabled	Provides user-level usage data for RHOSO clouds. You can use the data for customer billing, system monitoring, or alerts. Telemetry can collect data from notifications sent by existing RHOSO components such as Compute usage events, or by polling RHOSO infrastructure resources such as libvirt.
RabbitMQ	**rabbitmq-cluster-operator**	Enabled	Provides methods to deploy and manage RabbitMQ clusters.
Shared File Systems (manila)	**manila-operator**	Disabled	Provisions shared file systems that can be used by multiple virtual machine instances, bare-metal nodes, or containers.

1.2. FEATURES OF A RHOSO ENVIRONMENT

The basic architecture of a Red Hat OpenStack Services on OpenShift (RHOSO) environment includes the following features:

Container-native application delivery

RHOSO is delivered by using a container-native approach that spans the Red Hat OpenShift Container Platform (RHOCP) and RHEL platforms to deliver a container-native RHOSO deployment.

RHOCP-hosted services

RHOCP hosts infrastructure services and RHOSO controller services by using RHOCP Operators to provide lifecycle management.

Ansible-managed RHEL-hosted services

RHOSO workloads run on RHEL nodes that are managed by the OpenStack Operator. The OpenStack Operator runs Ansible jobs to configure the RHEL data plane nodes, such as the Compute nodes. RHOCP manages provisioning, DNS, and configuration management.

Installer-provisioned infrastructure

The RHOSO installer enables installer-provisioned infrastructure that uses RHOSO bare-metal machine management to provision the Compute nodes for the RHOSO cloud.

User-provisioned infrastructure

If you have your own machine ingest and provisioning workflow, you can use the RHOSO pre-provisioned model to add your pre-provisioned hardware into your RHOSO environment, while receiving the benefits of a container-native workflow.

Hosted RHOSO client

RHOSO provides a host **openstackclient** pod that is preconfigured with administrator access to the deployed RHOSO environment.

1.3. SUPPORTED TOPOLOGIES FOR A RHOSO ENVIRONMENT

Red Hat OpenStack Services on OpenShift (RHOSO) supports a compact control plane topology and a dedicated nodes control plane topology.

In a compact topology, the RHOSO control plane and the Red Hat OpenShift Container Platform (RHOCP) control plane share the same physical nodes.

In a dedicated nodes topology, the RHOCP control plane runs on one set of physical nodes and the RHOSO control plane runs on another set of physical nodes.

1.3.1. Compact topology

The compact RHOSO topology is the default, and consists of the following components:

OpenShift compact cluster

A Red Hat OpenShift cluster that hosts both the RHOSO and the RHOCP control planes. The RHOSO control plane consists of the OpenStack controller services pods that consist of services such as the Compute service (nova), the Networking service (neutron), and so on.

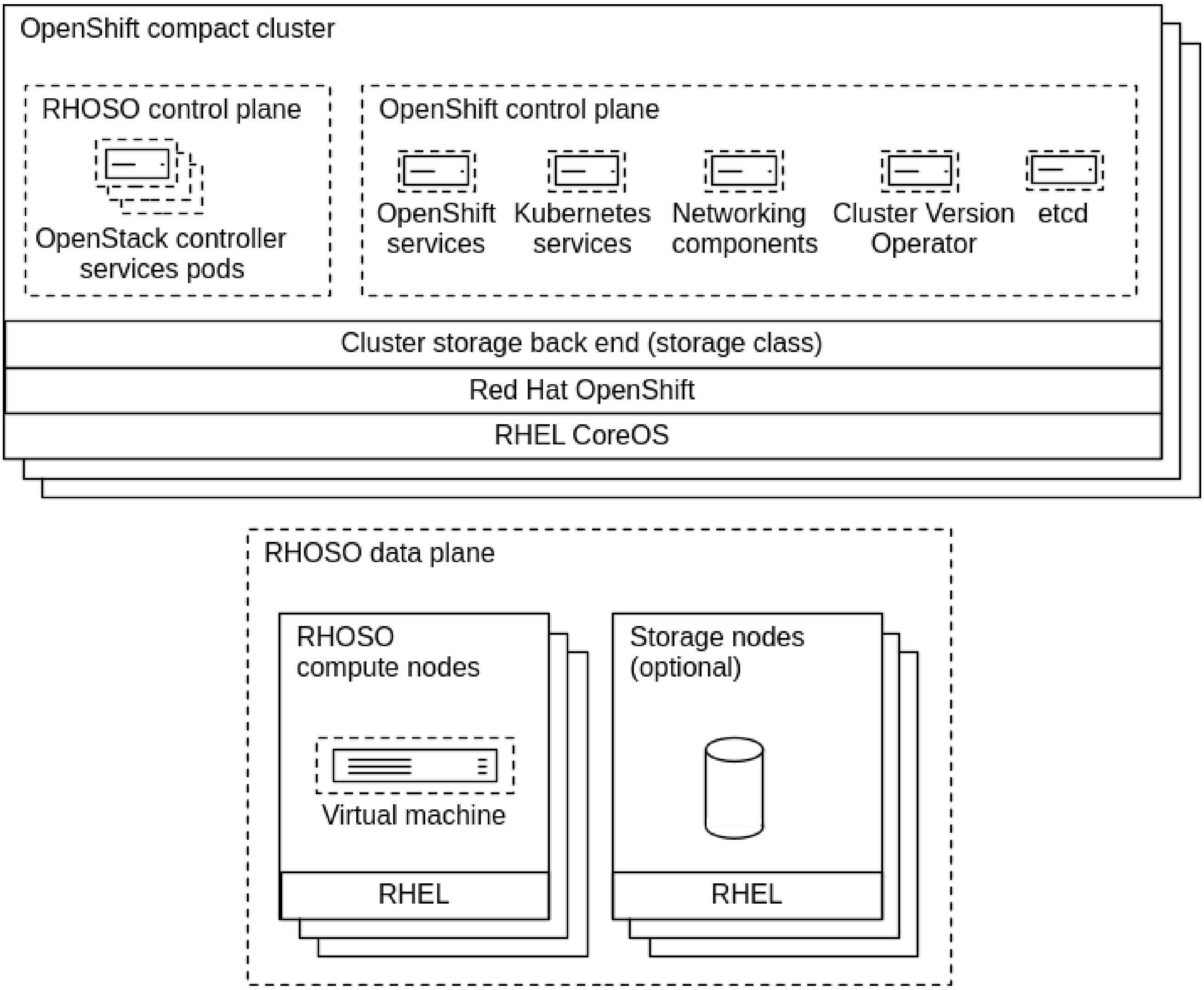
The OpenShift control plane hosts the pods that run the following services required for RHOCP: OpenShift services, Kubernetes services, networking components, Cluster Version Operator, and etcd.

For more information, see Introduction to OpenShift Container Platform in the RHOCP *Architecture* guide

RHOSO data plane

The RHOSO data plane consists of OpenStack Compute nodes. Nodes dedicated to storage are optional.

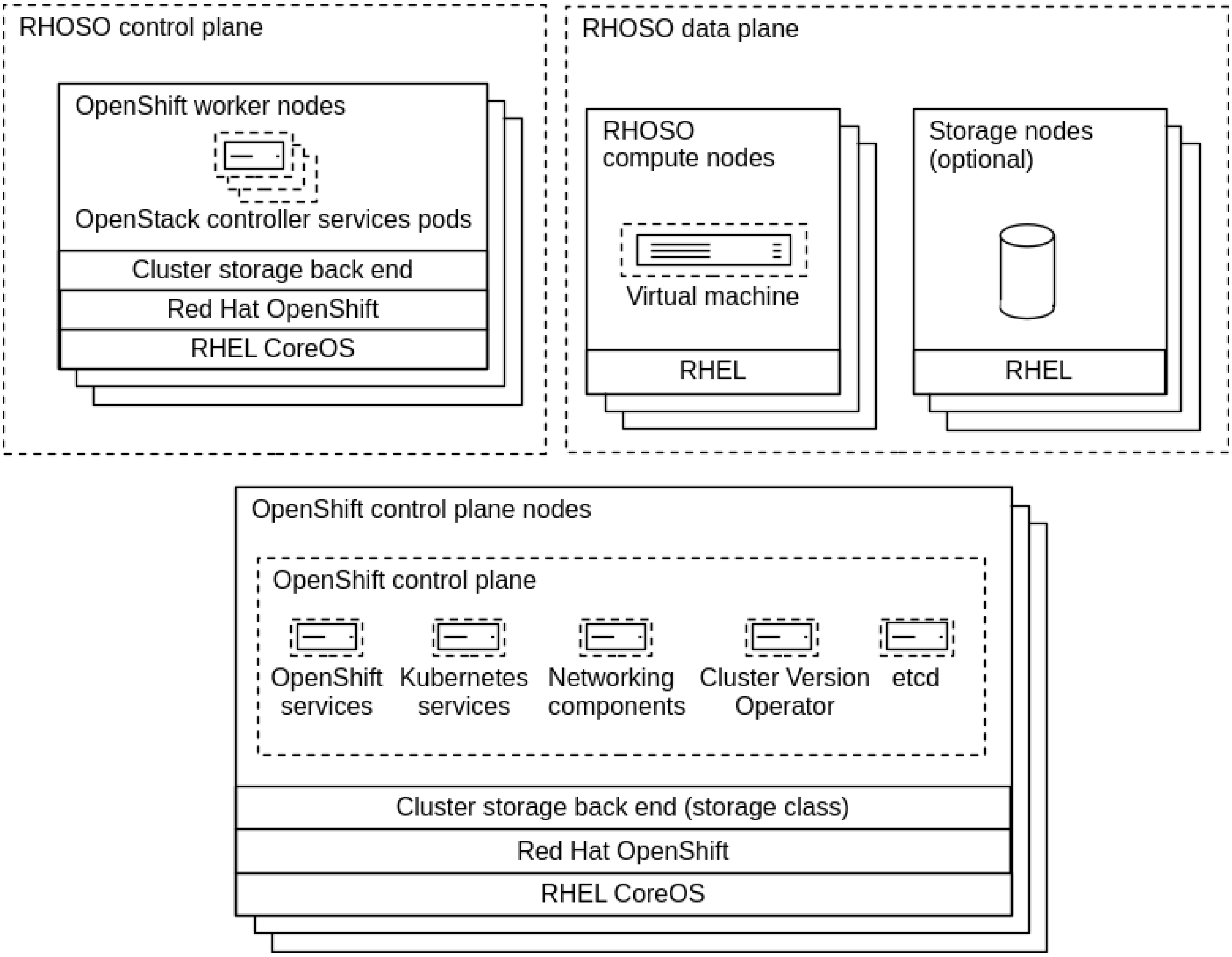
Figure 1.1. Compact RHOSO topology



1.3.2. Dedicated nodes topology

The dedicated nodes RHOSO topology differs from the compact topology in that there is a separate node cluster for the RHOSO control plane and a separate node cluster for the OpenShift control plane.

Figure 1.2. Dedicated nodes RHOSO topology



CHAPTER 2. PLANNING YOUR DEPLOYMENT

To deploy and operate your Red Hat OpenStack Services on OpenShift (RHOSO) environment, you use the tools and container infrastructure provided by the Red Hat OpenShift Container Platform (RHOCP).

RHOCP uses a modular system of Operators to extend the functions of your RHOCP cluster. The RHOSO OpenStack Operator (**openstack-operator**) installs and runs a RHOSO control plane within RHOCP and automates the deployment of a RHOSO data plane. The data plane is the collection of nodes that host RHOSO workloads. The OpenStack Operator prepares the nodes with the operating system configuration that is required to host the RHOSO services and workloads.

The OpenStack Operator manages a set of Custom Resource Definitions (CRDs) that define how you can deploy and manage the infrastructure and configuration of the RHOSO control plane and the data plane nodes. To create a RHOSO cloud with a RHOCP-hosted control plane, you use the OpenStack Operator CRDs to create a set of custom resources (CRs) that configure your control plane and your data plane.

2.1. HOW TO DEPLOY THE CLOUD INFRASTRUCTURE

To create a RHOSO cloud with a RHOCP hosted control plane, you must complete the following tasks:

1. Install OpenStack Operator (**openstack-operator**) on an operational RHOCP cluster.
2. Provide secure access to the RHOSO services.
3. Create and configure the control plane network.
4. Create and configure the data plane networks.
5. Create a control plane for your environment.
6. Customize the control plane for your environment.
7. Create and configure the data plane nodes.
8. Optional: Configure a storage solution for the RHOSO deployment.

You perform the control plane installation tasks and all data plane creation tasks on a workstation that has access to the RHOCP cluster.

Install OpenStack Operator (openstack-operator) on an operational RHOCP cluster

The RHOSO administrator installs the OpenStack Operator on the RHOCP cluster. For information about how to install the OpenStack Operator, see Installing and preparing the Operators in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

Provide secure access to the RHOSO services

You must create a Secret custom resource (CR) to provide secure access to the RHOSO service pods. For information, see Providing secure access to the Red Hat OpenStack Platform services in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

Create and configure the control plane network

You use RHOCP Operators to prepare the RHOCP cluster for the RHOSO control plane network. For information, see Preparing RHOCP for RHOSP networks in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

Create and configure the data plane networks

You use RHOCP Operators to prepare the RHOCP cluster for the RHOSO data plane network. For information, see Configuring the data plane network in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

Create a control plane for your environment

You configure and create an initial control plane with the recommended configurations for each service. For information, see Creating the control plane in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

Customize the control plane for your environment

You can customize your deployed control plane with the services required for your environment. For information, see Customizing the control plane in the *Customizing the Red Hat OpenStack Services on OpenShift deployment* guide.

Create and configure the data plane nodes

You configure and create a simple data plane with the minimum features. For information, see Creating the data plane in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

Customize the data plane for your environment

You can customize your deployed data plane with the features and configuration required for your environment. For information, see Customizing the data plane in the *Customizing the Red Hat OpenStack Services on OpenShift deployment* guide.

Configure a storage solution for the RHOSO deployment

You can optionally configure a storage solution for your RHOSO deployment. For information, see the Configuring persistent storage guide.

2.2. CUSTOM RESOURCE DEFINITIONS (CRDS)

The OpenStack Operator includes a set of custom resource definitions (CRDs) that you can use to create and manage RHOSP resources.

- Use the following command to view a complete list of the RHOSP CRDs:
  $ oc get crd | grep "^openstack"
- Use the following command to view the definition for a specific CRD:

```
$ oc describe crd openstackcontrolplane
Name:         openstackcontrolplane.openstack.org
Namespace:
Labels:       operators.coreos.com/operator.openstack=
Annotations:  cert-manager.io/inject-ca-from:
$(CERTIFICATE_NAMESPACE)/$(CERTIFICATE_NAME)
              controller-gen.kubebuilder.io/version: v0.3.0
API Version:  apiextensions.k8s.io/v1
Kind:         CustomResourceDefinition
...
```

- Use the following command to view descriptions of the fields you can use to configure a specific CRD:

```
$ oc explain openstackcontrolplane.spec
KIND:     OpenStackControlPlane
VERSION:  core.openstack.org/v1beta1

RESOURCE: spec <Object>
```

```
DESCRIPTION:
    <empty>

FIELDS:
  ceilometer   <Object>
  cinder       <Object>
  dns  <Object>
  extraMounts  <[]Object>
...
```

Additional resources

- Managing resources from custom resource definitions

2.2.1. CRD naming conventions

Each CRD contains multiple names in the **spec.names** section. Use these names depending on the context of your actions:

- Use **kind** when you create and interact with resource manifests:

  ```
  apiVersion: core.openstack.org/v1beta1
  kind: OpenStackControlPlane
  ...
  ```

  The **kind** name in the resource manifest correlates to the **kind** name in the respective CRD.

- Use **singular** when you interact with a single resource:

  ```
  $ oc describe openstackcontrolplane/compute
  ```

CHAPTER 3. SYSTEM REQUIREMENTS

You must plan your Red Hat OpenStack Services on OpenShift (RHOSO) deployment to determine the system requirements for your environment.

3.1. RED HAT OPENSHIFT CONTAINER PLATFORM CLUSTER REQUIREMENTS

The minimum requirements for the Red Hat OpenShift Container Platform (RHOCP) cluster that hosts your Red Hat OpenStack Services on OpenShift (RHOSO) control plane are as follows:

Hardware

- An operational, pre-provisioned 3-node RHOCP compact cluster, version 4.16.
- Each node in the compact cluster must have the following resources:
  - 64 GB RAM
  - 16 CPU cores
  - 120GB NVMe or SSD for the root disk plus 250 GB storage (NVMe or SSD is strongly recommended)

    NOTE

    The images, volumes and root disks for the virtual machine instances running on the deployed environment are hosted on dedicated external storage nodes. However, the service logs, databases, and metadata are stored in a RHOCP Persistent Volume Claim (PVC). A minimum of 150 GB is required for testing.

  - 2 physical NICs

    NOTE

    In a 6-node cluster with 3 controllers and 3 workers, only the worker nodes require 2 physical NICs.

- Persistent Volume Claim (PVC) storage on the cluster:
  - 150 GB persistent volume (PV) pool for service logs, databases, file import conversion, and metadata.

> NOTE
>
> - You must plan the size of the PV pool that you require for your RHOSO pods based on your RHOSO workload. For example, the Image service image conversion PVC should be large enough to host the largest image and that image after it is converted, as well as any other concurrent conversions. You must make similar considerations for the storage requirements if your RHOSO deployment uses the Object Storage service (swift).
> - The PV pool is required for the Image service, however the actual images are stored on the Image service back end, such as Red Hat Ceph Storage or SAN.

- 5 GB of the available PVs must be backed by local SSDs for control plane services such as the Galera, OVN, and RabbitMQ databases.

Software

- The RHOCP environment supports Multus CNI.
- The following Operators are installed on the RHOCP cluster:
  - The Kubernetes NMState Operator. This Operator must be started by creating an **nmstate** instance. For information, see Installing the Kubernetes NMState Operator in the RHOCP *Networking* guide.
  - The MetalLB Operator. This Operator must be started by creating a **metallb** instance. For information, see Installing the MetalLB Operator in the RHOCP *Networking* guide.

    > NOTE
    >
    > When you start MetalLB with the MetalLB Operator, the Operator starts an instance of a **speaker** pod on each node in the cluster. When using an extended architecture such as 3 OCP controller/master and 3 OCP computes/workers, if your OCP controllers do not have access to the **ctlplane** and **internalapi** networks, you must limit the **speaker** pods to the OCP compute/worker nodes. For more information about speaker pods, see Limit speaker pods to specific nodes .

  - The cert-manager Operator. For information, see cert-manager Operator for Red Hat OpenShift in the RHOCP *Security and compliance* guide.
  - The Cluster Observability Operator. For information, see Installing the Cluster Observability Operator.
  - The Cluster Baremetal Operator (CBO). The CBO deploys the Bare Metal Operator (BMO) component, which is required to provision bare-metal nodes as part of the data plane deployment process.
- The following tools are installed on the cluster workstation:
  - The **oc** command line tool.
  - The **podman** command line tool.

- The RHOCP storage back end is configured.
- The RHOCP storage class is defined, and has access to persistent volumes of type **ReadWriteOnce** and **ReadWriteMany**.
- For installer-provisioned infrastructure, you must prepare an operating system image for use with bare-metal provisioning. You can use the following image as the bare-metal image: https://catalog.redhat.com/software/containers/rhel9/rhel-guest-image/6197bdceb4dcabca7fe351d5?container-tabs=overview

Additional resources

- Storage and Post-installation storage configuration

3.2. DATA PLANE NODE REQUIREMENTS

You can use pre-provisioned nodes or unprovisioned bare-metal nodes to create the data plane. The minimum requirements for data plane nodes are as follows:

- Pre-provisioned nodes:
  - RHEL 9.4.
  - Configured for SSH access with the SSH keys generated during data plane creation. The SSH user must either be **root** or have unrestricted and password-less sudo enabled. For more information, see Creating the data plane secrets in the *Deploying Red Hat OpenStack Services on OpenShift* guide.
  - Routable IP address on the control plane network to enable Ansible access through SSH.

IMPORTANT

Some network architectures may require the following networking capabilities:

- A dedicated NIC on RHOCP worker nodes for RHOSP isolated networks.
- Port switches with VLANs for the required isolated networks.

Consult with your RHOCP and network administrators about whether these are requirements in your deployment.

For information on the required isolated networks, see Default Red Hat OpenStack Platform networks in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

3.3. COMPUTE NODE REQUIREMENTS

Compute nodes are responsible for running virtual machine instances after they are launched. Compute nodes require bare metal systems that support hardware virtualization. Compute nodes must also have enough memory and disk space to support the requirements of the virtual machine instances that they host.

NOTE

Red Hat OpenStack Services on OpenShift (RHOSO) 18.0 does not support using QEMU architecture emulation.

Processor

64-bit x86 processor with support for the Intel 64 or AMD64 CPU extensions, and the AMD-V or Intel VT hardware virtualization extensions enabled. It is recommended that this processor has a minimum of 4 cores.

Memory

A minimum of 6 GB of RAM for the host operating system, plus additional memory to accommodate for the following considerations:

- Add additional memory that you intend to make available to virtual machine instances.
- Add additional memory to run special features or additional resources on the host, such as additional kernel modules, virtual switches, monitoring solutions, and other additional background tasks.
- If you intend to use non-uniform memory access (NUMA), Red Hat recommends 8GB per CPU socket node or 16 GB per socket node if you have more than 256 GB of physical RAM.
- Configure at least 4 GB of swap space.

For more information about planning for Compute node memory configuration, see Configuring the Compute service for instance creation.

Disk space

A minimum of 50 GB of available disk space.

Network Interface Cards

A minimum of one 1 Gbps Network Interface Cards, although it is recommended to use at least two NICs in a production environment. Use additional network interface cards for bonded interfaces or to delegate tagged VLAN traffic.

Platform management

Compute nodes that are installer-provisioned require a supported platform management interface, such as an Intelligent Platform Management Interface (IPMI) functionality, on the server motherboard. This interface is not required for pre-provisioned nodes.

CHAPTER 4. PLANNING YOUR NETWORKS

Before you deploy RHOSO, take inventory of your networking requirements and the overall environment to inform your network design decisions.

4.1. DEFAULT PHYSICAL NETWORKS

The following physical data center networks are typically implemented for a Red Hat OpenStack Services on OpenShift (RHOSO) deployment:

- Control plane network
- External network (optional)
- Internal API network
- Storage network
- Tenant (project) network
- Storage management network (optional)

For more information, see Default Red Hat OpenStack Services on OpenShift networks in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

4.2. RHOSO NETWORK ISOLATION

You must plan how your deployment hosts specific types of network traffic in isolation. This includes planning IP ranges, subnets, and virtual IPs, and configuring your NIC layout.

The Red Hat OpenStack Services on OpenShift (RHOSO) control plane services run as a Red Hat OpenShift Container Platform (RHOCP) workload. On the control plane, you use the NMState Operator to connect the worker nodes to the required isolated networks. You create a NetworkAttachmentDefinition (nad) custom resource (CR) for each isolated network to attach service pods to the isolated networks, where needed. You use the MetalLB Operator to expose internal service endpoints on the isolated networks. By default, the public service endpoints are exposed as RHOCP routes.

You must also create an **L2Advertisement** resource to define how the VIPs are announced, and an **IpAddressPool** resource to configure which IPs can be used as VIPs. In layer 2 mode, one node assumes the responsibility of advertising a service to the local network.

For more information, see Preparing RHOCP for RHOSO network isolation in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

To create the data plane network, you define a NetConfig custom resource (CR) and specify all the subnets for the data plane networks. You must define at least one control plane network for your data plane. You can also define VLAN networks to create network isolation for composable networks, such as InternalAPI, Storage, and External. Each network definition must include the IP address assignment.

For more information, see Creating the data plane network in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

4.3. NICS

A compact RHOSO deployment requires at least two NICs on each RHOSO control plane worker node.

One NIC on each worker node serves OpenShift. It provides connection between OpenShift components in the OpenShift cluster network.

The other NIC serves OpenStack. It connects the OpenStack services running on the worker nodes to the isolated networks on the RHOSO data plane.

4.3.1. Scaling considerations

Network requirements vary based on environment and business requirements. For example, you may require the following networking capabilities:

- Dedicated NICs on RHOCP worker nodes for particular RHOSP isolated networks.
- Port switches with VLANs for the required isolated networks.

Consult with your RHOCP and network administrators about whether these are requirements in your deployment. Each Compute node requires at least one NIC. You can scale up to provide connections to the isolated networks.

4.3.2. Troubleshooting Compute hostname change detection

If you start the Compute service (nova) and your Compute host detects a name change, you receive the following error message:

```
Found 7 instances on hypervisor but no ComputeNode record exists with host
compute01.example.com node compute01.example.com; potential rename detected - aborting
startup
```

To recover from this error, you must know the reason for the change of the hostname. Here are the three possible reasons for the change of the hostname:

- You have changed the name of the Compute host.
- The DNS is not working correctly and has assigned a different name to the Compute host.
- There is another deployment issue causing a Compute hostname to be incorrect.

When you resolve the issue, you must restart the Compute service.

NOTE

The **nova-libvirt** container contains information about the hostname. If you resolve an issue with a Compute hostname, restart both the **nova-libvirt** and **nova-compute** containers to ensure that all components contain the correct name.

For further assistance, contact the Red Hat Technical Support Team .

4.4. PLANNING FOR IRONIC AND BAREMETAL PROVISIONING NETWORKS

You need to plan for provisioning bare-metal data plane nodes. For more information, see the following additional resources:

- Provisioning bare-metal data plane nodes in the *Deploying Red Hat OpenStack Services on OpenShift* guide.

- Deploying installer-provisioned clusters on bare metal in *Installing:Installing and configuring OpenShift Container Platform clusters.*

- Scaling a user-provisioned cluster with the Bare Metal Operator in the Red Hat OpenShift Container Platform (RHOCP) *Installing* guide.

4.5. STORAGE NETWORK PLANNING CONSIDERATIONS

For more information, see *Storage networks* in this guide.

4.6. NETWORK FUNCTIONS VIRTUALIZATION (NFV)

Network functions virtualization (NFV) is a software-based solution that helps communication service providers (CSPs) to move beyond the traditional, proprietary hardware to achieve greater efficiency and agility and to reduce operational costs.

Using NFV in a Red Hat OpenStack Services on OpenShift (RHOSO) environment allows for IT and network convergence by providing a virtualized infrastructure that uses the standard virtualization technologies to virtualize network functions (VNFs) that run on hardware devices such as switches, routers, and storage. An NFV environment takes advantage of Data Plane Development Kit (DPDK) and Single Root I/O Virtualization (SR-IOV) technologies to improve packet processing speeds.

If you choose an NFV deployment, you must use Deploying a Network Functions Virtualization environment as your deployment guide instead of *Deploying Red Hat OpenStack Services on OpenShift* .

4.7. ADDITIONAL RESOURCES FOR RHOSO NETWORK PLANNING

- Kubernetes NMState Operator

- The Kubernetes NMState project

- Load balancing with MetalLB

- MetalLB documentation

- MetalLB in layer 2 mode

- Specify network interfaces that LB IP can be announced from

- Multiple networks

- Using the Multus CNI in OpenShift

- macvlan plugin

- whereabouts IPAM CNI plugin - Extended configuration

- About advertising for the IP address pools

CHAPTER 5. FEDERAL INFORMATION PROCESSING STANDARD ON RED HAT OPENSTACK SERVICES ON OPENSHIFT

The Federal Information Processing Standards (FIPS) is a set of security requirements developed by the National Institute of Standards and Technology (NIST). In Red Hat Enterprise Linux 9, the supported standard is FIPS publication 140-3: *Security Requirements for Cryptographic Modules*. For details about the supported standard, see the Federal Information Processing Standards Publication 140-3.

FIPS 140-3 validated cryptographic modules are cryptographic libraries that have completed the NIST CMVP process and have received a certificate from NIST. For current information on Red Hat FIPS 140 validated modules, see Compliance Activities and Government Standards.

FIPS is enabled by default in Red Hat OpenStack Services on OpenShift (RHOSO) when RHOSO is installed on a FIPS enabled Red Hat OpenShift Container Platform (RHOCP) cluster. You must enable FIPS on the initial install of RHOCP. For more information on installing a RHOCP cluster in FIPS mode, see Installing a cluster in FIPS mode .

When you use the system-wide cryptographic policy, **FIPS 140 mode**, RHEL and CoreOS are designed to restrict the use of core cryptographic modules and libraries to those that have been FIPS-validated. Paramiko however, implements cryptographic functions in code, and has not been FIPS-validated. RHOSO core components use the RHEL cryptographic libraries submitted to NIST for FIPS validation unless they call paramiko.

5.1. PREPARING TO INSTALL A FIPS ENABLED RED HAT OPENSTACK SERVICES ON OPENSHIFT CONTROL PLANE

Before you install the Red Hat OpenStack Services on OpenShift (RHOSO) control plane, you must modify **iscsi.conf** to remove MD5 and SHA1. The iSCSId configuration for the control plane is not handled by RHOSO operators, so you must complete this step on the Red Hat OpenShift Container Platform (RHOCP) cluster.

Prerequisites

- You have a pre-existing RHOCP cluster with FIPS enabled. For more information about FIPS on RHOCP, see Support for FIPS cryptography.

Procedure

- On each of your nodes, ensure that the value of **node.session.auth.chap_algs** in the **/etc/iscsi/iscsi.conf** file is set to **SHA3-256,SHA256**.

5.2. VERIFICATION OF FIPS STATUS

You can check the FIPS status of RHOCP or deployed worker nodes.

Procedure

1. Log in to your Red Hat OpenShift Container Platform (RHOCP) cluster with an account with cluster-admin privileges.

2. Get a list of the nodes in the cluster:

```
$ oc get nodes
```

Example output:

```
NAME   STATUS   ROLES             AGE VERSION
master1   Ready control-plane,master   7d1h   v1.28.6+6216ea1
master2   Ready control-plane,master   7d1h   v1.28.6+6216ea1
master3   Ready control-plane,master   7d1h   v1.28.6+6216ea1
worker1   Ready worker             7d1h   v1.28.6+6216ea1
worker2   Ready worker             7d1h   v1.28.6+6216ea1
worker3   Ready worker
```

3. Open a debug pod on one of the nodes shown in the output of the previous step:

```
$ oc debug node/worker2
```

Example output:

```
Temporary namespace openshift-debug-rq2m8 is created for debugging node...
Starting pod/worker2-debug-5shqt ...
To use host binaries, run `chroot /host`
Pod IP: 192.168.50.112
If you don't see a command prompt, try pressing enter.
sh-5.1#
```

4. Check for **fips_enabled** in **/proc**

```
sh-5.1# cat /proc/sys/crypto/fips_enabled
```

Example output. **1** is displayed for enabled, **0** for disabled:

```
1
```

For more information about installing Red Hat OpenShift Cluster Platform in FIPS mode, see Support for FIPS cryptography in the RHOCP *Installing* guide.

CHAPTER 6. PLANNING STORAGE AND SHARED FILE SYSTEMS

Red Hat OpenStack Services on OpenShift (RHOSO) uses ephemeral and persistent storage to service the storage needs of the deployment.

Ephemeral storage is associated with a specific Compute instance. When that instance is terminated, so is the associated ephemeral storage. This type of storage is useful for runtime requirements, such as storing the operating system of an instance.

Persistent storage is designed to survive (persist) independent of any running instance. This storage is used for any data that needs to be reused, either by different instances or beyond the life of a specific instance.

The storage requirements of the deployment should be taken into consideration and carefully planned before beginning the deployment. This includes considerations such as:

- Supported features and topologies
- Storage technologies
- Networking
- Scalability
- Accessibility
- Performances
- Costs
- Security
- Redundancy and disaster recovery
- Storage management

6.1. SUPPORTED STORAGE FEATURES AND TOPOLOGIES

RHOSO supports the following storage and networking features:

- Red Hat Ceph Storage integration:
  - Ceph Block Device (RBD) with the Block Storage service (cinder) for persistent storage, the Image service (glance), and the Compute service (nova) for ephemeral storage.
  - Ceph File System (Native CephFS or CephFS via NFS) with the Shared File Systems service (manila).
  - Object Storage service integration with Ceph Object Gateway (RGW)
  - Hyperconverged infrastructure (HCI): Hyperconverged infrastructures consist of hyperconverged nodes. Hyperconverged nodes are external data plane nodes with Compute and Red Hat Ceph Storage services colocated on the same nodes for optimized hardware footprint.

- Transport protocols for the Block Storage service with appropriate configuration and drivers:
    - NVMe over TCP
    - RBD
    - NFS
    - FC

        > **NOTE**
        >
        > You must install host bus adapters (HBAs) on all Compute and OCP workers nodes in any deployment that uses the Block Storage service and a Fibre Channel (FC) back end.

    - iSCSI
- Multipathing with iSCSI, FC, and NVMe over TCP is available on the control plane with the appropriate RHOCP MachineConfig.
- Transport protocols for the Shared File Systems service with appropriate configuration and drivers:
    - CephFS
    - NFS
    - CIFS
- Object Storage through native Swift or Amazon S3 compatible API

RHOSO supports the following storage services.

Service	Back ends
Image service (glance)	● Red Hat Ceph Storage RBD ● Block Storage (cinder) ● Object Storage (swift) ● NFS
Compute service (nova)	● local file storage ● Red Hat Ceph Storage RBD

Service	Back ends
Block Storage service (cinder)	● Red Hat Ceph Storage RBD ● Fiber Channel ● iSCSI ● NFS ● NVMe over TCP **NOTE** Support is provided through third party drivers.
Shared File Systems service (manila)	● Red Hat Ceph Storage CephFS ● Red Hat Ceph Storage CephFS-NFS ● NFS or CIFS through third party vendor storage systems
Object Storage service (swift)	● disks on external data plane nodes ● PersistentVolumes (PVs) on OpenShift nodes (default) ● Integration with Ceph RGW

6.2. STORAGE TECHNOLOGIES

RHOSO supports a number of storage technologies that can act separately or in combination to provide the storage solution for your deployment.

6.2.1. Red Hat Ceph Storage

Red Hat Ceph Storage is a distributed data object store designed for performance, reliability, and scalability. Distributed object stores use unstructured data to simultaneously service modern and legacy object interfaces. It provides access to block, file, and object storage.

Red Hat Ceph Storage is deployed as a cluster. A cluster consists of two primary types of daemons:

- Ceph Object Storage Daemon (CephOSD) - The CephOSD performs data storage, data replication, rebalancing, recovery, monitoring, and reporting tasks.

- Ceph Monitor (CephMon) - The CephMon maintains the primary copy of the cluster map with the current state of the cluster.

RHOSO supports Red Hat Ceph Storage 7 in the following deployment scenarios:

- Integration with an externally deployed Red Hat Ceph Storage 7 cluster.
- A hyperconverged infrastructure (HCI) environment that consists of external data plane nodes that have Compute and Red Hat Ceph Storage services colocated on the same nodes for optimized resource use.

NOTE

Red Hat OpenStack Services on OpenShift (RHOSO) 18.0 supports erasure coding with Red Hat Ceph Storage Object Gateway (RGW). Erasure coding with the Red Hat Ceph Storage Block Device (RDB) is not currently supported.

For more information about Red Hat Ceph Storage architecture, see the Red Hat Ceph Storage 7 Architecture Guide.

6.2.2. Block storage (cinder)

The Block Storage service (cinder) allows users to provision block storage volumes on back ends. Users can attach volumes to instances to augment their ephemeral storage with general-purpose persistent storage. You can detach and re-attach volumes to instances, but you can only access these volumes through the attached instance.

You can also configure instances so that they do not use ephemeral storage. Instead of using ephemeral storage, you can configure the Block Storage service to write images to a volume. You can then use the volume as a bootable root volume for an instance. Volumes also provide inherent redundancy and disaster recovery through backups and snapshots. However, backups are only provided if you deploy the optional Block Storage backup service. In addition, you can encrypt volumes for added security.

6.2.3. Images (glance)

The Image service (glance) provides discovery, registration, and delivery services for instance images. It also provides the ability to store snapshots of instances ephemeral disks for cloning or restore purposes. You can use stored images as templates to commission new servers quickly and more consistently than installing a server operating system and individually configuring services.

6.2.4. Object Storage (swift)

The Object Storage service (swift) provides a fully-distributed storage solution that you can use to store any kind of static data or binary object; such as media files, large datasets, and disk images. The Object Storage service organizes objects by using object containers, which are similar to directories in a file system, but they cannot be nested. You can use the Object Storage service as a repository for nearly every service in the cloud.

Red Hat Ceph Storage RGW can be used as an alternative to the Object Storage service.

6.2.5. Shared File Systems (manila)

The Shared File Systems service (manila) provides the means to provision remote, shareable file systems. These are known as shares. Shares allow projects in the cloud to share POSIX compliant storage, and they can be consumed by multiple instances simultaneously.

Shares are used for instance consumption, and they can be consumed by multiple instances at the same time with read/write access mode.

6.3. STORAGE NETWORKS

Two default storage-related networks are configured during the RHOSO installation: the Storage and Storage Management networks. These isolated networks follow best practices for network connectivity between storage components and the deployments.

The Storage network is used for data storage access and retrieval.

The Storage Management network is used by RHOSO services to have access to specific interfaces in the storage solution that allows access to the management consoles. For example, Red Hat Ceph Storage uses the Storage Management network in a hyperconverged infrastructure (HCI) environment as the cluster_network to replicate data.

The following table lists the properties of the default storage-related networks.

Network name	VLAN	CIDR	NetConfig allocationRange	MetalLB IPAddressPool range	nad ipam range	OCP worker nncp range
storage	21	172.18.0.0/24	172.18.0.100 - 172.18.0.250	N/A	172.18.0.30 - 172.18.0.70	172.18.0.10 - 172.18.0.20
storageMgmt	23	172.20.0.0/24	172.20.0.100 - 172.20.0.250	N/A	172.20.0.30 - 172.20.0.70	172.20.0.10 - 172.20.0.20

Your storage solution may require additional network configurations. These defaults provide a basis for building a full deployment.

All Block Storage services with backends (cinder-volume and cinder-backup) require access to all the storage networks, which may not include the storage management network depending on the backend. Block Storage services with backends require access only to their storage management network. In most deployments there’s a single management network, but if there are multiple storage management networks, each service-backend pair only needs access to their respective management network.

You must install host bus adapters (HBAs) on all OCP worker nodes in any deployment that uses the Block Storage service and a Fibre Channel (FC) back end.

6.3.1. Planning networking for the Block Storage service

Storage best practices recommend using two different networks:

- One network for data I/O
- One network for storage management

These networks are referred to as **storage** and **storageMgmt**. If your deployment diverges from the architecture of two networks, adapt the documented examples as necessary. For example, if the management interface for the storage system is available on the storage network, replace **storageMgmt** with **storage** when there is only one network, and remove **storageMgmt** when the storage network is already present.

The storage services in Red Hat OpenStack Services on OpenShift (RHOSO), with the exception of the Object Storage service (swift), require access to the **storage** and **storageMgmt** networks. You can configure the **storage** and **storageMgmt** networks in the **networkAttachments** field of your **OpenStackControlPlane** CR. The **networkAttachments** field accepts a list of strings with all the networks the component requires access to. Different components can have different network requirements, for example, the Block Storage service (cinder) API component does not require access to any of the storage networks.

The following example shows the **networkAttachments** for Block Storage volumes:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      cinderVolumes:
        iscsi:
          networkAttachments:
          - storage
          - storageMgmt
```

6.3.2. Planning networking for the Shared File Systems service

Plan the networking on your cloud to ensure that cloud users can connect their shares to workloads that run on Red Hat OpenStack Services on OpenShift (RHOSO) virtual machines, bare-metal servers, and containers.

Depending on the level of security and isolation required for cloud users, you can set the **driver_handles_share_servers** parameter (DHSS) to **true** or **false**.

6.3.2.1. Setting DHSS to true

If you set the DHSS parameter to **true**, you can use the Shared File Systems service to export shares to end-user defined share networks with isolated share servers. Users can provision their workloads on self-service share networks to ensure that isolated NAS file servers on dedicated network segments export their shares.

As a project administrator, you must ensure that the physical network to which you map the isolated networks extends to your storage infrastructure. You must also ensure that the storage system that you are using supports network segments. Storage systems, such as NetApp ONTAP and Dell EMC PowerMax, Unity, and VNX, do not support virtual overlay segmentation styles such as GENEVE or VXLAN.

As an alternative to overlay networking, you can do any of the following:

- Use VLAN networking for your project networks.
- Allow VLAN segments on shared provider networks.
- Provide access to a pre-existing segmented network that is already connected to your storage system.

6.3.2.2. Setting DHSS to false

If you set the DHSS parameter to **false**, cloud users cannot create shares on their own share networks. You can create a dedicated shared storage network, and cloud users must connect their clients to the configured network to access their shares.

Not all Shared File System storage drivers support both **DHSS=true** and **DHSS=false**. Both **DHSS=true** and **DHSS=false** ensure data path multi-tenancy isolation. However, if you require network path multi-tenancy isolation for tenant workloads as part of a self-service model, you must deploy the Shared File Systems service with back ends that support **DHSS=true**.

For information about network connectivity to the share, see Ensuring network connectivity to the share.

6.3.2.3. Ensuring network connectivity to the share

To connect to a file share, clients must have network connectivity to one or more of the export locations for that share. When administrators set the **driver_handles_share_servers** parameter (DHSS) for a share type to **true**, cloud users can create a share network with the details of a network to which the Compute instance attaches. Cloud users can then reference the share network when creating shares.

When administrators set the DHSS parameter for a share type to **false**, cloud users must connect their Compute instance to the shared storage network that the project administrator has configured. For more information about how to configure and validate network connectivity to a shared network, see Connecting to a shared network to access shares.

6.4. SCALABILITY AND BACK-END STORAGE

In general, a clustered storage solution provides greater back end scalability and resiliency. For example, when you use Red Hat Ceph Storage as a Block Storage (cinder) back end, you can scale storage capacity and redundancy by adding more Ceph Object Storage Daemon (OSD) nodes. Block Storage, Object Storage (swift), and Shared File Systems Storage (manila) services support Red Hat Ceph Storage as a back end.

The Block Storage service can use multiple storage solutions as discrete back ends. At the service level, you can scale capacity by adding more back ends.

By default, the Object Storage service consumes space by allocating persistent volumes in the OpenShift underlying infrastructure. It can be configured to use a file system on dedicated storage nodes, and it can use as much space as is available. The Object Storage service supports the XFS and ext4 file systems, and you can scale both file systems to consume as much underlying block storage as is available. You can also scale capacity by adding more storage devices to the storage node.

The Shared File Systems service provisions file shares from designated storage pools that are managed by Red Hat Ceph Storage or other back-end storage systems. You can scale this shared storage by increasing the size or number of storage pools available to the service or by adding more back-end storage systems to the deployment. Each back-end storage system is integrated with a dedicated service to interact with and manage the storage system.

6.5. STORAGE ACCESSIBILITY AND ADMINISTRATION

Volumes are consumed only through instances. Users can extend, create snapshots of volumes and use the snapshots to clone or restore a volume to a previous state.

You can use the Block Storage service (cinder) to create volume types, which aggregate volume settings. You can associate volume types with encryption and Quality of Service (QoS) specifications to provide different levels of performance for your cloud users. Your cloud users can specify the volume type they require when creating new volumes. For example, volumes that use higher performance QoS specifications could provide your users with more IOPS, or your users could assign lighter workloads to volumes that use lower performance QoS specifications to conserve resources. Shares can be consumed simultaneously by one or more instances, bare metal nodes or containers. The Shared File Systems service (manila) also supports share resize, snapshots and cloning, and administrators can create share types to aggregate settings.

Users can access objects in a container by using the Object Storage service (swift) API, and administrators can make objects accessible to instances and services in the cloud. This accessibility makes objects ideal as repositories for services; for example, you can store Image service (glance) images in containers that are managed by the Object Storage service.

6.6. STORAGE SECURITY

The Block Storage service provides data security through the Key Manager service (barbican). The Block Storage service uses a one-to-one, key to volume mapping with the key managed by the Key Manager service. The encryption type is defined when configuring the volume type.

Security can also be improved at the backend level by encrypting control and/or data traffic, for example with Red Hat Ceph Storage, this can be achieved by enabling messengerv2 secure mode. This way, network traffic amongst Ceph services as well as from OpenStack compute nodes are encrypted.

You configure object and container security at the service and node level. The Object Storage service (swift) provides no native encryption for containers and objects. However, with the Key Manager service enabled, the Object Storage service can transparently encrypt and decrypt your stored (at-rest) objects. At-rest encryption is distinct from in-transit encryption in that it refers to the objects being encrypted while being stored on disk.

The Shared File Systems service (manila) can secure shares through access restriction, whether by instance IP, user or group, or TLS certificate. Some Shared File Systems service deployments can feature separate share servers to manage the relationship between share networks and shares. Some share servers support, or even require, additional network security. For example, a CIFS share server requires the deployment of an LDAP, Active Directory, or Kerberos authentication service.

Some backends also support encrypting the data AT REST. This enables extra security by encrypting the backend disks themselves, preventing physical security threats such as theft or unwiped recycled disks.

For more information about configuring security options for the Block Storage service, Object Storage service, and Shared File Systems service, see Configuring security services.

6.7. STORAGE REDUNDANCY AND DISASTER RECOVERY

If you deploy the optional Block Storage backup service, then the Block Storage service (cinder) provides volume backup and restoration for basic disaster recovery of user storage. You can use backups to protect volume contents. The Block Storage service also supports snapshots. In addition to cloning, you can use snapshots to restore a volume to a previous state.

If your environment includes multiple back ends, you can also migrate volumes between these back ends. This is useful if you need to take a back end offline for maintenance. Backups are typically stored in a storage back end separate from their source volumes to help protect the data. This is not possible with snapshots because snapshots are dependent on their source volumes.

The Block Storage service also supports the creation of consistency groups to group volumes together for simultaneous snapshot creation. This provides a greater level of data consistency across multiple volumes.

NOTE

Red Hat does not currently support Block Storage service replication.

The Object Storage service (swift) provides no built-in backup features. You must perform all backups at the file system or node level. However, the Object Storage service features robust redundancy and fault tolerance. Even the most basic deployment of the Object Storage service replicates objects multiple times. You can use failover features like device mapper multipathing (DM Multipath) to enhance redundancy.

The Shared File Systems service (manila) provides no built-in backup features for shares, but you can create snapshots for cloning and restoration.

6.8. MANAGING THE STORAGE SOLUTION

You can manage your RHOSO configuration using the RHOSO Dashboard (horizon) or the RHOSO command line interface (CLI). You can perform most procedures using either method but some advanced procedures can only be completed using the CLI.

You can manage your storage solution configuration using the dedicated management interface provided by the storage vendor.

6.9. SIZING RED HAT OPENSHIFT STORAGE

The Image and Object Storage services can be configured to allocate space in the Red Hat OpenShift backing storage. In this scenario, the Red Hat OpenShift storage sizing should be estimated based on the expected use of these services.

6.9.1. Image service considerations

The Image service (glance) requires a staging area to manipulate data during an import operation. It is possible to copy image data into multiple stores so some persistence is required for the Image service. Although PVCs represent the main storage model for the Image service, an External model can also be chosen.

External model

If External is chosen, no PVCs are created and the Image service acts like a stateless instance with no persistence provided. In this instance, persistence must be provided using **extraMounts**. NFS is often used to provide persistence. It can be mapped to **/var/lib/glance**:

```
...
default:
  storage:
    external: true
...
...
extraMounts:
- extraVol:
  - extraVolType: NFS
    mounts:
    - mountPath: /var/lib/glance/os_glance_staging_store
      name: nfs
    volumes:
    - name: nfs
```

```
      nfs:
        path: <nfs_export_path>
        server: <nfs_ip_address>
```

- Replace **<nfs_export_path>** with the export path of your NFS share.
- Replace **<nfs_ip_address>** with the IP address of your NFS share. This IP address must be part of the overlay network that is reachable by the Image service.

It should be noted that the configuration sample conflicts with the distributed image import feature. Distributed image import requires RWO storage plugged into a particular instance; it owns the data and receives requests in case its staged data is required for an upload operation. When the External model is adopted, if Red Hat Ceph Storage is used as a backend, and an image conversion operation is run in one of the existing replicas, the **glance-operator** does not have to make any assumption about the underlying storage that is tied to the staging area, and the conversion operation that uses the **os_glance_staging_store directory** (within the Pod) interacts with the RWX NFS backend provided via **extraMounts**. With this scenario, no image-cache PVC can be requested and mounted to a subPath, because it should be the administrator's responsibility to plan for persistence using **extraMounts**.

PVC model

The PVC model is the default. When a GlanceAPI instance is deployed, a PVC is created and bound to **/var/lib/glance** according to the **storageClass** and **storageRequest** passed as input.

```
...
default:
  replicas: 3
  storage:
    storageRequest: 10G
...
```

In this model, if Red Hat Ceph Storage is set as a backend, no dedicated image conversion PVC is created. The administrator must think about the PVC sizing in advance; the size of the PVC should be at least up to the largest converted image size. Concurrent conversions within the same Pod might be problematic in terms of PVC size; a conversion will fail or cannot take place if the PVC is full and there's not enough space. The upload should be retried after the previous conversion is over and the staging area space is released. However, concurrent conversion operations might happen in different Pods. You should deploy at least 3 replicas for a particular **glanceAPI**. This helps to handle heavy operations like image conversion.

For a PVC-based layout, the scale out of a **glanceAPI** in terms of replicas is limited by the available storage provided by the **storageClass**, and depends on the **storageRequest**. The **storageRequest** is a critical parameter, it can be globally defined for all the **glanceAPI**, or defined with a different value for each API. It will influence the scale out operations for each of them. Other than a local PVC required for the staging area, it is possible to enable image cache, which is translated into an additional PVC bound to each **glanceAPI** instance. A **glance-cache** PVC is bound to **/var/lib/glance/image-cache**. The glance-operator configures the glanceAPI instance accordingly, setting both **image_cache_max_size** and the **image_cache_dir** parameters. The number of image cache PVCs follows the same rules described for the local PVC, and the number of requested PVCs is proportional to the number of replicas.

6.9.2. Object Storage service considerations

The Object Storage service requires storage devices for data. These devices must be accessible using the same hostname or IP address during their lifetime. The configuration of a StatefulSet with a Headless Service is how this is achieved.

If you want to use storage volumes to provide persistence for your workload, you can use a StatefulSet as part of the solution. Although individual Pods in a StatefulSet are susceptible to failure, the persistent Pod identifiers make it easier to match existing volumes to the new Pods that replace any that have failed.

The Object Storage service requires quite a few services to access these PVs, and all of them are running in a single pod.

Additionally, volumes are not deleted if the StatefulSet is deleted. An unwanted removal of the StatefulSet (or the whole deployment) will not immediately result in a catastrophic data loss, but can be recovered from with administrator interaction.

The Headless Service makes it possible to access the storage pod directly by using a DNS name. For example, if the pod name is **swift-storage-0** and the **SwiftStorage** instance is named **swift-storage**, it becomes accessible using **swift-storage-0.swift-storage**. This makes it easily usable within the Object Storage service rings, and IP changes are now transparent and don’t require an update of the rings.

Parallel pod management tells the StatefulSet controller to launch or terminate all Pods in parallel, and to not wait for Pods to become **Running** and **Ready** or completely terminated prior to launching or terminating another Pod. This option only affects the behavior for scaling operations. Updates are not affected.

This is required to scale by more than one; including new deployments with more than one replica. It is required to create all pods at the same time, otherwise there will be PVCs that are not bound and the Object Storage service rings cannot be created, eventually blocking the start of these pods.

Storage pods should be distributed to different nodes to avoid single points of failure. A **podAntiAffinity** rule with **preferredDuringSchedulingIgnoredDuringExecution** is used to distribute pods to different nodes if possible. Using a separate **storageClass** and PersistentVolumes that are located on different nodes can be used to enforce further distribution.

Object Storage service backend services must only be accessible by other backend services and the Object Storage service proxy. To limit access, a **NetworkPolicy** is added to allow only traffic between these pods. The **NetworkPolicy** itself depends on labels, and these must match to allow traffic. Therefore labels must not be unique; instead all pods must use the same label to allow access. This is also the reason why the **swift-operator** is not using labels from **lib-common**.

Object Storage service rings require information about the disks to use, and this includes sizes and hostnames or IPs. Sizes are not known when starting the StatefulSet using PVCs, the size requirement is a lower limit, but the actual PVs might be much bigger.

However, StatefulSets do create PVCs before the **ConfigMaps** are available and simply wait starting the pods until these become available. The **SwiftRing** reconciler is watching the **SwiftStorage** instances and iterates over PVCs to get actual information about the used disks. Once these are bound, the size is known and the **swift-ring-rebalance** job creates the Swift rings and eventually the **ConfigMap**. After the **ConfigMap** becomes available, StatefulSets will start the service pods.

Rings are stored in a **ConfigMap** mounted by the **SwiftProxy** and **SwiftStorage** instances using projected volumes. This makes it possible to mount all required files at the same place, without merging these from other places. Updated **ConfigMaps** will update these files, and these changes are are detected by the Swift services eventually reloading these.

Some operators are using the **customServiceConfig** option to customize settings. However, the **SwiftRing** instance deploys multiple backend services, and each of these requires specific files to be customized. Therefore only **defaultConfigOverwrite** using specific keys as filenames is supported when using the **swift-operator**.

CHAPTER 7. INTEGRATION

You can integrate Red Hat OpenStack Services on OpenShift (RHOSO) with the following third-party software - Tested and Approved Software

You can deploy RHOSO on trusted cloud providers. For the certified list of products, see Hardware - Tested and Approved.

CHAPTER 8. SUBSCRIPTIONS

To install Red Hat OpenStack Services on OpenShift (RHOSO), you must register all systems in the RHOSO environment with Red Hat Subscription Manager, and subscribe to the required channels.