Red Hat OpenShift Data Foundation 4.19

Planning your deployment

Important considerations when deploying Red Hat OpenShift Data Foundation 4.19

Last Updated: 2026-02-18

Red Hat OpenShift Data Foundation 4.19 Planning your deployment

Important considerations when deploying Red Hat OpenShift Data Foundation 4.19

Abstract

Read this document for important considerations when planning your Red Hat OpenShift Data Foundation deployment.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Do let us know how we can make it better.

To give feedback, create a Jira ticket:

1. Log in to the Jira.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the Summary field.
4. Enter your suggestion for improvement in the Description field. Include links to the relevant parts of the documentation.
5. Select **Documentation** in the Components field.
6. Click Create at the bottom of the dialogue.

CHAPTER 1. INTRODUCTION TO OPENSHIFT DATA FOUNDATION

Red Hat OpenShift Data Foundation is a highly integrated collection of cloud storage and data services for Red Hat OpenShift Container Platform. It is available as part of the Red Hat OpenShift Container Platform Service Catalog, packaged as an operator to facilitate simple deployment and management.

Red Hat OpenShift Data Foundation services are primarily made available to applications by way of storage classes that represent the following components:

- Block storage devices, catering primarily to database workloads. Prime examples include Red Hat OpenShift Container Platform logging and monitoring, and PostgreSQL.

IMPORTANT

Block storage should be used for any worklaod only when it does not require sharing the data across multiple containers.

- Shared and distributed file system, catering primarily to software development, messaging, and data aggregation workloads. Examples include Jenkins build sources and artifacts, Wordpress uploaded content, Red Hat OpenShift Container Platform registry, and messaging using JBoss AMQ.
- Multicloud object storage, featuring a lightweight S3 API endpoint that can abstract the storage and retrieval of data from multiple cloud object stores.
- On premises object storage, featuring a robust S3 API endpoint that scales to tens of petabytes and billions of objects, primarily targeting data intensive applications. Examples include the storage and access of row, columnar, and semi-structured data with applications like Spark, Presto, Red Hat AMQ Streams (Kafka), and even machine learning frameworks like TensorFlow and Pytorch.

NOTE

Running PostgresSQL workload on CephFS persistent volume is not supported and it is recommended to use RADOS Block Device (RBD) volume. For more information, see the knowledgebase solution ODF Database Workloads Must Not Use CephFS PVs/PVCs .

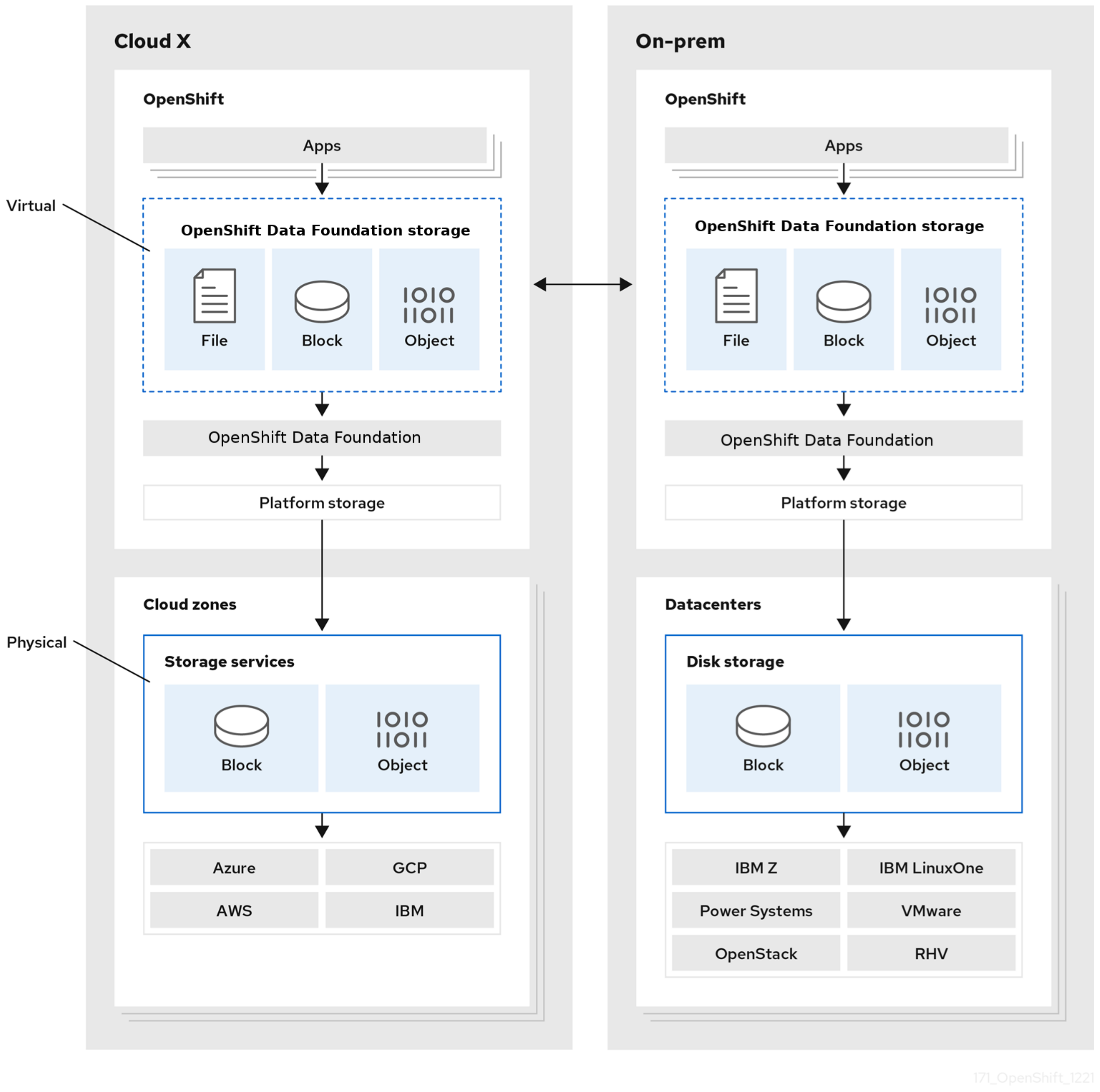
Red Hat OpenShift Data Foundation version 4.x integrates a collection of software projects, including:

- Ceph, providing block storage, a shared and distributed file system, and on-premises object storage
- Ceph CSI, to manage provisioning and lifecycle of persistent volumes and claims
- NooBaa, providing a Multicloud Object Gateway
- OpenShift Data Foundation, Rook-Ceph, and NooBaa operators to initialize and manage OpenShift Data Foundation services.

CHAPTER 2. ARCHITECTURE OF OPENSHIFT DATA FOUNDATION

Red Hat OpenShift Data Foundation provides services for, and can run internally from the Red Hat OpenShift Container Platform.

Figure 2.1. Red Hat OpenShift Data Foundation architecture



Red Hat OpenShift Data Foundation supports deployment into Red Hat OpenShift Container Platform clusters deployed on installer-provisioned or user-provisioned infrastructure.

For details about these two approaches, see OpenShift Container Platform - Installation process.

To ensure your environment meets the requirements, consult the Red Hat OpenShift Data Foundation Supportability and Interoperability Checker to verify component compatibility, supported limits, features, and integration matrices.

For information about the architecture and lifecycle of OpenShift Container Platform, see OpenShift Container Platform architecture.

TIP

For IBM Power, see Installing on IBM Power .

2.1. ABOUT OPERATORS

Red Hat OpenShift Data Foundation comprises of three main operators, which codify administrative tasks and custom resources so that you can easily automate the task and resource characteristics. Administrators define the desired end state of the cluster, and the OpenShift Data Foundation operators ensure the cluster is either in that state, or approaching that state, with minimal administrator intervention.

OpenShift Data Foundation operator

A meta-operator that draws on other operators in specific tested ways to codify and enforce the recommendations and requirements of a supported Red Hat OpenShift Data Foundation deployment. The rook-ceph and noobaa operators provide the storage cluster resource that wraps these resources.

Rook-ceph operator

This operator automates the packaging, deployment, management, upgrading, and scaling of persistent storage and file, block, and object services. It creates block and file storage classes for all environments, and creates an object storage class and services Object Bucket Claims (OBCs) made against it in on-premises environments.

Additionally, for internal mode clusters, it provides the ceph cluster resource, which manages the deployments and services representing the following:

- Object Storage Daemons (OSDs)
- Monitors (MONs)
- Manager (MGR)
- Metadata servers (MDS)
- RADOS Object Gateways (RGWs) on-premises only

Multicloud Object Gateway operator

This operator automates the packaging, deployment, management, upgrading, and scaling of the Multicloud Object Gateway (MCG) object service. It creates an object storage class and services the OBCs made against it.

Additionally, it provides the NooBaa cluster resource, which manages the deployments and services for NooBaa core, database, and endpoint.

NOTE

OpenShift Data Foundation's default configuration for MCG is optimized for low resource consumption and not performance. If you plan to use MCG often, see information about increasing resource limits in the knowledebase article Performance tuning guide for Multicloud Object Gateway.

2.2. STORAGE CLUSTER DEPLOYMENT APPROACHES

The growing list of operating modalities is an evidence that flexibility is a core tenet of Red Hat OpenShift Data Foundation. This section provides you with information that will help you to select the most appropriate approach for your environments.

You can deploy Red Hat OpenShift Data Foundation either entirely within OpenShift Container Platform (Internal approach) or to make available the services from a cluster running outside of OpenShift Container Platform (External approach).

2.2.1. Internal approach

Deployment of Red Hat OpenShift Data Foundation entirely within Red Hat OpenShift Container Platform has all the benefits of operator based deployment and management. You can use the internal-attached device approach in the graphical user interface (GUI) to deploy Red Hat OpenShift Data Foundation in internal mode using the local storage operator and local storage devices.

Ease of deployment and management are the highlights of running OpenShift Data Foundation services internally on OpenShift Container Platform. There are two different deployment modalities available when Red Hat OpenShift Data Foundation is running entirely within Red Hat OpenShift Container Platform:

- Simple
- Optimized

Simple deployment

Red Hat OpenShift Data Foundation services run co-resident with applications. The operators in Red Hat OpenShift Container Platform manages these applications.

A simple deployment is best for situations where,

- Storage requirements are not clear.
- Red Hat OpenShift Data Foundation services runs co-resident with the applications.
- Creating a node instance of a specific size is difficult, for example, on bare metal.

For Red Hat OpenShift Data Foundation to run co-resident with the applications, the nodes must have local storage devices, or portable storage devices attached to them dynamically, like EBS volumes on EC2, or vSphere Virtual Volumes on VMware, or SAN volumes.

NOTE

PowerVC dynamically provisions the SAN volumes.

Optimized deployment

Red Hat OpenShift Data Foundation services run on dedicated infrastructure nodes. Red Hat OpenShift Container Platform manages these infrastructure nodes.

An optimized approach is best for situations when,

- Storage requirements are clear.

- Red Hat OpenShift Data Foundation services run on dedicated infrastructure nodes.
- Creating a node instance of a specific size is easy, for example, on cloud, virtualized environment, and so on.

2.2.2. External approach

Red Hat OpenShift Data Foundation exposes the Red Hat Ceph Storage services running outside of the OpenShift Container Platform cluster as storage classes.

The external approach is best used when,

- Storage requirements are significant (600+ storage devices).
- Multiple OpenShift Container Platform clusters need to consume storage services from a common external cluster.
- Another team, Site Reliability Engineering (SRE), storage, and so on, needs to manage the external cluster providing storage services. Possibly a pre-existing one.

2.3. NODE TYPES

Nodes run the container runtime, as well as services, to ensure that the containers are running, and maintain network communication and separation between the pods. In OpenShift Data Foundation, there are three types of nodes.

Table 2.1. Types of nodes

Node Type	Description
Master	These nodes run processes that expose the Kubernetes API, watch and schedule newly created pods, maintain node health and quantity, and control interaction with underlying cloud providers.
Infrastructure (Infra)	Infra nodes run cluster level infrastructure services such as logging, metrics, registry, and routing. These are optional in OpenShift Container Platform clusters. In order to separate OpenShift Data Foundation layer workload from applications, ensure that you use infra nodes for OpenShift Data Foundation in virtualized and cloud environments. To create Infra nodes, you can provision new nodes labeled as **infra**. For more information, see How to use dedicated worker nodes for Red Hat OpenShift Data Foundation

Node Type	Description
Worker	Worker nodes are also known as application nodes since they run applications. When OpenShift Data Foundation is deployed in internal mode, you require a minimal cluster of 3 worker nodes. Make sure that the nodes are spread across 3 different racks, or availability zones, to ensure availability. In order for OpenShift Data Foundation to run on worker nodes, you need to attach the local storage devices, or portable storage devices to the worker nodes dynamically. When OpenShift Data Foundation is deployed in external mode, it runs on multiple nodes. This allows Kubernetes to reschedule on the available nodes in case of a failure.

NOTE

OpenShift Data Foundation requires the same number of subsciptions as OpenShift Container Platform. However, if OpenShift Data Foundation is running on infra nodes, OpenShift does not require OpenShift Container Platform subscription for these nodes. Therefore, the OpenShift Data Foundation control plane does not require additional OpenShift Container Platform and OpenShift Data Foundation subscriptions. For more information, see Chapter 6, *Subscriptions*.

CHAPTER 3. INTERNAL STORAGE SERVICES

Red Hat OpenShift Data Foundation service is available for consumption internally to the Red Hat OpenShift Container Platform that runs on the following infrastructure:

- Amazon Web Services (AWS)
- Bare metal
- VMware vSphere
- Microsoft Azure
- Google Cloud
- Red Hat OpenStack 13 or higher (installer-provisioned infrastructure) [Technology Preview]
- IBM Power
- IBM Z and IBM® LinuxONE
- ROSA with hosted control planes (HCP)

Creation of an internal cluster resource results in the internal provisioning of the OpenShift Data Foundation base services, and makes additional storage classes available to the applications.

CHAPTER 4. EXTERNAL STORAGE SERVICES

Red Hat OpenShift Data Foundation can make services from an external Red Hat Ceph Storage cluster available for consumption through OpenShift Container Platform clusters running on the following platforms:

- VMware vSphere
- Bare metal
- Red Hat OpenStack platform (Technology Preview)
- IBM Power
- IBM Z

The OpenShift Data Foundation operators create and manage services to satisfy Persistent Volume (PV) and Object Bucket Claims (OBCs) against the external services. External cluster can serve block, file and object storage classes for applications that run on OpenShift Container Platform. The operators do not deploy or manage the external clusters.

CHAPTER 5. SECURITY CONSIDERATIONS

5.1. FIPS CRYPTOGRAPHY

The Federal Information Processing Standard Publication 140-2/140-3 (FIPS 140-2/140-3) is a standard that defines a set of security requirements for the use of cryptographic modules. Law mandates this standard for the US government agencies and contractors and is also referenced in other international and industry specific standards.

Red Hat OpenShift Data Foundation is designed for FIPS. When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components, including OpenShift Data Foundation, use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures. For more information, see the Support for FIPS cryptography section in OpenShift documentation.

Currently, the Cryptographic Module Validation Program (CMVP) processes the cryptography modules. You can see the state of these modules at Modules in Process List. For more up-to-date information, see the Red Hat Knowledgebase solution RHEL core crypto components.

5.2. PROXY ENVIRONMENT

A proxy environment is a production environment that denies direct access to the internet and provides an available HTTP or HTTPS proxy instead. Red Hat Openshift Container Platform is configured to use a proxy by modifying the proxy object for existing clusters or by configuring the proxy settings in the install-config.yaml file for new clusters.

Red Hat supports deployment of OpenShift Data Foundation in proxy environments when OpenShift Container Platform has been configured according to configuring the cluster-wide proxy.

5.3. DATA ENCRYPTION OPTIONS

Encryption lets you encode your data to make it impossible to read without the required encryption keys. This mechanism protects the confidentiality of your data in the event of a physical security breach that results in a physical media to escape your custody. The per-PV encryption also provides access protection from other namespaces inside the same OpenShift Container Platform cluster. Data is encrypted when it is written to the disk, and decrypted when it is read from the disk. Working with encrypted data might incur a small penalty to performance.

Encryption is only supported for new clusters deployed using Red Hat OpenShift Data Foundation 4.6 or higher. An existing encrypted cluster that is not using an external Key Management System (KMS) cannot be migrated to use an external KMS.

Previously, HashiCorp Vault was the only supported KMS for Cluster-wide and Persistent Volume encryptions. With OpenShift Data Foundation 4.7.0 and 4.7.1, only HashiCorp Vault Key/Value (KV) secret engine API, version 1 is supported. Starting with OpenShift Data Foundation 4.7.2, HashiCorp Vault KV secret engine API, versions 1 and 2 are supported. As of OpenShift Data Foundation 4.12, Thales CipherTrust Manager has been introduced as an additional supported KMS.

IMPORTANT

- KMS is required for StorageClass encryption, and is optional for cluster-wide encryption.
- To start with, Storage class encryption requires a valid Red Hat OpenShift Data Foundation Advanced subscription. For more information, see the knowledgebase article on OpenShift Data Foundation subscriptions .

Red Hat works with the technology partners to provide this documentation as a service to the customers. However, Red Hat does not provide support for the Hashicorp product. For technical assistance with this product, contact Hashicorp.

5.3.1. Cluster-wide encryption

Red Hat OpenShift Data Foundation supports cluster-wide encryption (encryption-at-rest) for all the disks and Multicloud Object Gateway operations in the storage cluster. OpenShift Data Foundation uses Linux Unified Key System (LUKS) version 2 based encryption with a key size of 512 bits and the **aes-xts-plain64** cipher where each device has a different encryption key. The keys are stored using a Kubernetes secret or an external KMS. Both methods are mutually exclusive and you can not migrate between methods.

Encryption is disabled by default for block and file storage. You can enable encryption for the cluster at the time of deployment. The MultiCloud Object Gateway supports encryption by default. See the deployment guides for more information.

OpenShift Data Foundation supports cluster wide encryption with and without Key Management System (KMS). Cluster wide encryption with KMS is supported using the following service providers:

- HashiCorp Vault - Tested with Version 1.9.1 and 1.21.1
- Thales Cipher Trust Manager - Tested with Version 2.9.0+7637, crypto version: 1.5.0

Security common practices require periodic encryption key rotation. OpenShift Data Foundation automatically rotates encryption keys stored in kubernetes secret (non-KMS) and Vault on a weekly basis. However, key rotation for Vault KMS must be enabled after the storage cluster creation and does not happen by default. For more information refer to the deployment guides.

NOTE

Requires a valid Red Hat OpenShift Data Foundation Advanced subscription. To know how subscriptions for OpenShift Data Foundation work, see knowledgebase article on OpenShift Data Foundation subscriptions.

Cluster wide encryption with HashiCorp Vault KMS provides two authentication methods:

- **Token**: This method allows authentication using vault tokens. A kubernetes secret containing the vault token is created in the openshift-storage namespace and is used for authentication. If this authentication method is selected then the administrator has to provide the vault token that provides access to the backend path in Vault, where the encryption keys are stored.
- **Kubernetes**: This method allows authentication with vault using serviceaccounts. If this authentication method is selected then the administrator has to provide the name of the role configured in Vault that provides access to the backend path, where the encryption keys are stored. The value of this role is then added to the **ocs-kms-connection-details** config map.

NOTE

OpenShift Data Foundation on IBM Cloud platform supports Hyper Protect Crypto Services (HPCS) Key Management Services (KMS) as the encryption solution in addition to HashiCorp Vault KMS.

IMPORTANT

Red Hat works with the technology partners to provide this documentation as a service to the customers. However, Red Hat does not provide support for the Hashicorp product. For technical assistance with this product, contact Hashicorp.

5.3.2. Storage class encryption

You can encrypt persistent volumes (block only) with storage class encryption using an external Key Management System (KMS) to store device encryption keys. Persistent volume encryption is only available for RADOS Block Device (RBD) persistent volumes. See how to create a storage class with persistent volume encryption.

Storage class encryption is supported in OpenShift Data Foundation 4.7 or higher with HashiCorp Vault KMS. Storage class encryption is supported in OpenShift Data Foundation 4.12 or higher with both HashiCorp Vault KMS and Thales CipherTrust Manager KMS.

NOTE

Requires a valid Red Hat OpenShift Data Foundation Advanced subscription. To know how subscriptions for OpenShift Data Foundation work, see knowledgebase article on OpenShift Data Foundation subscriptions.

5.3.3. CipherTrust manager

Red Hat OpenShift Data Foundation version 4.12 introduced Thales CipherTrust Manager as an additional Key Management System (KMS) provider for your deployment. Thales CipherTrust Manager provides centralized key lifecycle management. CipherTrust Manager supports Key Management Interoperability Protocol (KMIP), which enables communication between key management systems.

CipherTrust Manager is enabled during deployment.

5.3.4. Data encryption in-transit using Red Hat Ceph Storage's messenger version 2 protocol (msgr2)

Starting with OpenShift Data Foundation version 4.14, Red Hat Ceph Storage's messenger version 2 protocol can be used to encrypt data in-transit. This provides an important security requirement for your infrastructure.

In-transit encryption can be enabled during deployment while the cluster is being created. See the deployment guide for your environment for instructions on enabling data encryption in-transit during cluster creation.

The msgr2 protocol supports two connection modes:

crc

- Provides strong initial authentication when a connection is established with cephx.

- Provides a crc32c integrity check to protect against bit flips.
- Does not provide protection against a malicious man-in-the-middle attack.
- Does not prevent an eavesdropper from seeing all post-authentication traffic.

secure

- Provides strong initial authentication when a connection is established with cephx.
- Provides full encryption of all post-authentication traffic.
- Provides a cryptographic integrity check.

The default mode is **crc**.

5.4. ENCRYPTION IN TRANSIT

You need to enable IPsec so that all the network traffic between the nodes on the OVN-Kubernetes Container Network Interface (CNI) cluster network travels through an encrypted tunnel.

By default, IPsec is disabled. You can enable it either during or after installing the cluster. If you need to enable IPsec after cluster installation, you must first resize your cluster MTU to account for the overhead of the IPsec ESP IP header.

For more information on how to configure the IPsec encryption, see *Configuring IPsec encryption* of the Networking guide in OpenShift Container Platform documentation.

CHAPTER 6. SUBSCRIPTIONS

6.1. SUBSCRIPTION OFFERINGS

Red Hat OpenShift Data Foundation subscription is based on “core-pairs,” similar to Red Hat OpenShift Container Platform. The Red Hat OpenShift Data Foundation 2-core subscription is based on the number of logical cores on the CPUs in the system where OpenShift Container Platform runs.

As with OpenShift Container Platform:

- OpenShift Data Foundation subscriptions are stackable to cover larger hosts.
- Cores can be distributed across as many virtual machines (VMs) as needed. For example, ten 2-core subscriptions will provide 20 cores and in case of IBM Power a 2-core subscription at SMT level of 8 will provide 2 cores or 16 vCPUs that can be used across any number of VMs.
- OpenShift Data Foundation subscriptions are available with Premium or Standard support.

6.2. DISASTER RECOVERY SUBSCRIPTION REQUIREMENT

Disaster Recovery features supported by Red Hat OpenShift Data Foundation require all of the following prerequisites to successfully implement a disaster recovery solution:

- A valid Red Hat OpenShift Data Foundation Advanced entitlement
- A valid Red Hat Advanced Cluster Management for Kubernetes subscription

Any Red Hat OpenShift Data Foundation Cluster containing PVs participating in active replication either as a source or destination requires OpenShift Data Foundation Advanced entitlement. This subscription should be active on both source and destination clusters.

To know how subscriptions for OpenShift Data Foundation work, see knowledgebase article on OpenShift Data Foundation subscriptions.

IMPORTANT

OpenShift Data Foundation deployed with Multus networking is not supported for Regional Disaster Recovery (Regional-DR) setups.

6.3. CORES VERSUS VCPUS AND HYPERTHREADING

Making a determination about whether or not a particular system consumes one or more cores is currently dependent on whether or not that system has hyperthreading available. Hyperthreading is only a feature of Intel CPUs.

For more information, see the Self-managed Red Hat OpenShift subscription guide .

6.4. SPLITTING CORES

Systems that require an odd number of cores need to consume a full 2-core subscription. For example, a system that is calculated to require only 1 core will end up consuming a full 2-core subscription once it is registered and subscribed.

When a single virtual machine (VM) with 2 vCPUs uses hyperthreading resulting in 1 calculated vCPU, a full 2-core subscription is required; a single 2-core subscription may not be split across two VMs with 2 vCPUs using hyperthreading. See section Cores versus vCPUs and hyperthreading for more information.

It is recommended that virtual instances be sized so that they require an even number of cores.

6.4.1. Shared Processor Pools for IBM Power

IBM Power have a notion of shared processor pools. The processors in a shared processor pool can be shared across the nodes in the cluster. The aggregate compute capacity required for a Red Hat OpenShift Data Foundation should be a multiple of core-pairs.

6.5. SUBSCRIPTION REQUIREMENTS

Red Hat OpenShift Data Foundation components can run on either OpenShift Container Platform worker or infrastructure nodes, for which you can use either Red Hat CoreOS (RHCOS) or Red Hat Enterprise Linux (RHEL) 8.4 as the host operating system. RHEL 7 is now deprecated. OpenShift Data Foundation subscriptions are required for every OpenShift Container Platform subscribed core with a ratio of 1:1.

When using infrastructure nodes, the rule to subscribe all OpenShift worker node cores for OpenShift Data Foundation applies even though they don’t need any OpenShift Container Platform or any OpenShift Data Foundation subscriptions. You can use labels to state whether a node is a worker or an infrastructure node.

For more information, see How to use dedicated worker nodes for Red Hat OpenShift Data Foundation in the Managing and Allocating Storage Resources guide.

CHAPTER 7. INFRASTRUCTURE REQUIREMENTS

7.1. PLATFORM REQUIREMENTS

Red Hat OpenShift Data Foundation 4.19 is supported only on OpenShift Container Platform version 4.19 and its next minor versions.

Bug fixes for previous version of Red Hat OpenShift Data Foundation will be released as bug fix versions. For more details, see the Red Hat OpenShift Container Platform Life Cycle Policy .

For external cluster subscription requirements, see the Red Hat Knowledgebase article OpenShift Data Foundation Subscription Guide.

For a complete list of supported platform versions, see the Red Hat OpenShift Data Foundation Supportability and Interoperability Checker.

7.1.1. Amazon EC2

Supports internal Red Hat OpenShift Data Foundation clusters only.

An Internal cluster must meet both, storage device requirements and have a storage class that provides, EBS storage via the aws-ebs provisioner.

OpenShift Data Foundation supports **gp2-csi** and **gp3-csi** drivers that were introduced by Amazon Web Services (AWS). These drivers offer better storage expansion capabilities and a reduced monthly price point (**gp3-csi**). You can now select the new drivers when selecting your storage class. In case a high throughput is required, **gp3-csi** is recommended to be used when deploying OpenShift Data Foundation.

If you need a high input/output operation per second (IOPS), the recommended EC2 instance types are **D2** or **D3**.

7.1.2. Bare Metal

Supports internal clusters and consuming external clusters.

An internal cluster must meet both the storage device requirements and have a storage class that provide local SSD (NVMe/SATA/SAS, SAN) via the Local Storage Operator. For production clusters, a minimum storage capacity of 0.5 TiB per storage device is required.

7.1.3. VMware vSphere

Supports internal clusters and consuming external clusters.

Recommended versions:

- vSphere 8.0 or later
- VMware Cloud Foundation 9 and VMware vSphere Foundation 9 (Technology preview)

For more details, see the VMware vSphere infrastructure requirements.

NOTE

If VMware ESXi does not recognize its devices as flash, mark them as flash devices. Before Red Hat OpenShift Data Foundation deployment, refer to Mark Storage Devices as Flash.

Additionally, an Internal cluster must meet both the, storage device requirements and have a storage class providing either,

- vSAN or VMFS datastore via the vsphere-volume provisioner
- VMDK, RDM, or DirectPath storage devices via the Local Storage Operator.

7.1.4. Microsoft Azure

Supports internal Red Hat OpenShift Data Foundation clusters only.

An internal cluster must meet both, storage device requirements and have a storage class that provides, an azure disk via the azure-disk provisioner.

7.1.5. Google Cloud

Supports internal Red Hat OpenShift Data Foundation clusters only.

An internal cluster must meet both, storage device requirements and have a storage class that provides, a GCE Persistent Disk via the gce-pd provisioner.

7.1.6. Red Hat OpenStack Platform [Technology Preview]

Supports internal Red Hat OpenShift Data Foundation clusters and consuming external clusters.

An internal cluster must meet both, storage device requirements and have a storage class that provides a standard disk via the Cinder provisioner.

7.1.7. IBM Power

Supports internal Red Hat OpenShift Data Foundation clusters and consuming external clusters.

An Internal cluster must meet both, storage device requirements and have a storage class providing local SSD (NVMe/SATA/SAS, SAN) via the Local Storage Operator.

7.1.8. IBM Z and IBM® LinuxONE

Supports internal Red Hat OpenShift Data Foundation clusters. Also, supports external mode where Red Hat Ceph Storage is running on x86.

An Internal cluster must meet both, storage device requirements and have a storage class providing local SSD (NVMe/SATA/SAS, SAN) via the Local Storage Operator.

7.1.9. ROSA with hosted control planes (HCP)

Supports internal Red Hat OpenShift Data Foundation clusters only.

An internal cluster must meet both, storage device requirements and have a storage class that provides AWS EBS volumes via **gp3-csi** provisioner.

7.1.10. Any platform

Supports internal clusters and consuming external clusters.

An internal cluster must meet both the storage device requirements and have a storage class that provide local SSD (NVMe/SATA/SAS, SAN) via the Local Storage Operator.

7.2. EXTERNAL MODE REQUIREMENT

7.2.1. Red Hat Ceph Storage

To check the supportability and interoperability of Red Hat Ceph Storage (RHCS) with Red Hat OpenShift Data Foundation in external mode, go to the lab Red Hat OpenShift Data Foundation Supportability and Interoperability Checker.

1. Select **Service Type** as **ODF as Self-Managed Service**.
2. Select appropriate **Version** from the drop down.
3. On the Versions tab, click the **Supported RHCS Compatibility** tab.

For instructions regarding how to install a RHCS cluster, see the installation guide.

7.3. RESOURCE REQUIREMENTS

Red Hat OpenShift Data Foundation services consist of an initial set of base services, and can be extended with additional device sets. All of these Red Hat OpenShift Data Foundation services pods are scheduled by kubernetes on OpenShift Container Platform nodes. Expanding the cluster in multiples of three, one node in each failure domain, is an easy way to satisfy the pod placement rules.

IMPORTANT

These requirements relate to OpenShift Data Foundation services only, and not to any other services, operators or workloads that are running on these nodes.

Table 7.1. Aggregate available resource requirements for Red Hat OpenShift Data Foundation only

Deployment Mode	Base services	Additional device Set
Internal	• 30 CPU (logical) • 72 GiB memory • 3 storage devices	• 6 CPU (logical) • 15 GiB memory • 3 storage devices

Deployment Mode	Base services	Additional device Set
External	• 4 CPU (logical) • 16 GiB memory	Not applicable

Example: For a 3 node cluster in an internal mode deployment with a single device set, a minimum of 3 x 10 = 30 units of CPU are required.

For more information, see Chapter 6, *Subscriptions* and CPU units.

For additional guidance with designing your Red Hat OpenShift Data Foundation cluster, see the ODF Sizing Tool.

CPU units

In this section, 1 CPU Unit maps to the Kubernetes concept of 1 CPU unit.

- 1 unit of CPU is equivalent to 1 core for non-hyperthreaded CPUs.
- 2 units of CPU are equivalent to 1 core for hyperthreaded CPUs.
- Red Hat OpenShift Data Foundation core-based subscriptions always come in pairs (2 cores).

Table 7.2. Aggregate minimum resource requirements for IBM Power

Deployment Mode	Base services
Internal	• 48 CPU (logical) • 192 GiB memory • 3 storage devices, each with additional 500GB of disk
External	• 24 CPU (logical) • 48 GiB memory

Example: For a 3 node cluster in an internal-attached devices mode deployment, a minimum of 3 x 16 = 48 units of CPU and 3 x 64 = 192 GB of memory is required.

7.3.1. Resource requirements for IBM Z and IBM LinuxONE infrastructure

Red Hat OpenShift Data Foundation services consist of an initial set of base services, and can be extended with additional device sets.

All of these Red Hat OpenShift Data Foundation services pods are scheduled by kubernetes on OpenShift Container Platform nodes . Expanding the cluster in multiples of three, one node in each failure domain, is an easy way to satisfy the pod placement rules.

Table 7.3. Aggregate available resource requirements for Red Hat OpenShift Data Foundation only (IBM Z and IBM® LinuxONE)

Deployment Mode	Base services	Additional device Set	IBM Z and IBM® LinuxONE minimum hardware requirements
Internal	• 30 CPU (logical) 　◦ 3 nodes with 10 CPUs (logical) each • 72 GiB memory • 3 storage devices	• 6 CPU (logical) • 15 GiB memory • 3 storage devices	1 IFL
External	• 4 CPU (logical) • 16 GiB memory	Not applicable	Not applicable

CPU

Is the number of virtual cores defined in the hypervisor, IBM Z/VM, Kernel Virtual Machine (KVM), or both.

IFL (Integrated Facility for Linux)

Is the physical core for IBM Z and IBM® LinuxONE.

Minimum system environment

- In order to operate a minimal cluster with 1 logical partition (LPAR), one additional IFL is required on top of the 6 IFLs. OpenShift Container Platform consumes these IFLs .

7.3.2. Minimum deployment resource requirements

An OpenShift Data Foundation cluster will be deployed with minimum configuration when the standard deployment resource requirement is not met.

IMPORTANT

These requirements relate to OpenShift Data Foundation services only, and not to any other services, operators or workloads that are running on these nodes.

Table 7.4. Aggregate resource requirements for OpenShift Data Foundation only

Deployment Mode	Base services
Internal	• 24 CPU (logical) • 72 GiB memory • 3 storage devices

If you want to add additional device sets, we recommend converting your minimum deployment to standard deployment.

7.3.3. Compact deployment resource requirements

Red Hat OpenShift Data Foundation can be installed on a three-node OpenShift compact bare metal cluster, where all the workloads run on three strong master nodes. There are no worker or storage nodes.

IMPORTANT

These requirements relate to OpenShift Data Foundation services only, and not to any other services, operators or workloads that are running on these nodes.

Table 7.5. Aggregate resource requirements for OpenShift Data Foundation only

Deployment Mode	Base services	Additional device Set
Internal	• 24 CPU (logical) • 72 GiB memory • 3 storage devices	• 6 CPU (logical) • 15 GiB memory • 3 storage devices

To configure OpenShift Container Platform on a compact bare metal cluster, see Configuring a three-node cluster and Delivering a Three-node Architecture for Edge Deployments.

7.3.4. Resource requirements for MCG only deployment

An OpenShift Data Foundation cluster deployed only with the Multicloud Object Gateway (MCG) component provides the flexibility in deployment and helps to reduce the resource consumption.

Table 7.6. Aggregate resource requirements for MCG only deployment

Deployment Mode	Core	Database (DB)	Endpoint

Deployment Mode	Core	Database (DB)	Endpoint
Internal	• 1 CPU • 4 GiB memory	• 0.5 CPU • 4 GiB memory	• 1 CPU • 2 GiB memory **NOTE** The default auto scale is between 1 - 2.

7.3.5. Resource requirements for using Network File system

You can create exports using Network File System (NFS) that can then be accessed externally from the OpenShift cluster. If you plan to use this feature, the NFS service consumes 3 CPUs and 8Gi of Ram. NFS is optional and is disabled by default.

The NFS volume can be accessed two ways:

- In-cluster: by an application pod inside of the Openshift cluster.
- Out of cluster: from outside of the Openshift cluster.

For more information about the NFS feature, see Creating exports using NFS

7.3.6. Resource requirements for performance profiles

OpenShift Data Foundation provides three performance profiles to enhance the performance of the clusters. You can choose one of these profiles based on your available resources and desired performance level during deployment or post deployment.

Table 7.7. Recommended resource requirement for different performance profiles

Performance profile	CPU	Memory
Lean	24	72 GiB
Balanced	30	72 GiB
Performance	45	96 GiB

IMPORTANT

Make sure to select the profiles based on the available free resources as you might already be running other workloads.

7.4. POD PLACEMENT RULES

Kubernetes is responsible for pod placement based on declarative placement rules. The Red Hat OpenShift Data Foundation base service placement rules for Internal cluster can be summarized as follows:

- Nodes are labeled with the **cluster.ocs.openshift.io/openshift-storage** key
- Nodes are sorted into pseudo failure domains if none exist
- Components requiring high availability are spread across failure domains
- A storage device must be accessible in each failure domain

This leads to the requirement that there be at least three nodes, and that nodes be in three distinct rack or zone failure domains in the case of pre-existing topology labels.

For additional device sets, there must be a storage device, and sufficient resources for the pod consuming it, in each of the three failure domains. Manual placement rules can be used to override default placement rules, but generally this approach is only suitable for bare metal deployments.

NOTE

Ensure that CPU and RAM resources remain available across all failure domains, even in the event of a failure. By default, the allocation is 6 CPUs and 15 GB of RAM. Any changes to this default configuration or the addition of backing stores must be considered in resource planning.

7.5. STORAGE DEVICE REQUIREMENTS

Use this section to understand the different storage capacity requirements that you can consider when planning internal mode deployments and upgrades. We generally recommend 12 devices or less per node. This recommendation ensures both that nodes stay below cloud provider dynamic storage device attachment limits, and to limit the recovery time after node failures with local storage devices. Expanding the cluster in multiples of three, one node in each failure domain, is an easy way to satisfy pod placement rules.

Storage nodes should have at least two disks, one for the operating system and the remaining disks for OpenShift Data Foundation components.

NOTE

You can expand the storage capacity only in the increment of the capacity selected at the time of installation.

7.5.1. Dynamic storage devices

Red Hat OpenShift Data Foundation permits the selection of either 0.5 TiB, 1 TiB, 2 TiB, 4 TiB, or 8 TiB capacities as the request size for dynamic storage device sizes. The number of dynamic storage devices that can run per node is a function of the node size, underlying provisioner limits and resource requirements.

7.5.2. Local storage devices

For local storage deployment, any disk size of 16 TiB or less can be used, and all disks should be of the same size and type. The number of local storage devices that can run per node is a function of the node size and resource requirements. Expanding the cluster in multiples of three, one node in each failure domain, is an easy way to satisfy pod placement rules.

NOTE

Disk partitioning is not supported, except for DASD-based deployment on IBM Z.

7.5.3. Capacity planning

Always ensure that available storage capacity stays ahead of consumption. Recovery is difficult if available storage capacity is completely exhausted, and requires more intervention than simply adding capacity or deleting or migrating content.

Capacity alerts are issued when cluster storage capacity reaches 75% (near-full) and 85% (full) of total capacity. Always address capacity warnings promptly, and review your storage regularly to ensure that you do not run out of storage space. When you get to 75% (near-full), either free up space or expand the cluster. When you get the 85% (full) alert, it indicates that you have run out of storage space completely and cannot free up space using standard commands. At this point, contact Red Hat Customer Support.

The following tables show example node configurations for Red Hat OpenShift Data Foundation with dynamic storage devices.

Table 7.8. Example initial configurations with 3 nodes

Storage Device size	Storage Devices per node	Total capacity	Usable storage capacity
0.5 TiB	1	1.5 TiB	0.5 TiB
2 TiB	1	6 TiB	2 TiB
4 TiB	1	12 TiB	4 TiB

Table 7.9. Example of expanded configurations with 30 nodes (N)

Storage Device size (D)	Storage Devices per node (M)	Total capacity (D * M * N)	Usable storage capacity (D*M*N/3)
0.5 TiB	3	45 TiB	15 TiB
2 TiB	6	360 TiB	120 TiB
4 TiB	9	1080 TiB	360 TiB

CHAPTER 8. NETWORK REQUIREMENTS

OpenShift Data Foundation requires that at least one network interface that is used for the cluster network to be capable of at least 10 gigabit network speeds. This section further covers different network considerations for planning deployments.

8.1. IPV6 SUPPORT

Red Hat OpenShift Data Foundation version 4.12 introduced the support of IPv6. IPv6 is supported in single stack only, and cannot be used simultaneously with IPv4. IPv6 is the default behavior in OpenShift Data Foundation when IPv6 is turned on in Openshift Container Platform.

Red Hat OpenShift Data Foundation version 4.14 introduces IPv6 auto detection and configuration. Clusters using IPv6 will automatically be configured accordingly.

OpenShift Container Platform dual stack with Red Hat OpenShift Data Foundation IPv4 is supported from version 4.13 and later. Dual stack on Red Hat OpenShift Data Foundation IPv6 is not supported.

8.2. MULTI NETWORK PLUG-IN (MULTUS) SUPPORT

OpenShift Data Foundation supports the ability to use multi-network plug-in Multus on bare metal infrastructures to improve security and performance by isolating the different types of network traffic. By using Multus, one or more network interfaces on hosts can be reserved for exclusive use of OpenShift Data Foundation.

To use Multus, first run the Multus prerequisite validation tool. For instructions to use the tool, see OpenShift Data Foundation - Multus prerequisite validation tool . For more information about Multus networks, see Multiple networks.

You can configure your Multus networks to use IPv4. You can also configure your network to use IPv6 is a technology preview. Multus networks can only be configured to use either IPv4 or IPv6. Mixing modes is not supported.

IMPORTANT

Technology Preview features provide early access to upcoming product innovations, enabling you to test functionality and provide feedback during the development process. However, these features are not fully supported under Red Hat Service Level Agreements, may not be functionally complete, and are not intended for production use. As Red Hat considers making future iterations of Technology Preview features generally available, we will attempt to resolve any issues that customers experience when using these features.

See Technology Preview Features Support Scope for more information.

8.2.1. Multus prerequisites

In order for Ceph-CSI to communicate with a Multus-enabled CephCluster, some setup is required for Kubernetes hosts.

These prerequisites require an understanding of how Multus networks are configured and how Rook uses them. This section will help clarify questions that could arise.

Two basic requirements must be met:

- OpenShift hosts must be able to route successfully to the Multus public network.
- Pods on the Multus public network must be able to route successfully to OpenShift hosts.

These two requirements can be broken down further as follows:

- For routing Kubernetes hosts to the Multus public network, each host must ensure the following:
    - The host must have an interface connected to the Multus public network (the "public-network-interface").
    - The "public-network-interface" must have an IP address.
    - A route must exist to direct traffic destined for pods on the Multus public network through the "public-network-interface".
- For routing pods on the Multus public network to Kubernetes hosts, the public NetworkAttachmentDefinition must be configured to ensure the following:
    - The definition must have its IP Address Management (IPAM) configured to route traffic destined for nodes through the network.
- To ensure routing between the two networks works properly, no IP address assigned to a node can overlap with any IP address assigned to a pod on the Multus public network.
- Generally, both the NetworkAttachmentDefinition, and node configurations must use the same network technology (Macvlan) to connect to the Multus public network.

Node configurations and pod configurations are interrelated and tightly coupled. Both must be planned at the same time, and OpenShift Data Foundation cannot support Multus public networks without both.

The “public-network-interface” must be the same for both. Generally, the connection technology (Macvlan) should also be the same for both. IP range(s) in the NetworkAttachmentDefinition must be encoded as routes on nodes, and, in mirror, IP ranges for nodes must be encoded as routes in the NetworkAttachmentDefinition.

Some installations might not want to use the same public network IP address range for both pods and nodes. In the case where there are different ranges for pods and nodes, additional steps must be taken to ensure each range routes to the other so that they act as a single, contiguous network.These requirements require careful planning. See Multus examples to help understand and implement these requirements.

TIP

There are often ten or more OpenShift Data Foundation pods per storage node. The pod address space usually needs to be several times larger (or more) than the host address space.

OpenShift Container Platform recommends using the NMState operator’s NodeNetworkConfigurationPolicies as a good method of configuring hosts to meet host requirements. Other methods can be used as well if needed.

8.2.1.1. Multus network address space sizing

Networks must have enough addresses to account for the number of storage pods that will attach to the network, plus some additional space to account for failover events.

It is highly recommended to also plan ahead for future storage cluster expansion and estimate how large the OpenShift Container Platform and OpenShift Data Foundation clusters may grow in the future. Reserving addresses for future expansion means that there is lower risk of depleting the IP address pool unexpectedly during expansion.

It is safest to allocate 25% more addresses (or more) than the total maximum number of addresses that are expected to be needed at one time in the storage cluster's lifetime. This helps lower the risk of depleting the IP address pool during failover and maintenance.

For ease of writing corresponding network CIDR configurations, rounding totals up to the nearest power of 2 is also recommended.

Three ranges must be planned:

- If used, the public Network Attachment Definition address space must include enough IPs for the total number of ODF pods running in the openshift-storage namespace
- If used, the cluster Network Attachment Definition address space must include enough IPs for the total number of OSD pods running in the openshift-storage namespace
- If the Multus public network is used, the node public network address space must include enough IPs for the total number of OpenShift nodes connected to the Multus public network.

NOTE

If the cluster uses a unified address space for the public Network Attachment Definition and node public network attachments, add these two requirements together. This is relevant, for example, if DHCP is used to manage IPs for the public network.

IMPORTANT

For users with environments with piecewise CIDRs, that is one network with two or more different CIDRs, auto-detection is likely to find only a single CIDR, meaning Ceph daemons may fail to start or fail to connect to the network.

8.2.1.1.1. Recommendation

The following recommendation suffices for most organizations. The recommendation uses the last 6.25% (1/16) of the reserved private address space (192.168.0.0/16), assuming the beginning of the range is in use or otherwise desirable. Approximate maximums (accounting for 25% overhead) are given.

Table 8.1. Multus recommendations

Network	Network range CIDR	Approximate maximums
Public Network Attachment Definition	192.168.240.0/21	1,600 total ODF pods
Cluster Network Attachment Definition	192.168.248.0/22	800 OSDs
Node public network attachments	192.168.252.0/23	400 total nodes

8.2.1.1.2. Calculation

More detailed address space sizes can be determined as follows:

1. Determine the maximum number of OSDs that are likely to be needed in the future. Add 25%, then add 5. Round the result up to the nearest power of 2. This is the cluster address space size.

2. Begin with the un-rounded number calculated in step 1. Add 64, then add 25%. Round the result up to the nearest power of 2. This is the public address space size for pods.

3. Determine the maximum number of total OpenShift nodes (including storage nodes) that are likely to be needed in the future. Add 25%. Round the result up to the nearest power of 2. This is the public address space size for nodes.

8.2.1.2. Verifying requirements have been met

After configuring nodes and creating the Multus public NetworkAttachmentDefinition (see Creating network attachment definitions) check that the node configurations and NetworkAttachmentDefinition configurations are compatible. To do so, verify that each node can **ping** pods via the public network.

Start a daemonset similar to the following example:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: multus-public-test
  namespace: openshift-storage
  labels:
    app: multus-public-test
spec:
 selector:
   matchLabels:
     app: multus-public-test
 template:
   metadata:
     labels:
       app: multus-public-test
     annotations:
       k8s.v1.cni.cncf.io/networks: openshift-storage/public-net #
   spec:
     containers:
       - name: test
         image: quay.io/ceph/ceph:v18 # image known to have ‘ping’ installed
         command:
           - sleep
           - infinity
         resources: {}
```

List the Multus public network IPs assigned to test pods using a command like the following example. This example command lists all IPs assigned to all test pods (each will have 2 IPs). From the output, it is easy to manually extract the IPs associated with the Multus public network.

```
$ oc -n openshift-storage describe pod -l app=multus-public-test | grep -o -E 'Add .* from .*'
Add eth0 [10.128.2.86/23] from ovn-kubernetes
Add net1 [192.168.20.22/24] from default/public-net
Add eth0 [10.129.2.173/23] from ovn-kubernetes
Add net1 [192.168.20.29/24] from default/public-net
Add eth0 [10.131.0.108/23] from ovn-kubernetes
Add net1 [192.168.20.23/24] from default/public-net
```

In the previous example, test pod IPs on the Multus public network are:

- 192.168.20.22
- 192.168.20.29
- 192.168.20.23

Check that each node (NODE) can reach all test pod IPs over the public network:

```
$ oc debug node/NODE
Starting pod/NODE-debug ...
To use host binaries, run `chroot /host`
Pod IP: ****
If you don't see a command prompt, try pressing enter.


sh-5.1# chroot /host


sh-5.1# ping 192.168.20.22
PING 192.168.20.22 (192.168.20.22) 56(84) bytes of data.
64 bytes from 192.168.20.22: icmp_seq=1 ttl=64 time=0.093 ms
64 bytes from 192.168.20.22: icmp_seq=2 ttl=64 time=0.056 ms
^C
--- 192.168.20.22 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1046ms
rtt min/avg/max/mdev = 0.056/0.074/0.093/0.018 ms


sh-5.1# ping 192.168.20.29
PING 192.168.20.29 (192.168.20.29) 56(84) bytes of data.
64 bytes from 192.168.20.29: icmp_seq=1 ttl=64 time=0.403 ms
64 bytes from 192.168.20.29: icmp_seq=2 ttl=64 time=0.181 ms
^C
--- 192.168.20.29 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1007ms
rtt min/avg/max/mdev = 0.181/0.292/0.403/0.111 ms


sh-5.1# ping 192.168.20.23
PING 192.168.20.23 (192.168.20.23) 56(84) bytes of data.
64 bytes from 192.168.20.23: icmp_seq=1 ttl=64 time=0.329 ms
64 bytes from 192.168.20.23: icmp_seq=2 ttl=64 time=0.227 ms
^C
--- 192.168.20.23 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1047ms
rtt min/avg/max/mdev = 0.227/0.278/0.329/0.051 ms
```

If any node does not get a successful ping to a running pod, it is not safe to proceed. Diagnose and fix the issue, then repeat this testing. Some reasons you may encounter a problem include:

- The host may not be properly attached to the Multus public network (via Macvlan)
- The host may not be properly configured to route to the pod IP range
- The public NetworkAttachmentDefinition may not be properly configured to route back to the host IP range
- The host may have a firewall rule blocking the connection in either direction
- The network switch may have a firewall or security rule blocking the connection

Suggested debugging steps:

- Ensure nodes can ping each other over using public network “shim” IPs
- Ensure the output of **ip address**

8.2.2. Multus examples

The relevant network plan for this cluster is as follows:

- A dedicated NIC provides eth0 for the Multus public network
- Macvlan will be used to attach OpenShift pods to eth0
- The IP range 192.168.0.0/16 is free in the example cluster – pods and nodes will share this IP range on the Multus public network
- Nodes will get the IP range 192.168.252.0/22 (this allows up to 1024 Kubernetes hosts, more than the example organization will ever need)
- Pods will get the remainder of the ranges (192.168.0.1 to 192.168.251.255)
- The example organization does not want to use DHCP unless necessary; therefore, nodes will have IPs on the Multus network (via eth0) assigned statically using the NMState operator’s NodeNetworkConfigurationPolicy resources
- With DHCP unavailable, Whereabouts will be used to assign IPs to the Multus public network because it is easy to use out of the box
- There are 3 compute nodes in the OpenShift cluster on which OpenShift Data Foundation also runs: compute-0, compute-1, and compute-2

Nodes’ network policies must be configured to route to pods on the Multus public network.

Because pods will be connecting via Macvlan, and because Macvlan does not allow hosts and pods to route between each other, the host must also be connected via Macvlan. Generally speaking, the host must connect to the Multus public network using the same technology that pods do. Pod connections are configured in the Network Attachment Definition.

Because the host IP range is a subset of the whole range, hosts are not able to route to pods simply by IP assignment. A route must be added to hosts to allow them to route to the whole 192.168.0.0/16 range.

NodeNetworkConfigurationPolicy **desiredState** specs will look like the following:

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
```

```
metadata:
  name: ceph-public-net-shim-compute-0
  namespace: openshift-storage
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: ""
    kubernetes.io/hostname: compute-0
 desiredState:
   interfaces:
     - name: odf-pub-shim
       description: Shim interface used to connect host to OpenShift Data Foundation public Multus
network
       type: mac-vlan
       state: up
       mac-vlan:
         base-iface: eth0
         mode: bridge
         promiscuous: true
       ipv4:
         enabled: true
         dhcp: false
         address:
           - ip: 192.168.252.1 # STATIC IP FOR compute-0
             prefix-length: 22
   routes:
     config:
       - destination: 192.168.0.0/16
         next-hop-interface: odf-pub-shim
---
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: ceph-public-net-shim-compute-1
  namespace: openshift-storage
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: ""
    kubernetes.io/hostname: compute-1
 desiredState:
   interfaces:
     - name: odf-pub-shim
       description: Shim interface used to connect host to OpenShift Data Foundation public Multus
network
       type: mac-vlan
       state: up
       mac-vlan:
         base-iface: eth0
         mode: bridge
         promiscuous: true
       ipv4:
         enabled: true
         dhcp: false
         address:
           - ip: 192.168.252.1 # STATIC IP FOR compute-1
             prefix-length: 22
   routes:
    config:
      - destination: 192.168.0.0/16
        next-hop-interface: odf-pub-shim
---
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: ceph-public-net-shim-compute-2 # [1]
  namespace: openshift-storage
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: ""
    kubernetes.io/hostname: compute-2 # [2]
 desiredState:
   interfaces: [3]
     - name: odf-pub-shim
       description: Shim interface used to connect host to OpenShift Data Foundation public Multus
network
       type: mac-vlan # [4]
       state: up
       mac-vlan:
         base-iface: eth0 # [5]
         mode: bridge
         promiscuous: true
       ipv4: # [6]
         enabled: true
         dhcp: false
         address:
           - ip: 192.168.252.2 # STATIC IP FOR compute-2 # [7]
             prefix-length: 22
   routes: # [8]
     config:
       - destination: 192.168.0.0/16 # [9]
         next-hop-interface: odf-pub-shim
```

1. For static IP management, each node must have a different NodeNetworkConfigurationPolicy.
2. Select separate nodes for each policy to configure static networks.
3. A “shim” interface is used to connect hosts to the Multus public network using the same technology as the Network Attachment Definition will use.
4. The host’s “shim” must be of the same type as planned for pods, **macvlan** in this example.
5. The interface must match the Multus public network interface selected in planning, **eth0** in this example.
6. The **ipv4** (or **ipv6**) section configures node IP addresses on the Multus public network.
7. IPs assigned to this node’s shim must match the plan. This example uses 192.168.252.0/22 for node IPs on the Multus public network.
8. For static IP management, do not forget to change the IP for each node.
9. The **routes** section instructs nodes how to reach pods on the Multus public network.

10. The route destination(s) must match the CIDR range planned for pods. In this case, it is safe to use the entire 192.168.0.0/16 range because it won't affect nodes' ability to reach other nodes over their "shim" interfaces. In general, this must match the CIDR used in the Multus public NetworkAttachmentDefinition.

The NetworkAttachmentDefinition for the public network would look like the following, using Whereabouts' **exclude** option to simplify the **range** request. The Whereabouts **routes[].dst** option ensures pods route to hosts via the Multus public network.

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: public-net
  namespace: openshift-storage
spec:
  config: '{
     "cniVersion": "0.3.1",
     "type": "macvlan", # [1]
     "master": "eth0", # [2]
     "mode": "bridge",
     "ipam": {
       "type": "whereabouts", # [3]
       "range": "192.168.0.0/16", # [4]
       "exclude": [
          "192.168.252.0/22" # [5]
       ],
       "routes": [              # [6]
         {"dst": "192.168.252.0/22"} # [7]
       ]
     }
   }'
```

1. This must match the plan for how to attach pods to the Multus public network. Nodes must attach using the same technology, Macvlan.

2. The interface must match the Multus public network interface selected in planning, **eth0** in this example.

3. The plan for this example uses whereabouts instead of DHCP for assigning IPs to pods.

4. For this example, it was decided that pods could be assigned any IP in the range 192.168.0.0/16 with the exception of a portion of the range allocated to nodes (see 5).

5. **whereabouts** provides an **exclude** directive that allows easily excluding the range allocated for nodes from its pool. This allows keeping the **range** directive (see 4) simple.

6. The **routes** section instructs pods how to reach nodes on the Multus public network.

7. The route destination (**dst**) must match the CIDR range planned for nodes.

8.2.3. Segregating storage traffic using Multus

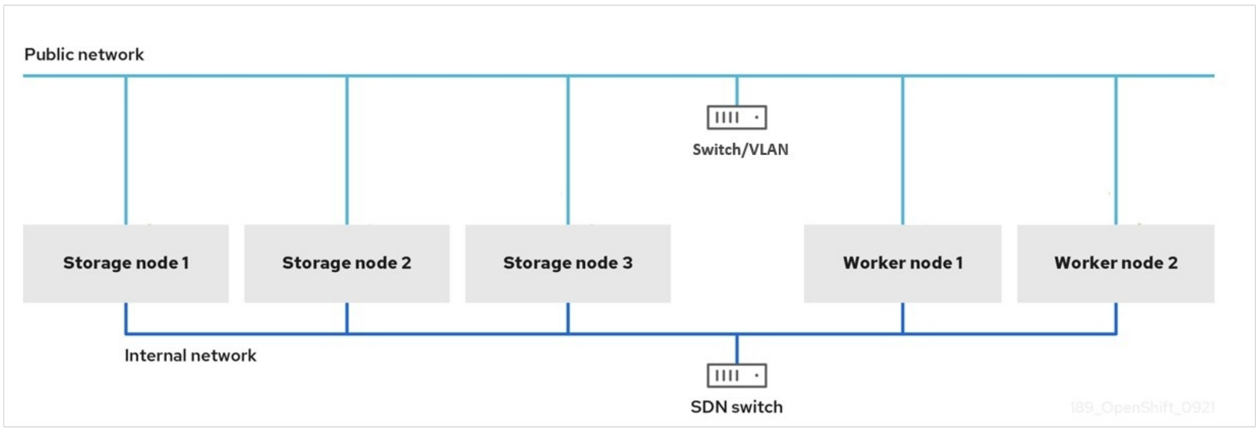
By default, Red Hat OpenShift Data Foundation is configured to use the Red Hat OpenShift Software Defined Network (SDN). The default SDN carries the following types of traffic:

- Pod-to-pod traffic
- Pod-to-storage traffic, known as public network traffic when the storage is OpenShift Data Foundation
- OpenShift Data Foundation internal replication and rebalancing traffic, known as cluster network traffic

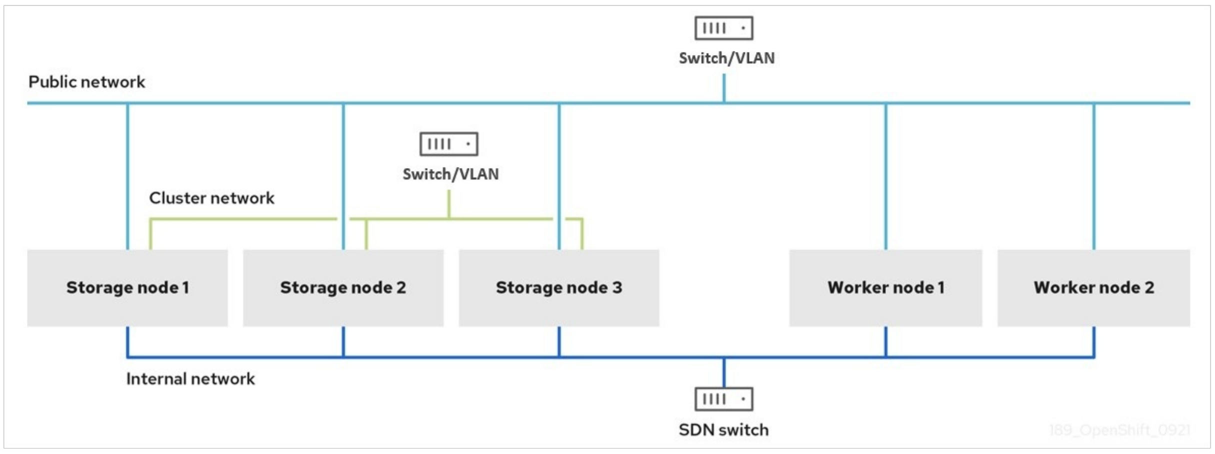
There are three ways to segregate OpenShift Data Foundation from OpenShift default network:

1. Reserve a network interface on the host for the public network of OpenShift Data Foundation
    - Pod-to-storage and internal storage replication traffic coexist on a network that is isolated from pod-to-pod network traffic.
    - Application pods have access to the maximum public network storage bandwidth when the OpenShift Data Foundation cluster is healthy.
    - When the OpenShift Data Foundation cluster is recovering from failure, the application pods will have reduced bandwidth due to ongoing replication and rebalancing traffic.
2. Reserve a network interface on the host for OpenShift Data Foundation’s cluster network
    - Pod-to-pod and pod-to-storage traffic both continue to use OpenShift’s default network.
    - Pod-to-storage bandwidth is less affected by the health of the OpenShift Data Foundation cluster.
    - Pod-to-pod and pod-to-storage OpenShift Data Foundation traffic might contend for network bandwidth in busy OpenShift clusters.
    - The storage internal network often has an overabundance of bandwidth that is unused, reserved for use during failures.
3. Reserve two network interfaces on the host for OpenShift Data Foundation: one for the public network and one for the cluster network
    - Pod-to-pod, pod-to-storage, and storage internal traffic are all isolated, and none of the traffic types will contend for resources.
    - Service level agreements for all traffic types are more able to be ensured.
    - During healthy runtime, more network bandwidth is reserved but unused across all three networks.

Dual network interface segregated configuration schematic example:



Triple network interface full segregated configuration schematic example:



8.2.4. When to use Multus

Use Multus for OpenShift Data Foundation when you need the following:

Improved latency - Multus with ODF always improves latency. Use host interfaces at near-host network speeds and bypass OpenShift's software-defined Pod network. You can also perform Linux per interface level tuning for each interface.

Improved bandwidth - Dedicated interfaces for OpenShift Data Foundation client data traffic and internal data traffic. These dedicated interfaces reserve full bandwidth.

Improved security - Multus isolates storage network traffic from application network traffic for added security. Bandwidth or performance might not be isolated when networks share an interface, however, you can use QoS or traffic shaping to prioritize bandwidth on shared interfaces.

8.2.5. Multus configuration

To use Multus, you must create network attachment definitions (NADs) before deploying the OpenShift Data Foundation cluster, which is later attached to the cluster. For more information, see Creating network attachment definitions.

To attach additional network interfaces to a pod, you must create configurations that define how the interfaces are attached. You specify each interface by using a **NetworkAttachmentDefinition** custom resource (CR). A Container Network Interface (CNI) configuration inside each of these CRs defines how that interface is created.

OpenShift Data Foundation supports the **macvlan** driver, which includes the following features:

- Each connection gets a sub-interface of the parent interface with its own MAC address and is isolated from the host network.
- Uses less CPU and provides better throughput than Linux bridge or **ipvlan**.
- Bridge mode is almost always the best choice.
- Near-host performance when network interface card (NIC) supports virtual ports/virtual local area networks (VLANs) in hardware.

OpenShift Data Foundation supports the following two types IP address management:

whereabouts	DHCP
Uses OpenShift/Kubernetes **leases** to select unique IP addresses per Pod.	Does not require **range** field.
Does not require a DHCP server to provide IPs for Pods.	Network DHCP server can give out the same range to Multus Pods as well as any other hosts on the same network.

CAUTION

If there is a DHCP server, ensure Multus configured IPAM does not give out the same range so that multiple MAC addresses on the network cannot have the same IP.

8.2.6. Requirements for Multus configuration

Prerequisites

- The interface used for the public network must have the same interface name on each OpenShift storage and worker node, and the interfaces must all be connected to the same underlying network.

- The interface used for the cluster network must have the same interface name on each OpenShift storage node, and the interfaces must all be connected to the same underlying network. Cluster network interfaces do not have to be present on the OpenShift worker nodes.
- Each network interface used for the public or cluster network must be capable of at least 10 gigabit network speeds.
- Each network requires a separate virtual local area network (VLAN) or subnet.

See Creating Multus networks for the necessary steps to configure a Multus based configuration on bare metal.

CHAPTER 9. DISASTER RECOVERY

Disaster Recovery (DR) helps an organization to recover and resume business critical functions or normal operations when there are disruptions or disasters. OpenShift Data Foundation provides High Availability (HA) & DR solutions for stateful apps which are broadly categorized into two broad categories:

- **Metro-DR**: Single Region and cross data center protection with no data loss.
- **Regional-DR**: Cross Region protection with minimal potential data loss.
- **Disaster Recovery with stretch cluster**: Single OpenShift Data Foundation cluster is stretched between two different locations to provide the storage infrastructure with disaster recovery capabilities.

9.1. METRO-DR

Metropolitan disaster recovery (Metro-DR) is composed of Red Hat Advanced Cluster Management for Kubernetes (RHACM), Red Hat Ceph Storage and OpenShift Data Foundation components to provide application and data mobility across OpenShift Container Platform clusters.

This release of Metro-DR solution provides volume persistent data and metadata replication across sites that are geographically dispersed. In the public cloud these would be similar to protecting from an Availability Zone failure. Metro-DR ensures business continuity during the unavailability of a data center with no data loss. This solution is entitled with Red Hat Advanced Cluster Management (RHACM) and OpenShift Data Foundation Advanced SKUs and related bundles.

IMPORTANT

You can now easily set up Metropolitan disaster recovery solutions for workloads based on OpenShift virtualization technology using OpenShift Data Foundation. For more information, see the knowledgebase article.

Prerequisites

- Disaster Recovery features supported by Red Hat OpenShift Data Foundation require all of the following prerequisites in order to successfully implement a Disaster Recovery solution:
  - A valid Red Hat OpenShift Data Foundation Advanced entitlement
  - A valid Red Hat Advanced Cluster Management for Kubernetes subscription

  To know how subscriptions for OpenShift Data Foundation work, see knowledgebase article on OpenShift Data Foundation subscriptions.
- Ensure that the primary managed cluster (Site-1) is co-situated with the active RHACM hub cluster while the passive hub cluster is situated along with the secondary managed cluster (Site-2). Alternatively, the active RHACM hub cluster can be placed in a neutral site (site-3) that is not impacted by the failures of either of the primary managed cluster at Site-1 or the secondary cluster at Site-2. In this situation, if a passive hub cluster is used it can be placed with the secondary cluster at Site-2.

NOTE

Hub recovery for Metro-DR is a Technology Preview feature and is subject to Technology Preview support limitations.

For detailed solution requirements, see Metro-DR requirements, deployment requirements for Red Hat Ceph Storage stretch cluster with arbiter and RHACM requirements.

9.2. REGIONAL-DR

Regional disaster recovery (Regional-DR) is composed of Red Hat Advanced Cluster Management for Kubernetes (RHACM) and OpenShift Data Foundation components to provide application and data mobility across OpenShift Container Platform clusters. It is built on Asynchronous data replication and hence could have a potential data loss but provides the protection against a broad set of failures.

Red Hat OpenShift Data Foundation is backed by Ceph as the storage provider, whose lifecycle is managed by Rook and it's enhanced with the ability to:

- Enable pools for mirroring.
- Automatically mirror images across RBD pools.
- Provides csi-addons to manage per Persistent Volume Claim mirroring.

This release of Regional-DR supports Multi-Cluster configuration that is deployed across different regions and data centers. For example, a 2-way replication across two managed clusters located in two different regions or data centers. This solution is entitled with Red Hat Advanced Cluster Management (RHACM) and OpenShift Data Foundation Advanced SKUs and related bundles.

IMPORTANT

You can now easily set up Regional disaster recovery solutions for workloads based on OpenShift virtualization technology using OpenShift Data Foundation. For more information, see the knowledgebase article.

Prerequisites

- Disaster Recovery features supported by Red Hat OpenShift Data Foundation require all of the following prerequisites in order to successfully implement a Disaster Recovery solution:
  - A valid Red Hat OpenShift Data Foundation Advanced entitlement
  - A valid Red Hat Advanced Cluster Management for Kubernetes subscription

  To know how subscriptions for OpenShift Data Foundation work, see knowledgebase article on OpenShift Data Foundation subscriptions.
- Ensure that the primary managed cluster (Site-1) is co-situated with the active RHACM hub cluster while the passive hub cluster is situated along with the secondary managed cluster (Site-2). Alternatively, the active RHACM hub cluster can be placed in a neutral site (site-3) that is not impacted by the failures of either of the primary managed cluster at Site-1 or the secondary cluster at Site-2. In this situation, if a passive hub cluster is used it can be placed with the secondary cluster at Site-2.

For detailed solution requirements, see Regional-DR requirements and RHACM requirements.

9.3. DISASTER RECOVERY WITH STRETCH CLUSTER

In this case, a single cluster is stretched across two zones with a third zone as the location for the arbiter. This feature is currently intended for deployment in the OpenShift Container Platform on-premises and in the same location. This solution is not recommended for deployments stretching over multiple data centers. Instead, consider Metro-DR as a first option for no data loss DR solution deployed over multiple data centers with low latency networks.

NOTE

The stretch cluster solution is designed for deployments where latencies do not exceed 10 ms maximum round-trip time (RTT) between the zones containing data volumes. For Arbiter nodes follow the latency requirements specified for etcd, see Guidance for Red Hat OpenShift Container Platform Clusters - Deployments Spanning Multiple Sites(Data Centers/Regions). Contact Red Hat Customer Support if you are planning to deploy with higher latencies.

To use the stretch cluster,

- You must have a minimum of five nodes across three zones, where:
  - Two nodes per zone are used for each data-center zone, and one additional zone with one node is used for arbiter zone (the arbiter can be on a master node).
- All the nodes must be manually labeled with the zone labels prior to cluster creation.
  For example, the zones can be labeled as:
  - **topology.kubernetes.io/zone=arbiter** (master or worker node)
  - **topology.kubernetes.io/zone=datacenter1** (minimum two worker nodes)
  - **topology.kubernetes.io/zone=datacenter2** (minimum two worker nodes)

For more information, see Configuring OpenShift Data Foundation for stretch cluster .

To know how subscriptions for OpenShift Data Foundation work, see knowledgebase article on OpenShift Data Foundation subscriptions.

IMPORTANT

You can now easily set up disaster recovery with stretch cluster for workloads based on OpenShift virtualization technology using OpenShift Data Foundation. For more information, see OpenShift Virtualization in OpenShift Container Platform guide.

CHAPTER 10. DISCONNECTED ENVIRONMENT

Disconnected environment is a network restricted environment where the Operator Lifecycle Manager (OLM) cannot access the default Operator Hub and image registries, which require internet connectivity.

Red Hat supports deployment of OpenShift Data Foundation in disconnected environments where you have installed OpenShift Container Platform in restricted networks.

To install OpenShift Data Foundation in a disconnected environment, see *Using Operator Lifecycle Manager in disconnected environments* of the Operators guide in OpenShift Container Platform documentation.

NOTE

When you install OpenShift Data Foundation in a restricted network environment, apply a custom Network Time Protocol (NTP) configuration to the nodes, because by default, internet connectivity is assumed in OpenShift Container Platform and **chronyd** is configured to use the ***.rhel.pool.ntp.org** servers.

For more information, see the Red Hat Knowledgebase solution A newly deployed OCS 4 cluster status shows as "Degraded", Why? and *Configuring chrony time service* of the Installing guide in OpenShift Container Platform documentation.

The Agent-based Installer allows you to use a mirror registry for disconnected installations. For more information, see Preparing to install with Agent-based Installer .

Packages to include for OpenShift Data Foundation

When you prune the **redhat-operator** index image, include the following operator bundles for the OpenShift Data Foundation deployment:

- **ocs-operator**
- **odf-operator**
- **mcg-operator**
- **odf-csi-addons-operator**
- **ocs-client-operator**
- **odf-prometheus-operator**
- **recipe**
- **rook-ceph-operator**
- **cephcsi-operator**
- **odf-dependencies**

Only for local storage deployments:

- **local-storage-operator**

Only for Regional Disaster Recovery (Regional-DR) configuration or Metro Disaster Recovery (Metro-DR) configuration:

Only for Regional Disaster Recovery (Regional-DR) or Metro Disaster Recovery (Metro-DR) configuration:

- **odf-multicluster-orchestrator**
- **odr-cluster-operator**
- **odr-hub-operator**

IMPORTANT

Make sure to name the **CatalogSource** as **redhat-operators**.

Upgrade requirements

When mirroring OpenShift Data Foundation for y-stream upgrades in a disconnected environment, ensure the following:

- Include all the OpenShift Data Foundation operator bundles in the mirror configuration file.
- Package both operator versions, the currently installed version and the target upgrade version within the same operator catalog index image.
  For example:

```
- name: odf-operator
  channels:
  - name: stable-<target-odf-version>
    minVersion : <target-upgrade-version>-rhodf
    maxVersion : <target-upgrade-version>-rhodf
  - name: stable-<current-odf-version>
    minVersion : <current-version>-rhodf
    maxVersion : <current-version>-rhodf
- name: ocs-operator
  channels:
[...]
```

CHAPTER 11. PERFORMANCE CONSIDERATIONS AND BENCHMARKING

To accurately predict performance and identify potential bottlenecks in your OpenShift Data Foundation cluster, benchmarking is necessary. This knowledgebase article explains the tools and procedures for performing these benchmarks and analyzing the resulting data.

CHAPTER 12. SUPPORTED AND UNSUPPORTED FEATURES FOR IBM POWER AND IBM Z

Table 12.1. List of supported and unsupported features on IBM Power and IBM Z

Features	IBM Power	IBM Z
Compact deployment	Unsupported	Unsupported
Dynamic storage devices	Unsupported	Supported
Federal Information Processing Standard Publication (FIPS)	Unsupported	Unsupported
Alerts to control overprovision	Supported	Unsupported
Alerts when Ceph Monitor runs out of space	Supported	Unsupported
Extended OpenShift Data Foundation control plane, which allows pluggable external storage such as IBM FlashSystem	Unsupported	Unsupported
IPv6 support	Unsupported	Unsupported
Multus	Unsupported	Supported
Minimum deployment	Unsupported	Unsupported
Agnostic deployment of OpenShift Data Foundation on any Openshift supported platform	Unsupported	Unsupported
Installer-provisioned deployment of OpenShift Data Foundation using bare metal infrastructure	Unsupported	Unsupported
OpenShift dual stack with OpenShift Data Foundation using IPv4	Unsupported	Unsupported
Ability to disable Multicloud Object Gateway external service during deployment	Unsupported	Unsupported
RHACM KubeVirt DR integration (Virtualization required)	Unsupported	Unsupported

CHAPTER 13. NEXT STEPS

To start deploying your OpenShift Data Foundation, you can use the internal mode within OpenShift Container Platform or use external mode to make available services from a cluster running outside of OpenShift Container Platform.

Depending on your requirement, go to the respective deployment guides.

Internal mode

- Deploying OpenShift Data Foundation using Amazon web services
- Deploying OpenShift Data Foundation using Bare Metal
- Deploying OpenShift Data Foundation using VMWare vSphere
- Deploying OpenShift Data Foundation using Microsoft Azure
- Deploying OpenShift Data Foundation using Google Cloud
- Deploying OpenShift Data Foundation using Red Hat OpenStack Platform [Technology Preview]
- Deploying OpenShift Data Foundation on IBM Power
- Deploying OpenShift Data Foundation on IBM Z
- Deploying OpenShift Data Foundation on any platform

External mode

- Deploying OpenShift Data Foundation in external mode

Internal or external

For deploying multiple clusters, see Deploying multiple OpenShift Data Foundation clusters .