Veritas InfoScale™ 8.0 Support for Containers - Linux

VERITAS™

Last updated: 2021-12-21

Legal Notice

Veritas Technologies LLC
2625 Augustine Drive
Santa Clara, CA 95054
http://www.veritas.com

Technical Support

Technical Support maintains support centers globally. All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policies. For information about our support offerings and how to contact Technical Support, visit our website:
https://www.veritas.com/support

You can manage your Veritas account information at the following URL:
https://my.veritas.com

If you have questions regarding an existing support agreement, please email the support agreement administration team for your region as follows:

Worldwide (except Japan)	CustomerCare@veritas.com
Japan	CustomerCare_Japan@veritas.com

Documentation

Make sure that you have the current version of the documentation. Each document displays the date of the last update on page 2. The latest documentation is available on the Veritas website:
https://sort.veritas.com/documents

Documentation feedback

Your feedback is important to us. Suggest improvements or report errors or omissions to the documentation. Include the document title, document version, chapter title, and section title of the text on which you are reporting. Send feedback to:
infoscaledocs@veritas.com

You can also see documentation information or ask a question on the Veritas community site:
http://www.veritas.com/community/

Veritas Services and Operations Readiness Tools (SORT)

Veritas Services and Operations Readiness Tools (SORT) is a website that provides information and tools to automate and simplify certain time-consuming administrative tasks. Depending on the product, SORT helps you prepare for installations and upgrades, identify risks in your datacenters, and improve operational efficiency. To see what services and tools SORT provides for your product, see the data sheet:
https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

Contents

Chapter 1

Overview

This chapter includes the following topics:

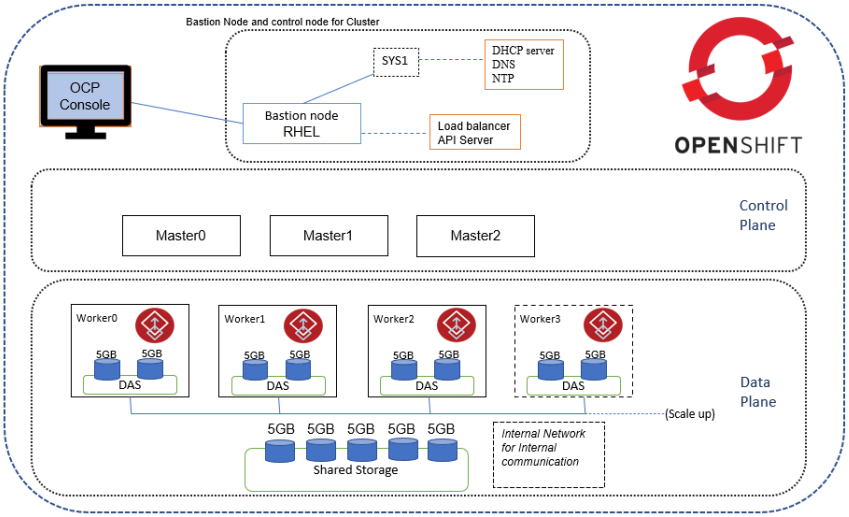
Introduction

With organizations increasingly adopting Container environments, it is necessary that all applications and storage must be available on these environments.

InfoScale provides high-performance shared storage for the OpenShift or Kubernetes clusters by using the fast storage, directly attached to the cluster nodes. InfoScale Storage provides highly available persistent storage that conforms to CSI specifications for enterprise applications by using high-performance parallel storage access on shared storage (SAN) or in Flexible Storage Sharing (FSS) environments.

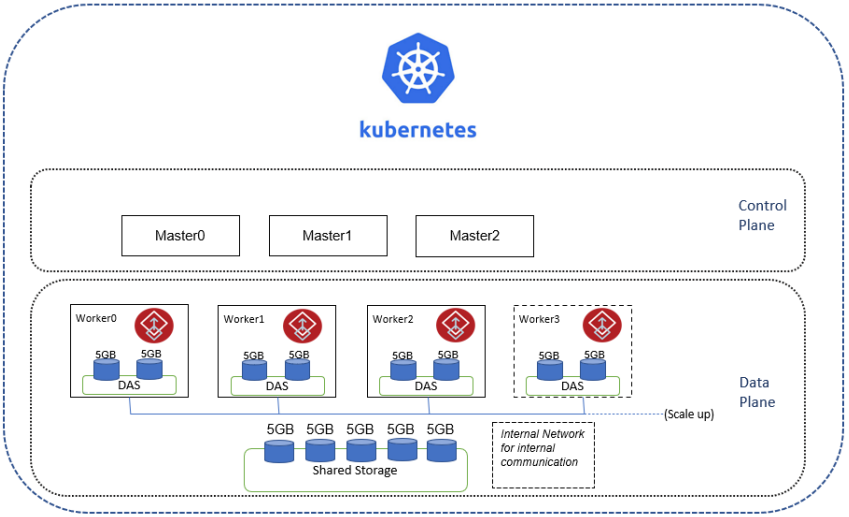
For OpenShift, you can download files from the Red Hat registry and deploy InfoScale. With an active Red Hat account, you can access the InfoScale images. Download from a single source with a single sign-on ensures a high level of security. An example of an OpenShift cluster is as under

Figure 1-1 OpenShift cluster comprising three master nodes, four worker nodes, and one bastion node. Storage can be SAN or DAS.



For Kubernetes, you can download files from the Veritas Download Center and deploy InfoScale. An example of a Kubernetes cluster is as under

Figure 1-2 Kubernetes cluster comprising three master and four worker nodes. Storage can be SAN or DAS.



Features of InfoScale in Containerized environment

- Support for Direct-Attached Storage(DAS) and Storage Area Network(SAN) in a multi-vendor environment.
- High-performance parallel storage that provides better performance and reliability than NFS.
- Optimized resource utilization with the ability to use either existing SAN storage or the InfoScale advanced FSS option that provides better performance than SAN at a reduced cost.
- Support for Dynamic Multipathing (DMP).

Besides the traditional features listed above, new features are -

- Integrated I/O fencing and arbitration to protect against data corruption and to provide fast recovery in the event of a failure.
- Snapshot copies of the production data for analytics and disaster recovery.
- Volume cloning to address storage disk and node failures.

- Support for stateful applications like MySQL, Oracle, PostgreSQL.

CSI Introduction

CSI is a standardized mechanism for Container Orchestrators (COs) to expose arbitrary storage systems to their containerized workloads. The InfoScale CSI plugin is used to provide persistent InfoScale Storage to container workloads or applications. The InfoScale CSI plugin supports creation of storage classes for high availability, performance, and capacity. Online expansion of capacity as well as the usage of snapshots and clones is also supported.

I/O fencing

InfoScale uses majority-based I/O fencing to guarantee data protection and provide persistent storage in the Container environment. Fencing ensures that data protection gets highest priority and stops running the systems when a split-brain condition is encountered. The systems thus cannot start services and data is protected. InfoScale checks for the connectivity with each peer nodes periodically while OpenShift or Kubernetes check it for the master to worker nodes.

The OpenShift or Kubernetes cluster performs failover or restart of applications for the nodes that have reached a `NotReady` state in the OpenShift or Kubernetes cluster. If an application is configured as a statefulset pod, the Container stops the failover of such application pods till the node becomes active again. In such scenarios, InfoScale uses the fencing module to ensure that the application pods running on such unreachable nodes cannot access the persistent storage so that OpenShift or Kubernetes can restart these pods on the active cluster without the risk of any data corruption.

When InfoScale cluster is deployed on an OpenShift or Kubernetes, InfoScale uses a custom fencing controller to provide the fencing infrastructure. The custom controller interacts with the InfoScale fencing driver and enables failover in OpenShift or Kubernetes in case of network split. An agent running on the controller ensures that InfoScale fences out the persistent storage and performs the pod failover for the fenced-out node. It also ensures that the fencing decisions of InfoScale I/O fencing do not conflict with the fencing decisions of the fencing controller.

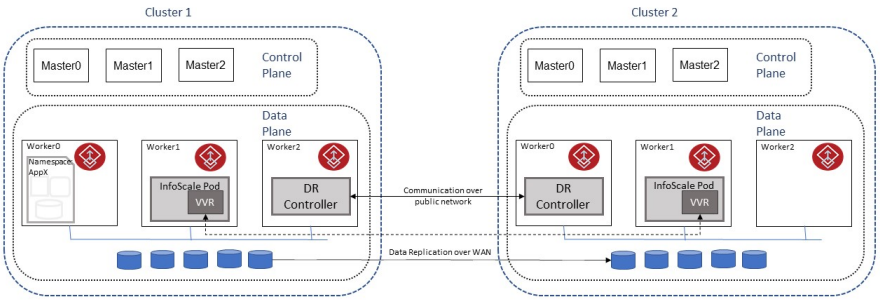
For deployment in containerized environments, when you install InfoScale by using the product installer, the fencing module is automatically installed and configured in majority mode. In case of network split, the I/O fencing module takes fencing decisions based on the number of nodes in a sub-cluster.

The hostnames of InfoScale nodes must exactly match the FQDN of the OpenShift or Kubernetes nodes for a successful configuration.

TECHNOLOGY PREVIEW: Disaster Recovery

Disaster Recovery(DR) is provided to applications hosted in container ecosystems. Native container HA capabilities provide high availability to application components within a cluster. However, DR functionality provides disaster recovery in the event of a cluster failure and application components can migrate to another peer cluster in membership. You can form a logical notion called 'Global Cluster' comprising clusters that can be used to migrate DR-enabled objects. DR-enabled objects migrate to peer cluster in case of a disaster like entire cluster going down, loss of connectivity with a particular cluster, user-initiated planned migration across cluster(s). Peer-to-peer communication between DR controllers is encrypted by using a self-signed certificate. These self-signed certificates are auto-generated while configuring DR.

Figure 1-3 DR Configuration comprising two clusters



You can configure a Disaster Recovery Plan(DR Plan) for a given namespace. For a more granular control, you can specify labels along with the namespace. In DR plan, you also specify the primary cluster and a DR cluster. Workload is shifted to the DR cluster if the primary cluster fails. For maintenance activities, you can also initiate a graceful migration of DR plan across peer cluster. Application instances are migrated along with associated persistent data(in case of stateful application). For replicating persistent data across peer cluster, it uses Veritas Volume Replicator(VVR).

Chapter 2

System requirements

This chapter includes the following topics:

Introduction

The System Requirements to run InfoScale in Containerized environment are listed here. The supported container platforms are also listed.

This document is intended for the use of a Storage Administrator who knows how to install, configure, and administer Applications in a Linux environment. Additionally, it is assumed that the user knows Container-related concepts like nodes, pods, and types of nodes and commands to access nodes. To know more about these concepts and commands, OpenShift and Kubernetes documentation can be referred.

Supported platforms

The following table lists the supported container platforms.

Table 2-1 Supported Container platforms

Platform / Configuration	Version
OpenShift Container Platform (OCP) version	4.9.6
Kubernetes version	v1.20.11 on OEL 8.4 1.21.3, 1.21.5 on SLES15 SP2

The following table lists the kernel versions per Operating System

Table 2-2 Supported Operating Systems and kernel versions

Operating system	Major kernel version	Minor kernel versions
RHEL Core OS	Core OS 8.4	4.18.0-305 4.18.0-305.3.1 4.18.0-305.7.1 4.18.0-305.10.2 4.18.0-305.12.1 4.18.0-305.17.1 4.18.0-305.19.1 4.18.0-305.25.1
OL8	OL 8.4	4.18.0-305 4.18.0-305.10.2
SLES15	SLES 15.2	5.3.18-22-default

UPI, IPI, or any other customized solution is supported for installation on OCP. Deployment environment can be Bare Metal or virtual (VMware ESXi).

The configuration comprises a network of master nodes, worker nodes, and a bastion node. CLI access from the bastion node to the master and worker nodes must be enabled.

You can perform a rolling upgrade of OCP or Kubernetes platforms. Before upgrading, see the table above for the supported platform and kernel versions. Ensure you upgrade to an InfoScale-supported version. Refer to OpenShift or Kubernetes documentation for steps to upgrade. For redundant Storage and Applications, the downtime during upgrade is zero.

Disk space requirements

The following table lists the minimum disk space requirements for Oracle Linux 8.4 (OEL) for each product when the `/opt`, `/root`, `/var`, and `/bin` directories are created on the same disk.

Table 2-3 Disk space requirements for Oracle Linux 8.4

Product name	Oracle Linux 8.4 (MB)
Veritas InfoScale Storage	3305

Note: OpenShift Container Platform runs special components which consume memory. See the OpenShift documentation - https://docs.openshift.com/container-platform/4.9/ nodes/clusters/nodes-cluster-resource-configure.html to understand memory requirements. As an OpenShift administrator, set resource limit for the Prometheus pod. See https://access.redhat.com/solutions/3867881 to know the Red Hat recommendations. Modify `resources.limits` and `resources.requests` before installing InfoScale.

Hardware requirements

Table 2-4 Hardware requirements

Requirement	Description
Memory (Operating System)	Minimum 24 GB
CPU (on Kubernetes)	On Physical servers - a minimum of 2 processors with 6/8 cores each. On Virtual machines (VMware-like environment) - a minimum of 4 vCPUs.
CPU (on OpenShift)	On Physical servers - a minimum of 2 processors with 6/8 cores each. On Virtual machines (VMware-like environment) - a minimum of 12 vCPUs for master node and 8 vCPUs for worker nodes.
Node	All nodes in a Cluster must have the same operating system version.

Table 2-4 Hardware requirements *(continued)*

Requirement	Description
Storage	Storage can be one or more shared disks, or a disk array connected either directly to the nodes of the cluster or through a Fibre Channel Switch. Nodes can also have non-shared or local devices on a local I/O channel. In a Flexible Storage Sharing (FSS) environment, shared storage may not be required.
Cluster platforms	There are several hardware platforms that can function as nodes in a Veritas InfoScale cluster. For the InfoScale cluster to work correctly, all nodes must have the same time. If you are not running the Network Time Protocol (NTP) daemon, ensure the time on all the systems comprising your cluster is synchronized.
SAS or FCoE	Each node in the cluster must have an SAS or FCoE I/O channel to access shared storage devices. The primary components of the SAS or Fibre Channel over Ethernet (FCoE) fabric are the switches and HBAs.

For additional information, see the hardware compatibility list (HCL) at: https://www.veritas.com/content/support/en_US/doc/infoscale_hcl_8x_unix.

Number of nodes supported

Veritas InfoScale for Containers supports cluster configurations comprising up to 16 worker nodes.

TECHNOLOGY PREVIEW: DR support

DR supports two cluster configuration - one primary cluster and one secondary/DR cluster.

Chapter 3

Preparing to install InfoScale on Containers

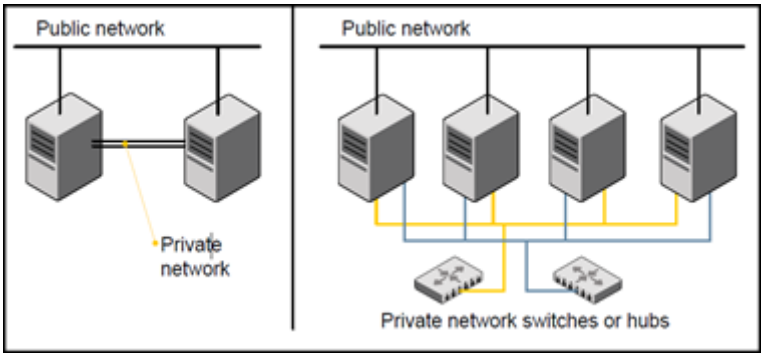
This chapter includes the following topics:

Setting up the private network

If you do not specify IP addresses in `cr.yaml` (the configuration file used for specifying details like Node names, IP addresses), InfoScale for Containers uses the Container links. Veritas recommends setting up a private network between the systems for optimal performance. You can use either NICs or aggregated interfaces to set up private network. You can use network switches instead of hubs. Refer to the Cluster Server Administrator's Guide to review performance considerations.

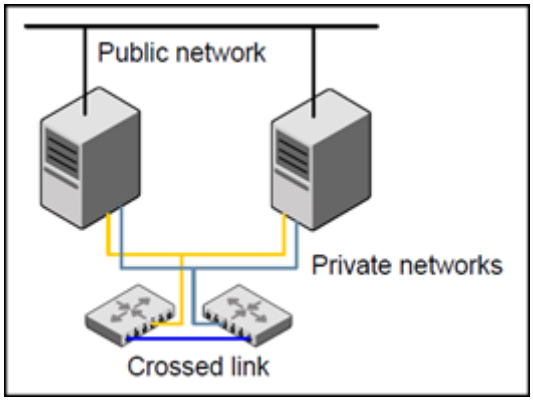
The following figure shows two private networks for use with InfoScale.

Figure 3-1 Private network setups: two-node and four-node clusters



You need to configure at least two independent networks between the cluster nodes with a network switch for each network. You can also interconnect multiple layer 2 (L2) switches for advanced failure protection. Such connections for LLT are called cross-links. The following figure shows a private network configuration with crossed links between the network switches.

Figure 3-2 Private network setup with crossed links



Veritas recommends the following configuration

- Use at least two private interconnect links. The private interconnect link is used to share cluster status across all the systems, which is important for membership arbitration and storage access.

To set up the private network

1. Install the required network interface cards (NICs). Create aggregated interfaces if you want to use the NICs to set up private network
2. Use crossover Ethernet cables, switches, or independent hubs for each Veritas InfoScale communication network. Note that the crossover Ethernet cables are supported only on two systems.

   Ensure that you meet the following requirements:

   - The power to the switches or hubs must come from separate sources.
   - On each system, you must use two independent network cards to provide redundancy.
   - If a network interface is part of an aggregated interface, you must not configure the network interface under LLT. However, you can configure the aggregated interface under LLT.
   - When you configure Ethernet switches for LLT private interconnect, disable the spanning tree algorithm on the ports used for the interconnect.
3. During the process of setting up heartbeat connections, consider a case where a failure removes all communications between the systems.

   Note: Note that a chance for data corruption exists if the systems are still running and can access the shared storage.
4. Test the network connections. Assign network addresses and use telnet or ping to verify communications.

Guidelines for setting the media speed for LLT interconnects

Review the following guidelines for setting the media speed for LLT interconnects:

- Veritas recommends that you set the same media speed setting on each Ethernet card on each node.
- If you have hubs or switches for LLT interconnects, then set the hub or switch port to the same setting as used on the cards on each node. Details for setting the media speeds for specific devices are outside of the scope of this manual. Consult the device's documentation or the operating system manual for more information.

Guidelines for setting the maximum transmission unit (MTU) for LLT

LLT can operate on default 1500 MTU but Veritas recommends increasing MTU to achieve maximum performance. Review the following guidelines for setting the MTU for LLT interconnects:

- Set the maximum transmission unit (MTU) to the highest value (typically 9000) supported by the NICs when LLT links are configured over Ethernet or UDP. Ensure that the switch is also set to 9000 MTU.
- For virtual NICs, all components (the virtual NIC, the corresponding physical NIC, and the virtual switch) must be set to 9000 MTU.

Synchronizing time settings on cluster nodes

Ensure that the time settings on all OpenShift and Kubernetes cluster nodes are synchronized. If the nodes are not synchronized, timestamps for change (ctime) and modification (mtime) may not be consistent with the sequence in which operations actually happened. For instructions, see the operating system documentation.

Securing your InfoScale deployment

Consider the following measures on your OpenShift and Kubernetes clusters. After adopting these measures, InfoScale deployment on these clusters is more secure.

See OpenShift and Kubernetes documentation to know more about these measures.

1. On an air gapped system on OpenShift or a Kubernetes cluster, configure a secure image registry. This registry is used to download and host InfoScale images.

   Enable the following to reduce security risks.

   - Set up secure, encrypted channels to connect to the registry.
   - Authenticate users and control access to registry.
   - Scan images for vulnerabilities found in the Common Vulnerabilities and Exploits (CVE) database and sign these as known and trusted.

2. Enable encryption at rest and assign RBAC for sensitive data stored in OpenShift and Kubernetes Secrets. By default, data is stored unencrypted in the API server's underlying data store (`etcd`). Anyone with API access or access to `etcd`, can retrieve or modify a Secret. Additionally, anyone who is authorized to create a pod in a namespace can use that access to read any Secret in that namespace; this includes indirect access such as the ability to create a

deployment. When encryption at rest is enabled with appropriate RBAC to secrets, the sensitive data remains protected.

3. Configure the OpenShift or Kubernetes API server with TLS 1.2 or higher, and TLS ciphers to exclude vulnerable ciphers such as ciphers using block ciphers in CBC mode and ciphers using low-length encryption keys like DES block ciphers (56-bit encryption key).

   After this TLS configuration, use of SSL, unauthorized versions of TLS protocols, and vulnerable TLS ciphers is blocked and confidentiality of sensitive data during electronic transmission is maintained.

Configuring kdump

Veritas recommends configuring kdump on each of the cluster nodes before installing InfoScale. kdump creates crash dumps in the event of a kernel crash which help Veritas support in troubleshooting issues.

For OpenShift, see https://docs.openshift.com/container-platform/4.9/ support/troubleshooting/troubleshooting-operating-system-issues.html. For Kubernetes, you can refer to the Operating System documentation for the generic steps.

Chapter 4

Installing Veritas InfoScale on OpenShift

This chapter includes the following topics:

Introduction

This chapter informs you how to install InfoScale on an OpenShift cluster. For air gapped systems on OpenShift, installer files and container images must be downloaded from the Veritas Download Center. The container images are different for each platform. OpenShift systems with internet connectivity need to download installer files (yamls) only. You can install InfoScale from a VM/Server termed as the bastion node on an OpenShift cluster.

Note: As InfoScale supports HyperConverged architecture, all worker nodes that are a part of OpenShift cluster must be used for creating an InfoScale cluster. Veritas InfoScale is deployed on all the nodes you specify in the Custom Resource yaml file.

Prerequisites

1. Be ready with the following information -

- Names of all the nodes.

  Note: Run `oc get nodes -o wide` on the bastion node to obtain Names and IP addresses of the nodes.

  Use `NAME` and `INTERNAL-IP` from the output similar to the following -

```
NAME                 STATUS ROLES  AGE VERSION         INTERNAL-IP
ocp-cp-1.lab.ocp.lan Ready master 54d v1.22.1+d8c4430 192.168.22.201
77.rhaos4.9.gitd745cab.el8
ocp-cp-2.lab.ocp.lan Ready master 54d v1.22.1+d8c4430 192.168.22.202
77.rhaos4.9.gitd745cab.el8
ocp-cp-3.lab.ocp.lan Ready master 54d v1.22.1+d8c4430 192.168.22.203
77.rhaos4.9.gitd745cab.el8
ocp-w-1.lab.ocp.lan Ready worker  54d v1.22.1+d8c4430 192.168.22.211
77.rhaos4.9.gitd745cab.el8
ocp-w-2.lab.ocp.lan Ready worker  54d v1.22.1+d8c4430 192.168.22.212
77.rhaos4.9.gitd745cab.el8
ocp-w-3.lab.ocp.lan Ready worker  54d v1.22.1+d8c4430 192.168.22.213
77.rhaos4.9.gitd745cab.el8
ocp-w-4.lab.ocp.lan Ready worker  54d v1.22.1+d8c4430 192.168.22.214
77.rhaos4.9.gitd745cab.el8
```

- Operating system device path of the disks which are being managed by other storage vendors that need to be excluded from InfoScale disk group.
- Optionally if you want to exclude boot disks, device path to the boot disks.

  Note: Veritas recommends excluding boot disks.

- If you have internet connectivity and download is allowed, you must be logged in to Red Hat registry.
- For air gapped systems, Custom Registry address to set up registry where InfoScale images are pushed.

2. Ensure that all nodes are synchronized with the NTP Server.
3. Reserve network ports for exclusive use of InfoScale as under -

Component	Port
LLT over UDP	Serially onwards 50000 (as many as configured LLT links)
VVR (Needed only if you want to configure DR)	4145 (UDP), 8199 (TCP), 8989 (TCP)

4. Add local or shared storage to all the worker nodes before you proceed with the deployment.
5. Ensure that stale InfoScale kernel modules (`vxio`/`vxdmp`/`veki`/`vxspec`/`vxfs`/`odm`/`glm`/`gms`) from previous installation do not exist on any of the worker nodes.

Note: You can reboot a worker node to unload all stale InfoScale kernel modules.

Installing InfoScale on a system with Internet connectivity

If your system is connected to the Internet, you can download operators and install. With a Red Hat account, you can connect to the Red Hat portal to download operators by using Command Line Interface (CLI) or the web console. Click the appropriate link below.

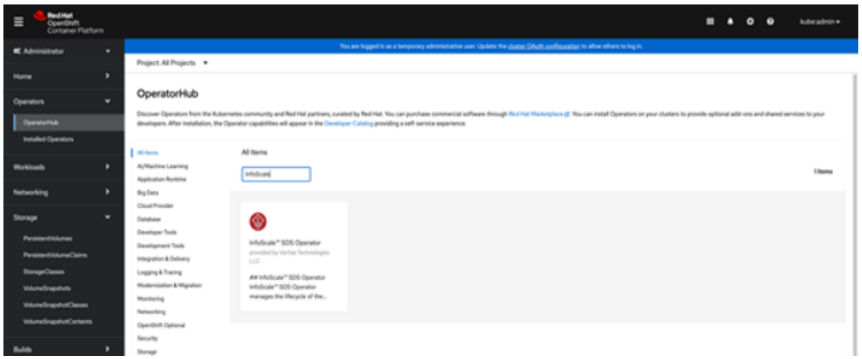
- Using web console of OperatorHub
- Installing from OperatorHub by using Command Line Interface (CLI)
- Installing by using YAML.tar

Using web console of OperatorHub

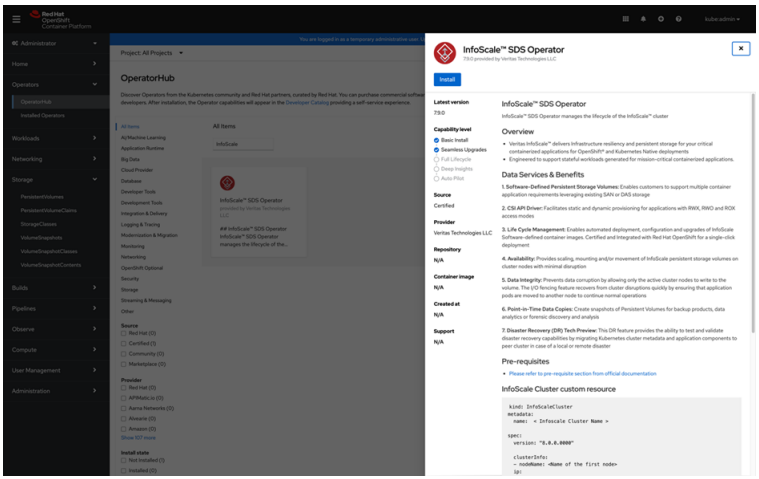
Complete the following steps to install InfoScale operator .

1. Connect to the OpenShift console and access the Catalog menu.
2. In the left frame, click **Operators > OperatorHub**. You can select and install the operator here.

3 Enter InfoScale in **All Items**. The InfoScale Operator is displayed.

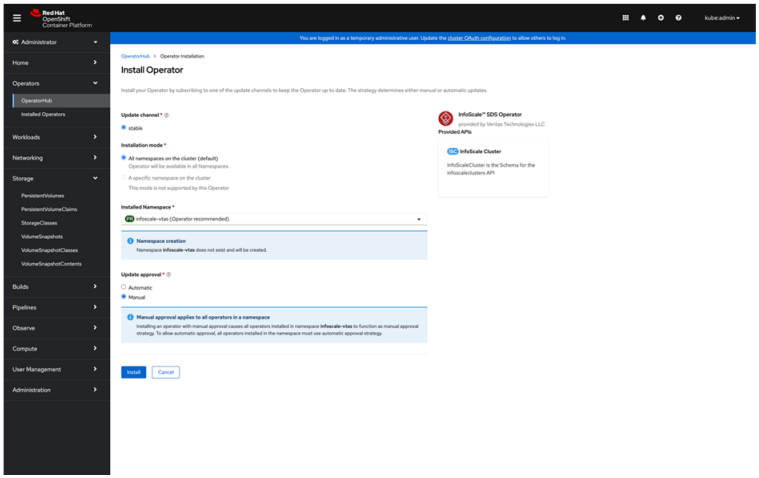


4 Select the Operator and click **Install** in the following screen.

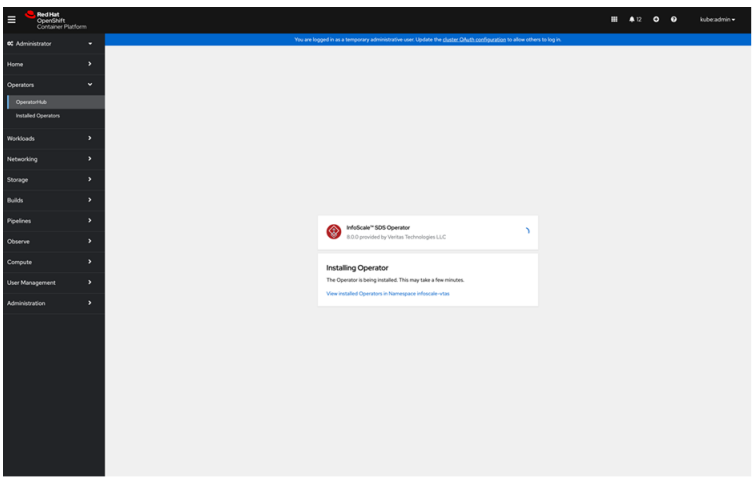


5 In the following screen, select **infoscale-vtas** in **Installed Namespace** and **Manual** in **Update approval**.

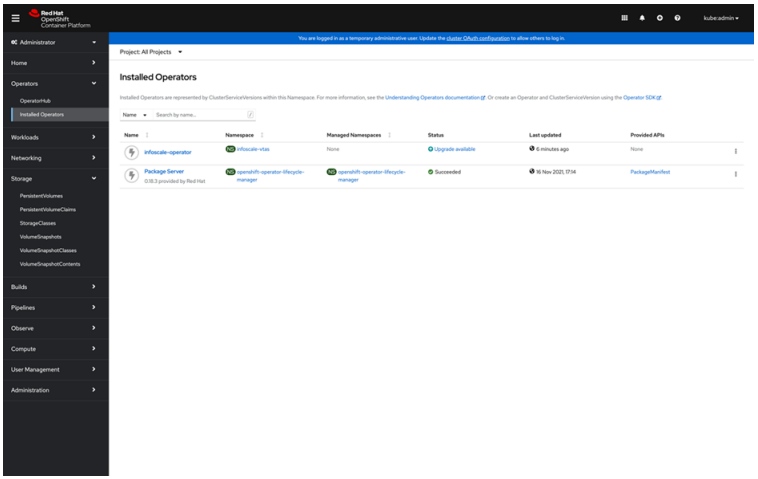
Note: Veritas recommends these configurations. You can select any other **Namespace** (including openshift-operators) for installation. Selecting Manual as the Install plan avoids automatic updates of the operator.



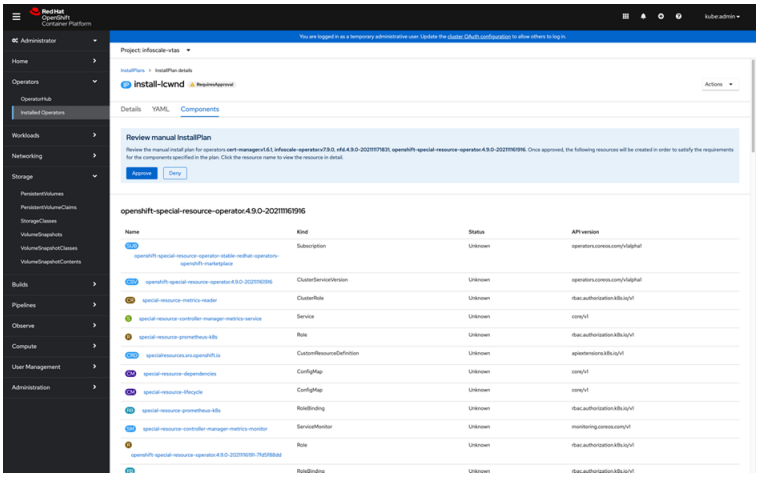
6 Click **Install**. InfoScale installation begins as under.



7 In the left frame, select **Operators > Installed Operators**. In the following screen, click **Upgrade available** for **infoscale-operator**.

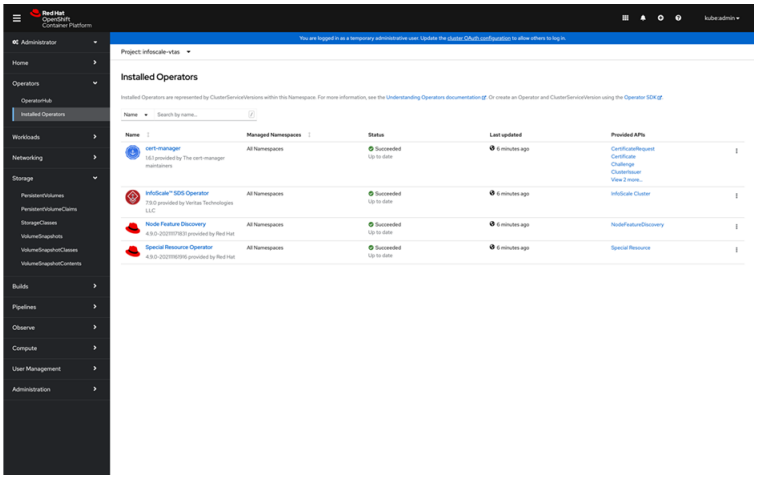


8 In the screen that opens; In **Review manual InstallPlan**, click **Preview InstallPlan** followed by **Approve** as under. Installation begins.



9 Wait till installation is complete. Cert-manager, Special Resource Operator, and Node Feature Discovery Operator along with InfoScale SDS operator are installed in `infoscale-vtas` or the namespace you provide. Cert-manager, Special Resource Operator, and Node Feature Discovery Operator are the dependencies for InfoScale installation. If these dependencies are already installed in `infoscale-vtas` or `openshift-operators`, installation is skipped.

10 In the left frame, select **Operators > Installed Operators**. Check if Status of all Operators is Succeeded as under.

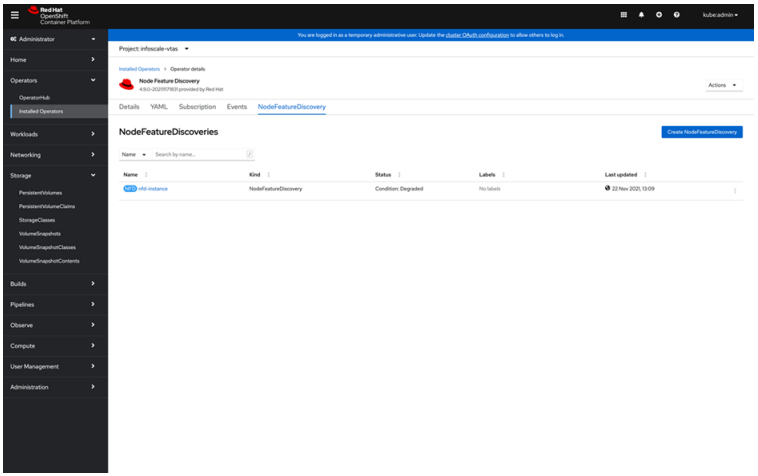


11 After all these Operators are installed successfully, click **NodeFeatureDiscovery** in **Provided APIs**.

Note: If NFD instance is already created, go to step 16.

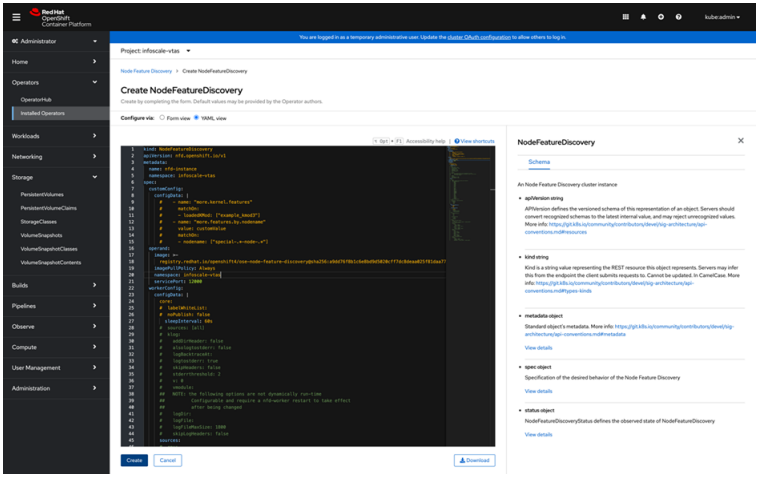
---

12 In **NodeFeatureDiscovery,** click **Create NodeFeatureDiscovery** in the upper-right corner of the screen.



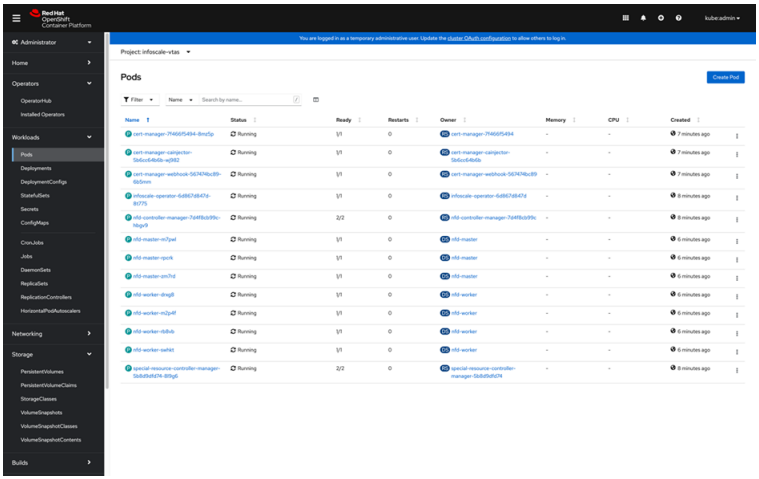
13 In the **CreateNodeDiscovery** screen, click **YAML view** in **Configure via:**.

- In metadata, change namespace to **infoscale-vtas**.
- Optionally, in spec:operand, change namespace to **infoscale-vtas**.



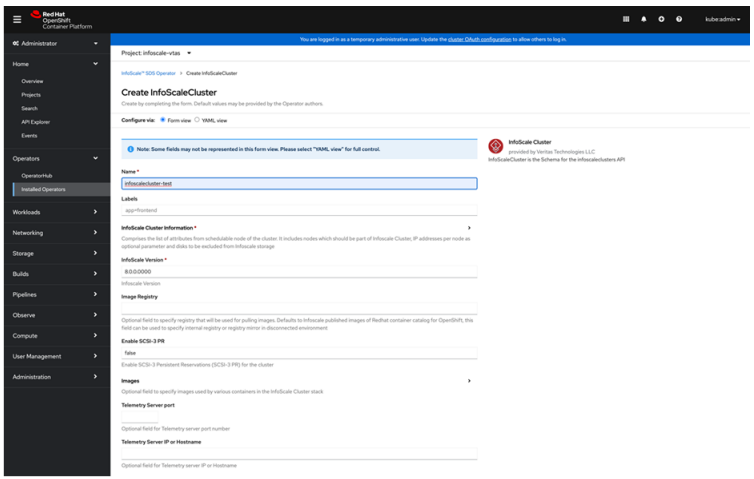
14 Click **Create** to create Node Feature Discovery instance.

15 After a successful creation of Node Feature Discovery, click **Workloads > Pods** in the left frame. Review names of the listed pods. Node Feature Discovery must be created on all nodes and is indicated by a prefix **nfd**.

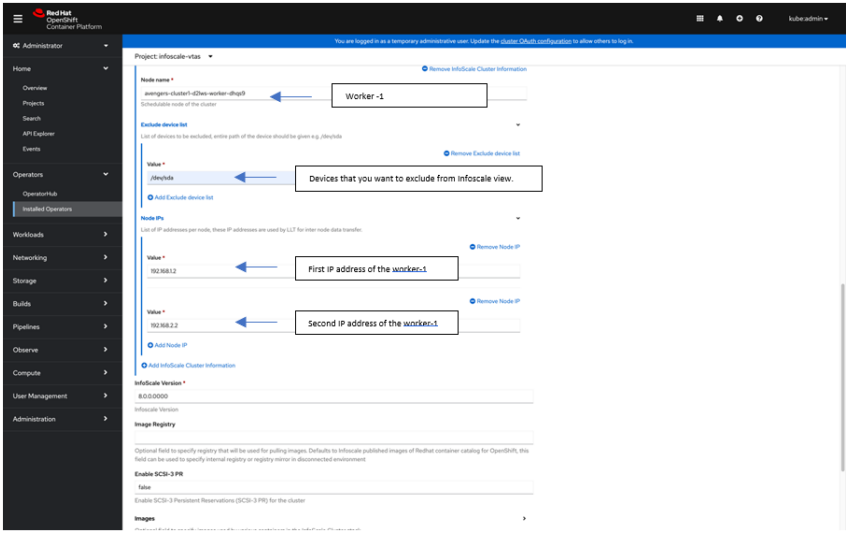


16 Click **Operators > Installed Operators** in the left frame. You can now create an InfoScale cluster. Click **InfoScale Cluster** in **Provided APIs**.

17 Click **Create InfoScaleCluster** in the upper-right corner of your screen. The following screen opens.



18 Enter **Name** for the cluster and click **InfoScale Cluster Information**. Enter information about the nodes here.



19 Enter **Node name** for at least two nodes. Optionally, you can enter IP addresses of nodes in **Node IPs** and the device path of the disk that you want to exclude from the InfoScale disk group in **Exclude-device list**. For each node, you must add two IP addresses.

Note: OpenShift cluster must have at least two nodes as minimum two nodes are needed to form a cluster.

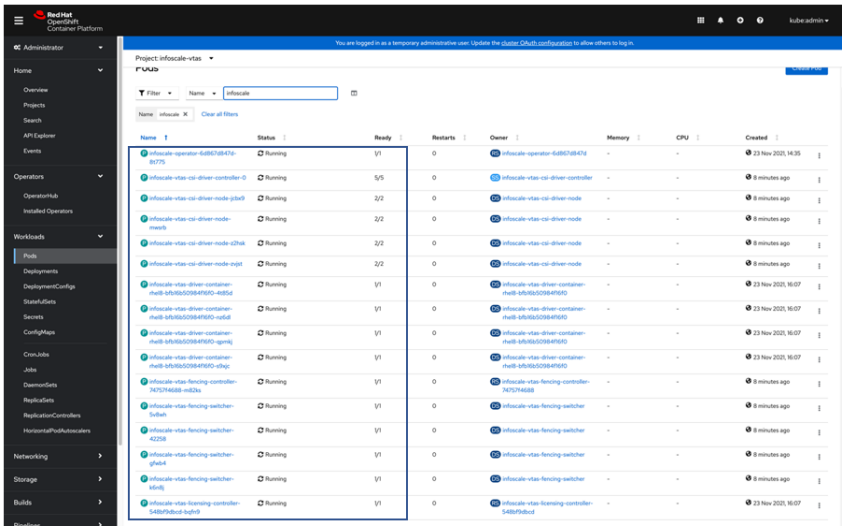
---

20 To add more nodes, click **Add InfoScale Cluster Information**. You can add up to 16 nodes.

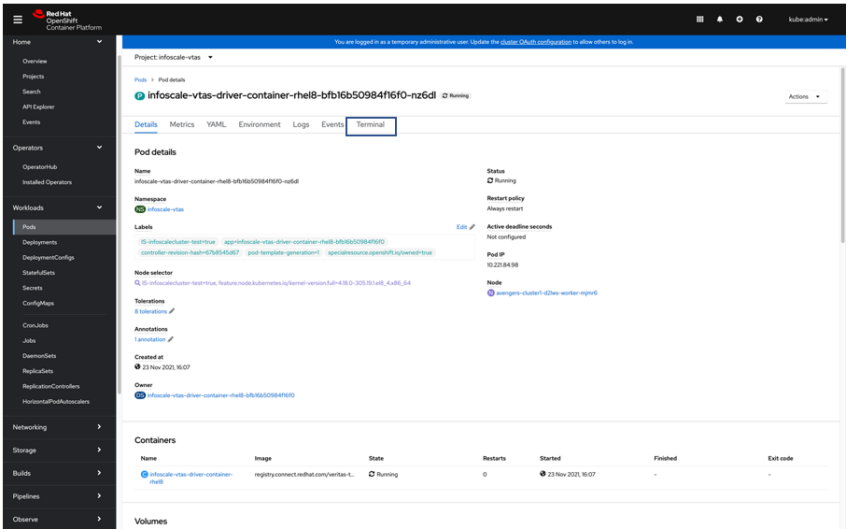
21 Do not enter any information in **Image Registry**, **Telemetry Server port**, and **Telemetry Server IP or Hostname**. Skip these fields.

22 Click **Create** to create an InfoScale cluster.

23 Wait till the cluster is created. Click **Workload > Pods** in the left frame. Review the name of the Node and Status in output similar to the following output. Status of the node must be 'Running'



24 Click any of driver containers. The following screen opens.



25 Click **Terminal** and run

`/etc/vx/bin/vxclustadm nidmap`.

Nodes are listed on screen. The status must be 'Joined'

After a successful deployment of InfoScale, diskgroup gets automatically created.

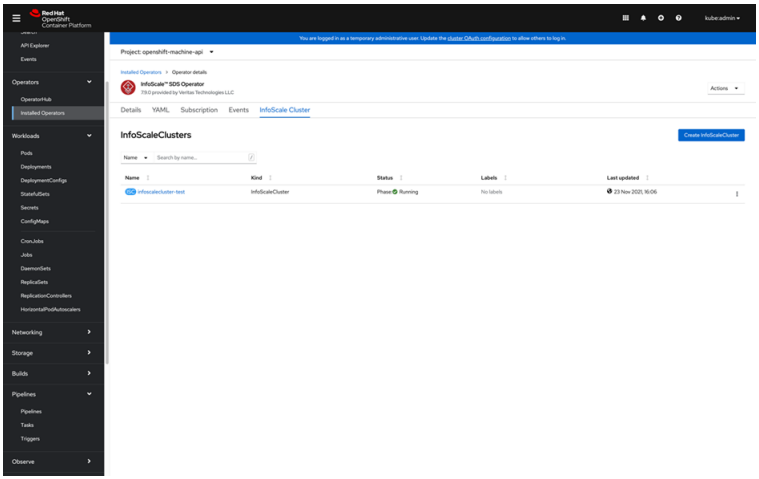
Adding Nodes to an InfoScale cluster by using OLM

You can add nodes to an already configured InfoScale cluster. Complete the following steps

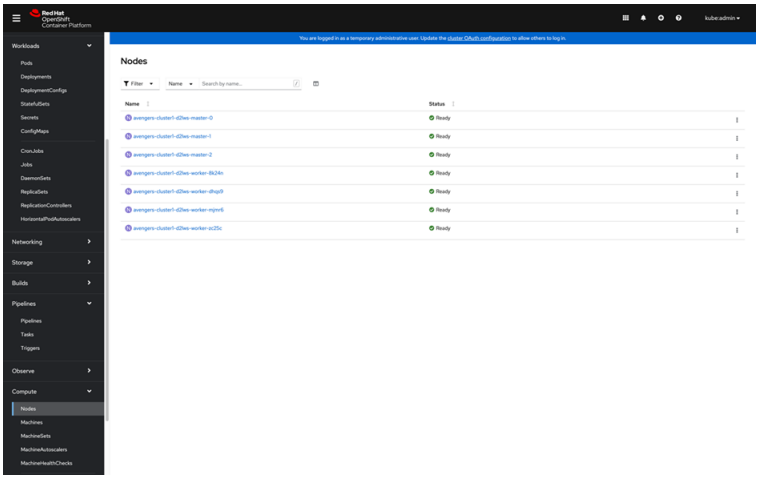
1 Connect to the OpenShift console and access the Catalog menu.

2 In the left frame, select **Operators > Installed Operators**. Click **InfoScale Cluster** under **Provided APIs**.

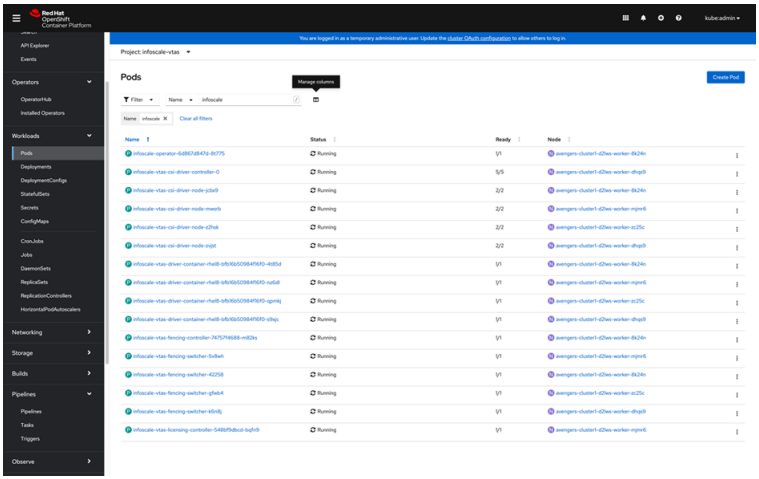
3 Review the status of the cluster to which you want to add nodes. The Status of the cluster must be 'Running'.



4 Click **Compute > Nodes** in the left frame. Review status of the nodes you want to add to the InfoScale cluster. Status of the nodes must be 'Ready'

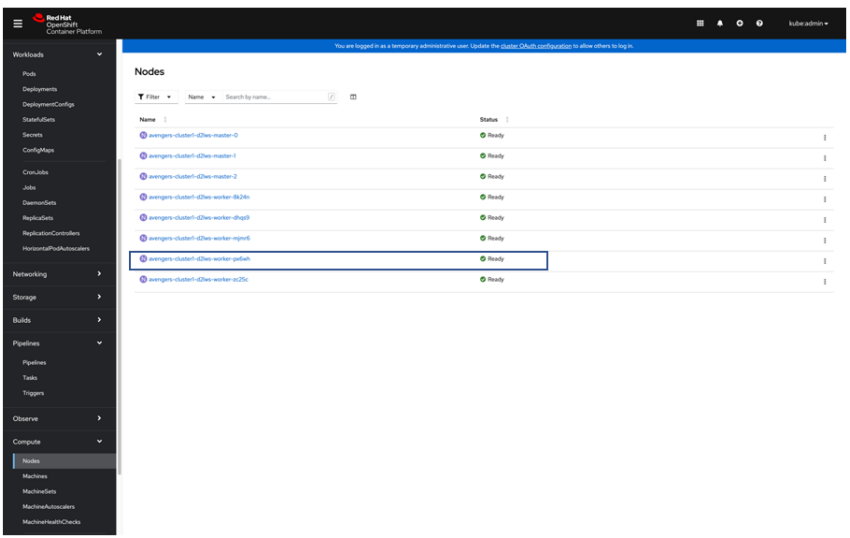


5 Click **Workload > Pods** in the left frame. Review status of the Pods. Pods must be 'Running'.



6 Add node to the OpenShift cluster. Refer to OpenShift documentation.

The node must be ready as under.



7 Refer to step 4 to ensure that the Node status is 'Ready'. You can add these nodes to the InfoScale cluster.

Note: You must add all OpenShift worker nodes to InfoScale cluster.

---

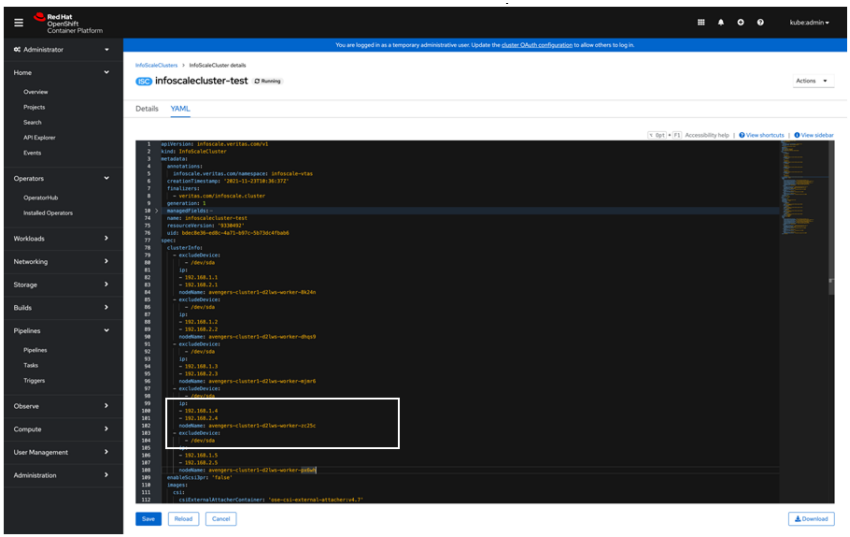
8 In the left frame, select **Operators > Installed Operators**. Click **InfoScale Cluster** under **Provided APIs**.

9 In **InfoScaleClusters**, click the **Name** of the cluster to which you want to add nodes.

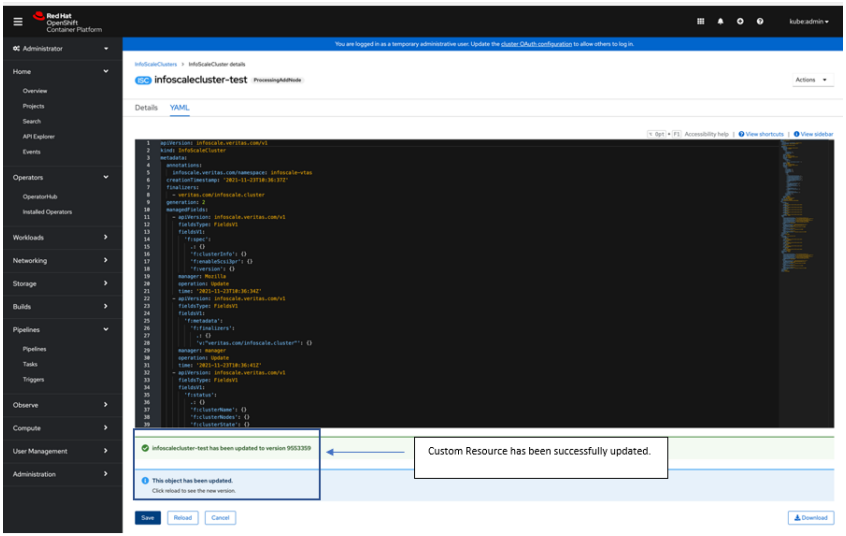
10 Click **YAML** in the screen that opens.

11 Edit the YAML to add information about the nodes like **nodeName**, **IP**, and **excludeDevice**. IP addresses for the node and the path to exclude devices is optional. You must enter the name of the node as **nodeName**. Click **Save**.

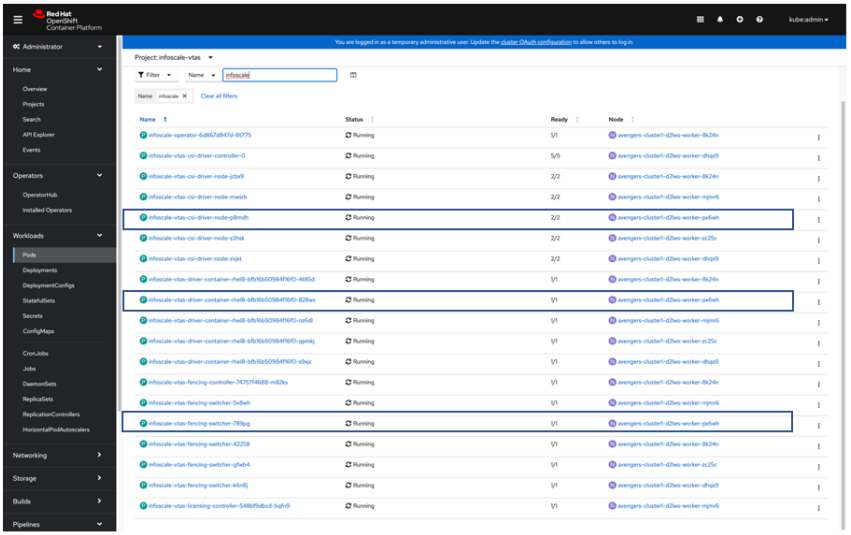
Note: If IP addresses are indicated for the existing nodes in the cluster, you must add IP addresses for the nodes you are adding. The number of IP addresses for the new nodes must be the same as the number of IP addresses for the existing nodes.



12 Messages in the following screen indicate that nodes addition is successful.



13 Review status of the Pods. See step 5 above. The newly added pods must be 'Running'.

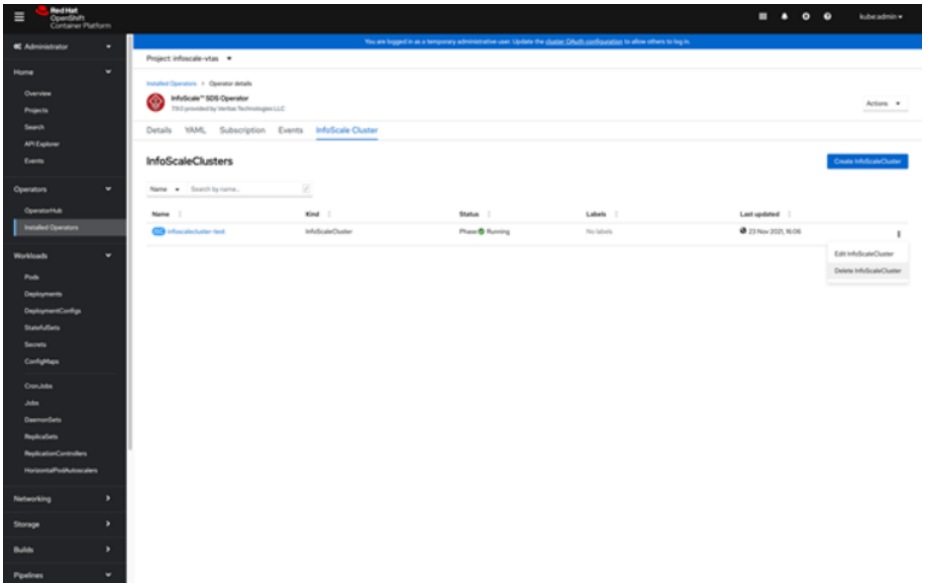


14 Review status of the InfoScale cluster. See step 3 above. The cluster must be 'Running'.

Undeploying and uninstalling InfoScale

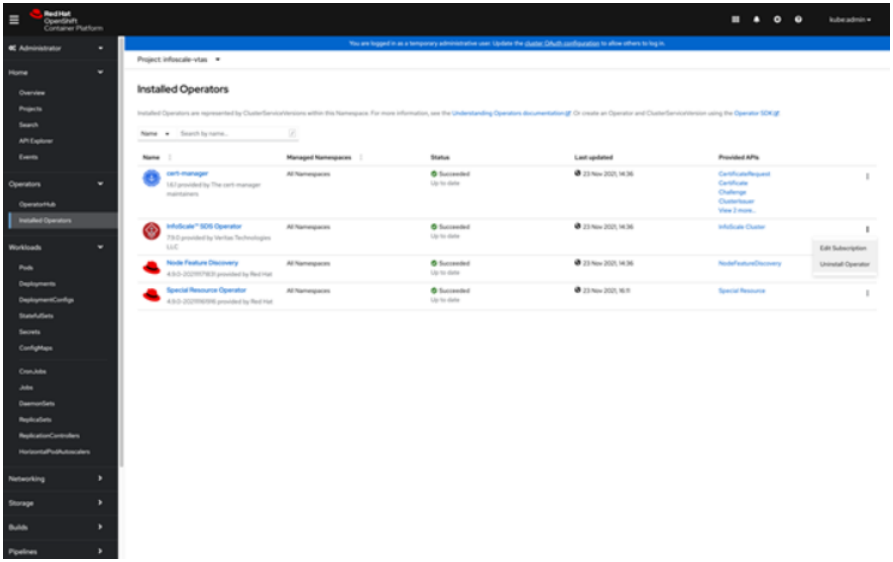
Complete the following steps

1. Connect to the OpenShift console and access the Catalog menu.
2. In the left frame, select **Operators > Installed Operators**. Click **InfoScale Cluster** under **Provided APIs**.
3. The installed and deployed InfoScale clusters are listed. Right-click the three vertical dots on the right of the screen for the cluster you want to delete. Select and click **Delete**.



4. Confirm delete and click **Workloads > Pods**. The pods on the Worker nodes must not be listed. You can now delete all Operators.

5 Click **Operators > Installed Operators**. All Installed Operators are listed.



6 Right-click the three vertical dots on the right of the screen for InfoScale SDS Operator. Select and click **Delete**. Confirm Delete.

7 Similarly, delete Special Resource Operator, Node Feature Discovery, and cert-manager. Follow this order for uninstalling operators.

Installing from OperatorHub by using Command Line Interface (CLI)

Complete the following steps.

Downloading `YAML.tar`

1 Download `YAML.tar` from the Veritas Download Center.

2 Untar `YAML.tar`.

After you untar `YAML.tar`, a folder `/YAML/OpenShift/OLM/` is created and all files required for installation are available in the folder.

Note: An OpenShift cluster already has a namespace `openshift-operators`. You can choose to install InfoScale in `openshift-operators`.

Optionally, you can configure a new user - infoscale-admin, associated with a Role-based Access Control (RBAC) clusterrole defined in

`infoscale-admin-role.yaml`, to deploy InfoScale and its dependent components. infoscale-admin as a user when configured has clusterwide access to only those resources needed to deploy InfoScale and its dependent components such as SRO/NFD/Cert Manager in the desired namespaces.

To provide a secure and isolated environment for InfoScale deployment and associated resources, the namespace associated with these resources must be protected from access of all other users (except super user of the cluster), with appropriate RBAC implemented.

Run the following commands on the bastion node to create a new user - infoscale-admin and a new project and assign role or clusterrole to infoscale-admin. You must be logged in as a super user.

Configuring a new user

1. `oc new-project <New Project name>`

   A new project is created for InfoScale deployment.

2. `oc adm policy add-role-to-user admin infoscale-admin`

   Following output indicates that administrator privileges are assigned to the new user - infoscale-admin within the new project.

   ```
   clusterrole.rbac.authorization.k8s.io/admin added: "infoscale-admin"
   ```

3. `oc apply -f /YAML/OpenShift/OLM/infoscale-admin-role.yaml`

   Following output indicates that a clusterrole is created.

   ```
   clusterrole.rbac.authorization.k8s.io/infoscale-admin-role created
   ```

4. `oc adm policy add-cluster-role-to-user infoscale-admin-role infoscale-admin`

   Following output indicates that a clusterrole created is associated with infoscale-admin.

   ```
   clusterrole.rbac.authorization.k8s.io/infoscale-admin-role added:
             "infoscale-admin"
   ```

After creating this user, you can login as `infoscale-admin` to perform all operations involved in installing InfoScale, configuring cluster, and adding nodes.

Installing Operators

1 Run the following command on the bastion node.

Note: Ignore this step if you want to install in `openshift-operators`.

```
oc create namespace infoscale-vtas
```

Review output similar to the following to verify whether the namespace is created successfully.

```
namespace/infoscale-vtas created
```

2 Run the following command on the bastion node to create subscription.

Note: If you want to install InfoScale in `openshift-operators`, edit `/YAML/OpenShift/OLM/infoscale-sub.yaml`. Change namespace from **infoscale-vtas** to **openshift-operators**

```
oc create -f /YAML/OpenShift/OLM/infoscale-sub.yaml
```

Following output indicates a successful command run.

```
subscription.operators.coreos.com/infoscale-operator created
```

3 Run the following command on the bastion node to create an operator group.

Note: Ignore this step if you want to install in `openshift-operators`.

```
oc create -f /YAML/OpenShift/OLM/infoscale-og.yaml
```

Following output indicates a successful command run.

```
operatorgroup.operators.coreos.com/infoscale-opgroup created
```

4 Run the following command on the bastion node.

```
oc get sub,og -n infoscale-vtas
```

Following output indicates a successful command run.

```
NAME                                    PACKAGE
subscription.operators.coreos.com   infoscale-operator
/infoscale-operator


SOURCE                CHANNEL
certified-operators   stable


NAME                                                        AGE
operatorgroup.operators.coreos.com/infoscale-opgroup   24s
```

5 Run the following command on the bastion node.

```
oc get ip -A
```

Use **installation-name** from the output similar to the following output.

```
NAMESPACE        NAME
infoscale-vtas   <installation-name>


CSV                                          APPROVAL   APPROVED
openshift-special-resource-operator.4.9.0  Manual     false
-202111041612
```

6 Run the following command on the bastion node.

Note: Do not include the angle brackets (< >) in the command.

```
oc patch installplan <installation-name> --namespace
infoscale-vtas --type merge --patch '{"spec":{"approved":true}}'
```

Following output indicates a successful command run.

```
installplan.operators.coreos.com/<installation-name> patched
```

7 Run the following command on the bastion node.

```
oc get ip -A
```

Review output similar to the following . Check if `APPROVED` is `true`.

```
NAMESPACE        NAME
infoscale-vtas   <installation-name>


CSV                                          APPROVAL   APPROVED
openshift-special-resource-operator.4.9.0    Manual     true
-202111041612
```

8 Run the following command on the bastion node to check the status of csv.

```
oc get csv
```

Components which are getting installed or are pending are listed, as under.

```
NAME              DISPLAY        VERSION               REPLACES   PHASE
cert-manager.v1.6.1        cert-manager              1.6.1    Installing
infoscale-operator.v8.0.0  InfoScale™ SDS Operator   8.0.0    Installing
nfd.4.9.0-202111020858     Node Feature Discovery 4.9.0-202111020858
                                                               Pending
openshift-special-resource Special Resource Operator 4.9.0-202111041612
-operator.4.9.0-202111041612                                   Installing
```

9 Run the following command on the bastion node to check the status of operator group and subscription.

```
oc get og,sub -n infoscale-vtas
```

Review output similar to the following output for the status.

```
operatorgroup.operators.coreos.com/infoscale-opgroup   47m

NAME
subscription.operators.coreos.com/cert-manager-stable-community
-operators-openshift-marketplace
subscription.operators.coreos.com/infoscale-operator
subscription.operators.coreos.com/nfd-stable-redhat-operators
-openshift-marketplace
subscription.operators.coreos.com/openshift-special-resource-operator-
stable-redhat-operators-openshift-marketplace


 PACKAGE                              SOURCE                CHANNEL
cert-manager                          community-operators   stable
infoscale-operator                    certified-operators   stable
nfd                                   redhat-operators      stable
openshift-special-resource-operator redhat-operators      stable
```

10 Run the following command on the bastion node again.

```
oc get csv
```

Review the output if all components are installed successfully.

```
NAME             DISPLAY        VERSION              REPLACES   PHASE
cert-manager.v1.6.1        cert-manager             1.6.1    Succeeded
infoscale-operator.v8.0.0  InfoScale™ SDS Operator   8.0.0    Succeeded
nfd.4.9.0-202111020858     Node Feature Discovery 4.9.0-202111020858
                                                              Succeeded
openshift-special-resource Special Resource Operator 4.9.0-202111041612
-operator.4.9.0-202111041612                                  Succeeded
```

11 After a successful installation of these components, create a NodeFeatureDiscovery CR. `NodeFeatureDiscovery.yaml` is a NodeFeatureDiscovery Custom Resource (CR).

12 Run the following command on the bastion node.

Note: If you want to install InfoScale in `openshift-operators`, edit `/YAML/OpenShift/OLM/NodeFeatureDiscovery.yaml`. Change namespace from **infoscale-vtas** to **openshift-operators** for the nfd instance.

```
oc create -f /YAML/OpenShift/OLM/NodeFeatureDiscovery.yaml
```

Following output indicates a successful creation.

```
nodefeaturediscovery.nfd.openshift.io/nfd-instance created
```

13 Run the following command to check the status of all operator pods in infoscale-vtas.

Note: If you have installed in `openshift-operators`, run `oc get pods -n openshift-operators`.

```
oc get pods -n infoscale-vtas
```

```
NAME                                    READY  STATUS    RESTARTS   AGE
cert-manager-64c9cb7499-ppgbk           1/1    Running   0          165m
cert-manager-cainjector-5596f8f575
                                -2f246  1/1    Running   0          165m
cert-manager-webhook-7485d9dd59-8641l4  1/1    Running   0          165m
infoscale-operator-6dd8d77bf8-qwg2p     1/1    Running   0          165m
nfd-controller-manager-5fc85ff79-gx4qb  2/2    Running   0          165m
nfd-master-6zs5p                        1/1    Running   0          55m
nfd-master-ktc7s                        1/1    Running   0          55m
nfd-master-n2dh9                        1/1    Running   0          55m
nfd-worker-795vs                        1/1    Running   0          55m
nfd-worker-8n2m9                        1/1    Running   0          55m
nfd-worker-9j845                        1/1    Running   0          55m
nfd-worker-vwkwq                        1/1    Running   0          55m
special-resource-controller-manager-
dc5d6b768-2sk4k                         2/2    Running   0          165m
```

Configuring cluster

After successfully installing InfoScale operator, you can create a cluster.

1. Edit **clusterInfo** section of the sample `/YAML/OpenShift/OLM/cr.yaml` for InfoScale specifications as under -

---

Note: You can specify up to 16 worker nodes in `cr.yaml`. Although cluster configuration is allowed even with one Network Interface Card, Veritas recommends a minimum of two physical links for performance and High Availability (HA). Number of links for each network link must be same on all nodes. Optionally, you can enter node level IP addresses. If IP addresses are not provided, IP addresses of OpenShift cluster nodes are used.

```
clusterInfo:
- nodeName: <Name of the first node>
  ip:
  -  <Optional - First IP address of the first node >
  -  <Optional - Second IP address of the first node>
  excludeDevice:
  - <Optional - Device path of the disk on the node that you want
                              to exclude from Infoscale disk group.>
- nodeName: <Name of the second node>
  ip:
 -  <Optional - First IP address of the second node >
 -  <Optional - Second IP address of the second node>
  excludeDevice:
  - <Optional - Device path of the disk on the node that you want
                              to exclude from Infoscale disk group.>
- nodeName: <Name of the third node>
  ip:
 -  <Optional - First IP address of the third node >
 -  <Optional - Second IP address of the third node>
  excludeDevice:
  - <Optional - Device path of the disk on the node that you want
                              to exclude from Infoscale disk group.>
.
.
.

 YOU CAN ADD UP TO 16 NODES.
```

Note: Do not enclose parameter values in angle brackets (<>). For example, Primarynode is the name of the first node; for **nodeName : <Name of the first node>** , enter **nodeName : Primarynode**. InfoScale on OpenShift is a keyless deployment.

2. Run the following command on the bastion node.

   ```
   oc create -f /YAML/OpenShift/OLM/cr.yaml
   ```

3. Run the following command on the bastion node to know the name and namespace of the cluster.

   ```
   oc get infoscalecluster
   ```

   Use the namespace from the output similar to the following.

   ```
   NAME                  NAMESPACE         VERSION      STATE          AGE
   infoscalecluster-dev   infoscale-vtas   8.0.0.0000   Running   1m15s
   ```

4. Run the following command on the bastion node to verify whether the pods are created successfully.

   ```
   oc get pods -n infoscale-vtas
   ```

   An output similar to the following indicates a successful creation of nodes

   ```
   NAME                              READY STATUS   RESTARTS  AGE
   infoscale-vtas-csi-driver-node-5tnct      2/2   Running  0         2m27s
   infoscale-vtas-csi-driver-node-6w2q7      2/2   Running  0         2m27s
   infoscale-vtas-csi-driver-node-lj4xz      2/2   Running  0         2m27s
   infoscale-vtas-csi-driver-node-vzq7s      2/2   Running  0         2m27s
   infoscale-vtas-driver-container-rhel8-7zcrk 1/1   Running  0         10m
   infoscale-vtas-driver-container-rhel8-f7h4f 1/1   Running  0         10m
   infoscale-vtas-driver-container-rhel8-qqjkv 1/1   Running  0         10m
   infoscale-vtas-driver-container-rhel8-ww8md 1/1   Running  0         10m
   infoscale-vtas-fencing-controller-5dd876748d-rbbgn 1/1  Running 0  2m39s
   infoscale-vtas-fencing-switcher-7tqwg     1/1   Running   0        2m49s
   infoscale-vtas-fencing-switcher-chllt     1/1   Running   0        2m49s
   infoscale-vtas-fencing-switcher-m5hp4     1/1   Running   0        2m49s
   infoscale-vtas-fencing-switcher-wdcqw     1/1   Running   0        2m49s
   infoscale-vtas-licensing-controller-7b749fb8d-xdwjn 1/1 Running 0  11m
   infoscale-operator-75667df67b-vjm5p              1/1   Running   0
   ```

After a successful InfoScale deployment, a disk group is automatically created. You can now create Persistent Volumes/ Persistent Volume Claims (PV / PVC) by using the corresponding Storage class.

Adding nodes to an existing cluster

Complete the following steps to add nodes to an existing InfoScale cluster-

1. Ensure that you add the worker nodes to the OCP cluster.

   Note: You must add all OpenShift worker nodes to the InfoScale cluster.

2. Run the following command on the bastion node to check whether the newly added node is Ready.

   ```
   oc get nodes -A
   ```

   Review output similar to the following

   ```
   NAME                   STATUS ROLES  AGE VERSION
   ocp-cp-1.lab.ocp.lan   Ready  master 54d v1.22.1+d8c4430
   ocp-cp-2.lab.ocp.lan   Ready  master 54d v1.22.1+d8c4430
   ocp-cp-3.lab.ocp.lan   Ready  master 54d v1.22.1+d8c4430
   ocp-w-1.lab.ocp.lan    Ready  worker 54d v1.22.1+d8c4430
   ocp-w-2.lab.ocp.lan    Ready  worker 54d v1.22.1+d8c4430
   ocp-w-3.lab.ocp.lan    Ready  worker 54d v1.22.1+d8c4430
   ocp-w-4.lab.ocp.lan    Ready  worker 54d v1.22.1+d8c4430
   ```

3. To add new nodes to an existing cluster, the cluster must be in a running state. Run the following command on the bastion node to verify.

   ```
   oc get infoscalecluster
   ```

   See the State in the output similar to the following -

   ```
   NAME                 NAMESPACE      VERSION    STATE   AGE
   infoscalecluster-dev infoscale-vtas 8.0.0.0000 Running 1m15s
   ```

4 Edit **clusterInfo** section of the sample `/YAML/OpenShift/cr.yaml` to add information about the new nodes.

In this example, worker-node-1 and worker-node-2 exist. worker-node-3 is being added.

Note: The number of IP addresses must be same for all nodes.

```
apiVersion: infoscale.veritas.com/v1
kind: InfoScaleCluster
metadata:
name: infoscalecluster-dev

spec:
  version: "8.0.0.0000"

  clusterInfo:
  - nodeName: "worker-node-1"
    ip:
    - "<IP address of worker-node-1>"
  - nodeName: "worker-node-2"
    ip:
    - "<IP address of worker-node-2>"
  - nodeName: "worker-node-3"
    ip:
    - "<IP address of worker-node-3>"
    excludeDevice:
    - /dev/sdm
    - /dev/sdn

.
.

 YOU CAN ADD UP TO 16 NODES.
```

5 Run the following command on the bastion node to initiate add node workflow.

```
oc apply -f /YAML/OpenShift/cr.yaml
```

6 You can run the following commands on the bastion node when node addition is in progress.

a. `oc get infoscalecluster`

See the State in the output as under. ProcessingAddNode indicates node is getting added.

```
NAME                    NAMESPACE         VERSION      STATE
infoscalecluster-dev    infoscale-vtas    8.0.0.0000   ProcessingAddNode
```

b. `oc describe infoscalecluster -n infoscale-vtas`

Output similar to following indicates the cluster status during add node. The cluster is Degraded when node addition is in progress.

```
Cluster Name:  infoscalecluster-dev
  Cluster Nodes:
    Exclude Device:
      /dev/sdm
      /dev/sdn
    Node Name:  worker-node-1
    Role:       Joined,Master
    Node Name:  worker-node-2
    Role:       Joined,Slave
    Node Name:  worker-node-3
    Role:         Out of Cluster
   Cluster State:  Degraded
   enableScsi3pr:  false
   Images:
    Csi:
      Csi External Attacher Container:      csi-attacher:v3.1.0
```

7 Run the following command on the bastion node to verify if pods are created successfully. It may take some time for the pods to be created.

```
oc get pods -n infoscale-vtas
```

Output similar to the following indicates a successful creation.

```
NAME                                    READY STATUS RESTARTS AGE
infoscale-vtas-csi-driver-node-5tnct  2/2   Running 0       2m27s
infoscale-vtas-csi-driver-node-6w2q7  2/2   Running 0       2m27s
infoscale-vtas-csi-driver-node-lj4xz  2/2   Running 0       2m27s
infoscale-vtas-driver-container-rhel8
                                -7zcrk 1/1   Running 0       10m
infoscale-vtas-driver-container-rhel8
                                -f7h4f 1/1   Running 0       10m
infoscale-vtas-driver-container-rhel8
                                -qqjkv 1/1   Running 0       10m
infoscale-vtas-fencing-controller
                     -5dd876748d-rbbgn 1/1   Running 0       2m39s
infoscale-vtas-fencing-switcher-7tqwg 1/1   Running 0       2m49s
infoscale-vtas-fencing-switcher-chllt 1/1   Running 0       2m49s
infoscale-vtas-fencing-switcher-m5hp4 1/1   Running 0       2m49s
infoscale-vtas-licensing-controller
                      -7b749fb8d-xdwjn 1/1   Running 0       11m
infoscale-operator-75667df67b-vjm5p   1/1   Running 0       11m
```

8 Run the following command on the bastion node to verify if the cluster is 'Running'

```
oc get infoscalecluster
```

See the State in the output similar to the following -

```
NAME                  NAMESPACE      VERSION    STATE   AGE
infoscalecluster-dev infoscale-vtas 8.0.0.0000 Running 1m15s
```

9 Run the following command on the bastion node to verify whether the cluster is 'Healthy'.

```
oc describe infoscalecluster
```

Check the **Cluster State** in the output similar to the following-

```
Status:
  Cluster Name:  infoscalecluster-dev
  Cluster Nodes:
    Node Name:    worker-node-1
    Role:         Joined,Master
    Node Name:    worker-node-2
    Role:         Joined,Slave
    Node Name:    worker-node-3
    Role:         Joined,Slave
  Cluster State:  Healthy
```

Undeploying and uninstalling InfoScale by using CLI

For a custom namespace, complete the following steps to undeploy and uninstall InfoScale.

1 Run the following command on the bastion node to undeploy.

```
oc delete -f /YAML/OpenShift/OLM/cr.yaml
```

Note: `cr.yaml` must be the same that was used for deployment.

2 Run the following command on the bastion node to delete the operator group.

```
oc delete og -n infoscale-vtas infoscale-opgroup
```

If InfoScale is installed in **openshift-operators**, run .

```
oc delete og -n openshift-operators infoscale-opgroup
```

3 Run the following command on the bastion node to delete subscription for the InfoScale operator.

Note: Ignore this step if you have installed in `openshift-operators`.

```
oc delete sub -n infoscale-vtas -all
```

4 Run the following commands on the bastion node to delete the ClusterServiceVersion.

- `oc get csv |egrep "cert-manager|Node Feature|Infoscale|Special Resource"|awk '{print $1}'`
  Use the csv_name and clusterserviceversion returned from this command in the following commands.
- `oc delete csv <csv_name> -n infoscale-vtas`

  Note: Ignore this step if you have installed in `openshift-operators`.

- `oc delete clusterserviceversion <csv_name>`

Note: While entering the command, ensure that you do not enclose the csv_name and crd_name in angle brackets.

5 Run the following commands on the bastion node the delete the CRDs (Custom Resource Definitions)

- `oc get crd | egrep 'cert-manager|special|info|nfd`
  All CRDs are listed. Use the names of the listed CRDs in the following commands to delete the CRDs one -by-one.
- `oc delete crd <crd_name>`

6 Run the following command on the bastion node to delete the namespace. Ignore if you had installed in `openshift-operators`.

```
oc delete ns infoscale-vtas
```

Note: After uninstallation, ensure that stale InfoScale kernel modules (`vxio`/`vxdmp`/`veki`/`vxspec`/`vxfs`/`odm`/`glm`/`gms`) do not remain loaded on any of the worker nodes. Rebooting a worker node deletes all such modules.

Installing by using YAML.tar

You must install a Special Resource Operator (SRO) first, before installing Veritas InfoScale. After the SRO is installed, the system is enabled for installing Veritas InfoScale.

1. Download `YAML.tar` from the Veritas Download Center.
2. Untar `YAML.tar`.

   After you untar `YAML.tar`, the folders `/YAML/OpenShift/`, `/YAML/OpenShift/air-gapped-systems` , `/YAML/DR`, and `/YAML/Kubernetes` are created. Each folder contains files required for installation.

Optionally, you can configure a new user - infoscale-admin, associated with a Role-based Access Control (RBAC) clusterrole defined in`infoscale-admin-role.yaml`, to deploy InfoScale and its dependent components. infoscale-admin as a user when configured has clusterwide access to only those resources needed to deploy InfoScale and its dependent components such as SRO/NFD/Cert Manager in the desired namespaces.

To provide a secure and isolated environment for InfoScale deployment and associated resources, the namespace associated with these resources must be protected from access of all other users (except super user of the cluster), with appropriate RBAC implemented.

Run the following commands on the bastion node to create a new user - infoscale-admin and a new project and assign role or clusterrole to infoscale-admin. You must be logged in as a super user.

1 `oc new-project <New Project name>`

A new project is created for InfoScale deployment.

2 `oc adm policy add-role-to-user admin infoscale-admin`

Following output indicates that administrator privileges are assigned to the new user - infoscale-admin within the new project.

```
clusterrole.rbac.authorization.k8s.io/admin added: "infoscale-admin"
```

3 `oc apply -f /YAML/OpenShift/infoscale-admin-role.yaml`

Following output indicates that a clusterrole is created.

```
clusterrole.rbac.authorization.k8s.io/infoscale-admin-role created
```

4 `oc adm policy add-cluster-role-to-user infoscale-admin-role infoscale-admin`

Following output indicates that a clusterrole created is associated with infoscale-admin.

```
clusterrole.rbac.authorization.k8s.io/infoscale-admin-role added:
          "infoscale-admin"
```

You must now perform all installation-related activities by logging in as infoscale-admin. A cluster super-user can also install InfoScale.

1. Run the following commands on the bastion node to install -
   - Run `oc create -f /YAML/OpenShift/sro.yaml` on the bastion node to install the Special Resource Operator (SRO) .
   - Run `oc create -f /YAML/OpenShift/sr.yaml` on the bastion node to create Special Resource.
2. Run the following commands on the bastion node and review the output to verify whether SR creation and SRO installation is successful.
   - `oc get pods -n openshift-special-resource-operator`
     Output similar to the following indicates a successful installation.

     ```
     NAME                                  READY   STATUS    RESTARTS   AGE
     special-resource-controller-manager-66c8fc64b5-9wv6l 1/1  Running 0
     ```

     Note: The name in the output here is used in the following command.

   - `oc logs special-resource-controller-manager-66c8fc64b5-9wv6l -n openshift-special-resource-operator -c manager`
     Output similar to the following indicates a successful installation.

     ```
     <timestamp>    INFO    status
      RECONCILE SUCCESS: Reconcile
     ```

   - `oc get SpecialResource`
     Output similar to the following indicates a successful installation.

```
NAME                          AGE
special-resource-preamble     2m24s
```

As your system is connected with the Internet, you must login to the Red Hat registry before you install InfoScale. All information about the worker nodes must be added to the `cr.yaml` file. All worker nodes become part of InfoScale cluster after `cr.yaml` is applied.

Note: After you download and untar `YAML.tar`, all files required for installation are available.

Complete the following steps to install `iso.yaml`.

1. Run the following command on all the worker nodes.

   ```
   podman login registry.connect.redhat.com Username:
   {REGISTRY-SERVICE-ACCOUNT-USERNAME} Password:
   {REGISTRY-SERVICE-ACCOUNT-PASSWORD}
   ```

   Wait for the message - **Login successful**.

2. Run the following command on the bastion node to install Veritas InfoScale.

   ```
   oc create -f /YAML/OpenShift/iso.yaml
   ```

3. Run the following command on the bastion node to verify whether the installation is successful

   ```
   oc get pods -n infoscale-vtas|grep infoscale-operator
   ```

   An output similar to the following indicates a successful installation. `READY 1/1` indicates that Storage cluster resources can be created.

   ```
   NAME                    READY   STATUS    RESTARTS    AGE
   infoscale-operator-6dc9bc8856-lh72f   1/1     Running   0        2d18h
   ```

Configuring cluster

After successfully installing InfoScale operator, you can create a cluster.

1. Edit **clusterInfo** section of the sample `/YAML/OpenShift/cr.yaml` for InfoScale specifications as under -

Note: You can specify up to 16 worker nodes in `cr.yaml`. Although cluster configuration is allowed even with one Network Interface Card, Veritas recommends a minimum of two physical links for performance and High Availability (HA). Number of links for each network link must be same on all nodes. Optionally, you can enter node level IP addresses. If IP addresses are not provided, IP addresses of OpenShift cluster nodes are used.

```
  clusterInfo:
  - nodeName: <Name of the first node>
    ip:
    -  <Optional - First IP address of the first node >
    -  <Optional - Second IP address of the first node>
    excludeDevice:
    - <Optional - Device path of the disk on the node that you want
                                to exclude from Infoscale disk group.>
  - nodeName: <Name of the second node>
    ip:
   -  <Optional - First IP address of the second node >
   -  <Optional - Second IP address of the second node>
    excludeDevice:
    - <Optional - Device path of the disk on the node that you want
                                to exclude from Infoscale disk group.>
  - nodeName: <Name of the third node>
    ip:
   -  <Optional - First IP address of the third node >
   -  <Optional - Second IP address of the third node>
    excludeDevice:
    - <Optional - Device path of the disk on the node that you want
                                to exclude from Infoscale disk group.>
.
.
.

 YOU CAN ADD UP TO 16 NODES.
```

Note: Do not enclose parameter values in angle brackets (<>). For example, Primarynode is the name of the first node; for **nodeName : <Name of the first node>** , enter **nodeName : Primarynode**. InfoScale on OpenShift is a keyless deployment.

2. Run the following command on the bastion node.

```
oc create -f /YAML/OpenShift/cr.yaml
```

3. Run the following command on the bastion node to know the name and namespace of the cluster.

```
oc get infoscalecluster
```

Use the namespace from the output similar to the following.

```
NAME                  NAMESPACE         VERSION      STATE          AGE
infoscalecluster-dev   infoscale-vtas   8.0.0.0000   Running   1m15s
```

4. Run the following command on the bastion node to verify whether the pods are created successfully.

```
oc get pods -n infoscale-vtas
```

An output similar to the following indicates a successful creation of nodes

```
NAME                              READY STATUS   RESTARTS  AGE
infoscale-vtas-csi-driver-node-5tnct      2/2   Running  0         2m27s
infoscale-vtas-csi-driver-node-6w2q7      2/2   Running  0         2m27s
infoscale-vtas-csi-driver-node-lj4xz      2/2   Running  0         2m27s
infoscale-vtas-csi-driver-node-vzq7s      2/2   Running  0         2m27s
infoscale-vtas-driver-container-rhel8-7zcrk 1/1   Running  0         10m
infoscale-vtas-driver-container-rhel8-f7h4f 1/1   Running  0         10m
infoscale-vtas-driver-container-rhel8-qqjkv 1/1   Running  0         10m
infoscale-vtas-driver-container-rhel8-ww8md 1/1   Running  0         10m
infoscale-vtas-fencing-controller-5dd876748d-rbbgn 1/1  Running 0  2m39s
infoscale-vtas-fencing-switcher-7tqwg     1/1   Running   0        2m49s
infoscale-vtas-fencing-switcher-chllt     1/1   Running   0        2m49s
infoscale-vtas-fencing-switcher-m5hp4     1/1   Running   0        2m49s
infoscale-vtas-fencing-switcher-wdcqw     1/1   Running   0        2m49s
infoscale-vtas-licensing-controller-7b749fb8d-xdwjn 1/1 Running 0  11m
infoscale-operator-75667df67b-vjm5p              1/1   Running   0
```

After a successful InfoScale deployment, a disk group is automatically created. You can now create Persistent Volumes/ Persistent Volume Claims (PV / PVC) by using the corresponding Storage class.

Adding nodes to an existing cluster

Complete the following steps to add nodes to an existing InfoScale cluster-

1. Ensure that you add the worker nodes to the OCP cluster.

   Note: You must add all OpenShift worker nodes to the InfoScale cluster.

2. Run the following command on the bastion node to check whether the newly added node is Ready.

   ```
   oc get nodes -A
   ```

   Review output similar to the following

   ```
   NAME                  STATUS ROLES  AGE VERSION
   ocp-cp-1.lab.ocp.lan  Ready  master 54d v1.22.1+d8c4430
   ocp-cp-2.lab.ocp.lan  Ready  master 54d v1.22.1+d8c4430
   ocp-cp-3.lab.ocp.lan  Ready  master 54d v1.22.1+d8c4430
   ocp-w-1.lab.ocp.lan   Ready  worker 54d v1.22.1+d8c4430
   ocp-w-2.lab.ocp.lan   Ready  worker 54d v1.22.1+d8c4430
   ocp-w-3.lab.ocp.lan   Ready  worker 54d v1.22.1+d8c4430
   ocp-w-4.lab.ocp.lan   Ready  worker 54d v1.22.1+d8c4430
   ```

3. To add new nodes to an existing cluster, the cluster must be in a running state. Run the following command on the bastion node to verify.

   ```
   oc get infoscalecluster
   ```

   See the State in the output similar to the following -

   ```
   NAME                 NAMESPACE      VERSION    STATE   AGE
   infoscalecluster-dev infoscale-vtas 8.0.0.0000 Running 1m15s
   ```

4 Edit **clusterInfo** section of the sample `/YAML/OpenShift/cr.yaml` to add information about the new nodes.

In this example, worker-node-1 and worker-node-2 exist. worker-node-3 is being added.

Note: The number of IP addresses must be same for all nodes.

```
apiVersion: infoscale.veritas.com/v1
kind: InfoScaleCluster
metadata:
name: infoscalecluster-dev

spec:
  version: "8.0.0.0000"

  clusterInfo:
  - nodeName: "worker-node-1"
    ip:
    - "<IP address of worker-node-1>"
  - nodeName: "worker-node-2"
    ip:
    - "<IP address of worker-node-2>"
  - nodeName: "worker-node-3"
    ip:
    - "<IP address of worker-node-3>"
    excludeDevice:
    - /dev/sdm
    - /dev/sdn

.
.

 YOU CAN ADD UP TO 16 NODES.
```

5 Run the following command on the bastion node to initiate add node workflow.

```
oc apply -f /YAML/OpenShift/cr.yaml
```

6 You can run the following commands on the bastion node when node addition is in progress.

a. `oc get infoscalecluster`

See the State in the output as under. ProcessingAddNode indicates node is getting added.

```
NAME                    NAMESPACE         VERSION      STATE
infoscalecluster-dev    infoscale-vtas    8.0.0.0000   ProcessingAddNode
```

b. `oc describe infoscalecluster -n infoscale-vtas`

Output similar to following indicates the cluster status during add node. The cluster is Degraded when node addition is in progress.

```
Cluster Name:  infoscalecluster-dev
  Cluster Nodes:
    Exclude Device:
      /dev/sdm
      /dev/sdn
    Node Name:  worker-node-1
    Role:       Joined,Master
    Node Name:  worker-node-2
    Role:       Joined,Slave
    Node Name:  worker-node-3
    Role:         Out of Cluster
   Cluster State:  Degraded
   enableScsi3pr:  false
   Images:
    Csi:
      Csi External Attacher Container:      csi-attacher:v3.1.0
```

7 Run the following command on the bastion node to verify if pods are created successfully. It may take some time for the pods to be created.

```
oc get pods -n infoscale-vtas
```

Output similar to the following indicates a successful creation.

```
NAME                                    READY STATUS RESTARTS AGE
infoscale-vtas-csi-driver-node-5tnct  2/2   Running 0       2m27s
infoscale-vtas-csi-driver-node-6w2q7  2/2   Running 0       2m27s
infoscale-vtas-csi-driver-node-lj4xz  2/2   Running 0       2m27s
infoscale-vtas-driver-container-rhel8
                                -7zcrk 1/1   Running 0       10m
infoscale-vtas-driver-container-rhel8
                                -f7h4f 1/1   Running 0       10m
infoscale-vtas-driver-container-rhel8
                                -qqjkv 1/1   Running 0       10m
infoscale-vtas-fencing-controller
                     -5dd876748d-rbbgn 1/1   Running 0       2m39s
infoscale-vtas-fencing-switcher-7tqwg 1/1   Running 0       2m49s
infoscale-vtas-fencing-switcher-chllt 1/1   Running 0       2m49s
infoscale-vtas-fencing-switcher-m5hp4 1/1   Running 0       2m49s
infoscale-vtas-licensing-controller
                      -7b749fb8d-xdwjn 1/1   Running 0       11m
infoscale-operator-75667df67b-vjm5p   1/1   Running 0       11m
```

8 Run the following command on the bastion node to verify if the cluster is 'Running'

```
oc get infoscalecluster
```

See the State in the output similar to the following -

```
NAME                  NAMESPACE      VERSION    STATE    AGE
infoscalecluster-dev infoscale-vtas 8.0.0.0000 Running 1m15s
```

9 Run the following command on the bastion node to verify whether the cluster is 'Healthy'.

```
oc describe infoscalecluster
```

Check the **Cluster State** in the output similar to the following-

```
Status:
  Cluster Name:  infoscalecluster-dev
  Cluster Nodes:
    Node Name:    worker-node-1
    Role:         Joined,Master
    Node Name:    worker-node-2
    Role:         Joined,Slave
    Node Name:    worker-node-3
    Role:         Joined,Slave
  Cluster State:  Healthy
```

Undeploying and uninstalling InfoScale

You can run the following command to undeploy InfoScale on your OpenShift cluster. Additionally, see Deleting Operators from a cluster to ensure a clean undeployment.

- Run the following commands on the bastion node

```
oc delete -f /YAML/OpenShift/cr.yaml
```

The commands to clean up InfoScale components like the Operator, SR, and SRO are as under

Note: Run these commands only after all InfoScale pods are terminated.

```
oc delete -f /YAML/OpenShift/iso.yaml
oc delete -f /YAML/OpenShift/sr.yaml
oc delete -f /YAML/OpenShift/sro.yaml
```

Note: After uninstallation, ensure that stale InfoScale kernel modules (vxio/vxdmp/veki/vxspec/vxfs/odm/glm/gms) do not remain loaded on any of the worker nodes. Rebooting a worker node deletes all such modules.

Installing InfoScale in an air gapped system

An air gapped system is not connected to the Internet. It is therefore necessary to prepare the system.

Before installing InfoScale on an air gapped system, mirror the Node Feature Discovery (NFD) operator catalog first. You can perform mirroring and installation of Node Feature Discovery (NFD) from any OpenShift cluster node that has Internet connectivity and is also connected with the air gapped system.

Note: In the following steps, ${JUMP_HOST}:5000 is on the same network. JUMP_HOST is a system connected to Internet and has a registry setup. 5000 is an indicative port number.

Mirroring the Node Feature Discovery (NFD) operator catalog

1. Run the following command on the bastion node to authenticate with registry.redhat.io and your custom registry.

   ```
   export REGISTRY_AUTH_FILE=<path_to_pull_secret>/pull-secret.json
   ```

2. Run the following command on the bastion node to set the following environment variable export

   ```
   JUMP_HOST="<IP address of custom registry>"
   ```

3. Run the following command on the bastion node to disable the sources for the default catalogs.

   ```
   oc patch OperatorHub cluster --type json -p '[{"op": "add",
   "path": "/spec/disableAllDefaultSources", "value": true}]'
   ```

4. Run the following command on the bastion node to retain only the specified package in the source index.

   ```
   nfd opm index prune -f
   registry.redhat.io/redhat/redhat-operator-index:v4.9 -p nfd -t
   ${JUMP_HOST}:5000/catalog/redhat-operator-index:v4.9
   ```

5. Run the following command on the bastion node to push the Node Feature Discovery Operator index image to your custom registry.

   ```
   podman push ${JUMP_HOST}:5000/catalog/redhat-operator-index:v4.9
   ```

6 Run the following command on the bastion node to mirror the Node Feature Discovery Operator

```
oc adm catalog mirror \ --insecure=true \
--index-filter-by-os='linux/amd64' \ -a ${REGISTRY_AUTH_FILE} \
${JUMP_HOST}:5000/catalog/redhat-operator-index:v4.9
${JUMP_HOST}:5000/operators
```

7 Inspect the manifests directory that is generated in your current directory. The manifest directory format is `manifests-<index_image_name>-<random_number>`. For example `manifests-redhat-operator-index-1638334101`.

8 Run the following command on the bastion node to create the ImageContentSourcePolicy (ICSP) object by specifying `imageContentSourcePolicy.yaml` in your manifests directory

```
oc create -f <path to the manifests directory for your mirrored
content>/imageContentSourcePolicy.yaml
```

9 Run the following command on the bastion node to customize `mapping.txt` with `REGISTRY_AUTH_FILE`.

```
oc image mirror -f <path/to/manifests/dir>/mapping.txt -a
${REGISTRY_AUTH_FILE} –insecure
```

10 Copy the following content and save it as `catalogSource_redhat_operator.yaml` .

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
   name: redhat-operator-index
   namespace: openshift-marketplace
spec:
  image: ${JUMP_HOST}:5000/operators/catalog-redhat-operator-index:v4.9
  sourceType: grpc
  displayName: My Operator Catalog
  publisher: <publisher_name>
  updateStrategy:
    registryPoll:
      interval: 30m
```

11 Run the following command on the bastion node to create the CatalogSource object

```
oc apply -f catalogSource_redhat_operator.yaml
```

12 Run the following command on the bastion node to check the status of pods.

```
oc get pods -n openshift-marketplace
```

Review output as under. Status of the pods must be 'Running'.

```
NAME                           READY   STATUS      RESTARTS      AGE
certified-operator-
              index-bq7bt      1/1     Running     0             17h
marketplace-operator-
 d6985d479bc-7zbckj            1/1     Running     0             23d
redhat-operator-index
           -785tv              1/1     Running     0             17h
```

13 Check the package manifest

```
oc get packagemanifest -n openshift-marketplace
```

Review output to the following output.

```
NAME                         DISPLAY              TYPE PUBLISHER      AGE
certified-operator-index   Openshift Telco Docs grpc Openshift Docs 20h
redhat-operator-index      Openshift Telco Docs grpc Openshift Docs 20h
```

14 Run the following commands on the bastion node to check the catalogsource

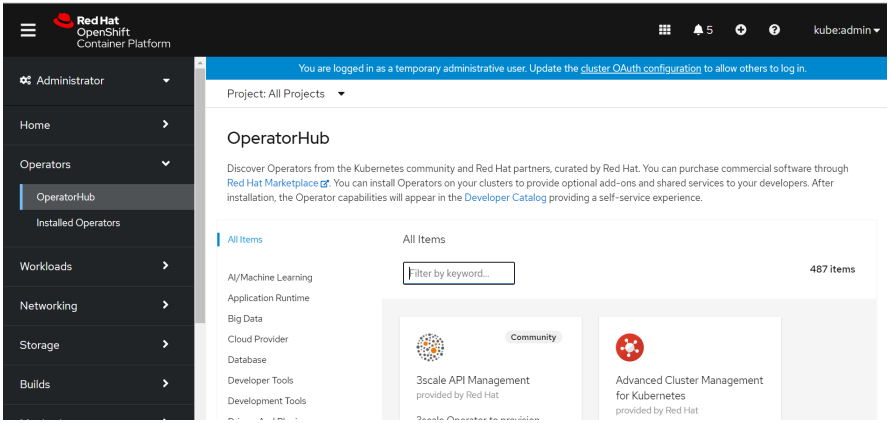
```
oc get catalogsource -n openshift-marketplace
```

```
oc get pods -n openshift-marketplace
```

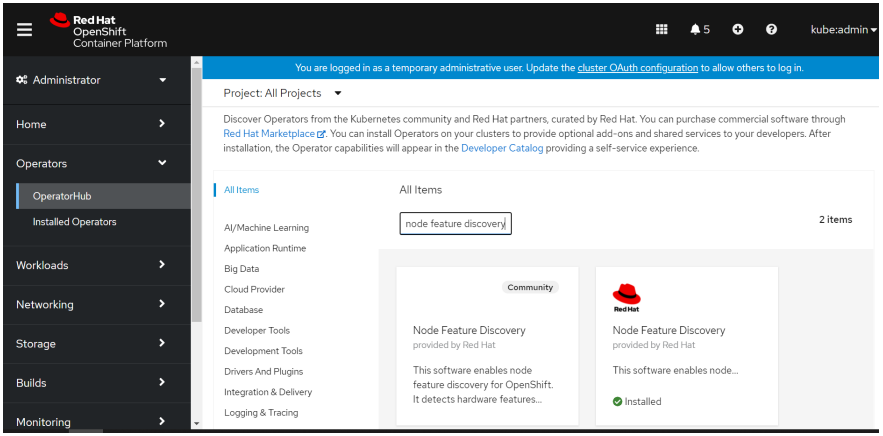
15 Login to the OCP web console and click **Operators > OperatorHub**. The mirrored operator must be listed here.

Installing Node Feature Discovery (NFD) Operator

1. Connect to the OpenShift console.
2. In the left frame, click **Operators > OperatorHub**. You can select and install the operator here.



3. In **Filter by keyword**, enter Node Feature Discovery. Node Feature Discovery is listed as under.



Note: If the Operator is already installed, it is indicated. See the last step to apply Cert-Manager.

4 Select the Node Feature Discovery Operator and follow onscreen instructions to install.

5 After a successful installation, Node Feature Discovery is listed under **Operators > Installed Operators** in the left frame.

6 In **Node Feature Discovery**, see a box under **Provided APIs**.

7 Click **Create instance**. Edit the values of the `NodeFeatureDiscovery CR`.

8 Click **Create**.

9 To verify whether the installation is successful and check status of NFD instances on each node, run the following command on the bastion node.

```
oc get pods -A |grep nfd
```

Review the sample output as under. Here, the prefix `nfd-` is of the nfd operator.

```
openshift-operators   nfd-master-4hqbq    1/1   Running     0   62m
openshift-operators   nfd-master-brt9f    1/1   Running     0   62m
openshift-operators   nfd-master-pplqr    1/1   Running     0   62m
openshift-operators   nfd-operator-59454bd5c9-gf6h7 1/1  Running 0 5d2h
openshift-operators   nfd-worker-8l6wh    1/1   Running     0   62m
openshift-operators   nfd-worker-bngbq    1/1   Running     0   62m
openshift-operators   nfd-worker-d5btm    1/1   Running     0   62m
openshift-operators   nfd-worker-hx6xl    1/1   Running     0   62m
```

Note: You can refer to the OpenShift documentation for Node Feature Discovery.

Installing cert-manager

1 Pull the following images

- quay.io/jetstack/cert-manager-cainjector:v1.6.1
- quay.io/jetstack/cert-manager-controller:v1.6.1
- quay.io/jetstack/cert-manager-webhook:v1.6.1

2 Tag and push the images to the Custom registry at `<IP address of custom registry>/veritas/`.

3 Edit `/YAML/OpenShift/air-gapped-systems/cert-manager.yaml` as under

- Replace `192.168.1.21/veritas/cert-manager-cainjector:v1.6.1` with `image: <IP address of custom registry>/veritas/cert-manager-cainjector:v1.6.1`.

- Replace `192.168.1.21/veritas/cert-manager-controller:v1.6.1` with `image: <IP address of custom registry>/veritas/cert-manager-controller:v1.6.1`.
- Replace `192.168.1.21/veritas/cert-manager-webhook:v1.6.1` with `image: <IP address of custom registry>/veritas/cert-manager-webhook:v1.6.1`.

4 Run the following command on the bastion node to install cert-manager

```
oc apply -f /YAML/OpenShift/air-gapped-systems/cert-manager.yaml
```

5 Run the following command on the bastion node to check the status of pods.

```
oc get all -n cert-manager
```

Status similar to the following indicates a successful installation.

```
NAME                                       READY STATUS    RESTARTS AGE
pod/cert-manager-5986867bb9-v95t7          1/1   Running   0        56s
pod/cert-manager-cainjector-b475c485b
                                  -bxj89   1/1   Running   0        56s
pod/cert-manager-webhook-55b6c54579
                                -95gcw     1/1   Running   0        56s


NAME                   TYPE      CLUSTER-IP   EXTERNAL-IP PORT(S)  AGE
service/cert-manager   ClusterIP 172.30.72.54 <none>      9402/TCP 57s
service/cert-manager
            -webhook   ClusterIP 172.30.180.10<none>      443/TCP  57s


NAME                              READY UP-TO-DATE AVAILABLE   AGE
deployment.apps/cert-manager      1/1   1          1           57s
deployment.apps/cert-manager
                  -cainjector     1/1   1          1           57s
deployment.apps/cert-manager
                    -webhook      1/1   1          1           57s


NAME                              DESIRED CURRENT READY AGE
replicaset.apps/cert-manager
                 -5986867bb9      1       1       1     56s
replicaset.apps/cert-manager
       -cainjector-b475c485b      1       1       1     56s
replicaset.apps/cert-manager
         -webhook-55b6c54579      1       1       1     56s
```

You must install a Special Resource Operator (SRO) first, before installing Veritas InfoScale. After the SRO is installed, the system is enabled for installing Veritas InfoScale.

Installing Special Resource Operator (SRO) and InfoScale Operator

1. Download `YAML.tar` from the Veritas Download Center.
2. Untar `YAML.tar`.

   After you untar `YAML.tar`, the folders `/YAML/OpenShift/`, `/YAML/OpenShift/air-gapped-systems` , `/YAML/DR`, and `/YAML/Kubernetes` are created. Each folder contains files required for installation.
3. On the bastion node -
   - Download

     ```
     registry.redhat.io/openshift4/special-resource-rhel8-operator:
           v4.9.0-202111161916.p0.gf6ed01a.assembly.stream
     ```

     , tag, and push it to custom registry as

     ```
     <IP address of custom registry>/special-resource-rhel8-operator:
            v4.9.0-202111161916.p0.gf6ed01a.assembly.stream
     ```

   - Download `registry.redhat.io/openshift4/ose-kube-rbac-proxy`, tag and push it to custom registry as `<IP address of custom registry>/ose-kube-rbac-proxy:v4.9`.
   - Edit `/YAML/OpenShift/air-gapped-systems/sro.yaml` as under Replace

     ```
     192.168.1.21/veritas/special-resource-rhel8-operator:
         v4.9.0-202111161916.p0.gf6ed01a.assembly.stream
     ```

     with

     ```
     image:<IP address of custom registry>/special-resource-rhel8-operator:
     v4.9.0-202111161916.p0.gf6ed01a.assembly.stream
     ```

     and
     Replace **image:192.168.1.21/veritas/ose-kube-rbac-proxy:v4.9** with **image:<IP address of custom registry>/ose-kube-rbac-proxy:v4.9**.
   - Run the following command

     ```
     oc create -f /YAML/OpenShift/air-gapped-systems/sro.yaml
     ```

- Run `oc create -f /YAML/OpenShift/air-gapped-systems/sr.yaml` to create Special Resource.

4 Run the following commands and review the output to verify whether SR creation and SRO installation is successful.

- `oc get pods -n openshift-special-resource-operator`
  Output similar to the following indicates a successful installation.

```
NAME                                      READY   STATUS    RESTARTS   AGE
special-resource-controller-manager-66c8fc64b5-9wv6l 1/1  Running 0
```

Note: The name in the output here is used in the following command.

- `oc logs special-resource-controller-manager-66c8fc64b5-9wv6l -n openshift-special-resource-operator -c manager`
  Output similar to the following indicates a successful installation.

```
<timestamp>    INFO    status
 RECONCILE SUCCESS: Reconcile
```

- `oc get SpecialResource`
  Output similar to the following indicates a successful installation.

```
NAME                          AGE
special-resource-preamble   2m24s
```

All information about the worker nodes must be added to the `cr.yaml` file. All worker nodes become part of InfoScale cluster after `cr.yaml` is applied. After you download and untar `YAML.tar`, all files required for installation are available.

Note: You must download images required for installation from the Red Hat registry and push those to the Custom registry.

Optionally, configure a new user - infoscale-admin, associated with a Role-based Access Control (RBAC) clusterrole defined in`infoscale-admin-role.yaml`, to deploy InfoScale and its dependent components. infoscale-admin as a user when configured has clusterwide access to only those resources needed to deploy InfoScale and its dependent components such as SRO/NFD/Cert Manager in the desired namespaces.

To provide a secure and isolated environment for InfoScale deployment and associated resources, the namespace associated with these resources must be

protected from access of all other users (except super user of the cluster), with appropriate RBAC implemented.

Run the following commands on the bastion node to create a new user - infoscale-admin and a new project and assign role or clusterrole to infoscale-admin. You must be logged in as a super user.

1. `oc new-project <New Project name>`

   A new project is created for InfoScale deployment.

2. `oc adm policy add-role-to-user admin infoscale-admin`

   Following output indicates that administrator privileges are assigned to the new user - infoscale-admin within the new project.

   ```
   clusterrole.rbac.authorization.k8s.io/admin added: "infoscale-admin"
   ```

3. `oc apply -f /YAML/OpenShift/air-gapped-systems/infoscale-admin-role.yaml`

   Following output indicates that a clusterrole is created.

   ```
   clusterrole.rbac.authorization.k8s.io/infoscale-admin-role created
   ```

4. `oc adm policy add-cluster-role-to-user infoscale-admin-role infoscale-admin`

   Following output indicates that clusterrole created is associated with infoscale-admin.

   ```
   clusterrole.rbac.authorization.k8s.io/infoscale-admin-role added:
             "infoscale-admin"
   ```

You must perform all installation activities by logging in as infoscale-admin.

Download the following images -

- registry.connect.redhat.com/veritas-technologies/infoscale-operator:8.0.0-rhel8
- registry.connect.redhat.com/veritas-technologies/infoscale-vxfen:2.0.0.0000-rhel8
- registry.connect.redhat.com/veritas-technologies/infoscale-csi-plugin:2.0.0.0000-rhel8
- registry.connect.redhat.com/veritas-technologies/infoscale-license:8.0.0.0000-rhel8
- registry.connect.redhat.com/veritas-technologies/infoscale-dr-operator:1.0.0.0000-rhel8
- registry.connect.redhat.com/veritas-technologies/infoscale-operator:8.0.0-rhel8

- registry.connect.redhat.com/veritas-technologies/infoscale:8.0.0.0000-rhel8.4-<kernel release version> where, kernel release version = `uname -r` output from worker node.
- registry.redhat.io/openshift4/ose-csi-driver-registrar:v4.3
- registry.redhat.io/openshift4/ose-csi-external-provisioner-rhel8:v4.7
- registry.redhat.io/openshift4/ose-csi-external-attacher:v4.7
- registry.redhat.io/openshift4/ose-csi-external-resizer-rhel8:v4.7
- registry.redhat.io/openshift4/ose-csi-external-snapshotter-rhel8:v4.7
- docker.io/kvaps/kube-fencing-switcher:v2.1.0
- docker.io/kube-fencing-controller:v2.1.0

After you download, tag the images, and push those to the Custom registry.

1. Edit `/YAML/OpenShift/air-gapped-systems/iso.yaml` as under

   Replace **image: 192.168.1.21/veritas/infoscale-operator:8.0.0-rhel8** with **image: <IP address of custom registry>/infoscale-operator:8.0.0-rhel8**.

2. Run the following command on the bastion node to install Veritas InfoScale.

   ```
   oc create -f /YAML/OpenShift/air-gapped-systems/iso.yaml
   ```

3. Run the following command on the bastion node to verify whether the installation is successful

   ```
   oc get pods -n infoscale-vtas|grep infoscale
   ```

   An output similar to the following indicates a successful installation. `READY 1/1` indicates that Storage cluster resources can be created.

   ```
   NAME                        READY   STATUS    RESTARTS    AGE
   infoscale-operator-6dc9bc8856-lh72f    1/1     Running   0        2d18h
   ```

Configuring cluster

After successfully installing InfoScale operator, you can create a cluster.

1. Edit **clusterInfo** section of the sample `/YAML/OpenShift/air-gapped-systems/cr.yaml` for InfoScale specifications as under -

Note: You can specify up to 16 worker nodes in `cr.yaml`. Although cluster configuration is allowed even with one Network Interface Card, Veritas recommends a minimum of two physical links for performance and High Availability (HA). Number of links for each network link must be same on all nodes. Optionally, you can enter node level IP addresses. If IP addresses are not provided, IP addresses of OpenShift cluster nodes are used.

```
  clusterInfo:
  - nodeName: <Name of the first node>
    ip:
    -  <Optional - First IP address of the first node >
    -  <Optional - Second IP address of the first node>
    excludeDevice:
    - <Optional - Device path of the disk on the node that you want
                                to exclude from Infoscale disk group.>
  - nodeName: <Name of the second node>
    ip:
   -  <Optional - First IP address of the second node >
   -  <Optional - Second IP address of the second node>
    excludeDevice:
    - <Optional - Device path of the disk on the node that you want
                                to exclude from Infoscale disk group.>
  - nodeName: <Name of the third node>
    ip:
   -  <Optional - First IP address of the third node >
   -  <Optional - Second IP address of the third node>
    excludeDevice:
    - <Optional - Device path of the disk on the node that you want
                                 to exclude from Infoscale disk group.>
.
.
.
  YOU CAN ADD UP TO 16 NODES.



  customImageRegistry: <Custom registry name as you
                               are downloading in an air gapped systems/
                     <IP address of the custom registry>:<port number> >
```

Note: Description of various `.yaml` parameters is in angle brackets (<>). While entering the parameter value, do not include the angle brackets. For example, Primarynode is the name of the first node; for **nodeName : <Name of the first node>** , enter **nodeName : Primarynode**.InfoScale on OpenShift is a keyless deployment.

2. Run the following command on the bastion node.

   ```
   oc create -f /YAML/OpenShift/air-gapped-systems/cr.yaml
   ```

3. Run the following command on the bastion node to know the name and namespace of the cluster.

   ```
   oc get infoscalecluster
   ```

   Use the namespace from the output similar to the following -

   ```
   NAME                    NAMESPACE        VERSION      STATE          AGE
   infoscalecluster-dev    infoscale-vtas   8.0.0.0000   Running   1m15s
   ```

4. Run the following command on the bastion node to verify whether the pods are created successfully.

   ```
   oc get pods -n infoscale-vtas
   ```

   An output similar to the following indicates a successful creation of nodes

   ```
   NAME                              READY STATUS   RESTARTS  AGE
   infoscale-vtas-csi-driver-node-5tnct      2/2   Running  0         2m27s
   infoscale-vtas-csi-driver-node-6w2q7      2/2   Running  0         2m27s
   infoscale-vtas-csi-driver-node-lj4xz      2/2   Running  0         2m27s
   infoscale-vtas-csi-driver-node-vzq7s      2/2   Running  0         2m27s
   infoscale-vtas-driver-container-rhel8-7zcrk 1/1   Running  0         10m
   infoscale-vtas-driver-container-rhel8-f7h4f 1/1   Running  0         10m
   infoscale-vtas-driver-container-rhel8-qqjkv 1/1   Running  0         10m
   infoscale-vtas-driver-container-rhel8-ww8md 1/1   Running  0         10m
   infoscale-vtas-fencing-controller-5dd876748d-rbbgn 1/1  Running 0  2m39s
   infoscale-vtas-fencing-switcher-7tqwg     1/1   Running   0        2m49s
   infoscale-vtas-fencing-switcher-chllt     1/1   Running   0        2m49s
   infoscale-vtas-fencing-switcher-m5hp4     1/1   Running   0        2m49s
   infoscale-vtas-fencing-switcher-wdcqw     1/1   Running   0        2m49s
   infoscale-vtas-licensing-controller-7b749fb8d-xdwjn 1/1 Running 0  11m
   infoscale-operator-75667df67b-vjm5p              1/1   Running   0
   ```

After a successful InfoScale deployment, a disk group is automatically created. You can now create Persistent Volumes/ Persistent Volume Claims (PV / PVC) by using the corresponding Storage class.

Adding nodes to an existing cluster

Complete the following steps to add nodes to an existing InfoScale cluster-

1. Ensure that you add the worker nodes to the OCP cluster.

   Note: You must add all OpenShift worker nodes to the InfoScale cluster.

2. Run the following command on the bastion node to check whether the newly added node is Ready.

   ```
   oc get nodes -A
   ```

   Review output similar to the following

   ```
   NAME                   STATUS ROLES  AGE VERSION
   ocp-cp-1.lab.ocp.lan   Ready  master 54d v1.22.1+d8c4430
   ocp-cp-2.lab.ocp.lan   Ready  master 54d v1.22.1+d8c4430
   ocp-cp-3.lab.ocp.lan   Ready  master 54d v1.22.1+d8c4430
   ocp-w-1.lab.ocp.lan    Ready  worker 54d v1.22.1+d8c4430
   ocp-w-2.lab.ocp.lan    Ready  worker 54d v1.22.1+d8c4430
   ocp-w-3.lab.ocp.lan    Ready  worker 54d v1.22.1+d8c4430
   ocp-w-4.lab.ocp.lan    Ready  worker 54d v1.22.1+d8c4430
   ```

3. Login to each worker node that you want to add and push images to the custom registry.
4. To add new nodes to an existing cluster, the cluster must be in a running state. Run the following command on the bastion node to verify.

   ```
   oc get infoscalecluster
   ```

   See the State in the output similar to the following -

   ```
   NAME                 NAMESPACE      VERSION    STATE   AGE
   infoscalecluster-dev infoscale-vtas 8.0.0.0000 Running 1m15s
   ```

5 Edit **clusterInfo** section of the sample `/YAML/OpenShift/air-gapped-systems/cr.yaml` to add information about the new nodes.

In this example, worker-node-1 and worker-node-2 exist. worker-node-3 is being added.

Note: The number of IP addresses must be same for all nodes.

```
apiVersion: infoscale.veritas.com/v1
kind: InfoScaleCluster
metadata:
name: infoscalecluster-dev

spec:
  version: "8.0.0.0000"

  clusterInfo:
  - nodeName: "worker-node-1"
    ip:
    - "<IP address of worker-node-1>"
  - nodeName: "worker-node-2"
    ip:
    - "<IP address of worker-node-2>"
  - nodeName: "worker-node-3"
    ip:
    - "<IP address of worker-node-3>"
    excludeDevice:
    - /dev/sdm
    - /dev/sdn

.
.
.
  YOU CAN ADD UP TO 16 NODES.

  customImageRegistry: <Custom registry name as you
                                are downloading in an air gapped systems/
                      <IP address of the custom registry>:<port number> >
```

6 Run the following command on the bastion node to initiate add node workflow.

```
oc apply -f /YAML/OpenShift/air-gapped-systems/cr.yaml
```

7 You can run the following commands on the bastion node when node addition is in progress.

a. `oc get infoscalecluster`

See the State in the output as under. ProcessingAddNode indicates node is getting added.

```
NAME                    NAMESPACE         VERSION      STATE
infoscalecluster-dev    infoscale-vtas    8.0.0.0000   ProcessingAddNode
```

b. `oc describe infoscalecluster -n infoscale-vtas`

Output similar to following indicates the cluster status during add node. The cluster is Degraded when node addition is in progress.

```
Cluster Name:  infoscalecluster-dev
  Cluster Nodes:
    Exclude Device:
      /dev/sdm
      /dev/sdn
    Node Name:  worker-node-1
    Role:       Joined,Master
    Node Name:  worker-node-2
    Role:       Joined,Slave
    Node Name:  worker-node-3
    Role:         Out of Cluster
   Cluster State:  Degraded
   enableScsi3pr:  false
   Images:
    Csi:
      Csi External Attacher Container:      csi-attacher:v3.1.0
```

8 Run the following command on the bastion node to verify if pods are created successfully. It may take some time for the pods to be created.

```
oc get pods -n infoscale-vtas
```

Output similar to the following indicates a successful creation.

```
NAME                                    READY STATUS RESTARTS AGE
infoscale-vtas-csi-driver-node-5tnct  2/2   Running 0       2m27s
infoscale-vtas-csi-driver-node-6w2q7  2/2   Running 0       2m27s
infoscale-vtas-csi-driver-node-lj4xz  2/2   Running 0       2m27s
infoscale-vtas-driver-container-rhel8
                                -7zcrk 1/1   Running 0       10m
infoscale-vtas-driver-container-rhel8
                                -f7h4f 1/1   Running 0       10m
infoscale-vtas-driver-container-rhel8
                                -qqjkv 1/1   Running 0       10m
infoscale-vtas-fencing-controller
                     -5dd876748d-rbbgn 1/1   Running 0       2m39s
infoscale-vtas-fencing-switcher-7tqwg 1/1   Running 0       2m49s
infoscale-vtas-fencing-switcher-chllt 1/1   Running 0       2m49s
infoscale-vtas-fencing-switcher-m5hp4 1/1   Running 0       2m49s
infoscale-vtas-licensing-controller
                      -7b749fb8d-xdwjn 1/1   Running 0       11m
infoscale-operator-75667df67b-vjm5p   1/1   Running 0       11m
```

9 Run the following command on the bastion node to verify if the cluster is 'Running'

```
oc get infoscalecluster
```

See the State in the output similar to the following -

```
NAME                  NAMESPACE      VERSION    STATE    AGE
infoscalecluster-dev infoscale-vtas 8.0.0.0000 Running 1m15s
```

10 Run the following command on the bastion node to verify whether the cluster is 'Healthy'.

```
oc describe infoscalecluster
```

Check the **Cluster State** in the output similar to the following-

```
Status:
  Cluster Name:  infoscalecluster-dev
  Cluster Nodes:
    Node Name:    worker-node-1
    Role:         Joined,Master
    Node Name:    worker-node-2
    Role:         Joined,Slave
    Node Name:    worker-node-3
    Role:         Joined,Slave
  Cluster State:  Healthy
```

Undeploying and uninstalling InfoScale

You can run the following command to undeploy InfoScale on your OpenShift cluster. Additionally, see Deleting Operators from a cluster to ensure a clean undeployment.

- Run the following command on the bastion node

  ```
  oc delete -f /YAML/OpenShift/air-gapped-systems/cr.yaml
  ```

  The commands to clean up InfoScale components like the Operator, SR, and SRO are as under

  Note: Run these commands only after all InfoScale pods are terminated.

  ```
  oc delete -f /YAML/OpenShift/air-gapped-systems/iso.yaml
  oc delete -f /YAML/OpenShift/air-gapped-systems/sr.yaml
  oc delete -f /YAML/OpenShift/air-gapped-systems/sro.yaml
  ```

Note: After uninstallation, ensure that stale InfoScale kernel modules (`vxio`/`vxdmp`/`veki`/`vxspec`/`vxfs`/`odm`/`glm`/`gms`) do not remain loaded on any of the worker nodes. Rebooting a worker node deletes all such modules.

Chapter 5

Installing Veritas InfoScale on Kubernetes

This chapter includes the following topics:

Introduction

This chapter informs you how to install InfoScale on a Kubernetes cluster.

On Kubernetes systems, installer files and container images must be downloaded from the Veritas Download Center. Commands are run from the master node of a Kubernetes cluster.

Note: As InfoScale supports HyperConverged architecture, all worker nodes that are a part of Kubernetes cluster must be used for creating an InfoScale cluster. Veritas InfoScale is deployed on all the worker nodes you specify in the Custom Resource yaml file.

Prerequisites

1. Be ready with the following information -
   - Names of all the nodes.

     Note: Run `kubectl get nodes -o wide` on the master node to obtain Names and IP addresses of the nodes.

     Use `NAME` and `INTERNAL-IP` from the output similar to the following -

     ```
     NAME      STATUS ROLES  AGE VERSION         INTERNAL-IP
     k8s-cp-1.lab.k8s.lan Ready  master 75d v1.20.0+558d959 192.168.22.201
     k8s-cp-2.lab.k8s.lan Ready  master 75d v1.20.0+558d959 192.168.22.202
     k8s-cp-3.lab.k8s.lan Ready  master 75d v1.20.0+558d959 192.168.22.203
     k8s-w-1.lab.k8s.lan  Ready  worker 75d v1.20.0+558d959 192.168.22.211
     ```

   - Operating system device path of the disks which are being managed by other storage vendors that need to be excluded from InfoScale disk group.
   - Optionally if you want to exclude boot disks, device path to the boot disks.

     Note: Veritas recommends excluding boot disks.

   - Custom Registry address to set up registry where InfoScale images are pushed.
2. Ensure that all nodes are synchronized with the NTP Server.
3. Reserve network ports for exclusive use of InfoScale as under -

Component	Port
LLT over UDP	Serially onwards 50000 (as many as configured LLT links)
VVR (Needed only if you want to configure DR)	4145 (UDP), 8199 (TCP), 8989 (TCP)

4. Add local or shared storage to all the worker nodes before you proceed with the deployment.
5. Ensure that stale InfoScale kernel modules (`vxio`/`vxdmp`/`veki`/`vxspec`/`vxfs`/`odm`/`glm`/`gms`) from previous installation do not exist on any of the worker nodes.

Note: You can reboot a worker node to unload all stale InfoScale kernel modules.

Installing Node Feature Discovery (NFD) Operator and Cert-Manager on Kubernetes

Complete the following steps to enable Node Feature Discovery 0.8.2 and Cert-Manager 1.6.1 on Kubernetes.

1. Run the following command on the master node to install Cert-Manager:

   ```
   kubectl apply -f
   https://github.com/jetstack/cert-manager/releases/download/v1.6.1/cert-manager.yaml
   ```

2. Run the following commands on the master node to install Node Feature Discovery (NFD) Operator:

   ```
   kubectl apply -f
   https://raw.githubusercontent.com/kubernetes-sigs/node-feature-discovery/v0.8.2/nfd-master.yaml.template

   kubectl apply -f
   https://raw.githubusercontent.com/kubernetes-sigs/node-feature-discovery/v0.8.2/nfd-worker-daemonset.yaml.template
   ```

Note: Refer to the Kubernetes documentation for more information about Node Feature Discovery.

Installing the Special Resource Operator

You must install a Special Resource Operator (SRO) first, before installing Veritas InfoScale. After the SRO is installed on the system, InfoScale can be deployed.

1. Download `YAML.tar` from the Veritas Download Center.
2. Untar `YAML.tar`.

   After you untar `YAML.tar`, the folders `/YAML/OpenShift/`, `/YAML/DR`, and `/YAML/Kubernetes` are created. Each folder contains files required for installation.
3. On a Kubernetes cluster -
   - Run `kubectl create -f /YAML/Kubernetes/sro.yaml` on the master node to install the Special Resource Operator (SRO) .
   - Run `kubectl create -f /YAML/Kubernetes/sr.yaml` on the master node to create Special Resource.

4. Run the following commands and review the output to verify whether SR creation and SRO installation is successful.

   - `kubectl get SpecialResource`

     Output similar to the following indicates a successful installation.

     ```
     NAME                          AGE
     special-resource-preamble   2m24s
     ```

   - `kubectl get pods -n openshift-special-resource-operator`

     Output similar to the following indicates a successful installation.

     ```
     NAME                                    READY  STATUS    RESTARTS   AGE
     special-resource-controller-manager-66c8fc64b5-9wv6l 1/1  Running 0
     ```

     Note: The name in the output here is used in the following command.

   - `kubectl logs special-resource-controller-manager-66c8fc64b5-9wv6l -n openshift-special-resource-operator -c manager`

     Output similar to the following indicates a successful installation.

     ```
     <timestamp>    INFO    status
      RECONCILE SUCCESS: Reconcile
     ```

Tagging the InfoScale images on Kubernetes

Complete the following steps to upload and tag InfoScale images in the private registry/repository and prepare the cluster for installing InfoScale.

Prerequisites

- You must have a docker registry or you must set up a new docker registry.
- If the registry is insecure, your Kubernetes nodes must be configured to access the registry by using `http` method.

Complete the following steps

- Download `Veritas_InfoScale_8.0_Containers_Oracle_<OS>_Linux.tar` and `setup_vtas_registry.sh` from the Veritas Download Center to a node where the private repository is configured.

Note: `setup_vtas_registry.sh` is available in `tools.tar`.

- Run the following command on this node

  ```
  setup_vtas_registry.sh -c <IP address of custom registry>:<port number>/vtas_test -t Veritas_InfoScale_8.0_Containers_Oracle_<OS>_Linux.tar
  ```

After this command is successfully run, you have the ISO image as `<IP address of custom registry>:<port number>/vtas_test/infoscale-operator:1.0.0.0000-ol8` and CR custom registry as `<IP address of custom registry>:<port number>/vtas_test`.

Alternatively, you can download `Veritas_InfoScale_8.0_Containers_Oracle_<OS>_Linux.tar` and individually load, tag, and push each image file. See the following steps.

1. Download `Veritas_InfoScale_8.0_Containers_Oracle_<OS>_Linux.tar` and run the following command to extract the tar file.

   ```
   tar -xvf Veritas_InfoScale_8.0_Containers_Oracle_<OS>_Linux.tar
   ```

2. After you untar `Veritas_InfoScale_8.0_Containers_Oracle_<OS>_Linux.tar`, you get the following image files

   - `infoscale-operator-1.0.0.0000-<os-version>.img`
   - `infoscale-license-8.0.0.0000-<os-version-major>.img`
   - `infoscale-8.0.0.0000-<os-version>.img`
   - `infoscale-vxfen-2.0.0.0000-<os-version-major>.img`
   - `infoscale-csi-plugin-2.0.0.0000-<os-version-major>.img`

   You must load, tag, and push each image file into the custom registry.

   Note: The following commands are applicable to docker as the runtime environment. These commands change as per your container runtime environment. Refer to the documentation for the equivalent commands.

3. For `infoscale-operator-1.0.0.0000-<os-version>.img`, run the following commands-

   - `docker load -i infoscale-operator-1.0.0.0000-<os-version>.img`

- `docker tag localhost/veritas/infoscale-operator:1.0.0.0000-<os-version> <IP address of custom registry>:<port number>/infoscale-operator:1.0.0.0000-<os-version>`
- `docker push <IP address of custom registry>:<port number>/infoscale-operator:1.0.0.0000-<os-version>`

4. For `infoscale-license-8.0.0.0000-<os-version-major>.img`, run the following commands-
   - `docker load -i infoscale-license-8.0.0.0000-<os-version-major>.img`
   - `docker tag localhost/infoscale-license:8.0.0.0000-<os-version-major> <IP address of custom registry>:<port number>/infoscale-license-8.0.0.0000-<os-version-major>`
   - `docker push <IP address of custom registry>:<port number>/infoscale-license-8.0.0.0000-<os-version-major>`
5. For `infoscale-8.0.0.0000-<os-version>.img` , run the following commands

Note: You must be ready with the kernel version of the Operating system on each worker node. You can run `uname -r` to know the kernel version. If worker nodes have different kernel versions, you must run the following commands separately for worker nodes with identical kernel versions.

   - `docker load -i infoscale-8.0.0.0000-<os-version>.img`
   - `docker tag localhost/infoscale:8.0.0.0000-<os-version>-<kernel-version> <IP address of custom registry>:<port number>/infoscale:8.0.0.0000-<os-version>-<kernel-version>`
   - `docker push <IP address of custom registry>:<port number>/infoscale:8.0.0.0000-<os-version>-<kernel-version>`
6. For `infoscale-vxfen-2.0.0.0000-<os-version-major>.img`, run the following commands -
   - `docker load -i infoscale-vxfen-2.0.0.0000-<os-version-major>.img`
   - `docker tag localhost/veritas/infoscale-vxfen:2.0.0.0000-<os-version-major>`

```
<IP address of custom registry>:<port
number>/infoscale-vxfen:2.0.0.0000-<os-version-major>
```

- `docker push <IP address of custom registry>:<port number>/infoscale-vxfen:2.0.0.0000-<os-version-major>`

7. For `infoscale-csi-plugin-2.0.0.0000-<os-version-major>.img`, run the following commands -

   - `docker load -i infoscale-csi-plugin-2.0.0.0000-<os-version-major>.img`
   - `docker tag localhost/veritas/infoscale-csi-plugin:2.0.0.0000-<os-version-major> <IP address of custom registry>:<port number>/infoscale-csi-plugin:2.0.0.0000-<os-version-major>`
   - `docker push localhost/veritas/infoscale-csi-plugin:2.0.0.0000-<os-version-major> <IP address of custom registry>:<port number>/infoscale-csi-plugin:2.0.0.0000-<os-version-major>`

Downloading side car images

Following table lists the side car images for CSI plugin and fencing containers. You must download the CSI plugin-related images from https://console.cloud.google.com/gcr/images/k8s-artifacts-prod/asia/sig-storage and the fencing-related images from https://hub.docker.com/r/kvaps/kube-fencing-agents

Note: Perform these steps if you are manually tagging images instead of using `setup_vtas_registry.sh` for tagging images.

Table 5-1 Image names and sources for side car containers

Image name	Source
csi-snapshotter	k8s.gcr.io/sig-storage/csi-snapshotter
csi-provisioner	k8s.gcr.io/sig-storage/csi-provisioner
csi-resizer	k8s.gcr.io/sig-storage/csi-resizer
csi-node-driver-registrar	k8s.gcr.io/sig-storage/csi-node-driver-registrar
csi-attacher	k8s.gcr.io/sig-storage/csi-attacher

Table 5-1 Image names and sources for side car containers *(continued)*

Image name	Source
kube-fencing-switcher	docker.io/kvaps/kube-fencing-switcher
kube-fencing-controller	docker.io/kvaps/kube-fencing-controller

Table 5-2 Image names with tags for side car containers

Image name	Tag	Required tag
csi-snapshotter	v2.1.4	<IP address of custom registry>:<port number>/csi-snapshotter:v2.1.4
csi-provisioner	v2.1.0	<IP address of custom registry>:<port number>/csi-provisioner:v2.1.0
csi-resizer	v1.1.0	<IP address of custom registry>:<port number>/csi-resizer:v1.1.0
csi-node-driver-registrar	v2.1.0	<IP address of custom registry>:<port number>/csi-node-driver-registrar:v2.1.0
csi-attacher	v3.1.0	<IP address of custom registry>:<port number>/csi-attacher:v3.1.0
kube-fencing-switcher	v2.1.0	<IP address of custom registry>:<port number>/kube-fencing-switcher:v2.1.0
kube-fencing-controller	v2.1.0	<IP address of custom registry>:<port number>/kube-fencing-controller:v.2.1.0

You must pull these images, tag correctly, and subsequently push the tagged images to the custom registry. The commands are in the following format.

- Pulling the image

  ```
  docker pull <source from the above table>:<tag from the above table>
  ```

- Correctly tagging the image

  ```
  docker tag <source from the above table>:<tag from the above table> <Required tag from the above table>
  ```

- Pushing the image

  ```
  docker push <Required tag from the above table>
  ```

An example of commands for csi-snapshotter is as under -

- `docker pull k8s.gcr.io/sig-storage/csi-snapshotter:v2.1.4`

- `docker tag k8s.gcr.io/sig-storage/csi-snapshotter:v2.1.4 <IP address of custom registry>:<port number>/csi-snapshotter:v2.1.4`
- `docker push <IP address of custom registry>:<port number>/csi-snapshotter:v2.1.4`

Run these commands on all the images.

Installing InfoScale on Kubernetes

All information about the worker nodes must be added to the `cr.yaml` file. All worker nodes become part of InfoScale cluster after `cr.yaml` is applied. After you download and untar `YAML.tar`, all files required for installation are available.

Note: You must download images required for installation from the Veritas Download Center and push those to the Custom registry.

Configure a new user - infoscale-admin, associated with a Role-based Access Control (RBAC) clusterrole defined in`infoscale-admin-role.yaml`, to deploy InfoScale and its dependent components. infoscale-admin as a user when configured has clusterwide access to only those resources needed to deploy InfoScale and its dependent components such as SRO/NFD/Cert Manager in the desired namespaces.

To provide a secure and isolated environment for InfoScale deployment and associated resources, the namespace associated with these resources must be protected from access of all other users (except super user of the cluster), with appropriate RBAC implemented.

Run the following commands on the master node to create a new user - infoscale-admin and a new project and assign role or clusterrole to infoscale-admin. You must be logged in as a super user.

1 `kubectl create ns <New Project name>`

```
namespace/<New Project name>  created
```

indicates that a new project is created.

2 `kubectl create rolebinding infoscale-admin --namespace=<New Project name> --clusterrole=admin --user=infoscale-admin`

Following output indicates that administrator privileges are assigned to infoscale-admin within the new project.

```
rolebinding.rbac.authorization.k8s.io/infoscale-admin created
```

3 `kubectl apply -f /YAML/Kubernetes/infoscale-admin-role.yaml`

Following output indicates that a clusterrolebinding is created.

```
clusterrole.rbac.authorization.k8s.io/infoscale-admin-role created
```

4 `kubectl create clusterrolebinding infoscale-admin-role --clusterrole=infoscale-admin-role --user=infoscale-admin`

Following output indicates that a clusterrole created is associated with infoscale-admin by using a specified ClusterRoleBinding.

```
clusterrolebinding.rbac.authorization.k8s.io/infoscale-admin-role created
```

You must perform all installation-related activities by logging in as infoscale-admin. A cluster super-user can also install InfoScale.

1. Edit `/YAML/Kubernetes/iso.yaml` as under -

   Replace **image: 192.168.1.21/veritas/infoscale-operator:8.0.0-ol8** with **image: <IP address of custom registry>/infoscale-operator:8.0.0-ol8**.

2. Run the following command on the master node to install Veritas InfoScale.

   ```
   kubectl create -f /YAML/Kubernetes/iso.yaml
   ```

3. Run the following command on the master node to verify whether the installation is successful

   ```
   kubectl get pods -n infoscale-vtas|grep infoscale-operator
   ```

   An output similar to the following indicates a successful installation. `READY 1/1` indicates that Storage cluster resources can be created.

   ```
   NAME                    READY   STATUS    RESTARTS    AGE
   infoscale-operator-6dc9bc8856-lh72f   1/1     Running   0        2d18h
   ```

Configuring cluster

After successfully installing InfoScale operator, you can create a cluster.

1. Edit **clusterInfo** section of the sample `/YAML/Kubernetes/cr.yaml` for InfoScale specifications as under -

Note: You can specify up to 16 worker nodes in `cr.yaml`. Although cluster configuration is allowed even with one Network Interface Card, Veritas recommends a minimum of two physical links for performance and High Availability (HA). Number of links for each network link must be same on all nodes. Optionally, you can enter node level IP addresses. If IP addresses are not provided, IP addresses of Kubernetes cluster nodes are used.

```
clusterInfo:
- nodeName: <Name of the first node>
  ip:
  -  <Optional - First IP address of the first node >
  -  <Optional - Second IP address of the first node>
  excludeDevice:
  - <Optional - Device path of the disk on the node that you want
                              to exclude from Infoscale disk group.>
- nodeName: <Name of the second node>
  ip:
 -  <Optional - First IP address of the second node >
 -  <Optional - Second IP address of the second node>
  excludeDevice:
  - <Optional - Device path of the disk on the node that you want
                              to exclude from Infoscale disk group.>
- nodeName: <Name of the third node>
  ip:
 -  <Optional - First IP address of the third node >
 -  <Optional - Second IP address of the third node>
  excludeDevice:
  - <Optional - Device path of the disk on the node that you want
                               to exclude from Infoscale disk group.>
.
.
.
YOU CAN ADD UP TO 16 NODES.



customImageRegistry:  customImageRegistry: <Custom registry name /
                    <IP address of the custom registry>:<port number> >
```

Note: Do not enclose parameter values in angle brackets (<>). For example, Primarynode is the name of the first node; for **nodeName : <Name of the first node>** , enter **nodeName : Primarynode**. InfoScale on Kubernetes is a keyless deployment.

2. Run the following command on the master node.

```
kubectl create -f /YAML/Kubernetes/cr.yaml
```

3. Run the following command on the master node to know the name and namespace of the cluster.

```
kubectl get infoscalecluster
```

Use the namespace from the output similar to the following -

```
NAME                    NAMESPACE         VERSION      STATE          AGE
infoscalecluster-dev    infoscale-vtas    8.0.0.0000   Running    1m15s
```

4. Run the following command on the master node to verify whether the pods are created successfully.

```
kubectl get pods -n infoscale-vtas
```

An output similar to the following indicates a successful creation of nodes

```
NAME                                  READY  STATUS   RESTARTS AGE
infoscale-operator-665fcb664b-7cv8m   1/1    Running  0        22h
infoscale-vtas-csi-driver-controller-0 5/5   Running  0        22h
infoscale-vtas-csi-driver-node-75gz7  2/2    Running  0        22h
infoscale-vtas-csi-driver-node-86gp9  2/2    Running  0        22h
infoscale-vtas-csi-driver-node-8mtvn  2/2    Running  0        22h
infoscale-vtas-csi-driver-node-dvvh8  2/2    Running  0        22h
infoscale-vtas-csi-driver-node-vhdh2  2/2    Running  0        22h
infoscale-vtas-csi-driver-node-xk26c  2/2    Running  0        22h
infoscale-vtas-csi-driver-node-xxgml  2/2    Running  0        22h
infoscale-vtas-driver-container-ol8
             -396f682197e94c38-6nvjj  1/1    Running  0        22h
infoscale-vtas-driver-container-ol8
             -396f682197e94c38-87nsd  1/1    Running  0        22h
infoscale-vtas-driver-container-ol8
             -396f682197e94c38-b5fl8  1/1    Running  0        22h
infoscale-vtas-driver-container-ol8
             -396f682197e94c38-d6zvd  1/1    Running  0        22h
```

```
infoscale-vtas-driver-container-ol8
             -396f682197e94c38-hbmkl   1/1   Running 0        22h
infoscale-vtas-driver-container-ol8
             -396f682197e94c38-hv44n   1/1   Running 0        22h
infoscale-vtas-driver-container-ol8
             -396f682197e94c38-nnftq   1/1   Running 0        22h
infoscale-vtas-fencing-controller
                     -6bdb97fc88-2hkst 1/1   Running 0        22h
infoscale-vtas-fencing-switcher-42tl6  1/1   Running 0        22h
infoscale-vtas-fencing-switcher-7qcsw  1/1   Running 0        22h
infoscale-vtas-fencing-switcher-9rqxj  1/1   Running 0        22h
infoscale-vtas-fencing-switcher-mjrs9  1/1   Running 0        22h
infoscale-vtas-fencing-switcher-qc6m6  1/1   Running 0        22h
infoscale-vtas-fencing-switcher-qv2mk  1/1   Running 0        22h
infoscale-vtas-fencing-switcher-z75qz  1/1   Running 0        22h
infoscale-vtas-licensing-controller
           -7b4c5c664b-9h2lq           1/1   Running 0        22h
```

After a successful InfoScale deployment, a disk group is automatically created. You can now create Persistent Volumes/ Persistent Volume Claims (PV / PVC) by using the corresponding Storage class.

Adding nodes to an existing cluster

Complete the following steps to add nodes to an existing InfoScale cluster-

1 Ensure that you add the worker nodes to the Kubernetes cluster.

Note: You must add all Kubernetes worker nodes to the InfoScale cluster.

2 Run the following command on the master node to check whether the newly added node is Ready.

```
kubectl get nodes -A
```

Review output similar to the following

```
NAME           STATUS  ROLES         AGE  VERSION
worker-node-1  Ready   control-plane,
                              master 222d v1.21.0
worker-node-2  Ready   worker        222d v1.21.0
worker-node-3  Ready   worker        222d v1.21.0
```

3 To add new nodes to an existing cluster, the cluster must be in a running state. Run the following command on the master node to verify.

```
kubectl get infoscalecluster
```

See the State in the output similar to the following -

```
NAME                 NAMESPACE      VERSION    STATE   AGE
infoscalecluster-dev infoscale-vtas 8.0.0.0000 Running 1m15s
```

4 Edit **clusterInfo** section of the sample `/YAML/Kubernetes/cr.yaml` to add information about the new nodes.

In this example, worker-node-1 and worker-node-2 exist. worker-node-3 is being added.

Note: If you specify IP addresses, the number of IP addresses for the new nodes must be same as the number of IP addresses for the existing nodes.

```
apiVersion: infoscale.veritas.com/v1
kind: InfoScaleCluster
metadata:
name: infoscalecluster-dev

spec:
  version: "8.0.0.0000"

  clusterInfo:
  - nodeName: "worker-node-1"
    ip:
    - "<IP address of worker-node-1>"
  - nodeName: "worker-node-2"
    ip:
    - "<IP address of worker-node-2>"
  - nodeName: "worker-node-3"
    ip:
    - "<IP address of worker-node-3>"
    excludeDevice:
    - /dev/sdm
    - /dev/sdn

 .
 .
 .
  YOU CAN ADD UP TO 16 NODES.

  customImageRegistry: <Custom registry name /
                        <IP address of the custom registry>:<port number>
```

5 Run the following command on the master node to initiate add node workflow.

```
kubectl apply -f /YAML/Kubernetes/cr.yaml
```

6 You can run the following commands on the master node when node addition is in progress.

a. `kubectl get infoscalecluster`

See the State in the output as under. ProcessingAddNode indicates node is getting added.

```
NAME                    NAMESPACE         VERSION      STATE                
infoscalecluster-dev    infoscale-vtas    8.0.0.0000   ProcessingAddNode
```

b. `kubectl describe infoscalecluster -n infoscale-vtas`

Output similar to following indicates the cluster status during add node. The cluster is Degraded when node addition is in progress.

```
Cluster Name:  infoscalecluster-dev
  Cluster Nodes:
    Exclude Device:
      /dev/sdm
      /dev/sdn
    Node Name:  worker-node-1
    Role:       Joined,Master
    Node Name:  worker-node-2
    Role:       Joined,Slave
    Node Name:  worker-node-3
    Role:         Out of Cluster
   Cluster State:  Degraded
   enableScsi3pr:  false
   Images:
    Csi:
      Csi External Attacher Container:      csi-attacher:v3.1.0
```

7 Run the following command on the master node to verify if pods are created successfully. It may take some time for the pods to be created.

```
kubectl get pods -n infoscale-vtas
```

Output similar to the following indicates a successful creation.

```
NAME                                    READY STATUS RESTARTS AGE
infoscale-vtas-csi-driver-node-5tnct  2/2   Running 0       2m27s
infoscale-vtas-csi-driver-node-6w2q7  2/2   Running 0       2m27s
infoscale-vtas-csi-driver-node-lj4xz  2/2   Running 0       2m27s
infoscale-vtas-driver-container-rhel8
                                -7zcrk 1/1   Running 0       10m
infoscale-vtas-driver-container-rhel8
                                -f7h4f 1/1   Running 0       10m
infoscale-vtas-driver-container-rhel8
                                -qqjkv 1/1   Running 0       10m
infoscale-vtas-fencing-controller
                     -5dd876748d-rbbgn 1/1   Running 0       2m39s
infoscale-vtas-fencing-switcher-7tqwg 1/1   Running 0       2m49s
infoscale-vtas-fencing-switcher-chllt 1/1   Running 0       2m49s
infoscale-vtas-fencing-switcher-m5hp4 1/1   Running 0       2m49s
infoscale-vtas-licensing-controller
                      -7b749fb8d-xdwjn 1/1   Running 0       11m
infoscale-operator-75667df67b-vjm5p   1/1   Running 0       11m
```

8 Run the following command on the master node to verify if the cluster is 'Running'

```
kubectl get infoscalecluster
```

See the State in the output similar to the following -

```
NAME                  NAMESPACE      VERSION    STATE   AGE
infoscalecluster-dev infoscale-vtas 8.0.0.0000 Running 1m15s
```

9 Run the following command on the master node to verify whether the cluster is 'Healthy'.

```
kubectl describe infoscalecluster
```

Check the **Cluster State** in the output similar to the following-

```
Status:
  Cluster Name:  infoscalecluster-dev
  Cluster Nodes:
    Node Name:    worker-node-1
    Role:         Joined,Master
    Node Name:    worker-node-2
    Role:         Joined,Slave
    Node Name:    worker-node-3
    Role:         Joined,Slave
  Cluster State:  Healthy
```

Undeploying and uninstalling InfoScale

You can run the following command to undeploy and uninstall InfoScale on your Kubernetes cluster.

```
kubectl delete -f /YAML/Kubernetes/cr.yaml
```

The commands to clean up InfoScale components like the Operator, SR, and SRO are as under

Note: Run these commands only after all InfoScale pods are terminated.

```
kubectl delete -f /YAML/Kubernetes/iso.yaml

kubectl delete -f /YAML/Kubernetes/sr.yaml

kubectl delete -f /YAML/Kubernetes/sro.yaml
```

Note: After uninstallation, ensure that stale InfoScale kernel modules (`vxio`/`vxdmp`/`veki`/`vxspec`/`vxfs`/`odm`/`glm`/`gms`) do not remain loaded on any of the worker nodes. Rebooting a worker node deletes all such modules.

Chapter 6

InfoScale CSI deployment in Container environment

This chapter includes the following topics:

CSI plugin deployment

CSI is a standardized mechanism for Container Orchestrators (COs) to expose arbitrary storage systems to their containerized workloads. InfoScale CSI plugin is used to provide persistent storage to OpenShift or Kubernetes. InfoScale CSI also supports creation of storage classes for high availability, performance, and capacity. It also supports online expansion of capacity as well as snapshot and clone functionality.

InfoScale CSI is automatically deployed while installing InfoScale on OpenShift or Kubernetes.

After you download and untar `YAML.tar`, a folder `/YAML/Common-CSI-yamls` is automatically created. Within `/YAML/Common-CSI-yamls`, following sub folders are created and the files listed are saved.

Note: The commands listed in this chapter are applicable to OpenShift. If you are on Kubernetes, replace `oc` with `kubectl`.

- `-- dynamic-provisioning`
  - `-- csi-dynamic-pvc.yaml`
  - `-- csi-dynamic-snapshot-restore.yaml`
  - `-- csi-dynamic-snapshot.yaml`
  - `-- csi-dynamic-volume-clone.yaml`
  - `-- csi-pod.yaml`
- `-- snapshot-class-templates`
  - `-- csi-infoscale-snapclass.yaml`
- `-- static-provisioning`
  - `-- csi-pod.yaml`
  - `-- csi-static-pvc.yaml`
  - `-- csi-static-pv.yaml`
  - `-- csi-static-snapshot-content.yaml`
  - `-- csi-static-snapshot.yaml`
- `-- storage-class-templates`
  - `-- csi-infoscale-performance-sc.yaml`
  - `-- csi-infoscale-resiliency-sc.yaml`
  - `-- csi-infoscale-sc.yaml`

After CSI deployment is complete, you can create YAML files specific to your requirements and use these for:

- Dynamic provisioning of volumes
- Static provisioning of volumes
- Snapshot provisioning (Creating volume snapshots)

- Creating volume clones

InfoScale CSI supports static and dynamic provisioning of volumes on shared storage as well as shared nothing storage (FSS).

Note: Only one disk group - `vrts_kube_dg` is supported for all CSI operations, and the same disk group is used throughout the CSI plugin lifecycle. The command examples are applicable to OpenShift. For Kubernetes, replace `oc` by `kubectl`. `vrts_kube_dg` is created automatically during cluster creation by using disks which are not under any other File System or Logical Volume Manager.

An application container requests for the required storage through a Persistent Volume claim (PVC). The PVC uses the storage class to identify and provision the Persistent Volume that belongs to the storage class. After the volume is created, a Persistent Volume object is created and is bound to the PVC, and persistent storage is made available to the application.

While provisioning volumes, the InfoScale CSI plugin supports the following access modes that determine how the volumes can be mounted:

- **ReadWriteOnce (RWO)** -- the volume can be mounted as read-write by a single node.
- **ReadOnlyMany (ROX)** -- the volume can be mounted read-only by many nodes.
- **ReadWriteMany (RWX)** -- the volume can be mounted as read-write by many nodes.

Note: The permission in a Persistent Volume Claim is per node and not per pod. For example, a PVC with RWO mode does not prevent mounting same volume in more than one pod on same node.

Static provisioning

You can use static provisioning if you want to make the existing persistent storage objects available to the cluster. You can statically provision a volume over shared storage (CVM) and shared nothing (FSS) storage.

Static provisioning allows cluster administrators to make existing storage objects available to a cluster. To use static provisioning, you must know the details of the storage object, its supported configurations, and mount options. To make existing storage available to a cluster user, you must manually create a Persistent Volume, and a Persistent Volume Claim before referencing the storage in a pod.

Note: You must ensure that the VxFS file system is created before provisioning the volumes statically. If the VxFS file system does not exist, you must create it manually by using the `mkfs` command from the InfoScale driver container .

Creating Static Provisioning

1 You can create a Storage Class by running the `csi-infoscale-sc.yaml` file which is as under-.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-infoscale-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "false"
provisioner: org.veritas.infoscale
reclaimPolicy: Delete
allowVolumeExpansion: true
parameters:
  fstype: vxfs

  # (optional) Specifies a volume layout type.
  # Supported layouts: stripe, mirror, stripe-mirror, mirror-stripe,
  #                            concat, concat-mirror, mirror-concat
  # If omitted, InfoScale internally chooses the best suited layout
  #                                  based on the environment.
  # layout: "mirror"
  #
  # (optional) Specifies the number of disk or host failures a
  #                                     storage object can tolerate.
  # faultTolerance: "1"
  #
  # (optional) Specifies the number of stripe columns to use when
  #                                       creating a striped volume.
  # nstripe: "3"

  # (optional) Specifies the stripe unit size to use for striped
  #                                        volume.
  # stripeUnit: "64k"
  #
  # (optional) Specifies disks with the specified media type. All
  # disks with the given mediatype are selected for volume creation.
  # Supported values: hdd, ssd
  # mediaType: "hdd"
```

Run `oc create -f csi-infoscale-sc.yaml`

2 You must be ready with the VxVM volume name to define the Persistent Volume object.

Run `oc exec -ti -n <namespace> <driver-container> -- <cmd>` to list Volumes from the InfoScale Driver Container.

An example of this command is `oc exec -ti -n infoscale-vtas infoscale-vtas-driver-container-rhel8-bwvwb -- vxprint -g vrts_kube_dg -vuh | grep -w fsgen`

3 In the `csi-static-pv.yaml`, define the Persistent Volume object and specify the existing VxVM volume name in the volumeHandle attribute.

```
csi-static-pv.yaml
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: csi-infoscale-pv
  annotations:
    pv.kubernetes.io/provisioned-by: org.veritas.infoscale
spec:
  storageClassName: csi-infoscale-sc
  persistentVolumeReclaimPolicy: Delete
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  csi:
    driver: org.veritas.infoscale
    # Please provide pre-provisioned Infoscale volume name.
    volumeHandle: <existing_VxVM_volume_name>
    fsType: vxfs
```

4 Create a Persistent Volume using the yaml.

```
oc create -f csi-static-pv.yaml
```

5 Define the Persistent Volume Claim (PVC) with appropriate access mode and storage capacity.

```
csi-static-pvc.yaml
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-infoscale-pvc
spec:
  accessModes:
   - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: csi-infoscale-sc
```

6 Create a Persistent Volume Claim by using the yaml. This PVC automatically gets bound with the newly created PV.

```
oc create -f csi-static-pvc.yaml
```

7 Update the application yaml file (`mysql-deployment.yaml`) and specify the persistent Volume Claim name.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql:latest
          ports:
            - containerPort: 3306
          volumeMounts:
            - mountPath: "/var/lib/mysql"
              name: mysql-data
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: root123
      volumes:
        - name: mysql-data
          persistentVolumeClaim:
            claimName: csi-infoscale-pvc
```

8 Create the application pod.

```
oc create -f mysql-deployment.yaml
```

9 Check that old data exists on the persistent volume. Run the following commands

`oc get pods | grep mysql` and `oc exec -it mysql-deployment<id> -- mysql -uroot -pRoot12345!.`

Dynamic provisioning

You can dynamically provision a volume over shared storage (CVM) and shared nothing (FSS) storage. In dynamic provisioning, you must create a Storage Class that define the storage provisioner and the required parameters in the storage class yaml file and create the Persistent Volume Claim. The Pod references the Storage Class through an existing Persistent Volume Claim and dynamically allocates storage for the requesting Pod.

While allocating storage to pods dynamically, you can reclaim the storage when the previously provisioned storage is available for other applications to use. You can resize an existing volume using the Persistent Volume Claim (PVC) object.

Perform the following steps for allocating storage dynamically to container workloads:

1. Create a Storage Class using a yaml file.

```
oc create -f csi-infoscale-sc.yaml
```

2. Define the Persistent Volume Claim and specify the appropriate Storage Class, access mode, and the required storage size.

```
csi-dynamic-pvc.yaml
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-infoscale-pvc
spec:
  storageClassName: csi-infoscale-sc
  accessModes:
    - ReadWriteMany

  resources:
```

```
    requests:
      storage: 5Gi
```

3. Create a Persistent Volume Claim using the yaml.

```
oc create -f csi-dynamic-pvc.yaml
```

4. Update `csi-mysql-app.yaml` and specify the Persistent Volume Claim name.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql:latest
          ports:
            - containerPort: 3306
          volumeMounts:
            - mountPath: "/var/lib/mysql"
              name: mysql-data
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: root123
      volumes:
        - name: mysql-data
          persistentVolumeClaim:
            claimName: csi-infoscale-pvc
```

5. Create the application pod.

```
oc create -f csi-mysql-app.yaml
```

After the pod is created, start using the InfoScale PVC as a Persistent Storage.

Reclaiming provisioned storage

When a previously provisioned storage is no longer required by an application, you can delete the corresponding PVC objects from the APIs and reclaim the storage for other applications to use. The reclaim policy for a Persistent Volume states what action the cluster must take on the volume after it is released from the PVC. You can use the following command to delete a PVC:

```
oc delete pvc <pvc_name>
```

InfoScale supports the following reclaim policies. You must specify the reclaim policy while creating a storage class for dynamic provisioning.

- Retain: Indicates that the Persistent Volume must be reclaimed manually.
- Delete: (Default) Indicates that the Persistent Volume and the associated storage gets automatically deleted when the PVC is deleted.

For more information on reclaim policies, see Kubernetes - Persistent Volumes documentation.

Resizing Persistent Volumes (CSI volume expansion)

Using the persistent volume expansion feature, you can easily expand the storage capacity of a persistent volume by just updating the Persistent Volume Claim storage specification. However, to use this feature, you must set the `allowVolumeExpansion` attribute to true in their `StorageClass` object. Only the PVCs created by using such Storage Class allow volume expansion. When the storage attribute of such a PVC object is updated, Container Orchestrator interprets it as a change request and triggers automatic volume resizing.

The following sample Storage Class `yaml` shows the `allowVolumeExpansion` attribute definition.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: csi-infoscale-sc
annotations:
storageclass.kubernetes.io/is-default-class: "false"
```

```
provisioner: org.veritas.infoscale
reclaimPolicy: Delete
allowVolumeExpansion: true
parameters:
fstype: vxfs
```

While performing the volume expansion operation, you must note the following:

- Ensure that the PVC is in use by some application pod while performing resize operation. InfoScale supports dynamic volume expansion in 'Online' mode.
- Do not perform the volume expansion operation from InfoScale driver-container pods. In such case, OpenShift is not aware of these changes and updated volume size is not reflected in the PV and PVC objects.
- InfoScale does not support the shrinking of persistent volume as OpenShift does not support it.

After the volume is provisioned, create the container application pod, run the application, and access the volume. If the volume is full and must be resized, use one of the following ways:

- Edit the Persistent Volume Claim
- Use the `oc patch pvc` command

Note: Resize operation on volume is not supported when it is provisioned in ReadOnlyMany mode.

Resizing a Persistent Volume by editing the Persistent Volume Claim

1 Find the Persistent Volume Claim to resize.

```
oc get pvc
```

Output similar to this is displayed:

```
NAME                STATUS VOLUME  CAPACITYACCESS  STORAGECLASS         AGE
                                          MODES

csi-infoscale-pvc   Bound  pvc-<id>5Gi     RWX     csi-infoscale-sc     32m
```

2 To resize the storage capacity, edit the PVC.

```
oc edit pvc csi-infoscale-pvc
```

3. From your text editor, change the storage capacity to the required larger value. For example, from 5Gi to 10Gi .
4. Check the status of the Persistent Volume Claim and Persistent Volume to verify if the size is updated.

   ```
   oc get pvc
   ```

   Output similar to this is displayed:

```
NAME                 STATUS  VOLUME  CAPACITY ACCESS   STORAGECLASS          AGE
                                             MODES

csi-infoscale-pvc    Bound   pvc-<id>10Gi     RWX      csi-infoscale-sc      32m
```

Resizing a Persistent Volume using the 'patch pvc' command

1. Find the Persistent Volume Claim to resize.

```
oc get pvc
```

Output similar to this is displayed:

```
NAME                  STATUS VOLUME  CAPACITY ACCESS  STORAGECLASS         AGE
                                              MODES

csi-infoscale-pvc     Bound  pvc-<id>5Gi      RWX     csi-infoscale-sc     32m
```

2. To resize the storage capacity to the required larger value, for example, from 5Gi to 10Gi, run the following command.

```
 oc patch pvc csi-infoscale-pvc --patch
'{"spec": {"resources": {"requests": {"storage": "10Gi"}}}}'
```

3. Check the status of the Persistent Volume Claim and Persistent Volume to verify if the size is updated.

```
 oc get pvc
```

Output similar to this is displayed:

```
NAME                  STATUS  VOLUME  CAPACITYACCESS     STORAGECLASS       AGE
                                              MODES

csi-infoscale-pvc     Bound   pvc-<id>10Gi    RWX        csi-infoscale-sc   32m
```

Snapshot provisioning (Creating volume snapshots)

Volume snapshot represents a point-in-time and space-optimized copy of volume on storage system. The InfoScale CSI Plugin supports snapshot provisioning. You can create one or more snapshots of Persistent Volume that is provisioned dynamically or statically. You can also restore a Snapshot to reinstate the volume contents on a completely new Persistent Volume that you want to provision. The snapshots can also be consumed directly as PVC through static provisioning.

For using the point-in-time copies, Veritas recommends that you:

- Use the space-optimized snapshots for read-intensive applications that run on top of either a source Persistent Volume or a snapshot copy. You can use full-instant snapshots for the write-intensive applications.
- Use the Volume Clones feature for write- intensive applications. The volume Clones makes the exact copy of a Persistent volume immediately available for the read, write, and update operations.

To create a snapshot, you must create the following objects by using the `yaml` files:

1. A `VolumeSnapshotContent` is a cluster resource to create a snapshot of a volume in the cluster that is provisioned by an administrator. This resource is similar to a `PersistentVolume`.
2. A `VolumeSnapshot` is a cluster resource for request to create a snapshot of a volume in the cluster that is provisioned by a user. This resource is similar to a `PersistentVolumeClaim`.
3. A `VolumeSnapshotClass` describe the storage classes when provisioning a volume snapshot. The `VolumeSnapshotClass` acts as a template for creating a snapshot and includes attributes like the type of snapshot, synchronization parameters, and other configuration parameters.

Note: The `VolumeSnapshot`, `VolumeSnapshotContent`, and `VolumeSnapshotClass` API objects are Custom Resource Definitions (CRDs) and not a part of the core API. These CRDs and snapshot-controller are pre-installed on OpenShift, but must be manually deployed on Kubernetes.

In the beta version of `VolumeSnapshot`, you must deploy a snapshot controller into the control plane.

Dynamic provisioning of a snapshot

To perform Dynamic provisioning of a snapshot:

1. Define the `VolumeSnapshotClass` object using the `yaml` and specify the `deletionPolicy` and `snapType`.

```
csi-infoscale-snapclass.yaml
---
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: csi-infoscale-snapclass
  annotations:
    snapshot.storage.kubernetes.io/is-default-class: "true"
driver: org.veritas.infoscale
deletionPolicy: Delete
#parameters:
  # (optional) Specifies the type of the snapshot to be created.
  # If omitted,  by default creates space-optimized snapshot.
  # Supported values: space-optimized, full-instant
  # snapType: space-optimized

  # (optional) Specifies the size of the cache volume
    to be created for space-optimized snapshots.
  # If omitted, InfoScale internally chooses the
    cacheSize as 30% of orignial volume size.
  # cacheSize: 500m
```

2. Create Volume Snapshot Class

```
oc create -f csi-infoscale-snapclass.yaml
```

3. Define the Volume Snapshot.

```
csi-dynamic-snapshot.yaml
---
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
  name: csi-dynamic-snapshot
spec:
  volumeSnapshotClassName: csi-infoscale-snapclass
  source:
    persistentVolumeClaimName: csi-infoscale-pvc
```

4 Create Volume Snapshot

```
oc create -f csi-dynamic-snapshot.yaml
```

5 On successful creation of a snapshot, the corresponding volume snapshot content is created and bound to the volume Snapshot object.

Static provisioning of an existing snapshot

To perform static Provisioning of an existing snapshot:

1 You must be ready with the VxVM volume name to define the Persistent Volume object.

Run

```
oc exec -ti -n <namespace> <driver-container> -- <cmd>
```

to list Volumes from the InfoScale Driver Container. You have to specify a Volume for `snapshotHandle` in `csi-static-snapshot-content.yaml`.

An example of this command is

```
oc exec -ti -n infoscale-vtas
infoscale-vtas-driver-container-rhel8-bwvwb -- vxprint -g
vrts_kube_dg -vuh | grep -w fsgen
```

2 Define the volume snapshot content object using the `yaml` file and specify the `snapshotHandle`.

```
csi-static-snapshot-content.yaml
---
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotContent
metadata:
name: csi-static-snapshot-content
spec:
deletionPolicy: Retain
driver: org.veritas.infoscale
source:
# Provide pre-provisioned Infoscale snapshot volume name
snapshotHandle: testSnapVol
volumeSnapshotRef:
name: csi-static-snapshot
namespace: default
```

3 Define the volume snapshot object using the `yaml` and specify the `volumeSnapshotContentName`.

```
csi-static-snapshot.yaml
---
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
name: csi-static-snapshot
spec:
volumeSnapshotClassName: csi-infoscale-snapclass
source:
volumeSnapshotContentName: csi-static-snapshot-content
```

4 Create Volume Snapshot

```
 oc create -f csi-static-snapshot.yaml
```

On successful creation of VolumeSnapshot object, the corresponding volume snapshot content is bound to the volume Snapshot object.

Using a snapshot

You can use the snapshots created using the `VolumeSnapshot` request by restoring them to a new PVC and provisioning that PVC with the pre-populated data from snapshot to an application pod.

You can also use the snapshot volumes as static Persistent Volumes by specifying the snapshot volume name as a value for the `volumeHandle` parameter while provisioning a static PV.

Restoring a snapshot to new PVC

If you want to use and update a point-in-time copy of the application data, you can restore the snapshot of that application's persistent volume to a new persistent volume that represents the previous state described by the snapshot. To restore a volume from a snapshot, you must specify the name of the `VolumeSnapshot` object that you want to restore as the value of the `dataSource` attribute.

Note: While restoring a snapshot or a clone to a new PVC, you must specify the exact same storage details as specified in the source PVC.

To restore a snapshot to a new PVC:

1. Define a Persistent Volume Claim object using the `yaml` and specify the name of the `VolumeSnapshot` object that you want to restore in the `dataSource` attribute:

```
csi-dynamic-snapshot-restore.yaml
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: csi-infoscale-snapshot-restore
spec:
storageClassName: csi-infoscale-sc
accessModes:
- ReadWriteMany
resources:
requests:
storage: 5Gi
dataSource:
name: csi-dynamic-snapshot
kind: VolumeSnapshot
apiGroup: snapshot.storage.k8s.io
```

2. Create the Persistent Volume Claim.

```
oc create -f csi-dynamic-snapshot-restore.yaml
```

Deleting a volume snapshot

You can delete one or more snapshots by deleting the volume snapshot object associated with snapshot. If you set the `DeletionPolicy` to Delete while defining the snapshot object, then the underlying storage snapshot is automatically deleted when the `VolumeSnapshotContent` object is deleted. Use the following command to delete the `VolumeSnapshot` object:

```
oc delete volumesnapshot csi-dynamic-snapshot
```

Note: For space-optimized snapshots, InfoScale maintains the association between the source PVC and the snapshot volume. Therefore, you must delete the snapshot objects before deleting the source PVC. For full-instant snapshots, you can delete the source PVC before deleting the snapshot object after the synchronization between these two is completed. Review CSI controller logs for details of events or observed errors.

Managing InfoScale volume snapshots with Velero

Velero is a backup and recovery solution that assists in backing up and restoring the applications and their corresponding persistent volumes in an OpenShift or Kubernetes environment.

You can integrate InfoScale CSI plugin with Velero to backup and restore CSI-backed volumes across clusters. The following example shows how to configure and use Velero with InfoScale CSI plugin snapshots feature. This example uses MinIO object storage server for storing objects metadata.

Setting up Velero with InfoScale CSI

Prerequisite:

Download and install Velero CLI. For more information, see Velero documentation.

Perform these steps to configure Velero:

1. Set up the InfoScale CSI environment. See "CSI plugin deployment" on page 101.
2. Set up the `MinIO` server. The `00-minio-deployment.yaml` file to set up the `MinIO` server is included in the Velero package. You must edit the IP addresses and ports in the yaml as required.

   ```
   oc apply -f 00-minio-deployment.yaml
   ```

3. Create the Velero secret file with the credentials to access the MinIO server.

   ```
   [default]
   aws_access_key_id=<user_id>
   aws_secret_access_key=<passowrd>
   ```

4. Install Velero by running the below command:

   ```
   velero install \
   --provider aws \
   --features=EnableCSI \
   ```

```
--plugins=velero/velero-plugin-for-csi:v0.1.0,
velero/velero-plugin-for-aws:v1.0.0 \
--bucket velero \
--secret-file ./credentials-velero \
--use-volume-snapshots=True \
--backup-location-config region=minio,s3ForcePathStyle="true",
s3Url=http://minio.velero.svc:9000,publicUrl=http://<ip>:<port> \
--snapshot-location-config region=default,profile=default
```

5. Deploy the application that uses the CSI backed InfoScale volumes.
6. Create a `VolumeSnapshotClass` for the CSI backed volumes using the `csi-infoscale-snapclass yaml`.

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: csi-infoscale-snapclass
  annotations:
    snapshot.storage.kubernetes.io/is-default-class: "false"
driver: org.veritas.infoscale
deletionPolicy: Retain
parameters:
  snapType: full-instant
```

Run the following command to create a VolumeSnapshotClass

```
oc create -f csi-infoscale-snapclass.yaml
```

7. After `VolumeSnapshotClass` is created, set `velero.io/csi-volumesnapshot-class:`, set to `"true"`.

Velero chooses this to back up InfoScale PersistentVolumeClaims.

Taking the Velero backup

After you configure the Velero setup, you can back up all objects in your cluster, or you can filter objects by type, namespace, and label. For more information, see Velero documentation.

Use the velero backup create command to back up applications that are using the CSI volumes.

For example, to back up a namespace run the following command:

```
# velero backup create <backup_name>
--include-namespaces=<namespace_name> –wait
```

Note: When you back up by using Velero, the PVCs of the CSI Volumes are backed up as snapshots on the on-premises InfoScale host.

Creating a schedule for a backup

The schedule operation allows you to back up your data at specified periodic intervals. The first backup is performed when the schedule is created, and subsequent backups happen at the scheduled interval.

Scheduled backups are saved with the name `<SCHEDULE NAME>-<TIMESTAMP>`, where `<TIMESTAMP>` is formatted as `YYYYMMDDhhmmss`.

In an OpenShift or Kubernetes environment, the scheduled backup operation create snapshots of the CSI-backed volumes on a pre-defined time interval.

For example, use the following sample `yaml` file to create backup schedules of the nginx-app namespace after every 30 minutes that has a validity of 2 hours.

```
apiVersion: velero.io/v1
kind: Schedule
metadata:
  name: daily
  namespace: velero
spec:
  schedule: "*/30 * * * *"
  template:
    hooks: {}
    includedNamespaces:
    -   nginx-app
    ttl: 02h00m0s
```

Restoring from the Velero backup

The restore operation allows you to restore all objects and persistent volumes from a previously created backup. You can also restore only a subset of objects and persistent volumes. For more information, see Velero documentation.

The default name of a restore is `<BACKUP NAME>-<TIMESTAMP>`, where `<TIMESTAMP>` is formatted as `YYYYMMDDhhmmss`. You can also specify a custom name.

Use the velero restore create command to restore the OpenShift or Kubernetes objects and InfoScale CSI volumes from the previously created backup.

For example:

```
velero restore create <restore-name>  --from-backup <backup-name>
```

Volume cloning

The CSI volume clone feature duplicates an existing Persistent Volume at given point in time. Cloning creates an exact duplicate of the specified volume on the backend rather than creating a new empty volume. When a clone is created, it is an independent object that can be used as any other PVC. The data of the cloned volume is also in sync with the data of the original dataSource PVC. The cloned volume can be consumed, cloned, snapshotted, or deleted without affecting the original dataSource PVC.

You can clone a PVC only when the following conditions are met:

- The source and destination PVCs are in the same namespace.
- The source and destination Storage Class are the same.

Creating volume clones

The cloning feature enables you to specify an existing PVC as a `dataSource` while creating a new PVC. Prerequisites are as under:

- The source PVC is bound and available for use
- A valid Storage Class is available
- The source PVC is created using the InfoScale CSI driver that supports volume cloning

To clone a PVC from an existing PVC:

1. Identify the PVC that you want to clone.

```
oc get pvc
```

2. Define the PersistentVolumeClaim object using the yaml and specify the name of the PVC object to use as source

```
csi-dynamic-volume-clone.yaml
---
  kind: PersistentVolumeClaim
  apiVersion: v1
  metadata:
    name: csi-infoscale-volume-clone
  spec:
    storageClassName: csi-infoscale-sc
    accessModes:
      - ReadWriteMany
    resources:
      requests:
        storage: 5Gi
    dataSource:
      kind: PersistentVolumeClaim
      name: csi-infoscale-pvc
```

3. Create a volume clone

```
oc create -f csi-dynamic-volume-clone.yaml
```

On successful creation of a clone, it is pre-populated with the data from the specified PVC `dataSource` volume.

Deleting a volume clone

To delete a volume clone, use the following command:

```
oc delete pvc csi-infoscale-volume-clone
```

Verify that the volume clone is deleted and not displayed in the command output.

Using InfoScale with non-root containers

While using InfoScale with containers that are not running as the root user, the storage ownership might need to be changed to ensure that the containers are able

to read or write to the file system. You can specify an `fsGroup` attribute in the pod security context to enable read or write. Using the `fsGroup` attribute instructs OpenShift or Kubernetes to change the ownership of the file system to the specified group. It also instructs runtime to add the specified group to the supplemental groups the container is run with. This ensures that the container processes are able to read and write files in the volume. In the following example `securityContext` includes an explicit `fsGroup`

```
securityContext:
     runAsUser: 1000
     runAsGroup: 3000
     fsGroup: 5000
     fsGroupChangePolicy: "OnRootMismatch"
```

Using InfoScale in SELinux environments

If InfoScale CSI is used to provision volumes in an environment where SELinux is enabled in enforcing mode, the pod definition must explicitly specify a SELinux label. Files in the provisioned volume are then re-labeled and the containers associated with the pod are started in the appropriate SELinux context.

For example, the following `securityContext` includes explicit SELinux options:

```
securityContext:
     runAsUser: 1000
     runAsGroup: 3000
     fsGroup: 5000
     fsGroupChangePolicy: "OnRootMismatch"
     seLinuxOptions:
       level: "s0:c447,c946"
```

To avoid weakening security posture, ensure that you do not reuse the same label for pods that are not expected to access the same volume. Without explicit labels specified, pods may lose access to previously created files, or files that were created from a different node, for the case of `ReadWriteMany` volumes.

CSI Drivers

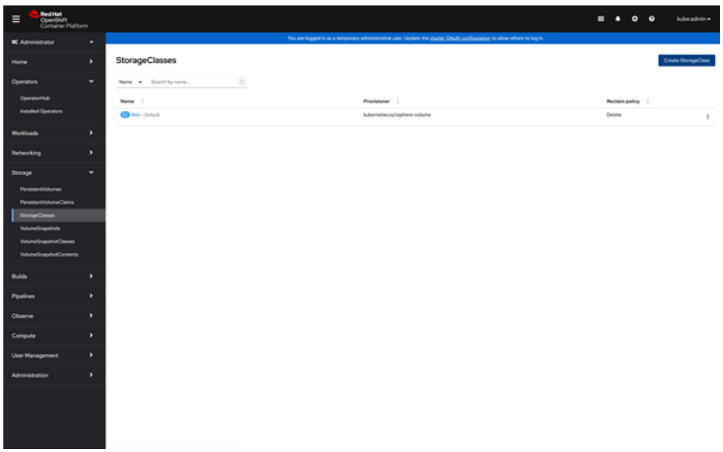
Veritas CSI Driver is a Production Driver. See Kubernetes Drivers for a complete list of Production Drivers.

Creating CSI Objects for OpenShift

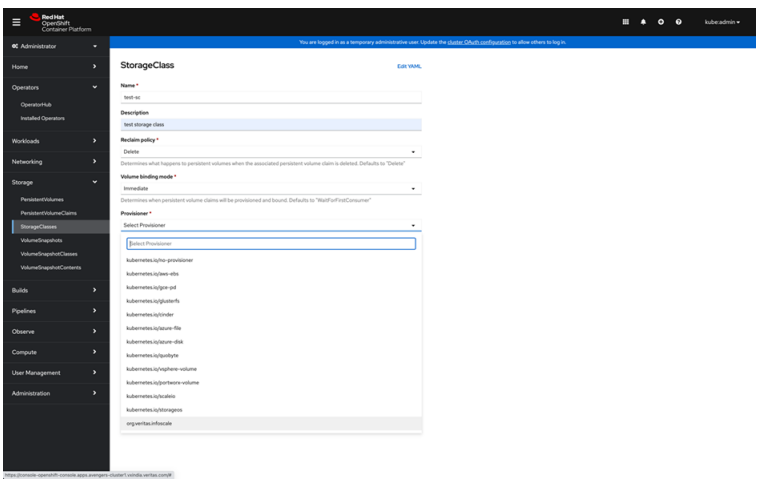
Complete the following steps to install InfoScale operator .

Creating StorageClass

1. Connect to the OpenShift console and access the Catalog menu.
2. In the left frame, click **Storage > StorageClasses**. Click **Create StorageClass** in the upper-right corner of the screen.

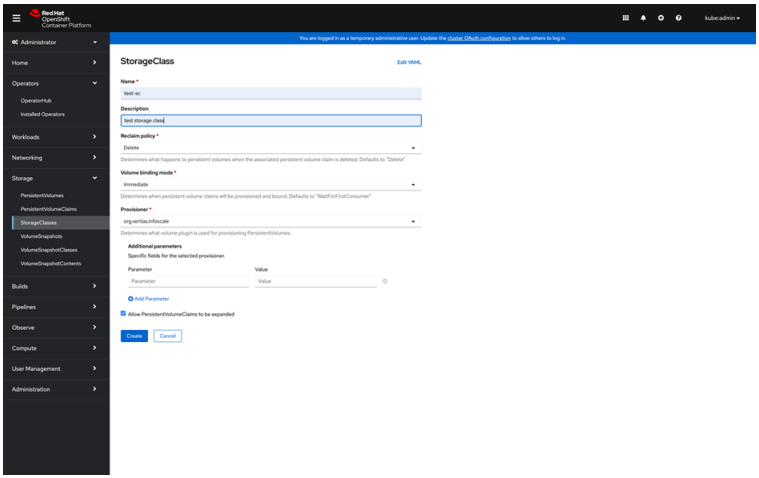


3. Assign a **Name** to the StorageClass in the following screen. Optionally, you can enter Description.



4 Select Immediate as the **Volume binding mode** and org.veritas.infoscale as the **Provisioner**.

5 Click **Add Parameter** in **Additional Parameters**.



6 For **fstype** as the Parameter, enter **vxfs** as its Value.

7 Optionally, you can enter the following parameters.

Parameter	Value
layout	Enter one of the following - stripe, mirror, stripe-mirror, mirror-stripe, concat, concat-mirror, and mirror-concat. By default, a best-suited layout is selected based on the environment
faultTolerance	Number of disk or host failures a storage object can tolerate.
nstripe	Number of stripe columns to use when creating a striped volume
stripeUnit	Stripe unit size to use for striped volume
mediaType	Type of disks to be used for Volume creation (HDD or SSD).

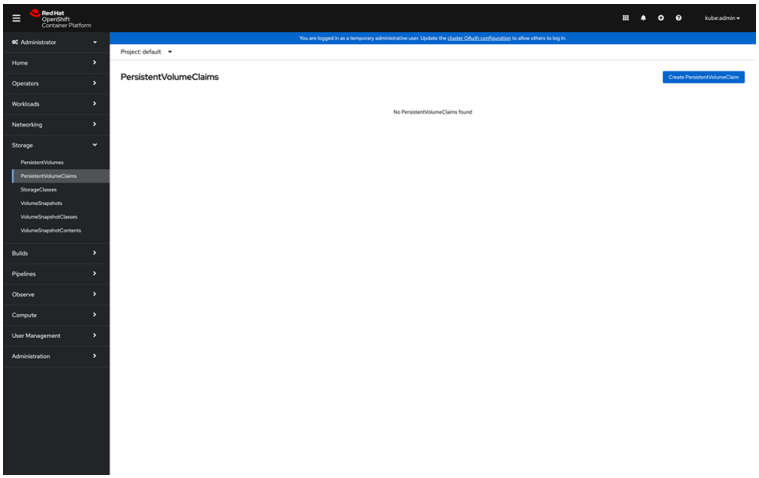
8 Click **Create**. Wait for the StorageClass to be created.

9 Review all details for the StorageClass.

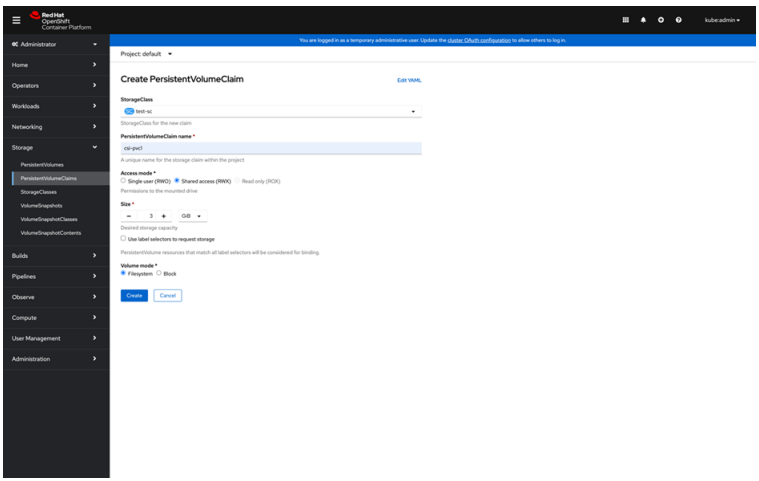
Now you can create a PersistentVolumeClaim.

Creating PersistentVolumeClaims

1 In the left frame, click **Storage > PersistentVolumeClaims**. Click **Create PersistentVolumeClaim** in the upper-right corner of the screen.



2 Select the StorageClass you just created.



3 Enter the following details

Parameter	Value
PersistentVolumeClaim	Assign a name.
Access Mode	Choose **Single User (RWO)** or **Shared Access (RWX)**.
Size	Select the Storage capacity you want to assign to this Volume.
Volume mode	Choose **Filesystem** or **Block**.

4 Click **Create**. Wait for its successful creation.

5 Review information on the screen that is displayed.

After installing InfoScale and creating this PersistentVolumeClaim, Persistent Storage is now available for the Applications.

You can similarly create other storage objects or clones from the OpenShift web console.

Chapter 7

Installing InfoScale DR on OpenShift

This chapter includes the following topics:

Introduction

This section informs you how to install Custom Resource (CR) files related to Disaster Recovery (DR).

Prerequisites

1. InfoScale pods must be configured on the clusters.
2. Load balancer must be installed and configured. If you choose to install Metallb as the load balancer, steps are listed in `Installing DR Operator`.
3. Be ready with the following information for every member cluster
   - GlobalClusterMembership: Virtual IP address and a port for each member cluster in GCM. This information is captured in GCM custom resource.
   - Data Replication: Virtual IP address, port, netmask and network interface (NIC) required for each cluster to be specified in DataReplication CR. This information is exclusive for a DataReplication CR and is used to configure Veritas Volume Replicator .

4. On an OpenShift cluster, `fsGroup` must be set to **RunAsAny** in default-restricted SCC.

   You can run the following command on the bastion node to set the value

   ```
   oc edit scc restricted
   ```

5. Ensure that stale Custom Resources (CR) and Custom Resource Definitions (CRD) related to DR do not exist on the clusters.
6. Storage class used for the Application persistent storage must be based on InfoScale CSI with `RECLAIMPOLICY = Retain`.

   To know RECLAIMPOLICY, run the following command and verify the output

   ```
   oc get sc


   NAME              PROVISIONER            RECLAIMPOLICY VOLUMEBINDINGMODE
   csi-infoscale-sc org.veritas.infoscale Retain        Immediate


   ALLOWVOLUMEEXPANSION AGE
   true                 49d
   ```

7. Optionally if you want to configure DNS resource, **DNS key** and **DNS private key**. To know how to obtain keys, see https://www.veritas.com/content/support/en_US/ doc/129694359-129694362-0/uxrt-731_id-SF1J0175244-129694362 .

External dependencies

- Load balancer service must be installed to allocate Virtual IP addresses for the cluster. Virtual IP addresses ensure a resilient communication between the clusters. See Installing Metallb to install Metallb. Alternatively, you can install any other load balancer service. Refer to its documentation.
- Velero must be installed on all clusters.
  On the bastion node
  - Download Velero binaries from https://github.com/vmware-tanzu/velero/releases/tag/v1.6.0.
  - Run to following command to create Velero namespace.
    ```
    oc create ns velero
    ```
  - Run the following command to install

```
velero install --provider aws --plugins
velero/velero-plugin-for-aws:v1.2.0 --no-default-backup-location
--no-secret
```

Installing InfoScale DR

Complete the following steps to install and configure Disaster Recovery for your InfoScale cluster.

Note: When you download and untar `YAML.tar`, all files required for installation are available.

Configuring DR Operator

Complete the following steps to install the DR operator on the source and the target DR cluster.

1. Run the following command on the bastion node of each cluster.

   ```
   oc apply -f /YAML/DR/dro_deployment.yaml
   ```

2. Wait till the command execution is complete.

3. Run the following command on the bastion node to verify if the deployment is successful.

   ```
   oc -n infoscale-vtas get pods
   ```

   See the Status in the output similar to the following

   ```
   NAME                          READY   STATUS     RESTARTS    AGE
   dr-controller-manager-xxxx    1/1     Running    0           114m
   ```

   Status must change from `ContainerCreating` to `Running`.

4 Run the following commands to configure Metallb.

Note: Run these steps only if you want Metallb as the load balancer. If you choose any other load balancer, refer to its documentation for installation and configuration.

```
oc -n infoscale-vtas expose deployment dr-controller-manager
--name my-lb-service --type LoadBalancer --protocol TCP --port
14155 --target-port 14155
```

Here, DR controller uses port 14155 internally to communicate across peer clusters. After a successful installation and configuration, you can verify by running the following command

5 `oc get svc my-lb-service`

An output similar to the following indicates that installation and configuration is successful

```
NAME            TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)
my-lb-service   LoadBalancer <IP address>  <IP address>   14155:14155/TCP
```

Run this command on both the clusters and verify if installation and configuration is successful. Verify whether `EXTERNAL-IP` is accessible from one cluster to the other cluster.

Configuring Global Cluster Membership (GCM)

With Global Cluster Membership (GCM), you can define membership of clusters for disaster recovery. The GCM CR must be configured and applied on all clusters. When configured, the Global Cluster Membership forms a logical notion called 'Global Cluster' with all underlying clusters as 'Member Clusters'. Member clusters are OpenShift clusters providing disaster recovery capabilities to application components. To provide DR, these member clusters

1. Send heartbeats with each other periodically.
2. Exchange information like state, configuration, operation.
3. Perform/participate in operation like migration.

Complete the following steps

1. Edit `/YAML/DR/SampleGlobalClusterMembership.yaml` as under

```
apiVersion: infoscale.veritas.com/v1
kind: GlobalClusterMembership
```

```
metadata:
  name: global-cluster-membership
spec:

  localClusterName: <Cluster for which you want to create a DR backup>
  globalMemberClusters:

    - clusterID: <ID of the cluster for which you want a DR backup>
      drControllerAddress: "<Load balancer IP address (haproxy)
                                              of the local cluster>"
      drControllerPort: "<Load balancer port number>"

   - clusterID: <ID of the Cluster to be used for a backup>
      drControllerAddress: "<Load balancer IP address (haproxy)
                                               of the DR site>"
      drControllerPort: "<Load balancer port number>"
  # Required details if velero is not installed in "velero" namespace
  # and/or user needs to set a specific User ID, fsGroup in security
  #  context
  veleroConfig:
    # Specify namespace in which velero is installed. This field is
    #  optional
    # if velero is installed in the default "velero" namespace.
    veleroNamespace: "<Namespace where Velero is installed>"

    # User id to enable volume mount
    # This is to comply with default security context constraint.
    # This field is optional for Kubernetes but required for OpenShift
    # if default ID below needs to be changed.
    userID: 1000640000
       <You can change the default value to a valid value for
                    both Primary and DR clusters>

    # Supplemental group to enable volume mount.
    # This field is optional for Kubernetes but required for OpenShift
    # if default ID below needs to be changed.
    FSGroup: 1000640000
       <You can change the default value to a valid value for
                    both Primary and DR clusters>
```

Note: Do not enclose the parameter values in angle brackets(< >) . For example, if 8334 is the Load balancer port number; enter **drControllerPort: "8334"** for **drControllerPort: "<Load balancer port number>"**. **localClusterName** and **clusterID** can have maximum 20 characters.

2. Run the following command on the bastion node of the source cluster.

```
oc apply -f /YAML/DR/SampleGlobalClusterMembership.yaml
```

3. Edit another instance of `/YAML/DR/SampleGlobalClusterMembership.yaml` to add DR site as under

```
apiVersion: infoscale.veritas.com/v1
kind: GlobalClusterMembership
metadata:
  name: global-cluster-membership
spec:

  localClusterName: <Cluster for which you want to create a DR backup>
  globalMemberClusters:

    - clusterID: <ID of the cluster for which you want a DR backup>
      drControllerAddress: "<Load balancer IP address (haproxy)
                                              of the local cluster>"
      drControllerPort: "<Load balancer port number>"

   - clusterID: <ID of the Cluster to be used for a backup>
      drControllerAddress: "<Load balancer IP address (haproxy)
                                               of the DR site>"
      drControllerPort: "<Load balancer port number>"
  # Required details if velero is not installed in "velero" namespace
  # and/or user needs to set a specific User ID, fsGroup in security
  # context
  veleroConfig:
    # Specify namespace in which velero is installed. This field is
    # optional
    # if velero is installed in the default "velero" namespace.
    veleroNamespace: "<Namespace where Velero is installed>"

    # User id to enable volume mount
    # This is to comply with default security context constraint.
    # This field is optional for Kubernetes but required for OpenShift
    # if default ID below needs to be changed.
    userID: 1000640000
```

```
      <You can change the default value to a valid value for
                  both Primary and DR clusters>

   # Supplemental group to enable volume mount.
   # This field is optional for Kubernetes but required for OpenShift
   # if default ID below needs to be changed.
   FSGroup: 1000640000
      <You can change the default value to a valid value for
                  both Primary and DR clusters>
```

4. Copy this file to the DR site and Run the following command again on the bastion node of the DR site.

   `oc apply -f /YAML/DR/SampleGlobalClusterMembership.yaml`

5. Manually verify on all clusters whether the GLOBALCLUSTERSTATE is DISCOVER_WAIT by running `oc get gcm`.

   Various states are

State	Description
UNKNOWN	A transient default Global-Cluster state. After initial configuration/setup, cluster state must transition to DISCOVER_WAIT. Prolonged UNKNOWN state indicates errors in initial configuration/setup. Review DR Controller log for the ongoing activities.
DISCOVER_WAIT	Although local cluster has a copy of GCM and member cluster details, it is not certain whether local copy of GCM and member cluster is up-to-date. Waits till you seed the cluster by updating **GlobalClusterOperation** to **localbuild**. When a member cluster transitions to RUNNING state, all peer clusters with identical membership transition to RUNNING state.
ADMIN_WAIT	If local membership definition does not match with peer cluster's membership definition, clusters transition to this state. Update membership on peer clusters and ensure that it is identical. Peer clusters then transition to RUNNING state.
RUNNING	Cluster transitions to RUNNING state if you seed cluster membership by updating **GlobalClusterOperation** to **localbuild**. Cluster transitions to RUNNING state even when local copy of membership matches with peer clusters.
EXITING	You have initiated DR Controller stop.

State	Description
EXITED	DR Controller stopped.

DISCOVER_WAIT indicates that the cluster is initialized. You can now trigger `localbuild`. Verify the cluster membership details and initiate `localbuild` as under.

6. Run the following command on the bastion node of the primary/source cluster.

   ```
   oc edit gcm global-cluster-membership
   ```

7. Update on the source cluster as under

   ```
   globalClusterOperation: "localbuild"
   ```

   The cluster transitions into RUNNING state and broadcasts membership copy to all peer clusters. A peer cluster with same membership also transitions into RUNNING state, whereas a peer cluster with different membership transitions into ADMIT_WAIT state. Update **Spec:GlobalMemberClusters** to rectify any discrepancy.

8. To verify whether the Global Cluster is successfully created, run the following command on the bastion node.

   ```
   oc get gcm
   ```

9. Review the cluster names, GlobalClusterState, and PeerLinkState in the output similar to the following. GlobalClusterState must be Running and PeerLinkState must be Connected.

   ```
   NAME         LOCALCLUSTER GLOBALCLUSTERSTATE     PEERLINKSTATE
   <Name of     <Cluster ID  Running  {"<Cluster ID for back up>":"Connected"
    the Global  for back up>       ,"<Cluster ID for backing up>":"Connected"
     cluster>
   ```

Configuring Data Replication

Using Data Replication custom resource you can configure replication for persistent data(PVs and PVCs) associated with application components in a namespace. Custom resource created on a cluster is automatically synchronized on all peer clusters. Hence, this CR needs to be configured on the primary cluster only. After CR is configured, replication is set up. Veritas Volume Replicator(VVR) is responsible for performing replication. You can check status of underlying replication and perform operations like stop, pause, resume, and migrate data replication.

You must also configure Data Replication custom resources for Velero. Velero is used to capture application metadata on the primary cluster and restore it on the DR cluster by using VVR. For configuring Velero, you must run the CR on both clusters.

Note: You must configure at least three CR files. One for Velero replication from the primary to the DR, one for Velero replication from the DR to the primary, and one per application /namespace you want to replicate.

Complete the following steps

1. Edit `/YAML/DR/SampleDataReplication.yaml` to configure Velero replication from the primary to the DR as under
2.
```
apiVersion: infoscale.veritas.com/v1
kind: DataReplication
metadata:
  name: <Name for Data replication>
spec:
  localHostAddress: <Virtual IP address to configure VVR>
  localNetMask: <Corresponding netmask to configure VVR>
  localNICMap: <corresponding network interface to configure VVR>
  "host1" : "eth0"
  "host2" : "eth0"
  "host3" : "eth0"
  "host4" : "eth1"
  selector:
    namespace: <namespace where velero is installed, same
                                          as specified in GCM>
   labels:
    component: minio-infoscale-dr-bkp

  currentPrimary: <Current primary cluster name -
                          Name of the cluster you want to back up>

  remoteClusterDetails:
    - clusterName: <ID of the Cluster to be used for a backup>
      remoteHostAddress: <Virtual IP address for VVR configuration of
                                                  this cluster>
      remoteNetMask: <Netmask of this cluster>
      remoteNICMap: <Network interface of this cluster>
      "host5" : "eth1"
      "host6" : "eth0"
```

```
      "host7" : "eth0"
      "host8" : "eth1"


      replicationType: sync
```

Run the following command on the bastion node

```
oc apply -f /YAML/DR/SampleDataReplication.yaml
```

3. Similarly copy `SampleDataReplication.yaml` and edit the file to update currentPrimary, local/remote cluster details appropriately. Apply `SampleDataReplication.yaml` to configure metadata replication from the DR site to the primary.
4. Run the following command on the bastion node to verify whether data replication is set up on both clusters.

   ```
   oc get datarep
   ```

5. Edit another copy of `/YAML/DR/SampleDataReplication.yaml` on the primary cluster as under for replication of persistent data(PVs and PVCs) associated with application components in the specified namespace and labels.

   ```
   apiVersion: infoscale.veritas.com/v1
   kind: DataReplication
   metadata:
     name: <Name for Data replication>
   spec:
     # Virtual IP address to configure VVR
     localHostAddress:  <Virtual IP address to configure VVR>
     # Corresponding netmask to configure VVR
     localNetMask: <Corresponding netmask to configure VVR>
     # Corresponding network interface map (hostname and NIC name map)
     # to configure VVR
     localNICMap: <corresponding network interface to configure VVR>
       "host1" : "eth0"
       "host2" : "eth0"
       "host3" : "eth0"
       "host4" : "eth1"
     # Namespace and optionally labels for which you
     # want to configure data replication
     selector:
       namespace: prod
   ```

```
  labels:
    env: prod
# Current primary cluster name - Name of the cluster you want
# to back up
currentPrimary: <Current primary cluster name -
                       Name of the cluster you want to back up>
# (optional) In case of takeover operation, specify force to
# true along with
# the updated currentPriamry value. In case of migrate operation,
# force should be specified as false and only currentPrimary
# needs to be updated.
#force: false

# Secondary cluster details
remoteClusterDetails:
    # ID of the Cluster to be used for a backup
  - clusterName: <ID of the Cluster to be used for a backup>
    # Virtual IP address for VVR configuration of this cluster
    remoteHostAddress: <Virtual IP address for
                        VVR configuration of this cluster>
    # Corresponsding Netmask of this cluster
    remoteNetMask: <Netmask of this cluster>
    # Corresponding Network interface map of this cluster
    remoteNICMap:<Network interface of this cluster>
      "host5" : "eth1"
      "host6" : "eth0"
      "host7" : "eth0"
      "host8" : "eth1"
    # (optional) replication type can be sync or async.
    # default value will be async if not specified.
    #replicationType: async

    # (optional) replicationState can have values start, stop,
    #  pause and resume.
    # This field can be updated to start/stop/pasue/resume
    #  replication.
    # Default value will be set to start during initial
    # configuration.
    #replicationState: start

    # (optional) network transport protocol can be TCP or UDP.
    # Default value will be set to TCP during initial configuration and
    # can be later changed to UDP.
```

```
#networkTransportProtocol: TCP

# (optional) By default, it will be set to N/A during
# initial configuration, which means the available bandwidth
#      will be used.
# It can be later changed to set the maximum network bandwidth
# (in bits per second).
#bandwidthLimit: N/A

# (optional) Supported values for latency protection are: fail,
#   disable and override.
# By default it will be set to disable during initial configuration
# and can be changed later.
#latencyProtection: disable

# (optional) Supported values log (SRL) protection are: autodcm,
# dcm, fail, disable and override.
# By default it will be set to autodcm during initial configuration
#  and can be changed later.
#logProtection: autodcm
```

Note: Ensure that the current primary cluster name you enter here must be the same that you plan to specify in `DisasterRecoveryPlan.yaml`. For every Disaster Recovery Plan, you must create a separate Data Replication CR. Ensure that namespace and labels in Disaster Recovery Plan and its corresponding Data Replication CR are identical.

6. Run the following command on the bastion node

   ```
   oc apply -f /YAML/DR/SampleDataReplication.yaml
   ```

7. After these commands are executed, run the following command on the bastion node

   ```
   oc get datarep
   ```

8. Review the output similar to the following

   ```
   NAME         SPECCURRENTPRIMARY  STATUSCURRENTPRIMARY RVGNAME
   <Name for    ID of the cluster   ID of the current
   Data         which you want      working cluster
   replication> to back up
   ```

9. Wait for the initial synchronization of the application Persistent Volumes to complete on the DR site. Run the following command on the bastion node of the DR site.

   ```
   oc describe datareplications.infoscale.veritas.com <Data rep name
   for the application>
   ```

   Review the status in the output similar to the following. Data Status must be consistent up-to-date.

   ```
   Spec:
   ..
   ..
   Status:
   ..
   ..
      Primary Status:
      ..
      ..
      Secondary Status:
      ..
      Data Status:  consistent,up-to-date
   ```

Configuring DNS

Optionally, using DNS custom resource you can configure a DNS resource that updates the DNS server entries in the event of a failover or migration. The DNS CR must to be separately applied on all DR clusters. When configured, the DNS CR monitors the resource records for the hostname and IP address mappings on the DNS servers. When the Disaster Recovery Plan is configured, the DNS pointer can be provided in the Disaster Recovery Plan CR. Whenever, the DR plan is activated on any primary cluster, the configured DNS is also activated with the provided hostname and IP addresses. When the disaster recovery plan is migrated, the DNS entries from the primary site are removed and the DNS entries on the secondary site are updated. State of the DNS resource can be -.

State	Description
INIT	Default state, not active.

State	Description
OFFLINE	Corresponding DNS resource is offline. State on non-active cluster.
ONLINE	DNS entries are configured and DNS resource is online. State on the active primary cluster.
FAULTED	Underlying DNS resource is faulted

1 Following steps are the prerequisites for `SampleDNS.yaml`. **DNS private key** and **DNS key** must be added to `infoscale-dns-secret`.

- Run the following command on the bastion node
  `cat dns.private | base64`
  Copy the <dns private key> that is displayed.
- Run the following command on the bastion node
  `cat dns.key | base64`
  Copy the <dns key> that is displayed.
- Run the following command on the bastion node and add the keys
  `oc edit secret infoscale-dns-secret -n infoscale-vtas`

```
apiVersion: v1
data:
 dns.private: <dns private key>
 dns.key: <dns key>
kind: Secret
```

Note: You can add the **data:** section if it is not present in the file.

- Save and close the file.
- Run the following command to verify whether addition of keys is successful
  `oc get secret infoscale-dns-secret -n infoscale-vtas -o json`
  Review the output as under

```
{
    "apiVersion": "v1",
    "data": {
        "dns.key": "<dns key>",
        "dns.private": "<dns private key>"
    },
    "kind": "Secret",
```

.
.
.
.

- The private key files are created in `/etc/vx/dns-certs/` . You can run the following command on any of the InfoScale pods.

  ```
  ls -l /etc/vx/dns-certs/dns.*
  ```

  Review the output as under

  ```
  lrwxrwxrwx. 1 root root 18 Oct 18 05:10 /etc/vx/dns-certs/dns.key
                                                    -> ..data/dns.key
  lrwxrwxrwx. 1 root root 18 Oct 18 05:10 /etc/vx/dns-certs/dns.private
  ```

```
-> ..data/dns.private
```

2 Edit `/YAML/DR/SampleDNS.yaml` as under

```
apiVersion: infoscale.veritas.com/v1
kind: DNS
metadata:
  name: <Add 'Name of DNS' here>
spec:
  # Domain name for the DNS
  domain: "<Add 'example.com' here>"
  # (optional) Path for the file containing private TSIG key,
  # required for secure DNS updates.
  # Configure only for UNIX based DNS server
  tsigKeyFile: "/<Add '/etc/vx/dns-certs/dns.private' here>"
  # (optional) The list of primary master name servers in
  # the domain.
  stealthMasters: ["1.2.3.4"]
  # (optional) An association of DNS resource record value
  # Specify the key values in map format
  resRecord:
        "r7515-054-vm8" : "10.221.85.81"
        "r7515-054-vm9" : "10.221.85.82"
        "r7515-054-vm10" : "10.221.85.83"
        "www" : "r7515-054-vm8"
        "abc" : "r7515-054-vm9"
        "xyz" : "r7515-054-vm10"
  # (optional) Time to Live value, in seconds for DNS entries
  # in the zone
  # default value is 86400
  #ttl: 86400

  # (optional) Time in seconds after which DNS agent
  # attempts to refresh resrecords on DNS server
  #refreshInterval: 0

  # (optional)  Set to "true" if the DNS server that you have
  # configured is a Windows DNS server and only if it accepts
  # secure dynamic updates default is false
  #useGSSAPI: false

  # (optional) Set to "true" if you want to clean up all
  # the existing DNS records for the configured keys before
  # adding new records default is false
  #cleanRRKeys: false
```

```
# (optional) Set to "true" if DNS online should create
# PTR records for each RR of type A or AAAA
# default is false
#createPTR: false

# (optional) Set to "true" if if DNS offline should
# remove all records defined by ResRecord
# default is false
#offDelRR: false
```

Note: name and **domain** are mandatory here. Update **tsigKeyFile** for secure DNS only.

3 Run the following command on the bastion node

```
oc apply -f /YAML/DR/SampleDNS.yaml
```

4 To verify whether DNS resource is created successfully, run the following command on the bastion node

```
oc -n infoscale-vtas get dns.infoscale.veritas.com/Name of DNS
```

5 Review output similar to the following

```
NAME         DOMAIN         STATE
Name of DNS example.com   INIT
```

Note: You must create a DNS resource with its attributes on each member cluster as DNS CR is not synchronized across peer clusters.

Configuring Disaster Recovery Plan

With a Disaster Recovery Plan (DR Plan) you can enable disaster recovery for a particular namespace. For a more granular control, you can selectively label components in the namespace and create a DR Plan with namespace and labels. A DR Plan cannot span multiple namespaces. DR Plan must be created only on the primary cluster. DR Plan is automatically created and synchronized on all peer clusters after its creation on the primary cluster. Migration and other operations on the namespace can be triggered by updating certain attributes.

1 Edit `/YAML/DR/SampleDisasterRecoveryPlan.yaml` as under to create DR plan for application components in a given namespace.

```
apiVersion: infoscale.veritas.com/v1
kind: DisasterRecoveryPlan
metadata:
  name: test-disaster-recovery-plan
spec:
  # Name of cluster that should be treated as primary for this DR plan
  primaryCluster: <ID of the cluster you want to back up>
  # (optional) Set Force To True If Peer Cluster(S) Is Not Reachable
  # And Local Cluster Needs To Perform Takeover
  force: false
  # List Of Member Cluster(s) Where This DRPlan Can FailOver
  # Sequence Of MemberCluster Specified In This List Denotes Relative
  # Preference Of Member Cluster(s)
  # Must Be Subset Of Global Cluster Membership
  preferredClusterList: ["<ID of the cluster you want to back up>",
                         "<ID of the cluster where you want to back up>"]
  # Kind Of Corrective Action In Case Of Disaster
  # default value will be "Manual" if not specified
  clusterFailOverPolicy: Manual
  # Specify Namespace And Optionally Labels to decide what all
  #  needs to be part of the disaster recovery plan
  selector:
    namespace: sample
    labels:
      app: sise
  # (optional) Pointer To Manage Storage Replication
  dataReplicationPointer: test-datareplication
  # (optional) Pointer To Manage DNS Endpoints
  dnsPointer: test-dns
```

Note: If you are configuring multiple Disaster Recovery plans, ensure that any two plans do not have first 24 characters identical. **dataReplicationPointer** is needed only if you have stateful applications that require data replication across peer clusters.

2 Run the following command on the bastion node

```
oc apply -f /YAML/DR/SampleDisasterRecoveryPlan.yaml
```

3 Wait till the command run is successful and the following message appears.

```
disasterrecoveryplan.infoscale.veritas.com/
                  <Name of Disaster recovery plan> created
```

4 Run the following command on the bastion node

```
oc get drplan
```

5 Review the output similar to the following

```
NAME     PREFERREDCLUSTERLIST SPEC.PRIMARYCLUSTER
<Name of("ID of the cluster  "ID of cluster
Disaster you want "           where you want
recovery  to back up          to back up")
plan>




STATUS.PRIMARYCLUSTER DATAREPLICATION DNS
ID of the current     ID of the current
cluster               cluster
```

Chapter 8

Installing InfoScale DR on Kubernetes

This chapter includes the following topics:

Introduction

This section informs you how to install Custom Resource (CR) files related to Disaster Recovery (DR).

Prerequisites

1. InfoScale pods must be configured on the clusters.
2. Load balancer must be installed and configured. If you choose to install Metallb as the load balancer, steps are listed in `Installing DR Operator`.
3. Be ready with the following information for every member cluster
   - GlobalClusterMembership: Virtual IP address and a port for each member cluster in GCM. This information is captured in GCM custom resource.
   - Data Replication: Virtual IP address, port, netmask and network interface (NIC) required for each cluster to be specified in DataReplication CR. This information is exclusive for a DataReplication CR and is used to configure Veritas Volume Replicator .

4. Ensure that stale Custom Resources (CR) and Custom Resource Definitions (CRD) related to DR do not exist on the clusters.
5. Storage class used for the Application persistent storage must be based on InfoScale CSI with `RECLAIMPOLICY = Retain`.

   To know RECLAIMPOLICY, run the following command and verify the output

   ```
   kubectl get sc


   NAME              PROVISIONER           RECLAIMPOLICY VOLUMEBINDINGMODE
   csi-infoscale-sc org.veritas.infoscale Retain        Immediate


   ALLOWVOLUMEEXPANSION AGE
   true                 49d
   ```

6. Optionally if you want to configure DNS resource, **DNS key** and **DNS private key**. To know how to obtain keys, see https://www.veritas.com/content/support/en_US/ doc/129694359-129694362-0/uxrt-731_id-SF1J0175244-129694362 .

External dependencies

- Load balancer service must be installed to allocate Virtual IP addresses for the cluster. Virtual IP addresses ensure a resilient communication between the clusters. See Installing Metallb to install Metallb. Alternatively, you can install any other load balancer service. Refer to its documentation.
- Velero must be installed on all clusters.
  On the master node
  - Download Velero binaries from https://github.com/vmware-tanzu/velero/releases/tag/v1.6.0.
  - Run to following command to create Velero namespace.
    ```
    kubectl create ns velero
    ```
  - Run the following command to install
    ```
    velero install --provider aws --plugins
    velero/velero-plugin-for-aws:v1.2.0 --no-default-backup-location
    --no-secret
    ```

Installing InfoScale DR

Complete the following steps to install and configure Disaster Recovery for your InfoScale cluster.

Note: When you download and untar `YAML.tar`, all files required for installation are available.

Configuring DR Operator

Complete the following steps to install the DR operator on the source and target DR cluster.

1. Edit `/YAML/DR/dro_deployment.yaml` to update the path with private repository path where DR operator image is saved and pulled for installation.
2. Run the following command on the master node of each cluster.

   ```
   kubectl apply -f /YAML/DR/dro_deployment.yaml
   ```

3. Wait till the command execution is complete.
4. Run the following command on the master node to verify if the deployment is successful.

   ```
   kubectl -n infoscale-vtas get pods
   ```

   See the Status in the output similar to the following

   ```
   NAME                              READY   STATUS      RESTARTS     AGE
   dr-controller-manager-xxxx        1/1     Running     0            114m
   ```

   Status must change from `ContainerCreating` to `Running`.

5 Run the following commands to configure Metallb.

Note: Run these steps only if you want Metallb as the load balancer. If you choose any other load balancer, refer to its documentation for installation and configuration.

```
kubectl-n infoscale-vtas expose deployment dr-controller-manager
--name my-lb-service --type LoadBalancer --protocol TCP --port
14155 --target-port 14155
```

Here, DR controller uses port 14155 internally to communicate across peer clusters. After a successful installation and configuration, you can verify by running the following command

6 `kubectl get svc my-lb-service`

An output similar to the following indicates that installation and configuration is successful

```
NAME            TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)
my-lb-service   LoadBalancer <IP address>  <IP address>   14155:14155/TCP
```

Run this command on both the clusters and verify if installation and configuration is successful. Verify whether `EXTERNAL-IP` is accessible from one cluster to the other cluster.

Configuring Global Cluster Membership (GCM)

With Global Cluster Membership (GCM), you can define membership of clusters for disaster recovery. The GCM CR must be configured and applied on all clusters. When configured, the Global Cluster Membership forms a logical notion called 'Global Cluster' with all underlying clusters as 'Member Clusters'. Member clusters are Kubernetes clusters providing disaster recovery capabilities to application components. To provide DR, these member clusters

1. Send heartbeats with each other periodically.
2. Exchange information like state, configuration, operation.
3. Perform/participate in operation like migration.

Complete the following steps

1. Edit `/YAML/DR/SampleGlobalClusterMembership.yaml` as under

```
apiVersion: infoscale.veritas.com/v1
kind: GlobalClusterMembership
```

```
metadata:
  name: global-cluster-membership
spec:

  localClusterName: <Cluster for which you want to create a DR backup>
  globalMemberClusters:

    - clusterID: <ID of the cluster for which you want a DR backup>
      drControllerAddress: "<Load balancer IP address (haproxy)
                                              of the local cluster>"
      drControllerPort: "<Load balancer port number>"

   - clusterID: <ID of the Cluster to be used for a backup>
      drControllerAddress: "<Load balancer IP address (haproxy)
                                               of the DR site>"
      drControllerPort: "<Load balancer port number>"
  # Required details if velero is not installed in "velero" namespace
  # and/or user needs to set a specific User ID, fsGroup in security
  # context
  veleroConfig:
    # Specify namespace in which velero is installed. This field is
    # optional
    # if velero is installed in the default "velero" namespace.
    veleroNamespace: "<Namespace where Velero is installed>"

    # User id to enable volume mount
    # This is to comply with default security context constraint.
    # This field is optional for Kubernetes but required for OpenShift
    # if default ID below needs to be changed.
    userID: 1000640000
       <You can change the default value to a valid value for
                     both Primary and DR clusters>

    # Supplemental group to enable volume mount.
    # This field is optional for Kubernetes but required for OpenShift
    # if default ID below needs to be changed.
    FSGroup: 1000640000
       <You can change the default value to a valid value for
                     both Primary and DR clusters>
```

Note: Do not enclose the parameter values in angle brackets(< >) . For example, if 8334 is the Load balancer port number; enter **drControllerPort: "8334"** for **drControllerPort: "<Load balancer port number>"**. **localClusterName** and **clusterID** can have maximum 20 characters.

2. Run the following command on the master node of the source cluster.

```
kubectl apply -f /YAML/DR/SampleGlobalClusterMembership.yaml
```

3. Edit another instance of `/YAML/DR/SampleGlobalClusterMembership.yaml` to add DR site as under

```
apiVersion: infoscale.veritas.com/v1
kind: GlobalClusterMembership
metadata:
  name: global-cluster-membership
spec:

  localClusterName: <Cluster for which you want to create a DR backup>
  globalMemberClusters:

    - clusterID: <ID of the cluster for which you want a DR backup>
      drControllerAddress: "<Load balancer IP address (haproxy)
                                              of the local cluster>"
      drControllerPort: "<Load balancer port number>"

   - clusterID: <ID of the Cluster to be used for a backup>
      drControllerAddress: "<Load balancer IP address (haproxy)
                                               of the DR site>"
      drControllerPort: "<Load balancer port number>"
  # Required details if velero is not installed in "velero" namespace
  # and/or user needs to set a specific User ID, fsGroup in security
  # context
  veleroConfig:
    # Specify namespace in which velero is installed. This field is
    # optional
    # if velero is installed in the default "velero" namespace.
    veleroNamespace: "<Namespace where Velero is installed>"

    # User id to enable volume mount
    # This is to comply with default security context constraint.
    # This field is optional for Kubernetes but required for OpenShift
    # if default ID below needs to be changed.
    userID: 1000640000
```

```
      <You can change the default value to a valid value for
                  both Primary and DR clusters>

   # Supplemental group to enable volume mount.
   # This field is optional for Kubernetes but required for OpenShift
   # if default ID below needs to be changed.
   FSGroup: 1000640000
      <You can change the default value to a valid value for
                  both Primary and DR clusters>
```

4. Copy this file to the DR site and Run the following command again on the master node of the DR site.

   ```
   kubectl apply -f /YAML/DR/SampleGlobalClusterMembership.yaml
   ```

5. Manually verify on all clusters whether the GLOBALCLUSTERSTATE is DISCOVER_WAIT by running the command on the master node of the cluster- `kubectl get gcm`.

   Various states are

State	Description
UNKNOWN	A transient default Global-Cluster state. After initial configuration/setup, cluster state must transition to DISCOVER_WAIT. Prolonged UNKNOWN state indicates errors in initial configuration/setup. Review DR Controller log for the ongoing activities.
DISCOVER_WAIT	Although local cluster has a copy of GCM and member cluster details, it is not certain whether local copy of GCM and member cluster is up-to-date. Waits till you seed the cluster by updating **GlobalClusterOperation** to **localbuild**. When a member cluster transitions to RUNNING state, all peer clusters with identical membership transition to RUNNING state.
ADMIN_WAIT	If local membership definition does not match with peer cluster's membership definition, clusters transition to this state. Update membership on peer clusters and ensure that it is identical. Peer clusters then transition to RUNNING state.
RUNNING	Cluster transitions to RUNNING state if you seed cluster membership by updating **GlobalClusterOperation** to **localbuild**. Cluster transitions to RUNNING state even when local copy of membership matches with peer clusters.

State	Description
EXITING	You have initiated DR Controller stop.
EXITED	DR Controller stopped.

DISCOVER_WAIT indicates that the cluster is initialized. You can now trigger `localbuild`. Verify the cluster membership details and initiate `localbuild` as under.

6. Run the following command on the master node of the primary/source cluster.

   ```
   kubectl edit gcm global-cluster-membership
   ```

7. Update on the source cluster as under

   ```
   globalClusterOperation: "localbuild"
   ```

   The cluster transitions into RUNNING state and broadcasts membership copy to all peer clusters. A peer cluster with same membership also transitions into RUNNING state, whereas a peer cluster with different membership transitions into ADMIT_WAIT state. Update **Spec:GlobalMemberClusters** to rectify any discrepancy.

8. To verify whether the Global Cluster is successfully created, run the following command on the master node.

   ```
   kubectl get gcm
   ```

9. Review the cluster names, GlobalClusterState, and PeerLinkState in the output similar to the following. GlobalClusterState must be Running and PeerLinkState must be Connected.

   ```
   NAME         LOCALCLUSTER  GLOBALCLUSTERSTATE     PEERLINKSTATE
   <Name of     <Cluster ID   Running  {"<Cluster ID for back up>":"Connected
    the Global  for back up>         ,"<Cluster ID for backing up>":"Connecte
     cluster>
   ```

Configuring Data Replication

Using Data Replication custom resource you can configure replication for persistent data(PVs and PVCs) associated with application components in a namespace. Custom resource created on a cluster is automatically synchronized on all peer clusters. Hence, this CR needs to be configured on the primary cluster only. After CR is configured, replication is set up. Veritas Volume Replicator(VVR) is responsible

for performing replication. You can check status of underlying replication and perform operations like stop, pause, resume, and migrate data replication.

You must also configure Data Replication custom resources for Velero. Velero is used to capture application metadata on the primary cluster and restore it on the DR cluster by using VVR. For configuring Velero, you must run the CR on both clusters.

Note: You must configure at least three CR files. One for Velero replication from the primary to the DR, one for Velero replication from the DR to the primary, and one per application /namespace you want to replicate.

Complete the following steps

1. Edit `/YAML/DR/SampleDataReplication.yaml` to configure Velero replication from the primary to the DR as under

```
apiVersion: infoscale.veritas.com/v1
kind: DataReplication
metadata:
  name: <Name for Data replication>
spec:
  localHostAddress: <Virtual IP address to configure VVR>
  localNetMask: <Corresponding netmask to configure VVR>
  localNICMap: <corresponding network interface to configure VVR>
  "host1" : "eth0"
  "host2" : "eth0"
  "host3" : "eth0"
  "host4" : "eth1" selector:
    namespace: <namespace where velero is installed, same
                                          as specified in GCM>
   labels:
    component: minio-infoscale-dr-bkp

  currentPrimary: <Current primary cluster name -
                          Name of the cluster you want to back up>

  remoteClusterDetails:
    - clusterName: <ID of the Cluster to be used for a backup>
      remoteHostAddress: <Virtual IP address for VVR configuration of
                                                    this cluster>
      remoteNetMask: <Netmask of this cluster>
      remoteNICMap: <Network interface of this cluster>
      "host5" : "eth1"
```

```
    "host6" : "eth0"
    "host7" : "eth0"
    "host8" : "eth1"
    replicationType: sync
```

2. Run the following command on the master node

   ```
   kubectl apply -f /YAML/DR/SampleDataReplication.yaml
   ```

3. Similarly copy `SampleDataReplication.yaml` and edit the file to update currentPrimary, local/remote cluster details appropriately. Apply `SampleDataReplication.yaml` to configure metadata replication from the DR site to the primary.
4. Run the following command on the master node to verify whether data replication is set up on both clusters.

   ```
   kubectl get datarep
   ```

5. Edit another copy of `/YAML/DR/SampleDataReplication.yaml` on the primary cluster as under for replication of persistent data(PVs and PVCs) associated with application components in the specified namespace and labels.

   ```
   apiVersion: infoscale.veritas.com/v1
   kind: DataReplication
   metadata:
     name: <Name for Data replication>
   spec:
     # Virtual IP address to configure VVR
     localHostAddress:  <Virtual IP address to configure VVR>
     # Corresponding netmask to configure VVR
     localNetMask: <Corresponding netmask to configure VVR>
     # Corresponding network interface map (hostname and NIC name map)
     # to configure VVR
     localNICMap: <corresponding network interface to configure VVR>
       "host1" : "eth0"
       "host2" : "eth0"
       "host3" : "eth0"
       "host4" : "eth1"
     # Namespace and optionally labels for which you
     # want to configure data replication
       selector:
       namespace: prod
       labels:
   ```

```
    env: prod
# Current primary cluster name - Name of the cluster you want
# to back up
currentPrimary: <Current primary cluster name -
                        Name of the cluster you want to back up>
# (optional) In case of takeover operation, specify force to
# true along with
# the updated currentPriamry value. In case of migrate operation,
# force should be specified as false and only currentPrimary
# needs to be updated.
#force: false

# Secondary cluster details
remoteClusterDetails:
    # ID of the Cluster to be used for a backup
  - clusterName: <ID of the Cluster to be used for a backup>
    # Virtual IP address for VVR configuration of this cluster
    remoteHostAddress: <Virtual IP address for
                         VVR configuration of this cluster>
    # Corresponsding Netmask of this cluster
    remoteNetMask: <Netmask of this cluster>
    # Corresponding Network interface map of this cluster
    remoteNICMap:<Network interface of this cluster>
      "host5" : "eth1"
      "host6" : "eth0"
      "host7" : "eth0"
      "host8" : "eth1"
    # (optional) replication type can be sync or async.
    # default value will be async if not specified.
    #replicationType: async

    # (optional) replicationState can have values start, stop,
    #  pause and resume.
    # This field can be updated to start/stop/pasue/resume
    #  replication.
    # Default value will be set to start during initial
    # configuration.
    #replicationState: start

    # (optional) network transport protocol can be TCP or UDP.
    # Default value will be set to TCP during initial configuration and
    # can be later changed to UDP.
    #networkTransportProtocol: TCP
```

```
# (optional) By default, it will be set to N/A during
# initial configuration, which means the available bandwidth
#      will be used.
# It can be later changed to set the maximum network bandwidth
# (in bits per second).
#bandwidthLimit: N/A

# (optional) Supported values for latency protection are: fail,
#   disable and override.
# By default it will be set to disable during initial configuration
# and can be changed later.
#latencyProtection: disable

# (optional) Supported values log (SRL) protection are: autodcm,
# dcm, fail, disable and override.
# By default it will be set to autodcm during initial configuration
#  and can be changed later.
#logProtection: autodcm
```

Note: Ensure that the current primary cluster name you enter here must be the same that you plan to specify in `DisasterRecoveryPlan.yaml`. For every Disaster Recovery Plan, you must create a separate Data Replication CR. Ensure that namespace and labels in Disaster Recovery Plan and its corresponding Data Replication CR are identical.

6. Run the following command on the master node

   ```
   kubectl apply -f /YAML/DR/SampleDataReplication.yaml
   ```

7. After these commands are executed, run the following command on the master node

   ```
   kubectl get datarep
   ```

8. Review the output similar to the following

   ```
   NAME         SPECCURRENTPRIMARY  STATUSCURRENTPRIMARY RVGNAME
   <Name for    ID of the cluster   ID of the current
   Data         which you want      working cluster
   replication> to back up
   ```

9. Wait for the initial synchronization of the application Persistent Volumes to complete on the DR site. Run the following command on the master node of the DR site.

   ```
   kubectl describe datareplications.infoscale.veritas.com <Data rep
   name for the application>
   ```

   Review the status in the output similar to the following. Data Status must be consistent up-to-date.

   ```
   Spec:
   ..
   ..
   Status:
   ..
   ..
      Primary Status:
      ..
      ..
      Secondary Status:
      ..
      Data Status:  consistent,up-to-date
   ```

Configuring DNS

Optionally, using DNS custom resource you can configure a DNS resource that updates the DNS server entries in the event of a failover or migration. The DNS CR must to be separately applied on all DR clusters. When configured, the DNS CR monitors the resource records for the hostname and IP address mappings on the DNS servers. When the Disaster Recovery Plan is configured, the DNS pointer can be provided in the Disaster Recovery Plan CR. Whenever, the DR plan is activated on any primary cluster, the configured DNS is also activated with the provided hostname and IP addresses. When the disaster recovery plan is migrated, the DNS entries from the primary site are removed and the DNS entries on the secondary site are updated. State of the DNS resource can be -.

State	Description
INIT	Default state, not active.

State	Description
OFFLINE	Corresponding DNS resource is offline. State on non-active cluster.
ONLINE	DNS entries are configured and DNS resource is online. State on the active primary cluster.
FAULTED	Underlying DNS resource is faulted

1 Following steps are the prerequisites for `SampleDNS.yaml`. **DNS private key** and **DNS key** must be added to `infoscale-dns-secret`.

- Run the following command on the master node
  `cat dns.private | base64`
  Copy the <dns private key> that is displayed.
- Run the following command on the master node
  `cat dns.key | base64`
  Copy the <dns key> that is displayed.
- Run the following command on the master node and add the keys
  `kubectl edit secret infoscale-dns-secret -n infoscale-vtas`

```
apiVersion: v1
data:
 dns.private: <dns private key>
 dns.key: <dns key>
kind: Secret
```

Note: You can add the **data:** section if it is not present in the file.

- Save and close the file.
- Run the following command to verify whether addition of keys is successful
  `kubectl get secret infoscale-dns-secret -n infoscale-vtas -o json`
  Review the output as under

```
{
    "apiVersion": "v1",
    "data": {
        "dns.key": "<dns key>",
        "dns.private": "<dns private key>"
    },
```

```
    "kind": "Secret",
.
.
```

- The private key files are created in `/etc/vx/dns-certs/` . You can run the following command on any of the InfoScale pods.

  `ls -l /etc/vx/dns-certs/dns.*`

  Review the output as under

```
lrwxrwxrwx. 1 root root 18 Oct 18 05:10 /etc/vx/dns-certs/dns.key
                                                    -> ..data/dns.key
lrwxrwxrwx. 1 root root 18 Oct 18 05:10 /etc/vx/dns-certs/dns.private
```

```
-> ..data/dns.private
```

2 Edit `/YAML/DR/SampleDNS.yaml` as under

```
apiVersion: infoscale.veritas.com/v1
kind: DNS
metadata:
  name: <Add 'Name of DNS' here>
spec:
  # Domain name for the DNS
  domain: "<Add 'example.com' here>"
  # (optional) Path for the file containing private TSIG key,
  # required for secure DNS updates.
  # Configure only for UNIX based DNS server
  tsigKeyFile: "/<Add '/etc/vx/dns-certs/dns.private' here>"
  # (optional) The list of primary master name servers in
  # the domain.
  stealthMasters: ["1.2.3.4"]
  # (optional) An association of DNS resource record value
  # Specify the key values in map format
  resRecord:
        "r7515-054-vm8" : "10.221.85.81"
        "r7515-054-vm9" : "10.221.85.82"
        "r7515-054-vm10" : "10.221.85.83"
        "www" : "r7515-054-vm8"
        "abc" : "r7515-054-vm9"
        "xyz" : "r7515-054-vm10"
  # (optional) Time to Live value, in seconds for DNS entries
  # in the zone
  # default value is 86400
  #ttl: 86400

  # (optional) Time in seconds after which DNS agent
  # attempts to refresh resrecords on DNS server
  #refreshInterval: 0

  # (optional)  Set to "true" if the DNS server that you have
  # configured is a Windows DNS server and only if it accepts
  # secure dynamic updates default is false
  #useGSSAPI: false

  # (optional) Set to "true" if you want to clean up all
  # the existing DNS records for the configured keys before
  # adding new records default is false
  #cleanRRKeys: false
```

```
# (optional) Set to "true" if DNS online should create
# PTR records for each RR of type A or AAAA
# default is false
#createPTR: false

# (optional) Set to "true" if if DNS offline should
# remove all records defined by ResRecord
# default is false
#offDelRR: false
```

Note: name and **domain** are mandatory here. Update **tsigKeyFile** for secure DNS only.

3 Run the following command on the master node

```
kubectl apply -f /YAML/DR/SampleDNS.yaml
```

4 To verify whether DNS resource is created successfully, run the following command on the master node

```
kubectl -n infoscale-vtas get dns.infoscale.veritas.com/Name of DNS
```

5 Review output similar to the following

```
NAME         DOMAIN         STATE
Name of DNS example.com   INIT
```

Note: You must create a DNS resource with its attributes on each member cluster as DNS CR is not synchronized across peer clusters.

Configuring Disaster Recovery Plan

With a Disaster Recovery Plan (DR Plan) you can enable disaster recovery for a particular namespace. For a more granular control, you can selectively label components in the namespace and create a DR Plan with namespace and labels. A DR Plan cannot span multiple namespaces. DR Plan must be created only on the primary cluster. DR Plan is automatically created and synchronized on all peer clusters after its creation on the primary cluster. Migration and other operations on the namespace can be triggered by updating certain attributes.

1 Edit `/YAML/DR/SampleDisasterRecoveryPlan.yaml` as under to create DR plan for application components in a given namespace.

```
apiVersion: infoscale.veritas.com/v1
kind: DisasterRecoveryPlan
metadata:
  name: test-disaster-recovery-plan
spec:
  # Name of cluster that should be treated as primary for this DR plan
  primaryCluster: <ID of the cluster you want to back up>
  # (optional) Set Force To True If Peer Cluster(S) Is Not Reachable
  # And Local Cluster Needs To Perform Takeover
  force: false
  # List Of Member Cluster(s) Where This DRPlan Can FailOver
  # Sequence Of MemberCluster Specified In This List Denotes Relative
  # Preference Of Member Cluster(s)
  # Must Be Subset Of Global Cluster Membership
  preferredClusterList: ["<ID of the cluster you want to back up>",
                         "<ID of the cluster where you want to back up>"]
  # Kind Of Corrective Action In Case Of Disaster
  # default value will be "Manual" if not specified
  clusterFailOverPolicy: Manual
  # Specify Namespace And Optionally Labels to decide what all
  #  needs to be part of the disaster recovery plan
  selector:
    namespace: sample
    labels:
      app: sise
  # (optional) Pointer To Manage Storage Replication
  dataReplicationPointer: test-datareplication
  # (optional) Pointer To Manage DNS Endpoints
  dnsPointer: test-dns
```

Note: If you are configuring multiple Disaster Recovery plans, ensure that any two plans do not have first 24 characters identical. **dataReplicationPointer** is needed only if you have stateful applications that require data replication across peer clusters.

---

2 Run the following command on the master node

```
kubectl apply -f /YAML/DR/SampleDisasterRecoveryPlan.yaml
```

3 Wait till the command run is successful and the following message appears.

```
disasterrecoveryplan.infoscale.veritas.com/
                  <Name of Disaster recovery plan> created
```

4 Run the following command on the master node

```
kubectl get drplan
```

5 Review the output similar to the following

```
NAME     PREFERREDCLUSTERLIST SPEC.PRIMARYCLUSTER
<Name of("ID of the cluster  "ID of cluster
Disaster you want "           where you want
recovery  to back up          to back up")
plan>




STATUS.PRIMARYCLUSTER DATAREPLICATION DNS
ID of the current     ID of the current
cluster               cluster
```

Chapter 9

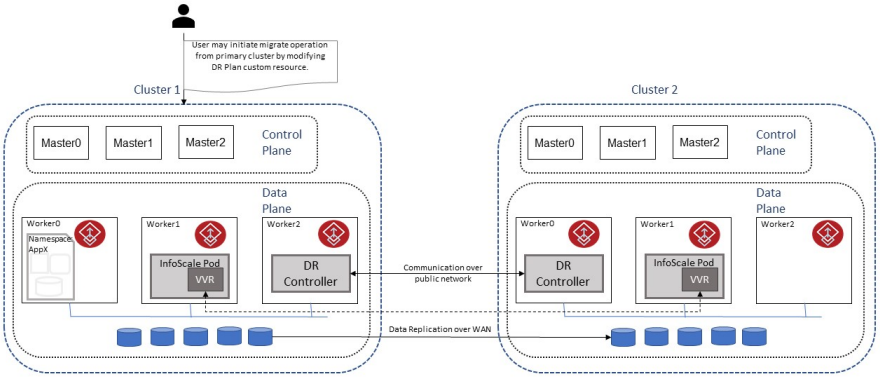
TECHNOLOGY PREVIEW: Disaster Recovery scenarios

This chapter includes the following topics:

Migration

You can initiate migration on a primary cluster when peer clusters are connected, configured for Disaster Recovery (DR), and application stack is online. Migration must be initiated from the primary cluster only – this is the source cluster. Initiating migration from target cluster can result in unstable cluster states. You can run the command `kubectl or oc edit/patch <Name of DR plan>` and update **Spec:PrimaryCluster** to change current primary cluster details. A different **Spec:PrimaryCluster** in specifications and status indicates that migration is ongoing.

Figure 9-1 Migration initiated. Namespace resides on Cluster 1.

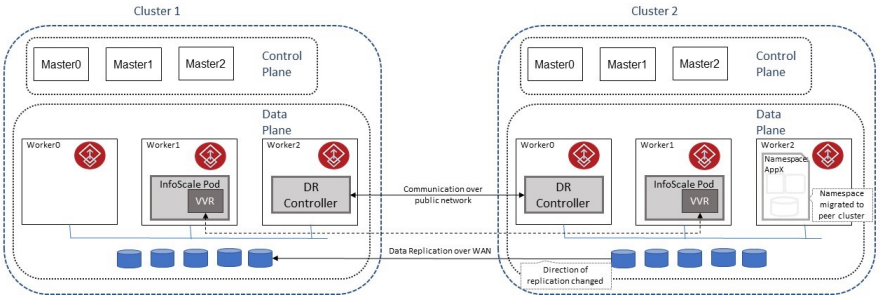


Following entities are updated during migration

1. Application metadata - When migration is initiated from source cluster, latest snapshot of managed application's metadata is taken and tagged for restoration. This latest snapshot is replicated across peer clusters (target cluster) for restoring. Thereafter, application goes offline on the source cluster. On the target cluster, this latest snapshot is used for restoring application stack.
2. Application data - For stateful applications, you must have configured Data Replication CR and updated **DisasterRecoveryPlan:Spec:DataReplicationPointer** accordingly. Data Replication CR manages replication of application data from primary cluster to peer clusters (source to target). Currently, Veritas Volume Replicator(VVR) is used for application data replication. When migration is initiated from the source cluster, the cluster roles are swapped. The proposed primary cluster assumes 'Primary' role whereas current primary cluster assumes 'Secondary' role.
3. DNS endpoints - The DNS custom resource updates and monitors the mapping for:
   - The host name to IP address (A, AAAA, or PTR record)
   - Alias to hostname or canonical name (CNAME)

   When migration is initiated, the DNS resource records are updated appropriately.

Figure 9-2 Migration complete. Namespace resides on Cluster 2.



You can check intermediate transient states like BackupStatus, ScheduleStatus, RestoreStatus, and DataReplicationStatus attributes of Disaster Recovery Plan during migration. To check logs if migration is stuck, run `kubectl/oc logs -f --tail=100 deployments.apps/dr-controller-manager -n infoscale-vtas`. After migration is complete, these transient states are cleaned and **Status:PrimaryCluster** in Disaster Recovery Plan is updated to the new primary.

Chapter 10

Configuring InfoScale

This chapter includes the following topics:

Logging mechanism

InfoScale runs primarily as daemonsets on an OpenShift or a Kubernetes cluster. To access logs of a failed pod/container, the logs must persist beyond the lifecycle of the container. Containerized InfoScale generates logs as log files and container logs. Logs are helpful for debugging purposes. Log files generated by containerized InfoScale persist on the hostPath `/var/VRTS/log` of each host. You can access Container logs of running InfoScale pods/containers by using `oc logs` or `kubectl logs` command.

If DR Controller is configured, controller logs are also included in the Container logs. DR controller logs are independently generated on all peer clusters added in the Global Cluster Membership and hence, logs from all peer OpenShift or Kubernetes clusters must be collected.

To persist container logs beyond the lifecycle of the Container, following methods can be used.

EFK (ElasticSearch, Fluentd, Kibana) - EFK is the default logging stack on OpenShift. See the OpenShift documentation for configuring EFK.

Fluentd log collector - You can use Fluentd log collector to save the container logs at `/var/VRTS/log`. Fluentd runs as a daemonset on the OpenShift or Kubernetes cluster. Hence, it can collect logs generated at each node. On OpenShift or Kubernetes, Fluentd needs to run with privileged mode to access hostPath volumes. Run the following command to enable this -

```
oc adm policy add-scc-to-user privileged -z fluentd
```

Create a file `fluentd-infoscale-spec.yaml`, and apply the following configuration by using oc command.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: infoscale-fluentd-config
  namespace: kube-system
data:
  fluent.conf: |
    <source>
      @type tail
      @id container-input
      read_from_head true
      format none
      path "/var/log/containers/infoscale*.log"
      pos_file "/var/log/infoscale.log.pos"
      tag infoscale.tail.*
    </source>
    <match infoscale.tail.**>
      @type file
      format json
      append true
      path /containerlogs/${tag[5]}
      <buffer tag>
        flush_interval 5s
      </buffer>
      <format>
        @type single_value
        message_key message
      </format>
    </match>

---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: fluentd
  namespace: kube-system

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
```

```
metadata:
  name: fluentd
  namespace: kube-system
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - namespaces
  verbs:
  - get
  - list
  - watch

---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: fluentd
roleRef:
  kind: ClusterRole
  name: fluentd
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
  name: fluentd
  namespace: kube-system
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
    kubernetes.io/cluster-service: "true"
spec:
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
      labels:
```

```
        k8s-app: fluentd-logging
        kubernetes.io/cluster-service: "true"
        name: fluentd
    spec:
      serviceAccount: fluentd
      serviceAccountName: fluentd
      containers:
        - name: fluentd
#
# On Openshift, fluentd needs to run as privileged due to hostPath mounts
#
          securityContext:
            privileged: true

          image: fluent/fluentd-kubernetes-daemonset:v1-debian-elasticsearch
          env:
            - name: FLUENT_UID
              value: "0"
            - name: FLUENT_ELASTICSEARCH_SED_DISABLE
              value: "true"
          volumeMounts:
            - name: config-volume
              mountPath: /fluentd/etc/fluent.conf
              subPath: fluent.conf
            - name: container-logs
              mountPath: /var/log
            - name: hostpath-containerlogs
              mountPath: /containerlogs
#
# for docker containers (usual on vanilla kubernetes), enable below mountpoin
#            - name: varlibdockercontainers
#              mountPath: /var/lib/docker/containers
      volumes:
        - name: config-volume
          configMap:
            name: infoscale-fluentd-config
        - name: container-logs
          hostPath:
            path: /var/log
            type: Directory
        - name: hostpath-containerlogs
          hostPath:
            path: /var/VRTS/log/containers
```

```
            type: DirectoryOrCreate
#
# for docker containers (usual on vanilla kubernetes), enable below volume
#         - name: varlibdockercontainers
#           hostPath:
#             path: /var/lib/docker/containers
#             type: Directory
```

If you have configured DR, add the following to `fluentd-infoscale-spec.yaml`. You can add it at the end of this file.

```
<source>
@type tail
@id container-input
read_from_head true
format none
path "/var/log/containers/dr-controller-manager*.log"
pos_file "/var/log/dr-controller.log.pos"
tag infoscale.tail.*
</source>
```

This configuration enables Fluentd to log all Infoscale containers to the hostPath directory `/var/VRTS/log/`containers of each host.

Configuring Veritas Oracle Data Manager (VRTSodm)

Veritas Oracle Data Manager (VRTSodm) is offered as a part of InfoScale suite. With VRTSodm, Oracle Applications bypass caching and locks of the file system thus enabling a faster connection.

VRTSodm is enabled by the linking `libodm.so` with the Oracle Application. The I/O calls from Oracle Application are then routed through the ODM kernel module.

Following changes are needed to the Oracle database `yaml` file to enable it to run with Veritas ODM.

1. Update the VxFS Data Volume (**<vxfs pvc>**) in the following code and add it to the `.yaml`.

Note: Oracle Container image requires the data volume to be mounted at `/opt/oracle/oradata`. This volume also needs to be writable by the 'oracle' (uid: 54321) user inside the container. VxFS data volume must be mounted at this path by using a PVC. To handle this permissions issue, the following `initContainer` can be used.

```
initContainers:
- name: fix-volume-permission
  image: ubuntu
  command:
  - sh
  - -c
  - mkdir -p /opt/oracle/oradata
    && chown -R 54321:54321 /opt/oracle/oradata
    && chmod 0700 /opt/oracle/oradata
  volumeMounts:
  - name: <vxfs pvc>
  mountPath: /opt/oracle/oradata
  readOnly: false
```

2. Add the following to your `.yaml` to disable DNFS.

```
args:
- sh
- -c
- cd /opt/oracle/product/19c/dbhome_1/rdbms/lib/ &&
 make -f ins_rdbms.mk dnfs_off && cd $WORKDIR &&
 $ORACLE_BASE/$RUN_FILE
```

3. Create a hostpath volume `devodm` in the `.yaml`, and mount at `/dev/odm`.

Note: On selinux-enabled systems (including OpenShift), the Oracle database container must be run as privileged.

4. Use the `libodm.so` that Veritas provides. Run the following commands on the bastion/master nodes.
   - `oc/kubectl cp <infoscalepod>:/opt/VRTSodm/lib64/libodm.so.`
   - `oc/kubectl create configmap libodm --from-file libodm.so.`
   - Mount `libodm.so` inside the oracle container as under

```
      - name: libodm-cmapvol
        mountPath: /opt/oracle/product/19c/dbhome_1/rdbms/lib/odm/libodm.so
        subPath: libodm.so

      volumes:
      - name: libodm-cmapvol
        configMap:
          name: libodm
          items:
          - key: libodm.so
            path: libodm.so
```

Run your .yaml on the bastion mode of the OpenShift cluster or the master node of the Kubernetes cluster.

Alternatively, copy the following content and create a new file `oracle-odm.yaml` .

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: oracle-odm
  labels:
    app: oracledb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: oracledb
  template:
    metadata:
      labels:
        app: oracledb
    spec:
      initContainers:
      - name: fix-volume-permission
        image: ubuntu
        command:
        - sh
        - -c
        - mkdir -p /opt/oracle/oradata && chown -R 54321:54321
          /opt/oracle/oradata && chmod 0700 /opt/oracle/oradata
        volumeMounts:
```

```
  - name: oracle-datavol
    mountPath: /opt/oracle/oradata
    readOnly: false
containers:
- name: oracle-app
  securityContext:
    privileged: true
  image:#replace this with the link for patched oracle container image
  imagePullPolicy: IfNotPresent
  # Modification to the args to disable dnfs before starting database
  args:
  - sh
  - -c
  - cd /opt/oracle/product/19c/dbhome_1/rdbms/lib/ && make -f
       ins_rdbms.mk dnfs_off && cd $WORKDIR && $ORACLE_BASE/$RUN_FILE
  resources:
    requests:
      memory: 8Gi
  env:
  - name: ORACLE_SID
    value: "orainst1"
  - name: ORACLE_PDB
    value: orapdb1
  - name: ORACLE_PWD
    value: oracle
  ports:
  - name: listener
    containerPort: 1521
    hostPort: 1521
  volumeMounts:
  - name: oracle-datavol
    mountPath: /opt/oracle/oradata
    readOnly: false
  - name: devodm
    mountPath: /dev/odm
  - name: libodm-cmapvol
   mountPath: /opt/oracle/product/19c/dbhome_1/rdbms/lib/odm/libodm.so
    subPath: libodm.so
volumes:
- name: oracle-datavol
  persistentVolumeClaim:
    claimName: oracle-data-pvc
- name: devodm
```

```
        hostPath:
          path: /dev/odm
          type: Directory
      - name: libodm-cmapvol
        configMap:
          name: libodm
          items:
          - key: libodm.so
            path: libodm.so
---
apiVersion: v1
kind: Service
metadata:
  name: ora-listener
  namespace: default
  labels:
    app: oracledb
spec:
  selector:
    app: oracledb
  type: NodePort
  ports:
  - name: ora-listener
    protocol: TCP
    port: 1521
    targetPort: 1521
```

Save the file.

Run the file on the bastion mode of the OpenShift cluster or the master node of the Kubernetes cluster to enable a faster connection.

Chapter 11

Troubleshooting

This chapter includes the following topics:

Known Issues

Following are the issues observed during testing in an internal test environment. The issues have remained unresolved and you might encounter these issues. The issue is described and if a workaround exists to resolve the issue, it is mentioned.

Note: Workaround is a temporary solution to the issue. Veritas is working towards fixing the issue.

Table 11-1 Issue description and workaround

Description	Workaround
On a setup with multiple Network Interface Cards, deployment of SRO might fail as the underlying NFD worker pods are not reachable to the NFD master (4045909)	Set the appropriate `IP_AUTODETECTION_METHOD` to be used for Kubernetes communication if Calico CNI is used. or Keep only the valid Network Interface Card interfaces online
Deleting InfoScale pods by using `oc delete pods` or `kubectl delete pods` command might result in Configuration failure. (4045599)	To undeploy InfoScale, delete by using InfoScale CR procedure.

Table 11-1 Issue description and workaround *(continued)*

Description	Workaround
On a VMware Virtual Machine, deployment of InfoScale on OpenShift or Kubernetes fails if Disk UUID is not enabled. (4046388)	Enable Disk UUID before deployment. See Enabling disk UUID on virtual machines.
For space-optimized snapshots, if writes on volume are more than size of cache object size, a write error is observed. This leads to snapshot volume getting marked as INVALID after detaching the mirror. (4043239)	Use space-optimized snapshots only in cases where expected rate of data change is much smaller than actual data volume size.
When a file system is 100% full, and PVC resize is attempted, allocating space for the metadata or the config files required for file system resize might fail, causing PVC resize failure. (4045020)	Contact Veritas support for system recovery.
Deployment of pods with PVC which are restored from a snapshot or are cloned from another PVC and is initiated in ReadOnlyMany(ROM) access mode fails. Deployment goes into **CreateContainerError** state. (4040975)	Set the following deployment parameters to **True** - **Pod.spec.volumes.persistentVolumeClaim.readOnly** and **Pod.spec.containers.volumeMounts[x].readOnly**.
Disk initialization performed by using the `vxdisksetup` command fails with the following error message - `VxVM vxdisksetup ERROR V-5-2-1120 node002_vmdk0_0: Disk is tagged as imported to a shared disk group. Can not proceed..` (4045033)	Ensure that the disk does not belong to any other diskgroup. If it does not belong to any other diskgroup, the disk might have some stale metadata. Run `vxdiskunsetup` on the disk and try disk initialization again.
A message 'File missing or empty: /boot/grub/menu.lst' is displayed even after a successful disk initialization. (4039351)	Ignore the message.
When majority of the nodes in a cluster go in a 'NotReady' state, fast failover (kube-fencing) panics nodes. InfoScale fencing panics rest of the nodes. Even after the cluster is back with majority of the nodes in a 'Ready' state, unfinished kube-fencing jobs continue to panic nodes. (4044408)	Manually delete the kube-fencing jobs till the cluster is up.

Table 11-1 Issue description and workaround *(continued)*

Description	Workaround
With a heavy workload, node goes in a 'NotReady' state and InfoScale pods are getting killed. (4044963)	OpenShift Container Platform (OCP) runs extra system pods which consume memory. With heavy workloads, pods are killed to clear memory. Try the following - ■ Place a resource cap on less important OCP system pods like Prometheus (OCP Monitoring service). See OpenShift documentation. ■ Set pod eviction thresholds and set Kube-reserved and System-reserved resources. Pods are evicted when resources available for the node fall below the limits specified. See OpenShift documentation. ■ Provision higher physical memory for the node.
When a back enclosure is disabled and enabled in a cvm-slave node, disk fails to attach back to the disk group. (4046928)	Run `/etc/vx/bin/vxreattach`.
After faulting a slave node, one or more volumes do not get mounted or existing volumes get unmounted inside application pod. (4044533)	Reschedule/restart the pod to mount the volume.
If a worker node is powered off or rebooted, the node goes into emergency shell and enters NotReady state, thus becoming inaccessible. (4053892)	Reinstall or reconfigure the control plane on the worker node. See OpenShift documentation.
After creating PVC in RWX mode, data written on an application pod running on one node is not accessible from an application pod scheduled on another node. (4046460)	See https://access.redhat.com/solutions/6153272 for the recommended solution.
In container form factor if public/private NICs to be used for LLT are bonded and the underlying bonded NICs have been configured on the same switch,then the worker nodes on which InfoScale is configured might panic randomly with a message - `kernel BUG at mm/slub.c:305!`. (4048786)	If NIC bonding is required for the LLT links, ensure that the underlying NICs are configured on different switches to avoid the kernel node panic, even though the crash has no functional impact. If private links are connected to the same switch, the bond mode must be Active-Backup
Disks from a node of the InfoScale cluster do not get added to the disk group - `vrts_kube_dg`. The following error message `V-5-1-18986 sal_map_devices: da_online failed with error 142 for SAL disk <disk_name>` is logged in `syslog` on the master node. On running `kubectl describe infoscalecluster -n infoscale-vtas`, Output indicates disk addition failure. (4055278)	Add these disks to the disk group manually from the `infoscale-vtas-driver-container` pod.

Table 11-1 Issue description and workaround *(continued)*

Description	Workaround
During InfoScale deployment, InfoScale configuration is not complete on one of the nodes and the node remains in a 'Not ready' state (4047598)	Remove `cr.yaml` and deploy InfoScale again by using `cr.yaml`.
If InfoScale is undeployed on all nodes while retaining the disk groups, re-creating InfoScale cluster on some nodes and adding nodes to the cluster fails. (4047205)	Undeploy and re-create InfoScale cluster on identical number of nodes. To undeploy InfoScale on all nodes and re-create InfoScale clusters on some nodes, contact Veritas support.
After restoring space-optimized snapshot to new PVC, mount on restored PVC may fail if the source snapshot volume is detached (4012858)	Try one of the following - ■ Use CSI space-optimized snapshot functionality for read-intensive applications. ■ Use full-instant snapshot or CSI clone functionality for write-intensive or read-write-update applications. ■ Manually set the values of the configurable parameters like cachesize to an appropriate value based on the application workload while creating CSI volumesnapshotclass object
CSI controller pod remains in 'Terminating' state in case of graceful node shutdown or power-off (4011482)	Try one of the following - ■ If a node must be kept shut down for certain period, to ensure availability, use the following command to drain the node before shutting it down: `kubectl drain <node_name> --force --ignore-daemonsets--delete-local-data` ■ f you intend to delete the node from the Kubernetes cluster, delete the node object. In such case, you need not drain the node manually.
CSI node pods does not get rescheduled on other worker nodes when its parent node is drained (4011384)	None
While restoring a snapshot, PVC goes into pending state after rebooting all nodes except the master node in the cluster (4014525, 4048825)	Delete the snapshot volume by using the `vxedit` command. Kubernetes automatically reattempts to create a volume snapshot again.
In case of storage failure, application IOs to the mountpoint inside container fails and pod goes into CreateContainerConfigError or Error state (4011219, 4014758, 4015259)	Manually restart the application pod after the storage failure is resolved.

Table 11-1 Issue description and workaround *(continued)*

Description	Workaround
Current application on the primary must not be deleted until it is clear that DR is possible. In some cases, DR fails and application gets deleted on the primary.(4047475)	Ensure that the peer clusters are connected and Data Replication is in a healthy state.
If all worker nodes go down at the same time, InfoScale availability configuration is lost (4050355)	After recovery, InfoScale configuration is re-created. It might take up to 20 minutes.
If applications on a cluster with Load Balancer configuration are migrated, Load balancer service appears in 'Pending' state if the target cluster's Load balancer IP addresses are different. (4051429)	If you are using Load Balancer service, use DNS custom resources to manage DNS endpoint mapping.
Delete datarep operation goes in an unresponsive state and force delete in CR fails. (4050857)	Complete the following steps to clean up **1** Check DR controller logs to check which cleanup part is failing. **2** Delete DataReplication Custom Resource (CR) on all clusters by using kubectl or oc edit datarep <name> command and removing finalizer string `infoscale.veritas.com.datareplication/finalizer`. **3** Login to InfoScale cluster pod infoscale-vtas-driver-container-* on all clusters **4** Complete the following steps for Veritas Volume Replicator (VVR) objects cleanup ■ Stop replication for relevant RVG ■ Delete secondary ■ Delete primary ■ Delete corresponding SRL volume **5** Complete the following steps for Veritas Cluster Server (VCS) objects cleanup ■ Change cluster operation to RW ■ Offline VIPgroup and RVGShared service groups corresponding to Datareplication CR (service group names are shown in CR status) ■ Delete resources available in these service. groups ■ Delete service groups' dependencies if any ■ Delete VIPgroup and RVGShared service groups ■ Change cluster operation to RO See Veritas Volume Replicator and Veritas Cluster Server documentation for details.

Table 11-1 Issue description and workaround *(continued)*

Description	Workaround
In case DR migrate fails to complete and running the command `kubectl describe datareplication.infoscale.veritas.com/<datarep_name>` returns a message **vradmin migrate command failed**. (4053632)	Complete the following steps **1** Run the command to know the RVG name `kubectl get datareplications.infoscale.veritas.com <application Data Replication name>` **2** Login to one of the Infoscale driver containers. **3** Run the following command `vxprint -g vrts_kube_dg <RVG name >` **4** Review the output. If tutil is set to 'CONVERTING' , run the following command `vxedit -g vrts_kube_dg -f set tutil0="" <RVG name>` tutil is cleared and DR migration completes

Table 11-1 Issue description and workaround *(continued)*

Description	Workaround
Migration failed to secondary where bind mount for restore velero pod is failing. (4051066)	Complete the following steps to recover data on the source cluster **1** On the target cluster ■ Check the DataReplication for the application, revert to source cluster by editing the Data Replication CR for the application. ■ Confirm that DataReplication for the application is now pointing to the source cluster. ■ Check the DR plan for the application, revert to source cluster by editing the DR plan for the application. ■ Confirm that DR plan for the application is now pointing to the source cluster **2** On the source cluster ■ Confirm that DataReplication for the application is now pointing to the source cluster. ■ Check the DR plan for the application revert to source cluster by editing the DR plan for the application. ■ Confirm that DR plan for the application is now pointing to the source cluster. **3** On the target cluster ■ Delete the minIO pod in velero namespace that was stuck in terminating state. You might have to forcefully delete the pod. ■ Ensure that the corresponding PVC and PV for the deleted velero pod are also deleted. **4** On the source cluster - validate data on the application.
Kube-fencing is not functional when REST service and InfoScale operator pod is not running. (4054545)	None
When multiple PVC snapshot or clone operations are triggered, more than intended snapshot volumes are created. (4054313)	Run `oc / kubectl edit statefulset -n infoscale-vtas infoscale-vtas-csi-driver-controller` to edit csi-controller statefulset pod. Set **timeout value** of csi-snapshotter sidecar container to **300** seconds.

Table 11-1 Issue description and workaround *(continued)*

Description	Workaround
In VVR environment, `vradmin migrate` might fail with the following error message `VxVM VVR vxrvg ERROR V-5-1-1617 giving up: utility fields must be cleared by executing: vxedit -f set tutil0="" <rvg>` (4057713)	Run the following command - `vxedit -f set tutil0="" <rvg>` to clear `tutil` on the RVG and retry the 'vradmin migrate' operation.

During cluster configuration or while adding new nodes on OpenShift or Kubernetes, node join might fail if the disks in the cluster have old disk group records from previous deployments. Output similar to the following indicates old disk group records.
 	
`vxvm:vxconfigd[238047]: V-5-1-11092 cleanup_client: (Disk in use by another cluster) 223`
`esxd05vm06 kernel: VxVM vxio V-5-0-164 Failed to join cluster infoscale_22670, aborting :`
 	
(4057178)
	Reset the cluster ID on the disks of joiner node, in order to allow the node to join.
	1 Check cluster ID on existing/already-joined nodes in cluster: Run
	`vxdisk list node000_vmdk0_9 | grep -i cluster`
	Cluster ID is returned in the output.
	2 If the node join fails, verify if cluster ID is different on the joiner node using same command: Run
	`vxdisk list node001_vmdk0_5 | grep -i cluster`
	A different Cluster ID is returned in the output.
	3 Change the cluster ID to match it with existing nodes in cluster.
	`/etc/vx/diag.d/vxprivutil set /dev/vx/dmp/node001_vmdk0_5 hostid=<Cluster ID returned in the first command>`
	Ensure that disks belong to same disk group. Consult Veritas Technical support.

Stale InfoScale kernel modules might be left over after undeploying InfoScale cluster on OpenShift or Kubernetes.(4042642)	Before deploying InfoScale on OpenShift or Kubernetes, check if any stale InfoScale kernel modules (`vxio/vxdmp/veki/vxspec/vxfs/odm/glm/gms`) are loaded. If stale modules from old deployments are still loaded, reboot all worker nodes and then proceed with the InfoScale deployment.

Table 11-1 Issue description and workaround *(continued)*

Description	Workaround
LLT tools is unable to detect duplicate InfoScale cluster id on OpenShift.(4057800)	Re-deploy InfoScale CR to avoid duplicate cluster id match.

Limitations

Following are the functional limitations due to external factors.

Table 11-2 Limitation description and workaround

Description	Workaround
Storage expansion by adding disks to the node (scale up) is not supported. (4046551) **Note:** This limitation is applicable to Direct Attached Storage (DAS) only.	Reboot the entire cluster after manually adding disks to a node. Then add new disks to the disk group manually when the cluster comes up.
When DataReplication Custom Resource (CR) is configured for a namespace, adding a new PVC to this namespace can result in data corruption. (4052704)	If you want to add a new PVC to an existing namespace **1** Stop data replication by setting ReplicationState attribute to 'stop' and RemoteClusterDetails:Force attribute to 'true' in DataReplication CR. **2** Add the new PVC and verify whether it is visible in DataReplication CR. **3** Reset RemoteClusterDetails:Force as 'false' in DataReplication and start data replication by resetting ReplicationState to 'start'.
If data replication is configured and Volume is mounted only on a slave node, Volume resize operation (vxresize) fails with an error message 'file system is not mounted'. (4047378)	None
When DataReplication for DR is configured for a Galera 3 pod cluster, VVR performs a full synchronization instead of a smartsync (4051639)	None. Wait for initial full synchronization to complete before triggering the migration.

Table 11-2 Limitation description and workaround *(continued)*

Description	Workaround
Remove node is not supported for InfoScale on OpenShift or Kubernetes. (4044647)	To remove node, contact Veritas Technical support.