# Veritas InfoScale™ for Kubernetes Environments 8.0.310 - Linux

VERITAS™

Last updated: 2024-02-09

## 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:



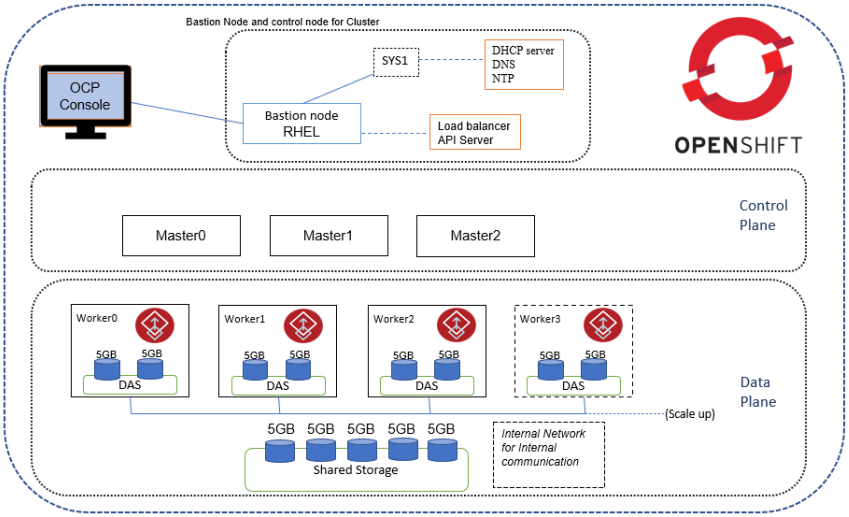
## Introduction

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 and 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. Multiple InfoScale clusters can be configured within a single OpenShift or Kubernetes cluster.

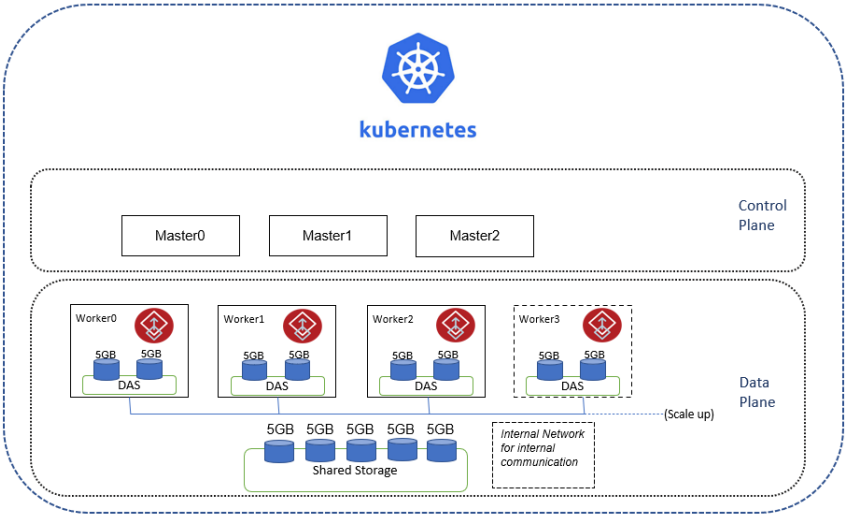
For OpenShift, you can download files from the Red Hat registry and deploy InfoScale. With an active Red Hat account, you can access the InfoScale images. Download from a single source with a single sign-on ensures a high level of security. An example of an OpenShift cluster is as under

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



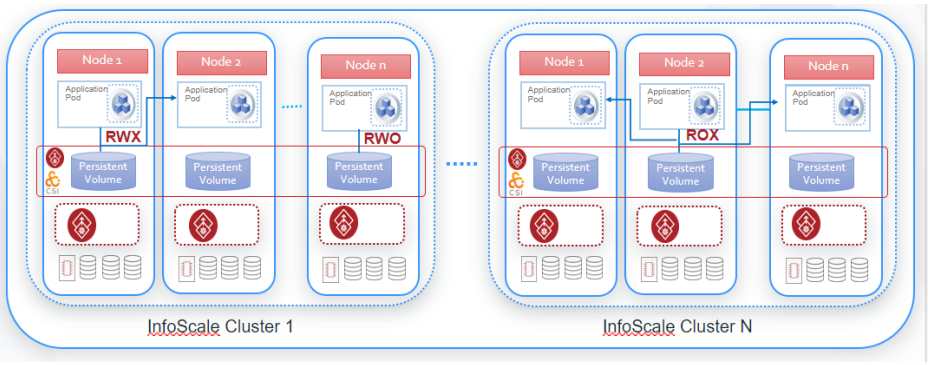

For Kubernetes, you can download files from the Veritas Download Center and deploy InfoScale. An example of a Kubernetes cluster is as under

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



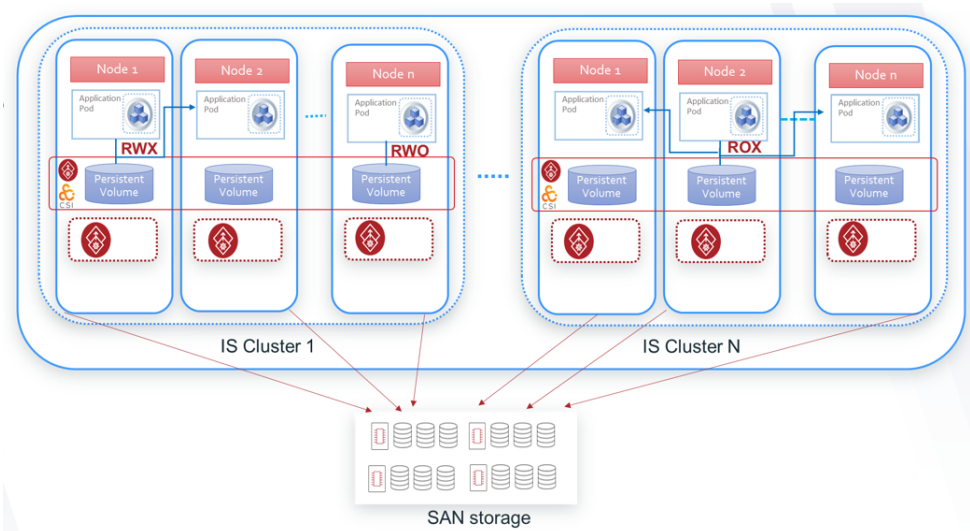

Veritas InfoScale in Kubernetes environment also offers scalability support which is helpful in case the Container environment has large number of nodes. You can configure any number of InfoScale clusters within a single OpenShift or Kubernetes cluster. Each cluster can have upto 16 nodes. To configure multiple clusters, you must modify the configuration resource file and add information about the nodes you want to be included in that InfoScale cluster.

**Figure 1-3** Multiple InfoScale clusters within a single OpenShift or Kubernetes cluster with Direct Attached Storage (DAS)



Multiple InfoScale clusters enable you to run containerized applications on a subset of nodes. Storage attached to all the nodes across the cluster can thus be consumed. However, InfoScale clusters cannot share storage with each other.

**Figure 1-4** Multiple InfoScale clusters within a single OpenShift or Kubernetes cluster in a SAN environment




# 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).
- Support for encryption at disk group and volume level.

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.
- Rolling upgrades (both yaml-based and operator-based) can be performed to subsequent patch releases.
- Support for multiple InfoScale clusters within a single OpenShift or Kubernetes cluster.

# 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. Raw block volume is also supported, where applications directly write to the provided block device.

# I/O fencing

InfoScale uses majority-based and disk-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 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 a 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.

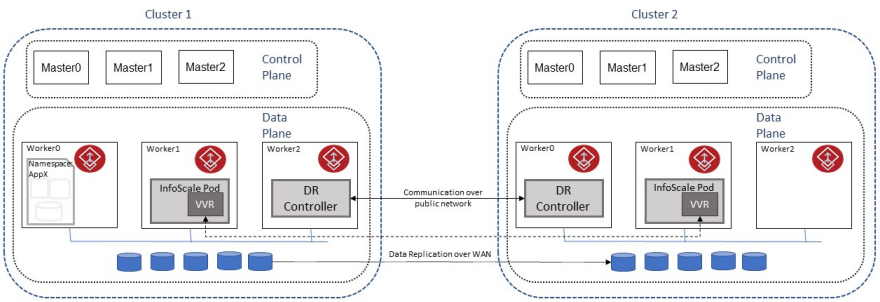
For deployment in containerized environments, when you install InfoScale by using the product installer, the fencing module is automatically installed and configured in majority mode by default. For the default flexible shared storage (fss) disk group configuration, default majority-based fencing can be used. In case of network split, the I/O fencing module takes fencing decisions based on the number of nodes in a sub-cluster. If majority of the nodes are down, all nodes in the cluster are brought down.

If you are using fully-shared storage, where all disks are connected to all the nodes; Veritas recommends using disk-based fencing. With disk-based fencing, cluster is up even with a single functional node.

# 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.

**Figure 1-5** 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).

Disaster Recovery can be configured across on-premise and cloud combinations. Primary site can be on-premise or on cloud. Similarly, the secondary site can be on-premise or on cloud. Disaster Recovery can be set up in any combination of the primary and secondary site. Load balancer can be configured when the site is on cloud. Load balancer configuration ensures traffic during data replication is directed to the appropriate worker nodes. Currently the supported cloud services are - Azure and Amazon Web Services (AWS).

---

**Note:** InfoScale Licensing Custom Resource (CR) must be created before creating the Disaster Recovery (DR) custom resources. To enable DR, you must have Enterprise type of license.

---

# Licensing

You must create a license to enable use of InfoScale. InfoScale in Containers Environment is offered in three editions of licenses which are -

- Developer - InfoScale on three nodes on a single cluster.
- Storage - InfoScale on a trial basis for 60 days without any restriction on the number of nodes and clusters.
- Enterprise - InfoScale and Disaster Recovery (DR) on a trial basis for 60 days without any restriction on the number of nodes and clusters.

You can choose the license edition while creating licenses during installation. After completion of the trial period of 60 days, you must connect with Veritas InfoScale Operations Manager (VIOM) for a subscription license.

---

**Note:** Disaster Recovery (DR) is not supported on Developer and Storage editions. If you are upgrading InfoScale from an earlier Trialware edition, choose Enterprise as the edition.

---

# Encryption

With encryption, you can secure data at rest and during transmission. Encryption is a technology which converts data into a code by using a key. Data when encrypted is not readable. It is readable only when decrypted by authorized users using the same key that was used for encryption.

Encryption is performed by using the FIPS-compliant Advanced Encryption Standard (AES) 256-bit method. By default, encryption is not enabled. You can enable encryption at disk group level while configuring `cr.yaml`.

When Disaster Recovery (DR) is configured, encrypted data is transferred across sites. For DR configuration, encryption is supported at volume level only.

With encryption, you can -

- Protect data from unauthorized access.
- Retire disks without the overhead expense of clearing data.

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 | 4.12.36, 4.12.37, 4.12.39, 4.12.40, 4.12.41<br>4.13.15, 4.13.19 |
| Kubernetes version | 1.25.1 on RHEL 8.6<br>1.27.2, 1.27.3, 1.27.4, and 1.28.2 on RHEL 8.6 |
| Red Hat Cert Manager versions | Cert Manager v1.11.1 or later on OCP 4.12.x<br>Cert Manager v1.11.1 or later on OCP 4.13.x |

The following table lists the kernel versions per Operating System for every Container environment.

**Table 2-2** Supported Operating Systems and kernel versions

| Container Environment | Operating system with version | Minor kernel versions |
|---|---|---|
| 4.12.36, 4.12.37, 4.12.39, 4.12.40, 4.12.41 | CoreOS | 4.12.36 - 4.18.0-372.73.1.el8_6.x86_64<br>4.12.37 - 4.18.0-372.75.1.el8_6.x86_64<br>4.12.39- 4.18.0-372.76.1.el8_6.x86_64<br>4.12.40- 4.18.0-372.76.1.el8_6.x86_64<br>4.12.41- 4.18.0-372.76.1.el8_6.x86_64 |
| 4.13.15 | CoreOS | 5.14.0-284.34.1.el9_2.x86_64 |
| 4.13.19 | CoreOS | 5.14.0-284.36.1.el9_2.x86_64 |

Containerd is the supported runtime environment for Kubernetes. Crio is the supported runtime environment for OpenShift.

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 each product when the `/opt`, `/root`, `/var`, and `/bin` directories are created on the same disk.

**Table 2-3** Disk space requirements

| Product name | Disk space (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

**Note:** These requirements are for a basic cluster configuration. Depending on your workloads/applications, higher values might be required.

**Table 2-4** Hardware requirements

| Requirement | Description |
|---|---|
| Memory (Operating System) | Minimum 24 GB |

**Table 2-4** Hardware requirements *(continued)*

| Requirement | Description |
|---|---|
| CPU (on Kubernetes) | On Physical servers - a minimum of 2 processors with 6/8 cores each.<br>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.<br>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. |
| 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.<br>In a Flexible Storage Sharing (FSS) environment, shared storage may not be required.<br>**Note:** Virtual Machine Disk (VMDK) cannot be used in a full-shared mode. Raw device mapping can be used to enable shared storage in a virtual environment. |
| Cluster platforms | There are several hardware platforms that can function as nodes in a Veritas InfoScale cluster.<br>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. Any number of InfoScale clusters can be configured.

# 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:

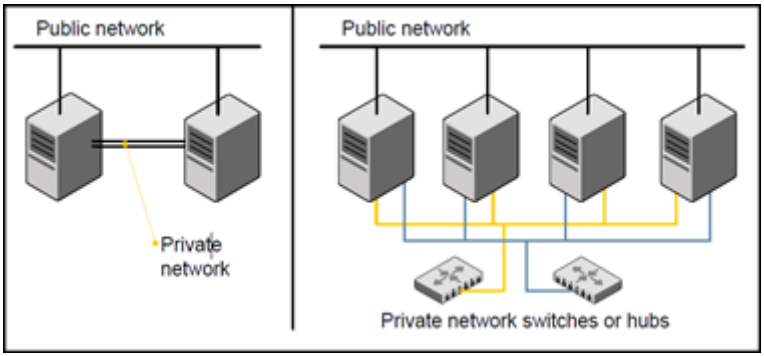


## Setting up the private network

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.

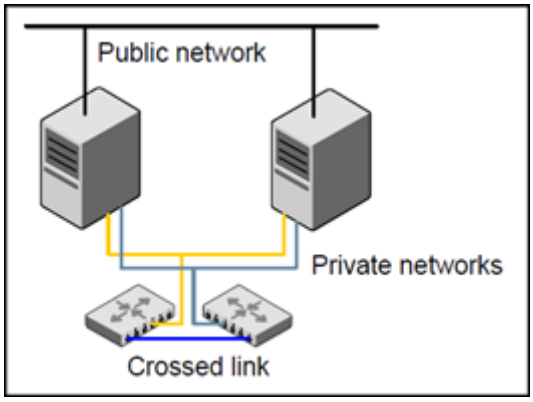
The following figure shows two private networks for use with InfoScale.

**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.

4. Two strong cipher suites are enabled by default. You can edit the operator yaml files and modify `TLS_CIPHER_SUITES` and change the values of the cipher suites. You can indicate two cipher suites separated by a comma.

   If you enter an invalid cipher input and change the input to a valid cipher input, the InfoScale pods might take time to restart. In such a case, wait for the pods to restart automatically or you can delete the pods manually and restart.

5. On a Kubernetes cluster, create a secret by name infoscale-imgpuller to use the secured custom registry. The secrets must be created in the namespace where you plan to install InfoScale operators and all the namespaces where you want to configure InfoScale clusters.

   Run the following command in all the namespaces.

   ```
   kubectl create secret docker-registry infoscale-imgpuller
   --docker-server=<FQDN of the private docker registry>
   --docker-username=<Docker username> --docker-password=<Docker
   password> --docker-email=<Docker email ID> --namespace <Namespace>
   ```

   To know more about image pull secret, refer to Kubernetes documentation.

# 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/4.10/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:** You can form multiple InfoScale clusters. Each cluster can have upto 16 nodes. Veritas InfoScale is deployed on all the nodes you specify in the Custom Resource yaml file. With HyperConverged Architecture supported, a stateful application consuming storage from an InfoScale cluster must run on any of the nodes that are part of that InfoScale cluster.

# 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 hardware 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.

6. Hostnames of the InfoScale nodes must exactly match with the fully qualified domain names(FQDN) of the OpenShift nodes.
7. OpenShift cluster must be configured by using IPv4 only.
8. Cert-manager v1.11 or later on OCP 4.12 or Cert-manager v1.7.1 on OCP 4.11 (Red Hat-certified) must be installed.
9. NFD must be installed.

# Additional Prerequisites for Azure RedHat OpenShift (ARO)

In an Azure RedHat OpenShift (ARO) environment, for InfoScale to create and manage PVC, disks must be assigned to worker nodes for creation of an InfoScale cluster. Perform the following steps.

- Copy the following content into `storage-provisioning.yaml` for every cluster you want to install.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: storage
  labels:
    app: infoscale
spec:
  podManagementPolicy: "Parallel"
  serviceName: "provisioner"
  replicas: <number of nodes>
  selector:
    matchLabels:
      app: infoscale
  template:
    metadata:
      labels:
        app: infoscale
    spec:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: topology.kubernetes.io/zone
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - infoscale
      - maxSkew: 1
        topologyKey: kubernetes.io/hostname
        whenUnsatisfiable: ScheduleAnyway
        labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - infoscale
      containers:
      - image: busybox
```

```
        name: busybox
        command: ['sh','-c','--']
        args: ["while true; do sleep 300; done;"]
        volumeDevices:
        - devicePath: "/var/"
          name: infoscale-provision-pvc
  volumeClaimTemplates:
  - metadata:
      name: infoscale-provision-pvc
      namespace: <InfoScale namespace>
    spec:
      storageClassName: <Name of the storage class>
      volumeMode: Block
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: <Size of the disk you want to
                   configure under Infoscale>
---
{{- end }}
```

- Run `oc apply -f storage-provisioning.yaml` to apply the file for all worker nodes.
- To verify whether the PVC creation is successful, run `oc get pvc -n infoscale-vtas` on the bastion node.
- To verify whether selector is being used, run `oc get statefulset -n <InfoScale namespace>` on the bastion node.
  Output similar to the following indicates a successful creation of pods.

```
default   storage-0  1/1     Running    0  22h
default   storage-1  1/1     Running    0  22h
default   storage-2  1/1     Running    0  22h
default   storage-3  1/1     Running    0  22h
default   storage-4  1/1     Running    0  22h
```

- Run `oc get po -n infoscale-vtas` on the bastion node.
- Be ready with the temporary storage path on each node of the cluster. Ensure that you specify this path as `excludeDevice` while configuring `cr.yaml`. If you

do not specify this path, the temporary storage is consumed. As this storage is not persistent, it might lead to a data loss.

- Ensure that you do not power off the Azure RedHat OpenShift (ARO) cluster after provisioning storage.

---

**Note:** Auto scaling is not supported on Azure RedHat OpenShift (ARO) cluster.

---

# Considerations for configuring cluster or adding nodes to an existing cluster

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.

By default, Flexible Storage Sharing (FSS) disk group is created. If shared storage is the only storage available across nodes you can set `isSharedStorage` to true and fully shared non-FSS disk group is created while configuring clusters. Mirroring is performed across enclosures thus ensuring redundancies. While using shared storage, Veritas recommends changing the default majority-based fencing to disk-based fencing. With shared storage and majority-based fencing, storage becomes inaccessible when majority of the nodes go down; although all disks are connected to all the nodes. With disk-based fencing, storage is available to applications even when one node is up. To configure disk-based fencing, specify hardware path of the fencing disks for at least one node. Veritas recommends using at least three disks for fencing purpose. Ensure that the number of disks is odd in number.

You can enable encryption at the disk group level or for specific Volumes within the disk group. Encryption is not enabled by default. You can set `encrypted` to true to enable encryption. If you want the same encryption key for all Volumes, you can set `sameEnckey` to true. For different encryption keys per Volume, set `sameEnckey` to false.

---

**Note:** For Disaster Recovery (DR) configuration, only Volume level encryption is supported. Disk group level encryption is not supported.

---

# Creating multiple InfoScale clusters

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

You can configure multiple clusters in different namespaces. After a cluster is configured and pods are created successfully, you can update `cr.yaml` with the name, namespace, Cluster ID, and node-related information of the next cluster. You can choose to rename `cr.yaml` or you can overwrite the same `cr.yaml`. You can then create the cluster and after a successful creation of pods, update `cr.yaml` for the subsequent cluster. Likewise, you can configure multiple clusters.

Each InfoScale cluster must have a unique name even when clusters are created across namespaces.

# 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.

- Installing from OperatorHub by using web console
- Installing from OperatorHub by using Command Line Interface (CLI)
- Installing by using YAML

## Installing from OperatorHub by using web console

Complete the following steps to install InfoScale operator .

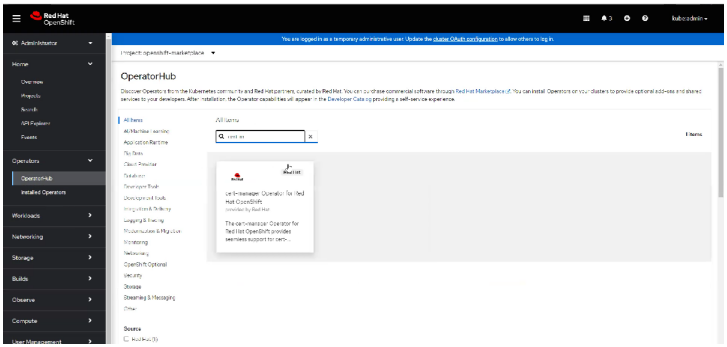
---

**Note:** Below images are for reference only. Ensure that you select the appropriate versions wherever required.

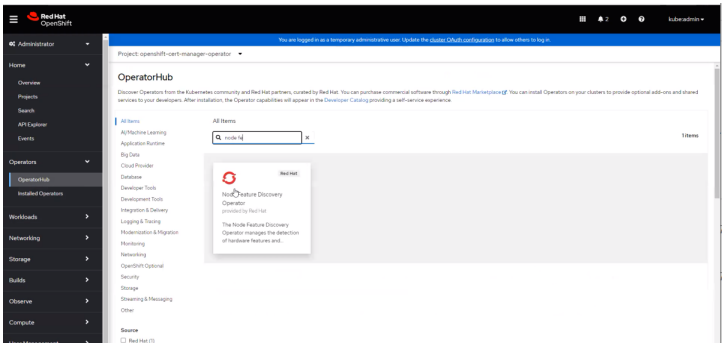
---

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. As a prerequisite, you must download and install Cert-Manager and Node Feature Discovery (NFD).

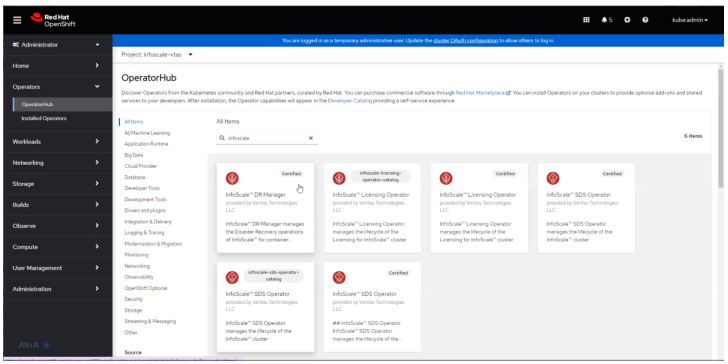
**4** Enter Cert-Manager in **All Items**. Install the cert-manager that is listed as under.



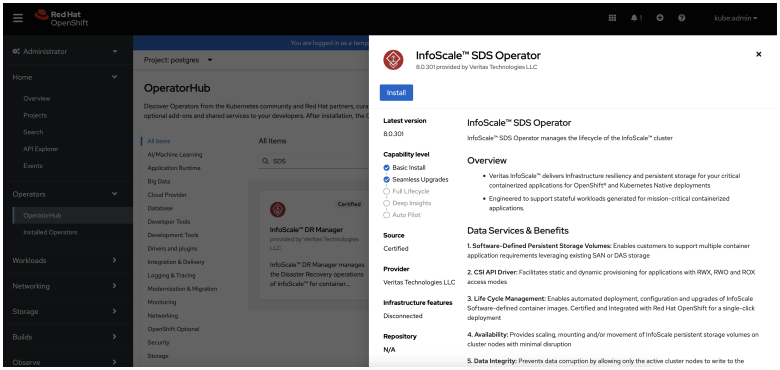

**5** Similarly, enter Node feature in **All Items**. Node Feature Discovery Operator is listed as under. Install it.



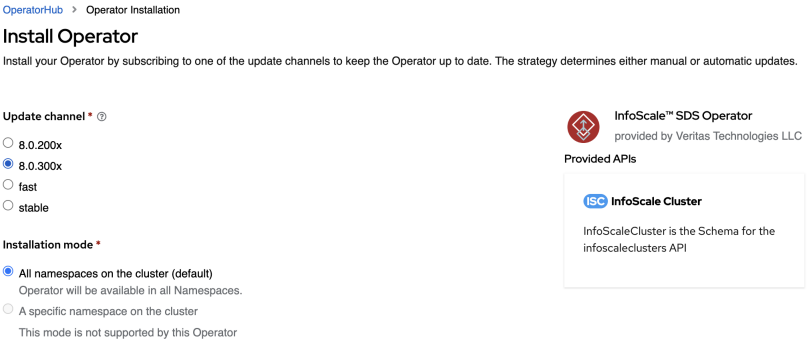

**6** Enter InfoScale in **All Items**. The InfoScale SDS Operator, InfoScale DR Manager, and the InfoScale Licensing Operator are displayed.



**7** Select the **InfoScale SDS Operator** and click **Install** in the following screen.




**8** During download and installation of operators, when prompted ensure that **8.0.300x** is selected as the **Update channel** and **All namespaces in the cluster** is selected as the **Installation mode** as under.



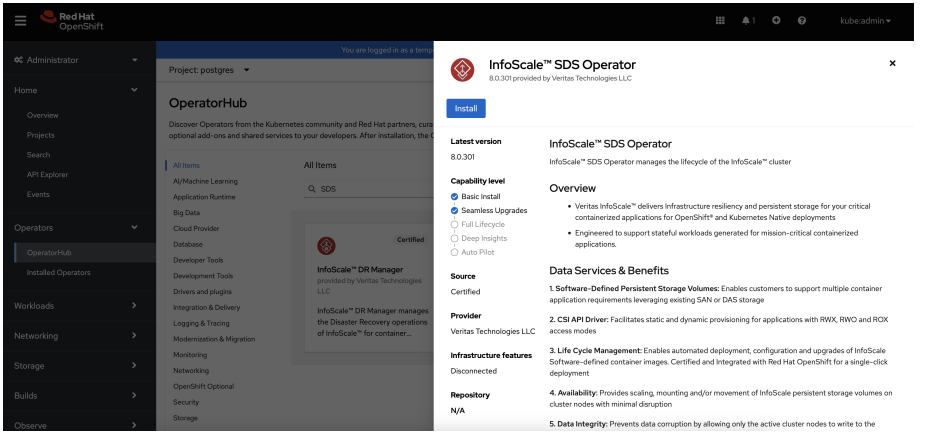
**9** In the following screen, select the namespace where you want to install in **Installed Namespace** and **Manual** in **Update approval**. Click **Install**.

---

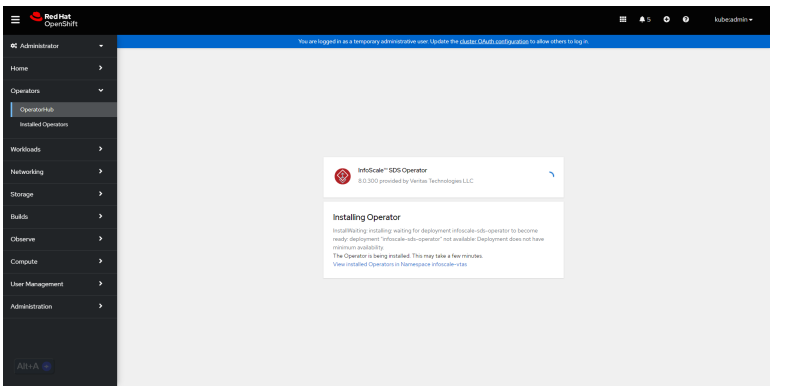
**Note:** Veritas recommends these configurations. Selecting Manual as the Install plan avoids automatic updates of the operator.

---

Click **Approve** for **Manual approval required**.

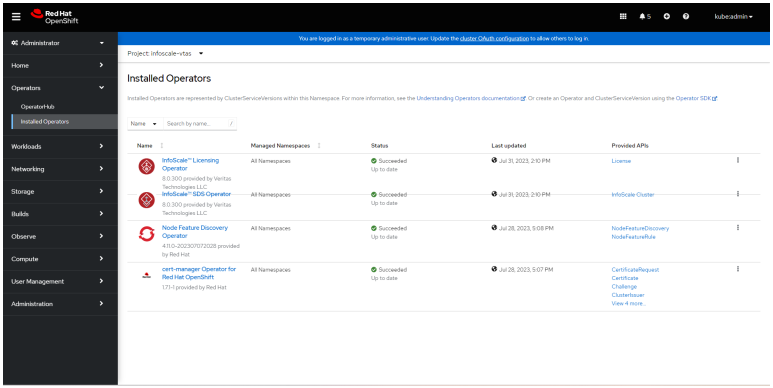



**10** InfoScale installation begins as under.



**11** Wait till installation is complete. InfoScale Licensing Operator and Node Feature Discovery Operator along with InfoScale SDS operator are installed in the namespace you provide. Cert-manager, InfoScale Licensing Operator, and Node Feature Discovery Operator are the dependencies for InfoScale installation. If any of these dependencies are already installed in the namespace you provide or `openshift-operators`, installation is skipped. License must be applied before installing InfoScale.

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



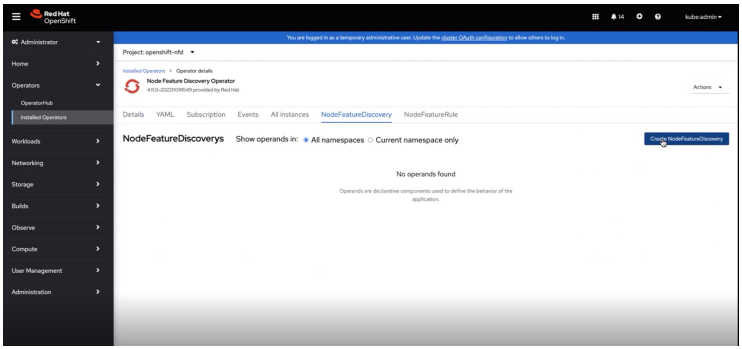

**13** After all these Operators are installed successfully, click **NodeFeatureDiscovery** in **Provided APIs**.

---

**Note:** If NFD instance is already created, skip all steps related to Node Feature Discovery.

---

**14** In **Node Feature Discovery Operator**, click **Create NodeFeatureDiscovery** in the upper-right corner of the screen.




**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** You can now apply the license. Click **Operators > Installed Operators** in the left frame.

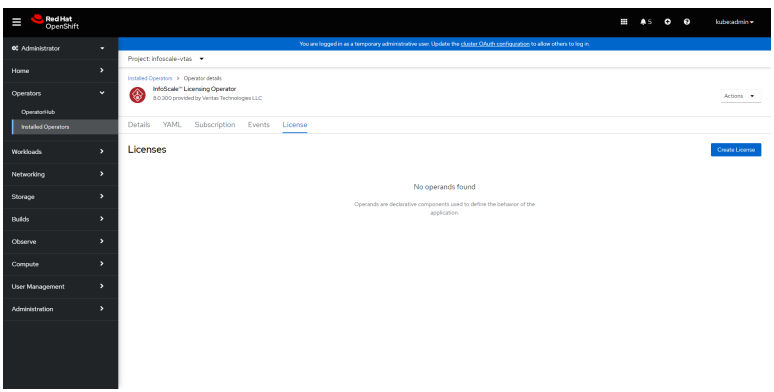
**17** Click **License** in **Provided APIs**.

---

**Note:** With a single license of Storage or Enterprise edition, you can install multiple InfoScale clusters.

---

**18** In the screen that opens, click **Create License** in the upper-right corner of your screen.



**19** Assign a name. Use the cluster name to associate the license with the cluster. Optionally, Enter IP address of the VIOM server on your system in **LicenseServer**.

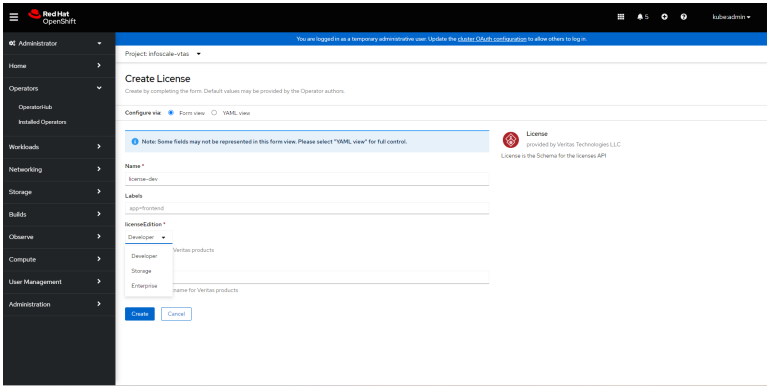
**20** In **License Edition**, select the type of license.

See “Licensing” on page 15.

---

**Note:** If you want to install and configure Disaster Recovery (DR), you must have Enterprise type of license.

---




**21** Click **Create** to apply the license.

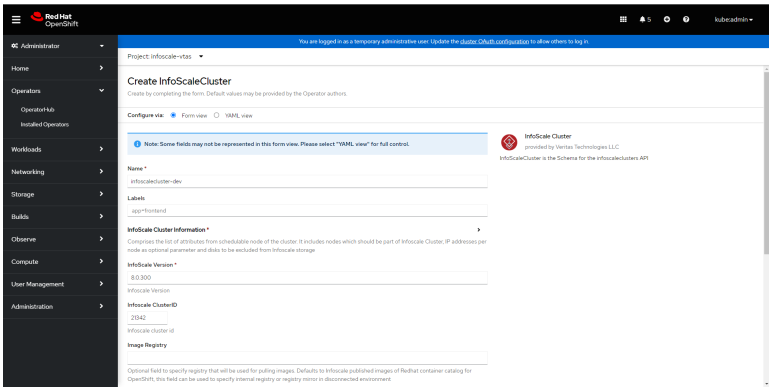
**22** After the license is successfully applied, it is listed in **Installed Operators > Operator details**.

**23** You can click the license name for details.

After the license is applied, you can configure InfoScale clusters.

**24** Click **Operators > Installed Operators** in the left frame. You can now create multiple InfoScale clusters. Click **InfoScale Cluster** in **Provided APIs**.

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

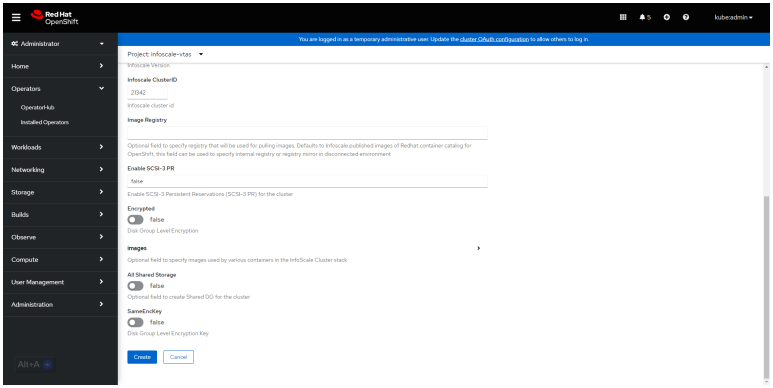



**26** Choose a project to create InfoScale cluster. Enter **Name**, **Namespace**, and **InfoScale ClusterID** for the cluster. If you are installing in an airgapped environment, enter **Image Registry** for pulling images. To install on a system with Internet connectivity, do not enter any value for **Image Registry**.

---

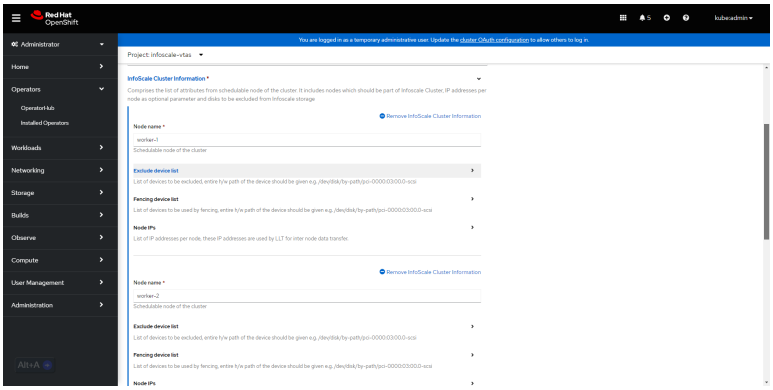
**Note:** If you do not enter **InfoScale ClusterID**, a **ClusterID** is randomly generated and assigned. The **ClusterID** is suffixed to the disk group.

---




See See “Considerations for configuring cluster or adding nodes to an existing cluster” on page ?. to change values of parameters.

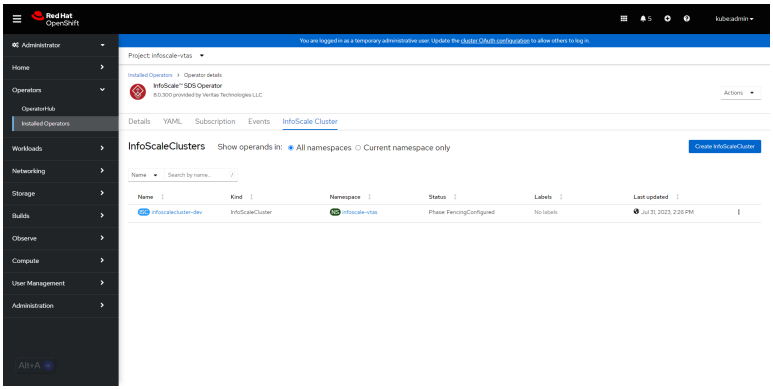
**27** Click **InfoScale Cluster Information**. Enter information about the nodes here. Enter **Node name** . Optionally, you can enter IP addresses of nodes in **Node IPs** and the device path of the disk that you want to exclude from the InfoScale disk group in **Exclude-device list**. You can also add fencing devices in **Fencing device list**. For each node, you must add two IP addresses.



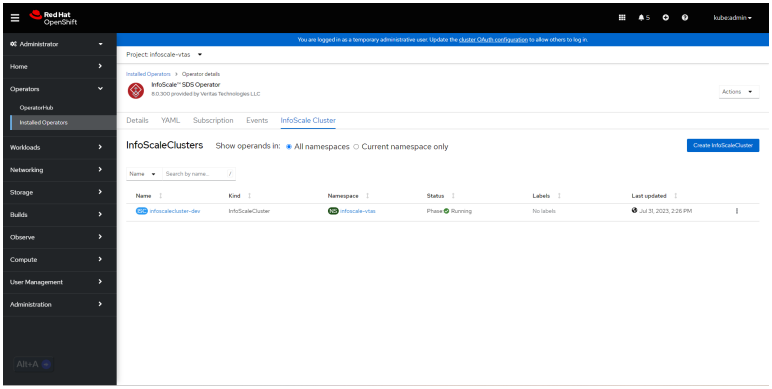

**Note:** OpenShift cluster must have at least two nodes as minimum two nodes are needed to form a cluster.

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

**29** Click **Create** to create an InfoScale cluster. Cluster formation begins. Watch the status message. It changes to FencingConfigured,




The status then changes to DgCreated and finally Running as under.




**30** If you want to create more clusters, run steps 25 to 29.

After a successful deployment of InfoScale, the diskgroup is automatically created.

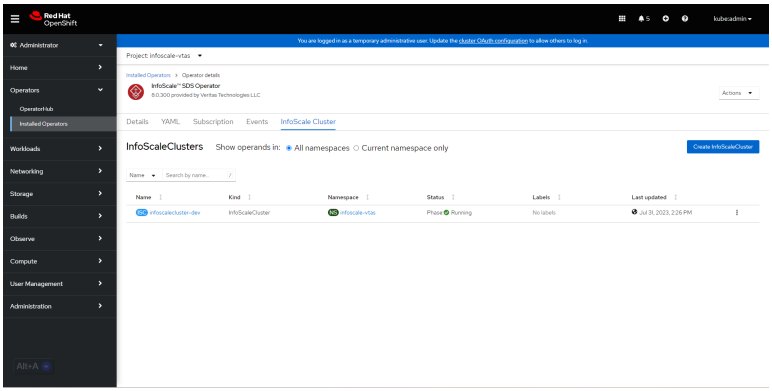
## Adding Nodes to an InfoScale cluster by using OLM

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

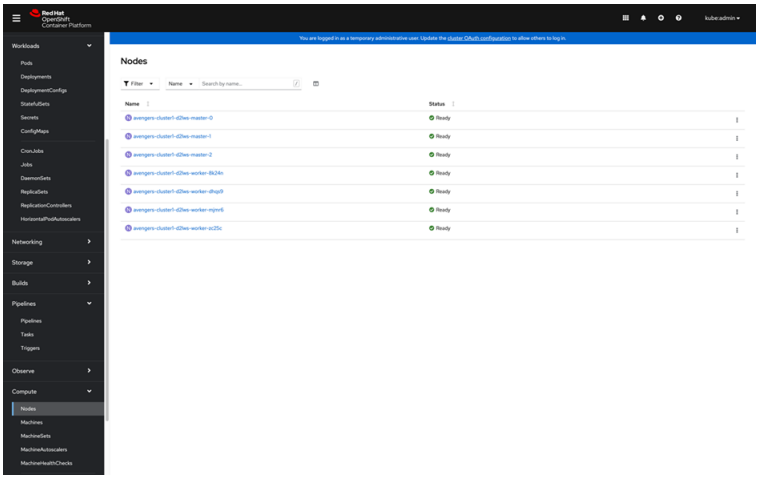
**1** Connect to the OpenShift console and access the Catalog menu.

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

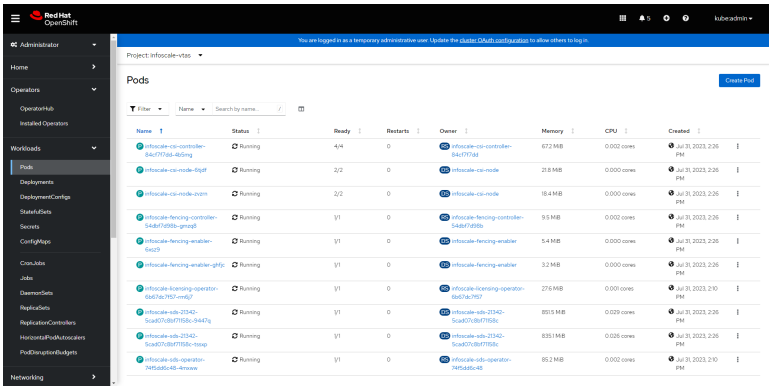
**3** Review the status of the cluster to which you want to add nodes. The Status of the cluster must be 'Running'.




**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'

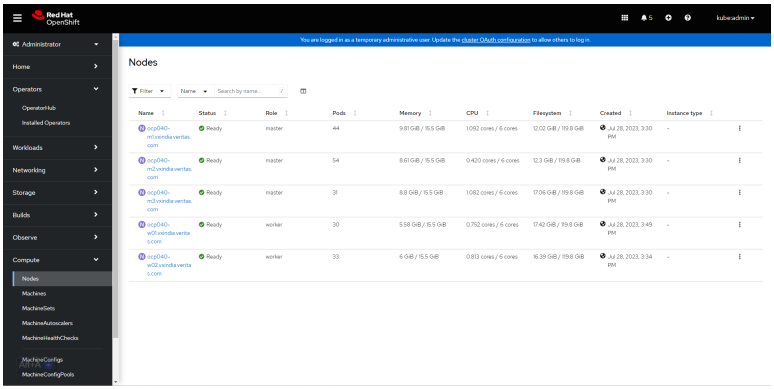


5 Click **Workload > Pods** in the left frame. Review status of the Pods. Pods must be 'Running'.




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.

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.

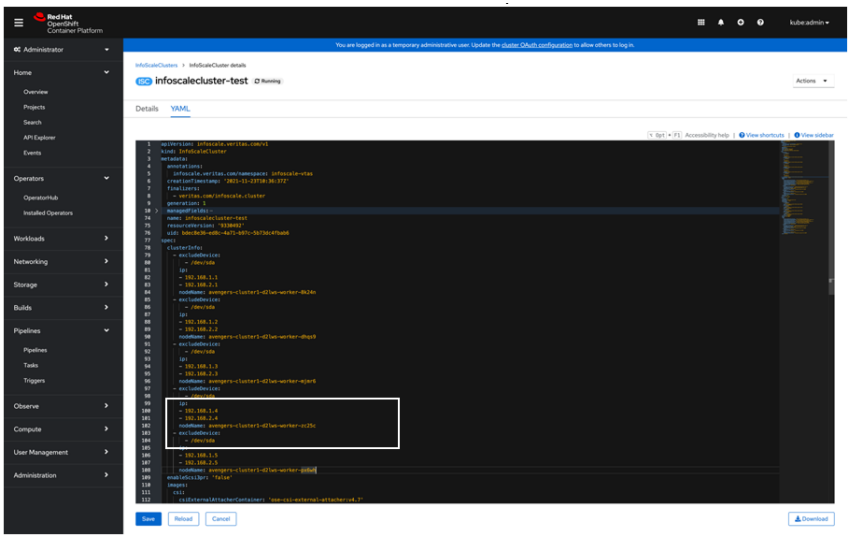
10 Click **YAML** in the screen that opens.

**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**.

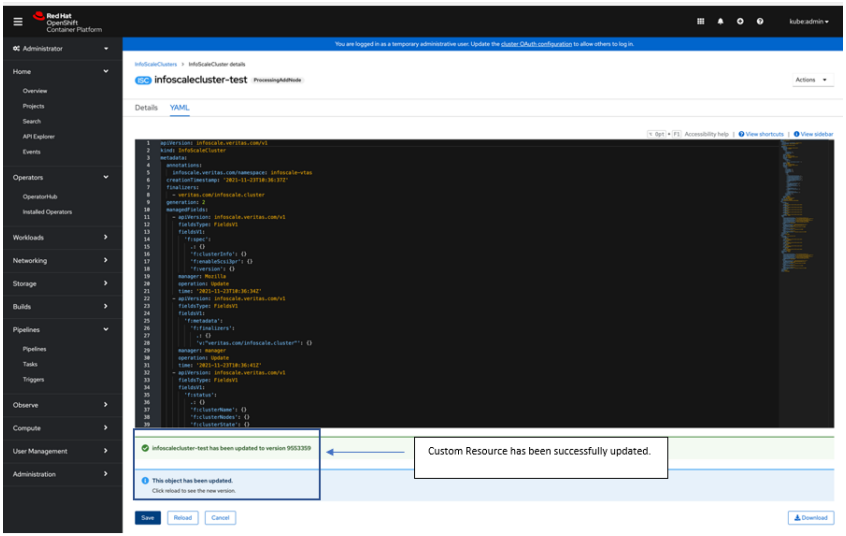
---

**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

---

**Note:** If you want to retain the disk group and re-use the Volumes you created, ensure that you note the `ClusterID` before undeploying and uninstalling InfoScale. You must use this `ClusterID` while re-installing InfoScale. Else, clean the disks before uninstalling.

---

---

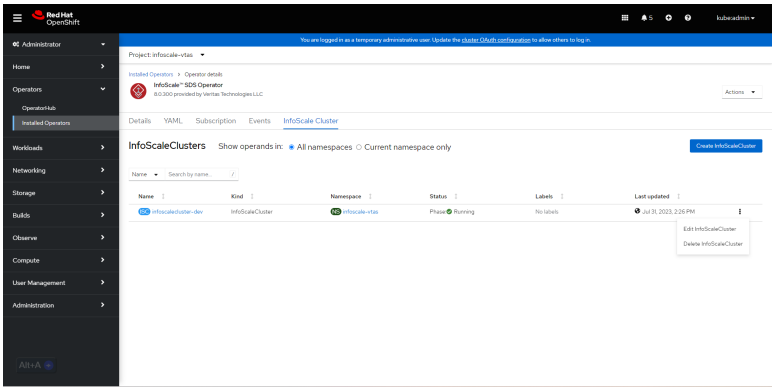
**Note:** Below images are for reference only. Ensure that you select the appropriate versions wherever required.

---

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

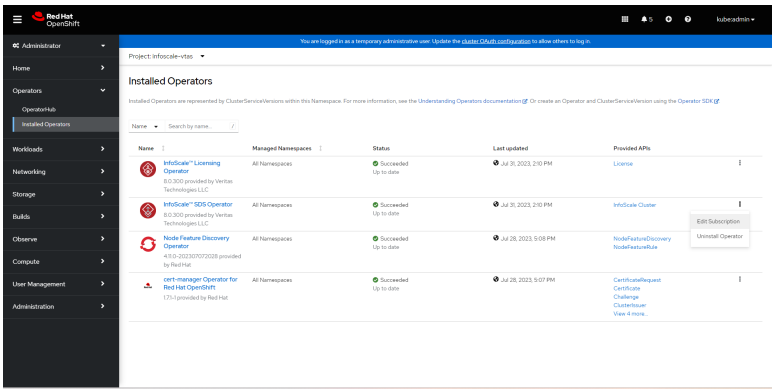
**2** In the left frame, select **Operators > Installed Operators**. Click **InfoScale Cluster** under **Provided APIs**.

3 The installed and deployed InfoScale clusters are listed. Click the Actions menu (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 After all InfoScale clusters are deleted, click **Operators > Installed Operators**. All Installed Operators are listed.




6 Click the Actions menu (three vertical dots) on the right of the screen for InfoScale SDS Operator. Select and click **Delete**. Confirm Delete.

7 Similarly, delete InfoScale Licensing 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_8.0.310.tar.gz`

1. Download `YAML_8.0.310.tar.gz` from the Veritas Download Center.
2. Unzip and untar the file. A folder `/YAML/OpenShift/OLM/` is created and all the files that 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`. cert-manager (Red Hat-certified) must be already installed for a successful installation of InfoScale.

3. If you have installed the cert manager in a namespace other than cert-manager, openshift-cert-manager, or openshift-operators; edit subscription yaml for `lico`, `iso` and `dr` operators and add

```
name: CERT_MANAGER_NS
value: <namespace where cert manager is installed>
```

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 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 <Namespace>
```

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

```
namespace/<Namespace> created
```

**2** Optionally, if you want to change the default kubelet path, edit `/YAML/OpenShift/OLM/infoscale-sub.yaml` as under.

```
env:

   - name: KUBELET_PATH

     value: <enter the new path>
```

The default path is `/var/lib/kubelet`.

---

**Note:** Do not change the kubelet path after clusters are configured.

---

**3** 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 **<Namespace>** to **openshift-operators**. To install the latest bundle, modify **startingCSV** to **infoscale-sds-operator.v8.0.310**.

---

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

Following output indicates a successful command run.

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

**4** Run the following command on the bastion node to deploy InfoScale licensing operator subscription.

```
oc create -f licensing-sub.yaml
```

Following output indicates a successful command run.

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

**5** 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
```

**6** Run the following command on the bastion node.

```
oc get sub,og -n <Namespace>
```

Following output indicates a successful command run.

```
NAME
subscription.operators.coreos.com/infoscale-licensing-operator-sub
subscription.operators.coreos.com/infoscale-sds-operator-sub



PACKAGE                          SOURCE                CHANNEL
infoscale-licensing-operator   certified-operators   stable
infoscale-sds-operator         certified-operators   stable


NAME                                                          AGE
operatorgroup.operators.coreos.com/infoscale-sds-opgroup   82s
```

**7** 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          CSV                      APPROVAL APPROVED
<Namespace> install-59vpz nfd.4.12.0-202307170916 Manual   false
<Namespace> install-wspfv infoscale-sds-operator. Manual   false
                             v8.0.310
```

**8** Run the following command on the bastion node.

---

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

---

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

Following output indicates a successful command run.

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

**9** 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
<Namespace>   <installation-name>
```

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

**10** 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
InfoScale Licensing Operator
infoscale-sds-operator.{version}       InfoScale™ SDS Operator
nfd.4.12.0-202307170916                Node Feature Discovery Operator
openshift-cert-manager.v1.12.0         cert-manager Operator for
                                       Red Hat OpenShift




VERSION               REPLACES PHASE
8.0.310                        Installing
8.0.310                        Installing
4.12.0-202307170916            Installing
1.11-1                         Succeeded
```

**11** Run the following command on the bastion node again till all operators are installed successfully.

```
oc get csv
```

Review output as under.

```
NAME                                      DISPLAY
infoscale-licensing-operator.v8.0.310     InfoScale Licensing Operator
infoscale-sds-operator.{version}          InfoScale™ SDS Operator
nfd.4.12.0-202307170916                   Node Feature Discovery Operator
openshift-cert-manager.v1.12.0            cert-manager Operator for
                                          Red Hat OpenShift




VERSION               REPLACES      PHASE
8.0.310                             Succeeded
8.0.310                             Succeeded
4.12.0-202307170916                 Succeeded
1.11-1                              Succeeded
```

**12** Access your web console. Follow steps 11 to 15 in Installing from OperatorHub by using web console to install NodeFeatureDiscovery.

**13** Run the following command to check the status of all operator pods in <Namespace>.

---

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

---

```
oc get pods -n <Namespace>
```

```
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-sds-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
```

### Applying Licenses

1 Edit `/YAML/OpenShift/license_cr.yaml` for the license edition. Optionally, you can change the license name. The default license edition is Developer. You can change the `licenseEdition`. If you want to configure Disaster Recovery (DR), you must have Enterprise as the license edition. To configure multiple InfoScale clusters, you must have Storage or Enterprise edition.

See "Licensing" on page 15.

```
apiVersion: vlic.veritas.com/v1
kind: License
metadata:
  name: license-dev
spec:
  # valid licenseEdition values are Developer,
            Storage or Enterprise
  licenseEdition: "Developer"
  licenseServer: <Optional - IP address of
                 the VIOM server on your system>
```

2 Run `oc create -f /YAML/OpenShift/license_cr.yaml` on the bastion node.

3 Run `oc get licenses` on the bastion node to verify whether licenses have been successfully applied.

An output similar to the following indicates that `license_cr.yaml` is successfully applied.

```
NAME          NAMESPACE   LICENSE-EDITION   AGE
license                   DEVELOPER         27s
```

## Configuring cluster

Complete the following steps for each cluster.

**Note:** Ensure that you add an OpenShift node only to a single InfoScale cluster.

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

```
metadata:
 name: < Assign a name to this cluster >
 namespace: < The namespace where you want to
                      create this cluster>


spec:
 clusterID: <Optional- Enter an ID for this cluster.
                The ID can be any number between 1 and 65535 >
 isSharedStorage: true
 - 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.

 fencingDevice: ["<Hardware path to the first fencing device>",
                 "<Hardware path to the second fencing device>",
                 "<Hardware path to the third fencing device>", ]
  enableScsi3pr:<Enter True to enable SCSI3 persistent reservation>
  encrypted: false
```

```
  sameEnckey: false
```

---

**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. You can choose to rename `cr.yaml`. If you rename the file, ensure that you use that name in the next step.

---

**Note:** Veritas recommends renaming `cr.yaml` and maintaining a custom resource for each cluster. The renamed `cr.yaml` is used to add more nodes to that InfoScale cluster.

---

3. Run the following command on the bastion node.

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

4. Run the following command on the bastion node to know the name and namespace of the cluster. Run the command for infoscale-vtas also.

   ```
   oc get infoscalecluster -A
   ```

   Use the namespace from the output similar to the following.

   ```
   NAMESPACE   NAME                VERSION   CLUSTERID   STATE   AGE
   .
   .
   <Namespace> <Name of the
                InfoScale cluster> 8.0.310  <Cluster ID> Running 25h
   .
   .
   ```

5. Run the following command on the bastion node to verify whether the pods are created successfully. Run the command for infoscale-vtas also.

   ```
   oc get pods -n <Namespace>
   ```

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

   ```
   NAME                                    READY STATUS  RESTARTS AGE
   ```

```
infoscale-csi-controller-1234-0      5/5   Running 0          19h
infoscale-csi-node-1234-gf9pf        2/2   Running 0          19h
infoscale-csi-node-1234-gg5dq        2/2   Running 0          18h
infoscale-csi-node-1234-nmt85        2/2   Running 0          18h
infoscale-csi-node-1234-r6jv8        2/2   Running 0          19h
infoscale-csi-node-1234-w5bln        2/2   Running 2          19h
infoscale-fencing-controller-
               234-864468775c-4sbxw 1/1   Running 0          18h
infoscale-fencing-enabler-1234-8b65z 1/1   Running 0          19h
infoscale-fencing-enabler-1234-bkbbh 1/1   Running 3 (18h ago) 18h
infoscale-fencing-enabler-1234-jvzjk 1/1   Running 5 (18h ago) 18h
infoscale-fencing-enabler-1234-pxfmt 1/1   Running 4 (18h ago) 19h
infoscale-fencing-enabler-1234-qmjrv 1/1   Running 0          19h
infoscale-sds-1234-e383247e62b56585-
                              2xxvh 1/1   Running 1          19h
infoscale-sds-1234-e383247e62b56585-
                              cnvkg 1/1   Running 0          18h
infoscale-sds-1234-e383247e62b56585-
                              l5z7m 1/1   Running 0          19h
infoscale-sds-1234-e383247e62b56585-
                              xlkf8 1/1   Running 0          18h
infoscale-sds-1234-e383247e62b56585-
                              zkpgt 1/1   Running 0          19h
infoscale-sds-operator-bb55cfc4d-
                              pclt5 1/1   Running 0          18h
infoscale-licensing-operator-
                  5fd897f68f-7p2f7  1/1   Running 0          18h
nfd-controller-manager-6bbf6df4d9-
                              dbxgl 2/2   Running 2          20h
nfd-master-2h7x6                    1/1   Running 0          19h
nfd-master-kclkq                    1/1   Running 0          19h
nfd-master-npjzm                    1/1   Running 0          19h
nfd-worker-8q4lz                    1/1   Running 0          19h
nfd-worker-cvkqp                    1/1   Running 0          19h
nfd-worker-js7tt                    1/1   Running 1          19h
```

InfoScale SDS pods are created in the namespace you specify in `cr.yaml`. Fencing and CSI pods are created in the operator namespace.

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

   ```
   oc get nodes
   ```

   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 -A
   ```

   See the State in the output similar to the following -

   ```
   NAMESPACE   NAME                VERSION   CLUSTERID   STATE   AGE
   .
   .
   <Namespace> <Name of the
                InfoScale cluster> 8.0.310  <Cluster ID> Running 25h
   .
   .
   ```

**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.

---

```
metadata:
 name: < Assign a name to this cluster >
 namespace: < The namespace where you want to
                      create this cluster>


spec:
 clusterID: <Optional- Enter an ID for this cluster.
                 The ID can be any number between 1 and 65535 >
 isSharedStorage: true
 - 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.
```

```
fencingDevice: ["<Hardware path to the first fencing device>",
                "<Hardware path to the second fencing device>",
                "<Hardware path to the third fencing device>", ]
encrypted: false
sameEnckey: false
```

**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 -A`

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

```
NAMESPACE   NAME            VERSION   CLUSTERID STATE             AGE
.
.
<Namespace> <Name of the              <Cluster
          InfoScale cluster> 8.0.310  ID>       ProcessingAddNode 25h
.
.
```

b. `oc describe infoscalecluster -n <Namespace>`

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

```
Cluster Name:  <Name of the cluster>
  Cluster Nodes:
    Exclude Device:
      <Excluded device path 1>
      <Excluded device path 2>
    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 <Namespace>
```

Output similar to the following indicates a successful creation.

```
NAME                                    READY STATUS  RESTARTS AGE
infoscale-csi-controller-1234-0         5/5   Running 0        19h
infoscale-csi-node-1234-gf9pf           2/2   Running 0        19h
infoscale-csi-node-1234-gg5dq           2/2   Running 0        18h
infoscale-csi-node-1234-nmt85           2/2   Running 0        18h
infoscale-csi-node-1234-r6jv8           2/2   Running 0        19h
infoscale-csi-node-1234-w5bln           2/2   Running 2        19h
infoscale-fencing-controller-
                234-864468775c-4sbxw 1/1   Running 0        18h
infoscale-fencing-enabler-1234-8b65z 1/1   Running 0        19h
infoscale-fencing-enabler-1234-bkbbh 1/1   Running 3 (18h ago) 18h
infoscale-fencing-enabler-1234-jvzjk 1/1   Running 5 (18h ago) 18h
infoscale-fencing-enabler-1234-pxfmt 1/1   Running 4 (18h ago) 19h
infoscale-fencing-enabler-1234-qmjrv 1/1   Running 0        19h
infoscale-sds-1234-e383247e62b56585-
                                2xxvh 1/1   Running 1        19h
infoscale-sds-1234-e383247e62b56585-
                                cnvkg 1/1   Running 0        18h
infoscale-sds-1234-e383247e62b56585-
                                l5z7m 1/1   Running 0        19h
infoscale-sds-1234-e383247e62b56585-
                                xlkf8 1/1   Running 0        18h
infoscale-sds-1234-e383247e62b56585-
                                zkpgt 1/1   Running 0        19h
infoscale-sds-operator-bb55cfc4d-
                                pclt5 1/1   Running 0        18h
infoscale-licensing-operator
                  -5fd897f68f-7p2f7  1/1   Running 0        18h
nfd-controller-manager-6bbf6df4d9-
                                dbxgl 2/2   Running 2        20h
nfd-master-2h7x6                      1/1   Running 0        19h
nfd-master-kclkq                      1/1   Running 0        19h
nfd-master-npjzm                      1/1   Running 0        19h
nfd-worker-8q4lz                      1/1   Running 0        19h
nfd-worker-cvkqp                      1/1   Running 0        19h
```

```
nfd-worker-js7tt                        1/1   Running 1        19h
```

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

```
oc get infoscalecluster -A
```

See the State in the output similar to the following -

```
NAMESPACE   NAME                VERSION   CLUSTERID   STATE   AGE
.
.
<Namespace> <Name of the
             InfoScale cluster> 8.0.310  <Cluster ID> Running 25h
.
.
```

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

```
oc describe infoscalecluster <Name of the cluster> -n <Namespace>
```

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

```
Status:
  Cluster Name:  <Name of the cluster>
  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.

**Note:** If you want to retain the disk group and re-use the Volumes you created, ensure that you note the `ClusterID` before undeploying and uninstalling InfoScale. You must use this `ClusterID` while re-installing InfoScale. Else, clean the disks before uninstalling.

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

   ```
   oc delete infoscalecluster -n <Namespace> <Name of the cluster>
   ```

   **Note:** Run the command for each cluster.

2. Delete NodeFeatureDiscovery custom resource by logging in to web console
3. Run the following command on the bastion node to delete license.

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

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

   ```
   oc delete og -n <Namespace> infoscale-opgroup
   ```

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

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

5. 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 <Namespace> -all
   ```

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

   - `oc get csv |egrep "license|Node Feature|Infoscale"|awk '{print $1}'`

     Use the csv_name and clusterserviceversion returned from this command in the following commands.

   - `oc delete csv <csv_name> -n <Namespace>`

     **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.

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

- `oc get crd | egrep 'cert-manager|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>`

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

```
oc delete ns <Namespace>
```

**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. When fencing is configured, certain keys are configured. Those must also be deleted after uninstallation. Run `./clearkeys.sh <Path to the first disk>, <Path to the second disk>,...` to remove stale keys that might have remained.

# Installing by using YAML

Complete the following steps -

1. Download `YAML_8.0.310.tar.gz` from the Veritas Download Center.
2. Unzip and untar the file. The folders `/YAML/OpenShift/`, `/YAML/OpenShift/air-gapped-systems` , `/YAML/DR`, and `/YAML/Kubernetes` are created. Each folder contains files required for installation.
3. If you have installed the cert manager in a namespace other than cert-manager, openshift-cert-manager, or openshift-operators; edit `iso.yaml`, `lico.yaml`, and `dro.yaml` and add

```
name: CERT_MANAGER_NS
value: <namespace where cert manager is installed>
```

**Note:** cert-manager and Node Feature Discovery (both Red Hat-certified) must be already installed for a successful installation of InfoScale. To know how to install, see steps 10 to 15 of Installing from OperatorHub by using web console. Node Feature Discovery custom resource must also be created in the same namespace where Node Feature Discovery is installed.

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 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.

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, unzip, and untar `YAML_8.0.310.tar.gz` , all the files that are required for installation are available.

---

### Adding image-related information

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

   Replace **image: 192.168.1.21/veritas/infoscale-sds-operator:8.0.310-<rhel8/ol8>** with **image: <IP address of custom registry>/infoscale-sds-operator:8.0.310-<rhel8/ol8>**.

2. Edit `/YAML/Openshift/lico.yaml` as under -

   Replace **image: 192.168.1.21/veritas/infoscale-licensing-operator:8.0.310-ol8** with **image: <IP address of custom registry>/infoscale-licensing-operator:8.0.310-<rhel8/ol8>**.

3. Edit `/YAML/DR/Openshift/dro_deployment.yaml` as under -

   Replace **image: 192.168.1.21/veritas/infoscale-dr-manager:8.0.310-rhel8** with **image: <IP address of custom registry>/veritas/infoscale-dr-manager:8.0.310-<rhel8/ol8>**.

### Applying Licenses

1. Run `oc create -f /YAML/OpenShift/lico.yaml` on the bastion node.

2. Run `oc get pods -n <Namespace>|grep -i licensing` on the bastion node to verify whether `lico.yaml` is successfully applied.

   An output similar to the following indicates that `lico.yaml` is successfully applied.

```
NAME                                    READY STATUS  RESTARTS AGE
infoscale-licensing-operator
                  -fbd8c7dc4-rcfz5    1/1   Running 0        2m
```

3 After `lico.yaml` is successfully applied, Wait till the licensing endpoints are activated.

Run `oc describe service/lico-webhook-service -n <Namespace>|grep Endpoints` on the master node and review the output.

4 Run the command again till you get an output in the following format.

```
Endpoints: <IP address of the endpoint>:<Port number>
```

5 Edit `/YAML/OpenShift/license_cr.yaml` for the license edition. Optionally, you can change the license name. The default license edition is Developer. You can change the `licenseEdition`. If you want to configure Disaster Recovery (DR), you must have Enterprise as the license edition.

See "Licensing" on page 15.

```
apiVersion: vlic.veritas.com/v1
kind: License
metadata:
  name: license-dev
spec:
  # valid licenseEdition values are Developer,
            Storage or Enterprise
  licenseEdition: "Developer"
  licenseServer: <Optional - IP address of
                  the VIOM server on your system>
```

6 Run `oc create -f /YAML/OpenShift/license_cr.yaml` on the bastion node.

7 Run `oc get licenses` on the bastion node to verify whether licenses have been successfully applied.

An output similar to the following indicates that `license_cr.yaml` is successfully applied.

```
NAME          NAMESPACE   LICENSE-EDITION   AGE
license                   DEVELOPER         27s
```

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 <Namespace>|grep infoscale-sds-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-sds-operator-bb55cfc4d-pclt5 1/1   Running 0        20h
```

## Configuring cluster

Complete the following steps for each cluster.

**Note:** Ensure that you add an OpenShift node only to a single InfoScale cluster.

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

```
metadata:
 name: < Assign a name to this cluster >
 namespace: < The namespace where you want to
                      create this cluster>


spec:
 clusterID: <Optional- Enter an ID for this cluster.
                The ID can be any number between 1 and 65535 >
 isSharedStorage: true
 - 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.

 fencingDevice: ["<Hardware path to the first fencing device>",
                 "<Hardware path to the second fencing device>",
                 "<Hardware path to the third fencing device>", ]
  enableScsi3pr:<Enter True to enable SCSI3 persistent reservation>
  encrypted: false
  sameEnckey: false
```

**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. You can choose to rename `cr.yaml`. If you rename the file, ensure that you use that name in the next step.

**Note:** Veritas recommends renaming `cr.yaml` and maintaining a custom resource for each cluster. The renamed `cr.yaml` is used to add more nodes to that InfoScale cluster.

3. Run the following command on the bastion node.

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

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

   ```
   oc get infoscalecluster -A
   ```

   Use the namespace from the output similar to the following.

   ```
   NAMESPACE   NAME                VERSION   CLUSTERID   STATE   AGE
   .
   .
   <Namespace> <Name of the
                InfoScale cluster> 8.0.310  <Cluster ID> Running 25h
   .
   .
   ```

5. Run the following command on the bastion node to verify whether the pods are created successfully. Run the command for infoscale-vtas also.

   ```
   oc get pods -n <Namespace>
   ```

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

   ```
   NAME                                READY STATUS  RESTARTS AGE
   infoscale-csi-controller-1234-0     5/5   Running 0        19h
   infoscale-csi-node-1234-gf9pf       2/2   Running 0        19h
   infoscale-csi-node-1234-gg5dq       2/2   Running 0        18h
   infoscale-csi-node-1234-nmt85       2/2   Running 0        18h
   infoscale-csi-node-1234-r6jv8       2/2   Running 0        19h
   infoscale-csi-node-1234-w5bln       2/2   Running 2        19h
   infoscale-fencing-controller-
                  234-864468775c-4sbxw 1/1   Running 0        18h
   infoscale-fencing-enabler-1234-8b65z 1/1   Running 0        19h
   infoscale-fencing-enabler-1234-bkbbh 1/1   Running 3 (18h ago) 18h
   infoscale-fencing-enabler-1234-jvzjk 1/1   Running 5 (18h ago) 18h
   infoscale-fencing-enabler-1234-pxfmt 1/1   Running 4 (18h ago) 19h
   infoscale-fencing-enabler-1234-qmjrv 1/1   Running 0        19h
   ```

```
infoscale-sds-1234-e383247e62b56585-
                              2xxvh 1/1   Running 1        19h
infoscale-sds-1234-e383247e62b56585-
                              cnvkg 1/1   Running 0        18h
infoscale-sds-1234-e383247e62b56585-
                              l5z7m 1/1   Running 0        19h
infoscale-sds-1234-e383247e62b56585-
                              xlkf8 1/1   Running 0        18h
infoscale-sds-1234-e383247e62b56585-
                              zkpgt 1/1   Running 0        19h
infoscale-sds-operator-bb55cfc4d-
                              pclt5 1/1   Running 0        18h
infoscale-licensing-operator
                  -5fd897f68f-7p2f7  1/1   Running 0        18h
nfd-controller-manager-6bbf6df4d9-
                              dbxgl 2/2   Running 2        20h
nfd-master-2h7x6                    1/1   Running 0        19h
nfd-master-kclkq                    1/1   Running 0        19h
nfd-master-npjzm                    1/1   Running 0        19h
nfd-worker-8q4lz                    1/1   Running 0        19h
nfd-worker-cvkqp                    1/1   Running 0        19h
nfd-worker-js7tt                    1/1   Running 1        19h
```

InfoScale SDS pods are created in the namespace you specify in `cr.yaml`. Fencing and CSI pods are created in the operator namespace.

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

   ```
   oc get nodes
   ```

   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 -A
   ```

   See the State in the output similar to the following -

   ```
   NAMESPACE   NAME                VERSION   CLUSTERID   STATE   AGE
   .
   .
   <Namespace> <Name of the
                InfoScale cluster> 8.0.310  <Cluster ID> Running 25h
   .
   .
   ```

**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.

---

```
metadata:
 name: < Assign a name to this cluster >
 namespace: < The namespace where you want to
                      create this cluster>


spec:
 clusterID: <Optional- Enter an ID for this cluster.
                 The ID can be any number between 1 and 65535 >
 isSharedStorage: true
 - 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.

fencingDevice: ["<Hardware path to the first fencing device>",
                "<Hardware path to the second fencing device>",
                "<Hardware path to the third fencing device>", ]
 enableScsi3pr:<Enter True to enable SCSI3 persistent reservation>
 encrypted: false
 sameEnckey: false
```

**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 -A`

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

```
NAMESPACE   NAME            VERSION   CLUSTERID STATE             AGE
.
.
<Namespace> <Name of the              <Cluster
          InfoScale cluster> 8.0.310  ID>       ProcessingAddNode 25h
.
.
```

b. `oc describe infoscalecluster -n <Namespace>`

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

```
Cluster Name:  <Name of the cluster>
  Cluster Nodes:
    Exclude Device:
      <Excluded device path 1>
      <Excluded device path 2>
    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 <Namespace>
```

Output similar to the following indicates a successful creation.

```
NAME                                    READY STATUS  RESTARTS AGE
infoscale-csi-controller-1234-0         5/5   Running 0        19h
infoscale-csi-node-1234-gf9pf           2/2   Running 0        19h
infoscale-csi-node-1234-gg5dq           2/2   Running 0        18h
infoscale-csi-node-1234-nmt85           2/2   Running 0        18h
infoscale-csi-node-1234-r6jv8           2/2   Running 0        19h
infoscale-csi-node-1234-w5bln           2/2   Running 2        19h
infoscale-fencing-controller-
                234-864468775c-4sbxw 1/1   Running 0        18h
infoscale-fencing-enabler-1234-8b65z 1/1   Running 0        19h
infoscale-fencing-enabler-1234-bkbbh 1/1   Running 3 (18h ago) 18h
infoscale-fencing-enabler-1234-jvzjk 1/1   Running 5 (18h ago) 18h
infoscale-fencing-enabler-1234-pxfmt 1/1   Running 4 (18h ago) 19h
infoscale-fencing-enabler-1234-qmjrv 1/1   Running 0        19h
infoscale-sds-1234-e383247e62b56585-
                                2xxvh 1/1   Running 1        19h
infoscale-sds-1234-e383247e62b56585-
                                cnvkg 1/1   Running 0        18h
infoscale-sds-1234-e383247e62b56585-
                                l5z7m 1/1   Running 0        19h
infoscale-sds-1234-e383247e62b56585-
                                xlkf8 1/1   Running 0        18h
infoscale-sds-1234-e383247e62b56585-
                                zkpgt 1/1   Running 0        19h
infoscale-sds-operator-bb55cfc4d-
                                pclt5 1/1   Running 0        18h
infoscale-licensing-operator
                  -5fd897f68f-7p2f7  1/1   Running 0        18h
nfd-controller-manager-6bbf6df4d9-
                                dbxgl 2/2   Running 2        20h
nfd-master-2h7x6                      1/1   Running 0        19h
nfd-master-kclkq                      1/1   Running 0        19h
nfd-master-npjzm                      1/1   Running 0        19h
nfd-worker-8q4lz                      1/1   Running 0        19h
nfd-worker-cvkqp                      1/1   Running 0        19h
```

```
nfd-worker-js7tt                        1/1   Running 1        19h
```

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

```
oc get infoscalecluster -A
```

See the State in the output similar to the following -

```
NAMESPACE   NAME                VERSION   CLUSTERID   STATE   AGE
.
.
<Namespace> <Name of the
             InfoScale cluster> 8.0.310  <Cluster ID> Running 25h
.
.
```

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

```
oc describe infoscalecluster <Name of the cluster> -n <Namespace>
```

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

```
Status:
  Cluster Name:  <Name of the cluster>
  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.

**Note:** If you want to retain the disk group and re-use the Volumes you created, ensure that you note the `ClusterID` before undeploying and uninstalling InfoScale. You must use this `ClusterID` while re-installing InfoScale. Else, clean the disks before uninstalling.

- Run the following commands on the bastion node

  ```
  oc delete infoscalecluster -n <Namespace> <Name of the cluster>
  ```

  **Note:** Run the command for each cluster.

  The commands to clean up InfoScale components 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/license_cr.yaml
  oc delete -f /YAML/OpenShift/lico.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. When fencing is configured, certain keys are configured. Those must also be deleted after uninstallation. Run `./clearkeys.sh <Path to the first disk>, <Path to the second disk>,...` to remove stale keys that might have remained.

# Installing InfoScale in an air gapped system

An air gapped system is not connected to the Internet. It is therefore necessary to copy the images to the local registry. After the images are copied and prerequisites are met, installation is much similar to the installation on a system with internet connectivity.

## Prerequisites to install by using YAML or OLM

A few operators are needed to install InfoScale by using OLM. As air gapped/restricted networks are disconnected from the internet, operators must be mirrored to local registry. For mirroring, it requires a host having internet as well as local registry connection, named mirror-host.

Complete the following steps on the `mirror-host` where `${LOCAL_REGISTRY}` is accessible.

For others, see https://docs.openshift.com/container-platform/4.11/installing/disconnected_install/installing-mirroring-disconnected.html

---

**Note:** In the following steps, ${LOCAL_REGISTRY} is on the same network. LOCAL_REGISTRY is a system connected to the mirror-host but disconnected from Internet,and has a registry setup.

---

1. Run the following command to set variables.

   ```
   export LOCAL_REGISTRY=<local_registry_host_name>:<local_registry_host_port>
   ```

2. Run the following commands to download and extract the latest `oc-mirror` plugin on the mirror host.

   ```
   wget -4 https://mirror.openshift.com/pub/openshift-v4/x86_64/clients/ocp/latest/oc-mirror.tar.gz

   tar -xvf oc-mirror.tar.gz -C /usr/local/bin/

   chmod 750 /usr/local/bin/oc-mirror
   ```

3. Download `YAML_8.0.310.tar.gz` from the Veritas Download Centre.

4. Extract `YAML_8.0.310.tar.gz` . A folder `YAML/OpenShift/air-gapped` is created and all files required for installation are available in the folder.

5. Run the following commands to login to registries.

   ```
   podman login registry.redhat.io

   podman login registry.connect.redhat.com

   podman login ${LOCAL_REGISTRY}
   ```

6. Run `oc mirror list operators --catalog <CATALOG> --package <OPERATOR>` to update `operators.packages.channels.name` in `YAML/OpenShift/air-gapped-systems/imageset-config.yaml`.

7. Edit `imageset-config.yaml` to

   - Replace `${LOCAL_REGISTRY}` with `<local_registry_host_name>:<local_registry_host_port>`.
   - Replace `${OCP_RELEASE}` with the `OCP version`.

**8** Run the following commands on the mirror host to mirror operators.

```
cd YAML/OpenShift/air-gapped-systems
```

```
oc mirror --config=./imageset-config.yaml
docker://${LOCAL_REGISTRY}/operators --continue-on-error
```

Review output as under.

```
Rendering catalog image "registry.lab.ocp.lan:5000
  /operators1/redhat/certified-operator-index:v4.10"
                             with file-based catalog
Rendering catalog image "registry.lab.ocp.lan:5000
  /operators1/redhat/redhat-operator-index:v4.10"
                           with file-based catalog
Writing image mapping to
 oc-mirror-workspace/results-1669176953/mapping.txt
Writing CatalogSource manifests to
            oc-mirror-workspace/results-1669176953
Writing ICSP manifests to
          oc-mirror-workspace/results-1669176953
```

---

**Note:** Here, `registry.lab.ocp.lan` is the local registry and 5000 is an indicative port number.

---

**9** Optionally, Update `oc-mirror-workspace/results-1669176953/catalogSource-redhat-operator-index.yaml` and `oc-mirror-workspace/results-1669176953/catalogSource-certified-operator-index.yaml`.

**10** Run the following command to copy `oc-mirror-workspace` to the bastion node.

```
scp -r oc-mirror-workspace ocp-svc:~/
```

**11** Run the following command on the bastion node to set `disableAllDefaultSources: true`. This disables all sources for the default catalogs.

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

**12** Run the following commands on the bastion node to deploy the image content source policy and catalog source.

```
oc create -f
oc-mirror-workspace/results-1669176953/imageContentSourcePolicy.yaml
```

```
oc create -f
oc-mirror-workspace/results-1669176953/catalogSource-redhat-operator-index.yaml
```

```
oc create -f
oc-mirror-workspace/results-1669176953/catalogSource-certified-operator-index.yaml
```

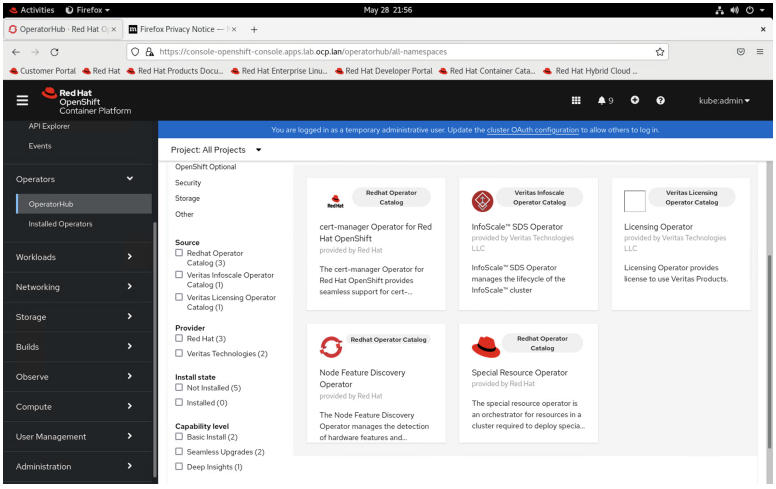
**13** Run the following commands on the bastion node to verify whether the resources are created successfully.

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

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

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

**14** Login to the OpenShift web console and check whether the mirrored operators are displayed in **Operators > OperatorHub** as under




**15** On the mirror host, Download, unzip, and untar `tools_8.0.310.tar.gz` from theVeritas Download Center. After you unzip and untar, `tools/setup_vtas_registry.sh` is automatically created.

**16** Run the following commands on the mirror host.

```
cd tools

OS_VER=<OS Version> # eg: 'rhel8.4'

KERNEL=<Kernel Version of worker>' # eg:
4.18.0-305.45.1.el8_4.x86_64'

./setup_vtas_registry.sh --airgap -o $OS_VER -k $KERNEL
```

A folder `infoscale_operand_images.tar` is automatically created.

**17** Copy `infoscale_operand_images.tar` from the mirror host to the bastion node.

**18** Run the following commands to load infoscale images onto local registry from bastion node

```
OS_VER=<OS Version> # eg: 'rhel8.4'

KERNEL=<Kernel Version of worker>' # eg:
4.18.0-305.45.1.el8_4.x86_64'

podman login ${LOCAL_REGISTRY}

./setup_vtas_registry.sh --airgap -o $OS_VER -k $KERNEL \ -c
${LOCAL_REGISTRY}/veritas -t infoscale_operand_images.tar
```

# Additional prerequisites to install by using yaml

**Complete the following steps on the bastion node**

**1** Copy the following images to the local registry.

```
${LOCAL_REGISTRY}/veritas
OS_VER=<OS Version> # eg: 'rhel8.4'
KERNEL=<Kernel Version of worker>'# eg: 4.18.0-305.45.1.el8_4.x86_64'
```

cert-manager Images

- quay.io/jetstack/cert-manager-cainjector:v1.11
- quay.io/jetstack/cert-manager-controller:v1.11
- quay.io/jetstack/cert-manager-webhook:v1.11

Operator Image

- registry.connect.redhat.com/veritas-technologies/infoscale-sds-operator:8.0.310-rhel8
- registry.connect.redhat.com/veritas-technologies/infoscale-licensing-operator:8.0.310-rhel8

To copy images to ${LOCAL_REGISTRY}

- If mirror host is connected to internet and `${LOCAL_REGISTRY}` both, run the following commands on mirror host

```
podman pull <src image>
podman tag <src image> <tgt image>
podman push <tgt image>

#If csi images push fail, try push with '--remove-signatures'
```

- If mirror host is connected to internet only, run the following commands on the mirror host

```
podman pull <src image>
podman save -o <image_name>.tar <src image> <tgt image>
-> Copy tar to ${LOCAL_REGISTRY} connected host (bastion node)
```

Run the following commands on the bastion node.

```
podman load -i <image_name>.tar
podman tag <src image> <tgt image>
podman push <tgt image>
```

**2** Update YAML

- Update `YAML/OpenShift/air-gapped/*.yaml` with your `${LOCAL_REGISTRY}`. Run the following commands.

```
cd YAML/OpenShift/air-gapped-systems
sed -i "s/192.168.1.21/${LOCAL_REGISTRY}/g" cert-manager.yaml
sed -i "s/192.168.1.21/${LOCAL_REGISTRY}/g" iso.yaml
sed -i "s/192.168.1.21/${LOCAL_REGISTRY}/g" lico.yaml
sed -i "s/192.168.1.21/${LOCAL_REGISTRY}/g" cr.yaml
```

---

**Note:** Ensure that you change the image path in all yamls to the local registry address.

---

## Installing from OperatorHub by using web console

1. Ensure that all Prerequisites to install by using YAML or OLM are met.
2. Follow steps listed in See " Installing from OperatorHub by using web console" on page 33..

---

**Note:** While creating the cluster, enter `<${LOCAL_REGISTRY}>/veritas` as the image path in web console.

---

## Installing from OperatorHub by using Command Line Interface (CLI)

1. Ensure that all See "Prerequisites to install by using YAML or OLM" on page 80. and See "Additional prerequisites to install by using yaml " on page 84. are met.
2. Follow steps listed in See "Installing from OperatorHub by using Command Line Interface (CLI)" on page 48..

## Installing by using YAML

Complete the following steps

1. Ensure that all See "Prerequisites to install by using YAML or OLM" on page 80. and See "Additional prerequisites to install by using yaml " on page 84.are met.
2. Install nfd operator from the web console.
3. Run the following command on the bastion node to install cert-manager by using yaml.

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

4. Follow steps listed in See "Installing by using YAML" on page 66.. Ensure that you replace the yaml path with `YAML/OpenShift/air-gapped-systems/`.

# Removing and adding back nodes to an Azure RedHat OpenShift (ARO) cluster

If any node is removed from an Azure RedHat OpenShift (ARO) cluster, you can run the following steps to remove that node from the InfoScale cluster.

Be ready with the names of the nodes to be removed and the IP addresses of the nodes to be added.

**Note:** You can remove one node at a time. To remove multiple nodes, run the following steps for every node.

1. Run the following command by logging on the InfoScale SDS pods of all nodes other than the node to be removed.

   ```
   gabconfig -m <Number of nodes minus 1>
   ```

2. Log on by using Exec command to one of the InfoScale SDS pods other than the pod scheduled on the node you want to be removed.

3. Run the following commands

   ```
   /opt/VRTSvcs/bin/haclus -value ReadOnly
   ```

   Skip if the output of the above command is 1. If the output of the above command is not 1, run `/opt/VRTSvcs/bin/haconf -makerw`

   ```
   ./opt/VRTSvcs/bin/hares -modify cvm_clus CVMNodeId -delete <name of the node to be removed>

   /opt/VRTSvcs/bin/hagrp -modify RestSG AutoStartList -delete <name of the node to be removed>

   /opt/VRTSvcs/bin/hagrp -modify RestSG SystemList -delete <name of the node to be removed>

   /opt/VRTSvcs/bin/hagrp -modify cvm AutoStartList -delete <name of the node to be removed>

   /opt/VRTSvcs/bin/hagrp -modify DISK_GROUP SystemList -delete <name of the node to be removed>

   /opt/VRTSvcs/bin/hagrp -modify cvm SystemList -delete <name of the node to be removed>

   /opt/VRTSvcs/bin/hasys -delete <name of the node to be removed>

   /opt/VRTSvcs/bin/haconf -dump -makero

   /opt/VRTSvcs/bin/haclus -value DumpingMembership
   ```

4. One-by-one, log on by using Exec command to all of the InfoScale SDS pods other than the pod scheduled on the node you want to be removed and run the following commands.

   - `/opt/VRTS/bin/vxclustadm -m vcs reinit`

   Edit `/etc/llthosts` and delete the name of the node you want to be removed.

5 Run the following command

```
oc edit infoscalecluster infoscalecluster-dev
```

Delete all information about the node you want to remove from `clusterInfo`.

6 Run the following commands to verify that the node is removed.

```
oc describe cm infoscalecluster-dev-configmap -n infoscale-vtas
```

```
oc describe infoscalecluster
```

7 Run the following command on all nodes in the cluster to add a new node.

```
lltconfig -a set <ID of the new node> link0 <IP address of the
new node>
```

8 Update `/YAML/OpenShift/cr.yaml` with the new node information.

9 Run `oc apply -f /YAML/OpenShift/cr.yaml` on the bastion node.

Chapter 5

# Installing Veritas InfoScale on Kubernetes

This chapter includes the following topics:



## Introduction

**Note:** Version 8.0.310 is not qualified for Kubernetes. Check the supported platform support matrix for more information. See “Supported platforms” on page 17.For installing on Kubernetes, refer to Veritas InfoScale™ for Kubernetes Environments 8.0.300 - Linux documentation on Veritas SORT.

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:** You can form multiple InfoScale clusters. Each cluster can have upto 16 nodes. Veritas InfoScale is deployed on all the nodes you specify in the Custom Resource yaml file. With HyperConverged Architecture supported, a stateful application consuming storage from an InfoScale cluster must run on any of the nodes that are part of that InfoScale cluster.

---

# 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 hardware 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.

6. Hostnames of the InfoScale nodes must exactly match with the fully qualified domain names(FQDN) of the Kubernetes nodes.
7. SE Linux is disabled on all the nodes
8. Kubernetes cluster must be configured by using IPv4 only.
9. Optionally, add custom ciphers to `kube-apiserver`. Refer to Kubernetes documentation to know how to add ciphers.
10. `git` must be installed on the master node.

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

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

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

   ```
   kubectl apply -f
   https://github.com/cert-manager/cert-manager/releases/download/v1.12.0/cert-manager.yaml
   ```

2. Run the following command on the master node to install Node Feature Discovery (NFD) daemonset on the master and the worker nodes.

   ```
   kubectl apply -k
   https://github.com/kubernetes-sigs/node-feature-discovery/deployment/overlays/default?ref=v0.13.2
   ```

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

# Downloading Installer

Complete the following steps to download the installer file.

1. Download `YAML_8.0.310.tar.gz` from the Veritas Download Center.
2. Unzip and untar the file. The folders `/YAML/OpenShift/`, `/YAML/DR`, and `/YAML/Kubernetes` are created. Each folder contains files required for installation.

# 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_8.0.310_Containers_<OS>_Linux.tar.gz` 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_8.0.310.tar.gz .`

- Run the following command on this node

  ```
  setup_vtas_registry.sh -c <IP address of custom registry>:<port number>/vtas_test -t Veritas_InfoScale_8.0.310_Containers_<OS>_Linux.tar.gz
  ```

After this command is successfully run, you have the ISO image as `<IP address of custom registry>:<port number>/vtas_test/infoscale-sds-operator-8.0.310-<rhel8/ol8>` and CR custom registry as `<IP address of custom registry>:<port number>/vtas_test`.

Alternatively, you can download `Veritas_InfoScale_8.0.310_Containers_<OS>_Linux.tar.gz` and individually load, tag, and push each image file. See the following steps.

1. Download `Veritas_InfoScale_8.0.310_Containers_<OS>_Linux.tar.gz` and run the following command to extract the tar file.

   ```
   tar -xvf Veritas_InfoScale_8.0.310_Containers_<OS>_Linux.tar.gz
   ```

2. After you unzip and untar `Veritas_InfoScale_8.0.310_Containers_<OS>_Linux.tar.gz`, you get the following image files

   - `infoscale-sds-8.0.310-<os-version>.img`
   - `infoscale-sds-operator-8.0.310-<os-version>.img`
   - `infoscale-sds-operator-bundle-8.0.310-<os-version>.img`
   - `infoscale-sds-operator-catalog-8.0.310-<os-version>.img`
   - `infoscale-licensing-operator-8.0.310-<os-version-major>.img`
   - `infoscale-licensing-operator-bundle-8.0.310-<os-version-major>.img`
   - `infoscale-licensing-operator-catalog-8.0.310-<os-version-major>.img`
   - `infoscale-fencing-2.0.310-<os-version-major>.img`
   - `infoscale-csi-2.0.310--<os-version-major>.img`
   - `infoscale-dr-manager-8.0.310-<os-version-major>.img`
   - `infoscale-dr-manager-bundle-8.0.310-<os-version-major>.img`
   - `infoscale-dr-manager-catalog-8.0.310-<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-sds-operator-8.0.310-<os-version>.img`, run the following commands-

   - `docker load -i infoscale-sds-operator-8.0.310-<os-version>.img`
   - `docker tag localhost/veritas/infoscale-sds-operator-8.0.310-<os-version>`

```
<IP address of custom registry>:<port
number>/infoscale-sds-operator-8.0.310-<os-version>
```

- `docker push <IP address of custom registry>:<port number>/ infoscale-sds-operator-8.0.310-<os-version>`

4. For `infoscale-licensing-operator-8.0.310-<os-version-major>.img`, run the following commands-
   - `docker load -i infoscale-licensing-operator-8.0.310-<os-version-major>.img`
   - `docker tag localhost/infoscale-licensing:8.0.310-<os-version-major> <IP address of custom registry>:<port number>/ infoscale-licensing-operator-8.0.310-<os-version-major>`
   - `docker push <IP address of custom registry>:<port number>/ infoscale-licensing-operator-8.0.310-<os-version-major>`
5. For `infoscale-sds-8.0.310-<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-sds-8.0.310-<os-version>.img`
   - `docker tag localhost/infoscale:8.0.310-<os-version>-<kernel-version> <IP address of custom registry>:<port number>/infoscale:8.0.310-<os-version>-<kernel-version>`
   - `docker push <IP address of custom registry>:<port number>/infoscale:8.0.310-<os-version>-<kernel-version>`
6. For `infoscale-fencing-2.0.310-<os-version-major>.img`, run the following commands -
   - `docker load -i infoscale-fencing-2.0.310-<os-version-major>.img`
   - `docker tag localhost/veritas/infoscale-fencing-2.0.310-<os-version-major> <IP address of custom registry>:<port number>/infoscale-fencing:2.0.310-<os-version-major>`

- `docker push <IP address of custom registry>:<port number>/infoscale-fencing-2.0.310-<os-version-major>`

7. For `infoscale-csi-2.0.310-<os-version-major>.img`, run the following commands -
   - `docker load -i infoscale-csi-2.0.310-<os-version-major>.img`
   - `docker tag localhost/veritas/infoscale-csi-2.0.310-<os-version-major> <IP address of custom registry>:<port number>/infoscale-csi-2.0.310-<os-version-major>`
   - `docker push localhost/veritas/infoscale-csi-2.0.310-<os-version-major> <IP address of custom registry>:<port number>/infoscale-csi-2.0.310-<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 |
| kube-fencing-switcher | docker.io/kvaps/kube-fencing-switcher |
| kube-fencing-controller | docker.io/kvaps/kube-fencing-controller |
| ose-csi-external-snapshotter-rhel8 | registry.redhat.io/openshift4/ose-csi-external-snapshotter-rhel8:v4.10 |

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

| Image name | Source |
|---|---|
| ose-csi-external-resizer-rhel8 | registry.redhat.io/openshift4/ose-csi-external-resizer-rhel8:v4.7 |
| ose-csi-external-provisioner-rhel8 | registry.redhat.io/openshift4/ose-csi-external-provisioner-rhel8:v4.10 |
| ose-csi-driver-registrar | registry.redhat.io/openshift4/ose-csi-driver-registrar:v4.3 |

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

| Image name | Tag | Required tag |
|---|---|---|
| csi-snapshotter | v6.0.1 | <IP address of custom registry>:<port number>/csi-snapshotter:v6.0.1 |
| csi-provisioner | v3.2.1 | <IP address of custom registry>:<port number>/csi-provisioner:v3.2.1 |
| 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 |
| kube-fencing-switcher | v2.3.0 | <IP address of custom registry>:<port number>/kube-fencing-switcher:v2.3.0 |
| kube-fencing-controller | v2.3.0 | <IP address of custom registry>:<port number>/kube-fencing-controller:v.2.3.0 |
| ose-csi-external-snapshotter-rhel8 | v4.10 | <IP address of custom registry>:<port number>/ose-csi-external-snapshotter-rhel8:v4.10 |
| ose-csi-external-resizer-rhel8 | v4.7 | <IP address of custom registry>:<port number>/ose-csi-external-resizer-rhel8:v4.7 |
| ose-csi-external-provisioner-rhel8 | v4.10 | <IP address of custom registry>:<port number>/ose-csi-external-provisioner-rhel8:v4.10 |
| ose-csi-driver-registrar | v4.3 | <IP address of custom registry>:<port number>/ose-csi-driver-registrar:v4.3 |

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:v6.0.1`
- `docker tag k8s.gcr.io/sig-storage/csi-snapshotter:v6.0.1 <IP address of custom registry>:<port number>/csi-snapshotter:v6.0.1`
- `docker push <IP address of custom registry>:<port number>/csi-snapshotter:v6.0.1`

Run these commands on all the images.

# Applying licenses

**To apply InfoScale licenses**

1. Run `kubectl apply -f /YAML/Kubernetes/lico.yaml` on the master node.
2. Run `kubectl get pods -A|grep -i licensing` on the master node to verify whether `lico.yaml` is successfully applied.

   An output similar to the following indicates that `lico.yaml` is successfully applied.

   ```
   NAME                                  READY STATUS  RESTARTS AGE
   infoscale-licensing-operator
                     -6d75886c84-hhw8s 1/1   Running 0        108s
   ```

3. After `lico.yaml` is successfully applied, wait till the licensing endpoints are activated.

   Run `kubectl describe service/lico-webhook-service -n infoscale-vtas|grep Endpoints` on the master node and review the output.

4. Run the command again till you get an output in the following format.

   ```
   Endpoints: <IP address of the endpoint>:<Port number>
   ```

5 Edit `/YAML/Kubernetes/license_cr.yaml` for the license edition. Optionally, you can change the license name. The default license edition is Developer. You can change the `licenseEdition`.

See "Licensing" on page 15.

If you want to configure Disaster Recovery (DR), you must have Enterprise as the license edition. To configure multiple InfoScale clusters, you must have Storage or Enterprise as the license edition.

```
apiVersion: vlic.veritas.com/v1
kind: License
metadata:
name: <You can change the name here>
spec:
# valid licenseEdition values are Developer,
 Storage or Enterprise
licenseEdition: "Developer"
licenseServer: <Optional - IP address of
                  the VIOM server on your system>
```

6 Run `kubectl create -f /YAML/Kubernetes/license_cr.yaml` on the master node.

7 Run `kubectl get licenses` on the master node to verify whether `license_cr.yaml` is successfully applied.

An output similar to the following indicates that `license_cr.yaml` with Developer as the `license-edition` is successfully applied.

```
NAME          NAMESPACE   LICENSE-EDITION   AGE
license                   DEVELOPER         27s
```

# Considerations for configuring cluster or adding nodes to an existing cluster

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.

By default, Flexible Storage Sharing (FSS) disk group is created. If shared storage is the only storage available across nodes you can set `isSharedStorage` to true and fully shared non-FSS disk group is created while configuring clusters. Mirroring is performed across enclosures thus ensuring redundancies. While using shared storage, Veritas recommends changing the default majority-based fencing to disk-based fencing. With shared storage and majority-based fencing, storage becomes inaccessible when majority of the nodes go down; although all disks are connected to all the nodes. With disk-based fencing, storage is available to applications even when one node is up. To configure disk-based fencing, specify hardware path of the fencing disks for at least one node. Veritas recommends using at least three disks for fencing purpose. Ensure that the number of disks is odd in number.

You can enable encryption at the disk group level or for specific Volumes within the disk group. Encryption is not enabled by default. You can set `encrypted` to true to enable encryption. If you want the same encryption key for all Volumes, you can set `sameEnckey` to true. For different encryption keys per Volume, set `sameEnckey` to false.

---

**Note:** For Disaster Recovery (DR) configuration, only Volume level encryption is supported. Disk group level encryption is not supported.

---

# Creating multiple InfoScale clusters

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

You can configure multiple clusters in different namespaces. After a cluster is configured and pods are created successfully, you can update `cr.yaml` with the name, namespace, Cluster ID, and node-related information of the next cluster. You can choose to rename `cr.yaml` or you can overwrite the same `cr.yaml`. You can then create the cluster and after a successful creation of pods, update `cr.yaml` for the subsequent cluster. Likewise, you can configure multiple clusters.

Each InfoScale cluster must have a unique name even when clusters are created across namespaces.

# 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,

unzip, and untar `YAML_8.0.310.tar.gz` , 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 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 assign a clusterrole to infoscale-admin. You must be logged in as a super user.

1. `kubectl config use-context kubernetes-admin@kubernetes`

   You have switched the context to `kubernetes-admin@kubernetes`.

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

   Here, `infoscale-admin-role.yaml` is downloaded as a part of `YAML_8.0.310.tar.gz` .

3. `kubectl create clusterrolebinding infoscale-admin-clusterrolebinding --clusterrole=infoscale-admin-role --user=infoscale-admin`

   This step binds the cluster role `infoscale-admin-role.yaml` to a new user - `infoscale-admin`.

4 `openssl genrsa -out infoscale-admin.key 2048`

You have generated a key for `infoscale-admin` by using 2048 bits.

```
openssl req -new -key infoscale-admin.key -out infoscale-admin.csr
-subj "/CN=infoscale-admin"
```

You have created a certificate signing request for `infoscale-admin`.

```
openssl x509 -req -in infoscale-admin.csr -CA
/etc/kubernetes/pki/ca.crt -CAkey /etc/kubernetes/pki/ca.key
-CAcreateserial -out infoscale-admin.crt -days <Number of days>
```

The certificate is signed by Kubernetes, a crt file is created, and is valid for the number of days you specified in the command.

5 `mkdir /certs && mv infoscale-admin.crt infoscale-admin.key /certs`

`infoscale-admin.crt` and `infoscale-admin.key` moved to `/certs` folder for the `lico.yaml` to reference.

6 `cd /certs`

```
kubectl config set-credentials infoscale-admin
--client-certificate=/certs/infoscale-admin.crt
--client-key=/certs/infoscale-admin.key
```

```
kubectl config set-context infoscale-admin.context
--cluster=kubernetes --user=infoscale-admin
```

```
kubectl config use-context infoscale-admin.context
```

Credentials are set for `infoscale-admin` and its context created.

7 `kubectl config current-context infoscale-admin.context`

Ensures that the current context is `infoscale-admin.context`.

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-sds-operator:8.0.310-<rhel8/ol8>** with **image: <IP address of custom registry>/infoscale-sds-operator:8.0.310-<rhel8/ol8>**.

2. Edit `/YAML/Kubernetes/lico.yaml` as under -

   Replace **image: 192.168.1.21/veritas/infoscale-licensing-operator:8.0.310-ol8** with **image:**

**<IP address of custom registry>/infoscale-licensing-operator:8.0.310-<rhel8/ol8>**.

3. Edit `/YAML/DR/Kubernetes/dro_deployment.yaml` as under -

   Replace **image: 192.168.1.21/veritas/infoscale-dr-manager:8.0.310-rhel8** with **image: <IP address of custom registry>/veritas/infoscale-dr-manager:8.0.310-<rhel8/ol8>**.

4. Optionally, if you want to change the default kubelet path, edit `/YAML/Kubernetes/iso.yaml` as under.

```
env:

   - name: KUBELET_PATH

     value: <enter the new path>
```

   The default path is `/var/lib/kubelet`.

   **Note:** Do not change the kubelet path after clusters are configured.

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

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

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

   ```
   kubectl get pods -n infoscale-vtas|grep infoscale-sds-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-sds-operator-6dc9bc8856-lh72f 1/1     Running 0        2d18h
```

# Configuring cluster

Complete the following steps for each cluster.

**Note:** Ensure that you add a Kubernetes node only to a single InfoScale cluster.

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

```
metadata:
 name: < Assign a name to this cluster >
 namespace: < The namespace where you want to
                    create this cluster>


spec:
 clusterID: <Optional- Enter an ID for this cluster.
               The ID can be any number between 1 and 65535 >
 isSharedStorage: true
 - 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.

  enableScsi3pr:<Enter True to enable SCSI3 persistent reservation>
```

```
fencingDevice: ["<Hardware path to the first fencing device>",
                "<Hardware path to the second fencing device>",
                "<Hardware path to the third fencing device>", ]
encrypted: false
sameEnckey: false

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. You can choose to rename `cr.yaml`. If you rename the file, ensure that you use that name in the next step.

---

**Note:** Veritas recommends renaming `cr.yaml` and maintaining a custom resource for each cluster. The renamed `cr.yaml` is used to add more nodes to that InfoScale cluster.

---

3. Run the following command on the master node.

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

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

   ```
   kubectl get infoscalecluster -A
   ```

   Use the namespace from the output similar to the following -

   ```
   NAMESPACE   NAME                VERSION   CLUSTERID   STATE   AGE
   .
   .
   <Namespace> <Name of the
                InfoScale cluster> 8.0.310  <Cluster ID> Running 25h
   .
   .
   ```

5. 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-csi-controller-35359-0 5/5   Running 0        12d
infoscale-csi-node-35359-7rjv9   2/2   Running 0        3d20h
infoscale-csi-node-35359-dlrxh   2/2   Running 0        4d21h
infoscale-csi-node-35359-dmxwq   2/2   Running 0        12d
infoscale-csi-node-35359-j9x7v   2/2   Running 0        12d
infoscale-csi-node-35359-w6wf2   2/2   Running 0        3d20h
infoscale-fencing-controller-
          35359-6cc6cd7b4d-l7jtc 1/1   Running 0        3d21h
infoscale-fencing-enabler-
                     35359-9gkb4 1/1   Running 0        12d
infoscale-fencing-enabler-35359-
                           gwn7w 1/1   Running 0        3d20h
infoscale-fencing-enabler-35359-
                           jrf2l 1/1   Running 0        12d
infoscale-fencing-enabler-35359-
                           qhzdt 1/1   Running 1        3d20h
infoscale-fencing-enabler-35359-
                           zqdvj 1/1   Running 1        4d21h
infoscale-sds-35359-
          ed05b7abb28053ad-7svqz 1/1   Running 0        13d
infoscale-sds-35359-
          ed05b7abb28053ad-c272q 1/1   Running 0        13d
infoscale-sds-35359-
          ed05b7abb28053ad-g4rbj 1/1   Running 0        4d21h
infoscale-sds-35359-
          ed05b7abb28053ad-hgf6h 1/1   Running 0        3d20h
infoscale-sds-35359-
          ed05b7abb28053ad-wk5ph 1/1   Running 0        3d20h
infoscale-sds-operator-
                 7fb7cd57c-rskms 1/1   Running 0        3d20h
infoscale-licensing-operator-
                756c854fdb-xvdnr 1/1   Running 0        13d
```

InfoScale SDS pods are created in the namespace you specify in `cr.yaml`. Fencing and CSI pods are created in the operator namespace.

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

   ```
   kubectl get nodes
   ```

   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 -A
   ```

   See the State in the output similar to the following -

   ```
   NAMESPACE   NAME                VERSION   CLUSTERID   STATE   AGE
   .
   .
   <Namespace> <Name of the
                InfoScale cluster> 8.0.310  <Cluster ID> Running 25h
   .
   .
   ```

**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.

---

```
metadata:
  name: < Assign a name to this cluster >
  namespace: < The namespace where you want to
                      create this cluster>


 spec:
  clusterID: <Optional- Enter an ID for this cluster.
                The ID can be any number between 1 and 65535 >
  isSharedStorage: true
  - 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.

  enableScsi3pr:<Enter True to enable SCSI3 persistent reservation>

  fencingDevice: ["<Hardware path to the first fencing device>",
                  "<Hardware path to the second fencing device>",
                  "<Hardware path to the third fencing device>", ]
  encrypted: false
  sameEnckey: false

  customImageRegistry:  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 -A`

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

```
NAMESPACE   NAME            VERSION   CLUSTERID STATE             AGE
.
.
<Namespace> <Name of the              <Cluster
          InfoScale cluster> 8.0.310  ID>       ProcessingAddNode 25h
.
.
```

b. `kubectl describe infoscalecluster -n <Namespace>`

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:
      <Excluded device path 1>
      <Excluded device path 2>
    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-csi-controller-35359-0 5/5   Running 0        12d
infoscale-csi-node-35359-7rjv9   2/2   Running 0        3d20h
infoscale-csi-node-35359-dlrxh   2/2   Running 0        4d21h
infoscale-csi-node-35359-dmxwq   2/2   Running 0        12d
infoscale-csi-node-35359-j9x7v   2/2   Running 0        12d
infoscale-csi-node-35359-w6wf2   2/2   Running 0        3d20h
infoscale-fencing-controller-
          35359-6cc6cd7b4d-l7jtc 1/1   Running 0        3d21h
infoscale-fencing-enabler-
                     35359-9gkb4 1/1   Running 0        12d
infoscale-fencing-enabler-35359-
                           gwn7w 1/1   Running 0        3d20h
infoscale-fencing-enabler-35359-
                           jrf2l 1/1   Running 0        12d
infoscale-fencing-enabler-35359-
                           qhzdt 1/1   Running 1        3d20h
infoscale-fencing-enabler-35359-
                           zqdvj 1/1   Running 1        4d21h
infoscale-sds-35359-
          ed05b7abb28053ad-7svqz 1/1   Running 0        13d
infoscale-sds-35359-
          ed05b7abb28053ad-c272q 1/1   Running 0        13d
infoscale-sds-35359-
          ed05b7abb28053ad-g4rbj 1/1   Running 0        4d21h
infoscale-sds-35359-
          ed05b7abb28053ad-hgf6h 1/1   Running 0        3d20h
infoscale-sds-35359-
          ed05b7abb28053ad-wk5ph 1/1   Running 0        3d20h
infoscale-sds-operator-
                 7fb7cd57c-rskms 1/1   Running 0        3d20h
infoscale-licensing-operator-
                756c854fdb-xvdnr 1/1   Running 0        13d
```

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

```
kubectl get infoscalecluster -A
```

See the State in the output similar to the following -

```
NAMESPACE   NAME                VERSION   CLUSTERID   STATE   AGE
.
.
<Namespace> <Name of the
             InfoScale cluster> 8.0.310  <Cluster ID> Running 25h
.
.
```

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

```
kubectl describe infoscalecluster <Cluster Name> -n <Namespace>
```

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

```
Status:
  Cluster Name:  <Cluster Name>
  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.

---

**Note:** If you want to retain the disk group and re-use the Volumes you created, ensure that you note the `ClusterID` before undeploying and uninstalling InfoScale. You must use this `ClusterID` while re-installing InfoScale. Else, clean the disks before uninstalling.

---

```
kubectl delete infoscalecluster -n <Namespace> <Name of the cluster>
```

**Note:** Run the command for each cluster.

The commands to clean up InfoScale components 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/license_cr.yaml

kubectl delete -f /YAML/Kubernetes/lico.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. When fencing is configured, certain keys are configured. Those must also be deleted after uninstallation. Run `./clearkeys.sh <Path to the first disk>, <Path to the second disk>,...` to remove stale keys that might have remained.

Chapter 6

# Configuring KMS-based Encryption on an OpenShift cluster

This chapter includes the following topics:



## Introduction

Veritas InfoScale in Containers Environment offers advanced security of data at rest with KMS-based (Key Management Server) encryption. You can use an external Key Management Server to manage the encryption key. Encryption is supported for Volumes and disk groups. However, DR configuration is not supported for disk group level encryption. You must add an external CA certificate (custom CA certificate) first to enable connection with the Key Management Server (KMS).

---

**Note:** Whenever the InfoScale cluster is reconfigured, ensure that you generate the client certificate again and update the KMS server.

---

# Adding a custom CA certificate

The following steps inform you how to add a custom CA certificate by using the `cfssl` tool. The steps are about a KubernetesCA certificate.

---

**Note:** As an Administrator, you can use any other compliant tool for signing. Refer to the procedure of that tool. Ensure that you set `"is_ca"` to `true`.

---

As a prerequisite, download the following packages from `github.com` repository by using python-pip.

`Tools`

- `cfssl`
- `cfssl-certinfo`
- `cfssljson`

If you are using external CA credentials, skip steps 1 to 6 and go to step 7. Ensure that you set `"is_ca"` to `true`.

Complete the following steps

1. On the bastion node, create a directory - `custom-ca`. Navigate to this directory to perform the next steps.
2. Copy the following content into a file and save it as `vxconfig.json`.

```
{
  "signing": {
    "default": {
      "expiry": "43800h"
    },
    "profiles": {
      "cluster": {
        "expiry": "8760h",
        "usages": [
          "signing",
          "key encipherment",
          "cert sign",
          "server auth",
          "client auth"
        ],
        "ca_constraint": {
          "is_ca": true
        }
      }
    }
  }
}
```

**3** Copy the following content into a file and save it as `csr_config.json`.

```
{
  "CN": "infoscale-ca",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "hosts": [
    "kubernetes"
  ],
  "names": [
    {
      "O": "system:nodes",
      "OU": "vx"
    }
  ]
}
```

**4** Run the following command to generate certificates.

```
cfssl genkey csr_config.json | cfssljson -bare infoscale-ca
```

Review output similar to the following output

```
2022/02/10 15:09:27 [INFO] generate received request
2022/02/10 15:09:27 [INFO] received CSR
2022/02/10 15:09:27 [INFO] generating key: rsa-2048
2022/02/10 15:09:28 [INFO] encoded CSR
```

**5** Now you must sign the certificate you just generated. Run the following command.

```
cfssl sign -ca /etc/kubernetes/pki/ca.crt -ca-key
/etc/kubernetes/pki/ca.key -hostname kubernetes -config
./vxconfig.json -profile cluster ./infoscale-ca.csr | cfssljson
-bare infoscale-ca
```

**6** Run `ls` to list files in the folder.

Following files must be created

```
infoscale-ca.csr  infoscale-ca-key.pem  infoscale-ca.pem
```

Here , `infoscale-ca.pem` is the external CA certificate.

**7** Copy the following content into a file and save it as `custom-ca.yaml`.

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: infoscale-sds-operator
  name: infoscale-vtas
---
apiVersion: v1
kind: Secret
metadata:
  name: infoscale-ca
  namespace: <namespace of cert-manager>
type: kubernetes.io/tls
data:
  ca.crt: $(oc get cm kube-root-ca.crt -o
            jsonpath="{.data['ca\.crt']}"| base64 -w0)
  tls.crt: $(base64 ./infoscale-ca.pem | tr -d '\n')
  tls.key: $(base64 ./infoscale-ca-key.pem | tr -d '\n')
```

You have to replace content for `ca.crt`, `tls.crt`, and `tls.key`.

**8** Run

```
oc get cm kube-root-ca.crt -o jsonpath="{.data['ca\.crt']}" |
base64 -w0
```

Copy the output of this command as

```
<Content of ca.crt>
```

**9** Modify `custom-ca.yaml` as under

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: infoscale-sds-operator
  name: infoscale-vtas
---
apiVersion: v1
kind: Secret
metadata:
  name: infoscale-ca
  namespace: <namespace of cert-manager>
type: kubernetes.io/tls
data:
  ca.crt: <Content of ca.crt>
  tls.crt: $(base64 ./infoscale-ca.pem | tr -d '\n')
  tls.key: $(base64 ./infoscale-ca-key.pem | tr -d '\n')
```

**10** Similarly, run

`base64 ./infoscale-ca.pem | tr -d '\n'` and update `tls.crt` in `custom-ca.yaml` with the output of this command.

**11** Run

`base64 ./infoscale-ca-key.pem | tr -d '\n'` and update `tls.key` in `custom-ca.yaml` with the output of this command.

12 Ensure that `custom-ca.yaml` is as under

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: infoscale-sds-operator
  name: <cert-manager namespace>
---
apiVersion: v1
kind: Secret
metadata:
  name: infoscale-ca
  namespace: <namespace of cert-manager>
type: kubernetes.io/tls
data:
  ca.crt: <Content of ca.crt>
  tls.crt: <Content of tls.crt>
  tls.key: <Content of tls.key>
```

13 Run `oc apply -f custom-ca.yaml`.

14 If you are configuring DR, copy this `custom-ca.yaml` to the DR cluster.

15 Run `oc apply -f custom-ca.yaml` on the DR cluster before applying license.

After `custom-ca.yaml` is successfully applied, you can apply `iso.yaml`. See the Installation section.

---

**Note:** This certificate is valid for 12 months. Ensure that you extend its validity in the eighth month.

---

# Configuring InfoScale to enable transfer of keys

You must configure InfoScale to enable a connection with the Key Management Server (KMS) to transfer and save rest certs.

If you have created multiple InfoScale clusters, ensure you run steps 23 to 28 for every cluster.

The rest certs are renewed every three months and the renewed certs must be uploaded to the KMS server. Run steps 23 to 28 every three months for encryption to work.

**Note:** After a client rest cert is renewed, ensure that you add the renewed client cert to the client group on the KMS server.

Complete the following steps

**Note:** The following steps inform you how to configure IBM Key Management Server. As an administrator, you can configure any KMIP-compliant server. Refer to the procedure of that KMIP-compliant server.

1. Be ready with the IP address and port number of the Key Management Server (KMS).
2. Run `echo "<IP address of the server >"| base64`

   Verify the output as under

   ```
   Server output for base64
   ```

3. Run `echo "<Port number of the server >"| base64`

   Verify the output as under

   ```
   Port number output for base64
   ```

4. Copy the following content into a file and save it as `infoscale-kmip-secret.yaml`.

   ```
   apiVersion: v1
   data:
     host: <Server output for base64>
     port: <Port number output for base64>
   kind: Secret
   metadata:
     name: infoscale-kmip-encrypt
     namespace: infoscale-vtas
   type: Opaque
   ```

5. Run `oc apply -f infoscale-kmip-secret.yaml` to deploy the InfoScale secret.
6. From another terminal, logon to

   https://www.ibm.com/docs/en/sgklm/&4.1.1?topic=objects-registering-client-by-using-graphical-user-interface.

7 Select **Advanced Configuration > Server Certificate**. Click **Add**. The **Add SSL/KMIP Certificate** screen opens.

8 Select **Request certificate from a third-party provider** and enter values for **Certificate label** and **Certificate description**.

9 Click **Add Certificate**. The certificate is listed as **Administer Server Certificates**.

10 Review the Status of the certificate. The status is **Certificate is pending**.

11 From the bastion node, run `ssh root@<IP address of the KMS >`. Enter the password and login.

12 The certificate you just created is listed under `/opt/IBM/WebSphere/AppServer/products/sklm/data/` as `<Time stamp>_<Certificate name>.csr`.

---

**Note:** The path might vary depending on the KMS version.

---

13 Copy content of `/opt/IBM/WebSphere/Liberty/products/sklm/data/<Time stamp>_<Certificate name>.csr` into another file `<Copy of server cert content>.pem`.

14 Run `openssl x509 -req -in <Time stamp>_<Certificate name>.csr -CA infoscale-ca.pem -CAkey infoscale-ca-key.pem -CAcreateserial -out <server-certificate-name> -days 1024 -sha256`

15 Review the output for the following message.

```
Signature ok
```

16 Copy `<Certificate name>.crt` to the root directory of the Key Management server.

17 On the Welcome screen of KMS, click **Third-party certificates pending import**.

18 In the Import Certificate screen, click **Browse** and navigate to the certificate you saved. Click **Select**.

19 Run `oc get secret -n infoscale-vtas`.

20 Review the output for the following

```
NAME
infoscale-ca
```

21 Run `oc get secret -n infoscale-vtas`.

22 Review the output for the following

```
NAME
infoscale-kmip-encrypt
```

23 Run `oc -n <namespace where cr is deployed> get secret infoscale-sds-rest-tls-cert-<cluster-id> -o jsonpath="{.data['tls\.crt']}" | base64 --decode > <kmip-device-client-cert>`,

24 Copy `<kmip-device-client-cert>` to the root directory of the KMS.

25 On the KMS, select **Advanced Configuration > Client Device Certificates**. Click **Import**.

26 In the **Import SSL/KMIP Certificate for Clients** window, assign a name and click Browse to select `<device-certificate>.crt` from the root directory.

27 Select the checkbox next to **Allow the server to trust this certificate with the associated client device**.

28 Click **Import**.

After a successful configuration, data is more secure and a need to back up keys required during Disaster Recovery is eliminated.

**For a DR configuration**

1 Complete steps 1 to 24 on one of the DR sites to configure `infoscale-kmip-encrypt` and the server certificate. Ensure that you configure `infoscale-kmip-encrypt` on all the sites. See steps 4 and 5.

2 Run the following command on the primary site to get the client certificate.

```
oc -n infoscale-vtas get secret
infoscale-sds-rest-tls-cert-<cluster-id> -o
jsonpath="{.data['tls\.crt']}" | base64 --decode >
<kmip-primary-cert>
```

3 Run the following command on the secondary site to get the client certificate.

```
oc -n infoscale-vtas get secret
infoscale-sds-rest-tls-cert-<cluster-id> -o
jsonpath="{.data['tls\.crt']}" | base64 --decode >
<kmip-secondary-cert>
```

4 Logon to the below URL and perform the following steps to register client and create client group.

https://www.ibm.com/docs/en/sgklm/4.1.1?topic=&objects-registering-client-by-using-graphical-user-interface

- Navigate to **Clients > Clients (subsection) > Create > fill details**. Enter <kmip-primary-cert> to register client.
- Similarly, enter <kmip-secondary-cert> to register client.

5 Logon to

https://www.ibm.com/docs/en/sgklm/4.1.1?&topic=mcgctco-creating-managing-client-group-by-using-graphical-user-interface

Navigate to **Clients > Client Groups (subsection) > Add > Provide Client Group Name > Create** . Select clients from the list and click **Save**.

6 Run steps from 23 to 28 again.

# Renewing with an external CA certificate

External CA certificates typically get renewed a few months before the validity end date. A cluster administrator re-creates the new certificates and populates the InfoScale cluster.

The new CA certificates can then be applied before the validity end date, before the earlier certificates expire.

When the external certificate is issued by the intermediary of the CA and the issuer knows the intermediary, the content of `tls.crt` is a resulting certificate followed by a certificate chain. The certificate chain does not include a root CA certificate, as it is stored in `ca.crt`.

For InfoScale the external CA certificate is valid for 12 months and as an Administrator, you can initiate its renewal after the eighth month.

---

**Note:** Self-signing certificate is automatically renewed without any intervention. Validity of the self-signing certificate is four months and it is automatically renewed in the third month. However, external CA certificate needs to renewed.

---

Complete the following steps, ensuring that NTP is synchronized across nodes.

1. Run the following commands to renew the CA certificate.

   ---

   **Note:** This is an example of cfssl tool. You can also use any other tool to renew CA certificates. Refer to the procedure of that tool.

   ---

   For generating the certificate.

```
cfssl gencsr -key /infoscale-ca-key.pem /csr_config.json |
cfssljson -bare infoscale-ca
```

For signing the certificate.

```
cfssl sign -ca /etc/kubernetes/pki/ca.crt -ca-key
/etc/kubernetes/pki/ca.key -hostname kubernetes -config
./vxconfig.json -profile cluster ./infoscale-ca.csr | cfssljson
-bare infoscale-ca
```

2. Run the following command to generate the new secrets.

   ```
   sh gen-cert
   ```

3. Run the following command to update the new `infoscale-ca` secret with renewed `infoscale-ca.pem`.

   ```
   oc apply -f custom-ca.yaml
   ```

   Wait for upto five minutes.

4. Now you need to delete secrets of the following certificates.

   - infoscale-sds-rest-tls-cert-<value>
   - infoscale-csi-tls-cert
   - infoscale-fencing-tls-cert
   - iso-tls-cert
   - webhook-tls-cert
   - lico-tls-cert

   Run the following command to delete secrets of these certificates.

   ```
   oc delete secret -n infoscale-vtas <certificate name>
   ```

5. To enable encryption, perform steps listed in Configuring InfoScale to enable transfer of keys again.

After these certificates are deleted, the cert-manager automatically re-creates the new certificates. Wait for 15 minutes.

---

**Note:** If DR is configured, ensure that you run these commands on the secondary site immediately after running these commands on the primary site.

---

Chapter 7

# Configuring KMS-based Encryption on a Kubernetes cluster

This chapter includes the following topics:



## Introduction

**Note:** Version 8.0.310 is not qualified for Kubernetes. Check the supported platform support matrix for more information. See “Supported platforms” on page 17.For installing on Kubernetes, refer to Veritas InfoScale™ for Kubernetes Environments 8.0.300 - Linux documentation on Veritas SORT.

Veritas InfoScale in Containers Environment offers advanced security of data at rest with KMS-based (Key Management Server) encryption. You can use an external Key Management Server to manage the encryption key. Encryption is supported for Volumes and disk groups. However, DR configuration is not supported for disk group level encryption. You must add an external CA certificate (custom CA certificate) first to enable connection with the Key Management Server (KMS).

**Note:** Whenever the InfoScale cluster is reconfigured, ensure that you generate the client certificate again and update the KMS server.

# Adding a custom CA certificate

The following steps inform you how to add a custom CA certificate by using the `cfssl` tool.The steps are about a KubernetesCA certificate.

**Note:** As an Administrator, you can use any other compliant tool for signing. Refer to the procedure of that tool. Ensure that you set `"is_ca"` to `true`.

As a prerequisite, download the following packages from `github.com` repository by using python-pip.

`Tools`

- `cfssl`
- `cfssl-certinfo`
- `cfssljson`

If you are using external CA credentials, skip steps 1 to 6 and go to step 7. Ensure that you set `"is_ca"` to `true`.

Complete the following steps

1. On the master node, create a directory - `custom-ca`. Navigate to this directory to perform the next steps.
2. Copy the following content into a file and save it as `vxconfig.json`.

```
{
  "signing": {
    "default": {
      "expiry": "43800h"
    },
    "profiles": {
      "cluster": {
        "expiry": "8760h",
        "usages": [
          "signing",
          "key encipherment",
          "cert sign",
          "server auth",
          "client auth"
        ],
        "ca_constraint": {
          "is_ca": true
        }
      }
    }
  }
}
```

**3** Copy the following content into a file and save it as `csr_config.json`.

```
{
  "CN": "infoscale-ca",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "hosts": [
    "kubernetes"
  ],
  "names": [
    {
      "O": "system:nodes",
      "OU": "vx"
    }
  ]
}
```

**4** Run the following command to generate certificates.

```
cfssl genkey csr_config.json | cfssljson -bare infoscale-ca.
```

Review output similar to the following output

```
2022/02/10 15:09:27 [INFO] generate received request
2022/02/10 15:09:27 [INFO] received CSR
2022/02/10 15:09:27 [INFO] generating key: rsa-2048
2022/02/10 15:09:28 [INFO] encoded CSR
```

**5** Now you must sign the certificate you just generated. Run the following command.

```
cfssl sign -ca /etc/kubernetes/pki/ca.crt -ca-key
/etc/kubernetes/pki/ca.key -hostname kubernetes -config
./vxconfig.json -profile cluster ./infoscale-ca.csr | cfssljson
-bare infoscale-ca
```

**6** Run `ls` to list files in the folder.

Following files must be created

```
infoscale-ca.csr  infoscale-ca-key.pem  infoscale-ca.pem
```

Here , `infoscale-ca.pem` is the external CA certificate.

**7** Copy the following content into a file and save it as `custom-ca.yaml`.

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: infoscale-sds-operator
  name: infoscale-vtas
---
apiVersion: v1
kind: Secret
metadata:
  name: infoscale-ca
  namespace: <namespace of cert-manager>
type: kubernetes.io/tls
data:
  ca.crt: $(kubectl get cm kube-root-ca.crt -o
             jsonpath="{.data['ca\.crt']}"| base64 -w0)

  tls.crt: $(base64 ./infoscale-ca.pem | tr -d '\n')
  tls.key: $(base64 ./infoscale-ca-key.pem | tr -d '\n')
```

You have to replace content for `ca.crt`, `tls.crt`, and `tls.key`.

**8** Run

```
kubectl get cm kube-root-ca.crt -o jsonpath="{.data['ca\.crt']}"
| base64 -w0.
```

Copy the output of this command as

```
<Content of ca.crt>
```

**9** Modify `custom-ca.yaml` as under

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: infoscale-sds-operator
  name: infoscale-vtas
---
apiVersion: v1
kind: Secret
metadata:
  name: infoscale-ca
  namespace: <namespace of cert-manager>
type: kubernetes.io/tls
data:
  ca.crt: <Content of ca.crt>
  tls.crt: $(base64 ./infoscale-ca.pem | tr -d '\n')
  tls.key: $(base64 ./infoscale-ca-key.pem | tr -d '\n')
```

**10** Similarly, run

`base64 ./infoscale-ca.pem | tr -d '\n'` and update `tls.crt` in `custom-ca.yaml` with the output of this command.

**11** Run

`base64 ./infoscale-ca-key.pem | tr -d '\n'` and update `tls.key` in `custom-ca.yaml` with the output of this command.

**12** Ensure that `custom-ca.yaml` is as under

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: infoscale-sds-operator
  name: infoscale-vtas
---
apiVersion: v1
kind: Secret
metadata:
  name: infoscale-ca
  namespace: <namespace of cert-manager>
type: kubernetes.io/tls
data:
  ca.crt: <Content of ca.crt>
  tls.crt: <Content of tls.crt>
  tls.key: <Content of tls.key>
```

**13** Run `kubectl apply -f custom-ca.yaml`.

**14** If you are configuring DR, copy this `custom-ca.yaml` to the DR cluster.

**15** Run `kubectl apply -f custom-ca.yaml` on the DR cluster before applying the license.

After `custom-ca.yaml` is successfully applied, you can apply `iso.yaml`. See the Installation section.

---

**Note:** This certificate is valid for 12 months. Ensure that you extend its validity in the eighth month.

---

# Configuring InfoScale to enable transfer of keys

You must configure InfoScale to enable a connection with the Key Management Server (KMS) to transfer and save rest certs.

If you have created multiple InfoScale clusters, ensure you run steps 23 to 28 for every cluster.

The rest certs are renewed every three months and the renewed certs must be uploaded to the KMS server. Run steps 23 to 28 every three months for encryption to work.

**Note:** After a client rest cert is renewed, ensure that you add the renewed client cert to the client group on the KMS server.

Complete the following steps

**Note:** The following steps inform you how to configure IBM Key Management Server. As an administrator, you can configure any KMIP-compliant server. Refer to the procedure of that KMIP-compliant server.

1. Be ready with the IP address and port number of the Key Management Server (KMS).
2. Run `echo "<IP address of the server >"| base64`

   Verify the output as under

   ```
   Server output for base64
   ```

3. Run `echo "<Port number of the server >"| base64`

   Verify the output as under

   ```
   Port number output for base64
   ```

4. Copy the following content into a file and save it as `infoscale-kmip-secret.yaml`.

   ```
   apiVersion: v1
   data:
     host: <Server output for base64>
     port: <Port number output for base64>
   kind: Secret
   metadata:
     name: infoscale-kmip-encrypt
     namespace: infoscale-vtas
   type: Opaque
   ```

5. Run `kubectl apply -f infoscale-kmip-secret.yaml` to deploy the InfoScale secret.
6. From another terminal, logon to

   https://www.ibm.com/docs/en/sgklm/4.1.1?topic=objects-registering-client-by-using-graphical-user-interface.

7 Select **Advanced Configuration > Server Certificate**. Click **Add**. The **Add SSL/KMIP Certificate** screen opens.

8 Select **Request certificate from a third-party provider** and enter values for **Certificate label** and **Certificate description**.

9 Click **Add Certificate**. The certificate is listed as **Administer Server Certificates**.

10 Review the Status of the certificate. The status is **Certificate is pending**.

11 From the master node, run `ssh root@<IP address of the KMS >`. Enter the password and login.

12 The certificate you just created is listed under `/opt/IBM/WebSphere/AppServer/products/sklm/data/` as `<Time stamp>_<Certificate name>.csr`.

**Note:** The path might vary depending on the KMS version.

13 Copy content of `/opt/IBM/WebSphere/Liberty/products/sklm/data/<Time stamp>_<Certificate name>.csr` into another file `<Copy of server cert content>.pem`.

14 Run `openssl x509 -req -in <Time stamp>_<Certificate name>.csr -CA infoscale-ca.pem -CAkey infoscale-ca-key.pem -CAcreateserial -out <server-certificate-name> -days 1024 -sha256`

15 Review the output for the following message.

```
Signature ok
```

16 Copy `<Certificate name>.crt` to the root directory of the Key Management server.

17 On the Welcome screen of KMS, click **Third-party certificates pending import**.

18 In the Import Certificate screen, click **Browse** and navigate to the certificate you saved. Click **Select**.

19 Run `kubectl get secret -n infoscale-vtas`.

20 Review the output for the following

```
NAME
infoscale-ca
```

21 Run `kubectl get secret -n infoscale-vtas`.

22 Review the output for the following

```
NAME
infoscale-kmip-encrypt
```

23 Run `kubectl -n <namespace where cr is deployed> get secret infoscale-sds-rest-tls-cert-<cluster-id> -o jsonpath="{.data['tls\.crt']}" | base64 --decode > <kmip-device-client-cert>`,

24 Copy `<kmip-device-client-cert>` to the root directory of the KMS.

25 On the KMS, select **Advanced Configuration > Client Device Certificates**. Click **Import**.

26 In the **Import SSL/KMIP Certificate for Clients** window, assign a name and click Browse to select `<device-certificate>.crt` from the root directory.

27 Select the checkbox next to **Allow the server to trust this certificate with the associated client device**.

28 Click **Import**.

After a successful configuration, data is more secure and a need to back up keys required during Disaster Recovery is eliminated.

**For a DR configuration**

1 Complete steps 1 to 24 on one of the DR sites to configure `infoscale-kmip-encrypt` and the server certificate. Ensure that you configure `infoscale-kmip-encrypt` on all the sites. See steps 4 and 5.

2 Run the following command on the primary site to get the client certificate.

```
kubectl -n infoscale-vtas get secret
infoscale-sds-rest-tls-cert-<cluster-id> -o
jsonpath="{.data['tls\.crt']}" | base64 --decode >
<kmip-primary-cert>
```

3 Run the following command on the secondary site to get the client certificate.

```
kubectl -n infoscale-vtas get secret
infoscale-sds-rest-tls-cert-<cluster-id> -o
jsonpath="{.data['tls\.crt']}" | base64 --decode >
<kmip-secondary-cert>
```

4 Logon to the below URL and perform the following steps to register client and create client group.

https://www.ibm.com/docs/en/sgklm/4.1.1?topic=objects-registering-client-by-using-graphical-user-interface

- Navigate to **Clients > Clients (subsection) > Create > fill details**. Enter <kmip-primary-cert> to register client.
- Similarly, enter <kmip-secondary-cert> to register client.

5 Logon to

https://www.ibm.com/docs/en/sgklm/4.1.1?topic=mcgctco-creating-managing-client-group-by-using-graphical-user-interface

Navigate to **Clients > Client Groups (subsection) > Add > Provide Client Group Name > Create** . Select clients from the list and click **Save**.

6 Run steps 23 to 28 again.

# Renewing with an external CA certificate

External CA certificates typically get renewed a few months before the validity end date. A cluster administrator re-creates the new certificates and populates the InfoScale cluster.

The new CA certificates can then be applied before the validity end date, before the earlier certificates expire.

When the external certificate is issued by the intermediary of the CA and the issuer knows the intermediary, the content of `tls.crt` is a resulting certificate followed by a certificate chain. The certificate chain does not include a root CA certificate, as it is stored in `ca.crt`.

For InfoScale the external CA certificate is valid for 12 months and as an Administrator, you can initiate its renewal after the eighth month.

---

**Note:** Self-signing certificate is automatically renewed without any intervention. Validity of the self-signing certificate is four months and it is automatically renewed in the third month. However, external CA certificate needs to renewed.

---

Complete the following steps, ensuring that NTP is synchronized across nodes.

1. Run the following commands to renew the CA certificate.

   ---

   **Note:** This is an example of cfssl tool. You can also use any other tool to renew CA certificates. Refer to the procedure of that tool.

   ---

   For generating the certificate.

```
cfssl gencsr -key /infoscale-ca-key.pem /csr_config.json |
cfssljson -bare infoscale-ca
```

For signing the certificate.

```
cfssl sign -ca /etc/kubernetes/pki/ca.crt -ca-key
/etc/kubernetes/pki/ca.key -hostname kubernetes -config
./vxconfig.json -profile cluster ./infoscale-ca.csr | cfssljson
-bare infoscale-ca
```

2. Run the following command to generate the new secrets.

   ```
   sh gen-cert
   ```

3. Run the following command to update the new `infoscale-ca` secret with renewed `infoscale-ca.pem`.

   ```
   kubectl apply -f custom-ca.yaml
   ```

   Wait for upto five minutes.

4. Now you need to delete secrets of the following certificates.

   - infoscale-sds-rest-tls-cert-<value>
   - infoscale-csi-tls-cert
   - infoscale-fencing-tls-cert
   - iso-tls-cert
   - webhook-tls-cert
   - lico-tls-cert

   Run the following command to delete secrets of these certificates.

   ```
   kubectl delete secret -n infoscale-vtas <certificate name>
   ```

5. To enable encryption, perform steps listed in Configuring InfoScale to enable transfer of keys again.

After these certificates are deleted, the cert-manager automatically re-creates the new certificates. Wait for 15 minutes.

---

**Note:** If DR is configured, ensure that you run these commands on the secondary site immediately after running these commands on the primary site.

---

Chapter 8

# InfoScale CSI deployment in Container environment

This chapter includes the following topics:



## CSI plugin deployment

CSI (Container Storage Interface) 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, unzip, and untar `YAML_8.0.310.tar.gz` , 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`
  - `-- csi-dynamic-block-pvc.yaml`
  - `-- csi-block-pod.yaml`
- `-- snapshot-class-templates`
  - `-- csi-infoscale-snapclass.yaml`
- `-- static-provisioning`
  - `-- csi-pod.yaml`
  - `-- csi-static-pvc.yaml`
  - `-- csi-static-pv-k8s.yaml`
  - `-- csi-static-pv-ocp.yaml`
  - `-- csi-static-snapshot-content.yaml`
  - `-- csi-static-snapshot.yaml`
  - `-- csi-static-snapshot-restore.yaml`
  - `-- csi-static-block-pv-k8s.yaml`

    - -- csi-static-block-pv-ocp.yaml
    - -- csi-static-block-pvc.yaml
    - -- csi-static-block-pod.yaml
- -- storage-class-templates
    - -- csi-infoscale-performance-sc.yaml
    - -- csi-infoscale-resiliency-sc.yaml
    - -- csi-infoscale-sc.yaml
    - -- csi-infoscale-topology-sc.yaml
    - -- csi-multicluster-app.yaml
    - -- csi-ephemeral-pod.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
- Enabling raw block volume support

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_<cluster id>` 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.

---

# Raw block volume support

Using raw block volume, the block device is directly exposed instead of exposing the mountpoint inside an application container. Applications can directly write or can create their own native filesystem on the provided block device.

Raw block volumes support is required for applications that are capable of accessing the device directly to store data. Database application is an example where data is directly accessed from the underlying storage, which is more efficient. To enable raw block volume support, `volumeMode` must be set to `Block`.

Following are the FileSystem-persistent volumes features which are supported on Block volumes.

- Create and Delete Volumes
- Expand Volumes
- Clone Volumes
- Create and Delete Snapshot
- Snapshot Restore

---

**Note:** Access mode RWO (ReadWriteOnce) is not supported for block volumes. Creation of a RWO block volume does not fail. The volume is created with RWX (ReadWriteMany) properties even when it is listed as RWO. For ROX (ReadOnlyMany) block volumes, the read-only condition is enforced on non-root users only.

---

For Block volumes, the CSI driver does not format the block device; it just binds the block device to the target path.

# 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:** In case you want to use filesystem-persistent volumes, ensure that the Veritas File System is created before provisioning the volumes. If the Veritas File System does not exist, you must create it manually by using the `mkfs` command from the InfoScale SDS pod.

---

## 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"

  # (optional) Specifies whether to store encrypted data on disks or not.
```

```
# Valid values are true or false
# encryption: "false"

#(optional) Specifies how to initialize a new volume.
# Valid values are "active", "zero" and "sync"
# initType: "active"
```

**Note:** The supported `initType` values are `"sync"`, `"active"`, or `"zero"`.

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

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

Run `oc/kubectl exec -ti -n <namespace> <InfoScale SDS> -- <cmd>` to list Volumes from the InfoScale SDS pod.

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

**3** In the `csi-static-pv-ocp.yaml` or `csi-static-pv-k8s.yaml`, define the Persistent Volume object and specify the existing VxVM volume name in the volumeHandle attribute.

```
csi-static-pv-<ocp/k8s>.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: Retain

  capacity:

    storage: 2Gi

  accessModes:

    - ReadWriteOnce

  csi:

    driver: org.veritas.infoscale

    # Please provide pre-provisioned Infoscale volume name.

    volumeHandle: clust_<cluster_id>/vrts_kube_dg-<cluster_id>/testVol

    fsType: vxfs
```

**Note:** Here, `clusterid` is what you use while configuring `cr.yaml`.

**4** Add the following to `csi-static-pv-ocp.yaml` or `csi-static-pv-k8s.yaml` for node affinity.

```
nodeAffinity:
      required:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.org.veritas.infoscale/cluster
            operator: In
            values:
            - <Name of the InfoScale cluster>
```

**Note:** You can skip this step and add **nodeSelector: topology.org.veritas.infoscale/cluster: "new1"** to `csi-pod.yaml`.

**5** Create a Persistent Volume using the yaml.

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

or

```
kubectl create -f csi-static-pv-k8s.yaml
```

**6** 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
```

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

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

**8** 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
```

**9** Create the application pod.

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

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

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

### Enabling raw block support with static provisioning

1. Run `oc/kubectl exec -ti -n <namespace> <InfoScale SDS> -- <cmd>` to list Volumes from the InfoScale SDS pod. Be ready with the names of the Volumes.
2. Update `volumeHandle` in `csi-static-block-pv-<ocp/k8s>.yaml` as under.

```
---
apiVersion: v1

kind: PersistentVolume
metadata:
  name: csi-infoscale-block-pv
  annotations:
    pv.kubernetes.io/provisioned-by: org.veritas.infoscale
spec:
  volumeMode: Block
  storageClassName: csi-infoscale-sc
  persistentVolumeReclaimPolicy: Retain
  capacity:
    storage: 5Gi

  accessModes:
    - ReadWriteOnce

  csi:
    driver: org.veritas.infoscale

    # Please provide pre-provisioned Infoscale volume name.
    volumeHandle: clust_<cluster_id>/vrts_kube_dg-<cluster_id>/testVol

    volumeAttributes:
      volumePath: "/dev/blk"
```

**Note:** Here, `clusterid` is what you use while configuring `cr.yaml`.

3. Run `oc create -f csi-static-pv-ocp.yaml` or `kubectl create -f csi-static-pv-k8s.yaml` to apply the yaml.

4 Define the Persistent Volume Claim (PVC) with appropriate access mode and storage capacity in `csi-static-block-pvc.yaml` as under.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-infoscale-block-pvc
spec:
  volumeMode: Block
  accessModes:
   - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: csi-infoscale-sc
```

5 Add the following to `csi-static-block-pv-<ocp/k8s>.yaml` for node affinity.

```
nodeAffinity:
      required:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.org.veritas.infoscale/cluster
            operator: In
            values:
            - <Name of the InfoScale cluster>
```

**Note:** You can skip this step and add **nodeSelector: topology.org.veritas.infoscale/cluster: "new1"** to `csi-pod.yaml`.

6 Run `oc/kubectl create -f csi-static-block-pvc.yaml` to apply the yaml.

7 Update the Persistent Volume Claim in `csi-static-block-pod.yaml` as under.

```
---
apiVersion: v1
kind: Pod
metadata:
  name: redis
  labels:
    name: redis
spec:
  containers:
    - name: redis
      image: redis
      imagePullPolicy: IfNotPresent
      volumeDevices:
        - devicePath: "/dev/blk"
          name: vol1
  volumes:
    - name: vol1
      persistentVolumeClaim:
        claimName: csi-infoscale-block-pvc
```

8 Run `oc create -f csi-static-block-pod.yaml` to create the application pod.

# 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.

**Enabling raw block volume support with dynamic provisioning**

**1** Ensure that `volumeMode` is `Block` in as under in `csi-dynamic-block-pvc.yaml` .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-infoscale-block-pvc
spec:
  volumeMode: Block
  storageClassName: csi-infoscale-sc
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 4Gi
```

**2** Run the following command to set Block as the VolumeMode `oc apply -f csi-dynamic-block-pvc.yaml`

Ensure that you resize the file system and drain I/O's from the file system under snapshots and clones for crash consistent snapshots.

After enabling raw block volume support, applications can write to the block device or create filesystem at the block level. Online resizing of block without detaching the block storage is supported. However, Resizing can be to a higher value only.

## 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 SDS 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.
- If this PVC is a part of any Disaster Recovery (DR) configuration, DR controller automatically detects this change and underlying Veritas Volume is expanded on the secondary cluster also. If PVC is resized on a primary cluster and if PVC with an identical name is available on secondary cluster, you must manually update the PVC size on the secondary cluster. Volume is resized automatically on the secondary cluster.

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   CAPACITY
csi-infoscale-pvc Bound   pvc-<id> 5Gi


ACCESS MODES STORAGECLASS     AGE
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
csi-infoscale-pvc  Bound  pvc-<id>


CAPACITY ACCESS MODES STORAGECLASS     AGE
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
csi-infoscale-pvc Bound  pvc-<id> 5Gi


ACCESS MODES STORAGECLASS     AGE
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
csi-infoscale-pvc    Bound    pvc-<id>


CAPACITY   ACCESS MODES STORAGECLASS     AGE
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.

Ensure that you install CRDs version v1 only and snapshot-controller version 6.1.0 or lower. For more details, see https://github.com/kubernetes-csi/external-snapshotter.

---

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> <InfoScale SDS> -- <cmd>
```

to list Volumes from the InfoScale SDS pod. 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-sds-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: clus_<clusterid>/<dgname>/<volname>
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
```

**To restore a raw block volume snapshot to a new PVC**

- Add `volumeMode: Block` to `csi-static-snapshot-restore.yaml` or `csi-dynamic-snapshot-restore.yaml` as under -

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <csi-static-snapshot-restore/csi-dynamic-snapshot-restore>
spec:
  volumeMode: Block
  storageClassName: csi-infoscale-sc
  dataSource:
    name: csi-static-snapshot
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteMany
  # Please provide storage capacity exactly matching to the dataSource
  resources:
    requests:
      storage: 10Gi
```

# 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.

---

## Creating snapshot of a raw block volume

Add `sourceVolumeMode: Block` to `csi-static-snapshot-content.yaml` as under -

```
---
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: csi-static-snapshot-content
spec:
  deletionPolicy: Retain
  driver: org.veritas.infoscale
  source:
    # Provide pre-provisioned Infoscale snapshot volume name
    snapshotHandle: clus_<clusterid>/<dgname>/<volname>
  volumeSnapshotRef:
    name: csi-static-snapshot
    namespace: default
  sourceVolumeMode: Block
```

# 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 137.

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

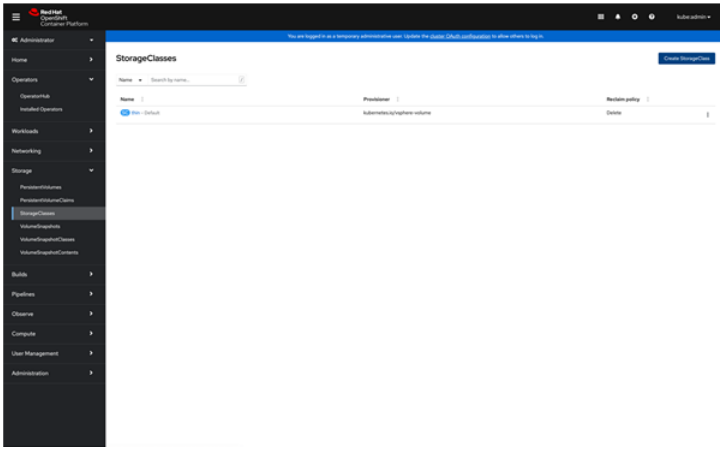
Veritas CSI Driver is a Production Driver. See Kubernetes Drivers for a complete list of Production Drivers.

# Creating CSI Objects for OpenShift

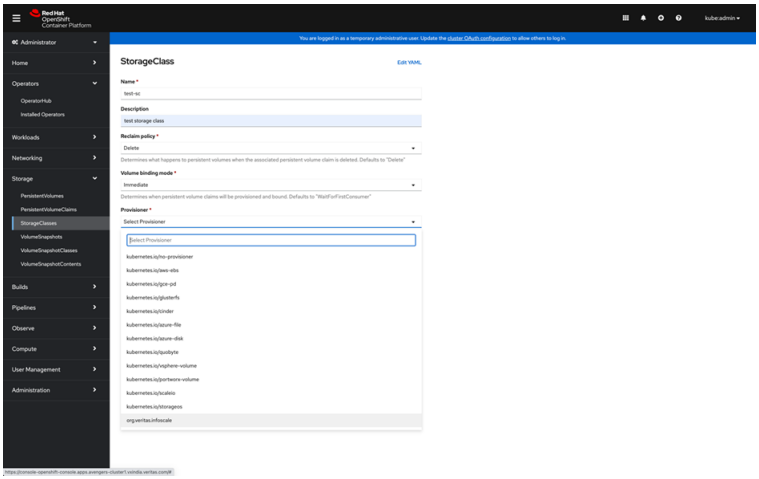
Complete the following steps to install InfoScale operator .

**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.

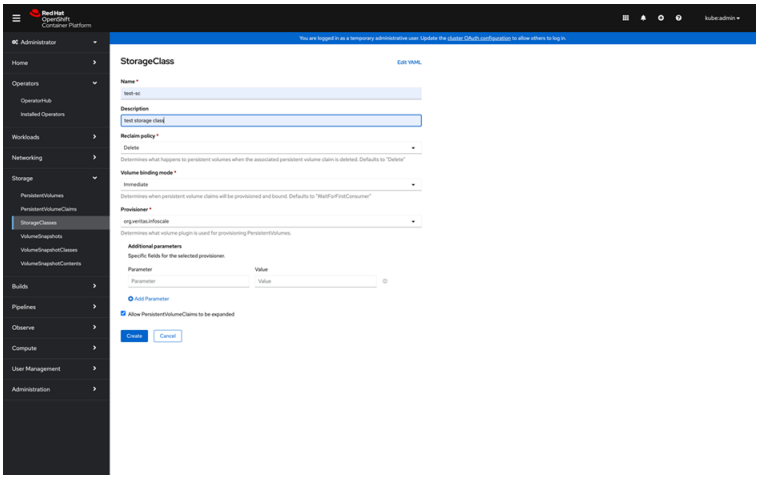


3 Assign a **Name** to the StorageClass in the following screen. Optionally, you can enter Description.




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). |

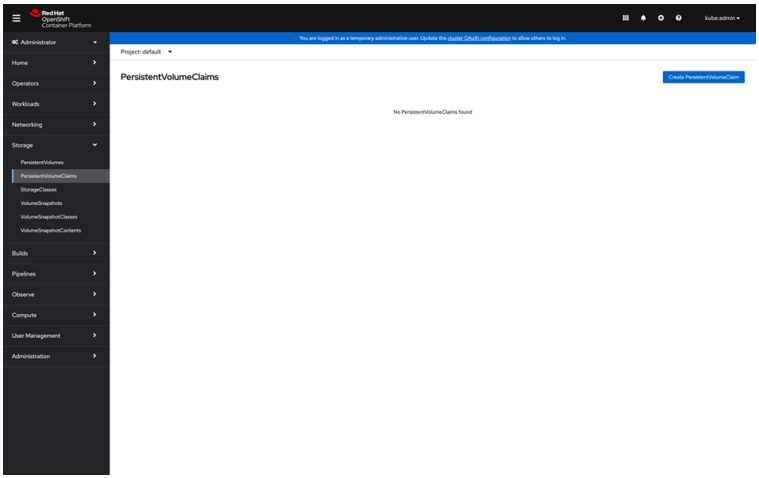
8 Click **Create**. Wait for the StorageClass to be created.

9 Review all details for the StorageClass.

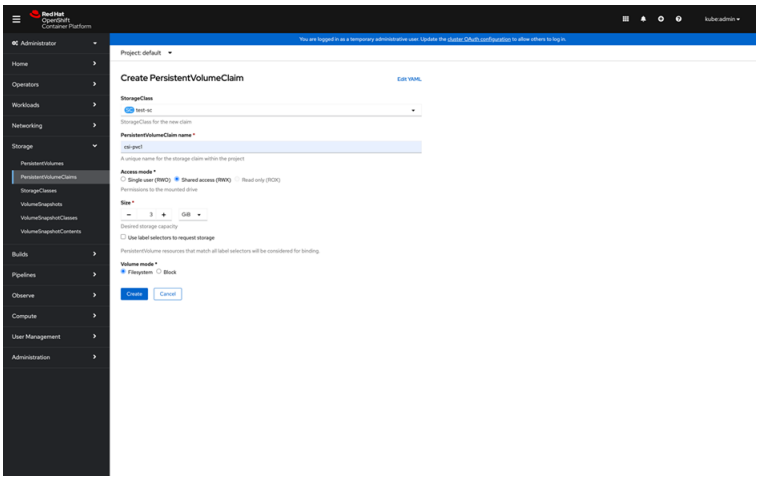
Now you can create a PersistentVolumeClaim.

### 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.

# Creating ephemeral volumes

Some applications like caching services do not require persistent storage. Ephemeral volumes are temporary volumes which such applications can utilize. You can use InfoScale CSI drivers to create generic ephemeral volumes.

Ephemeral volumes can be created in block mode also by setting volumeMode to 'Block'. Snapshotting, cloning, and resizing are also supported. See the sections above to know more.

Complete the following steps to create ephemeral volumes by using an InfoScale CSI driver storage class.

1. Copy the following content and save as `ephemeral_volume_pod.yaml`.

```
kind: Pod
apiVersion: v1
metadata:
  name: test-pod
spec:
  containers:
    - name: test-pod-container
      image: busybox
```

```
    volumeMounts:
    - mountPath: "/ephemeral"
      name: ephemeral-volume
    command: [ "sleep", "1000000" ]
volumes:
  - name: ephemeral-volume
    ephemeral:
      volumeClaimTemplate:
        metadata:
          labels:
            type: test-pod-volume
        spec:
          accessModes: [ "ReadWriteOnce" ]
          storageClassName: "vxvmsc-mirror"
          resources:
            requests:
              storage: 1Gi
```

2. Run the following command to create an application pod with the ephemeral volume.

   ```
   kubectl/oc apply -f ephemeral_volume_pod.yaml
   ```

3. Run the following command to verify.

   ```
   kubectl/oc get pvc
   ```

Chapter 9

# Installing and configuring InfoScale DR Manager 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).

---

**Note:** Multiple InfoScale clusters in different namespaces can be configured within a single OpenShift cluster. However, DR configuration is supported for a single InfoScale cluster per OpenShift cluster only.

---

# Prerequisites

1. InfoScale must be installed and 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 InfoScale DR`.
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. 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 .

# Creating Persistent Volume for metadata backup

Ensure that you run the following steps on the primary and secondary clusters to create a PVC for backing up and restoring application metadata across clusters.

- Create a Storage Class by running the following `sc.yaml`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: infoscale-dr-csi-sc
parameters:
  fstype: vxfs
  layout: mirror
allowVolumeExpansion: true
provisioner: org.veritas.infoscale
reclaimPolicy: Delete
```

- Create a Persistent Volume Claim (PVC) by running the following `pvc.yaml`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: infoscale-dr-meta-bkp-pvc
  namespace: infoscale-vtas
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 4Gi
  storageClassName: infoscale-dr-csi-sc
  volumeMode: Filesystem
```

# External dependencies

- Load balancer service must be installed on all clusters that are a part of Disaster Recovery (DR) to allocate Virtual IP addresses for cluster-to-cluster communications. 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. If you are installing in a cloud environment, ensure that you use native load balancers.

# Installing InfoScale DR Manager by using OLM

You can connect to the Red Hat portal by using web console or the Command Line Interface (CLI).

## Installing InfoScale DR Manager by using web console

Complete the following steps

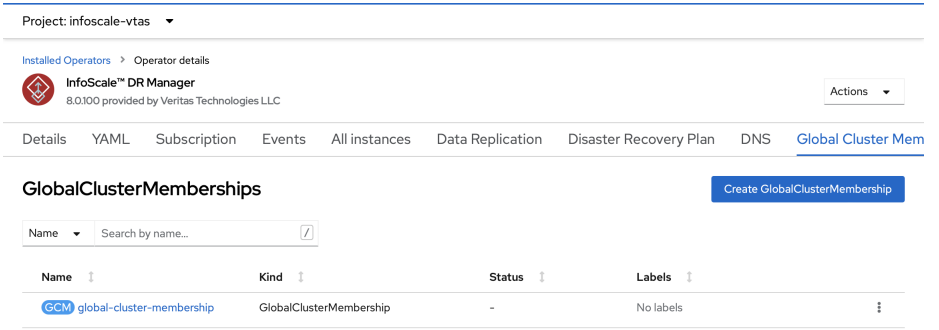
**On the primary cluster and the secondary cluster**

1. See " Installing from OperatorHub by using web console" on page 33. Install InfoScale DR Manager.
2. After the InfoScale DR Manager is installed, you can configure Global Cluster Membership, Data Replication, and Disaster Recovery Plan.

# Configuring InfoScale DR Manager by using web console

Complete the following steps.

1. Click **Operators > Installed Operators**. Select InfoScale DR Manager.
2. In the main menu, click **Global Cluster Membership**. You can create a cluster membership.
3. Click **Create GlobalClusterMembership** in the upper-right corner of the screen as under.




4. Assign a **Name** and click **Global Member Clusters**. You can enter cluster details here.
5. Enter **Cluster ID** of the primary cluster first and its IP address in **DR Controller Address**. Optionally, you can enter its port number in **DR Controller Port**.

---

**Note:** The IP address and port number are generated by using Load Balancer. An example of the command to generate the IP address is - `oc -n infoscale-vtas expose deployment infoscale-dr-manager --name dr-lb-service --type LoadBalancer --protocol TCP --port 14155 --target-port 14155`

The default port number for Load Balancer service in this example is 14155.

---

6 Enter **Cluster ID** of the secondary cluster and its IP address in **DR Controller Address**. Optionally, you can enter its port number in **DR Controller Port**.

---

**Note:** This is the IP address and port number of the peer cluster generated by using Load Balancer. Use the command mentioned in the above step.

---

7 Enter the Cluster ID of the primary cluster in **Local Cluster Name**.

8 Click **Create**. Wait till it is created. The newly created cluster is listed under **Global Cluster Membership**.

9 Repeat steps 1 to 8 on the secondary cluster. In step 7, ensure that you enter the cluster ID of the secondary cluster. After a successful configuration on the secondary cluster, the secondary cluster is the DR cluster.

10 Run the following command on the primary cluster to verify whether GCM is successfully configured - `oc get gcm`

An output similar to the following indicates a successful configuration.

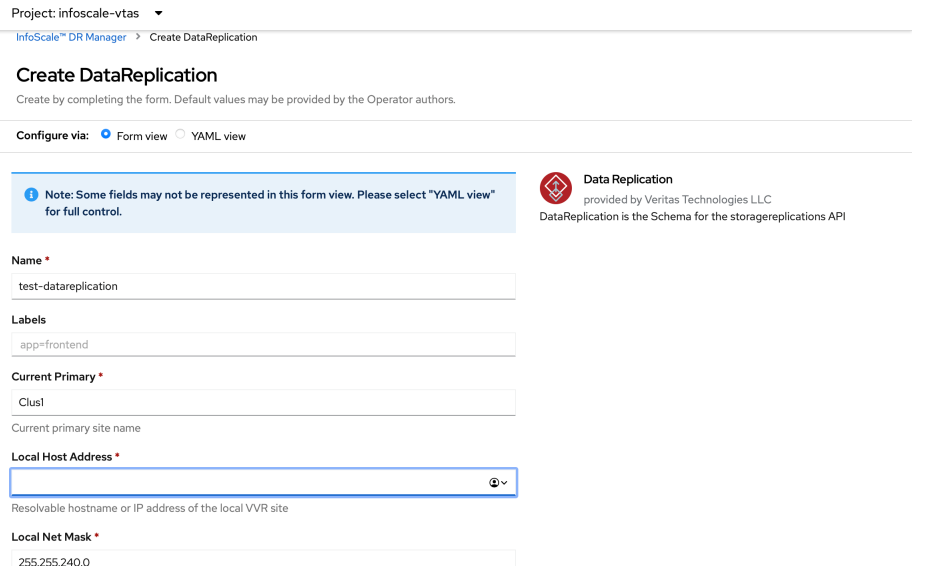
```
NAME                          CLUSTER NAME   CLUSTER STATE   PROTOCOL
global-cluster-membership     Clus1          RUNNING         10



PEER LINK STATE
{"Clus1":"CONNECTED","Clus2":"CONNECTED"}
```

11 You can now configure Data Replication. Select the appropriate project and click **Data Replication**.

12 Click **Create DataReplication** in the upper-right corner of the screen. The following screen opens.




13 For the primary cluster, enter the **Cluster Name**, its IP address (Can be any Virtual IP address available in the same subnet) in **Local Host Address**, and its corresponding Net Mask in **Local Net Mask**.

14 For the secondary cluster, enter its name in **Cluster Name**, its IP address (Can be any Virtual IP address available in the same subnet) in **Remote Host Address**, and its corresponding Net Mask in **Remote Netmask**. Enter the network interface in **Remote NIC**.

15 Enter the **Namespace** for which you want to configure data replication.

16 In **Local NIC**, enter the network interface of the primary cluster.

17 Click **Create**. Wait till Data Replication is configured and gets listed.

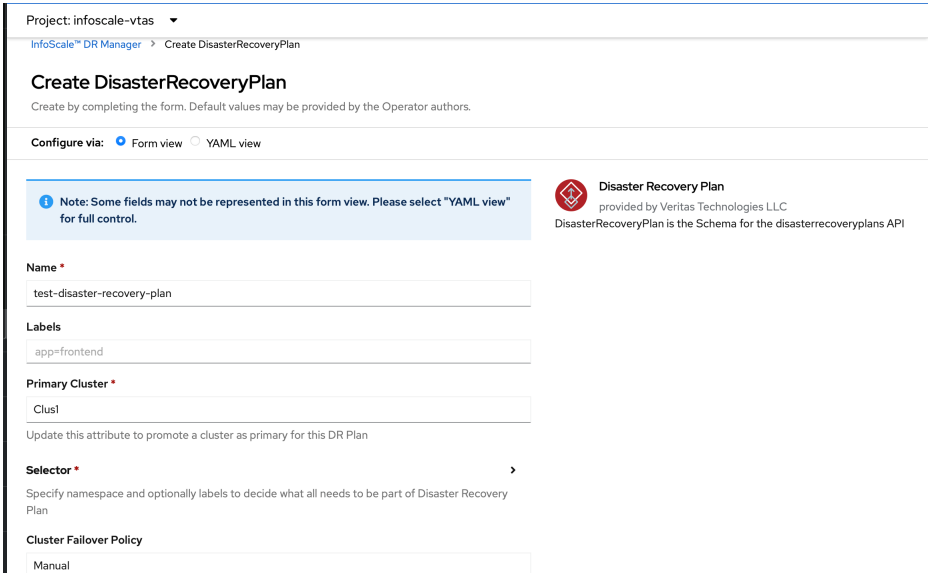
18 Run the following command on the primary cluster to verify whether Data Replication is successfully configured - `oc get datarep -o wide`

19 Review output for

```
consistent,up-to-date
```

20 Click **Disaster Recovery Plan** in the main menu to create a plan.

21 Click **Create DisasterRecoveryPlan** in the upper-right corner of the screen. The following screen opens.




22 Assign a **Name** to this plan.

---

**Note:** The names of clusters and the Data replication plan appear here.

---

23 Review the **Primary Cluster**. Enter the **Namespace** that you want to be a part of this plan.

24 Review the **Data Replication Pointer** and **Preferred Cluster List**.

25 Click **Create** and wait till the Disaster Recovery Plan is created and listed.

After these successful configurations, Disaster Recovery (DR) is ready.

# Installing from OperatorHub by using Command Line Interface (CLI)

After you download `YAML_8.0.310.tar.gz` from Veritas Download Center and unzip and untar it, a folder `/YAML/OpenShift/OLM/` is created and all files required for installation are available in the folder. An OpenShift cluster already has a namespace `openshift-operators`. You can choose to install InfoScale DR Manager in `openshift-operators`.

Complete the following steps on the primary and secondary clusters.

1 Copy the following content to a file and save the file as `infoscale-dr-sub.yaml`.

```
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: infoscale-dr-manager-sub
  # Namespace should be either infoscale-vtas
  # or openshift-operators
  namespace: infoscale-vtas
spec:
  channel: stable
  name: infoscale-dr-manager
  installPlanApproval: Manual
  # Source can be infoscale-dr-manager-catalog
  # if infoscale-dr-catsrc.yaml is used
  source: certified-operators
  sourceNamespace: openshift-marketplace
  startingCSV: infoscale-dr-manager.v8.0.310
```

2 Run the following command on the bastion node to create a namespace infoscale-vtas. Ignore 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
```

**3** Run the following command on the bastion node to create subscription.

---

**Note:** If you want to install InfoScale DR Manager in `openshift-operators` namespace, edit `infoscale-dr-sub.yaml`. Change namespace from `infoscale-vtas` to `openshift-operators`. To install the latest bundle, modify **startingCSV** to **infoscale-dr-manager.v8.0.310**.

---

```
oc create -f infoscale-dr-sub.yaml
```

Following output indicates a successful command run.

```
subscription.operators.coreos.com/infoscale-dr-manager-sub created
```

**4** Run the following command on the bastion node.

```
oc get subscriptions -n infoscale-vtas
```

Following output indicates a successful command run.

```
NAME                                PACKAGE
infoscale-dr-manager                infoscale-dr-manager
infoscale-licensing-operator        infoscale-licensing-operator
infoscale-operator                  infoscale-operator
nfd-stable-redhat-operators
             openshift-marketplace  nfd

SOURCE                               CHANNEL
infoscale-dr-manager-catalog         stable
infoscale-lic-operator-test-catsrc   stable
infoscale-sds-operator-catalog       stable
redhat-operators                     stable
```

**5** Run the following command on the bastion node.

Change `infoscale-vtas` to `openshift-operators`, if you are using that namespace.

```
oc get installplans --namespace infoscale-vtas
```

Use `<installation-name>` from the output similar to the following output

```
NAME                  CSV                              APPROVAL   APPROVED
<installation-name>   infoscale-dr-manager.v8.0.310    Manual     false
```

**6** Run the following command on the bastion node.

```
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 installplans --namespace infoscale-vtas
```

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

```
NAME                CSV                      APPROVAL APPROVED
<installation-name> infoscale-dr-manager.  Manual   True
                                  v8.0.310
```

**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. Run the command again till you see the phase for all components as 'Succeeded'

```
NAME                                      DISPLAY
cert-manager.v1.12.0                      cert-manager
infoscale-dr-manager.v8.0.310             InfoScale™ DR Manager
infoscale-licensing-operator.v8.0.310  InfoScale Licensing Operator
infoscale-operator.v8.0.310               InfoScale™ SDS Operator
nfd.4.12.0-202307170916                   Node Feature Discovery Operator
```

```
VERSION               REPLACES                      PHASE
1.12.0                cert-manager.v1.12.0          Succeeded
8.0.310                                             Succeeded
8.0.310                                             Succeeded
8.0.310               infoscale-operator.v8.0.20    Succeeded
4.10.0-202206010607                                 Succeeded
```

9 After InfoScale-dr-manager CSV is in a 'Succeeded' phase, proceed with the custom resource configuration and creation.

10 Follow steps listed in Configuring Global Cluster Membership (GCM), Configuring Data Replication, Additional requirements for replication on Cloud, Configuring DNS, and Configuring Disaster Recovery Plan.

# Installing InfoScale DR Manager by using YAML

This section informs you how to install and configure Disaster Recovery for your InfoScale cluster by using YAML.

---

**Note:** When you download, unzip, and untar `YAML_8.0.310.tar.gz` , all files required for installation are available.

---

Complete the following steps to install the InfoScale DR Manager on the source and the target DR cluster.

**Creating application user role**

1 Run the following command on the bastion node.

```
oc apply -f YAML/DR/infoscale-dr-admin-role.yaml
```

2 Copy the following content to `YAML/DR/ClusterRoleBinding .yaml`.

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
   name: infoscaledr-role-binding
roleRef:
   apiGroup: rbac.authorization.k8s.io
   kind: ClusterRole
   name: infoscaledr-admin-role
subjects:
-  kind: User
   name: postgres-admin
   apiGroup: rbac.authorization.k8s.io
```

This is an example for a ClusterRoleBinding role.

3 Run the following command on the bastion node.

```
oc apply -f YAML/DR/ClusterRoleBinding.yaml
```

For DR controller installation and Global Cluster Membership (GCM) configuration, you must use the kubeadmin role only. To configure DR plan and data replication, you can use this role or the kubeadmin role.

1. Run the following command on the bastion node of each cluster.

   ```
   oc apply -f /YAML/DR/OpenShift/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
   infoscale-dr-manager-xxxx       1/1     Running     0            114m
   ```

   Status must change from `ContainerCreating` to `Running`.

4. Run the following commands to configure ExternalIP for DR pod.

   ---

   **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 infoscale-dr-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:
  # Local cluster name in the global membership
  localClusterName: <Local cluster
                          where you want to apply this YAML>
  globalMemberClusters:
      # Cluster ID of each member of global cluster membership
    - clusterID: <A unique ID of the primary cluster>
      # Address Used For Communicating With Peer Cluster's DR Controller
      drControllerAddress: "<Load balancer IP address or haproxy
                                              of the local cluster>"
      # Port used for DR controller
      drControllerPort: "<Load balancer port number>"
    - clusterID: <A unique ID of the secondary cluster>
      drControllerAddress: "<Load balancer IP address or haproxy
                                               of the DR site>"
      drControllerPort: "<Load balancer port number>"
  # If heartbeat with peer cluster missed more than CounterMissTolerance
        times,then cluster will be moved to FAULTED state
  counterMissTolerance: 5
  globalClusterOperation: "none"
  # Application metadata backup sync frequency to DR site(s) in minutes
  metadataBackupInterval: 15
  # Refresh data replication status after specified minutes
```

```
  datarepRefreshStatusFrequency: 10
  # Include cluster-scoped Custom Resource Definitions (CRDs)
  #                          in disaster recovery plan backup
  backupClusterScopeCRD: true
  # Maximum metadata backup copies stored per DR plan
  maximumMetadataCopies: 5
```

**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:
  # Local cluster name in the global membership
localClusterName: <DR site cluster name
                       where you want to apply this YAML>

  globalMemberClusters:
  # Cluster ID of each member of global cluster membership
   - clusterID: <A unique ID of the primary cluster>
  # Address Used For Communicating With Peer Cluster's DR Controller
    drControllerAddress: "<Load balancer IP address or haproxy
                                  of the local cluster>"

  # Port used for DR controller
     drControllerPort: "<Load balancer port number>"
      - clusterID: <A unique ID of the secondary cluster>
     drControllerAddress: "<Load balancer IP address or haproxy
     of the DR site>"
      drControllerPort: "<Load balancer port number>"
  # If heartbeat with peer cluster missed more than CounterMissTolerance
```

```
# times, then cluster will be moved to FAULTED state
counterMissTolerance: 5
globalClusterOperation: "none"
# Application metadata backup sync frequency to DR site(s) in minutes
metadataBackupInterval: 15
# Refresh data replication status after specified minutes
datarepRefreshStatusFrequency: 10
# Include cluster-scoped Custom Resource Definitions (CRDs) in
# disaster recovery plan backup
backupClusterScopeCRD: true
# Maximum metadata backup copies stored per DR plan
maximumMetadataCopies: 5
```

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. If GCM and member cluster details are identical on all peer clusters then all clusters automatically transition to RUNNING state. If the details are not identical, 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. |

| State | Description |
|---|---|
| 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. |
| EXITED | DR Controller stopped. |

6. To verify whether the Global Cluster is successfully created, run the following command on the bastion node.

   ```
   oc get gcm
   ```

7. Review the cluster names, GlobalClusterState, and PeerLinkState in the output similar to the following. GlobalClusterState must be Running and PeerLinkState must be Connected.

   ```
   NAME                        CLUSTER NAME   CLUSTER STATE   PROTOCOL
   global-cluster-membership Clus1          RUNNING         10


   PEER LINK STATE
   {"Clus1":"CONNECTED","Clus2":"CONNECTED"}
   ```

   Here, `NAME` is the `Name of GlobalClusterMembership custom resource`, `CLUSTER NAME` is the `Local Cluster ID`, and `Clus1`, `Clus2` are the Cluster IDs that you defined for global membership.

# 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.

If you are configuring data replication for an on-premise and cloud combination, ensure that you select the appropriate value for `cloudVendor` and for load balancer-based network traffic management, set `cvmaster:true`. On ARO, load balancer is not supported.

**Note:** If a node specified in the data replication custom resource is down, data replication configuration is unresponsive. After the node is back, data replication is configured.

Complete the following steps

1. Edit `/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>
  #namespace: <Enter the namespace>
spec:
  # In case of load balancer based n/w traffic management,
  # set lbEnabled to true.
  #  else the default value should be always kept false.
  lbEnabled : false
  # Virtual IP address to configure VVR
  localHostAddress: <Any free Virtual IP address
                to configure VVR for the primary cluster>
  # Corresponding netmask to configure VVR
  localNetMask: <Corresponding netmask to configure VVR>
  # Corresponding network interface to configure VVR
  # (If NIC name is identical for all nodes)
  localNIC: eth0
  # Corresponding network interface list (hostname and NIC name )
  # to configure VVR
  # (If NIC name is not identical for all nodes)
  #localNICList:
  #  - "host1=eth0"
  #  - "host2=eth0"
  #  - "host3=eth0"
  #  - "host4=eth1"

  # (optional) Cloud Vendor (e.g Azure/Aws) on Primary VVR site.
  cloudVendor: Local

  # (optional) Applicable for Cloud-Vendor based environments,
  # If "localHostAddress" value is an Overlay N/w IP, then specify
```

```
#  all applicable Route table resource ids.
#routeTableResourceIds:
#  - "rtb-fb97ac9d"
#  - "rtb-f416eb8d"
#  - "rtb-e48be49d"

# Namespace and optionally labels for which you want
#  to configure data replication
selector:
  namespace: mysql
  #labels:
  #  - "app=db,env=dev"

# 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 currentPrimary 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>
    # In case of load balancer based n/w traffic management,
    #  set remoteLbEnabled to true.
    #  else the default value should be always kept false.
    remoteLbEnabled : false
    # Virtual IP address for VVR configuration of this cluster
    remoteHostAddress: <Any free Virtual IP address for
                        VVR configuration of the remote cluster>
    # Corresponding Netmask of this cluster
    remoteNetMask: <Corresponding netmask of the remote cluster>
    # Corresponding Network interface of this cluster
    remoteNIC: eth0
    # Corresponding Network interface list of this cluster
```

```
#remoteNICList:
#  - "host5=eth1"
#  - "host6=eth0"
#  - "host7=eth0"
#  - "host8=eth1"

# (optional) Cloud Vendor (e.g Azure/Aws) on remote VVR site.
remoteCloudVendor: Local

# (optional) Applicable for Cloud-Vendor based environments,
# If "remoteHostAddress" value is an Overlay N/w IP,
# then specify all applicable Route table resource ids.
#remoteRouteTableResourceIds:
#  - "rtb-fb97ac9d"
#  - "rtb-f416eb8d"
#  - "rtb-e48be49d"

# (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/pause/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. If you want to exclude any PVCs within the same namespace from replication, label the PVCs as `` `org.veritas.infoscale/exclude-from-backup=true` ``.

---

2. Run the following command on the bastion node

   ```
   oc apply -f /YAML/DR/SampleDataReplication.yaml
   ```

3. After these commands are executed, run the following commands on the bastion node

   ```
   oc get datarep
   ```

   Review the output similar to the following

   ```
   NAME PROPOSED PRIMARY CURRENT PRIMARY NAMESPACE LABELS
   <Name of data replication resource> <proposed cluster ID>
        <current cluster ID> <namespace> <labels if any>
   ```

   ```
   oc get datarep -o wide
   ```

   Review the output similar to the following

   ```
   NAME PROPOSED PRIMARY CURRENT PRIMARY NAMESPACE LABELS
   postgres-rep Clus1 Clus1 postgres <none>
    | replicating (connected) | behind by 0h 0m 0s


   REPLICATION   SUMMARY
   asynchronous | consistent,up-to-date
   ```

4. 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 datarep <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
   ```

## Additional requirements for replication on Cloud

If any of your sites is on the Cloud, you must provide High Availability (HA) to the Virtual IP addresses and configure load balancer service. Ensure that you have specified the cloud service for `cloudVendor` or `remoteCloudVendor` while configuring data replication.

In Azure and AWS environments, to provide High Availability (HA) to the Virtual IP addresses; Cluster server networking agents are used. Veritas Volume Replicator (VVR) uses the Virtual IP addresses for data replication. For this configuration with AWS Virtual IP addresses (AWSIP), roles must be attached to the worker nodes. See *Network Agents section* in *Cluster Server Bundled Agents Reference Guide* for AzureIP and AWSIP agent details and for the Identity Access Management (IAM) role creation for AWSIP agent. For AzureIP agent, Azure Authentication credentials are accepted as a Kubernetes secret . `Secret key`of AzureAuth is stored in an encrypted format by using `vcsencrypt`.

Load balancer can be used for both managed and non-managed clouds. lbEnabled and remoteLbEnabled are set in the datareplication yaml. Default value is false (set to true only in case where network traffic goes over load balancer). In this configuration, load balancer Virtual IP address must be provided as `HostAddress` (local and/or remote) in the datareplication yaml. The prerequisite for this feature is that the load balancer network Kubernetes service must have the following selector

in its spec- `cvmaster:true`. The sample files below are examples of Azure authentication secret and load balancer service.

---

**Note:** The TCP/UDP ports that Veritas Volume Replicator (VVR) uses must be open on all worker nodes of the cluster to enable communication between primary and secondary site. See *Choosing the network ports used by VVR* on the Veritas support portal.

---

**1** Update and copy the following content into a yaml file and save as `/YAML/DR/AzureAuth.yaml`.

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  # do not change "name" value.
  name:  infoscale-azure-auth
  namespace: infoscale-vtas
data:
  # base64-coded valid Identifier that uniquely
  # identifies your Azure subscription.
  subscriptionId: "aW5mb3NjYWxl"
  # base64-coded valid Identifier of the
  # Azure Active Directory (AAD) Application.
  clientId: "aW5mb3NjYWxlY2xpZW50"
  # base64-coded valid Authentication
  # key generated for the AAD application.
  secretKey:  "aW5mb3NjYWxlc2VjcmV0"
  # base64-coded valid Identifier of the
  # AAD directory in which you created the application.
  tenantId: "aW5mb3NjYWxldGVuYW50SWQ="
```

**2** Run the following command on the bastion node.

```
oc apply -f /YAML/DR/AzureAuth.yaml
```

**3** Perform the following steps only if you want to use a Load balancer for data replication.

Copy the following content into a yaml file and save the file as `/YAML/DR/vvr-lb-svc.yaml`.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/azure-load-balancer-internal: "true"

  name: vvr-lb-svc
  namespace: infoscale-vtas
spec:
  allocateLoadBalancerNodePorts: true
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: tcpportone
    port: 4145
    protocol: TCP
    targetPort: 4145
  - name: tcpporttwo
    port: 8199
    protocol: TCP
    targetPort: 8199
  - name: tcpportthree
    port: 8989
    protocol: TCP
    targetPort: 8989
  selector:
    cvmmaster: "true"
  sessionAffinity: None
  type: LoadBalancer
```

**4** Run the following command on the bastion node.

```
oc apply -f /YAML/DR/vvr-lb-svc.yaml
```

**5** Run the following command on the bastion node.

```
oc get svc vvr-lb-svc  -n infoscale-vtas
```

Load Balancer IP address is returned as the output.

**6** Update the load balancer IP address and copy the following content into a yaml file and save as `/YAML/DR/loadbalancer.yaml`.. Veritas Volume Replicator (VVR) requires both ports - TCP and UDP. Hence, a Load balancer service with mixed protocol support (TCP and UDP) is needed.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/azure-load-balancer-internal:"true"
  service.beta.kubernetes.io/azure-load-balancer-internal-subnet:"worker"
  name: vvr-lb-svc
  namespace: infoscale-vtas
spec:
  loadBalancerIP: <Load Balancer IP address>
  allocateLoadBalancerNodePorts: true
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: tcpportone
    port: 4145
    protocol: UDP
  - name: tcpporttwo
    port: 8199
    protocol: UDP
    targetPort: 8199
  - name: tcpportthree
    port: 8989
    protocol: UDP
    targetPort: 8989
  selector:
    cvmaster: "true"
  sessionAffinity: None
  type: LoadBalancer
```

**7** Run the following command on the bastion node.

```
oc apply -f /YAML/DR/loadbalancer.yaml
```

# 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. |
| 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 |

**Applicable only to Secure Linux DNS.**

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 and remove all unnecessary spaces from the <dns private key> that is displayed.
- Run the following command on the bastion node
  `cat dns.key | base64`
  Copy and remove all unnecessary spaces from the <dns key> that is displayed.
- Run the following command on the bastion node to generate the keys
  `oc get secret -n infoscale-vtas |grep infoscale-sds-dns-secret`
  Use the output in the following command.
- Run the following command on the bastion node and add the keys
  `oc edit secret <output for infoscale-sds-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

**Note:** Ensure that this resource is applied in the same namespace as that of data replication configuration resource.

```
apiVersion: infoscale.veritas.com/v1
kind: DNS
metadata:
  name: <Name of DNS>
  #namespace: <Enter the namespace>
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: "/etc/vx/dns-certs/dns.private"

  # (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:
        #"HostName_1" : "100.200.30.41"
        #"HostName_2" : "100.200.30.42"
        #"HostName_3" : "100.200.30.43"
        #"www" : "HostName_1"
        #"abc" : "HostName_2"
        #"xyz" : "HostName_3"

  # (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**, **namespace**, 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.

---

**Note:** Clusterwide(across namespaces) RBAC permissions are granted to the service account of `infoscale-dr-manager` pod as required. As a Storage Administrator, ensure that a non-privileged user does not have `oc exec` permissions on the infoscale-dr-manager pod.

---

1 Edit `/YAML/DR/SampleDisasterRecoveryPlan.yaml` as under to create DR plan for application components in a given namespace.

**Note:** Ensure that this resource is applied in the same namespace as that of data replication configuration resource.

```
apiVersion: infoscale.veritas.com/v1
kind: DisasterRecoveryPlan
metadata:
  name: <Name of the Disaster Recovery Plan>
  #namespace: <Enter the namespace>
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: mysql
    #labels:
    #  - "app=db,env=dev"

  # (optional) Pointer to manage data replication
  #dataReplicationPointer: test-datareplication
```

```
  # (optional) Pointer to manage DNS endpoints
  #dnsPointer: test-dns
```

**Note: 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
```

**Note:** Ensure that the Namespace you enter here is the same namespace specified in the data replication configuration resource.

**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 10

# Installing and configuring InfoScale DR Manager on Kubernetes

This chapter includes the following topics:



## Introduction

**Note:** Version 8.0.310 is not qualified for Kubernetes. Check the supported platform support matrix for more information. See “Supported platforms” on page 17.For installing on Kubernetes, refer to Veritas InfoScale™ for Kubernetes Environments 8.0.300 - Linux documentation on Veritas SORT.

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

**Note:** Multiple InfoScale clusters in different namespaces can be configured within a single Kubernetes cluster. However, DR configuration is supported for a single InfoScale cluster per Kubernetes cluster only.

# Prerequisites

1. InfoScale must be installed and 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 InfoScale DR .`
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. 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 .

# Creating Persistent Volume for metadata backup

Ensure that you run the following steps on the primary and secondary clusters to create a PVC for backing up and restoring application metadata across clusters.

- Create a Storage Class by running the following `sc.yaml`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: infoscale-dr-csi-sc
parameters:
  fstype: vxfs
  layout: mirror
allowVolumeExpansion: true
provisioner: org.veritas.infoscale
reclaimPolicy: Delete
```

- Create a Persistent Volume Claim (PVC) by running the following `pvc.yaml`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: infoscale-dr-meta-bkp-pvc
  namespace: infoscale-vtas
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 4Gi
  storageClassName: infoscale-dr-csi-sc
  volumeMode: Filesystem
```

# External dependencies

- Load balancer service must be installed on all clusters that are a part of Disaster Recovery (DR) to allocate Virtual IP addresses for cluster-to-cluster communications. 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. If you are installing in a cloud environment, ensure that you use native load balancers.

# Installing InfoScale DR Manager

This section informs you how to install and configure Disaster Recovery for your InfoScale cluster.

---

**Note:** When you download, unzip, and untar `YAML_8.0.310.tar.gz`, all files required for installation are available.

---

Complete the following steps to install the InfoScale DR Manager on the source and target DR cluster.

### Creating application user role

1. Run the following command on the master node.

   ```
   kubectl apply -f YAML/DR/infoscale-dr-admin-role.yaml
   ```

2. Copy the following content to `YAML/DR/ClusterRoleBinding .yaml`.

   ```
   ---
   apiVersion: rbac.authorization.k8s.io/v1
   kind: ClusterRoleBinding
   metadata:
      name: infoscaledr-role-binding
   roleRef:
      apiGroup: rbac.authorization.k8s.io
      kind: ClusterRole
      name: infoscaledr-admin-role
   subjects:
   -  kind: User
      name: postgres-admin
      apiGroup: rbac.authorization.k8s.io
   ```

   This is an example for a ClusterRoleBinding role.

3. Run the following command on the master node.

   ```
   kubectl apply -f YAML/DR/ClusterRoleBinding.yaml
   ```

For DR controller installation and Global Cluster Membership (GCM) configuration, you must use the kubeadmin role only. To configure DR plan and data replication, you can use this role or the kubeadmin role.

1. Edit `/YAML/DR/Kubernetes/dro_deployment.yaml` to update the path with private repository path where InfoScale DR Manager image is saved and pulled for installation.

2. Run the following command on the master node of each cluster.

   ```
   kubectl apply -f /YAML/DR/Kubernetes/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
infoscale-dr-manager-xxxx       1/1     Running     0           114m
```

Status must change from `ContainerCreating` to `Running`.

5 Run the following commands to configure ExternalIP for DR pod.

---

**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 infoscale-dr-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:
  # Local cluster name in the global membership
  localClusterName: <Local cluster
                          where you want to apply this YAML>
  globalMemberClusters:
      # Cluster ID of each member of global cluster membership
    - clusterID: <A unique ID of the primary cluster>
      # Address Used For Communicating With Peer Cluster's DR Controller
      drControllerAddress: "<Load balancer IP address or haproxy
                                            of the local cluster>"
      # Port used for DR controller
      drControllerPort: "<Load balancer port number>"
    - clusterID: <A unique ID of the secondary cluster>
      drControllerAddress: "<Load balancer IP address or haproxy
                                             of the DR site>"
      drControllerPort: "<Load balancer port number>"
  # If heartbeat with peer cluster missed more than CounterMissTolerance
        times,then cluster will be moved to FAULTED state
  counterMissTolerance: 5
  globalClusterOperation: "none"
  # Application metadata backup sync frequency to DR site(s) in minutes
  metadataBackupInterval: 15
  # Refresh data replication status after specified minutes
  datarepRefreshStatusFrequency: 10
  # Include cluster-scoped Custom Resource Definitions (CRDs)
  #                           in disaster recovery plan backup
  backupClusterScopeCRD: true
```

```
  # Maximum metadata backup copies stored per DR plan
  maximumMetadataCopies: 5
```

**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:
  # Local cluster name in the global membership
localClusterName: <DR site cluster name
                       where you want to apply this YAML>

  globalMemberClusters:
  # Cluster ID of each member of global cluster membership
   - clusterID: <A unique ID of the primary cluster>
  # Address Used For Communicating With Peer Cluster's DR Controller
    drControllerAddress: "<Load balancer IP address or haproxy
                                  of the local cluster>"

  # Port used for DR controller
     drControllerPort: "<Load balancer port number>"
      - clusterID: <A unique ID of the secondary cluster>
     drControllerAddress: "<Load balancer IP address or haproxy
     of the DR site>"
      drControllerPort: "<Load balancer port number>"
  # If heartbeat with peer cluster missed more than CounterMissTolerance
  # times, then cluster will be moved to FAULTED state
  counterMissTolerance: 5
  globalClusterOperation: "none"
  # Application metadata backup sync frequency to DR site(s) in minutes
```

```
metadataBackupInterval: 15
# Refresh data replication status after specified minutes
datarepRefreshStatusFrequency: 10
# Include cluster-scoped Custom Resource Definitions (CRDs) in
# disaster recovery plan backup
backupClusterScopeCRD: true
# Maximum metadata backup copies stored per DR plan
maximumMetadataCopies: 5
```

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. If GCM and member cluster details are identical on all peer clusters then all clusters automatically transition to RUNNING state. If the details are not identical, 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. |

| State | Description |
|---|---|
| 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. |
| EXITED | DR Controller stopped. |

6. To verify whether the Global Cluster is successfully created, run the following command on the master node.

   ```
   kubectl get gcm
   ```

7. Review the cluster names, GlobalClusterState, and PeerLinkState in the output similar to the following. GlobalClusterState must be Running and PeerLinkState must be Connected.

   ```
   NAME                        CLUSTER NAME   CLUSTER STATE   PROTOCOL
   global-cluster-membership Clus1          RUNNING         10


   PEER LINK STATE
   {"Clus1":"CONNECTED","Clus2":"CONNECTED"}
   ```

   Here, `NAME` is the `Name of GlobalClusterMembership custom resource`, `CLUSTER NAME` is the `Local Cluster ID`, and `Clus1`, `Clus2` are the Cluster IDs that you defined for global membership.

# 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.

If you are configuring data replication for an on-premise and cloud combination, ensure that you select the appropriate value for `cloudVendor` and for load balancer-based network traffic management, set `cvmaster:true`.

**Note:** If a node specified in the data replication custom resource is down, data replication configuration is unresponsive. After the node is back, data replication is configured.

Complete the following steps

1. Edit `/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>
  #namespace: <Enter the namespace>
spec:
  # In case of load balancer based n/w traffic management,
  # set lbEnabled to true.
  #  else the default value should be always kept false.
  lbEnabled : false
  # Virtual IP address to configure VVR
  localHostAddress: <Any free Virtual IP address
                to configure VVR for the primary cluster>
  # Corresponding netmask to configure VVR
  localNetMask: <Corresponding netmask to configure VVR>
  # Corresponding network interface to configure VVR
  # (If NIC name is identical for all nodes)
  localNIC: eth0
  # Corresponding network interface list (hostname and NIC name )
  # to configure VVR
  # (If NIC name is not identical for all nodes)
  #localNICList:
  #  - "host1=eth0"
  #  - "host2=eth0"
  #  - "host3=eth0"
  #  - "host4=eth1"

  # (optional) Cloud Vendor (e.g Azure/Aws) on Primary VVR site.
  cloudVendor: Local

  # (optional) Applicable for Cloud-Vendor based environments,
  # If "localHostAddress" value is an Overlay N/w IP, then specify
```

```
#  all applicable Route table resource ids.
#routeTableResourceIds:
#  - "rtb-fb97ac9d"
#  - "rtb-f416eb8d"
#  - "rtb-e48be49d"

# Namespace and optionally labels for which you want
#  to configure data replication
selector:
  namespace: mysql
  #labels:
  #  - "app=db,env=dev"

# 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 currentPrimary 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>
    # In case of load balancer based n/w traffic management,
    #  set remoteLbEnabled to true.
    #  else the default value should be always kept false.
    remoteLbEnabled : false
    # Virtual IP address for VVR configuration of this cluster
    remoteHostAddress: <Any free Virtual IP address for
                        VVR configuration of the remote cluster>
    # Corresponding Netmask of this cluster
    remoteNetMask: <Corresponding netmask of the remote cluster>
    # Corresponding Network interface of this cluster
    remoteNIC: eth0
    # Corresponding Network interface list of this cluster
```

```
#remoteNICList:
#  - "host5=eth1"
#  - "host6=eth0"
#  - "host7=eth0"
#  - "host8=eth1"

# (optional) Cloud Vendor (e.g Azure/Aws) on remote VVR site.
remoteCloudVendor: Local

# (optional) Applicable for Cloud-Vendor based environments,
# If "remoteHostAddress" value is an Overlay N/w IP,
# then specify all applicable Route table resource ids.
#remoteRouteTableResourceIds:
#  - "rtb-fb97ac9d"
#  - "rtb-f416eb8d"
#  - "rtb-e48be49d"

# (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/pause/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. If you want to exclude any PVCs within the same namespace from replication, label the PVCs as `` `org.veritas.infoscale/exclude-from-backup=true` ``.

---

2. Run the following command on the master node

   ```
   kubectl apply -f /YAML/DR/SampleDataReplication.yaml
   ```

3. After these commands are executed, run the following commands on the master node

   ```
   kubectl get datarep
   ```

4. Review the output similar to the following

   ```
   NAME PROPOSED PRIMARY CURRENT PRIMARY NAMESPACE LABELS
   <Name of data replication resource> <proposed cluster ID>
               <current cluster ID> <namespace> <labels if any>
   ```

   ```
   kubectl get datarep -o wide
   ```

   Review the output similar to the following

   ```
   NAME PROPOSED PRIMARY CURRENT PRIMARY NAMESPACE LABELS
   postgres-rep Clus1 Clus1 postgres <none>
    | replicating (connected) | behind by 0h 0m 0s
   ```

   ```
   REPLICATION   SUMMARY
   asynchronous | consistent,up-to-date
   ```

5. 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 datarep <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
   ```

## Additional requirements for replication on Cloud

If any of your sites is on the Cloud, you must configure load balancer service. Ensure that you have specified the cloud service for `cloudVendor` or `remoteCloudVendor` while configuring data replication.

Load balancer can be used for both managed and non-managed clouds. lbEnabled and remoteLbEnabled are set in the datareplication yaml. Default value is false (set to true only in case where network traffic goes over load balancer).

In this configuration, load balancer Virtual IP address must be provided as `HostAddress` (local and/or remote) in the datareplication yaml. The prerequisite for this feature is that the load balancer network Kubernetes service must have the following selector in its spec- `cvmaster:true`. The sample file below is an example of load balancer service.

---

**Note:** The TCP/UDP ports that Veritas Volume Replicator (VVR) uses must be open on all worker nodes of the cluster to enable communication between primary and secondary site. See *Choosing the network ports used by VVR* on the Veritas support portal.

---

**Note:** Perform the following steps only if you want to use a Load balancer for data replication.

1 Copy the following content into a yaml file and save the file as `/YAML/DR/vvr-lb-svc.yaml`.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/azure-load-balancer-internal: "true"

  name: vvr-lb-svc
  namespace: infoscale-vtas
spec:
  allocateLoadBalancerNodePorts: true
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: tcpportone
    port: 4145
    protocol: TCP
    targetPort: 4145
  - name: tcpporttwo
    port: 8199
    protocol: TCP
    targetPort: 8199
  - name: tcpportthree
    port: 8989
    protocol: TCP
    targetPort: 8989
  selector:
    cvmmaster: "true"
  sessionAffinity: None
  type: LoadBalancer
```

2 Run the following command on the master node.

```
kubectl apply -f /YAML/DR/vvr-lb-svc.yaml
```

**3** Run the following command on the master node.

```
kubectl get svc vvr-lb-svc  -n infoscale-vtas
```

Load Balancer IP address is returned as the output.

4 Update the load Balancer IP address and copy the following content into a yaml file and save as `/YAML/DR/loadbalancer.yaml`. Veritas Volume Replicator (VVR) requires both ports - TCP and UDP. Hence, a Load balancer service with mixed protocol support (TCP and UDP) is needed.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/azure-load-balancer-internal:"true"
  service.beta.kubernetes.io/azure-load-balancer-internal-subnet:"worker"

  name: vvr-lb-svc
  namespace: infoscale-vtas
spec:
  loadBalancerIP: <Load Balancer IP address>
  allocateLoadBalancerNodePorts: true
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: tcpportone
    port: 4145
    protocol: UDP
    targetPort: 4145
  - name: tcpporttwo
    port: 8199
    protocol: UDP
    targetPort: 8199
  - name: tcpportthree
    port: 8989
    protocol: UDP
    targetPort: 8989
  selector:
    cvmaster: "true"
  sessionAffinity: None
  type: LoadBalancer
```

5 Run the following command on the master node.

```
kubectl apply -f /YAML/DR/loadbalancer.yaml
```

# 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. |
| 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 |

**Applicable only to Secure Linux DNS.**

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 and remove all unnecessary spaces from the <dns private key> that is displayed.
- Run the following command on the master node
  `cat dns.key | base64`
  Copy and remove all unnecessary spaces from the <dns key> that is displayed.
- Run the following command on the bastion node to generate the keys
  `kubectl get secret -n infoscale-vtas |grep infoscale-sds-dns-secret`
  Use the output in the following command.
- Run the following command on the master node and add the keys

```
kubectl edit secret <output for infoscale-sds-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

**Note:** Ensure that this resource is applied in the same namespace as that of data replication configuration resource.

```
apiVersion: infoscale.veritas.com/v1
kind: DNS
metadata:
  name: <Name of DNS>
  #namespace: <Enter the namespace>
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: "/etc/vx/dns-certs/dns.private"

  # (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:
        #"HostName_1" : "100.200.30.41"
        #"HostName_2" : "100.200.30.42"
        #"HostName_3" : "100.200.30.43"
        #"www" : "HostName_1"
        #"abc" : "HostName_2"
        #"xyz" : "HostName_3"

  # (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**, **namespace**, 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.

To enable a non-kubeadmin user (any user other than infoscale-admin) access custom resources on a secondary cluster after applying Disaster Recovery Plan (DR Plan) custom resource on primary cluster, a role and role-binding must be created . After a DR Plan custom resource is created on the primary cluster, an application namespace is automatically created on the secondary cluster. The role and role-binding on the primary cluster can then be saved and applied on secondary cluster in the same namespace.

---

**Note:** Clusterwide(across namespaces) RBAC permissions are granted to the service account of `infoscale-dr-manager` pod as required. As a Storage Administrator, ensure that a non-privileged user does not have `kubectl exec` permissions on the infoscale-dr-manager pod.

---

1 Edit `/YAML/DR/SampleDisasterRecoveryPlan.yaml` as under to create DR plan for application components in a given namespace.

**Note:** Ensure that this resource is applied in the same namespace as that of data replication configuration resource.

```
apiVersion: infoscale.veritas.com/v1
kind: DisasterRecoveryPlan
metadata:
  name: <Name of the Disaster Recovery Plan>
  #namespace: <Enter the namespace>
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: mysql
    #labels:
    #  - "app=db,env=dev"

  # (optional) Pointer to manage data replication
  #dataReplicationPointer: test-datareplication

  # (optional) Pointer to manage DNS endpoints
```

```
  #dnsPointer: test-dns
```

**Note: 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 11

# Disaster Recovery scenarios

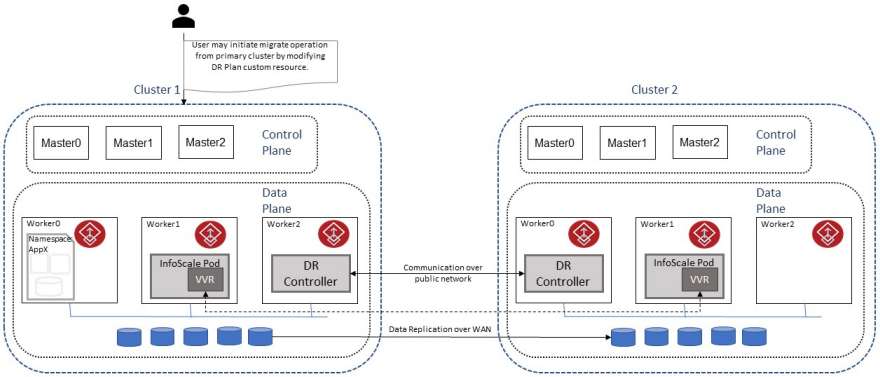
This chapter includes the following topics:



## 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. 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 11-1** Migration initiated. Namespace resides on Cluster 1.

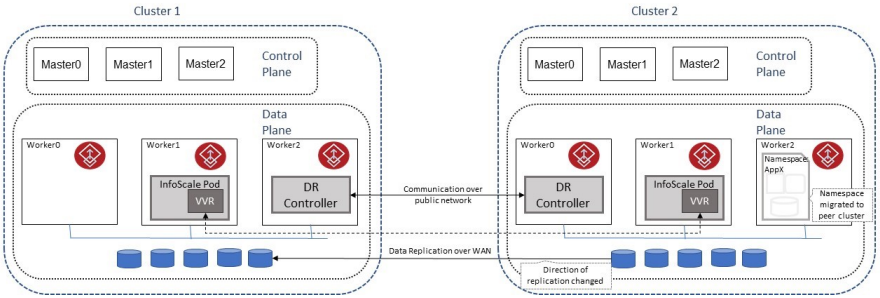



Following entities are updated during migration

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 11-2** Migration complete. Namespace resides on Cluster 2.



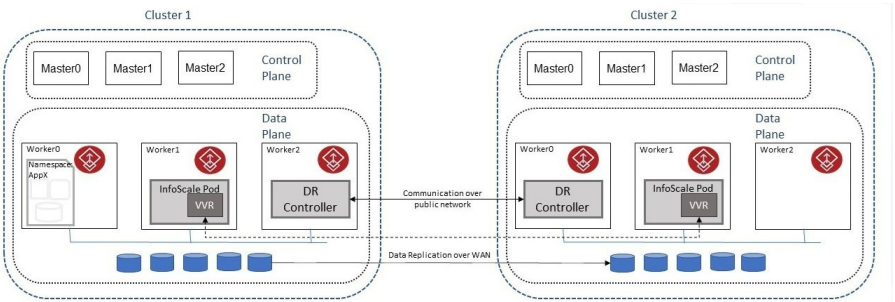

You can check intermediate transient states like BackupStatus, ScheduleStatus, RestoreStatus, and DataReplicationStatus attributes of Disaster Recovery Plan during migration. To check logs if migration is stuck, run `kubectl/oc logs -f --tail=100 deployments.apps/infoscale-dr-manager -n infoscale-vtas`. After migration is complete, these transient states are cleaned and **Status:PrimaryCluster** in Disaster Recovery Plan is updated to the new primary.

# Takeover

You can initiate takeover on a secondary cluster when peer primary cluster is down or disconnected. This can be used in case of disaster when primary cluster abruptly goes down or is disconnected, and application stack is online on the primary cluster. To recover, you can initiate a takeover. Takeover must be initiated from the surviving target cluster(the secondary cluster) where you want to recover application instances. On the secondary cluster, you can run `kubectl or oc edit/patch <NameofDRplan>` and update `Spec:Force` and `Spec:PrimaryCluster` attributes. `Spec:PrimaryCluster` must be changed to current primary cluster details.

`Spec:Force` attribute is a part of the DisasterRecoveryPlan object. Whenever you initiate a takeover, `Spec:Force` must be set to true. After `Spec:Force` is set to true, do not edit DisasterRecoveryPlan object. As peer clusters are not connected, any changes in DisasterRecoveryPlan cannot be synchronized across all peer clusters. As soon as all clusters are connected, stale application instances on outdated primary cluster are removed. `Spec:Force` is also automatically reset to false when all clusters are connected, thus concluding takeover operations.

**Figure 11-3** Before Takeover

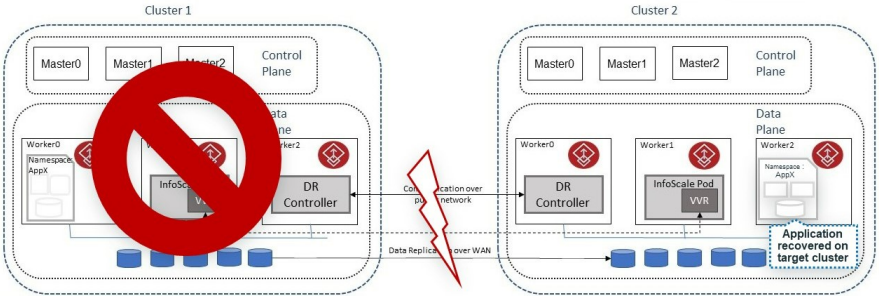



Following entities are updated during a takeover.

1. Application metadata : When a takeover is initiated on the target cluster, latest/recent snapshot of managed application's metadata from the primary cluster is tagged for restoration. This snapshot is used for restoring application stack.
2. Application data : For stateful applications, you must have configured DataReplication CR and updated `DisasterRecoveryPlan:Spec:DataReplicationPointer` accordingly. DataReplication CR manages replication of application data from the primary cluster to the peer clusters(source to target). When a takeover is initiated on the target cluster, the proposed primary cluster assumes 'Primary' role and is allowed read write on Volumes. When outdated primary cluster is connected, it joins as a 'Secondary' role.
3. DNS end points : The DNS custom resource updates and monitors the mapping for
   - The host name to IPaddress (A, AAAA, or PTR record)
   - Alias to hostname or canonical name(CNAME)

When a takeover is initiated, the DNS resource records are updated appropriately

**Figure 11-4** After Takeover




You can check intermediate transient states like `BackupStatus`, `ScheduleStatus`, `RestoreStatus`, and `DataReplicationStatus` attributes of the Disaster Recovery Plan during takeover. To check logs if migration is stuck, run `kubectl/oc logs -f --tail=100 deployments.apps/infoscale-dr-manager -n infoscale-vtas`. After migration is complete, these transient states are cleaned and `Status:PrimaryCluster` in DisasterRecoveryPlan is updated to the new primary.

Chapter 12

# 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/infoscale-dr-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.

# Enabling user access and other pod-related logs in Container environment

OpenShift and Kubernetes clusters have an in-built logging mechanism. You can configure the `/etc/kubernetes/manifests/kube-apiserver.yaml` to include the following types of logs.

- Software upgrades and configuration file changes
- System (Virtual or Physical) boot and halt
- Process launches

- Non-normal process exits
- SELinux policy violations
- System login attempts
- Services starts and stops
- Container starts and exits

You can thus log all events related to InfoScale pods, secrets and config maps.

---

**Note:** After you configure these files, API server must be restarted. Hence, the server experiences a downtime. Ensure that you inform about the downtime to the user community. If the files are not correctly configured, the API server might not restart. The configuration must be performed by a competent Storage Administrator.

---

Add the following code to `/etc/kubernetes/manifests/kube-apiserver.yaml`to log all user login attempts to the InfoScale pods. The user name, time, and whether the attempt is successful or not is logged.

```
apiVersion: audit.k8s.io/v1 # This is required.

kind: Policy

rules

# Log pod/exec requests at RequestResponse level

- level: RequestResponse

namespaces: ["infoscale-vtas"]

resources:

- group: ""

resources: ["pods/exec"]

# Log everything else at Metadata level

- level: Metadata

omitStages:

- "RequestReceived"
```

Similarly, add the following code to log pods creation and deletion, config map changes, and secrets changes.

```
apiVersion: audit.k8s.io/v1

kind: Policy
```

```
omitStages:
- "RequestReceived"
rules:
#Log the Metadata of Pod changes in given namespace
- level: Metadata
resources:
- group: ""
resources: ["pods"]
verbs: ["create", "patch", "update", "delete"]
namespaces: [""] #Fill namespace
#Please use the appropriate namespace where
# the infoscale pods are deployed in the namespace tag
#For eg: namespaces["infoscale-vtas"]
#Log the Request body of configmap changes in given namespace
- level: Request
resources:
- group: ""
resources: ["configmaps"]
namespaces: [""] #Fill namespace
#Log the Request of secrets changes in given namespace
- level: Request
resources:
- group: ""
resources: ["secrets"]
namespaces: [""] #Fill namespace
```

Login attempts to an OpenShift cluster get recorded in `oauth-openshift- pod` logs. The log level must be 'debug'. You can run `oc edit authentications.operator.openshift.io` to change the log level to 'debug'.

On an OpenShift cluster, pods creation and deletion gets logged in `journalctl`. Run `journalctl no-pager` on all the worker nodes for information about pods creation and deletion.

To enable extended logging for all core InfoScale components like VxVM, VxFS, and VCS, set the `EO_COMPLIANCE` to enabled.

---

**Note:** You must enable `EO_COMPLIANCE` for your InfoScale deployment first. As a prerequisite, ensure that DNS is correctly configured or `/etc/hosts` is used to define IP address, full-qualified domain name(FQDN), and host name for the cluster node. Edit sds-operator deployment. Run the following command `oc/kubectl edit deployment -n infoscale-vtas <deployment_name>` . Ensure that you update `EO_COMPLIANCE` here to enabled as under.

```
name: EO_COMPLIANCE

value: enabled
```

After you edit and save the sds-operator deployment, the InfoScale sds operator restarts automatically. You have to manually restart infoscale-sds pods. Ensure that you restart one pod at a time. After a restarted pod is in a 'Ready' state, restart the next pod.

If you want to enable `EO_COMPLIANCE` on an OpenShift cluster by using OLM, run `oc edit subscription infoscale-sds-operator -n infoscale-vtas` and add the following to spec:

```
config:

env:

name: EO_COMPLIANCE

value: enabled
```

---

Chapter 13

# Administering InfoScale on Containers

This chapter includes the following topics:



## Adding Storage to an InfoScale cluster

After the existing storage is consumed, you can present new disks to the InfoScale cluster and increase storage capacity. However, these disks are not automatically discovered or added to the InfoScale cluster.

---

**Note:** You can always check the free size in the disk group after creating Volume Snapshot and Persistent Volume Claim Kubernetes objects. Run the following command.

```
kubectl/oc describe infoscalestoragepool -n infoscale-vtas | grep
Free
```

---

Prerequisites

- InfoScale cluster must be 'Running'

- Additional storage must be presented to the node and detected by the operating system.

Run the following command -

```
kubectl/oc annotate infoscalecluster infoscalecluster-dev <InfoScale
namespace> infoscale.veritas.com/storage-scale-up='enabled'
```

To monitor progress, run the following command.

```
kubectl/oc describe infoscalecluster infoscalecluster-dev <InfoScale
namespace>
```

Review output similar to the following

```
  Type    Reason                        Age
  ----    ------                        ----


  Normal  Storage Scale Up operation:  12m (x2 over 26m)

  Normal  Storage Scale Up operation:  3m34s (x2 over 18m)


 From              Message
 ----              -------
 infoscalecluster  Started
 infoscalecluster  Ended
```

# Managing licenses

While installing InfoScale in an OpenShift or Kubernetes environment, you apply licenses. You select the license edition and optionally enter the IP address of the VIOM server in your environment. With a single license of Enterprise edition on an OpenShift or Kubernetes cluster, you can form multiple InfoScale clusters.

The license server IP address can be updated to `license_cr.yaml` subsequently, if not entered while installing. Edit `<path>/license_cr.yaml` and enter IP address of the VIOM server for `licenseServer`. After updating the yaml, run `kubectl/oc apply -f <path>/license_cr.yaml` to apply the license with the updated IP address of the VIOM server.

After the IP address of the VIOM server is specified and the license is applied, you can check the status. Run `kubectl describe license <path>/license_cr.yaml`. Review the `License Server Status` message with the following table. You can initiate appropriate action on the basis of the message.

**Table 13-1** License status message with the meaning

| Message.. | Means.. |
|---|---|
| Active | Connected successfully. |
| License Server is not registered with Veritas | VIOM Server is not registered with Veritas. |
| Subscription License has expired | Trial period of the license is over. |
| Subscription License not available on License Server | License is not available on the VIOM server to which the connection is successful. |
| Subscription License overused. Kindly check the usage limit. | Exceeded limits of the license edition. |
| Error getting License Server status | VIOM Server is successfully connected, but does not return any message to the licensing operator pod. |
| Connection Error : Not Connected with License Server | Licensing operator pod is unable to connect to the VIOM server on the specified IP address. |

# Monitoring InfoScale

With InfoScale installed in OpenShift or Kubernetes environments, monitoring tools like Prometheus, Grafana, and AlertManager are enabled for monitoring of your InfoScale installation. These tools are the monitoring mechanism that OpenShift and Kubernetes provide. Prometheus collects massive data at regular intervals and presents the data as 'Metrics'. You can use this data for monitoring of cluster components and analyze performance of your InfoScale installation.

**Note:** For a deported diskgroup, metrics is not visible.

On an OpenShift cluster, after you install InfoScale and configure InfoScale clusters, Prometheus monitoring is automatically installed.

**On a Kubernetes cluster, following are the additional steps to install Prometheus**

1. Check if helm is available on the Kubernetes cluster.
2. If it is available, go to step 4.

**3** Run the following steps -

```
url -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
```

```
chmod 700 get_helm.sh
```

```
./get_helm.sh
```

**4** `helm repo add prometheus-community`
`https://prometheus-community.github.io/helm-charts`

```
helm repo update
```

```
helm install <RELEASE_NAME>
prometheus-community/kube-prometheus-stack
```

**On an OpenShift cluster, following is the additional step**

- Create/Edit `cluster-monitoring-config` in the `openshift-monitoring` namespace as under

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    enableUserWorkload: true
```

Run the following commands to verify -

whether pods are created `oc/kubectl get pods -A |grep infoscale`

whether Prometheus monitoring pods are successfully installed.

On an OpenShift cluster - `oc get pods -n {openshift-monitoring} | grep prometheus`

On a Kubernetes cluster - `kubectl get pods -n {namespace} | grep prometheus`

Complete the following steps to enable Prometheus.

1. Run the following command to know the REST server name.

   ```
   oc/kubectl get svc -n {Namespace where InfoScale is installed}
   |grep rest | awk '{print $1}'
   ```

   Note `infoscale-sds-rest-<numerical value>` from the command output.

2. Optionally, to collect metrics with TLS verification; a client certificate is required. Copy the following content into `prom-secret.yaml` and apply the file.

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: infoscale-prom-cert
  namespace: infoscale-vtas
spec:
  commonName: infoscale-prom
  usages:
  - client auth
  duration: 2880h0m0s
  issuerRef:
    kind: ClusterIssuer
    name: infoscale-cert-issuer
  renewBefore: 720h0m0s
  secretName: infoscale-prom-tls
```

3. Copy the following content into `podmonitor.yaml` and apply the file. Ensure that you update `serverName` .

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: infoscale-metrics
  namespace: infoscale-vtas
  labels:
    release: kube-prometheus
spec:
  selector:
    matchLabels:
      cvmmaster: "true"
  podMetricsEndpoints:
  - path: /infoscale/api/2.0/metrics
    port: rest-endpoint
    interval: 2m
    scrapeTimeout: 30s
    scheme: https
    tlsConfig:
      ca:
```

```
    secret:
      key: ca.crt
      name: infoscale-prom-tls
  cert:
    secret:
      key: tls.crt
      name: infoscale-prom-tls
  keySecret:
    key: tls.key
    name: infoscale-prom-tls
  serverName: infoscale-sds-rest-<Cluster ID>
```

# Configuring Alerts for monitoring InfoScale

You can configure various alerts for your InfoScale. To configure, you need to update and apply `alertmanager.yaml`. The following table lists the alerts you can configure

**Table 13-2** Alert expressions with the messages

| **Alert with its expression**<br>**Message** |
|---|
| Volume/PVC size threshold<br>`vxfs_filesystem_used_percentage {job="infoscale-vtas/infoscale-metrics",namespace="infoscale-vtas"}> <Enter the threshold value in percentage>`<br>Volume size exceeds <threshold value> %. |
| Diskgroup-related alert<br>`vxvm_diskgroup_free_bytes {job="infoscale-vtas/infoscale-metrics",namespace="infoscale-vtas"} < <Enter the value in bytes>`<br>The diskgroup has <value> free bytes. |
| NodeNotReady<br>`kube_node_status_condition {job="kube-state-metrics",condition="Ready",status="true"} == 0`<br>Node is not ready. |

**Table 13-2** Alert expressions with the messages *(continued)*

| **Alert with its expression**<br>**Message** |
| --- |
| DiskFailure<br>`vxvm_disk_status {job="infoscale-vtas/infoscale-metrics",namespace="infoscale-vtas"} == 1`<br>Disk failed. |
| DR-related alerts<br>`vxvm_rvg_replication_status {job="infoscale-vtas/infoscale-metrics",namespace="infoscale-vtas"} == 0.0`<br>Replication failed. |

The expressions for the alerts with their messages are entered in the alert manager configuration yaml file.

**Configuring alert manager on an OpenShift cluster**

1 Be ready with the expression and its message for the alert you want to configure. See Table 13-2 on page 256.

2 Copy the following and save the file as alertmanager.yaml.

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: dr-alert
  namespace: <Namespace you want to monitor>
spec:
  groups:
  - name: dralert
    rules:
    - alert: <Name of the Alert>
      annotations:
        message: '<Alert Message>'
      expr: |
        <Alert Expression>
      for: 5m
      labels:
        severity: critical
        prometheus: openshift-monitoring/k8s
```

**Note:** Ensure that you update the alert message and expression.

3 Run the following command on the bastion node.

```
oc apply -f alertmanager.yaml
```

4 You can update and apply `alertmanager.yaml` for other alerts.

You can use the OpenShift monitoring features to configure alerts. See the following OpenShift help topics - Configuring the monitoring stack, Enabling monitoring for user-defined projects, and Enabling alert routing for user-defined projects.

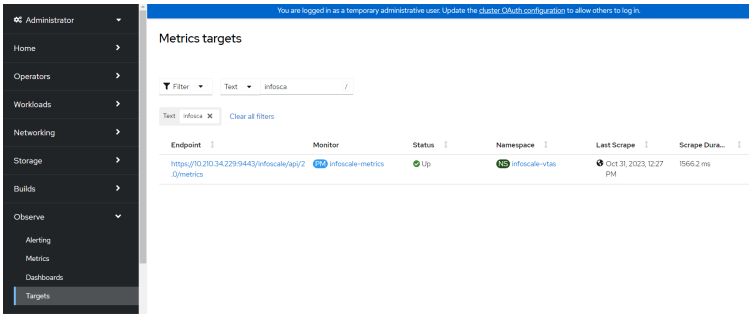
**Using OpenShift console to configure alerts**

1 Access the OpenShift console. Select **Operators > Installed Operators** in the left frame.

2 Click **InfoScale Cluster**. Ensure that the cluster is in a 'Running' state.

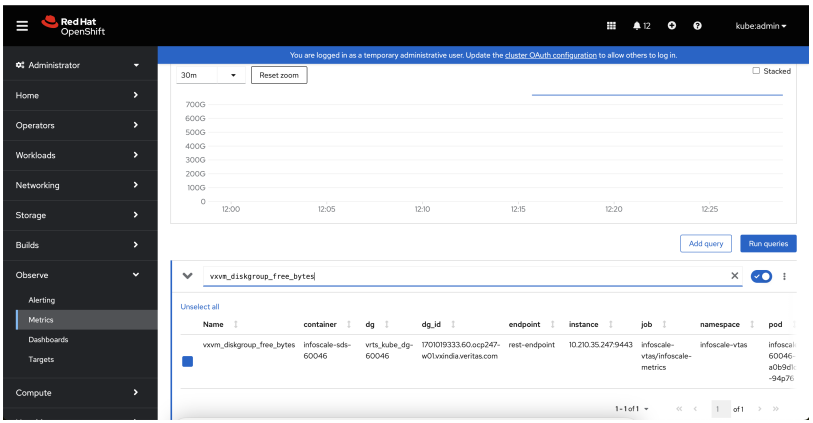
3 Select **Observe > Targets** in the left frame.

4 Enter InfoScale as the search string as under. InfoScale deployment is displayed.




5 Select **Observe > Metrics** in the left frame.

6 In the screen that opens, you can add a new query or run saved queries.




7 Click **Add query** to configure a new query.

8 Assign a name to the query and select the Container.

9 You can then select diskgroup, job, namespace, or pods. Click the parameter and select the value for which you want to configure an alert.

10 Save the query. The saved query now gets listed. You can select a query in the main frame and click **Run queries**.

11 You can also See Table 13-2 on page 256. and enter a query to search.

After you save the alert, the alert is listed in **Observe > Alerts**.

**Using OpenShift console to silence or stop configured alerts**

1. Select **Observe > Alerts** in the left frame.
2. Select the Alert you want to silence.
3. Click the Actions menu (three vertical dots) at the end of the page.

   Alert details are listed.
4. Click **Silence** to stop the alert.

### Configuring alert manager on a Kubernetes cluster

1. Be ready with the expression and its message for the alert you want to configure. See Table 13-2 on page 256.
2. Copy the following and save the file as alertmanager.yaml.

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  annotations:
    meta.helm.sh/release-name: prometheus
    meta.helm.sh/release-namespace: default
  generation: 1
  labels:
    app: kube-prometheus-stack
    app.kubernetes.io/instance: prometheus
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/part-of: kube-prometheus-stack
    app.kubernetes.io/version: 45.31.1
    chart: kube-prometheus-stack-45.31.1
    heritage: Helm
    release: prometheus
  name: prometheus-kube-prometheus-alertmanager-infoscale1.rules
  namespace: infoscale-vtas
spec:
  groups:
    - name: dralert.rules
      rules:
        - alert: <Alert>
          annotations:
            description: <Message>

            summary: <Summary>
          expr: |-
            <Expression>
          labels:
            severity: critical
```

**Note:** Ensure that you update the alert message and expression.

3 Run the following command on the master node.

kubectl apply -f alertmanager.yaml

4 You can update and apply `alertmanager.yaml` for other alerts.

# Draining InfoScale nodes

You might need to drain InfoScale nodes for maintenance reasons (upgrading your software or hardware), poweroff and restart, or any other maintenance reasons. Pods running or scheduled on those InfoScale nodes are taken to other nodes of InfoScale cluster.

In such a case, Veritas does not recommend draining more than one node at a time; due to constraints of infoscale-toolset pods.

# Using InfoScale toolset

After you install InfoScale, InfoScale toolset yaml files are automatically downloaded. You can use these files to know the storage consumption or Volume rekey operations. The files are downloaded at `templates/`.

**To know storage consumption**

1 Edit `templates/storage/infoscale-storage-usage.yaml`. You can specify the namespace and cluster. To know the storage consumption of all namespaces and clusters, do not specify anything in `infoscale-storage-usage.yaml`.

2 Run the following command to apply `infoscale-storage-usage.yaml`.

```
kubectl/oc apply -f
/templates/storage/infoscale-storage-usage.yaml
```

**3** Output similar to the following indicates success.

```
certificate.cert-manager.io/infoscale-tools-cert created
job.batch/infoscale-storage-usage created
```

**4** Run the following command to list storage consumption.

```
kubectl/oc get po -n infoscale-vtas
--sort-by=.metadata.creationTimestamp -o
'jsonpath={.items[-1].metadata.name}' | xargs kubectl/oc logs -n
infoscale-vtas
```

Review output similar to the following

```
NAMESPACE       CLUSTER         DISKGROUP
<namespace>     <cluster name> <diskgroup name>


STATE           VERSION    FREESIZE
imported shared <version> <Available disk space in GB>


 Disk information of Diskgroup-ID: <ID of the diskgroup>
 DISKS      DISKSIZE   MEDIATYPE
 <disk id> <disk size> <disk type>
.
.
.


.
.
.
ALL DISKGROUPS ARE LISTED
```

**To enforce rekey operation on encrypted Volumes**

1. Run the following command to prepare PVC for enforced rekey operation.

   ```
   kubectl/oc annotate pvc <PVC name>
   infoscale.veritas.com/rekey="true" --overwrite
   ```

   The following command indicates that the PVC is ready for enforced rekey operation.

   ```
   persistentvolumeclaim/<PVC name> annotated
   ```

2. Run the following command to enforce rekey.

   ```
   kubectl/oc apply -f /templates/rekey/infoscale-rekey-ondemand.yaml
   ```

**To schedule rekey operation on encrypted Volumes**

1. Edit `templates/rekey/infoscale-rekey-schedule.yaml` to add schedule for rekey operation.
2. Run the following command to apply the schedule.

   ```
   kubectl/oc apply -f /templates/rekey/infoscale-rekey-schedule.yaml
   ```

   Following output indicates a successful application of the schedule.

   ```
   certificate.cert-manager.io/infoscale-tools-cert created
   cronjob.batch/infoscale-rekey-schedule created
   ```

**To perform a diskgroup rekey operation**

- Run the following command -

  ```
  kubectl/oc apply -f /templates/rekey/infoscale-rekey-schedule.yaml
  ```

---

**Note:** After Disaster Recovery (DR) configuration, if rekey is performed on a Volume; the parameter `vxvm.attr.encwvek` is not replicated from the primary to the secondary cluster. Although `vxvm.attr.encwvek` is different on the secondary cluster, Disaster Recovery (DR) operation takes place successfully.

---

Chapter 14

# Migrating applications to InfoScale

This chapter includes the following topics:



## Migrating applications to InfoScale from earlier versions

This section informs you how to migrate applications to the current InfoScale version. Ensure that applications are running on InfoScale 8.0.2xx and PVs, PVCs, and snapshots are available.

---

**Note:** If the Volumes are unencrypted, skip steps (is this applicable here? Below steps were shared with Baloise).

---

**To migrate applications to InfoScale from earlier versions**

1. Note the active replica count of applications which are bound to InfoScale PVCs.
2. Downscale the replica count to 0.
3. Note the cluster ID and cluster name of `infoscalecluster` resource.
4. Delete `infoscalecluster` resource.
5. Delete licenses resource.
6. Uninstall InfoScale SDS operator and InfoScale Licensing operator.
7. If the application or PVCs are present in `infoscale-vtas` namespace, follow below steps. Else skip to step 11.

8. Delete the relevant resources of NFD operator if present in infoscale-vtas (install plans, sub, etc).
9. Delete operator group for InfoScale SDS installation.
10. Delete old install plans for Infoscale from `infoscale-vtas`, if any.
11. Delete `infoscale-vtas` namespace.
12. Delete Custom Resource Definitions (CRD) for the listed resources.
    - nfoscaleclusteres.infoscale.veritas.com
    - infoscalestoragepools.infoscale.veritas.com
13. Run following command to delete the cache.

    ```
    rm -rf
    /root/.kube/cache/discovery/api.<nodename>_6443/infoscale.veritas.com/
    ```

    **Note:** In version 8.0.230 `infoscalecluster` resource is cluster scope and in 8.0.310 it is namespace scope and hence it is necessary to clear the cache while creating `infoscalecluster` resource.

14. If NFD is installed in “All Namespace” mode with OpenShift Container Platform (OCP) version 4.12.31, follow below steps, else skip to step 15
    - Delete NFD instance.
    - Uninstall NFD operator.
    - Delete openshift-nfd namespace.
    - Delete NFD CRD

    **Note:** NFD operator in “All Namespace” mode is not supported with OCP version 4.13.15 or 4.13.19 and NFD operator may go in faulted state after OCP upgrade.

15. Upgrade OCP version to 4.13.15 or 4.1.3.19.
16. If NFD is not installed, then install NFD operator and NFD instance in `openshift-nfd` namespace as recommended by the operator.
17. Install InfoScale SDS operator v8.0.310 from Operator Hub.
18. Create licensing and `infoscalecluster` resource with same cluster name and same cluster ID.

**19** Wait for infoscalecluster to be in RUNNING state.

**20** Run the following command to migrate the data.

```
./pvc_volhandle_edit.sh-c <ClusterID you
notedbeforeuninstalling>-n <infoscaleclustername>
```

Script pvc_volhandle_edit.sh is available in tools.tar.

```
Example:
./pvc_volhandle_edit.sh -c 1000 -n infoscalecluster-dev
PV VolumeHandle format in 8.0.230: vol_d5gmreq183q7jz17672p
PV VolumeHandle format in 8.0.310: clust_1000/vrts_kube_dg-1000/vol_d5gmr
```

ClusterID is used with VolumeHandle in 8.0.310 version. The script modifies PV VolumeHandle format.

**21** Check PVC state which should be in bound state.

**22** Increase the replica count of the respective applications same as the previously deployed replicas.

---

**Note:** Static snapshots do not get migrated.

---

Chapter 15

# Upgrading InfoScale

This chapter includes the following topics:



## Prerequisities

1. The cluster status must be 'Running'.

   On an OpenShift cluster, run `oc get infoscalecluster <InfoScale namespace>`.

   On a Kubernetes cluster, run `kubectl get infoscalecluster <InfoScale namespace>`.

   Review output as under

   ```
   NAME                 NAMESPACE             VERSION           STATE   AGE
   infoscalecluster-dev <InfoScale namespace> <current version> Running 24h
   ```

2. The cluster state must be 'Healthy'.

   On an OpenShift cluster, run `oc get infoscaleclusters.infoscale.veritas.com infoscalecluster-dev <InfoScale namespace> -ojsonpath='{.status.clusterState}{"\n"}'` `Healthy`.

   On a Kubernetes cluster, run `kubectl get infoscaleclusters.infoscale.veritas.com infoscalecluster-dev <InfoScale namespace> -ojsonpath='{.status.clusterState}{"\n"}'` `Healthy`.

System returns as under

```
Healthy
```

Guidelines to follow if DR is configured

- DR controller must be upgraded after InfoScale upgrade is complete.
- Ensure that any migration or takeover operation is not in progress during upgrade.
- Veritas recommends upgrading the secondary cluster first followed by the primary cluster.

# On an OpenShift cluster

There are two methods to upgrade on an OpenShift cluster:

- Using the CLI
- Using OLM

**On an OpenShift cluster by using CLI**

1. Download `YAML_8.0.310.tar.gz` from the Veritas Download Center and unzip and untar the file.
2. Run the following command on the bastion node `oc apply -f /YAML/<path>/lico.yaml`.

   **Note:** Here and in the next command, use the <path> where the yaml files are saved.

3. Run the following command on the bastion node `oc apply -f /YAML/<path>/iso.yaml`.
4. Run the following command on the bastion node `oc patch infoscaleclusters.infoscale.veritas.com infoscalecluster-dev --type merge -p '{"spec": {"version": "<upgrade version>"}}'`.
5. Wait as it might take time to upgrade.

6 Run the following command on the bastion node `oc get infoscalecluster`

The State is 'Upgrading' as under.

```
NAME                  NAMESPACE      VERSION           STATE      AGE
infoscalecluster-dev infoscale-vtas <Upgrade version> Upgrading 25h
```

7 Upgrading InfoScale takes time. After a successful upgrade, the cluster state changes to 'Running'. You can run `oc get infoscalecluster` to verify.

8 If you have configured DR, run `oc apply -f /YAML/DR/dro_deployment.yaml`.

Ensure that you upgrade the primary and the secondary sites.

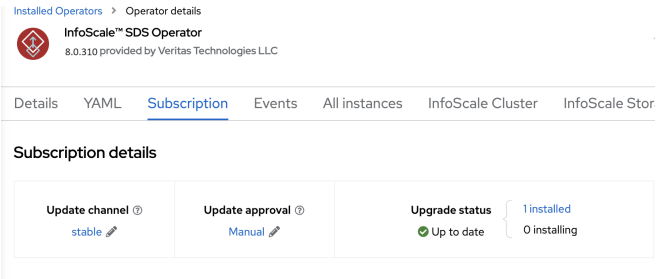
**On an OpenShift cluster by using OLM**

1 Connect to the OpenShift and access the catalog menu.

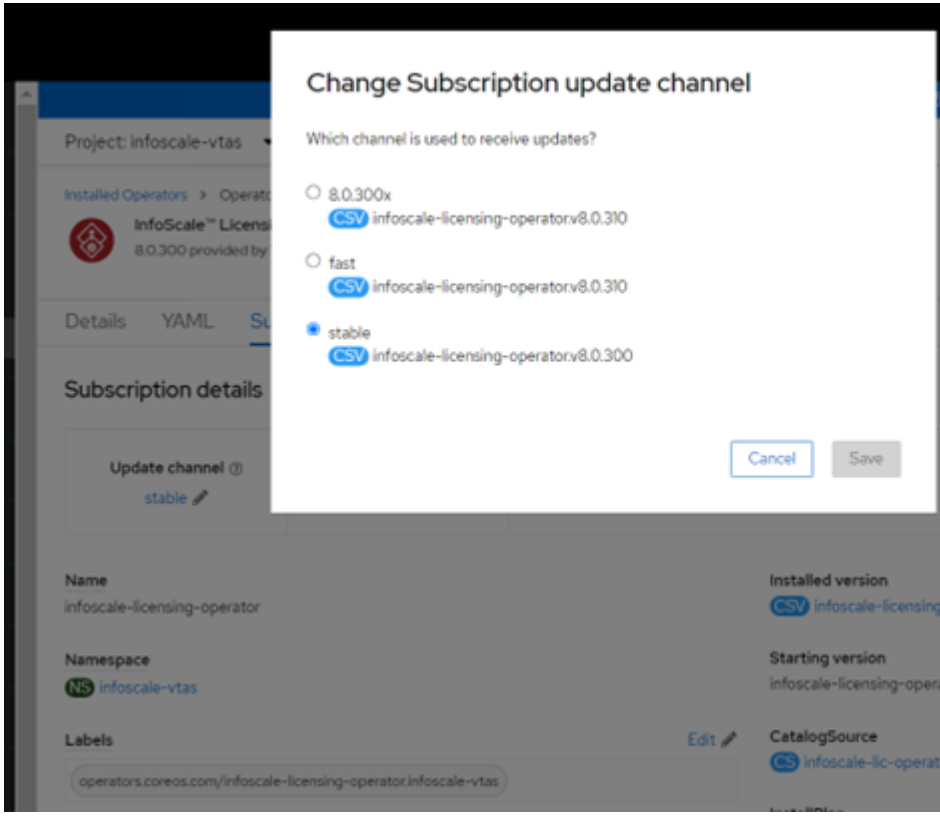
2 In the left frame, Select **Operators > Installed Operators**.

3 All InfoScale operators are listed here.

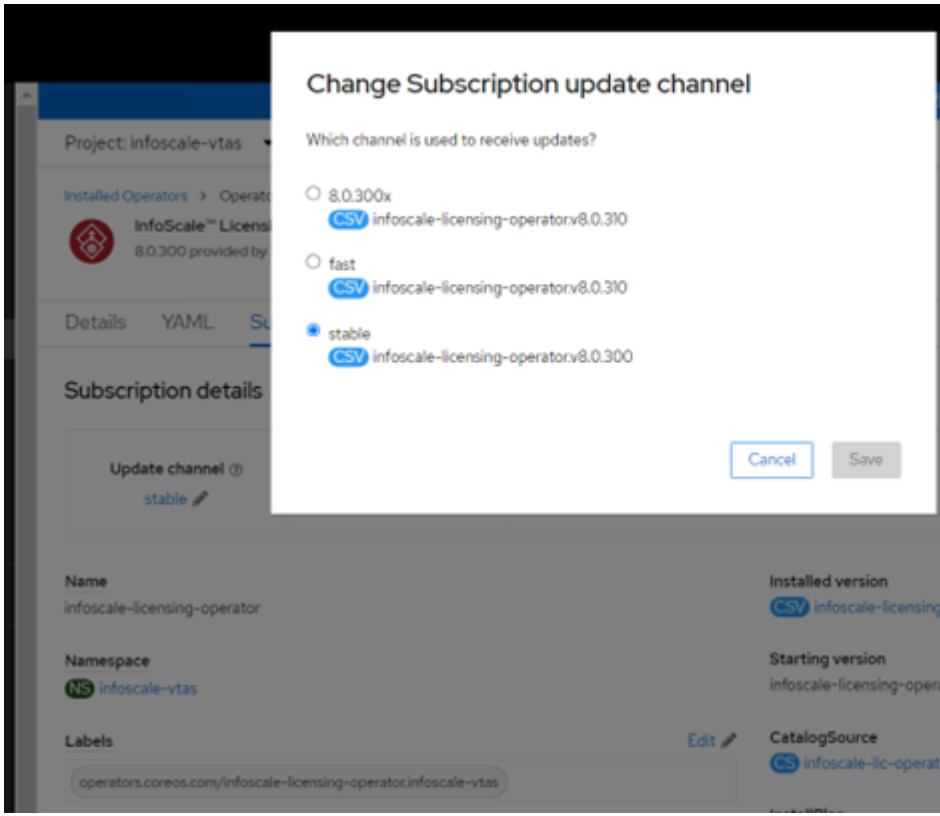
4 Click **InfoScale SDS Operator** and the **Subscription** tab. The following screen opens.



5 Click **Update channel**. Select **8.0.300x** as under and click **Save**.

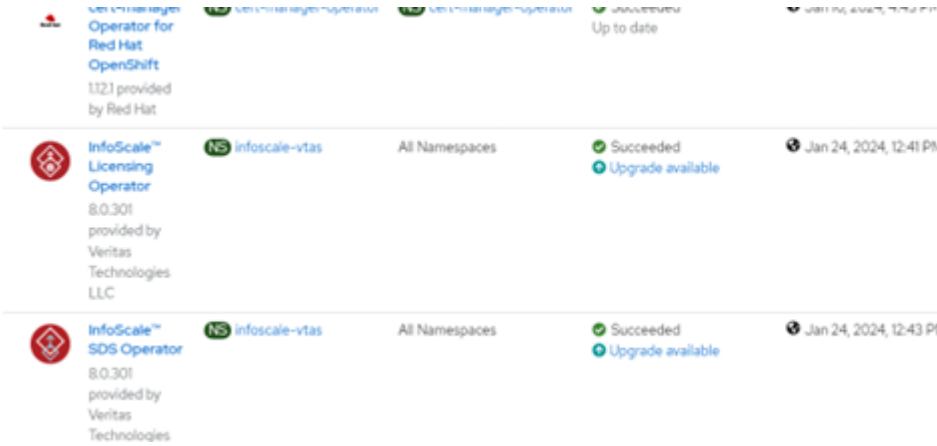


6 Follow identical steps for the **InfoScale Licensing Operator** and select **8.0.300x** as under.




7 Select **Operators > Installed Operators** again

8 For the InfoScale operators, **Upgrade Available** is reported in **Status** as under.




9 To upgrade InfoScale, click InfoScale or InfoScale Licensing Operator.

10 In **Subscription**, updates that need approval are displayed next to **Upgrade Status**.

11 Click the Operator that needs approval, followed by **Preview Install Plan**.

12 Review the details list. Click **Approve** if you want to proceed with the upgrade.

13 Wait till it is upgraded and listed in **Operators > Installed Operators**.

14 Run the following command on the bastion node `oc patch infoscaleclusters.infoscale.veritas.com infoscalecluster-dev --type merge -p '{"spec": {"version": "<upgrade version>"}}'`.

15 Wait as it might take time to upgrade.

16 Run the following command on the bastion node `oc get infoscalecluster`

The State is 'Upgrading' as under.

```
NAME                  NAMESPACE      VERSION           STATE      AGE
infoscalecluster-dev infoscale-vtas <Upgrade version> Upgrading 25h
```

17 Upgrading InfoScale takes time. After a successful upgrade, the cluster state changes to 'Running'. You can run `oc get infoscalecluster` to verify.

18 If DR is configured, perform the following steps.

- Click **Infoscale DR Manager Operator**. In Subscription, updates that need approval are displayed next to the Upgrade Status.

- Click the operator that needs approval, followed by **Preview Install Plan**.
- Review the details list. Click **Approve** if you want to proceed with the upgrade.
- Wait till it is upgraded and listed in **Operators > Installed Operators**.

Ensure that you upgrade the primary and the secondary site.

---

**Note:** After upgrading InfoScale, ensure that you upgrade OCP to any of the InfoScale-supported versions. To know the supported OCP platforms, see Supported platforms. Refer to OpenShift documentation to know how to upgrade OpenShift.

---

Chapter 16

# Troubleshooting

This chapter includes the following topics:



## Adding a sort data collector utility

A data collector utility is available on the Veritas support portal. To know more about the utility, see https://sort.veritas.com/data_collectors/overview.

**Complete the following steps**

1. Download https://sort.veritas.com/data/vos/collectors/sort_linux_x64.sh to collect evidences of the pods in `infoscale-vtas`.
2. Run the script on the master node of the Kubernetes cluster or on the bastion node of the OpenShift cluster.

## Collecting logs by using SORT Data Collector

For troubleshooting purposes, logs are most necessary, By using SORT Data Collector(`sortdc`), you can collect node-level and cluster-level logs and InfoScale-specific logs at the node and cluster level as well.

### Collecting logs on OpenShift

1. Run `sortdc` on the bastion node. With a single instance running on a bastion node, all node-level and cluster-level logs are collected.
2. Run the following command on the bastion node to export cluster's `admin.conf` to `KUBECONFIG`.

   ```
   export KUBECONFIG=/etc/kubernetes/admin.conf
   ```

3. Run the `sortdc` tool. OpenShift node-level and cluster-level logs are saved at the location you specify while running the tool.

### Collecting logs on Kubernetes

1. Run `sortdc` on the master node. With a single instance running on a master node, all node-level and cluster-level logs are collected.
2. Run the following command on the master node to export cluster's `admin.conf` to `KUBECONFIG`.

   ```
   export KUBECONFIG=/etc/kubernetes/admin.conf
   ```

3. Run the `sortdc` tool. Kubernetes node-level and cluster-level logs are saved at the location you specify while running the tool.

Run the following commands on every InfoScale SDS pod in the InfoScale cluster for InfoScale-specific node-level and cluster-level logs.

1. Login to the InfoScale SDS pod.
   - On an OpenShift cluster, run the following command on the bastion node.

     ```
     oc exec -it <InfoScale SDS PodName> -n <InfoScale namespace>
     -- /bin/bash
     ```

   - On a Kubernetes cluster, run the following command on the master node.

     ```
     kubectl exec -it <InfoScaleDriverContainerPodName> -n
     <InfoScale namespace> -- /bin/bash
     ```

2. Run the following commands to create and enable the swap file.

   ```
   dd if=/dev/zero of=/var/log/swapfile bs=1024 count=1000000;
   mkswap /var/log/swapfile;
   swapon /var/log/swapfile;
   ```

3 Run `/opt/VRTSspt/VRTSexplorer/VRTSexplorer`.

Specify log location as `/var/log/` to collect explorer files at the host-mounted path.

4 Run the following commands to disable and remove the swapfile.

```
swapoff /var/log/swapfile;
rm -f /var/log/swapfile;
```

**Note:** Ensure that you run these commands on all InfoScale SDS pods.

# Approving certificate signing requests (csr) for OpenShift

Sometimes entry into the shell of an OpenShift container returns an error message like

```
Error from server: error dialing backend: x509: certificate is valid...
```

This error message is due to pending certificate signing requests (csr) approval or expired certificate signing requests (csr). Complete the following steps to approve certificate signing requests (csr). You can also refer to OpenShift documentation to know about csr.

1 Ensure that entry into the shell returns an error on all pods of the node.

2 Run the following command to check for pending approvals.

```
oc get csr
```

3 Run the following command to approve pending csrs.

```
oc get csr -o go-template='{{range .items}}{{if not
.status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs oc adm
certificate approve
```

4 Try entry into the shell to ensure that it is unblocked.

# Cert Renewal related

1. If Prometheus is configured and if after the external CA certificate is renewed, the pod monitor fails to come up. Delete the prometheus cert secret followed

by deletion of all pods in the monitored namespace. Prometheus pods then come up.

2. If the following error message is logged -

```
x509: certificate has expired or is not yet valid: current time
<timestamp1> is after <timestamp2>
```

   - If you are using an External CA certificate and the certificate has expired. See 'Renewing with an External CA certificate' and renew the certificate.
   - If you are using a self-signed CA, delete the infoscale-CA secret and wait for 10 minutes. Delete the secrets of all the client certificates.

3. If one of the following error messages is logged -

```
error:veritas:tls parseCertsCertificate
                expired :/etc/vx/<component name>/certs/tls.crt
```

   or

```
 tls : bad certificate
```

   - If you are using an External CA certificate and the certificate has expired. See 'Renewing with an External CA certificate' and renew the certificate. If the certificate has not expired, delete all secrets of the client certificates,
   - If you are using a self-signed CA and the certificate has expired, delete the infoscale-CA secret and wait for 10 minutes. Delete the secrets of all the client certificates. If the certificate has not expired, delete the secrets of all the client certificates.

# 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 16-1** Issue description and workaround

| Description | Workaround |
|---|---|
| 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. |
| 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 16-1** 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 -<br>■ Place a resource cap on less important OCP system pods like Prometheus (OCP Monitoring service). See OpenShift documentation.<br>■ 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.<br>■ 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) | Login to the InfoScale SDS pod and 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 SDS pod. |

**Table 16-1** 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 -<br>■ Use CSI space-optimized snapshot functionality for read-intensive applications.<br>■ Use full-instant snapshot or CSI clone functionality for write-intensive or read-write-update applications.<br>■ 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 -<br>■ 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`<br>■ 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 16-1** 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)<br>Delete datarep operation fails if secondary site is not available. (VIKE-3885) | Complete the following steps to clean up<br>**1** Check DR controller logs to check which cleanup part is failing.<br>**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`.<br>**3** Login to InfoScale cluster pod infoscale-sds-* on all clusters<br>**4** Complete the following steps for Veritas Volume Replicator (VVR) objects cleanup<br>■ Stop replication for relevant RVG<br>■ Delete secondary<br>■ Delete primary<br>■ Delete corresponding SRL volume<br>**5** Complete the following steps for Veritas Cluster Server (VCS) objects cleanup<br>■ Change cluster operation to RW<br>■ Offline VIPgroup and RVGShared service groups corresponding to Datareplication CR (service group names are shown in CR status)<br>■ Delete resources available in these service. groups<br>■ Delete service groups' dependencies if any<br>■ Delete VIPgroup and RVGShared service groups<br>■ Change cluster operation to RO<br>See Veritas Volume Replicator and Veritas Cluster Server documentation for details. |

**Table 16-1** 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<br>**1** Run the command to know the RVG name `kubectl get datareplications.infoscale.veritas.com <application Data Replication name>`<br>**2** Login to one of the Infoscale SDS pods.<br>**3** Run the following command `vxprint -g vrts_kube_dg-<Infoscale cluster ID> <RVG name >`<br>**4** Review the output. If tutil is set to 'CONVERTING' , run the following command `vxedit -g vrts_kube_dg-<Infoscale cluster ID> -f set tutil0="" <RVG name>`<br>tutil is cleared and DR migration completes |
| Kube-fencing is not functional when REST service and InfoScale operator pod is not running. (4054545) | None |
| In VVR environment, `vradmin migrate` might fail with the following error message<br>`VxVM VVR vxrvg ERROR V-5-1-1617 giving up: utility fields must be cleared by executing: vxedit -f set tutil0="" <rvg>`<br>(4057713) | Run the following command - `vxedit -f set tutil0="" <rvg>` to clear `tutil` on the RVG and retry the 'vradmin migrate' operation. |

**Table 16-1** Issue description and workaround *(continued)*

| Description | Workaround |
|---|---|
| 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.<br><br>`vxvm:vxconfigd[238047]: V-5-1-11092 cleanup_client: (Disk in use by another cluster) 223`<br>`esxd05vm06 kernel: VxVM vxio V-5-0-164 Failed to join cluster infoscale_22670, aborting :`<br><br>(4057178) | Reset the cluster ID on the disks of joiner node, in order to allow the node to join.<br><br>**1** Check cluster ID on existing/already-joined nodes in cluster: Run<br><br>`vxdisk list node000_vmdk0_9 \| grep -i cluster`<br><br>Cluster ID is returned in the output.<br><br>**2** If the node join fails, verify if cluster ID is different on the joiner node using same command: Run<br><br>`vxdisk list node001_vmdk0_5 \| grep -i cluster`<br><br>A different Cluster ID is returned in the output.<br><br>**3** Change the cluster ID to match it with existing nodes in cluster.<br><br>`/etc/vx/diag.d/vxprivutil set /dev/vx/dmp/node001_vmdk0_5 hostid=<Cluster ID returned in the first command>`<br><br>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. |
| LLT tools is unable to detect duplicate InfoScale cluster id on OpenShift.(4057800) | Re-deploy InfoScale CR to avoid duplicate cluster id match. |
| RLINK detach is observed while replication autosync is in progress between the primary and secondary clusters. (VIKE-1290) | Run `kubectl describe datarep datareplicationName` to check datareplication status. If `ReplicationStatus` is stopped, Edit `datareplication` and set attributes `force: true` and `replicationState: stop` in remoteClusterDetails. Again, edit `datareplication` and set attributes `force: false` and `replicationState: start` in remoteClusterDetails. |

**Table 16-1** Issue description and workaround *(continued)*

| Description | Workaround |
|---|---|
| On a Kubernetes cluster, tcp traffic reaches the node but does not get forwarded to the pod. (VIKE-1108) | Disable tx and rx offload on all nodes for calico tunnel device. |
| Migration or takeover fails if the namespace being backed up pre-exists on the target cluster with different SCC (Security context constraints)- related annotations like uid-range, supplemental-groups, and seLinuxOptions.(VIKE-1277) | None |
| Takeover operation fails if it is initiated when 'Disaster Recovery plan' is configured before 'Data replication configuration' is complete on the primary cluster. (VIKE-1294) | Wait for the following error in Disaster Recovery Plan CR status to go away before attempting takeover - **metadataSyncStatus: Metadata backup transfer failed for all peer clusters**<br>or<br>Wait for data replication CR status to be consistent up-to-date before applying Disaster Recovery plan CR as mentioned in 'Configuring Data Replication'. |
| On OCP 4.9, `nfd-worker` pods are not getting created. (VIKE-909) | Run `oc edit scc nfd-worker` on the bastion node, change<br>`users:`<br>`- system:serviceaccount::nfd-worker`<br>to<br>`users:`<br>`- system:serviceaccount:`<br>`  <Name of the namespace>:nfd-worker`<br>Wait for some time for the nfd-worker pods to be in the 'Running' State. |
| If multiple Data Replication plans for Disaster Recovery are created in parallel, the initial VVR synchronization is slow due to the locking that needs to be acquired in VVR for each of the Data Replication plan. (VIKE-1505) | To create multiple Data Replication plans, create the subsequent plans after the current Data Replication plan's initial synchronization is complete and status is consistent-up-to-date. |

**Table 16-1** Issue description and workaround *(continued)*

| Description | Workaround |
|---|---|
| When UEFI secure boot is enabled, only the signed kernel modules that are authenticated by using a key on the system key ring can be successfully loaded. Hence InfoScale kernel modules might fail to load with UEFI secure boot. (VIKE-1578) | Disable UEFI secure boot. See the relevant documentation for details. |
| In the following conditions, addition of disks to a disk group in an already configured InfoScale cluster fails with the error message<br><br>`VxVM vxdg ERROR V-5-1-559`<br>`Disk node000_vmdk0_1: Name is already used`<br><br>**1** Existing disk group is imported during cluster creation<br>**2** Additional storage is presented to the cluster<br>**3** Sequence of the node name in the CR is changed.<br><br>(VIKE-2598) | None |
| If thin-provisioned LUNs are configured, a mismatch in the total size and free size of the Disk Group might be observed. (VIKE-2497) | None |
| After DR takeover is completed and the old primary is up, VVR failback synchronization of the data is initiated from the new primary site to the new secondary site. This data synchronization might be slow or the system might be unresponsive resulting in the VVR failback to be incomplete. (VIKE-2634) | **1** Pause all data replications on the primary site by setting the `replicationState` to `pause` in the data replication custom resources. Perform a sequential VVR failback synchronization operation then by setting the `replicationState` to `resume` on only one of the data replication custom resources, that is the active data replication.<br>**2** After the VVR failback synchronization is complete on the only active data replication, resume VVR failback synchronization on the next paused data replication. Complete VVR failback synchronization on all the configured data replications. |

**Table 16-1** Issue description and workaround *(continued)*

| Description | Workaround |
|---|---|
| Sometimes after a cluster fault, InfoScale nodes join the cluster back, infoscale-sds pods go in running state but the InfoScale disk group remains in a deported state.(VIKE-2666) | Login to any of the InfoScale-sds pods.<br>■ Run `/opt/VRTSvcs/bin/hagrp -clear DISK_GROUP` to clear the fault on `DISK_GROUP` and `/opt/VRTSvcs/bin/hagrp -online DISK_GROUP -any` to get `DISK_GROUP` online.<br>■ If the service group fails to be online, run `vxdg -s import <diskgroup_name>` to manually import the disk group.<br>■ If the disk group import fails with an error message `quorum lost`, reboot all InfoScale nodes in the cluster. |
| After InfoScale is deployed, cluster phase does not get updated and state is 'Degraded', even when it is Healthy. (VIKE-2586) | Restart InfoScale operator by deleting the pod. |
| If stale Volumes are present on the system, undeploying InfoScale is unresponsive with SDS pods in a terminating state (VIKE-2635) | Reboot all worker nodes. You can reboot sequentially, if applications in the cluster might be impacted due to a simultaneous reboot of all nodes. |
| Sometimes a primary cluster fault event might occur after a PVC resize operation on the primary cluster, followed by a takeover operation carried out from secondary cluster. In that case, when the old primary joins back the cluster membership, some Volumes on the old primary are in a `NEEDSYNC` state. The `NEEDSYNC` state is indicated by running `vxprint` on the old primary infoscale-sds pod. (VIKE-2917) | Run `vxrecover -s` from an infoscale-sds pod. The state changes to `Active`. |
| During upgrade, sometimes InfoScale sds pod remains in a terminating state. (VIKE-3003) | Reboot the node where the pod is stuck in a terminating state. |
| Error observed in Persistent Volume Claim addition in an application namespace if one of the worker nodes is down with the following error message in data replication status.<br><br>`VxVM vxassist ERROR V-5-1-2127 Plex srl_4x0g2eewpu3b6m30ay71-02 should be enabled`<br><br>(VIKE-4193) | Data replication status returns to normal state after the worker node restarts. |

**Table 16-1** Issue description and workaround *(continued)*

| Description | Workaround |
|---|---|
| When all the worker nodes are down, `kubectl / oc get` shows the incorrect InfoScale cluster status.(VIKE-4149) | Whenever any worker pod is 'Ready', InfoScale status is updated. |
| When an old primary cluster rejoins after a takeover operation, Volume number mismatch data replication config error is observed as under -<br>Output of `kubectl/oc describe datarep <datarep name>`<br>`Replication Config Errors:`<br>`<IP address>: volume-number mismatch, Primary-Primary configuration`<br>(VIKE-4151) | Run the following steps -<br>**1** On the new primary, stop replication by editing the datarep CR.<br>**2** On old primary(secondary), run the following commands on any infoscale-sds-* pod<br>■ `vxvol -g <dg_name> -f dis <vol_name>`<br>■ `vxrvg -g <dg_name> makesecondary <rvg_name>`<br>■ `vxrlink -g <dg_name> -f start <rlink_name>` (where rlink_name can be found by - `vxprint \| grep rlk_NewPrimaryVIP_rvgName`)<br>**3** On the new primary, start replication by editing the datarep CR. |
| Data replication is stuck in an inconsistent state while logging to DCM (VIKE-4369) | Run `kubectl/oc get datarep -o wide`. If the output indicates `inconsistent \| logging to DCM (needs dcm resynchronization)` for a data replication plan, edit the corresponding datarep cr to stop replication. Wait for the replication to stop. Restart replication by editing the same datarep cr. The repstatus subsequently is `replicating (connected).` |
| While adding or removing disks to Storage Arrays without AVID support, applications might not come up after a node reboot. (VIKE-4139) | Contact Veritas support. |
| After a node reboot, the toolset pod does not come online if the Volumes are in a 'Sync' state. (VIKE-4434) | Run the following command to change the state of the Volume from 'Sync' to 'Active '.<br>`vxvol -g <disk group name> -f resync <name of the Volume>` |
| In a DR setup after a node reboot, if the InfoScale SDS pod on that node is not in a 'Ready' state and the cluster CR status is 'Degraded' on either primary or secondary, applications running on that node might face issues in performing IO operations. (VIKE-4428) | Reboot that node again and check the IO operations. If IO operations issues persist, reboot all nodes in the cluster together. |

**Table 16-1** Issue description and workaround *(continued)*

| Description | Workaround |
|---|---|
| PVC creation on a cluster with cri-o container engine fails with the following error message -<br><br>`VxVM vxassist ERROR V-5-1-1401 Your license does not include support for mirroring. Cannot create volume with requested attributes`<br><br>(VIKE-4367) | Run `crictl images` on the worker nodes of the cluster to verify if InfoScale images are present. If NOT present, see https://access.redhat.com/solutions/5350721 without the 'node reboot' section for the resolution. |
| If cluster nodes reboot or get powered off in a cascaded manner, application pods might not failover to a running node. (VIKE-4375) | This issue might be due to Storage connectivity. If Storage is connected and this issue is observed -<br>**1** Identify the node where application pod is scheduled after failover.<br>**2** Login to InfoScale SDS pod running on that node and run `vxprint -uH \| grep LDISABLED` to identify the Volume which has a LDISABLED attribute.<br>**3** Run `vxvol -g <DG_NAME> -f enable <VOL_NAME>` to remove that attribute. |
| Deleting application pod operation may get stuck, when the file system provisioned through Persistent Volume Claims (PVC) consumed by the application pod is in a disabled state. (VIKE-4514) | Login to the node where the application pod is scheduled and manually unmount all the mount points that has file system disabled. |
| While upgrading InfoScale Stack from 8.0.300 to 8.0.310, the SDS driver pod is stuck or does not boot for longer duration than expected. (VIKE-4523) | Check cluster start logs using the below command:<br>`oc -n infoscale-vtas exec -it infoscale-sds-62453-a8d04dbf0095f7bc-wmgtx -- tail -f /var/VRTSvss/vss.log`<br>If you see error messages or abnormal number of re-attempts for starting cluster, then reboot the corresponding worker node where SDS pods resides. |
| While upgrading OCP/orchestrator platform, the process may not progress and there is a mismatch between the version of worker node operating system and that of the selector. (VIKE-4522) | Ensure that the NFD label sync on the nodes is complete. Restart the InfoScale Operator pod. |

**Table 16-1** Issue description and workaround *(continued)*

| Description | Workaround |
|---|---|
| Application pod is stuck in `init/ContainerCreating` state.<br>Check error in CSI node log of the node where the application pod is scheduled using the command:<br>`oc logs <csi-node-pod> -n infoscale-vtas -c infoscale-csi.`<br>Following error is displayed.<br>"Run: Error occurred while calling method" method=/csi.v1.Node/NodePublishVolume error="rpc error: code = Internal desc = Error occurred node staging is not done"" (VIKE 4532) | Delete the application pod. When the new pod is scheduled, it calls NodeStage and NodePublish. |

# Limitations

Following are the functional limitations due to external factors.

**Table 16-2** Limitation description and workaround

| Description | Workaround |
|---|---|
| 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. |
| Remove node is not supported for InfoScale on OpenShift or Kubernetes. (4044647) | To remove node, contact Veritas Technical support. |

**Table 16-2** Limitation description and workaround *(continued)*

| Description | Workaround |
|---|---|
| InfoScale modules fail to load on Kubernetes if SELinux is enabled on the worker nodes. (VIKE-1028) | Perform the following steps<br>**1** Run the following commands on the worker nodes to enable the tunables.<br>`setsebool -P container_manage_cgroup on`<br>`setsebool -P domain_kernel_load_modules 1`<br>**2** Set `selinux-enabled: true` in `/etc/docker/daemon.json`.<br>**3** Run the following command on the worker nodes to reboot.<br>`systemctl daemon-reload && systemctl restart docker` |
| Replacing disks or replacing coordination points without unconfiguring fencing is not supported. (VIKE-2563) | Reconfigure InfoScale cluster with the new coordination points. |
| During migration of an application or just before takeover of an application, if PVCs and PVs are added to its namespace; the PVCs and PVs might not successfully restore on the secondary site. | None |