



VAST CSI Driver 2.6

Administrator's Guide

22 April 2025

Always check support.vastdata.com for the latest

© 2025 VAST Data, Inc. All Rights Reserved. The content of this documentation is highly sensitive and confidential.

Contents

About VAST CSI Driver	4
Overview of VAST CSI Driver	4
A Deeper Dive into VAST CSI Driver	4
Automatic Creation of a View per Volume	5
Leveraging Trash Folder Access to Handle Deletions	5
Support of NFSv3 and NFSv4	6
Support of VAST Cluster Multi-Tenancy	6
Upgrading VAST CSI Driver	7
Steps to Upgrade VAST CSI Driver	7
Upgrading a Helm-Based Deployment of VAST CSI Driver	7
Create Automatic Views for Existing PVCs	8
Migrate PVs from VAST CSI Driver 2.0.x to 2.x	9
Remove VAST CSI Driver 2.1.x or 2.0.x	10
Remove VAST CSI Driver 1.0.x	11
Deploying VAST CSI Driver	12
VAST CSI Driver Requirements	12
Steps to Deploy VAST CSI Driver	12
Install CRDs for VAST Snapshots	13
Create a Kubernetes Namespace for VAST CSI Driver	13
Configure VAST Cluster for VAST CSI Driver	14
Create a Kubernetes Secret with VMS User Credentials for VAST CSI Driver	16
Add the Helm Repository for VAST CSI Driver	18
Create a Helm Chart Configuration File for VAST CSI Driver	18
Install the VAST CSI Driver Helm Chart	22
Verify VAST CSI Driver Deployment	23
Deploying VAST CSI Driver on OpenShift with VAST CSI Operator	25
Configuring SSL Encryption for VAST CSI Driver	27
Securing VAST CSI Driver in Kubernetes Multi-Tenant Environment	29
Provisioning Volumes with VAST CSI Driver	31
Using Storage Classes	31
Provisioning Static Volumes	36
Provisioning Ephemeral Volumes	37
Provisioning Volumes on Multiple VAST Clusters	38
Managing NFS Mount Options	39
Setting Up DNS-Based Virtual IP Resolution	39

Protecting Block Volumes with Snapshots and Clones	41
Overview of VAST Snapshots and Clones	41
Create a Volume Snapshot	41
Restore a Volume from a Snapshot	42
Clone a Volume or a Snapshot	43
Troubleshooting VAST CSI Driver	46
Steps to Troubleshoot VAST CSI Driver	46
Resolving PVC Timeouts.	48

About VAST CSI Driver

Overview of VAST CSI Driver

VAST CSI Driver allows container orchestration frameworks such as Kubernetes to dynamically provision storage on a VAST cluster using the [Container Storage Interface \(CSI\)](#). VAST provides scalable, reliable, and fast persistent storage that can be accessed remotely by any Kubernetes application container.

With dynamic provisioning, you do not need to create container-specific persistent volumes. Instead, you deploy a dynamic provisioner and a YAML class definition that specifies the dynamic provisioner and provisioning options. When an application makes a claim for storage, the Kubernetes framework employs the dynamic provisioner to dynamically create a persistent volume. Later, when a containerized application (via its YAML) references this claim, Kubernetes provides the storage to the container.

With VAST CSI Driver, you can:

- [Provision](#) volumes based on multiple Kubernetes storage classes
- [Provision](#) static volumes
- [Provision](#) ephemeral volumes
- [Protect](#) the volumes being provisioned with VAST snapshots and clones

VAST CSI Driver provides support for the following:

- [NFSv3 and NFSv4](#) access protocols
- [SSL encryption](#) of the connection to the VAST cluster

VAST CSI Driver capabilities include:

- [Automatic creation](#) of a VAST cluster view for each volume being provisioned
- [Leveraging](#) the VAST cluster's Trash Folder Access functionality to handle deletions
- [Support](#) for multiple VAST cluster tenants
- [Support](#) for multiple VAST clusters
- [DNS-based resolution](#) of virtual IPs

VAST CSI Driver is deployed using a [Helm](#) chart. Check the deployment [requirements](#) and follow these [steps](#) to implement dynamic provisioning with VAST CSI Driver. Note that the driver can also be deployed in OpenShift environments using [VAST CSI Operator](#).

A Deeper Dive into VAST CSI Driver

VAST CSI deployment has two components:

- A controller pod that is responsible for creating and deleting volumes.
- A node pod responsible for creating the mount on the node (host) it is running on. The node pod contains VAST CSI Driver that invokes mount and umount commands.

A node gets all necessary information about existing virtual IP pools and views from the controller. It does not communicate with the VAST Management Service (VMS) directly. The only exception to this is for handling Ephemeral Volumes, where

the node pod handles the entire volume lifecycle.

The operation sequence is as follows:

1. An application makes a Persistent Volume Claim (PVC), which indirectly references the storage class which, in turn, references the dynamic provisioner for VAST Cluster via CSI.
2. The CSI receives the request to create a persistent volume. Based on the NFS configuration information provided via the storage class, it creates a VAST Cluster view and a directory on the VAST cluster (named for the persistent volume) and informs Kubernetes that the volume has been created.
3. Kubernetes allocates the logical persistent volume which references the VAST Cluster directory. At this point, you can see the directory by listing the view on the VAST cluster.
4. Kubernetes begins the application launch sequence and recognizes that the application referenced a PVC.
5. Kubernetes asks the VAST CSI controller to publish the volume to the node where the application is to run.
6. The VAST CSI controller gets the API request and allocates one of the VAST Cluster's virtual IPs to the node.

Note that each volume will get a single virtual IP on each node. Multiple volumes on the same node or the same volume on different nodes will likely get different virtual IPs for proper load spreading across volumes and nodes.

7. Kubernetes asks VAST CSI Driver on the node to mount the volume, making it available for the application.
8. Kubernetes causes the application within the pod to start.
9. The application can create a file in what appears to be a local directory within its container, but actually points to a directory on the VAST cluster (the directory being the persistent volume).

Automatic Creation of a View per Volume

VAST CSI Driver creates a VAST Cluster view for each volume or bucket being provisioned. Such a view is automatically deleted when its volume or bucket is deleted.

The views created by VAST CSI Driver can be monitored using VAST Cluster's Web UI or CLI.

To control access permissions for storage exposed through these views, VAST CSI Driver uses a view policy that is attached to the view. You specify the view policy for each storage class.



Note

When [provisioning static volumes](#), automatic creation of views is disabled by default, but can be enabled for a PV by setting a flag in the PV definition.

Leveraging Trash Folder Access to Handle Deletions

VAST CSI Driver gains a substantial performance boost during volume deletions through the use of *Trash Folder Access*, a feature of VAST Cluster.



Note

VAST Cluster 4.7.0-SP6 or later is required for the VAST drivers to be able to use the Trash Folder Access feature.

When deleting a volume that resides on VAST Cluster 4.7.0-SP6 and later where the Trash Folder Access feature is enabled, the delete request is processed using the VAST REST API's `/folders/delete_folder/` endpoint. This means that the processing takes place locally on the VAST cluster.

VAST CSI Driver can delete either through Trash Folder Access, or by using a backward-compatibility mode which requires you to [dedicate](#) a virtual IP pool and a view policy for deletions through the `deletionVipPool` and `deletionViewPolicy` parameters in the VAST CSI Driver [Helm chart configuration file](#).

When deleting volumes that reside on a VAST Cluster version earlier than 4.7.0-SP6, or when Trash Folder Access is disabled for any reason, you still need to specify `deletionVipPool` and `deletionViewPolicy`.



Note

For deletions to succeed, the tenant in the deletion view policy (`deletionViewPolicy`) must match the tenant of the view policy (`viewPolicy`) specified for the storage class.

Support of NFSv3 and NFSv4

VAST NFS CSI Driver supports NFSv4 and NFSv3. Enabling NFSv4 access by VAST NFS CSI Driver is done the same way as for other NFSv4 clients.

To mount volumes with NFSv4, specify `mountOptions: "nfsvers=4"` for the storage class.

Support of VAST Cluster Multi-Tenancy

VAST CSI Driver can operate on multiple VAST Cluster tenants.

To determine from which tenant to provision storage for a claim, VAST Cluster looks at the view policy that is attached to the view that exposes the volume or bucket.

To be able to operate in a multi-tenant environment, VAST CSI Driver requires VAST Cluster 4.7.0-SP6 or later with the *Trash Folder Access* feature enabled.



Note

If a deletion view policy (`deletionViewPolicy`) is specified in the VAST CSI Driver's [Helm chart configuration file](#), the tenant in the deletion view policy must match the tenant of the view policy (`viewPolicy`) specified in the storage class. Otherwise, deletions would fail.

Upgrading VAST CSI Driver

Steps to Upgrade VAST CSI Driver

Choose your upgrade path:

- To upgrade from 2.2.1 or later to 2.4.0 or later:
 - Follow the [procedure](#) for upgrading a Helm-based deployment.
- To upgrade from 2.2.0 to 2.4.0 or later:
 1. Follow the [procedure](#) for upgrading a Helm-based deployment.
 2. (Optional) Run a [script](#) to create [View-per-Volume](#) views for existing PVCs.
- To upgrade from 2.1.x to 2.4.0 or later:
 1. [Remove](#) version 2.1.x.
 2. [Deploy](#) version 2.4.0 or later.
 3. (Optional) Run a [script](#) to create [View-per-Volume](#) views for existing PVCs.
- To upgrade from 2.0.x to 2.4.0 or later:
 1. [Remove](#) version 2.0.x.
 2. [Deploy](#) version 2.4.0 or later.
 3. [Migrate](#) existing Kubernetes persistent volumes.
 4. (Optional) Run a [script](#) to create [View-per-Volume](#) views for existing PVCs.
- To upgrade from 1.0.x to 2.4.0 or later:
 1. [Remove](#) version 1.0.x.
 2. [Deploy](#) version 2.4.0 or later.
 3. [Migrate](#) existing Kubernetes persistent volumes.
 4. (Optional) Run a [script](#) to create [View-per-Volume](#) views for existing PVCs.

Containers referencing VAST CSI Driver will run with the new version of VAST CSI Driver the next time they are created.

Upgrading a Helm-Based Deployment of VAST CSI Driver

Use the following procedure if both old and new release of VAST CSI Driver are deployed using a Helm chart.

1. Refresh Helm repository information to see which VAST CSI Driver versions are available for deployment:

```
helm repo update
```

2. Update the VAST CSI Driver's [Helm chart configuration file](#) (`values.yaml`) to specify the target release of VAST CSI Driver:

```
image:
```

```
csiVastPlugin:
  repository: <repo>
  tag: <new release>
  imagePullPolicy: IfNotPresent
```

Where:

- <repo> is the name of the VAST CSI Driver [Helm repository](#).
- <new release> is the VAST CSI Driver release to be upgraded to.

For example:

```
image:
  csiVastPlugin:
    repository: vastdataorg/csi
    tag: v2.6.0
    imagePullPolicy: IfNotPresent
```

3. Run Helm upgrade:

```
helm upgrade <old release> <repo>/vastcsi -f <filename>.yaml
```

Where:

- <old release> is the VAST CSI Driver release to be upgraded from.
- <repo> is the name of the VAST CSI Driver [Helm repository](#).
- vastcsi is the name of the VAST CSI Driver Helm chart.
- <filename>.yaml is the [Helm chart configuration file](#).

For example:

```
helm upgrade csi-driver vastcsi/vastcsi -f values.yaml
```

4. Verify the status of the new VAST CSI Driver chart release:

```
helm ls
```

Create Automatic Views for Existing PVCs

Run the view creation script `create-views.py` as part of migrating your environment from VAST CSI Driver 2.2.0 to VAST CSI Driver 2.2.1 and later.



Note

Running this script is an optional step during the upgrade.

Starting with version 2.2, VAST CSI Driver [automatically creates](#) a VAST Cluster view for each volume it provisions. A view policy is assigned to control access to automatically created views.

The view creation script creates a view for each PVC provisioned with pre-2.2.1 versions of the VAST CSI Driver and assigns a user-supplied view policy to such views. All quotas which were previously under the same root export view, get their own views pointing to the same location, with no impact on the data being stored. This ensures that the PVCs remain accessible in

case the root export view is manually deleted.

To run the view creation script:

1. Download the script from the VAST GitHub space:

```
wget https://raw.githubusercontent.com/vast-data/vast-csi/v2.2/scripts/create-views.py
```

2. Run the script on a host where `kubectl` is already installed and configured with the target Kubernetes cluster.

Use the following syntax:

```
create-views.py --view-policy <view policy>
```

Where `<view policy>` specifies an existing VAST Cluster view policy to be assigned to the automatically created views.

For example:

```
>> python3 create-views.py --view-policy mypolicy
```

3. Verify that the script completes successfully.

Migrate PVs from VAST CSI Driver 2.0.x to 2.x

You have to migrate Kubernetes persistent volumes (PVs) that were created with VAST CSI Driver 2.0.x and earlier to be able to use the PVs with VAST CSI Driver 2.x.

Each PV created with VAST CSI Driver versions 2.1.0 and later has the following parameters in its `volumeAttributes` configuration entry:

- `root_export`. This is the path within the VAST cluster where the volume directory is created.
- `vip_pool_name`. This is the name of the VAST virtual IP pool to be used when mounting the volume.
- `mount_options`. This parameter lists custom NFS mount options to be used when mounting the volume. If no custom mount options are needed, the parameter is specified with an empty value (`"`).

PVs created with earlier versions of VAST CSI Driver do not have these attributes and thus cannot be mounted with VAST CSI Driver 2.1.0 or later for Kubernetes pods.

To add the missing attributes to PVs created with pre-2.1.0 versions of VAST CSI Driver, download and run the PV migration script provided by VAST.

Running the PV Migration Script

Run the PV migration script `migrate-pv.py` as part of the procedure to migrate your environment from VAST CSI Driver 2.0.x or earlier to VAST CSI Driver 2.1.1.

The PV migration script adds `root_export` and `vip_pool_name` attributes to Kubernetes persistent volumes (PVs) created with VAST CSI Driver 2.0.x or earlier. After running this script, VAST CSI Driver 2.1.1 is able to mount the updated PVs for Kubernetes pods.

VAST recommends running the PV migration script before you remove the old version of VAST CSI Driver to let the script automatically determine and set proper `root_export`, `vip_pool_name` and `mount_options` values.

However, you can choose to run the script after having removed the old version of VAST CSI Driver and installed VAST CSI Driver 2.1.1. In this case, you have to specify the `root_export`, `vip_pool_name` and `mount_options` values manually.

Complete these steps:

1. Download the PV migration script from the VAST GitHub space:

```
wget https://raw.githubusercontent.com/vast-data/vast-csi/v2.1/scripts/migrate-pv.py
```

2. Run the script on a host where `kubectl` is already installed and configured with the target Kubernetes cluster.

- If the old version of VAST CSI Driver has not been removed yet, use the following syntax:

```
migrate-pv.py [--vast-csi-namespace='<namespace>']
```

Specify the `--vast-csi-namespace` parameter if the namespace of the old version of VAST CSI Driver is not `vast-csi`. For example:

```
>> python3 ./migrate-pv.py --vast-csi-namespace='kube-system'
```

- If you have removed the old version of VAST CSI Driver and installed VAST CSI Driver 2.1.1, use the following syntax:

```
migrate-pv.py --force --root_export=<path> --vip_pool_name=<vip_pool_name>
```

Where:

- `--force` is a required parameter that indicates that the script is run after you have removed the old version of VAST CSI Driver and installed VAST CSI Driver 2.1.1.
- `--root_export` identifies the storage path that was provided when the old version of VAST CSI Driver was deployed. This is the value you supplied at the “NFS Export Path” prompt.
- `--vip_pool_name` identifies the virtual IP pool to be used for the PVs. Specify a virtual IP pool that has been defined in VAST Cluster. It does not have to be the same as in the original deployment.
- `--mount_options` specifies custom NFS mount options. Enter a comma-separated list of options. Specify "" (empty string) if none are needed.

For example:

```
>> python3 ./migrate-pv.py --force --root_export=/data/k8s --vip_pool_name=vippool-1 --mount_options=''
```

3. Verify that the script output is similar to the following:

```
info: PV <pv_name> updated.  
info: Done.
```



Note

If an error was found in the user-supplied parameters, rerun the script with correct values specified.

Remove VAST CSI Driver 2.1.x or 2.0.x

After removal of VAST CSI Driver 2.1.x or 2.0.x, the existing containers will continue to run with the volumes provisioned by the removed VAST CSI Driver.

1. Remove all VAST storage classes:

```
kubectl delete sc vastdata-filesystem
```

2. Remove the VAST CSI driver:

```
kubectl delete csidriver csi.vastdata.com
```

3. Remove the VAST namespace:

Caution

If your PVCs are in the same namespace as the driver, take preliminary action to prevent deletion of the PVCs as a result of the namespace deletion.

```
kubectl delete all --all -n vast-csi
```

4. Remove the VAST ServiceAccounts:

```
kubectl delete sa csi-vast-controller-sa -n vast-csi  
kubectl delete sa csi-vast-node-sa -n vast-csi
```

5. Remove the VAST Secret:

```
kubectl delete secret csi-vast-mgmt -n vast-csi
```

6. Remove the VAST ClusterRole:

```
kubectl delete ClusterRole csi-vast-attacher-role  
kubectl delete ClusterRole csi-vast-provisioner-role
```

7. Remove the VAST ClusterRoleBinding:

```
kubectl delete ClusterRoleBinding csi-resizer-role  
kubectl delete ClusterRoleBinding csi-vast-attacher-binding  
kubectl delete ClusterRoleBinding csi-vast-provisioner-binding
```

8. Remove the VAST VolumeSnapshotClass:

```
kubectl delete VolumeSnapshotClass vastdata-snapshot
```

Remove VAST CSI Driver 1.0.x

To remove VAST CSI Driver 1.0.x, you'll need the YAML configuration file that was used to deploy VAST CSI Driver. You can generate the file using the 1.0.x container.

Run the following command:

Caution

If your PVCs are in the same namespace as the driver, take preliminary action to prevent deletion of the PVCs as a result of the namespace deletion.

```
kubectl delete -f vast-csi-deployment.yaml
```

Deploying VAST CSI Driver

VAST CSI Driver Requirements

Ensure that your environment meets the following requirements:

- A Kubernetes cluster is up and running.
- A VAST cluster is up and running.
- All nodes of the Kubernetes cluster are networked with the VAST cluster and can mount VAST cluster volumes via NFS.
- At least one node can communicate with the VAST Cluster management virtual IP.
- A host is available with a [Helm](#) client installed, preferably in the VMS network.

Supported Versions

VAST CSI Driver	Kubernetes	VAST Cluster	Helm
2.6.x	1.22 - 1.31.6	4.1 or later To work with multiple VAST Cluster tenants : 4.7.0-SP10 or later To handle deletions via <i>Trash Folder Access</i> : 4.7.0-SP6 or later To clone volumes or snapshots : 4.6.0 or later	3.x.x

Required Permissions

VAST CSI Driver requires root (sysadmin) privileges so that it is able to create mounts on the host machine.

Steps to Deploy VAST CSI Driver

Before you begin, ensure that your environment meets the [requirements](#).



Tip

If you are going to deploy VAST CSI Driver in an OpenShift environment, consider using VAST CSI Operator to facilitate the deployment. See [here](#) for more information and guidelines.

VAST CSI Driver is deployed using [Helm](#) charts. A *Helm chart* is an installation template that can be reused to install multiple instances of the software being deployed. Each instance is referred to as a *release*. Helm charts are available from *Helm repositories*.

Steps to deploy VAST CSI Driver include:

1. [Install](#) Custom Resource Definitions for snapshots.



Note

This step is required if the driver's Helm chart configuration file (`values.yaml`) has `secretName` and `Endpoint` specified as global options (on top of `values.yaml`). This step is optional only when `secretName` and `secretNamespace` are specified within one or more storage classes (or under `StorageClassDefaults`).

2. (Optional) [Create](#) a Kubernetes namespace for VAST CSI Driver.



Note

This step is required if you are going to deploy VAST CSI Driver in a Kubernetes namespace other than `default`. Otherwise, skip to step 4.

3. [Configure](#) the VAST cluster.
4. [Create](#) a Kubernetes secret with VMS user credentials for VAST CSI Driver.
5. [Add](#) the Helm repository that contains the VAST CSI Driver chart.
6. [Create](#) a Helm chart configuration file for VAST CSI Driver.
7. [Install](#) the Helm chart for VAST CSI Driver.
8. (Optional) [Verify](#) the deployment by launching a test application.

Install CRDs for VAST Snapshots

Run the following commands to install the CRDs for snapshots:

This step is required if the driver's Helm chart configuration file (`values.yaml`) has `secretName` and `Endpoint` specified as global options (on top of `values.yaml`). This step is optional only when `secretName` and `secretNamespace` are specified within one or more storage classes (or under `StorageClassDefaults`).

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v6.0.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v6.0.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v6.0.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v6.0.1/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v6.0.1/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

Create a Kubernetes Namespace for VAST CSI Driver

By default, VAST CSI Driver is deployed to the `default` namespace on the Kubernetes cluster.

If you want to use a different Kubernetes namespace for VAST CSI Driver, create it prior to deployment by running the following command:

```
kubect1 create ns <namespace_name>
```

Configure VAST Cluster for VAST CSI Driver

Complete the following steps to make your VAST cluster ready for integration with Kubernetes:

- [Set up](#) virtual IP pools to be used by VAST CSI Driver.
- [Configure](#) a view policy to be used for views created by VAST CSI Driver.
- [Set up](#) a VMS user to be used by VAST CSI Driver.
- (Optional) [Upload](#) your CA-signed SSL certificate to the VAST cluster.
- (Optional): [Configure](#) a QoS policy to be associated with views created by VAST CSI Driver.



Note

If you are going to use VAST CSI Driver with a VAST cluster where the Trash Folder Access feature is disabled or not used for any reason, you also need to complete [additional configuration steps](#).

Set up Virtual IP Pools

VAST CSI Driver distributes the load among virtual IPs in one or more VAST virtual IP pools.

You specify a virtual IP pool in the [storage class](#) definition. The virtual IP pool specified for a storage class is used to process read and write operations requested by the application that is using that particular class.

When using VAST Cluster 4.6.0 or later, ensure that the virtual IP pool set for a storage class belongs to the same VAST Cluster tenant as the [view policy](#) specified for that class.

In the VAST CSI Driver chart configuration file, the virtual IP pool can be specified by its name (`vipPool`) or FQDN (`vipPoolFQDN`). If you are going to use the FQDN, ensure that the VAST cluster has DNS configured, and the virtual IP pool has *Virtual IP Pool Domain Name* defined in its settings.

To view and manage virtual IP pools in VAST Web UI, log in and choose Network Access -> Virtual IP Pools in the main navigation menu. For more information about VAST Cluster virtual IP pools, see *VAST Cluster Administrator's Guide*.

Configure View Policies

VAST CSI Driver can automatically create a VAST Cluster view for each storage claim being provisioned. These views are controlled using VAST Cluster view policies.

You specify a view policy to be assigned to the automatically created views in the [storage class](#) definition. The view policy specified for a storage class is used when processing read and write operations requested by the application that is using that particular storage class.

When using VAST Cluster 4.6.0 or later, ensure that the view policy set for a storage class belongs to the same VAST Cluster tenant as the [virtual IP pool](#) specified for that class.

To view and manage existing view policies in VAST Web UI, log in and choose Element Store -> View Policies. For more information about VAST Cluster view policies, see *VAST Cluster Administrator's Guide*.

Set Up a VMS User

Set up a VMS user for VAST CSI Driver to communicate with the VAST Management Service (VMS) via VAST REST API.

You'll need to supply the VMS user credentials in a Kubernetes secret that is specified when creating the VAST CSI Driver's Helm chart configuration file.

VAST CSI Driver can use the default `admin` user that is created by VAST during the initial install, or you can create a new VMS user. The user does not need to have full VMS permissions. The required permissions are Create, View, Edit and Delete permissions in the Logical realm.

Permissions are granted to a VMS user by assigning a role. To view and manage roles in VAST Web UI, log in and choose Administrators -> Roles in the main navigation menu. For more information about managing roles, see *VAST Cluster Administrator's Guide*.

To view and manage VMS users in VAST Web UI, log in and choose Administrators -> Managers in the main navigation menu. For more information about managing VMS Manager users, see *VAST Cluster Administrator's Guide*.

Upload a CA-Signed SSL Certificate to VAST Cluster

If you want to use a Certificate Authority-signed SSL certificate to secure the connection to the VAST cluster, follow the SSL certificate upload procedure in the *VAST Cluster Administrator's Guide* to upload your SSL certificate to the VAST cluster.

For more information about configuring SSL encryption for VAST CSI Driver, see [Configuring SSL Encryption for VAST CSI Driver](#).

Configure a QoS Policy

You can optionally set up a Quality of Service (QoS) policy to be associated with the views that VAST CSI Driver creates. A QoS policy is specified per Kubernetes storage or bucket class configured for the VAST driver.



Notice

This capability requires VAST Cluster 4.6 or later.

To view and manage QoS policies via VAST Web UI, log in and choose Element Store -> QoS Policies. For more information about VAST Cluster QoS policies, see *VAST Cluster Administrator's Guide*.

Extra Configuration Steps for Legacy CSI Local-Mount Deletions

If you are going to use VAST CSI Driver with a VAST cluster where the Trash Folder Access feature is disabled or not used for any reason, you need to specify a deletion view policy and a deletion virtual IP pool. The deletion view policy and pool are used solely for deletions and must enable VAST CSI Driver to effectively handle deletions across all storage classes defined.

The deletion view policy and pool can be the same as those used for the volume views, or they can be different.

Ensure that the deletion view policy and pool meet the following requirements:

- For all versions of VAST Cluster:

The deletion view policy has the same security flavor as the view policy set for the storage class.

- For VAST Cluster 4.6.0 up to 4.7.0-SP3:

The deletion view policy and pool belong to the same VAST Cluster tenant as the view policy and pool specified for the storage class.

- For VAST Cluster earlier than 4.7.0-SP6:

The deletion view policy does not have the *Root Squash* setting enabled.

Create a Kubernetes Secret with VMS User Credentials for VAST CSI Driver

Create a Kubernetes [secret](#) to keep [VMS user](#) credentials that VAST CSI Driver uses to communicate with the VAST cluster. You need to supply the name of the secret when [creating](#) the driver's Helm chart configuration file.

The Kubernetes secret can also specify the VAST cluster on which you want to provision volumes for a particular storage class or snapshot class.

- [Create a Kubernetes Secret to Provision Storage on a Single VAST Cluster](#)
- [Create a Kubernetes Secret to Provision Storage on Multiple VAST Clusters](#)

Create a Kubernetes Secret to Provision Storage on a Single VAST Cluster

Use this procedure to create a single Kubernetes secret that contains VMS user credentials to connect to a single VAST cluster. The Kubernetes secret will be used for all storage classes and snapshot classes defined in the VAST CSI Driver Helm chart configuration file. The VAST cluster to connect is specified on the `endpoint` parameter in the configuration file.

To create a Kubernetes secret that will be used to provision storage on a single VAST cluster:

1. Create a YAML file with the following content. Note that the VMS user's username and password must be Base64-encoded.

```
apiVersion: v1
kind: Secret
metadata:
  name: <secret name>
type: Opaque
data:
  username: <VMS user's username>
  password: <VMS user's password>
```

2. Apply the YAML file:

```
kubectl apply -f <path to the YAML file>
```

Alternatively, you can create a secret with the following command:

```
kubectl create secret generic <secret name> \
  --from-literal=username='<VMS user's username>' \
```



```
--from-literal=password='<VMS user's password>'
```



Note

If you are creating the secret in a Kubernetes namespace that is different from the namespace used to install the VAST CSI Driver Helm chart, specify the secret's namespace on the command: `-n <secret's namespace>`.

Create a Kubernetes Secret to Provision Storage on Multiple VAST Clusters

Use this procedure to create one or more Kubernetes secrets that can be specified individually per storage class or snapshot class. Each of these Kubernetes secrets specifies the VAST cluster to connect to and the VMS user credentials, enabling you to [provision](#) volumes on multiple VAST clusters or using multiple VMS users on the same VAST cluster.



Tip

This type of secret can be used not only when there are multiple VAST clusters, but also anytime when the `secretName` and `secretNamespace` parameters in the Helm chart configuration file (`values.yaml`) are specified within a storage class or under `StorageClassDefaults`.

To create a Kubernetes secret that will be used to provision storage on multiple VAST clusters:

1. Create a YAML file with the following content:

```
apiVersion: v1
kind: Secret
metadata:
  name: <secret name>
type: Opaque
data:
  endpoint: <VAST cluster hostname>
  username: <VMS user's username>
  password: <VMS user's password>
  sslCert: <path to SSL certificate>
```

Where:

- (Required) `endpoint` sets the hostname FQDN or IP address of the VMS URL of the VAST cluster where you want to provision volumes.
- (Required) `username` is the username of the VMS user to connect to the VAST cluster. The username must be Base64-encoded.
- (Required) `password` is the password for the VMS user connecting to the VAST cluster. The password must be Base64-encoded.
- (Optional) `sslCert` is only required if you are using [SSL encryption](#) with a self-signed SSL certificate. It specifies the path to the SSL certificate.

2. Apply the YAML file:

```
kubectl apply -f <path to the YAML file>
```



Note

If you are creating the secret in a namespace that is different from the namespace used to install the VAST CSI Driver Helm chart, specify the secret's namespace on the command: `-n <secret's namespace>`.

Alternatively, you can create a secret with the following command:

```
kubectl create secret generic <secret name> \
  --from-literal=endpoint='vms.example.com' \
  --from-literal=username='<VMS user's username>' \
  --from-literal=password='<VMS user's password>' \
  --from-file=ssl_cert='<path to SSL certificate>' \
  -n <secret's namespace>
```

For example:

```
kubectl create secret generic vast-mgmt \
  --from-literal=endpoint='vms.example.com' \
  --from-literal=username='user1' \
  --from-literal=password='xxxxxxxx' \
  --from-file=ssl_cert='VastCerts/RootCA.crt' \
  -n secret_namespace
```

Add the Helm Repository for VAST CSI Driver

Add the Helm repository that contains the VAST CSI Driver Helm chart to the list of available repositories:

1. Add the Helm repository for VAST CSI Driver:

```
helm repo add <repo> https://vast-data.github.io/vast-csi
```

Specify any suitable name for `repo`. This name will be used to refer to the VAST CSI Driver repository when running Helm commands. For example: *vastcsi*

2. Verify that the repository has been added:

```
helm repo list
```

The output is similar to the following:

```
NAME      URL
vastcsi   https://vast-data.github.io/vast-csi
```

Create a Helm Chart Configuration File for VAST CSI Driver

In Helm-based deployments, VAST CSI Driver's Helm chart configuration file lets you override default installation settings provided in the chart with parameters that are specific to your environment.

The configuration file is a YAML file typically named `values.yaml`, although you can use any arbitrary name for it.

- [Create a Configuration File](#)

- [Verify the Configuration File](#)

Create a Configuration File

Create a YAML file as follows (see also the example below):



Note

For a detailed reference for parameters and values, refer to <https://github.com/vast-data/vast-csi/blob/v2.6/charts/vastcsi/values.yaml>.

```
secretName: "<secret>"
endpoint: "<endpoint>"
verifySsl: true|false
sslCertsSecretName: "<SSL secret>"
deletionVipPool: "<deletion virtual IP pool>"
deletionViewPolicy: "<deletion view policy>"

StorageClassDefaults:
  <option 1>
  <option 2>
  ...
  <option n>

storageClasses:
  <storage class name 1>:
    <option 1>
    <option 2>
    ...
    <option n>
  <storage class name 2>:
    <option 1>
    <option 2>
    ...
    <option n>
  ...
  <storage class name n>:
    <option 1>
    <option 2>
    ...
    <option n>
```

In the YAML file:

1. Set global or storage class-specific session options:

- Global session options are specified in the beginning of the configuration file. They apply to all storage classes:
 - `secretName: "<secret>"`: Specify the Kubernetes secret with VMS user credentials to be used by VAST CSI Driver. The secret must include the VMS user's username and password. For more information, see [Create a Kubernetes Secret with VMS User Credentials for VAST CSI Driver](#).

The `secretName` global session option is required unless the secret name is provided, together with `secretNamespace`, under `StorageClassDefaults` or under a specific storage class. If you supply the secret name under a specific storage class, you must specify the endpoint within the secret.

- `endpoint: "<endpoint>"` (required): Enter the VAST Cluster management hostname.

- `verifySsl: true|false` (optional): Specify `true` to enable [SSL encryption](#) for the connection to the VAST cluster. If set to `false` or not specified, SSL encryption is disabled.



Tip

When enabling SSL encryption, either upload a CA-signed SSL certificate to the VAST cluster, or [supply](#) a self-signed SSL certificate to the driver. The latter can be done either via the `sslCertsSecretName` option, or using `--set-file sslCert` on the Helm chart installation command.

- `sslCertsSecretName: "<SSL secret>"` (optional): Specify the Kubernetes secret that contains the self-signed SSL certificate to be used to secure communications between VAST CSI Driver and the VAST cluster. For more information, see [Configuring SSL Encryption for VAST CSI Driver](#).
- Storage class-specific session options are specified under a certain storage class (in [step 3](#)). They apply to that particular storage class only, taking precedence over global session options.
2. If you are going to use a VAST cluster that has its Trash Folder Access feature disabled or not used, set options to [handle deletions](#). When running against a VAST Cluster version prior to 4.7.0-SP6, these options are required.
 - `deletionVipPool: "<deletion virtual IP pool>"`: Specify the name of the VAST Cluster's virtual IP pool to be used when deleting volumes. It can match a virtual IP pool specified in the `vipPool` property of a storage class, or you can specify a different virtual IP pool. For more guidance, see [Extra Configuration Steps for Legacy CSI Local-Mount Deletions](#).
 - `deletionViewPolicy: "<deletion virtual IP pool>"`: Specify the name of the VAST Cluster view policy to be used when deleting volumes. It can match a view policy specified in the `viewPolicy` property of a storage class, or you can specify a different view policy. For more guidance, see [Extra Configuration Steps for Legacy CSI Local-Mount Deletions](#).
 3. Set storage class options:
 - `<storage class name>` (required): Provide a name to identify the storage class. For more information about Kubernetes storage classes, see [Using Storage Classes](#).



Note

Define at least one storage class.

- `<option 1>...<option n>`: Specify parameters to be used when provisioning storage for PVCs with this storage class. For information on each option, see [Storage Class Option Reference](#).

The required options are as follows:

```
storageClasses:
  <storage class name>:
    vipPoolFQDN: "<pool FQDN>" | vipPool: "<pool name>"
    storagePath: "<path>"
    viewPolicy: "<policy name>"
```

If you want to configure storage class-specific session options, add the following parameters:

- `secretName` (required if no global session options are set): The name of the Kubernetes [secret](#) that contains information about the VAST cluster on which to provision volumes for this particular storage class, the corresponding VMS user credentials and, optionally, the SSL certificate. For more information, see [Provisioning Volumes on Multiple VAST Clusters](#).
- `secretNamespace` (optional): If the storage class Kubernetes secret (specified on `secretName`) was created in a namespace that is different from that used to install the VAST CSI Driver's Helm chart, add this parameter to specify the namespace of the Kubernetes secret.

4. (Optional) Configure registration of VAST CSI Driver with `kubelet`:

- `kubeletPath`: "<your kubelet root directory>" (optional): Add this option if you are going to run VAST CSI Driver on a Kubernetes cluster where the kubelet root directory is not `/var/lib/kubelet`.

The following snippet shows a sample configuration file for VAST CSI Driver:

```
secretName: "vast-mgmt"
endpoint: "vms.example.com"
deletionVipPool: "vipool-1"
deletionViewPolicy: "default"
verifySsl: true

StorageClassDefaults:
  volumeNameFormat: "csi:{namespace}:{name}:{id}"
  ephemeralVolumeNameFormat: "eph:{namespace}:{name}:{id}"
  vipPool: "main"

storageClasses:
  vastdata-filesystem:
    vipPool: "vipool-1"
    storagePath: "/k8s"
    viewPolicy: "default"
    mountOptions:
      - proto=tcp
      - port=2049
      - vers=3
  vastdata-filesystem2:
    secretName: "session-options-for-fs2"
    secretNamespace: "nm"
    vipPool: "vipool-2"
    storagePath: "/fs2/path"
    viewPolicy: "policy-for-fs2"
    mountOptions:
      - proto=rdma
      - port=20049
  my-custom-storage-class:
    secretName: "custom-tenant"
    secretNamespace: "tenant2"
    storagePath: "/data"
    vipPool: "vip-tenant2"
```

Verify the Configuration File

Verify the newly created chart configuration file:

```
helm template <release name> <repo>/<chart> -f <filename>.yaml -n <namespace>
```

Where:

- `<release name>` identifies the release being deployed.
- `<repo>` is the name of the VAST CSI Driver [Helm repository](#).

- `<chart>` is the name of the VAST CSI Driver [Helm chart](#) (`vastcsi`).
- `<filename>.yaml` is the VAST driver chart configuration file.
- `<namespace>` determines the Kubernetes namespace to which the release is deployed. If this parameter is not specified, the default namespace is used. Otherwise, [create](#) a custom namespace prior to installing the VAST driver chart.

For example:

```
helm template csi-driver vastcsi/vastcsi -f values.yaml
```

The output is similar to the following:

```
---
# Source: vastcsi/templates/serviceaccount.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: csi-vast-controller-sa
  namespace: "default"
  labels:
    helm.sh/chart: vastcsi-0.1.0
    app.kubernetes.io/name: vastcsi
    app.kubernetes.io/instance: csi-driver
    app.kubernetes.io/version: "2.6.0"
    app.kubernetes.io/managed-by: Helm
---
# Source: vastcsi/templates/serviceaccount.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
<...>
```

Install the VAST CSI Driver Helm Chart

Installing a Helm chart results in deployment of a VAST driver's *release* in your Kubernetes environment. A release is identified with its release name, which you supply during the install.

To install the VAST CSI Driver Helm chart:

1. Refresh Helm repository information:

```
helm repo update
```

2. Run the following command to initiate the install:

```
helm install <release name> <repo>/<chart> -f <filename>.yaml -n <namespace> [--set-file s
slCert=VastCerts/RootCA.crt]
```

Where:

- `<release name>` identifies the release being deployed.
- `<repo>` is the name of the VAST CSI Driver [Helm repository](#).
- `<chart>` is the name of the Helm chart to be installed (`vastcsi`).
- `<filename>.yaml` is the [Helm chart configuration file](#) for VAST CSI Driver.
- `<namespace>` (optional) determines the Kubernetes namespace to which the release is deployed. If this parameter is not specified, the default namespace is used. Otherwise, [create](#) a custom namespace prior to installing the Helm

chart.

- `--set-file sslCert=VastCerts/RootCA.crt` (optional) specifies the path to a self-signed SSL certificate to [secure](#) the connection to the VAST cluster.

For example:

```
helm install csi-driver vastcsi/vastcsi -f values.yaml
```

The output is similar to the following:

```
NAME: csi-driver
LAST DEPLOYED: Thu Dec  5 05:24:36 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing vastcsi.
```

Your release is named `csi-driver`.
The release is installed in namespace `default`

To learn more about the release, try:

```
$ helm status -n default csi-driver
$ helm get all -n default csi-driver
```

<...>

3. Verify that the Helm chart has been installed as follows:

- Check the release status with the following command:

```
helm status -n <namespace> <release name>
```

For example:

```
helm status -n default csi-driver
```

- Ensure that the release appears in the list of releases:

```
helm list -n <namespace>
```

For example:

```
helm list -n default
```

The output is similar to the following:

NAME	NAMESPACE	REVISION	UPDATED
STATUS	CHART	APP VERSION	
csi-driver	default	1	2024-12-04 04:25:33.783236165 +0000 UTC
deployed	vastcsi-0.1.0	2.4.0	

Verify VAST CSI Driver Deployment

To verify your VAST CSI Driver deployment, test it by launching an application using a Persistent Volume Claim (PVC).

In the following example, VAST CSI Driver will create a PVC by provisioning a volume of 1Gi from VAST using the `vastdata-filesystem` storage class. It will create a set of 5 pods running an application consisting of Docker containers with mounts to that volume. The application is a shell program that appends a date to a text file.

To verify the deployment with a test application:

1. Create a Kubernetes YAML configuration file for a PVC as follows:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: vast-pvc-1
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: vastdata-fileSystem
```

2. Apply the PVC configuration file:

```
kubectl apply -f <filename>.yaml
```

3. Verify that the PVC has been created with the following commands:

```
kubectl get pvc
```

```
kubectl get pv
```

4. Create a Kubernetes YAML configuration file for the test application:

```
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: test-app-1
spec:
  serviceName: "test-service-1"
  replicas: 5
  selector:
    matchLabels:
      app: test-app-1
  template:
    metadata:
      labels:
        app: test-app-1
        role: test-app-2
    spec:
      containers:
        - name: my-frontend
          image: busybox
          volumeMounts:
            - mountPath: "/shared"
              name: my-shared-volume
          command: [ '/bin/sh', '-c', 'while true; do date -Iseconds >> /shared/$HOSTNAME;
sleep 1; done' ]
      volumes:
        - name: my-shared-volume
          persistentVolumeClaim:
            claimName: vast-pvc-1
```

5. Apply the test application configuration file:

```
kubectl apply -f <filename>.yaml
```

6. Monitor the VAST Cluster path specified in the definition of the `vastdata-fileSystem` storage class. You will see a PVC per container, in this example 5. In each, a date will be appended to a text file.

Deploying VAST CSI Driver on OpenShift with VAST CSI Operator

VAST CSI Operator is an [OpenShift operator](#) that helps deploy, configure and upgrade VAST CSI Driver in Red Hat OpenShift environments.

To deploy VAST CSI Driver using VAST CSI Operator:



Note

Before you start, ensure that the VAST cluster on which you are going to provision storage is [configured](#) for CSI use.

1. Follow [OpenShift Operator guidelines](#) to install VAST CSI Operator.
2. Use VAST CSI Operator to deploy the VAST CSI Driver [custom resources](#): `VastCSIDriver`, `VastCluster` and `VastStorage`.

Each of the three resources is required.
3. [Use](#) the storage class generated by the `VastStorage` custom resource instance in your PVCs to provision storage.
4. [Assign](#) OpenShift Security Context Constraints (SCCs) to VAST CSI Driver.



Note

VAST CSI Operator does not provide a CRD for VAST snapshots. If you are going to use VAST snapshots with VAST CSI Driver, [install](#) the snapshot CRDs manually.

Custom Resource Definitions in VAST CSI Operator

VAST CSI Operator introduces the following Custom Resource Definitions (CRD):

- `VastCSIDriver` is a common specification for the VAST CSI Controller and VAST CSI Node, for example:

```
apiVersion: storage.vastdata.com/v1
kind: VastCSIDriver
metadata:
  name: csidriver
  namespace: vast-csi
spec:
  image:
    csiVastPlugin:
      image: vastdataorg/csi:v2.6.0
```

- `VastCluster` specifies the VAST endpoint, username and password to connect to the VAST cluster where volumes will be provisioned. These settings are stored in a VAST CSI Driver's Kubernetes [secret](#).

For example:

```
apiVersion: storage.vastdata.com/v1
kind: VastCluster
```

```

metadata:
  name: cluster
  namespace: vast-csi
spec:
  endpoint: 198.151.100.12
  username: admin
  password: "xxxxxxx"

```

You can create multiple `VastCluster` instances to access different VAST clusters, or to access the same VAST cluster using different credentials.

- `VastStorage` defines VAST storage options to be used when provisioning volumes on the VAST cluster, such as the path to the volumes on the VAST cluster, VAST view policy, VAST virtual IP pool name or FQDN, additional mount options, and so on. This resource generates a [storage class](#) (and optionally a [snapshot class](#)) that you can use in your PVCs.

For example:

```

apiVersion: storage.vastdata.com/v1
kind: VastStorage
metadata:
  name: vastdata-filessystem
  namespace: vast-csi
spec:
  clusterName: cluster
  storagePath: "/k8s"
  viewPolicy: "default"
  vipPool: "vipool-1"
  allowVolumeExpansion: true
  createSnapshotClass: true

```

Using VastStorage-generated Storage Class in PVCs

In your PVC, specify the name of the storage class generated based on the `VastStorage` CRD as the `storageClassName`, for example:

For example:

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: vastdata-filessystem

```

Assigning SCCs to VAST CSI Driver

assign OpenShift Security Context Constraints (SCC) to VAST CSI Driver service accounts so that VAST CSI Driver is able to create mount points on the host machine:

```

oc adm policy add-scc-to-user privileged -z <release name>-vast-controller-sa -n <namespace name>
oc adm policy add-scc-to-user privileged -z <release name>-vast-node-sa -n <namespace name>

```

Where:

- `<release name>` identifies the VAST CSI Driver release being deployed.
- `<namespace name>` is the namespace to which VAST CSI Driver will be deployed.

For example:

```
oc adm policy add-scc-to-user privileged -z csi-vast-controller-sa -n vastcsi
oc adm policy add-scc-to-user privileged -z csi-vast-node-sa -n vastcsi
```

Configuring SSL Encryption for VAST CSI Driver

- [Overview](#)
- [Enabling SSL Encryption](#)
- [Uploading a CA-Signed SSL Certificate to VAST Cluster](#)
- [Supplying a Self-Signed SSL Certificate to VAST CSI Driver](#)
- [Replacing a Self-Signed SSL Certificate for VAST CSI Driver](#)
- [Removing a Self-Signed SSL Certificate from VAST CSI Driver](#)

Overview

You can secure the connection between VAST CSI Driver and the VAST cluster with SSL encryption as follows:

1. [Enable](#) SSL encryption.
2. Do one of the following to install an SSL certificate:
 - If you want to use a Certified Authority-signed SSL certificate, upload it to the VAST cluster. Follow the SSL certificate upload procedure provided in the *VAST Cluster Administrator's Guide*.
 - If you want to use a self-signed SSL certificate, [supply](#) it to VAST CSI Driver.

Enabling SSL Encryption

By default, SSL encryption is disabled.

To enable SSL encryption:

1. Add the `verifySsl=true` option to the VAST CSI Driver's [Helm chart configuration file](#), for example:

```
secretName: "vast-mgmt"
endpoint: "my.endpoint"
verifySsl: true
<...>
```

2. [Install](#) or [upgrade](#) the VAST CSI Driver Helm chart.

Uploading a CA-Signed SSL Certificate to VAST Cluster

Follow the guidelines provided in the *VAST Cluster Administrator's Guide* to upload a CA-signed SSL certificate to the VAST cluster.

Supplying a Self-Signed SSL Certificate to VAST CSI Driver

You can either point to a file that contains a self-signed SSL certificate file, or specify an existing Kubernetes secret that contains the certificate. These two methods are mutually exclusive.

Do either of the following:

- [Install](#) or [upgrade](#) the VAST CSI Driver Helm chart with the `--set-file sslCert=<path to certificate file>` option specified, for example:

```
helm install csi-driver vast/vastcsi -f values.yaml --set-file sslCert=<path to certificate file>
```

OR

- Create a Kubernetes secret with the SSL certificate and specify the secret using the `sslCertsSecretName` option in the VAST CSI Driver [Helm chart configuration file](#):

1. Create a Kubernetes secret that contains the SSL certificate, for example:

```
kubectl create secret generic vast-ca --from-file=ca-bundle.crt=<path to certificate file>
```

2. Specify the newly created secret on the `sslCertsSecretName` option in the VAST CSI Driver chart configuration file, for example:

```
secretName: "vast-mgmt"
endpoint: "my.endpoint"
verifySsl: true
sslCertsSecretName: "vast-ca"
<...>
```

3. [Install](#) or [upgrade](#) the VAST CSI Driver Helm chart (without specifying `--set-file sslCert`).

Replacing a Self-Signed SSL Certificate for VAST CSI Driver

Choose either of the following, depending on how you supplied the old self-signed SSL certificate:

- If you supplied the old SSL certificate using the `--set-file sslCert` option on the Helm chart install or upgrade command:

- [Upgrade](#) the Helm chart with `--set-file sslCert` pointing to the new SSL certificate file. For example:

```
helm upgrade csi-driver vast/vastcsi -f values.yaml --set-file sslCert=<path to new certificate>
```

OR

- If the old SSL certificate was supplied via `sslCertsSecretName` in the VAST CSI Driver chart configuration file:

1. Create a new Kubernetes secret with the new SSL certificate:

```
kubectl create secret generic vast-ca-new --from-file=ca-bundle.crt=<path to new certificate file>
```

2. Ensure that the new SSL certificate is specified on the `sslCertsSecretName` option in the VAST CSI Driver chart configuration file:

```
secretName: "vast-mgmt"
endpoint: "my.endpoint"
verifySsl: true
sslCertsSecretName: "vast-ca-new"
<...>
```

3. [Upgrade](#) the Helm chart, for example:

```
helm upgrade csi-driver vast/vastcsi -f values.yaml
```

Removing a Self-Signed SSL Certificate from VAST CSI Driver

Choose either of the following, depending on how you supplied the self-signed SSL certificate:

- If you used `--set-file sslCert` to supply the SSL certificate:
 - [Upgrade](#) the Helm chart without the `--set-file sslCert` option specified. For example:

```
helm upgrade csi-driver vast/vastcsi -f values.yaml
```

OR

- If the old SSL certificate was supplied via `sslCertsSecretName`:
 1. Remove the `sslCertsSecretName` option from the VAST CSI Driver chart configuration file.
 2. [Upgrade](#) the Helm chart (without specifying `--set-file sslCert`), for example:

```
helm upgrade csi-driver vast/vastcsi -f values.yaml
```

Securing VAST CSI Driver in Kubernetes Multi-Tenant Environment

When using VAST CSI Driver in a Kubernetes multi-tenant environment, follow these guidelines to prevent security issues where one client can see the IP address of NFS servers and try to exploit or access other mount points in a direct or indirect attack.

- Do not run your container as privileged.

This will block root from within a container from mounting, as shown in the following example where root within a container has searched the mount point, created a directory and attempted to mount, but failed:

```
[root@es-master-0 elasticsearch]# mkdir /mnttest
[root@es-master-0 elasticsearch]# mount|grep elastic
10.101.12.16:/elastic/pvc-44b42a13-8dc1-47ff-8c9c-2092311bde1f on /usr/share/elasticsearch/data type nfs (rw,relatime,vers=3,rsize=1048576,wsiz=1048576,namlen=255,hard,proto=rdma,port=20049,timeo=600,retrans=2,sec=sys,mountaddr=10.101.12.16,mountvers=3,mountproto=tcp,local_lock=none,addr=10.101.12.16)
[root@es-master-0 elasticsearch]# mount -t nfs 10.101.12.16:/elastic /mnttestmount: permission denied
```



Note

For information about running a container as privileged, see [this document](#).

- Isolate your pods and establish a proper flow of NFS calls from the container to the NFS server.

By default, pods are not isolated; they accept traffic from any source and can send traffic to any destination.

Pods become isolated by having a network policy that selects them. Once there is a network policy in a namespace selecting a particular pod, Kubernetes will reject any inbound or outbound connections that are not allowed by any network policy. Other pods in the namespace that are not selected by any network policy will continue to accept all traffic.

POSIX system calls from within a container are traveling out of the container to the NFS client on the host, and it is from the host that NFS packets are sent to the NFS server. So with proper network isolation, the container cannot communicate directly with the NFS server, even though it knows its IP from the mount information.



Note

For information about Kubernetes network policies, see [this document](#).

Provisioning Volumes with VAST CSI Driver

Using Storage Classes

VAST CSI Driver supports multiple Kubernetes storage classes, enabling you to provision multiple storage paths within VAST Cluster, each configured (via VMS) with its own set of access policies, path protection, or replication policies. Each storage class can have its own path, virtual IP pool, a set of mount options, and other parameters.

A storage path is a path within VAST Cluster where VAST CSI Driver will create volume directories. The storage path is specified in the `root_export` parameter in the YAML configuration file; for example: `/a/b/c`. (Note that `/` cannot be used as a storage path.) The storage path must be mountable by VAST CSI Driver.

During initial deployment of VAST CSI Driver, you define one or more storage classes in the VAST CSI Driver [Helm chart configuration file](#). Later you can add more storage classes by creating and applying a Kubernetes YAML configuration file.

- [Adding a Storage Class](#)
- [Storage Class Option Reference](#)

Adding a Storage Class

To add a storage class using a Kubernetes YAML configuration file:

1. Create a YAML configuration file that defines a new storage class with the following **required** parameters:



Note

For a complete list of options that can be specified for a storage class, see [Storage Class Option Reference](#).

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage class name>
provisioner: csi.vastdata.com
parameters:
  subsystem: <NVMe subsystem name>
  vip_pool_fqdn: <virtual IP pool FQDN> | vip_pool_name: <virtual IP pool name>
  root_export: '/k8s-2'
  view_policy: 'default'
  <optional: pairs of secrets and secret namespaces for each volume processing stage>
```

The following example shows creating a storage class named `vastdata-filesystem-2` that uses path `/k8s-2`, view policy `default`, and virtual IP pool `test1`. In the example, no custom mount options are set (`mountOptions` is an empty string).



Tip

For more examples, see <https://github.com/vast-data/vast-csi/tree/v2.6/examples/csi>.

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: vastdata-filesystem-2
provisioner: csi.vastdata.com
parameters:
  csi.storage.k8s.io/controller-expand-secret-name: vast-mgmt
  csi.storage.k8s.io/controller-expand-secret-namespace: default
  csi.storage.k8s.io/controller-publish-secret-name: vast-mgmt
  csi.storage.k8s.io/controller-publish-secret-namespace: default
  csi.storage.k8s.io/node-publish-secret-name: vast-mgmt
  csi.storage.k8s.io/node-publish-secret-namespace: default
  csi.storage.k8s.io/node-stage-secret-name: vast-mgmt
  csi.storage.k8s.io/node-stage-secret-namespace: default
  csi.storage.k8s.io/provisioner-secret-name: vast-mgmt
  csi.storage.k8s.io/provisioner-secret-namespace: default
  vip_pool_fqdn: 'MyDomain'
  root_export: '/k8s-2'
  view_policy: 'default'
  volume_name_fmt: csi:{namespace}:{name}:{id}
mountOptions:
  - ''
allowVolumeExpansion: true

```

2. Deploy the YAML configuration file:

```
kubectl apply -f <filename>.yaml
```

3. Verify that the storage class has been added:

```
kubectl get storageclasses
```


The output is similar to the following:

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE	ALLOWVOLUMEEXPANSION
standard (default)	k8s.io/minikube-hostpath	Delete	Immediate	false
4dlh				
vastdata-filesystem	(default) csi.vastdata.com	Delete	Immediate	true
58m				
vastdata-filesystem-2	(default) csi.vastdata.com	Delete	Immediate	true
56m				
vastdata-filesystem-3	(default) csi.vastdata.com	Delete	Immediate	true
54m				

Storage Class Option Reference

You can specify storage class options as follows:

- In the [Helm chart configuration file](#) created for VAST CSI Driver during initial deployment,
- In a Kubernetes YAML configuration file [deployed](#) at a later stage.
- In VAST CSI Operator's `VastStorage` custom resource definition, when deploying in an OpenShift environment using VAST CSI Operator.

Option in Helm chart configuration file / VAST CSI Operator's <code>VastStorage</code>	Option in Kubernetes YAML configuration file	Description
<code>ephemeralVolumeNameFormat: "<format>"</code>	<code>eph_volume_name_fmt: "<format>"</code>	<p>A format string that controls naming of Kubernetes ephemeral volumes created through VAST CSI Driver.</p> <p>If not specified, the default format <code>csi:{namespace}:{name}:{id}</code> is used.</p>
<code>mountOptions: "<options>"</code>	<code>mount_options: "<options>"</code>	<p>Specify NFS mount options for VAST CSI Driver to use when mounting a volume for a PVC with this storage class.</p> <p>Examples:</p> <pre>mountOptions: - "proto=rdma" - "port=20049"</pre> <pre>mountOptions: - debug - nosuid - soft</pre> <pre>mountOptions: - nfsvers=4</pre> <div>  <p>Note</p> <p>These mount options override host-specific mount options defined through <code>/etc/nfsmount.conf.d</code>.</p> </div>
<code>qosPolicy: "<QoS policy name>"</code>	<code>qos_policy: "<QoS policy name>"</code>	<p>A Quality of Service (QoS) policy to be associated with automatically created views.</p> <p>In a QoS policy, you can set limits to define minimum and maximum allowed bandwidth and/or IOPS for read and/or write operations per view. For more information, see Configure a QoS Policy.</p>
<code>secretName: "<secret name>"</code> <code>secretNamespace: "<secret's namespace>"</code>	<code><pairs of secrets and secret namespaces for each provisioning stage></code>	<p>These options lets you supply information for communicating with the VAST cluster:</p> <ul style="list-style-type: none"> <code><secret name></code> is the name of the Kubernetes secret that contains information about the VAST

Option in Helm chart configuration file / VAST CSI Operator's VastStorage	Option in Kubernetes YAML configuration file	Description
		<p>cluster on which to provision volumes for this particular storage class, the corresponding VMS user credentials and, optionally, the SSL certificate. For more information, see Provisioning Volumes on Multiple VAST Clusters.</p> <ul style="list-style-type: none"> <secret namespace>: if the storage class Kubernetes secret was created in a namespace that is different from that used to install the driver's Helm chart, add this parameter to specify the namespace of the Kubernetes secret. <p>If the secret and its namespace are defined as global options in the Helm chart configuration file, they are automatically propagated to each storage class and each provisioning stage therein. In this case, you do not need to explicitly include the per-stage secrets with their corresponding namespaces in the storage class definition.</p> <p>If no global settings exist for the secret and its namespace, you need to specify them directly in the storage class definition in the following format. Note that you can specify a different value for each stage:</p> <pre>csi.storage.k8s.io/controller-expand-secret-name: <secret name> csi.storage.k8s.io/controller-expand-secret-namespace: <secret namespace> csi.storage.k8s.io/controller-publish-secret-name: <secret name> csi.storage.k8s.io/controller-publish-secret-namespace: <secret namespace> csi.storage.k8s.io/node-publish-secret-name: <secret name> csi.storage.k8s.io/node-publish-secret-namespace: <secret namespace> csi.storage.k8s.io/node-stage-secret-name: <secret name> csi.storage.k8s.io/node-stage-secret-namespace: <secret namespace> csi.storage.k8s.io/provisioner-secret-name: <secret name> csi.storage.k8s.io/provisioner-secret-namespace: <secret namespace></pre>
storagePath: "<path>"	root_export: "<path>"	The storage path within VAST Cluster to be used when dynamically provisioning Kubernetes volumes. VAST CSI Driver will automatically create a VAST Cluster view for each volume being provisioned.

Option in Helm chart configuration file / VAST CSI Operator's <code>VastStorage</code>	Option in Kubernetes YAML configuration file	Description
		<div> Caution <p>Do not specify '/' as the <code><path></code>.</p> </div> <p>This option is required when defining a storage class in the Helm chart configuration file.</p>
<pre>viewPolicy: "<policy name>"</pre>	<pre>view_policy: "<policy name>"</pre>	<p>The name of the VAST Cluster view policy to be assigned to VAST Cluster views created by VAST CSI Driver.</p> <p>A view policy defines access settings for storage exposed through a VAST Cluster view. For more information, see Configure View Policies.</p> <p>All view policies used with VAST CSI Driver must have the same security flavor.</p> <p>If you are going to use VAST CSI Driver with VAST Cluster 4.6 or later, a view policy set for a storage class must belong to the same VAST Cluster tenant as the virtual IP pool(s) specified for that storage class.</p> <p>This option is required when defining a storage class in the Helm chart configuration file.</p>
<pre>vipPool: "<virtual IP pool name>"</pre>	<pre>vip_pool_name: "<virtual IP pool name>"</pre>	<p>The name of the virtual IP pool to be used by VAST CSI Driver. For more information, see Set up Virtual IP Pools.</p> <p>If you are going to use VAST CSI Driver with VAST Cluster 4.6 or later, a virtual IP pool that you specify for a storage class must belong to the same VAST Cluster tenant as the view policy specified for this storage class.</p> <p>Either <code>vipPool</code> or <code>vipPoolFQDN</code> option is required when defining a storage class in the Helm chart configuration file. These options are mutually exclusive. When <code>vipPool</code> is used, VAST CSI Driver makes an additional call to the VMS to obtain the IP, which may impact performance when mounting volumes.</p>

Option in Helm chart configuration file / VAST CSI Operator's <code>vastStorage</code>	Option in Kubernetes YAML configuration file	Description
<code>vipPoolFQDN: "virtual IP pool's domain name"</code>	<code>vip_pool_fqdn: "virtual IP pool's domain name"</code>	The domain name of the virtual IP pool to be used by VAST CSI Driver. For more information, see Set up Virtual IP Pools . Either <code>vipPoolFQDN</code> or <code>vipPool</code> option is required when defining a storage class in the Helm chart configuration file. These options are mutually exclusive. With <code>vipPoolFQDN</code> , the IP for volume mounting is obtained through DNS, which improves mounting times.
<code>volumeNameFormat: "<format>"</code>	<code>volume_name_fmt: "<format>"</code>	A format string that controls naming of volumes created through VAST CSI Driver. If not specified, the default format <code>csi:{namespace}:{name}:{id}</code> is used.

Provisioning Static Volumes

In addition to [dynamically creating](#) VAST views for each PVC, VAST CSI Driver can expose existing data as statically provisioned persistent volumes.

You can control whether VAST CSI Driver automatically creates a view and/or a quota for static volumes by leveraging the following options:

- `static_pv_create_views`, when set to `yes`, instructs VAST CSI Driver to create a view for the static volume in case no such view exists. If this option is omitted or set to `no`, VAST CSI Driver assumes that there is a predefined view on the VAST cluster that can be used for this volume.
- `static_pv_create_quotas`, when set to `yes`, instructs VAST CSI Driver to create a quota for the static volume if no quota exists. If this option is omitted or set to `no`, no quota is created.

These options are specified under `volumeAttributes` in the `csi` part of the PV definition YAML. You can enter `true` or `1` instead of `yes` as the option value.

Note that VAST CSI Driver does not automatically delete views or quotas it has created, even upon deletion of their respective PVs.

Below is an example of a YAML definition for a statically provisioned persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: csi-pv-static
spec:
  storageClassName: vastdata-filessystem
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  mountOptions:
    - nfsvers=3
```

```
csi:
  driver: csi.vastdata.com
  volumeAttributes:
    vip_pool_name: vipool-1
    view_policy: default
    size: 1G
    static_pv_create_views: "yes"
    static_pv_create_quotas: "yes"
  controllerPublishSecretRef:
    name: vast-mgmt
    namespace: default
  volumeHandle: /full/path/to/view
```

The PVC for a static volume is similar to the following:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc-static
spec:
  storageClassName: vastdata-filessystem
  volumeName: csi-pv-static
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Provisioning Ephemeral Volumes

VAST CSI Driver supports Ephemeral Volumes (EVs). You can provision PVCs directly from pod definitions, which are discarded once the pod is terminated.

Specify the `volumeAttributes` as follows:

- `root_export`: as in the storage class definition.
- `view_policy`: as in the storage class definition.
- `vip_pool_name`: as in the storage class definition.
- `size`: as in the PVC.

For example:

```
apiVersion: v1
kind: Pod
metadata:
  name: some-pod
spec:
  containers:
    - command:
        - sh
        - -c
        - while true; do date -Iseconds >> /shared/$HOSTNAME; sleep 1; done
      image: busybox
      name: my-frontend
      volumeMounts:
        - mountPath: /shared
          name: my-eph-vol
  volumes:
    - csi:
        driver: csi.vastdata.com
```

```

volumeAttributes:
  root_export: /k8s
  view_policy: default
  vip_pool_name: vipool-1
  size: 1G
name: my-eph-vol

```

Provisioning Volumes on Multiple VAST Clusters

VAST CSI Driver enables you to provision volumes on multiple VAST clusters.

The VAST cluster to use can be specified per storage class, together with the user credentials needed to connect to the VAST cluster. You can also configure multiple storage classes to connect to the same VAST cluster using different user credentials.

If your Kubernetes volumes are protected with VAST Cluster snapshots, you can specify the VAST cluster and user credentials per snapshot class.



Note

Class-specific parameters take precedence over global parameters that apply to all classes.

To configure a storage class or a snapshot class to use a particular combination of a VAST cluster and user credentials:

1. [Create](#) a Kubernetes secret that will be used to provision storage on multiple VAST clusters. The secret specifies the following:
 - The hostname of the **VAST cluster** on which you want to provision volumes
 - The VMS user's **username** and **password**
 - If using [SSL encryption](#) with a self-signed SSL certificate:
 - The **SSL certificate** to be used to protect the connection
 - If you are creating the secret in a namespace that is different from the namespace used to install the VAST CSI Driver Helm chart:
 - The **namespace** where the secret is created



Note

When provisioning storage on multiple VAST clusters for Kubernetes [ephemeral volumes](#), the namespace where the Kubernetes secret is created must be the same as the namespace where the VAST CSI Driver Helm chart is installed.

2. [Create](#) a new storage class (or a snapshot class) and supply the newly created secret using the `secretName` parameter of the class.

If you have creating the secret in a namespace that is different from the namespace used to install the VAST CSI Driver Helm chart, specify the secret's namespace on the `secretNamespace` parameter.

For example:

```
storageClasses:
  vastdata-filestorage1:
    storagePath: /foo
    viewPolicy: default1
    vipPool: vippool1
    secretName: vast-mgmt1
    secretNamespace: default
```

If a storage class or a snapshot class does not contain the `secretName` parameter, VAST CSI Driver uses the information from the `secretName` and `endpoint` parameters specified in the VAST CSI Driver [Helm chart configuration file](#) (`values.yaml`) for all storage classes.

Managing NFS Mount Options

NFS mount options that VAST CSI Driver uses when mounting a volume for a PVC can be set as follows:

- Per storage class

Supply mount options for a particular storage class using the `mountOptions` parameter in the VAST CSI Driver's [Helm chart configuration file](#) or the `mount_options` parameter in the Kubernetes [YAML configuration file](#).

- Per host

Supply mount options in a host-specific NFS mount configuration file (`.conf`) located in the `/etc/nfsmount.conf.d` directory.



Note

Ensure that the underlying OS of the worker nodes supports use of the `/etc/nfsmount.conf.d` directory for setting mount options (see `man nfsmount.conf`).

If both storage class-specific and host-specific options are specified, storage class-specific options take precedence.

To make VAST CSI Driver ignore host-specific mount options, set the `propagateHostMountOptions` parameter to `false` in the driver's Helm chart configuration file:

```
node:
  propagateHostMountOptions: false
```

Setting Up DNS-Based Virtual IP Resolution

When mounting volumes on the VAST cluster, VAST CSI Driver assigns a virtual IP to each volume being mounted. The virtual IP is taken from one or more virtual IP pools defined on the VAST cluster.

You can control how VAST CSI Driver obtains the IP to use, per storage class:

- By using DNS to resolve a virtual IP pool's FQDN into an IP. This method is available starting with VAST CSI Driver 2.4.1 to improve performance when mounting volumes.

Specify the FQDN of the virtual IP pool on the `vipPoolFQDN` option in the VAST CSI Driver Helm chart configuration file.

Ensure that the VAST cluster has DNS configured, and the virtual IP pool has *Virtual IP Pool Domain Name* defined in its

settings.

- By making an additional call to VMS to retrieve an IP. This method is used in VAST CSI Driver versions prior to 2.4.1.

Specify the name of the virtual IP pool on the `vipPool` parameter in the VAST CSI Driver Helm chart configuration file.

Either `vipPool` or `vipPoolFQDN` option is required when defining a storage class in the Helm chart configuration file. These options are mutually exclusive.

Protecting Block Volumes with Snapshots and Clones

Overview of VAST Snapshots and Clones

You can protect Kubernetes volumes using VAST Cluster snapshots.

A snapshot is a reference point that preserves the exact folder structure and data content of a data path at a point in time. Snapshots serve as a virtual copy of the data you had at previous points in time and enable you to restore files or directories after deletion or modification.



Note

For more information about VAST snapshots, see the *VAST Cluster Administrator's Guide*.

You can [create](#) snapshots for your Persistent Volume Claims (PVCs) to be able to [restore](#) the data based on an existing snapshot.

VAST CSI Driver lets you create an instantly writeable [clone](#) of a Kubernetes volume.

VAST CSI Driver also supports provisioning for read-only PVCs based on Kubernetes-created snapshots.

Create a Volume Snapshot

To create a snapshot of a Kubernetes volume using a Kubernetes YAML configuration file:

1. Create a YAML configuration file that adds a volume snapshot. In the file (see the example below for correct indentation and line breaks when specifying the keywords):

- In `metadata:`
 - `name` is the Persistent Volume Claim (PVC) name to store the snapshot.
- In `spec:`
 - `snapshotClassName:` the name of the snapshot class.
 - In `source:`
 - `name` is the PVC name for the volume being protected.
 - `kind` indicates that this is a persistent volume claim.
- In `parameters:`
 - `snapshot_name_fmt` sets the format for snapshot names.

The default format is `csi:{namespace}:{name}:{id}`, where:

- `namespace` is the Kubernetes namespace where snapshot parent workload is located.

- `name` is the volume name.
- `id` is the snapshot ID set by Kubernetes.

For example:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: snapshot-pvc-1
spec:
  snapshotClassName: vastdata-snapshot
  source:
    name: vast-pvc-1
    kind: PersistentVolumeClaim
parameters:
  snapshot_name_fmt: "csi:{namespace}:{name}:{id}"
```

2. Apply the YAML configuration file:

```
kubectl apply -f vast-csi-deployment.yaml
```

3. Verify that the PVC snapshot has been created with the following command:

```
kubectl get vs
```

The output is similar to the following:

NAME	READYTOUSE	SOURCEPVC	SOURCESNAPSHOTCONTENT	RESTORESIZE	SNAPSHOTCLASS
SNAPSHOTCONTENT			CREATIONTIME	AGE	
snapshot-pvc-1	true	vast-pvc-1	0		vastdata-snapshot
snapshotcontent-8b1627cc-d50a-4bdb-a1b4-67931d36eb85	50m			49m	

Restore a Volume from a Snapshot

To restore a PVC from an existing snapshot, attach the snapshot to another PVC.

You can choose to provide read-only or read/write access to the restored volume using `accessModes` options in the YAML configuration file:

- If attached in read-only mode (`ReadOnlyMany`), the restored volume is mounted to the snapshot's source folder (`<snapshot path>/ .snapshot/<snapshot name>`). It does not have a view or a quota on the VAST cluster, not can it be expanded through Kubernetes volume expansion.
- If attached in read/write mode (`ReadWriteMany`), a view and a quota is created automatically for it on the VAST cluster, and the restored volume is mounted to the view's directory. The restored volume can be expanded like any other regular volume.

To restore a volume:

1. Create a YAML configuration file to attach the snapshot to a PVC. In the file (see the example below for correct indentation and line breaks when specifying the keywords):
 - In `metadata`:
 - `name` is the Persistent Volume Claim (PVC) name for the volume that will contain the restored data.
 - In `spec`:
 - `storageClassName` is the name of the storage class used for the PVC.

- In `dataSource`:
 - `name` is the PVC name of the volume that stores the snapshot.
- In `accessModes`, specify the access mode for the restored volume:
 - `ReadWriteMany` for read/write
 - `ReadOnlyMany` for read-only

For example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-restore
spec:
  storageClassName: vastdata-fileSystem
  dataSource:
    name: snapshot-pvc-1
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

2. Deploy the YAML configuration file:

```
kubectl apply -f vast-csi-deployment.yaml
```

The output is similar to the following:

```
persistentvolumeclaim/pvc-restore created
```

3. Verify that the restored PVC can be displayed:

```
kubectl get pvc
```

The output is similar to the following:

NAME	STATUS	VOLUME	CAPACITY	ACCESS	MODES	STORAGEC
LASS	AGE					
pvc-restore	Bound	pvc-c212dc9e-1b72-49da-8a58-8c1ceb97b9e3	1Gi	RO		vastdat
a-fileSystem	7s					

Clone a Volume or a Snapshot

With VAST CSI Driver, you can create an instantly writable clone of either a Kubernetes snapshot or another Kubernetes volume.

When cloning a volume, a temporary snapshot is created behind the scenes. This transient snapshot is invisible to Kubernetes, and is eventually deleted by VAST CSI Driver once the clone operation is complete.

As with regular volumes, VAST CSI Driver automatically creates a view and a quota for the resulting clone volume on the VAST cluster. The clone is mounted to the underlying directory of the view and can be used for read and write operations. It can also be expanded through Kubernetes volume expansion.



Note

Cloning is done using *Global Snapshot Clones*, a feature of VAST Cluster 4.6.0 and later. Cloning is not available when using VAST CSI Driver with VAST Cluster 4.5 or earlier.

Requirements and limitations include:

- VAST Cluster 4.6.0 or later is required.
- Cloning cannot be used if VAST Catalog is enabled on the VAST cluster.
- Cloning to another VAST cluster is not supported.

To clone a volume from its snapshot:

1. Create a YAML configuration file for cloning and specify the following required parameters (see the example below for correct indentation and line breaks when specifying the keywords):

- In `metadata`:
 - `name` is the Persistent Volume Claim (PVC) name for the clone.
- In `spec`:
 - `storageClassName` is the name of the storage class used for the PVC.
 - In `dataSource`:
 - `name` is the PVC name of the volume that stores the snapshot from which cloning is done.

For example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-clone
spec:
  storageClassName: vastdata-filestorage
  dataSource:
    name: snapshot-pvc-1
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

2. Deploy the YAML configuration file:

```
kubectl apply -f vast-csi-deployment.yaml
```

The output is similar to the following:

```
persistentvolumeclaim/pvc-clone created
```

3. Verify that the clone can be displayed:

```
kubectl get pvc
```

The output is similar to the following:

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGEC
LASS	AGE				
pvc-clone	Bound	pvc-c212dc9e-1b72-49da-8a58-8c1ceb97b9e3	1Gi	RWX	vastdat
a-filesystem	7s				

Troubleshooting VAST CSI Driver

Steps to Troubleshoot VAST CSI Driver

- [Run Basic Troubleshooting Commands](#)
- [Check Kubernetes Logs](#)
- [Check VMS Logs](#)

Run Basic Troubleshooting Commands

Run the following commands to troubleshoot VAST CSI Driver:

- Display all pods:

```
kubectl get pods --all-namespaces -o wide
```

- Check logs from the controller.

Typically, there are four containers: `csi-provisioner`, `csi-attacher`, `csi-resizer`, and `csi-vast-plugin`. Check all of them.

Run the following command to display the most important information:

```
kubectl logs csi-vast-controller-0 --namespace vast-csi -c csi-vast-plugin | less
```

- Check logs from the node.

There are probably two containers: `csi-node-driver-registrar` and `csi-vast-plugin`. Check both.

Run the following command to display the most important information:

```
kubectl logs csi-vast-node-<NODE_ID> --namespace vast-csi -c csi-vast-plugin
```

- Check to see if there are any system-level events, such as errors.

Run the following command to list events sorted by timestamp:

```
kubectl get events --sort-by=.metadata.creationTimestamp
```

- Display persistent volume claims:

```
kubectl get pvc
```

The output is similar to the following:

NAME	STATUS	VOLUME	CAPACITY	ACCESS	MODES	STORAGE
CLASS	AGE					
shared-claim	Bound	pvc-41e3cd02-1a80-4dfb-bb50-6d272a9e649e	1Mi	RWO		manage
d-nfs-storage	12d					

- Display persistent volumes:

```
kubectl get pv
```

The output is similar to the following:

NAME	CAPACITY	ACCESS	MODES	RECLAIM	POLICY	STATUS	CLAIM
------	----------	--------	-------	---------	--------	--------	-------

STORAGECLASS	REASON	AGE				
pvc-41e3cd02-1a80-4dfb-bb50-6d272a9e649e	1Mi	RWO	Delete	Bound	defau	
lt/shared-claim managed-nfs-storage	12d					

- Display the defined storage classes:

```
kubectl get sc
```

The output is similar to the following:

NAME	PROVISIONER	AGE
managed-nfs-storage	nfs-client	13d



Note

To get more detailed information, use the `describe` option on each command.

Check Kubernetes Logs

If none of the above helps, check the underlying Kubernetes logs.

The location of Kubernetes logs depends on your operating system. On CentOS, the typical location is `/var/log/messages`.

These logs often contain more detailed information about the container startup process and provide an indication of errors.

Check VMS Logs

VAST CSI Driver contacts VAST Management Service (VMS) to create directories and quotas on behalf of the Kubernetes cluster. So checking VMS logs can help troubleshoot your environment.

VMS logs can be found in `/vast/vman/vms/log` on the VMS node.

Useful information is often found in `vms.log` and `vapi.log`. For example, the `vms.log` can provide the following information (the mount for VAST CSI Driver is `/keystest/k8s`):

```
grep k8s vms.log
[2022-01-05 17:18:47,313] INFO [services.py:382/430] AUDIT - VMS - Manager: admin (10.61.20
3.63) POST /api/quotas/ with body {'create_dir': True, 'name': 'csi:nfs-vol-claim:nfs-vol-cla
im:pvc-0f5d878e-4fa4-4c91-a915-7a5c', 'path': '/keystest/k8s/pvc-0f5d878e-4fa4-4c91-a915-7a5c
9fcdcf2c', 'hard_limit': 1048576}
[2022-01-05 17:18:47,562] INFO [services.py:837/430] Creating Quota with {'create_dir': True,
'name': 'csi:nfs-vol-claim:nfs-vol-claim:pvc-0f5d878e-4fa4-4c91-a915-7a5c', 'path': '/keystes
t/k8s/pvc-0f5d878e-4fa4-4c91-a915-7a5c9fcdcf2c', 'hard_limit': 1048576}
[2022-01-05 17:18:47,573] INFO [services.py:843/430] Creating directory path: /keystest/k8s/p
vc-0f5d878e-4fa4-4c91-a915-7a5c9fcdcf2c
[2022-01-05 17:18:47,578] INFO [services.py:852/430] Created directory path: /keystest/k8s/pv
c-0f5d878e-4fa4-4c91-a915-7a5c9fcdcf2c
```

Resolving PVC Timeouts

PVC Timeout due to Missing NFS Client on Kubernetes Worker Node

If a PVC does not mount with a timeout, a likely outcome is the Kubernetes worker node is missing the NFS client.

Run the `kubectl logs csi-vast-node-<NODE_ID> --namespace vast-csi -c csi-vast-plugin` command to check the node logs.

The output can be similar to the following:

```
Warning FailedMount 50s kubelet, k8s-a-node03 Unable to attach or mount volumes: unmounted volumes=[data], unattached volumes=[mariadb-credentials default-token-zp5 data config]: timed out waiting for the condition
Warning FailedMount 37s kubelet, k8s-a-node03 MountVolume.Setup failed for volume "pvc-4ebe1936-3613-4c37-957c-4d21e623aa98" : kubernetes.io/csi: mounter.SetupAt failed: rpc error: code = DeadlineExceeded desc = context deadline exceeded
```

To resolve the problem, install the NFS client and test mounts locally on the client, then retry via Kubernetes.

To install the NFS client:

```
sudo yum -y install nfs-utils
```

PVC Timeout due to No Communication with VMS

Kubernetes nodes can be on the VAST data network and not on the VAST management network. In this case, the return path from VMS to the Kubernetes node will fail.

Ensure that you have valid routes on the correct network.

On the Kubernetes node, run the following command and verify that the correct interface is used:

```
ip route get <vms_ip>
```

Run this command on the VMS node:

```
ip route get <k8s_node_ip>
```