Get started

Astra Trident

NetApp
February 09, 2023

Table of Contents

Get started . . . 1
- Try it out . . . 1
- Requirements . . . 1
- Install Astra Trident . . . 5
- What’s next? . . . 32

Get started

Try it out

NetApp provides a ready-to-use lab image that you can request through NetApp Test Drive.

Learn about the Test Drive

The Test Drive provides you with a sandbox environment that comes with a three-node Kubernetes cluster and Astra Trident installed and configured. It is a great way to familiarize yourself with Astra Trident and explore its features.

Another option is to see the kubeadm Install Guide provided by Kubernetes.

> You should not use the Kubernetes cluster that you build using these instructions in production. Use the production deployment guides provided by your distribution for creating clusters that are production ready.

If this is the first time you're using Kubernetes, familiarize yourself with the concepts and tools here.

Requirements

Before installing Astra Trident you should review these general system requirements. Specific backends might have additional requirements.

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.26 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

  Use of non-multipathing configuration or use of `find_multipaths: yes` or `find_multipaths: smart` value in multipath.conf file will result in mount failures. Trident has recommended the use of `find_multipaths: no` since the 21.07 release.

Supported frontends (orchestrators)

Astra Trident supports multiple container engines and orchestrators, including the following:

- Anthos On-Prem (VMware) and Anthos on bare metal 1.9, 1.10, 1.11
- Kubernetes 1.21 - 1.26
- Mirantis Kubernetes Engine 3.5

- OpenShift 4.9 - 4.12

The Trident operator is supported with these releases:

- Anthos On-Prem (VMware) and Anthos on bare metal 1.9, 1.10, 1.11
- Kubernetes 1.21 - 1.26
- OpenShift 4.9 - 4.12

Astra Trident also works with a host of other fully-managed and self-managed Kubernetes offerings, including Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Rancher, and VMWare Tanzu Portfolio.

Supported backends (storage)

To use Astra Trident, you need one or more of the following supported backends:

- Amazon FSx for NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Cloud Volumes Service for GCP
- FAS/AFF/Select 9.5 or later
- NetApp All SAN Array (ASA)
- NetApp HCI/Element software 11 or above

Feature requirements

The table below summarizes the features available with this release of Astra Trident and the versions of Kubernetes it supports.

Feature	Kubernetes version	Feature gates required?
CSI Trident	1.21 - 1.26	No
Volume Snapshots	1.21 - 1.26	No
PVC from Volume Snapshots	1.21 - 1.26	No
iSCSI PV resize	1.21 - 1.26	No
ONTAP Bidirectional CHAP	1.21 - 1.26	No
Dynamic Export Policies	1.21 - 1.26	No
Trident Operator	1.21 - 1.26	No
CSI Topology	1.21 - 1.26	No

Tested host operating systems

Though Astra Trident does not officially support specific operating systems, the following are known to work:

- RedHat CoreOS (RHCOS) versions as supported by OpenShift Container Platform
- RHEL 8+
- Ubuntu 22.04 or later
- Windows Server 2019

By default, Astra Trident runs in a container and will, therefore, run on any Linux worker. However, those workers need to be able to mount the volumes that Astra Trident provides using the standard NFS client or iSCSI initiator, depending on the backends you are using.

The `tridentctl` utility also runs on any of these distributions of Linux.

Host configuration

All worker nodes in the Kubernetes cluster must be able to mount the volumes you have provisioned for your pods. To prepare the worker nodes, you must install NFS or iSCSI tools based on your driver selection.

Prepare the worker node

Storage system configuration

Astra Trident might require changes to a storage system before a backend configuration can use it.

Configure backends

Astra Trident ports

Astra Trident requires access to specific ports for communication.

Astra Trident ports

Container images and corresponding Kubernetes versions

For air-gapped installations, the following list is a reference of container images needed to install Astra Trident. Use the `tridentctl images` command to verify the list of needed container images.

Kubernetes version	Container image
v1.21.0	• netapp/trident:23.01.0 • netapp/trident-autosupport:23.01 • k8s.io/sig-storage/csi-provisioner:v3.4.0 • k8s.io/sig-storage/csi-attacher:v4.1.0 • k8s.io/sig-storage/csi-resizer:v1.7.0 • k8s.io/sig-storage/csi-snapshotter:v6.2.1 • k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • netapp/trident-operator:23.01.0 (optional)
v1.22.0	• netapp/trident:23.01.0 • netapp/trident-autosupport:23.01 • k8s.io/sig-storage/csi-provisioner:v3.4.0 • k8s.io/sig-storage/csi-attacher:v4.1.0 • k8s.io/sig-storage/csi-resizer:v1.7.0 • k8s.io/sig-storage/csi-snapshotter:v6.2.1 • k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • netapp/trident-operator:23.01.0 (optional)
v1.23.0	• netapp/trident:23.01.0 • netapp/trident-autosupport:23.01 • k8s.io/sig-storage/csi-provisioner:v3.4.0 • k8s.io/sig-storage/csi-attacher:v4.1.0 • k8s.io/sig-storage/csi-resizer:v1.7.0 • k8s.io/sig-storage/csi-snapshotter:v6.2.1 • k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • netapp/trident-operator:23.01.0 (optional)
v1.24.0	• netapp/trident:23.01.0 • netapp/trident-autosupport:23.01 • k8s.io/sig-storage/csi-provisioner:v3.4.0 • k8s.io/sig-storage/csi-attacher:v4.1.0 • k8s.io/sig-storage/csi-resizer:v1.7.0 • k8s.io/sig-storage/csi-snapshotter:v6.2.1 • k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • netapp/trident-operator:23.01.0 (optional)

Kubernetes version	Container image
v1.25.0	• netapp/trident:23.01.0 • netapp/trident-autosupport:23.01 • k8s.io/sig-storage/csi-provisioner:v3.4.0 • k8s.io/sig-storage/csi-attacher:v4.1.0 • k8s.io/sig-storage/csi-resizer:v1.7.0 • k8s.io/sig-storage/csi-snapshotter:v6.2.1 • k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • netapp/trident-operator:23.01.0 (optional)
v1.26.0	• netapp/trident:23.01.0 • netapp/trident-autosupport:23.01 • k8s.io/sig-storage/csi-provisioner:v3.4.0 • k8s.io/sig-storage/csi-attacher:v4.1.0 • k8s.io/sig-storage/csi-resizer:v1.7.0 • k8s.io/sig-storage/csi-snapshotter:v6.2.1 • k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • netapp/trident-operator:23.01.0 (optional)

On Kubernetes version 1.21 and above, use the validated `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` image only if the `v1` version is serving the `volumesnapshots.snapshot.storage.k8s.gcr.io` CRD. If the `v1beta1` version is serving the CRD with/without the `v1` version, use the validated `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` image.

Install Astra Trident

Learn about Astra Trident installation

To ensure Astra Trident can be installed in a wide variety of environments and organizations, NetApp offers multiple installation options. You can install Astra Trident using the Trident operator (manually or using Helm) or with `tridentctl`. This topic provides important information for selecting the right installation process for you.

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.26 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

  Use of non-multipathing configuration or use of `find_multipaths: yes` or `find_multipaths: smart` value in multipath.conf file will result in mount failures. Trident has recommended the use of `find_multipaths: no` since the 21.07 release.

Before you begin

Regardless of your installation path, you must have:

- Full privileges to a supported Kubernetes cluster running a supported version of Kubernetes and feature requirements enabled. Review the requirements for details.
- Access to a supported NetApp storage system.
- Capability to mount volumes from all of the Kubernetes worker nodes.
- A Linux host with `kubectl` (or `oc`, if you are using OpenShift) installed and configured to manage the Kubernetes cluster that you want to use.
- The `KUBECONFIG` environment variable set to point to your Kubernetes cluster configuration.
- If you are using Kubernetes with Docker Enterprise, follow their steps to enable CLI access.

If you have not familiarized yourself with the basic concepts, now is a great time to do that.

Choose your installation method

Select the installation method that's right for you. You should also review the considerations for moving between methods before making your decision.

Using the Trident operator

Whether deploying manually or using Helm, the Trident operator is a great way to simplify installation and dynamically manage Astra Trident resources. You can even customize your Trident operator deployment using the attributes in the `TridentOrchestrator` custom resource (CR).

The benefits of using the Trident operator include:

Astra Trident object creation

The Trident operator automatically creates the following objects for your Kubernetes version.

- ServiceAccount for the operator
- ClusterRole and ClusterRoleBinding to the ServiceAccount
- Dedicated PodSecurityPolicy (for Kubernetes 1.25 and earlier)
- The operator itself

Self-healing capability

The operator monitors Astra Trident installation and actively takes measures to address issues, such as when the deployment is deleted or if it is accidentally modified. A `trident-operator-<generated-id>` pod is created that associates a `TridentOrchestrator` CR with an Astra Trident installation. This ensures there is only one instance of Astra Trident in the cluster and controls its setup, making sure the installation is idempotent. When changes are made to the installation (such as, deleting the deployment or node daemonset), the operator identifies them and fixes them individually.

Easy updates to existing installations

You can easily update an existing deployment with the operator. You only need to edit the `TridentOrchestrator` CR to make updates to an installation.

For example, consider a scenario where you need to enable Astra Trident to generate debug logs. To do this, patch your `TridentOrchestrator` to set `spec.debug` to `true`:

```
kubectl patch torc <trident-orchestrator-name> -n trident --type=merge
-p '{"spec":{"debug":true}}'
```

After `TridentOrchestrator` is updated, the operator processes the updates and patches the existing installation. This might triggers the creation of new pods to modify the installation accordingly.

Automatic Kubernetes upgrade handling

When the Kubernetes version of the cluster is upgraded to a supported version, the operator updates an existing Astra Trident installation automatically and changes it to ensure that it meets the requirements of the Kubernetes version.

> If the cluster is upgraded to an unsupported version, the operator prevents installing Astra Trident. If Astra Trident has already been installed with the operator, a warning is displayed to indicate that Astra Trident is installed on an unsupported Kubernetes version.

Kubernetes cluster management using BlueXP (formerly Cloud Manager)

With Astra Trident using BlueXP, you can upgrade to the latest version of Astra Trident, add and manage storage classes and connect them to Working Environments, and back up persistent volumes using Cloud Backup Service. BlueXP supports Astra Trident deployment using the Trident operator, either manually or using Helm.

Using `tridenctl`

If you have an existing deployment that must be upgraded or if you are looking to highly customize your deployment, you should consider installaing using tridentctl. This is the conventional method of deploying Astra Trident.

You can customize your `tridentctl` installation to generate the manifests for Trident resources. This

includes the deployment, daemonset, service account, and the cluster role that Astra Trident creates as part of its installation.

> Beginning with the 22.04 release, AES keys will no longer be regenerated every time Astra Trident is installed. With this release, Astra Trident will install a new secret object that persists across installations. This means, `tridentctl` in 22.04 can uninstall previous versions of Trident, but earlier versions cannot uninstall 22.04 installations.
> Select the appropriate installation *method*.

Choose your installation mode

Determine your deployment process based on the *installation mode* (Standard, Offline, or Remote) required by your organization.

Standard installation

This is the easiest way to install Astra Trident and works for most environments that do not impose network restrictions. Standard installation mode uses default registries to store required Trident (`docker.io`) and CSI (`registry.k8s.io`) images.

When you use standard mode, the Astra Trident installer:

- Fetches the container images over the Internet
- Creates a deployment or node daemonset, which spins up Astra Trident pods on all the eligible nodes in the Kubernetes cluster

Offline installation

Offline installation mode might be required in an air-gapped or secure location. In this scenario, you can create a single private, mirrored registry or two mirrored registries to store required Trident and CSI images.

> Regardless of your registry configuration, CSI images must reside in one registry.

Remote installation

Here is a high-level overview of the remote installation process:

- Deploy the appropriate version of `kubectl` on the remote machine from where you want to deploy Astra Trident.
- Copy the configuration files from the Kubernetes cluster and set the `KUBECONFIG` environment variable on the remote machine.
- Initiate a `kubectl get nodes` command to verify that you can connect to the required Kubernetes cluster.
- Complete the deployment from the remote machine by using the standard installation steps.

Select the process based on your method and mode

After you've made your decisions, select the appropriate process.

Method	Installation mode
Trident operator (manually)	Standard installation Offline installation
Trident operator (Helm)	Standard installation Offline installation
`tridentctl`	Standard or offline installation

Moving between installation methods

You can decide to change your installation method. Before doing so, consider the following:

- Always use the same method for installing and uninstalling Astra Trident. If you have deployed with `tridentctl`, you should use the appropriate version of the `tridentctl` binary to uninstall Astra Trident. Similarly, if you are deploying with the operator, you should edit the `TridentOrchestrator` CR and set `spec.uninstall=true` to uninstall Astra Trident.
- If you have an operator-based deployment that you want to remove and use instead `tridentctl` to deploy Astra Trident, you should first edit `TridentOrchestrator` and set `spec.uninstall=true` to uninstall Astra Trident. Then delete `TridentOrchestrator` and the operator deployment. You can then install using `tridentctl`.
- If you have a manual operator-based deployment, and you want to use Helm-based Trident operator deployment, you should manually uninstall the operator first, and then perform the Helm install. This enables Helm to deploy the Trident operator with the required labels and annotations. If you do not do this, your Helm-based Trident operator deployment will fail with label validation error and annotation validation error. If you have a `tridentctl`-based deployment, you can use Helm-based deployment without running into issues.

Other known configuration options

When installing Astra Trident on VMWare Tanzu Portfolio products:

- The cluster must support privileged workloads.
- The `--kubelet-dir` flag should be set to the location of kubelet directory. By default, this is `/var/vcap/data/kubelet`.

  Specifying the kubelet location using `--kubelet-dir` is known to work for Trident Operator, Helm, and `tridentctl` deployments.

Install using Trident operator

Manually deploy the Trident operator (Standard mode)

You can manually deploy the Trident operator to install Astra Trident. This process applies to installations where the container images required by Astra Trident are not stored in a private registry. If you do have a private image registry, use the process for

offline deployment.

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.26 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

  Use of non-multipathing configuration or use of `find_multipaths: yes` or `find_multipaths: smart` value in multipath.conf file will result in mount failures. Trident has recommended the use of `find_multipaths: no` since the 21.07 release.

Manually deploy the Trident operator and install Trident

Review the installation overview to ensure you've met installation prerequisites and selected the correct installation option for your environment.

Before you begin

Before you begin installation, log in to the Linux host and verify it is managing a working, supported Kubernetes cluster and that you have the necessary privileges.

> With OpenShift, use `oc` instead of `kubectl` in all of the examples that follow, and log in as **system:admin** first by running `oc login -u system:admin` or `oc login -u kube-admin`.

1. Verify your Kubernetes version:

```
kubectl version
```

2. Verify cluster administrator privileges:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verify you can launch a pod that uses an image from Docker Hub and reach your storage system over the pod network:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Step 1: Download the Trident installer package

The Astra Trident installer package contains everything you need to deploy the Trident operator and install Astra Trident. Download and extract the latest version of the Trident installer from the *Assets* section on GitHub.

```
wget https://github.com/NetApp/trident/releases/download/v23.01.0/trident-installer-23.01.0.tar.gz
tar -xf trident-installer-23.01.0.tar.gz
cd trident-installer
```

Step 2: Create the `TridentOrchestrator` CRD

Create the `TridentOrchestrator` Custom Resource Definition (CRD). You will create a `TridentOrchestrator` Custom Resources later. Use the appropriate CRD YAML version in `deploy/crds` to create the `TridentOrchestrator` CRD.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_<version>.yaml
```

Step 3: Deploy the Trident operator

The Astra Trident installer provides a bundle file that can be used to install the operator and create associated objects. The bundle file is an easy way to deploy the operator and install Astra Trident using a default configuration.

- For clusters running Kubernetes 1.24 or lower, use `bundle_pre_1_25.yaml`.
- For clusters running Kubernetes 1.25 or higher, use `bundle_post_1_25.yaml`.

The Trident installer deploys the operator in the `trident` namespace. If the `trident` namespace does not exist, use `kubectl apply -f deploy/namespace.yaml` to create it.

Steps

1. Create the resources and deploy the operator:

   ```
   kubectl create -f deploy/<bundle>.yaml
   ```

   To deploy the operator in a namespace other than the `trident` namespace, update `serviceaccount.yaml`, `clusterrolebinding.yaml` and `operator.yaml` and generate your bundle file using the `kustomization.yaml`:

   ```
   kubectl kustomize deploy/ > deploy/<bundle>.yaml
   ```

2. Verify the operator was deployed.

```
kubectl get deployment -n <operator-namespace>

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

There should only be **one instance** of the operator in a Kubernetes cluster. Do not create multiple deployments of the Trident operator.

Step 4: Create the `TridentOrchestrator` and install Trident

You can now create the `TridentOrchestrator` and install Astra Trident. Optionally, you can customize your Trident installation using the attributes in the `TridentOrchestrator` spec.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:23.01.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.01.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verify the installation

There are several ways to verify your installation.

Using `TridentOrchestrator` status

The status of `TridentOrchestrator` indicates if the installation was successful and displays the version of

Trident installed. During the installation, the status of `TridentOrchestrator` changes from `Installing` to `Installed`. If you observe the `Failed` status and the operator is unable to recover by itself, check the logs.

Status	Description
Installing	The operator is installing Astra Trident using this `TridentOrchestrator` CR.
Installed	Astra Trident has successfully installed.
Uninstalling	The operator is uninstalling Astra Trident, because `spec.uninstall=true`.
Uninstalled	Astra Trident is uninstalled.
Failed	The operator could not install, patch, update or uninstall Astra Trident; the operator will automatically try to recover from this state. If this state persists you will require troubleshooting.
Updating	The operator is updating an existing installation.
Error	The `TridentOrchestrator` is not used. Another one already exists.

Using pod creation status

You can confirm if the Astra Trident installation completed by reviewing the status of the created pods:

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw   6/6     Running   0
1m
trident-node-linux-mr6zc              2/2     Running   0
1m
trident-node-linux-xrp7w              2/2     Running   0
1m
trident-node-linux-zh2jt              2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv     1/1     Running   0
3m
```

Using `tridentctl`

You can use `tridentctl` to check the version of Astra Trident installed.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.0        | 23.01.0        |
+----------------+----------------+
```

What's next

Now you can create create a backend and storage class, provision a volume, and mount the volume in a pod.

Manually deploy the Trident operator (Offline mode)

You can manually deploy the Trident operator to install Astra Trident. This process applies to installations where the container images required by Astra Trident are stored in a private registry. If you do not have a private image registry, use the process for standard deployment.

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.26 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

  Use of non-multipathing configuration or use of `find_multipaths: yes` or `find_multipaths: smart` value in multipath.conf file will result in mount failures. Trident has recommended the use of `find_multipaths: no` since the 21.07 release.

Manually deploy the Trident operator and install Trident

Review the installation overview to ensure you've met installation prerequisites and selected the correct installation option for your environment.

Before you begin

Log in to the Linux host and verify it is managing a working and supported Kubernetes cluster and that you have the necessary privileges.

> With OpenShift, use `oc` instead of `kubectl` in all of the examples that follow, and log in as **system:admin** first by running `oc login -u system:admin` or `oc login -u kube-admin`.

1. Verify your Kubernetes version:

```
kubectl version
```

2. Verify cluster administrator privileges:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verify you can launch a pod that uses an image from Docker Hub and reach your storage system over the pod network:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Step 1: Download the Trident installer package

The Astra Trident installer package contains everything you need to deploy the Trident operator and install Astra Trident. Download and extract the latest version of the Trident installer from the *Assets* section on GitHub.

```
wget https://github.com/NetApp/trident/releases/download/v23.01.0/trident-
installer-23.01.0.tar.gz
tar -xf trident-installer-23.01.0.tar.gz
cd trident-installer
```

Step 2: Create the `TridentOrchestrator` CRD

Create the `TridentOrchestrator` Custom Resource Definition (CRD). You will create a `TridentOrchestrator` Custom Resources later. Use the appropriate CRD YAML version in `deploy/crds` to create the `TridentOrchestrator` CRD:

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_<VERSION>.yaml
```

Step 3: Update the registry location in the operator

In `operator.yaml`, update `image: docker.io/netapp/trident-operator:23.01.0` to reflect the location of your image registry. For example:.

- `image: <your-registry>/trident-operator:23.01.0` if your images are all located in one registry.

- `image: <your-registry>/netapp/trident-operator:23.01.0` if your Trident and CSI images are located in different registries.

Step 3: Deploy the Trident operator

The Trident installer deploys the operator in the `trident` namespace. If the `trident` namespace does not exist, use `kubectl apply -f deploy/namespace.yaml` to create it.

To deploy the operator in a namespace other than the `trident` namespace, update `serviceaccount.yaml`, `clusterrolebinding.yaml` and `operator.yaml` prior to deploying the operator.

1. Create the resources and deploy the operator:

```
kubectl kustomize deploy/ > deploy/<BUNDLE>.yaml
```

The Astra Trident installer provides a bundle file that can be used to install the operator and create associated objects. The bundle file is an easy way to deploy the operator and install Astra Trident using a default configuration.

- For clusters running Kubernetes 1.24 or lower, use `bundle_pre_1_25.yaml`.
- For clusters running Kubernetes 1.25 or higher, use `bundle_post_1_25.yaml`.

2. Verify the operator was deployed.

```
kubectl get deployment -n <operator-namespace>

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

There should only be **one instance** of the operator in a Kubernetes cluster. Do not create multiple deployments of the Trident operator.

Step 4: Update the image registry location in the `TridentOrchestrator`

Your Trident and CSI images can be located in one registry or different registries, but all CSI images must be located in the same registry. Update `deploy/crds/tridentorchestrator_cr.yaml` to add the additional location specs based on your registry configuration.

Trident and CSI images in one registry

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:23.01"
tridentImage: "<your-registry>/trident:23.01.0"
```

Trident and CSI images in different registries

You must append `sig-storage` to the `imageRegistry` to use different registry locations.

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.01"
tridentImage: "<your-registry>/netapp/trident:23.01.0"
```

Step 5: Create the `TridentOrchestrator` and install Trident

You can now create the `TridentOrchestrator` and install Astra Trident. Optionally, you can further customize your Trident installation using the attributes in the `TridentOrchestrator` spec. The following example shows an installation where Trident and CSI images are located in different registries.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/netapp/trident-autosupport:23.01
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.01.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>/sig-storage
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/netapp/trident:23.01.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.01.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verify the installation

There are several ways to verify your installation.

Using `TridentOrchestrator` status

The status of `TridentOrchestrator` indicates if the installation was successful and displays the version of Trident installed. During the installation, the status of `TridentOrchestrator` changes from `Installing` to `Installed`. If you observe the `Failed` status and the operator is unable to recover by itself, check the logs.

Status	Description
Installing	The operator is installing Astra Trident using this `TridentOrchestrator` CR.
Installed	Astra Trident has successfully installed.
Uninstalling	The operator is uninstalling Astra Trident, because `spec.uninstall=true`.
Uninstalled	Astra Trident is uninstalled.
Failed	The operator could not install, patch, update or uninstall Astra Trident; the operator will automatically try to recover from this state. If this state persists you will require troubleshooting.
Updating	The operator is updating an existing installation.
Error	The `TridentOrchestrator` is not used. Another one already exists.

Using pod creation status

You can confirm if the Astra Trident installation completed by reviewing the status of the created pods:

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv          1/1     Running   0
3m
```

Using `tridentctl`

You can use `tridentctl` to check the version of Astra Trident installed.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.0        | 23.01.0        |
+----------------+----------------+
```

What's next

Now you can create create a backend and storage class, provision a volume, and mount the volume in a pod.

Deploy Trident operator using Helm (Standard mode)

You can deploy the Trident operator and install Astra Trident using Helm. This process applies to installations where the container images required by Astra Trident are not stored in a private registry. If you do have a private image registry, use the process for offline deployment.

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.26 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

  Use of non-multipathing configuration or use of `find_multipaths: yes` or `find_multipaths: smart` value in multipath.conf file will result in mount failures. Trident has recommended the use of `find_multipaths: no` since the 21.07 release.

Deploy the Trident operator and install Astra Trident using Helm

Using the Trident Helm Chart you can deploy the Trident operator and install Trident in one step.

Review the installation overview to ensure you've met installation prerequisites and selected the correct installation option for your environment.

Before you begin

In addition to the deployment prerequisites you need Helm version 3.

Steps

1. Add the Astra Trident Helm repository:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Use `helm install` and specify a name for your deployment as in the following example where `23.01.0` is the version of Astra Trident you are installing.

```
helm install <name> netapp-trident/trident-operator --version 23.01.0
--create-namespace --namespace <trident-namespace>
```

> If you already created a namespace for Trident, the `--create-namespace` parameter will not create an additional namespace.

You can use `helm list` to review installation details such as name, namespace, chart, status, app version, and revision number.

Pass configuration data during install

There are two ways to pass configuration data during the install:

Option	Description
`--values` (or `-f`)	Specify a YAML file with overrides. This can be specified multiple times and the rightmost file will take precedence.
`--set`	Specify overrides on the command line.

For example, to change the default value of `debug`, run the following `--set` command where `23.01.0` is the version of Astra Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 23.01.0
--create-namespace --namespace --set tridentDebug=true
```

> The `values.yaml` file, which is part of the Helm chart provides the list of keys and their default values.

What's next

Now you can create create a backend and storage class, provision a volume, and mount the volume in a pod.

Deploy Trident operator using Helm (Offline mode)

You can deploy the Trident operator and install Astra Trident using Helm. This process applies to installations where the container images required by Astra Trident are stored in

a private registry. If you do not have a private image registry, use the process for standard deployment.

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.26 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

  Use of non-multipathing configuration or use of `find_multipaths: yes` or `find_multipaths: smart` value in multipath.conf file will result in mount failures. Trident has recommended the use of `find_multipaths: no` since the 21.07 release.

Deploy the Trident operator and install Astra Trident using Helm

Using the Trident Helm Chart you can deploy the Trident operator and install Trident in one step.

Review the installation overview to ensure you've met installation prerequisites and selected the correct installation option for your environment.

Before you begin

In addition to the deployment prerequisites you need Helm version 3.

Steps

1. Add the Astra Trident Helm repository:

   ```
   helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
   ```

2. Use `helm install` and specify a name for your deployment and image registry location or locations as in the following examples. In the examples, `23.01.0` is the version of Astra Trident you are installing.

Trident and CSI images in one registry

```
helm install <name> netapp-trident/trident-operator --version
23.01.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace>
```

Trident and CSI images in different registries

You must append `sig-storage` to the `imageRegistry` to use different registry locations.

```
helm install <name> netapp-trident/trident-operator --version
23.01.0 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.01.0 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.01 --set tridentImage=<your-
registry>/netapp/trident:23.01.0 --create-namespace --namespace
<trident-namespace>
```

If you already created a namespace for Trident, the `--create-namespace` parameter will not create an additional namespace.

You can use `helm list` to review installation details such as name, namespace, chart, status, app version, and revision number.

Pass configuration data during install

There are two ways to pass configuration data during the install:

Option	Description
`--values` (or `-f`)	Specify a YAML file with overrides. This can be specified multiple times and the rightmost file will take precedence.
`--set`	Specify overrides on the command line.

For example, to change the default value of `debug`, run the following `--set` command where `23.01.0` is the version of Astra Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 23.01.0
--create-namespace --namespace --set tridentDebug=true
```

The `values.yaml` file, which is part of the Helm chart provides the list of keys and their default values.

What's next

Now you can create create a backend and storage class, provision a volume, and mount the volume in a pod.

Customize Trident operator installation

The Trident operator allows you to customize Astra Trident installation using the attributes in the `TridentOrchestrator` spec.

If you want to customize the installation beyond what `TridentOrchestrator` arguments allow, you should consider using `tridentctl` to generate custom YAML manifests that you can modify as needed.

> `spec.namespace` is specified in `TridentOrchestrator` to signify which namespace where Astra Trident is installed. This parameter **cannot be updated after Astra Trident is installed**. Attempting to do so causes the `TridentOrchestrator` status to change to `Failed`. Astra Trident is not intended to be migrated across namespaces.

Configuration options

This table details `TridentOrchestrator` attributes:

Parameter	Description	Default
`namespace`	Namespace to install Astra Trident in	"default"
`debug`	Enable debugging for Astra Trident	false
`windows`	Setting to `true` enables installation on Windows worker nodes.	false
`IPv6`	Install Astra Trident over IPv6	false
`k8sTimeout`	Timeout for Kubernetes operations	30sec
`silenceAutosupport`	Don't send autosupport bundles to NetApp automatically	false
`enableNodePrep`	Manage worker node dependencies automatically (**BETA**)	false
`autosupportImage`	The container image for Autosupport Telemetry	"netapp/trident-autosupport:23.01.0"
`autosupportProxy`	The address/port of a proxy for sending Autosupport Telemetry	"http://proxy.example.com:8888"
`uninstall`	A flag used to uninstall Astra Trident	false
`logFormat`	Astra Trident logging format to be used [text,json]	"text"
`tridentImage`	Astra Trident image to install	"netapp/trident:21.04"

Parameter	Description	Default
`imageRegistry`	Path to internal registry, of the format `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage (k8s 1.19+) or quay.io/k8scsi"
`kubeletDir`	Path to the kubelet directory on the host	“/var/lib/kubelet”
`wipeout`	A list of resources to delete to perform a complete removal of Astra Trident	
`imagePullSecrets`	Secrets to pull images from an internal registry	
`imagePullPolicy`	Sets the image pull policy for the the Trident operator. Valid values are: `Always` to always pull the image. `IfNotPresent` to pull the image only if it does not already exist on the node. `Never` to never pull the image.	`IfNotPresent`
`controllerPluginNodeSelector`	Additional node selectors for pods running the Trident Controller CSI Plugin. Follows same format as pod.spec.nodeSelector.	No default; optional
`controllerPluginTolerations`	Overrides tolerations for pods running the Trident Controller CSI Plugin. Follows the same format as pod.spec.Tolerations.	No default; optional
`nodePluginNodeSelector`	Additional node selectors for pods running the Trident Node CSI Plugin. Follows same format as pod.spec.nodeSelector.	No default; optional
`nodePluginTolerations`	Overrides tolerations for pods running the Trident Node CSI Plugin. Follows the same format as pod.spec.Tolerations.	No default; optional

For more information on formatting pod parameters, see Assigning Pods to Nodes.

Sample configurations

You can use the attributes mentioned above when defining `TridentOrchestrator` to customize your installation.

Example 1: Basic custom configuration

This is an example for a basic custom configuration.

```
cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

Example 2: Deploy with node selectors

This example illustrates how Trident can be deployed with node selectors:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

Example 3: Deploy on Windows worker nodes

This example illustrates deployment on a Windows worker node.

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

Install using tridentctl

Install using `tridentctl`

You can install Astra Trident using `tridentctl`. This process applies to installations where Trident and CSI images are located in a `docker.io` and `registry.k8s.io` or a private, mirrored registry. To customize your `tridentctl` deployment, refer to Customize tridentctl deployment.

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.26 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

  Use of non-multipathing configuration or use of `find_multipaths: yes` or `find_multipaths: smart` value in multipath.conf file will result in mount failures. Trident has recommended the use of `find_multipaths: no` since the 21.07 release.

Install Astra Trident using `tridentctl`

Review the installation overview to ensure you've met installation prerequisites and selected the correct installation option for your environment.

Before you begin

Before you begin installation, log in to the Linux host and verify it is managing a working, supported Kubernetes cluster and that you have the necessary privileges.

With OpenShift, use `oc` instead of `kubectl` in all of the examples that follow, and log in as **system:admin** first by running `oc login -u system:admin` or `oc login -u kube-admin`.

1. Verify your Kubernetes version:

```
kubectl version
```

2. Verify cluster administrator privileges:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verify you can launch a pod that uses an image from Docker Hub and reach your storage system over the pod network:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Step 1: Download the Trident installer package

The Astra Trident installer package creates a Trident pod, configures the CRD objects that are used to maintain its state, and initializes the CSI sidecars to perform actions such as provisioning and attaching volumes to the cluster hosts. Download and extract the latest version of the Trident installer from the *Assets* section on GitHub. Update *<trident-installer-XX.XX.X.tar.gz>* in the example with your selected Astra Trident version.

```
wget https://github.com/NetApp/trident/releases/download/v23.01.0/trident-installer-23.01.0.tar.gz
tar -xf trident-installer-23.01.0.tar.gz
cd trident-installer
```

Step 2: Install Astra Trident

Install Astra Trident in the desired namespace by executing the `tridentctl install` command. You can add additional arguments to specify image registry location.

To enable Astra Trident to run on Windows nodes, add the `--windows` flag to the install command: `$ ./tridentctl install --windows -n trident`.

Standard mode

```
./tridentctl install -n trident
```

Trident and CSI images in one mirrored registry

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:23.01 --trident
-image <your-registry>/trident:23.01.0
```

Trident and CSI images in different mirrored registries

You must append `sig-storage` to the `imageRegistry` to use different registry locations.

```
./tridentctl install -n trident --image-registry <your-registry>/sig-
storage --autosupport-image <your-registry>/netapp/trident-
autosupport:23.01 --trident-image <your-
registry>/netapp/trident:23.01.0
```

Your installation status should look something like this.

```
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=23.01.0
INFO Trident installation succeeded.
....
```

Verify the installation

You can verify your installation using pod creation status or `tridentctl`.

Using pod creation status

You can confirm if the Astra Trident installation completed by reviewing the status of the created pods:

```
kubectl get pods -n trident

NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-679648bd45-cv2mx   6/6     Running   0          5m29s
trident-node-linux-vgc8n              2/2     Running   0          5m29s
```

If the installer does not complete successfully or `trident-controller-<generated id>` (`trident-csi-<generated id>` in versions prior to 23.01) does not have a **Running** status, the platform was not installed. Use `-d` to turn on debug mode and troubleshoot the issue.

Using `tridentctl`

You can use `tridentctl` to check the version of Astra Trident installed.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.0        | 23.01.0        |
+----------------+----------------+
```

What's next

Now you can create create a backend and storage class, provision a volume, and mount the volume in a pod.

Customize `tridentctl` installation

You can use the Astra Trident installer to customize installation.

Learn about the installer

The Astra Trident installer enables you to customize attributes. For example, if you have copied the Trident image to a private repository, you can specify the image name by using `--trident-image`. If you have copied the Trident image as well as the needed CSI sidecar images to a private repository, it might be preferable to specify the location of that repository by using the `--image-registry` switch, which takes the form `<registry FQDN>[:port]`.

If you are using a distribution of Kubernetes, where `kubelet` keeps its data on a path other than the usual `/var/lib/kubelet`, you can specify the alternate path by using `--kubelet-dir`.

If you need to customize the installation beyond what the installer's arguments allow, you can also customize the deployment files. Using the `--generate-custom-yaml` parameter creates the following YAML files in the installer's `setup` directory:

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`
- `trident-resourcequota.yaml`

After you have generated these files, you can modify them according to your needs and then use `--use-custom-yaml` to install your custom deployment.

```
./tridentctl install -n trident --use-custom-yaml
```

What's next?

After you install Astra Trident, you can proceed with creating a backend, creating a storage class, provisioning a volume, and mounting the volume in a pod.

Step 1: Create a backend

You can now go ahead and create a backend that will be used by Astra Trident to provision volumes. To do this, create a `backend.json` file that contains the necessary parameters. Sample configuration files for different backend types can be found in the `sample-input` directory.

See here for more details about how to configure the file for your backend type.

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

```
./tridentctl -n trident create backend -f backend.json
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+
```

If the creation fails, something was wrong with the backend configuration. You can view the logs to determine the cause by running the following command:

```
./tridentctl -n trident logs
```

After you address the problem, simply go back to the beginning of this step and try again. For more troubleshooting tips, see the troubleshooting section.

Step 2: Create a storage class

Kubernetes users provision volumes by using persistent volume claims (PVCs) that specify a storage class by name. The details are hidden from the users, but a storage class identifies the provisioner that is used for that class (in this case, Trident), and what that class means to the provisioner.

Create a storage class Kubernetes users will specify when they want a volume. The configuration of the class needs to model the backend that you created in the previous step, so that Astra Trident will use it to provision new volumes.

The simplest storage class to start with is one based on the `sample-input/storage-class-csi.yaml.templ` file that comes with the installer, replacing *`BACKEND_TYPE`* with the storage driver name.

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

This is a Kubernetes object, so you use `kubectl` to create it in Kubernetes.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

You should now see a **basic-csi** storage class in both Kubernetes and Astra Trident, and Astra Trident should have discovered the pools on the backend.

```
kubectl get sc basic-csi
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Step 3: Provision your first volume

Now you are ready to dynamically provision your first volume. This is done by creating a Kubernetes persistent volume claim (PVC) object.

Create a PVC for a volume that uses the storage class that you just created.

See `sample-input/pvc-basic-csi.yaml` for an example. Make sure the storage class name matches the one that you created.

```
kubectl create -f sample-input/pvc-basic-csi.yaml

kubectl get pvc --watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic          1s
basic     Pending   pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   0
basic          5s
basic     Bound     pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   1Gi
RWO           basic          7s
```

Step 4: Mount the volumes in a pod

Now let us mount the volume. We will launch an nginx pod that mounts the PV under `/usr/share/nginx/html`.

```
cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

At this point, the pod (application) no longer exists but the volume is still there. You can use it from another pod if you want to.

To delete the volume, delete the claim:

```
kubectl delete pvc basic
```

You can now do additional tasks, such as the following:

- Configure additional backends.
- Create additional storage classes.

Copyright information

Trademark information