# Table of Contents

- Install Astra Trident ................................................................. 1
  - Learn about Astra Trident installation .......................................... 1
  - Install using Trident operator .................................................... 5
  - Install using tridentctl .............................................................. 30
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 24.02

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.27 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

Resource accountability

The cluster-scoped Trident operator manages resources associated with an Astra Trident installation at the cluster level. This mitigates errors that might be caused when maintaining cluster-scoped resources using a namespace-scoped operator. This is essential for self-healing and patching.

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 trigger the creation of new pods to modify the installation accordingly.
Clean reinstallation

The cluster-scoped Trident operator enables clean removal of cluster-scoped resources. Users can completely uninstall Astra Trident and easily reinstall.

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 tridentctl

If you have an existing deployment that must be upgraded or if you are looking to highly customize your deployment, you should consider installing 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Installation mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trident operator (manually)</td>
<td>Standard installation&lt;br&gt;Offline installation</td>
</tr>
<tr>
<td>Trident operator (Helm)</td>
<td>Standard installation&lt;br&gt;Offline installation</td>
</tr>
<tr>
<td>tridentctl</td>
<td>Standard or offline installation</td>
</tr>
</tbody>
</table>
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 24.02

You must read the following critical information about Astra Trident.
Critical information about Astra Trident

- Kubernetes 1.27 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:

   ```bash
   kubectl version
   ```

2. Verify cluster administrator privileges:

   ```bash
   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:

   ```bash
   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/v24.02.0/trident-installer-24.02.0.tar.gz

tar -xf trident-installer-24.02.0.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_post1.16.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 earlier, use `bundle_pre_1_25.yaml`.
- For clusters running Kubernetes 1.25 or later, use `bundle_post_1_25.yaml`.

Before you begin

- By default, the Trident installer deploys the operator in the `trident` namespace. If the `trident` namespace does not exist, create it using:

```
kubectl apply -f deploy/namespace.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`.

  1. Create the `kustomization.yaml` using the following command where `<bundle.yaml>` is `bundle_pre_1_25.yaml` or `bundle_post_1_25.yaml` based on your Kubernetes version.

```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
```

  2. Compile the bundle using using the following command where `<bundle.yaml>` is `bundle_pre_1_25.yaml` or `bundle_post_1_25.yaml` based on your Kubernetes version.

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

1. Create the resources and deploy the operator:

   kubectl create -f deploy/<bundle.yaml>

2. Verify the operator, deployment, and replicaset were created.

   kubectl get all -n <operator-namespace>

   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:24.02
    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:24.02.0
    Message: Trident installed
    Namespace: trident

Status: Installed
Version: v24.02.0

Events:
  Type Reason Age From Message
  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.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing</td>
<td>The operator is installing Astra Trident using this TridentOrchestrator CR.</td>
</tr>
<tr>
<td>Installed</td>
<td>Astra Trident has successfully installed.</td>
</tr>
<tr>
<td>Uninstalling</td>
<td>The operator is uninstalling Astra Trident, because spec.uninstall=true.</td>
</tr>
<tr>
<td>Uninstalled</td>
<td>Astra Trident is uninstalled.</td>
</tr>
<tr>
<td>Failed</td>
<td>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.</td>
</tr>
<tr>
<td>Updating</td>
<td>The operator is updating an existing installation.</td>
</tr>
<tr>
<td>Error</td>
<td>The TridentOrchestrator is not used. Another one already exists.</td>
</tr>
</tbody>
</table>

**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
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>trident-controller-7d466bf5c7-v4cpw</td>
<td>6/6</td>
<td>Running</td>
<td>0</td>
</tr>
<tr>
<td>1m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trident-node-linux-mr6zc</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
</tr>
<tr>
<td>1m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trident-node-linux-xrp7w</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
</tr>
<tr>
<td>1m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trident-node-linux-zh2jt</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
</tr>
<tr>
<td>1m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trident-operator-766f7b8658-1dzsv</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
</tr>
<tr>
<td>3m</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Using tridentctl**

You can use tridentctl to check the version of Astra Trident installed.
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.02.0        | 24.02.0        |
+----------------+----------------+

**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 24.02**

You must read the following critical information about Astra Trident.

**Critical information about Astra Trident**

- Kubernetes 1.27 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 -- \<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/v24.02.0/trident-installer-24.02.0.tar.gz
   tar -xf trident-installer-24.02.0.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/<VERSION>.yaml

**Step 3: Update the registry location in the operator**

In /deploy/operator.yaml, update image: docker.io/netapp/trident-operator:24.02.0 to reflect the location of your image registry. 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. For example:

   - image: <your-registry>/trident-operator:24.02.0 if your images are all located in one registry.
   - image: <your-registry>/netapp/trident-operator:24.02.0 if your Trident image is located in...
Step 4: 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 earlier, use `bundle_pre_1_25.yaml`.
- For clusters running Kubernetes 1.25 or later, use `bundle_post_1_25.yaml`.

Before you begin

- By default, the Trident installer deploys the operator in the `trident` namespace. If the `trident` namespace does not exist, create it using:

  ```shell
  kubectl apply -f deploy/namespace.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`.

  1. Create the `kustomization.yaml` using the following command where `<bundle.yaml>` is `bundle_pre_1_25.yaml` or `bundle_post_1_25.yaml` based on your Kubernetes version.

  ```shell
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  2. Compile the bundle using the following command where `<bundle.yaml>` is `bundle_pre_1_25.yaml` or `bundle_post_1_25.yaml` based on your Kubernetes version.

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

Steps

1. Create the resources and deploy the operator:

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

2. Verify the operator, deployment, and replicaset were created.

   ```shell
   kubectl get all -n <operator-namespace>
   ```

There should only be one instance of the operator in a Kubernetes cluster. Do not create multiple deployments of the Trident operator.
Step 5: 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.

Images in one registry

```yaml
imageRegistry: "<your-registry>">
autosupportImage: "<your-registry>/trident-autosupport:24.02"
tridentImage: "<your-registry>/trident:24.02.0"
```

Images in different registries

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

```yaml
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:24.02"
tridentImage: "<your-registry>/netapp/trident:24.02.0"
```

Step 6: 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:24.02
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:24.02.0

Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:      
    Autosupport Image:         <your-registry>/netapp/trident-autosupport:24.02
  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:24.02.0
  Message:              Trident installed
  Namespace:            trident
  Status:               Installed
  Version:              v24.02.0

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.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing</td>
<td>The operator is installing Astra Trident using this TridentOrchestrator CR.</td>
</tr>
<tr>
<td>Installed</td>
<td>Astra Trident has successfully installed.</td>
</tr>
<tr>
<td>Uninstalling</td>
<td>The operator is uninstalling Astra Trident, because spec.uninstall=true.</td>
</tr>
<tr>
<td>Uninstalled</td>
<td>Astra Trident is uninstalled.</td>
</tr>
<tr>
<td>Failed</td>
<td>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.</td>
</tr>
<tr>
<td>Updating</td>
<td>The operator is updating an existing installation.</td>
</tr>
<tr>
<td>Error</td>
<td>The <strong>TridentOrchestrator</strong> is not used. Another one already exists.</td>
</tr>
</tbody>
</table>

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
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>trident-controller-7d466bf5c7-v4cpw</td>
<td>6/6</td>
<td>Running</td>
<td>0</td>
</tr>
<tr>
<td>trident-node-linux-mr6zc</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
</tr>
<tr>
<td>trident-node-linux-xrp7w</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
</tr>
<tr>
<td>trident-node-linux-zh2jt</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
</tr>
<tr>
<td>trident-operator-766f7b8658-1dzsv</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
</tr>
</tbody>
</table>

16
Using `tridentctl`

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

```
./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.02.0        | 24.02.0        |
+----------------+----------------+
```

**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 24.02**

You must read the following critical information about Astra Trident.

**Critical information about Astra Trident**

- Kubernetes 1.27 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 24.02.0 is the version of Astra Trident you are installing.

```
helm install <name> netapp-trident/trident-operator --version 24.02.0 --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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--values</code> (or <code>-f</code>)</td>
<td>Specify a YAML file with overrides. This can be specified multiple times and the rightmost file will take precedence.</td>
</tr>
<tr>
<td><code>--set</code></td>
<td>Specify overrides on the command line.</td>
</tr>
</tbody>
</table>

For example, to change the default value of `debug`, run the following `--set` command where 24.02.0 is the version of Astra Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 24.02.0 --create-namespace --namespace trident --set tridentDebug=true
```

**Configuration options**

This table and the `values.yaml` file, which is part of the Helm chart, provide the list of keys and their default values.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodeSelector</td>
<td>Node labels for pod assignment</td>
<td></td>
</tr>
<tr>
<td>podAnnotations</td>
<td>Pod annotations</td>
<td></td>
</tr>
<tr>
<td>deploymentAnnotations</td>
<td>Deployment annotations</td>
<td></td>
</tr>
<tr>
<td>tolerations</td>
<td>Tolerations for pod assignment</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>affinity</td>
<td>Affinity for pod assignment</td>
<td></td>
</tr>
<tr>
<td>tridentControllerPluginNodeSelector</td>
<td>Additional node selectors for pods. Refer to Understanding controller pods and node pods for details.</td>
<td></td>
</tr>
<tr>
<td>tridentControllerPluginTolerations</td>
<td>Overrides Kubernetes tolerations for pods. Refer to Understanding controller pods and node pods for details.</td>
<td></td>
</tr>
<tr>
<td>tridentNodePluginNodeSelector</td>
<td>Additional node selectors for pods. Refer to Understanding controller pods and node pods for details.</td>
<td></td>
</tr>
<tr>
<td>tridentNodePluginTolerations</td>
<td>Overrides Kubernetes tolerations for pods. Refer to Understanding controller pods and node pods for details.</td>
<td></td>
</tr>
<tr>
<td>imageRegistry</td>
<td>Identifies the registry for the trident-operator, trident, and other images. Leave empty to accept the default.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>imagePullPolicy</td>
<td>Sets the image pull policy for the trident-operator.</td>
<td>IfNotPresent</td>
</tr>
<tr>
<td>imagePullSecrets</td>
<td>Sets the image pull secrets for the trident-operator, trident, and other images.</td>
<td></td>
</tr>
<tr>
<td>kubeletDir</td>
<td>Allows overriding the host location of kubelet's internal state.</td>
<td>&quot;/var/lib/kubelet&quot;</td>
</tr>
<tr>
<td>operatorLogLevel</td>
<td>Allows the log level of the Trident operator to be set to: trace, debug, info, warn, error, or fatal.</td>
<td>&quot;info&quot;</td>
</tr>
<tr>
<td>operatorDebug</td>
<td>Allows the log level of the Trident operator to be set to debug.</td>
<td>true</td>
</tr>
<tr>
<td>operatorImage</td>
<td>Allows the complete override of the image for trident-operator.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>operatorImageTag</td>
<td>Allows overriding the tag of the trident-operator image.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>tridentIPv6</td>
<td>Allows enabling Astra Trident to work in IPv6 clusters.</td>
<td>false</td>
</tr>
<tr>
<td>tridentK8sTimeout</td>
<td>Overrides the default 30-second timeout for most Kubernetes API operations (if non-zero, in seconds).</td>
<td>0</td>
</tr>
<tr>
<td>tridentHttpRequestTimeout</td>
<td>Overrides the default 90-second timeout for the HTTP requests, with 0s being an infinite duration for the timeout. Negative values are not allowed.</td>
<td>&quot;90s&quot;</td>
</tr>
<tr>
<td>tridentSilenceAutoSupport</td>
<td>Allows disabling Astra Trident periodic AutoSupport reporting.</td>
<td>false</td>
</tr>
<tr>
<td>tridentAutoSupportImageTag</td>
<td>Allows overriding the tag of the image for Astra Trident AutoSupport container.</td>
<td>&lt;version&gt;</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>tridentAutosupportProxy</td>
<td>Allows Astra Trident AutoSupport container to phone home via an HTTP proxy.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>tridentLogFormat</td>
<td>Sets the Astra Trident logging format (text or json).</td>
<td>&quot;text&quot;</td>
</tr>
<tr>
<td>tridentDisableAuditLog</td>
<td>Disables Astra Trident audit logger.</td>
<td>true</td>
</tr>
<tr>
<td>tridentLogLevel</td>
<td>Allows the log level of Astra Trident to be set to: trace, debug, info, warn, error, or fatal.</td>
<td>&quot;info&quot;</td>
</tr>
<tr>
<td>tridentDebug</td>
<td>Allows the log level of Astra Trident to be set to debug.</td>
<td>false</td>
</tr>
<tr>
<td>tridentLogWorkflows</td>
<td>Allows specific Astra Trident workflows to be enabled for trace logging or log suppression.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>tridentLogLayers</td>
<td>Allows specific Astra Trident layers to be enabled for trace logging or log suppression.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>tridentImage</td>
<td>Allows the complete override of the image for Astra Trident.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>tridentImageTag</td>
<td>Allows overriding the tag of the image for Astra Trident.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>tridentProbePort</td>
<td>Allows overriding the default port used for Kubernetes liveness/readiness probes.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>windows</td>
<td>Allows Astra Trident to be installed on Windows worker node.</td>
<td>false</td>
</tr>
<tr>
<td>enableForceDetach</td>
<td>Allows enabling the force detach feature.</td>
<td>false</td>
</tr>
<tr>
<td>excludePodSecurityPolicy</td>
<td>Excludes the operator pod security policy from creation.</td>
<td>false</td>
</tr>
<tr>
<td>cloudProvider</td>
<td>Set to &quot;Azure&quot; when using managed identities or a cloud identity on an AKS cluster. Set to &quot;AWS&quot; when using a cloud identity on an EKS cluster.</td>
<td>&quot;&quot;</td>
</tr>
</tbody>
</table>

Understanding controller pods and node pods

Astra Trident runs as a single controller pod, plus a node pod on each worker node in the cluster. The node pod must be running on any host where you want to potentially mount an Astra Trident volume.

Kubernetes node selectors and tolerations and taints are used to constrain a pod to run on a specific or preferred node. Using the `ControllerPlugin` and `NodePlugin`, you can specify constraints and overrides.
• The controller plugin handles volume provisioning and management, such as snapshots and resizing.
• The node plugin handles attaching the storage to the node.

**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 24.02**

You must read the following critical information about Astra Trident.

**Critical information about Astra Trident**

- Kubernetes 1.27 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:

   ```bash
   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. 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. In the examples, `24.02.0` is the version of Astra Trident you are installing.
Images in one registry

```
helm install <name> netapp-trident/trident-operator --version 24.02.0 --set imageRegistry=<your-registry> --create-namespace --namespace <trident-namespace>
```

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 24.02.0 --set imageRegistry=<your-registry>/sig-storage --set operatorImage=<your-registry>/netapp/trident-operator:24.02.0 --set tridentAutosupportImage=<your-registry>/netapp/trident-autosupport:24.02.0 --set tridentImage=<your-registry>/netapp/trident:24.02.0 --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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--values</code> (or <code>-f</code>)</td>
<td>Specify a YAML file with overrides. This can be specified multiple times and the rightmost file will take precedence.</td>
</tr>
<tr>
<td><code>--set</code></td>
<td>Specify overrides on the command line.</td>
</tr>
</tbody>
</table>

For example, to change the default value of `debug`, run the following `--set` command where `24.02.0` is the version of Astra Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 24.02.0 --create-namespace --namespace trident --set tridentDebug=true
```
## Configuration options

This table and the values.yaml file, which is part of the Helm chart, provide the list of keys and their default values.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodeSelector</td>
<td>Node labels for pod assignment</td>
<td></td>
</tr>
<tr>
<td>podAnnotations</td>
<td>Pod annotations</td>
<td></td>
</tr>
<tr>
<td>deploymentAnnotations</td>
<td>Deployment annotations</td>
<td></td>
</tr>
<tr>
<td>tolerations</td>
<td>Tolerations for pod assignment</td>
<td></td>
</tr>
<tr>
<td>affinity</td>
<td>Affinity for pod assignment</td>
<td></td>
</tr>
<tr>
<td>tridentControllerPluginNodeSelector</td>
<td>Additional node selectors for pods. Refer to Understanding controller pods and node pods for details.</td>
<td></td>
</tr>
<tr>
<td>tridentControllerPluginTolerations</td>
<td>Overrides Kubernetes tolerations for pods. Refer to Understanding controller pods and node pods for details.</td>
<td></td>
</tr>
<tr>
<td>tridentNodePluginNodeSelector</td>
<td>Additional node selectors for pods. Refer to Understanding controller pods and node pods for details.</td>
<td></td>
</tr>
<tr>
<td>tridentNodePluginTolerations</td>
<td>Overrides Kubernetes tolerations for pods. Refer to Understanding controller pods and node pods for details.</td>
<td></td>
</tr>
<tr>
<td>imageRegistry</td>
<td>Identifies the registry for the trident-operator, trident, and other images. Leave empty to accept the default.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>imagePullPolicy</td>
<td>Sets the image pull policy for the trident-operator.</td>
<td>IfNotPresent</td>
</tr>
<tr>
<td>imagePullSecrets</td>
<td>Sets the image pull secrets for the trident-operator, trident, and other images.</td>
<td></td>
</tr>
<tr>
<td>kubeletDir</td>
<td>Allows overriding the host location of kubelet's internal state.</td>
<td>&quot;/var/lib/kubelet&quot;</td>
</tr>
<tr>
<td>operatorLogLevel</td>
<td>Allows the log level of the Trident operator to be set to: trace, debug, info, warn, error, or fatal.</td>
<td>&quot;info&quot;</td>
</tr>
<tr>
<td>operatorDebug</td>
<td>Allows the log level of the Trident operator to be set to debug.</td>
<td>true</td>
</tr>
<tr>
<td>operatorImage</td>
<td>Allows the complete override of the image for trident-operator.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td><code>operatorImageTag</code></td>
<td>Allows overriding the tag of the trident-operator image.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>tridentIPv6</code></td>
<td>Allows enabling Astra Trident to work in IPv6 clusters.</td>
<td>false</td>
</tr>
<tr>
<td><code>tridentK8sTimeout</code></td>
<td>Overrides the default 30-second timeout for most Kubernetes API operations (if non-zero, in seconds).</td>
<td>0</td>
</tr>
<tr>
<td><code>tridentHttpRequestTimeout</code></td>
<td>Overrides the default 90-second timeout for the HTTP requests, with 0s being an infinite duration for the timeout. Negative values are not allowed.</td>
<td>&quot;90s&quot;</td>
</tr>
<tr>
<td><code>tridentSilenceAutosupport</code></td>
<td>Allows disabling Astra Trident periodic AutoSupport reporting.</td>
<td>false</td>
</tr>
<tr>
<td><code>tridentAutosupportImageTag</code></td>
<td>Allows overriding the tag of the image for Astra Trident AutoSupport container.</td>
<td>&lt;version&gt;</td>
</tr>
<tr>
<td><code>tridentAutosupportProxy</code></td>
<td>Allows Astra Trident AutoSupport container to phone home via an HTTP proxy.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>tridentLogFormat</code></td>
<td>Sets the Astra Trident logging format (text or json).</td>
<td>&quot;text&quot;</td>
</tr>
<tr>
<td><code>tridentDisableAuditLog</code></td>
<td>Disables Astra Trident audit logger.</td>
<td>true</td>
</tr>
<tr>
<td><code>tridentLogLevel</code></td>
<td>Allows the log level of Astra Trident to be set to: trace, debug, info, warn, error, or fatal.</td>
<td>&quot;info&quot;</td>
</tr>
<tr>
<td><code>tridentDebug</code></td>
<td>Allows the log level of Astra Trident to be set to debug.</td>
<td>false</td>
</tr>
<tr>
<td><code>tridentLogWorkflows</code></td>
<td>Allows specific Astra Trident workflows to be enabled for trace logging or log suppression.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>tridentLogLayers</code></td>
<td>Allows specific Astra Trident layers to be enabled for trace logging or log suppression.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>tridentImage</code></td>
<td>Allows the complete override of the image for Astra Trident.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>tridentImageTag</code></td>
<td>Allows overriding the tag of the image for Astra Trident.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>tridentProbePort</code></td>
<td>Allows overriding the default port used for Kubernetes liveness/readiness probes.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>windows</td>
<td>Allows Astra Trident to be installed on Windows worker node.</td>
<td>false</td>
</tr>
<tr>
<td>enableForceDetach</td>
<td>Allows enabling the force detach feature.</td>
<td>false</td>
</tr>
<tr>
<td>excludePodSecurityPolicy</td>
<td>Excludes the operator pod security policy from creation.</td>
<td>false</td>
</tr>
</tbody>
</table>

**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, consider using `tridentctl` to generate custom YAML manifests to modify as needed.

**Understanding controller pods and node pods**

Astra Trident runs as a single controller pod, plus a node pod on each worker node in the cluster. The node pod must be running on any host where you want to potentially mount an Astra Trident volume.

Kubernetes node selectors and tolerations and taints are used to constrain a pod to run on a specific or preferred node. Using the 'ControllerPlugin' and NodePlugin, you can specify constraints and overrides.

- The controller plugin handles volume provisioning and management, such as snapshots and resizing.
- The node plugin handles attaching the storage to the node.

**Configuration options**

spec.namespace is specified in TridentOrchestrator to signify the 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.

This table details TridentOrchestrator attributes.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>Namespace to install Astra Trident in</td>
<td>&quot;default&quot;</td>
</tr>
<tr>
<td>debug</td>
<td>Enable debugging for Astra Trident</td>
<td>false</td>
</tr>
<tr>
<td>enableForceDetach</td>
<td>ontap-san and ontap-san-economy only.</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>Works with Kubernetes Non-Graceful Node Shutdown (NGNS) to grant cluster administrators ability to safely migrate workloads with mounted volumes to new nodes should a node become unhealthy.</td>
<td></td>
</tr>
<tr>
<td>windows</td>
<td>Setting to true enables installation on Windows worker nodes.</td>
<td>false</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>cloudProvider</td>
<td>Set to &quot;Azure&quot; when using managed identities or a cloud identity on an AKS cluster. Set to &quot;AWS&quot; when using a cloud identity on an EKS cluster.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>IPv6</td>
<td>Install Astra Trident over IPv6</td>
<td>false</td>
</tr>
<tr>
<td>k8sTimeout</td>
<td>Timeout for Kubernetes operations</td>
<td>30sec</td>
</tr>
<tr>
<td>silenceAutosupport</td>
<td>Don’t send autosupport bundles to NetApp automatically</td>
<td>false</td>
</tr>
<tr>
<td>autosupportImage</td>
<td>The container image for Autosupport Telemetry</td>
<td>&quot;netapp/trident-autosupport:24.02&quot;</td>
</tr>
<tr>
<td>autosupportProxy</td>
<td>The address/port of a proxy for sending Autosupport Telemetry</td>
<td>&quot;<a href="http://proxy.example.com:8888">http://proxy.example.com:8888</a>&quot;</td>
</tr>
<tr>
<td>uninstall</td>
<td>A flag used to uninstall Astra Trident</td>
<td>false</td>
</tr>
<tr>
<td>logFormat</td>
<td>Astra Trident logging format to be used [text,json]</td>
<td>&quot;text&quot;</td>
</tr>
<tr>
<td>tridentImage</td>
<td>Astra Trident image to install</td>
<td>&quot;netapp/trident:24.02&quot;</td>
</tr>
<tr>
<td>imageRegistry</td>
<td>Path to internal registry, of the format &lt;registry FQDN&gt;[:port][/subpath]</td>
<td>&quot;k8s.gcr.io/sig-storage&quot; (Kubernetes 1.19+) or &quot;quay.io/k8scsi&quot;</td>
</tr>
<tr>
<td>kubeletDir</td>
<td>Path to the kubelet directory on the host</td>
<td>&quot;/var/lib/kubelet&quot;</td>
</tr>
<tr>
<td>wipeout</td>
<td>A list of resources to delete to perform a complete removal of Astra Trident</td>
<td></td>
</tr>
<tr>
<td>imagePullSecrets</td>
<td>Secrets to pull images from an internal registry</td>
<td></td>
</tr>
<tr>
<td>imagePullPolicy</td>
<td>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.</td>
<td>IfNotPresent</td>
</tr>
<tr>
<td>controllerPluginNodeSelector</td>
<td>Additional node selectors for pods. Follows same format as pod.spec.nodeSelector.</td>
<td>No default; optional</td>
</tr>
</tbody>
</table>
For more information on formatting pod parameters, refer to Assigning Pods to Nodes.

### Details about force detach

Force detach is available for ontap-san and ontap-san-economy only. Before enabling force detach, non-graceful node shutdown (NGNS) must be enabled on the Kubernetes cluster. For more information, refer to Kubernetes: Non Graceful node shutdown.

Because Astra Trident relies on Kubernetes NGNS, do not remove out-of-service taints from an unhealthy node until all non-tolerable workloads are rescheduled. Recklessly applying or removing the taint can jeopardize backend data protection.

When the Kubernetes cluster administrator has applied the `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` taint to the node and `enableForceDetach` is set to true, Astra Trident will determine the node status and:

1. Cease backend I/O access for volumes mounted to that node.
2. Mark the Astra Trident node object as dirty (not safe for new publications).

   The Trident controller will reject new publish volume requests until the node is re-qualified (after having been marked as dirty) by the Trident node pod. Any workloads scheduled with a mounted PVC (even after the cluster node is healthy and ready) will not be accepted until Astra Trident can verify the node clean (safe for new publications).

When node health is restored and the taint is removed, Astra Trident will:

1. Identify and clean stale published paths on the node.
2. If the node is in a cleanable state (the out-of-service taint has been removed and the node is in Ready state) and all stale, published paths are clean, Astra Trident will readmit the node as clean and allow new published volumes to the node.

### Sample configurations

You can use the attributes in Configuration options when defining TridentOrchestrator to customize your installation.
**Basic custom configuration**

This is an example for a basic custom installation.

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

**Node selectors**

This example installs Astra Trident with node selectors.

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

This example installs Astra Trident 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
```

Managed identities on an AKS cluster

This example installs Astra Trident to enable managed identities on an AKS cluster.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
```

Cloud identity on an AKS cluster

This example installs Astra Trident for use with a cloud identity on an AKS cluster.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
  xxxx-xxxx-xxxxxxxxxxx'
```
Cloud identity on an EKS cluster

This example installs Astra Trident for use with a cloud identity on an AKS cluster.

```yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
    arn:aws:iam::123456:role/astratrident-role'"
```

Install using tridentctl

You can install Astra Trident using tridentctl. This process applies to installations where the container images required by Astra Trident are stored either in a private registry or not. To customize your tridentctl deployment, refer to Customize tridentctl deployment.

Critical information about Astra Trident 24.02

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.27 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:

   ```shell
kubectl version
   ```

2. Verify cluster administrator privileges:

   ```shell
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:

   ```shell
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.

```bash
wget https://github.com/NetApp/trident/releases/download/v24.02.0/trident-installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.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.

```bash
```

31
Standard mode

./tridentctl install -n trident

Images in one registry

./tridentctl install -n trident --image-registry <your-registry> --autosupport-image <your-registry>/trident-autosupport:24.02 --trident-image <your-registry>/trident:24.02.0

Images in different 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:24.02 --trident-image <your-registry>/netapp/trident:24.02.0

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=24.02.0
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:

```bash
cubectl get pods -n trident
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>trident-controller-679648bd45-cv2mx</td>
<td>6/6</td>
<td>Running</td>
<td>0</td>
<td>5m29s</td>
</tr>
<tr>
<td>trident-node-linux-vgc8n</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>5m29s</td>
</tr>
</tbody>
</table>

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.

```bash
./tridentctl -n trident version
```

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.02.0        | 24.02.0        |
+----------------+----------------+

Sample configurations

The following examples provide sample configurations for installing Astra Trident using tridentctl.

Windows nodes

To enable Astra Trident to run on Windows nodes:

```bash
tridentctl install --windows -n trident
```

Force detach

For more information about force detach, refer to Customize Trident operator installation.

```bash
tridentctl install --enable-force-detach=true -n trident
```
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