



Install Trident

Trident

NetApp
February 20, 2025

Table of Contents

- Install Trident 1
 - Learn about Trident installation 1
 - Install using Trident operator 5
 - Install using tridentctl 34

Install Trident

Learn about Trident installation

To ensure Trident can be installed in a wide variety of environments and organizations, NetApp offers multiple installation options. You can install 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 Trident 24.10

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.32 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- 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 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:

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 a 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 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 a Trident installation. This ensures there is only one instance of 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 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 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 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 Trident. If Trident has already been installed with the operator, a warning is displayed to indicate that Trident is installed on an unsupported Kubernetes version.

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 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 Trident creates as part of its installation.



Beginning with the 22.04 release, AES keys will no longer be regenerated every time Trident is installed. With this release, 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 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 Trident installer:

- Fetches the container images over the Internet
- Creates a deployment or node daemonset, which spins up 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 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
<code>tridentctl</code>	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 Trident. If you have deployed with `tridentctl`, you should use the appropriate version of the `tridentctl` binary to uninstall Trident. Similarly, if you are deploying with the operator, you should edit the `TridentOrchestrator` CR and set `spec.uninstall=true` to uninstall Trident.
- If you have an operator-based deployment that you want to remove and use instead `tridentctl` to deploy Trident, you should first edit `TridentOrchestrator` and set `spec.uninstall=true` to uninstall 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 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 Trident. This process applies to installations where the container images required by 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 Trident 24.10

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.32 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- 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 Trident installer package contains everything you need to deploy the Trident operator and install 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/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

Step 2: Create the `TridentOrchestrator` CRD

Create the `TridentOrchestrator` Custom Resource Definition (CRD). You create a `TridentOrchestrator` Custom Resource 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 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 Trident using a default configuration.

- For clusters running Kubernetes 1.24, 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 Trident. Optionally, you can [customize your Trident installation](#) using the attributes in the `TridentOrchestrator` spec.

```

kubect1 create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubect1 describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Debug:       true
  Namespace:   trident
  nodePrep:
    - iscsi
Status:
  Current Installation Params:
    IPv6:                false
    Autosupport Hostname:
    Autosupport Image:   netapp/trident-autosupport:24.10
    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:24.10.0
  Message:              Trident installed Namespace:
trident
  Status:               Installed
  Version:              v24.10.0
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](#).

Status	Description
Installing	The operator is installing Trident using this <code>TridentOrchestrator</code> CR.
Installed	Trident has successfully installed.
Uninstalling	The operator is uninstalling Trident, because <code>spec.uninstall=true</code> .
Uninstalled	Trident is uninstalled.
Failed	The operator could not install, patch, update or uninstall 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 <code>TridentOrchestrator</code> is not used. Another one already exists.

Using pod creation status

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

```
kubectl get pods -n trident
```

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

Using `tridentctl`

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

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

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 24.10.0        | 24.10.0        |
+-----+-----+
```

Manually deploy the Trident operator (Offline mode)

You can manually deploy the Trident operator to install Trident. This process applies to installations where the container images required by 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 Trident 24.10

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.32 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- 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 Trident installer package contains everything you need to deploy the Trident operator and install 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/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

Step 2: Create the `TridentOrchestrator` CRD

Create the `TridentOrchestrator` Custom Resource Definition (CRD). You 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:24.10.0` 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:24.10.0` if your images are all located in one registry.
- `image: <your-registry>/netapp/trident-operator:24.10.0` if your Trident image is located in a different registry from your CSI images.

Step 4: Deploy the Trident operator

The 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 Trident using a default configuration.

- For clusters running Kubernetes 1.24, 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 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

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:24.10"
tridentImage: "<your-registry>/trident:24.10.0"
```

Images in different registries

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:24.10"
tridentImage: "<your-registry>/trident:24.10.0"
```

Step 6: Create the `TridentOrchestrator` and install Trident

You can now create the `TridentOrchestrator` and install 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.


```

kubect1 create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubect1 describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Autosupport Image: <your-registry>/trident-autosupport:24.10
  Debug:              true
  Image Registry:    <your-registry>
  Namespace:         trident
  Trident Image:     <your-registry>/trident:24.10.0
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image: <your-registry>/trident-autosupport:24.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:              true
    Http Request Timeout: 90s
    Image Pull Secrets:
    Image Registry:    <your-registry>
    k8sTimeout:        30
    Kubelet Dir:       /var/lib/kubelet
    Log Format:         text
    Probe Port:        17546
    Silence Autosupport: false
    Trident Image:     <your-registry>/trident:24.10.0
  Message:             Trident installed
  Namespace:           trident
  Status:               Installed
  Version:              v24.10.0
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 Trident using this <code>TridentOrchestrator</code> CR.
Installed	Trident has successfully installed.
Uninstalling	The operator is uninstalling Trident, because <code>spec.uninstall=true</code> .
Uninstalled	Trident is uninstalled.
Failed	The operator could not install, patch, update or uninstall 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 <code>TridentOrchestrator</code> is not used. Another one already exists.

Using pod creation status

You can confirm if the 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 1m	6/6	Running	0
trident-node-linux-mr6zc 1m	2/2	Running	0
trident-node-linux-xrp7w 1m	2/2	Running	0
trident-node-linux-zh2jt 1m	2/2	Running	0
trident-operator-766f7b8658-ldzsv 3m	1/1	Running	0

Using `tridentctl`

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

```
./tridentctl -n trident version

+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 24.10.0        | 24.10.0        |
+-----+-----+
```

Deploy Trident operator using Helm (Standard mode)

You can deploy the Trident operator and install Trident using Helm. This process applies to installations where the container images required by 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 Trident 24.10

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.32 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- 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 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 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 `100.2404.0` is the version of Trident you are installing.

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0  
--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
<code>--values</code> (or <code>-f</code>)	Specify a YAML file with overrides. This can be specified multiple times and the rightmost file will take precedence.
<code>--set</code>	Specify overrides on the command line.


For example, to change the default value of `debug`, run the following command where `100.2410.0` is the version of Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0  
--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.

Option	Description	Default
<code>nodeSelector</code>	Node labels for pod assignment	
<code>podAnnotations</code>	Pod annotations	
<code>deploymentAnnotations</code>	Deployment annotations	

Option	Description	Default
tolerations	Tolerations for pod assignment	
affinity	Affinity for pod assignment	<pre> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  <p>Do not remove the default affinity from the values.yaml file. When you want to provide a custom affinity, extend the default affinity.</p> </div>
tridentControllerPluginNodeSelector	Additional node selectors for pods. Refer to Understanding controller pods and node pods for details.	
tridentControllerPluginTolerations	Overrides Kubernetes tolerations for pods. Refer to Understanding controller pods and node pods for details.	
tridentNodePluginNodeSelector	Additional node selectors for pods. Refer to Understanding controller pods and node pods for details.	
tridentNodePluginTolerations	Overrides Kubernetes tolerations for pods. Refer to Understanding controller pods and node pods for details.	

Option	Description	Default
imageRegistry	Identifies the registry for the <code>trident-operator</code> , <code>trident</code> , and other images. Leave empty to accept the default. IMPORTANT: When installing Trident in a private repository, if you are using the <code>imageRegistry</code> switch to specify the repository location, do not use <code>/netapp/</code> in the repository path.	""
imagePullPolicy	Sets the image pull policy for the <code>trident-operator</code> .	IfNotPresent
imagePullSecrets	Sets the image pull secrets for the <code>trident-operator</code> , <code>trident</code> , and other images.	
kubeletDir	Allows overriding the host location of kubelet's internal state.	"/var/lib/kubelet"
operatorLogLevel	Allows the log level of the Trident operator to be set to: <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> , or <code>fatal</code> .	"info"
operatorDebug	Allows the log level of the Trident operator to be set to debug.	true
operatorImage	Allows the complete override of the image for <code>trident-operator</code> .	""
operatorImageTag	Allows overriding the tag of the <code>trident-operator</code> image.	""
tridentIPv6	Allows enabling Trident to work in IPv6 clusters.	false
tridentK8sTimeout	Overrides the default 30-second timeout for most Kubernetes API operations (if non-zero, in seconds).	0
tridentHttpRequestTimeout	Overrides the default 90-second timeout for the HTTP requests, with 0s being an infinite duration for the timeout. Negative values are not allowed.	"90s"
tridentSilenceAutosupport	Allows disabling Trident periodic AutoSupport reporting.	false

Option	Description	Default
tridentAutosupportImageTag	Allows overriding the tag of the image for Trident AutoSupport container.	<version>
tridentAutosupportProxy	Enables Trident AutoSupport container to phone home via an HTTP proxy.	""
tridentLogFormat	Sets the Trident logging format (text or json).	"text"
tridentDisableAuditLog	Disables Trident audit logger.	true
tridentLogLevel	Allows the log level of Trident to be set to: trace, debug, info, warn, error, or fatal.	"info"
tridentDebug	Allows the log level of Trident to be set to debug.	false
tridentLogWorkflows	Allows specific Trident workflows to be enabled for trace logging or log suppression.	""
tridentLogLayers	Allows specific Trident layers to be enabled for trace logging or log suppression.	""
tridentImage	Allows the complete override of the image for Trident.	""
tridentImageTag	Allows overriding the tag of the image for Trident.	""
tridentProbePort	Allows overriding the default port used for Kubernetes liveness/readiness probes.	""
windows	Enables Trident to be installed on Windows worker node.	false
enableForceDetach	Allows enabling the force detach feature.	false
excludePodSecurityPolicy	Excludes the operator pod security policy from creation.	false
cloudProvider	Set to "Azure" when using managed identities or a cloud identity on an AKS cluster. Set to "AWS" when using a cloud identity on an EKS cluster.	""

Option	Description	Default
cloudIdentity	Set to workload identity ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx") when using cloud identity on an AKS cluster. Set to AWS IAM role ("eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role") when using cloud identity on an EKS cluster.	""
iscsiSelfHealingInterval	The interval at which the iSCSI self-healing is invoked.	5m0s
iscsiSelfHealingWaitTime	The duration after which iSCSI self-healing initiates an attempt to resolve a stale session by performing a logout and subsequent login.	7m0s
nodePrep	Enables Trident to prepare the nodes of the Kubernetes cluster to manage volumes using the specified data storage protocol. Currently, <code>iscsi</code> is the only value supported.	

Understanding controller pods and node pods

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 a 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 Trident using Helm. This process applies to installations where the container images required by 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 Trident 24.10

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.32 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- 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 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](#).



When installing Trident in a private repository, if you are using the `imageRegistry` switch to specify the repository location, do not use `/netapp/` in the repository path.

Steps

1. Add the 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. 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, `100.2410.0` is the version of Trident you are installing.

Images in one registry

```
helm install <name> netapp-trident/trident-operator --version
100.2410.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Images in different registries

```
helm install <name> netapp-trident/trident-operator --version
100.2410.0 --set imageRegistry=<your-registry> --set
operatorImage=<your-registry>/trident-operator:24.10.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:24.10
--set tridentImage=<your-registry>/trident:24.10.0 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```



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

For example, to change the default value of `debug`, run the following command where `100.2410.0` is the version of Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0
--create-namespace --namespace trident --set tridentDebug=true
```

To add the `nodePrep` value, run the following command:

```
helm install <name> netapp-trident/trident-operator --version 100.2406.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```


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.



Do not remove the default affinity from the `values.yaml` file. When you want to provide a custom affinity, extend the default affinity.

Option	Description	Default
<code>nodeSelector</code>	Node labels for pod assignment	
<code>podAnnotations</code>	Pod annotations	
<code>deploymentAnnotations</code>	Deployment annotations	
<code>tolerations</code>	Tolerations for pod assignment	

Option	Description	Default
affinity	Affinity for pod assignment	<pre data-bbox="1047 157 1489 1144"> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div data-bbox="1166 1165 1489 1438" style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  <p>Do not remove the default affinity from the values.yaml file. When you want to provide a custom affinity, extend the default affinity.</p> </div>
tridentControllerPluginNodeSelector	Additional node selectors for pods. Refer to Understanding controller pods and node pods for details.	
tridentControllerPluginTolerations	Overrides Kubernetes tolerations for pods. Refer to Understanding controller pods and node pods for details.	
tridentNodePluginNodeSelector	Additional node selectors for pods. Refer to Understanding controller pods and node pods for details.	

Option	Description	Default
tridentNodePluginTolerations	Overrides Kubernetes tolerations for pods. Refer to Understanding controller pods and node pods for details.	
imageRegistry	Identifies the registry for the trident-operator, trident, and other images. Leave empty to accept the default. IMPORTANT: When installing Trident in a private repository, if you are using the imageRegistry switch to specify the repository location, do not use /netapp/ in the repository path.	""
imagePullPolicy	Sets the image pull policy for the trident-operator.	IfNotPresent
imagePullSecrets	Sets the image pull secrets for the trident-operator, trident, and other images.	
kubeletDir	Allows overriding the host location of kubelet's internal state.	"/var/lib/kubelet"
operatorLogLevel	Allows the log level of the Trident operator to be set to: trace, debug, info, warn, error, or fatal.	"info"
operatorDebug	Allows the log level of the Trident operator to be set to debug.	true
operatorImage	Allows the complete override of the image for trident-operator.	""
operatorImageTag	Allows overriding the tag of the trident-operator image.	""
tridentIPv6	Allows enabling Trident to work in IPv6 clusters.	false
tridentK8sTimeout	Overrides the default 30-second timeout for most Kubernetes API operations (if non-zero, in seconds).	0
tridentHttpRequestTimeout	Overrides the default 90-second timeout for the HTTP requests, with 0s being an infinite duration for the timeout. Negative values are not allowed.	"90s"
tridentSilenceAutosupport	Allows disabling Trident periodic AutoSupport reporting.	false

Option	Description	Default
tridentAutosupportImageTag	Allows overriding the tag of the image for Trident AutoSupport container.	<version>
tridentAutosupportProxy	Enables Trident AutoSupport container to phone home via an HTTP proxy.	""
tridentLogFormat	Sets the Trident logging format (text or json).	"text"
tridentDisableAuditLog	Disables Trident audit logger.	true
tridentLogLevel	Allows the log level of Trident to be set to: trace, debug, info, warn, error, or fatal.	"info"
tridentDebug	Allows the log level of Trident to be set to debug.	false
tridentLogWorkflows	Allows specific Trident workflows to be enabled for trace logging or log suppression.	""
tridentLogLayers	Allows specific Trident layers to be enabled for trace logging or log suppression.	""
tridentImage	Allows the complete override of the image for Trident.	""
tridentImageTag	Allows overriding the tag of the image for Trident.	""
tridentProbePort	Allows overriding the default port used for Kubernetes liveness/readiness probes.	""
windows	Enables Trident to be installed on Windows worker node.	false
enableForceDetach	Allows enabling the force detach feature.	false
excludePodSecurityPolicy	Excludes the operator pod security policy from creation.	false
nodePrep	Enables Trident to prepare the nodes of the Kubernetes cluster to manage volumes using the specified data storage protocol. Currently, iSCSI is the only value supported.	

Customize Trident operator installation

The Trident operator allows you to customize 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

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 a 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 Trident is installed. This parameter **cannot be updated after Trident is installed**. Attempting to do so causes the `TridentOrchestrator` status to change to `Failed`. Trident is not intended to be migrated across namespaces.

This table details `TridentOrchestrator` attributes.

Parameter	Description	Default
<code>namespace</code>	Namespace to install Trident in	"default"
<code>debug</code>	Enable debugging for Trident	false
<code>enableForceDetach</code>	<p><code>ontap-san</code>, <code>ontap-san-economy</code>, and <code>ontap-nas-economy</code> only.</p> <p>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.</p>	false
<code>windows</code>	Setting to <code>true</code> enables installation on Windows worker nodes.	false
<code>cloudProvider</code>	Set to "Azure" when using managed identities or a cloud identity on an AKS cluster. Set to "AWS" when using a cloud identity on an EKS cluster.	""
<code>cloudIdentity</code>	Set to workload identity ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx") when using cloud identity on an AKS cluster. Set to AWS IAM role ("eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role") when using cloud identity on an EKS cluster.	""
<code>IPv6</code>	Install Trident over IPv6	false

Parameter	Description	Default
k8sTimeout	Timeout for Kubernetes operations	30sec
silenceAutosupport	Don't send autosupport bundles to NetApp automatically	false
autosupportImage	The container image for Autosupport Telemetry	"netapp/trident-autosupport:24.10"
autosupportProxy	The address/port of a proxy for sending Autosupport Telemetry	"http://proxy.example.com:8888"
uninstall	A flag used to uninstall Trident	false
logFormat	Trident logging format to be used [text,json]	"text"
tridentImage	Trident image to install	"netapp/trident:24.10"
imageRegistry	Path to internal registry, of the format <registry FQDN>[:port][/subpath]	"k8s.gcr.io" (Kubernetes 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 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. Follows same format as pod.spec.nodeSelector.	No default; optional
controllerPluginTolerations	Overrides Kubernetes tolerations for pods. Follows the same format as pod.spec.Tolerations.	No default; optional
nodePluginNodeSelector	Additional node selectors for pods. Follows same format as pod.spec.nodeSelector.	No default; optional
nodePluginTolerations	Overrides Kubernetes tolerations for pods. Follows the same format as pod.spec.Tolerations.	No default; optional
nodePrep	Enables Trident to prepare the nodes of the Kubernetes cluster to manage volumes using the specified data storage protocol. Currently, iscsi is the only value supported.	



For more information on formatting pod parameters, refer to [Assigning Pods to Nodes](#).

Details about force detach

Force detach is available for `ontap-san`, `ontap-san-economy` and `onstp-nas-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](#).



When using the `ontap-nas-economy` driver, you need to set the `autoExportPolicy` parameter in the backend configuration to `true` so that Trident can restrict access from the Kubernetes node with the taint applied using managed export policies.



Because 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`, Trident will determine the node status and:

1. Cease backend I/O access for volumes mounted to that node.
2. Mark the 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 be not be accepted until Trident can verify the node `clean` (safe for new publications).

When node health is restored and the taint is removed, 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, 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.

```
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 Trident with node selectors.

```
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 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 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 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-xxxxxxxxxxxxx'
```

Cloud identity on an EKS cluster

This example installs 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: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

Cloud identity for GKE

This example installs Trident for use with a cloud identity on a GKE cluster.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Install using tridentctl

Install using tridentctl

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

Critical information about Trident 24.10

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.27 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- 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 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 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 Trident version.

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

Step 2: Install Trident

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

Standard mode

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

Images in one registry

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:24.10 --trident
-image <your-registry>/trident:24.10.0
```

Images in different registries

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:24.10 --trident
-image <your-registry>/trident:24.10.0
```

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=24.10.0
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 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 Trident installed.

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

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 24.10.0        | 24.10.0        |
+-----+-----+
```

Sample configurations

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

Windows nodes

To enable Trident to run on Windows nodes:

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

Force detach

For more information about force detach, refer to [Customize Trident operator installation](#).

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

Customize tridentctl installation

You can use the Trident installer to customize installation.

Learn about the installer

The 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]`.



When installing Trident in a private repository, if you are using the `--image-registry` switch to specify the repository location, do not use `/netapp/` in the repository path. For example:

```
./tridentctl install --image-registry <image-registry> -n <namespace>
```

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
```

Copyright information

Copyright © 2025 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.