Install Trident

Trident

NetApp
June 30, 2026

Table of Contents

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 26.02

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.35 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.
- The cluster must support privileged workloads.

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
`tridentctl`	Standard or offline installation

Moving between installation methods

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

- Always use the same method for installing and uninstalling 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 perform Helm-based or Operator-based deployment without uninstalling Trident.

Other known configuration options

When installing Trident on VMWare Tanzu Portfolio products:

- 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 26.02

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.35 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/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Step 2: Create the `TridentOrchestrator` CRD

Create the `TridentOrchestrator` Custom Resource Definition (CRD). You will create a `TridentOrchestrator` Custom 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 replicasets 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.

```
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
  nodePrep:
  - iscsi
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:26.02
    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:26.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v26.02.0
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.

Status	Description
Installing	The operator is installing Trident using this `TridentOrchestrator` CR.
Installed	Trident has successfully installed.
Uninstalling	The operator is uninstalling Trident, because `spec.uninstall=true`.
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 `TridentOrchestrator` 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   6/6     Running   0
1m
trident-node-linux-mr6zc              2/2     Running   0
1m
trident-node-linux-xrp7w              2/2     Running   0
1m
trident-node-linux-zh2jt              2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv     1/1     Running   0
3m
```

Using `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

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

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.35 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/v6.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.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:26.02.0` 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:26.02.0` if your images are all located in one registry.
- `image: <your-registry>/netapp/trident-operator:26.02.0` 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 replicasets 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:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

Images in different registries

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

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.

```
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>/trident-autosupport:26.02
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.02
    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:26.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.02.0
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.

Status	Description
Installing	The operator is installing Trident using this `TridentOrchestrator` CR.
Installed	Trident has successfully installed.
Uninstalling	The operator is uninstalling Trident, because `spec.uninstall=true`.
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 `TridentOrchestrator` 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   6/6     Running   0
1m
trident-node-linux-mr6zc              2/2     Running   0
1m
trident-node-linux-xrp7w              2/2     Running   0
1m
trident-node-linux-zh2jt              2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv     1/1     Running   0
3m
```

Using `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

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 25.10

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.35 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..0` is the version of Trident you are installing.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace <trident-namespace>
```

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

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

Pass configuration data during install

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

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

For example, to change the default value of `debug`, run the following command where `100.2602.0` is the version of Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--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
`nodeSelector`	Node labels for pod assignment	
`podAnnotations`	Pod annotations	
`deploymentAnnotations`	Deployment annotations	

Option	Description	Default
`tolerations`	Tolerations for pod assignment	

`affinity`	Affinity for pod assignment	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		Do not remove the default affinity from the values.yaml file. When you want to provide a custom affinity, extend the default affinity.

`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 `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`. You can automate the force-detach process through integration with node health check (NHC) operator. For information, see Automating the failover of stateful applications with Trident.	`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`

Option	Description	Default
`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.	`""`
`cloudIdentity`	Set to workload identity ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") 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, `iscsi` is the only value supported.** NOTE: Beginning with OpenShift 4.19, the minimum Trident version supported for this feature is 25.06.1.	

Option	Description	Default
`enableConcurrency`	Enables concurrent Trident controller operations for improved throughput. **Tech Preview**: This feature is experimental and currently supports limited parallel workflows with the ONTAP-NAS (NFS only) and ONTAP-SAN (NVMe for unified ONTAP 9) drivers, in addition to the existing tech preview for the ONTAP-SAN driver (iSCSI and FCP protocols in unified ONTAP 9).	false
`k8sAPIQPS`	The queries per second (QPS) limit used by the controller while communicating with the Kubernetes API server. The Burst value is set automatically based on the QPS value.	`100`; optional

Option	Description	Default

`resources`	Sets Kubernetes resource limits and requests for the Trident controller, node, and operator pods. You can configure CPU and memory for each container and sidecar to manage resource allocation in Kubernetes.
	
	For more information about configuring resource requests and limits, refer to Resource Management for Pods and Containers.
	
	• DO NOT change the names of any containers or fields.
	• DO NOT change the indentation - YAML indentation is critical for proper parsing.
	
	• No limits are applied by default - only requests have default values and are automatically applied if not specified.
	• Container names are listed as they appear in the pod specifications.
	• Sidecars are listed under each main container.
	• Check the TORC's `status.CurrentInstallationParams` field to view the values currently applied.
		`resources:`
		`  controller:`
		`  trident-main:`
		`  requests:`
		`  cpu: 10m`
		`  memory: 80Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-provisioner:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-attacher:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-resizer:`
		`  requests:`
		`  cpu: 3m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-snapshotter:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  trident-autosupport:`
		`  requests:`
		`  cpu: 1m`
		`  memory: 30Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  node:`
		`  linux:`
		`  trident-main:`

Option	Description	Default
`httpsMetrics`	Enable HTTPS for Prometheus metrics endpoint.	false
`hostNetwork`	Enables host networking for the Trident controller. This is useful when you want to separate the frontend and backend traffic in a multi-home network.	false

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 26.02

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.35 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, `100.2602.0` is the version of Trident you are installing.

Images in one registry

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Images in different registries

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.02.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.02
--set tridentImage=<your-registry>/trident:26.02.0 --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
`--values` (or `-f`)	Specify a YAML file with overrides. This can be specified multiple times and the rightmost file will take precedence.
`--set`	Specify overrides on the command line.

For example, to change the default value of `debug`, run the following command where `100.2602.0` is the version of Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

To add the nodePrep value, run the following command:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--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
`nodeSelector`	Node labels for pod assignment	
`podAnnotations`	Pod annotations	
`deploymentAnnotations`	Deployment annotations	
`tolerations`	Tolerations for pod assignment	

Option	Description	Default

`affinity`	Affinity for pod assignment	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		Do not remove the default affinity from the values.yaml file. When you want to provide a custom affinity, extend the default affinity.

`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 180-second timeout for most Kubernetes API operations (if non-zero, in seconds). The `tridentK8sTimeout` parameter is applicable only for Trident installation.	`180`

Option	Description	Default
`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`
`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. You can automate the force-detach process through integration with node health check (NHC) operator. For information, see Automating the failover of stateful applications with Trident.	`false`

Option	Description	Default
`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.** Beginning with OpenShift 4.19, the minimum Trident version supported for this feature is 25.06.1.	

Option	Description	Default

`resources`	Sets Kubernetes resource limits and requests for the Trident controller, node, and operator pods. You can configure CPU and memory for each container and sidecar to manage resource allocation in Kubernetes.
	
	For more information about configuring resource requests and limits, refer to Resource Management for Pods and Containers.
	
	- DO NOT change the names of any containers or fields.
	- DO NOT change the indentation - YAML indentation is critical for proper parsing.
	
	- No limits are applied by default - only requests have default values.
	- Container names are listed as they appear in the pod specifications.
	- Sidecars are listed under each main container.
	- Check the TORC's `status.CurrentInstallationParams` field to view the values currently applied.
		`resources:`
		`  controller:`
		`  trident-main:`
		`  requests:`
		`  cpu: 10m`
		`  memory: 80Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-provisioner:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-attacher:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-resizer:`
		`  requests:`
		`  cpu: 3m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-snapshotter:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  trident-autosupport:`
		`  requests:`
		`  cpu: 1m`
		`  memory: 30Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  node:`
		`  linux:`

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 and 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
`namespace`	Namespace to install Trident in	`"default"`
`debug`	Enable debugging for Trident	`false`
`enableForceDetach`	`ontap-san`, `ontap-san-economy`, `ontap-nas`, and `ontap-nas-economy` only. 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. For information, see Automating the failover of stateful applications with Trident.	`false`
`windows`	Setting to `true` enables installation on Windows worker nodes.	`false`

```
    tridentmain:
      requests:
        cpu: 10m
        memory:
60Mi
      limits:
        cpu:
        memory:
    node-driver-
registrar:
      requests:
        cpu: 1m
        memory:
10Mi
      limits:
        cpu:
        memory:
    tridentctl:
      requests:
        cpu: 6m
        memory:
40Mi
      limits:
        cpu:
        memory:
  operator:
    requests:
      cpu: 10m
      memory: 40Mi
    limits:
```

Parameter	Description	Default
`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. Set to `"GCP"` when using a cloud identity on a GKE cluster.	`""`
`cloudIdentity`	Set to workload identity ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") 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. Set to cloud identity ("'iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com'") when using cloud identity on a GKE cluster.	`""`
`IPv6`	Install Trident over IPv6	false
`k8sTimeout`	Timeout for Kubernetes operations. The `k8sTimeout` parameter is applicable only for Trident installation.	`180sec`
`silenceAutosupport`	Don't send autosupport bundles to NetApp automatically	`false`
`autosupportImage`	The container image for Autosupport Telemetry	`"netapp/trident-autosupport10"`
`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:26.02"`
`imageRegistry`	Path to internal registry, of the format `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`
`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	

Parameter	Description	Default
`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.** Beginning with OpenShift 4.19, the minimum Trident version supported for this feature is 25.06.1.	
`k8sAPIQPS`	The queries per second (QPS) limit used by the controller while communicating with the Kubernetes API server. The Burst value is set automatically based on the QPS value.	`100`; optional
`enableConcurrency`	Enables concurrent Trident controller operations for improved throughput. **Tech Preview**: This feature is experimental and currently supports limited parallel workflows with the ONTAP-NAS (NFS only) and ONTAP-SAN (NVMe for unified ONTAP 9) drivers, in addition to the existing tech preview for the ONTAP-SAN driver (iSCSI and FCP protocols in unified ONTAP 9).	false

Parameter	Description	Default
`resources`	Sets Kubernetes resource limits and requests for the Trident controller and node pods. You can configure CPU and memory for each container and sidecar to manage resource allocation in Kubernetes. For more information about configuring resource requests and limits, refer to Resource Management for Pods and Containers. • DO NOT change the names of any containers or fields. • DO NOT change the indentation - YAML indentation is critical for proper parsing. • No limits are applied by default - only requests have default values and are applied automatically if not specified. • Container names are listed as they appear in the pod specifications. • Sidecars are listed under each main container. • Check the TORC's `status.CurrentInstallationParams` field to view the values currently applied.	`resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits:`

Parameter	Description	Default
`httpsMetrics`	Enable HTTPS for Prometheus metrics endpoint.	false
`hostNetwork`	Enables host networking for the Trident controller. This is useful when you want to separate the frontend and backend traffic in a multi-home network.	false

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

Sample configurations

You can use the attributes in Configuration options when defining `TridentOrchestrator` to customize your installation.

Basic custom configuration

This example, created after running the `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` command, represents a basic custom installation:

```
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, created after running the `cat deploy/crds/tridentorchestrator_cr.yaml` command, installs Trident on a Windows worker node.

```
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-xxxxxxxxxxx'
```

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
```

Kubernetes resource requests and limits configuration for Trident controller and Trident Linux node pods

This example configures Kubernetes resource requests and limits for Trident controller and Trident Linux node pods.

Disclaimer: The request and limit values provided in this example are for demonstration purposes only. Adjust these values based on your environment and workload requirements.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
      # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
          memory: 20Mi
```

```
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
```

Kubernetes resource requests and limits configuration for Trident controller and Trident Windows and Linux node pods

This example configures Kubernetes resource requests and limits for Trident controller and Trident Windows and Linux node pods.

Disclaimer: The request and limit values provided in this example are for demonstration purposes only. Adjust these values based on your environment and workload requirements.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
        # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
```

```
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
      windows:
        trident-main:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 200m
            memory: 128Mi
        # sidecars
        node-driver-registrar:
```

```
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 100m
            memory: 128Mi
        liveness-probe:
          requests:
            cpu: 2m
            memory: 40Mi
          limits:
            cpu: 50m
            memory: 64Mi
```

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 Trident10

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/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.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:26.02 --trident
-image <your-registry>/trident:26.02.0
```

Images in different registries

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident
-image <your-registry>/trident:26.02.0
```

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=26.10.0
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 |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

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 information, see Automating the failover of stateful applications with Trident.

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

Enable concurrent Trident controller operations

To enable concurrent Trident controller operations for improved throughput, add the `--enable-concurrency` option during the installation as shown in this example.

> **Tech Preview**: This feature is experimental and currently supports limited parallel workflows with the ONTAP-NAS (NFS only) and ONTAP-SAN (NVMe for unified ONTAP 9) drivers, in addition to the existing tech preview for the ONTAP-SAN driver (iSCSI and FCP protocols in unified ONTAP 9).

```
tridentctl install --enable-concurrency -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
```

Install using OpenShift certified operator

Install Trident using OpenShift OperatorHub

If you use Red Hat OpenShift, you can install NetApp Trident using the Red Hat certified operator. Use this procedure to install Trident from the Red Hat OpenShift Container Platform.

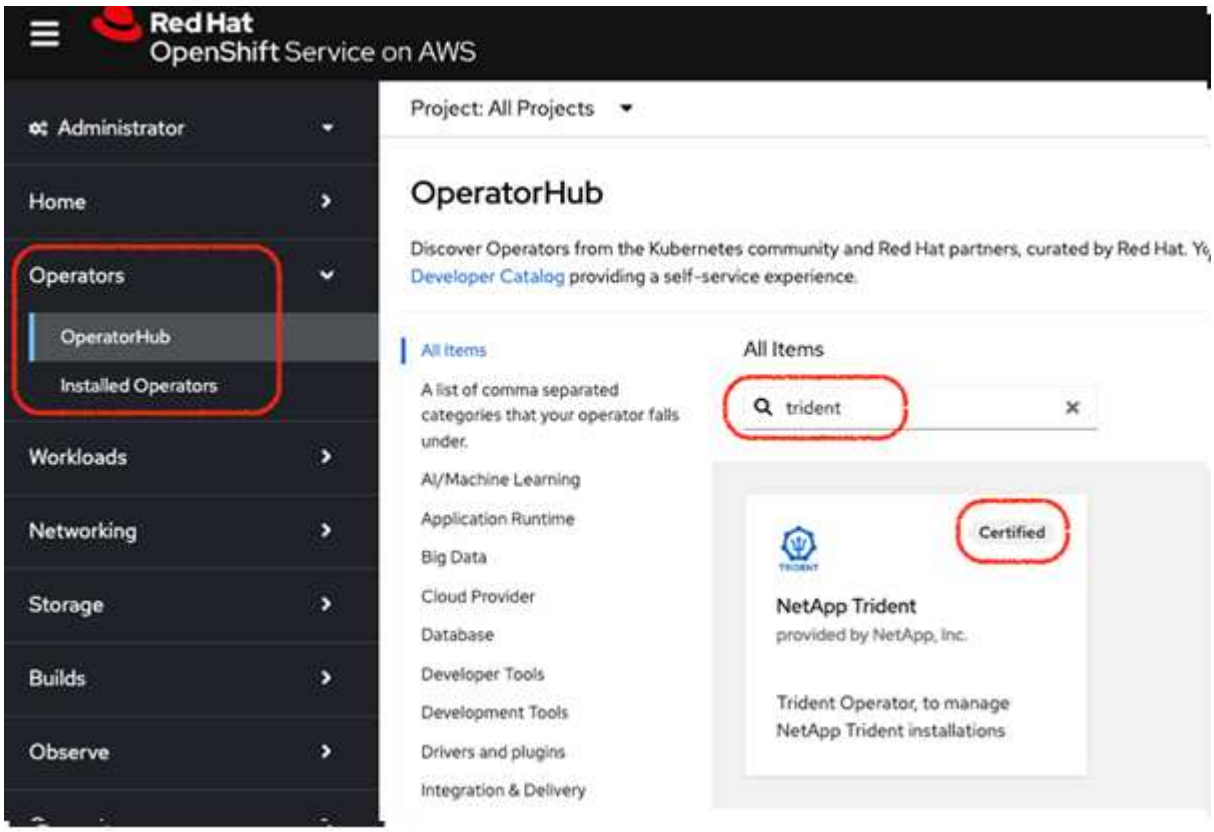
Before you begin

Before you begin the installation, prepare your environment for Trident installation.

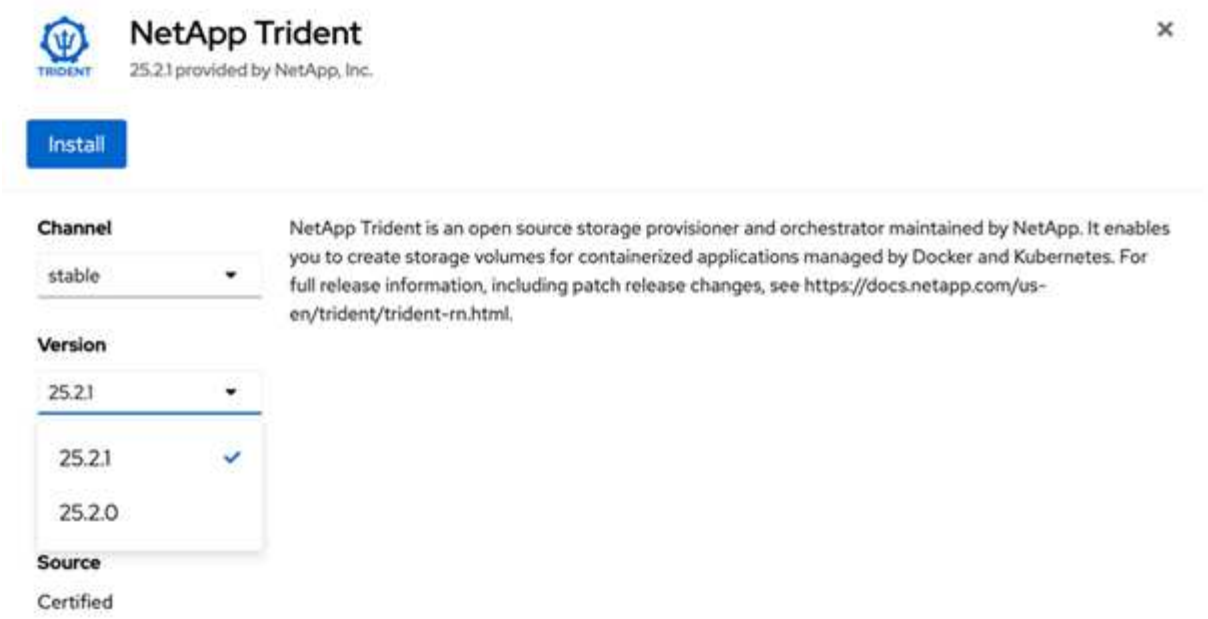
Find and install the Trident operator

Steps

1. Navigate to OpenShift OperatorHub and search for NetApp Trident.

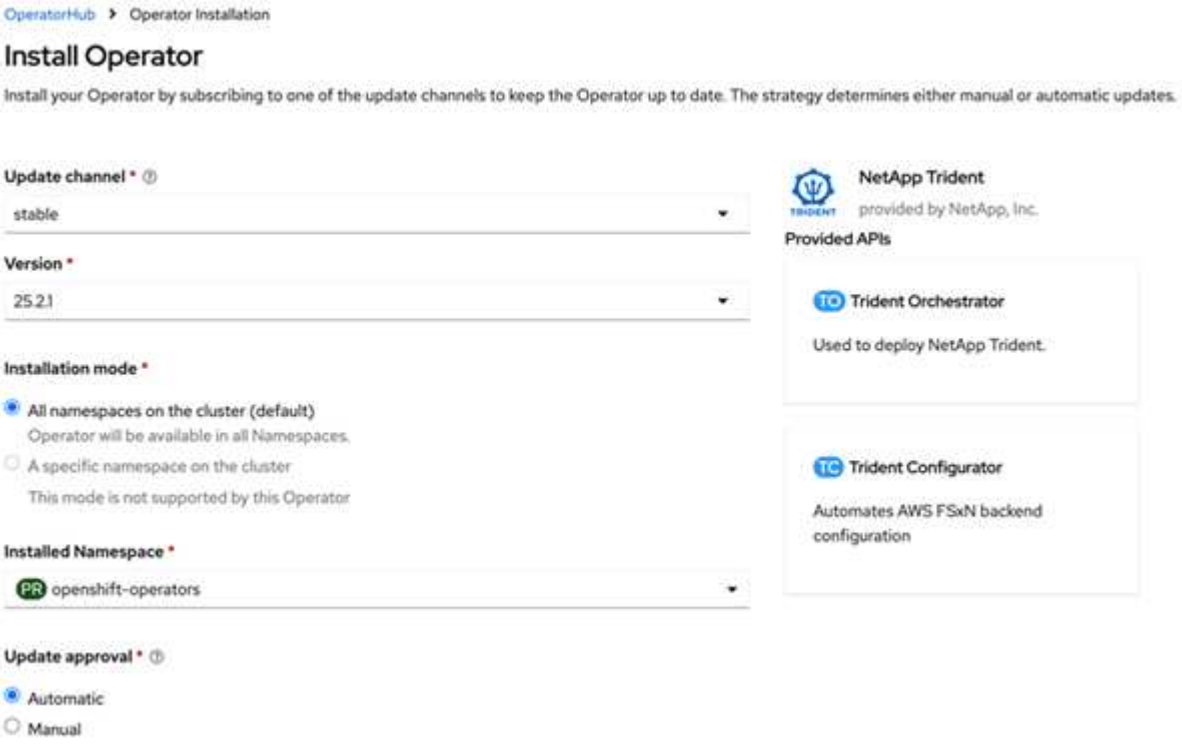


2. Click **NetApp Trident** to open the installation settings.
3. Select the required options and click **Install** to open the Operator configuration.

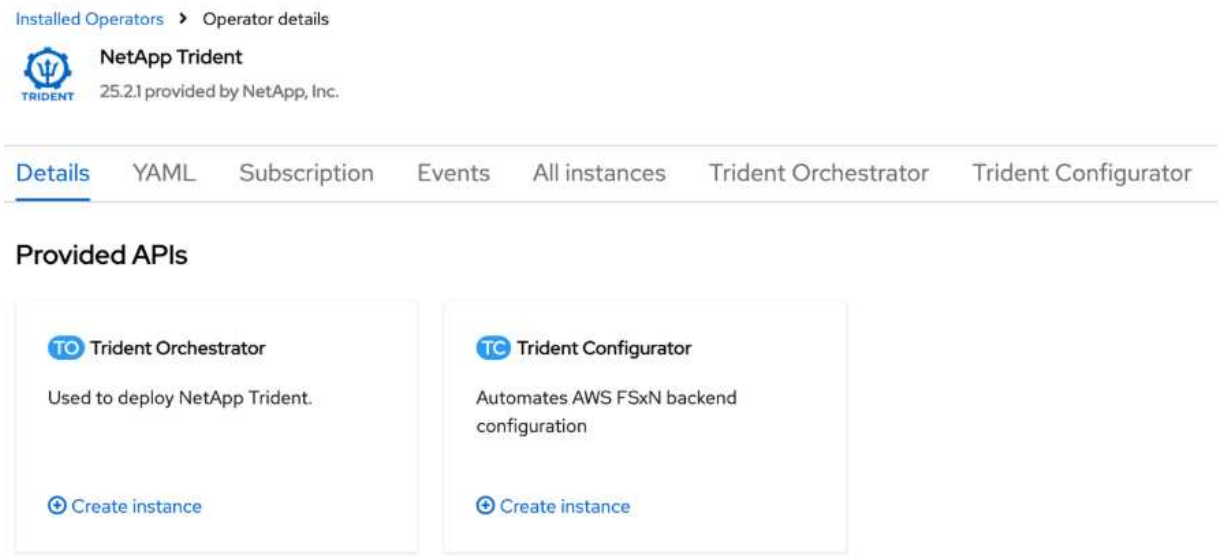


Make sure that you select the most recent Operator version.

4. Retain all the parameters as they are and click **Install**.



5. Click **View Operator** to view the details of the Operator.



6. Click **YAML view** and paste the following in the form:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false
```

The UI provides a default example. You can edit it directly instead of copying a full configuration.

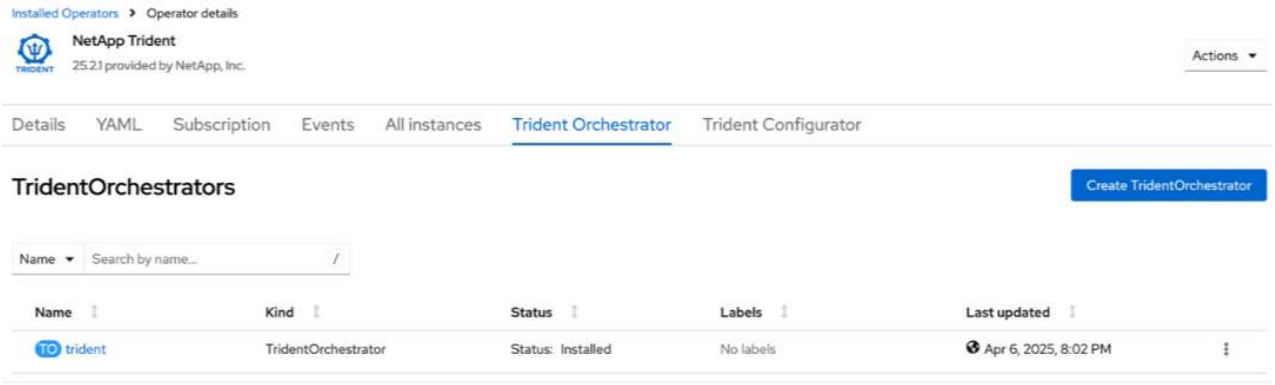
Optional: Enable concurrency

To enable concurrency, add the following field to the spec:

```
enableConcurrency: true
```

- Red Hat Enterprise Linux CoreOS (RHCOS) does not have iSCSI enabled and configured.
- You can add the `nodePrep` parameter to configure and enable both iSCSI and Multipath services on all OpenShift worker nodes.
- Beginning with OpenShift 4.19, the minimum Trident version supported for this feature is 25.06.1.

1. Click **Create**; the Trident Orchestrator will be fully installed.



Uninstall Trident operator

Steps

1. Select the Trident operator from the list of installed operator.
2. Select if you want to delete all the operand instance from the operator.

If you do not select the **Delete all operand instances from this operator** checkbox, Trident will not be uninstalled.

3. Click **Uninstall**.

Switch to the OpenShift certified Trident operator

You can switch to the Red Hat OpenShift certified Trident operator from the community operator, a Helm-based installation, or a manually deployed operator. The process for each method involves uninstalling the existing operator and then installing the certified operator using the OperatorHub.

Before you begin

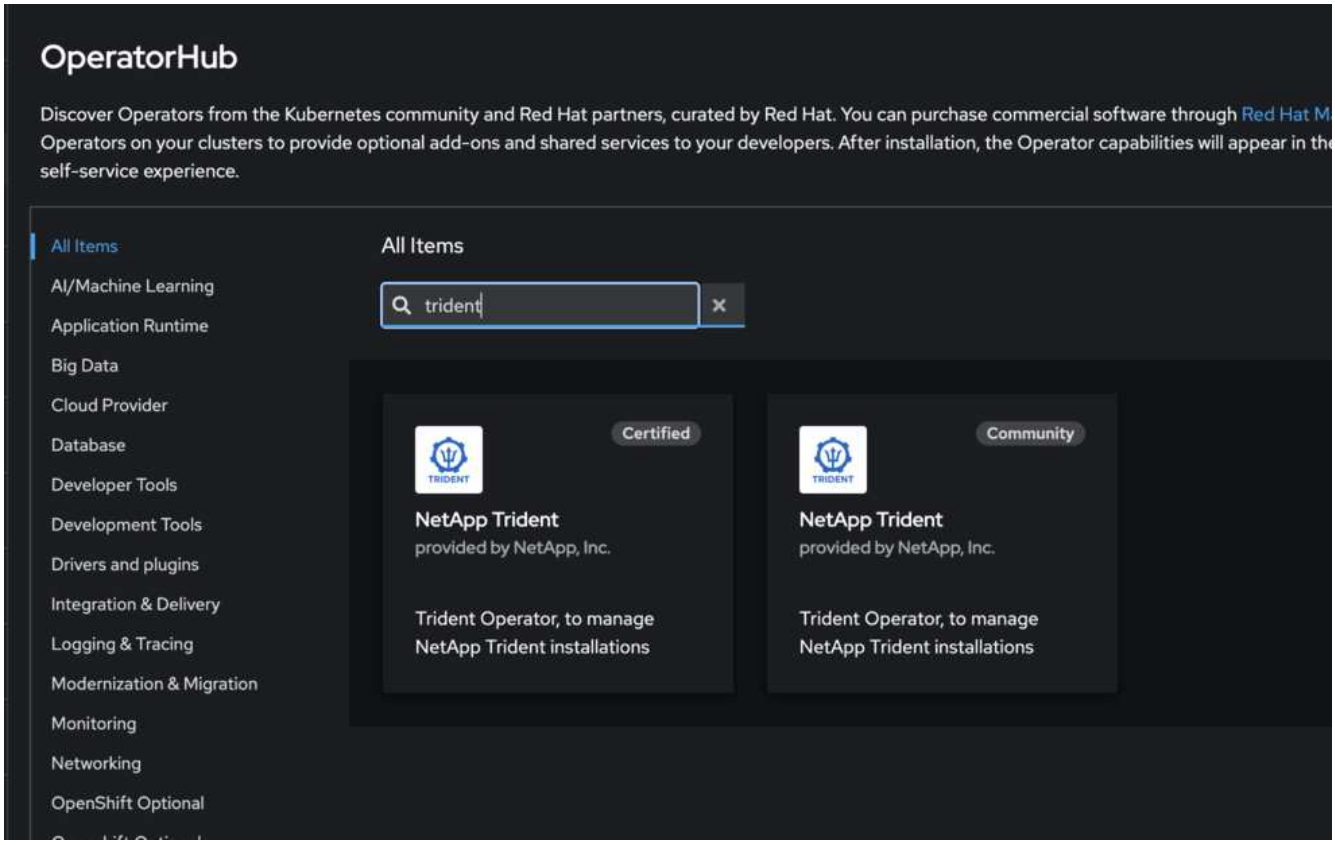
Before you begin the installation, prepare your environment for Trident installation.

Do not delete the `TridentOrchestrator` custom resource (CR) during the uninstall process. The `TridentOrchestrator` CR preserves your backend and storage class configuration, which is required after you install the certified operator.

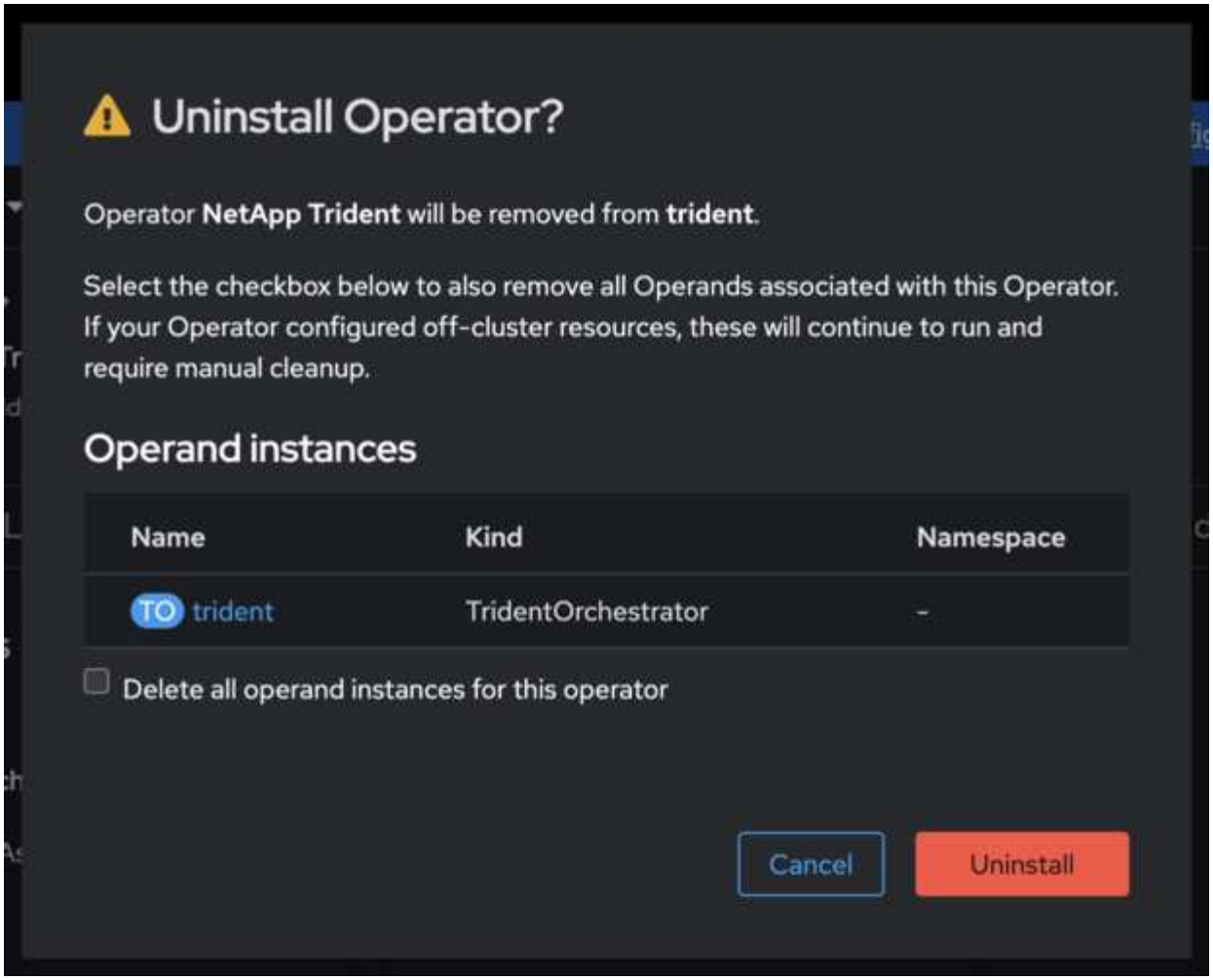
Switch from the community operator

Steps

1. Use the OpenShift console to navigate to the OperatorHub.



2. Find the NetApp Trident community operator.



Do not select **Delete all operand instances from this operator**.

3. Click **Uninstall**.
4. After the uninstall completes, proceed to Install the OpenShift certified operator.

Switch from a Helm-based operator installation

Steps

1. List the Helm release for your Trident installation:

```
helm ls -n trident
```

2. Uninstall the Helm release:

```
helm uninstall <release-name> -n trident
```

3. After the uninstall completes, proceed to Install the OpenShift certified operator.

Switch from a manually deployed operator

If you installed Trident by manually deploying the operator using a `bundle.yaml` from the installer package, remove it by deleting the same manifest.

Steps

1. Delete the operator deployment using the bundle manifest:

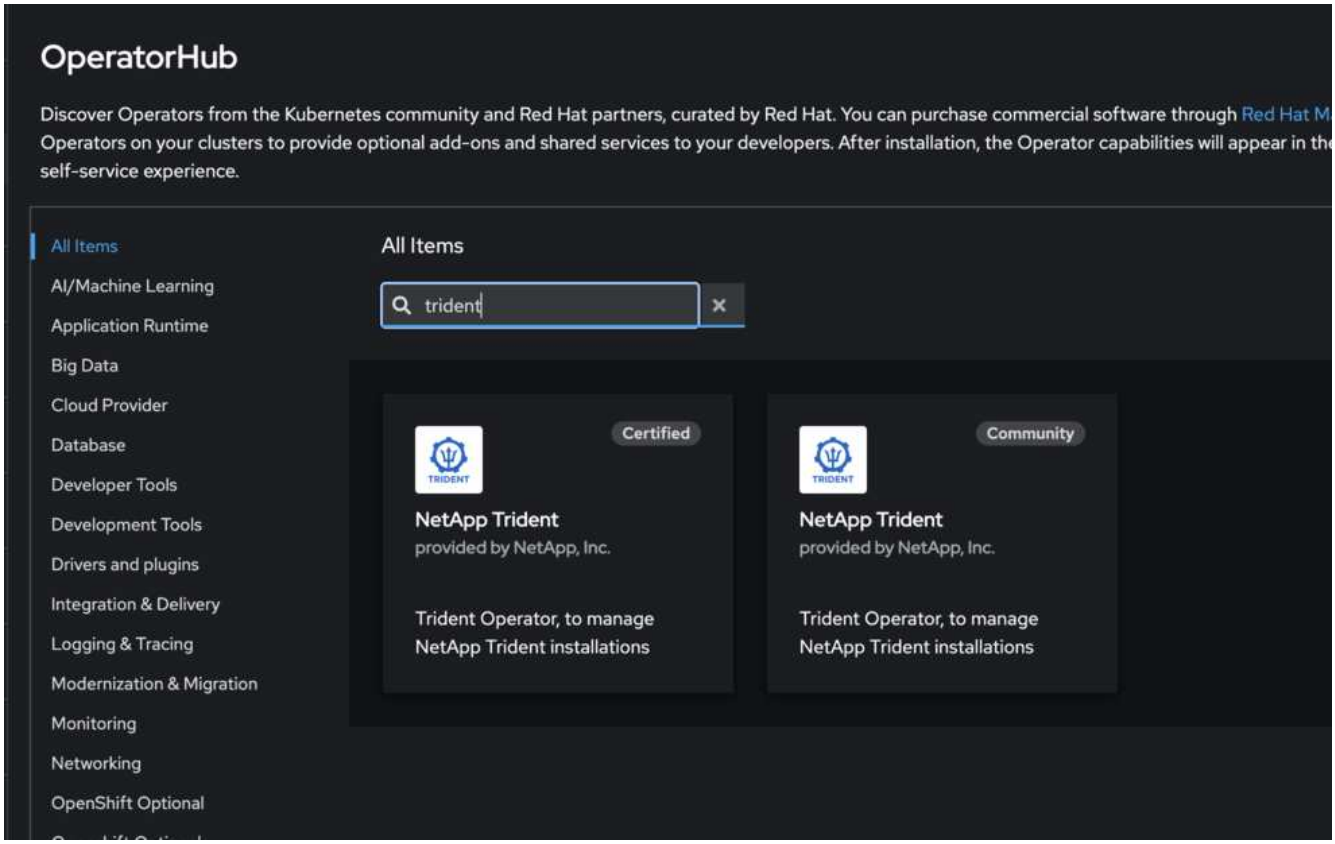
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. After the uninstall completes, proceed to Install the OpenShift certified operator.

Install the OpenShift certified operator

Steps

1. Navigate to the Red Hat OperatorHub.
2. Search for and select the NetApp Trident Operator.



3. Follow the on-screen instructions to install the operator.

Verification

- Check the OperatorHub in the console to ensure the new certified operator has been installed successfully.

Copyright information

Trademark information