



Install using Trident operator

Astra Trident

NetApp
October 22, 2024

Table of Contents

- Install using Trident operator 1
 - Manually deploy the Trident operator (Standard mode)..... 1
 - Manually deploy the Trident operator (Offline mode)..... 6
 - Deploy Trident operator using Helm (Standard mode)..... 12
 - Deploy Trident operator using Helm (Offline mode)..... 16
 - Customize Trident operator installation 20

Install using Trident operator

Manually deploy the Trident operator (Standard mode)

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

Critical information about Astra Trident 23.07

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.27 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in `multipath.conf` file.

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

Manually deploy the Trident operator and install Trident

Review [the installation overview](#) to ensure you've met installation prerequisites and selected the correct installation option for your environment.

Before you begin

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



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

1. Verify your Kubernetes version:

```
kubectl version
```

2. Verify cluster administrator privileges:

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

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

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

Step 1: Download the Trident installer package

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

```
wget https://github.com/NetApp/trident/releases/download/v23.07.1/trident-
installer-23.07.1.tar.gz
tar -xf trident-installer-23.07.1.tar.gz
cd trident-installer
```

Step 2: Create the TridentOrchestrator CRD

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

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

Step 3: Deploy the Trident operator

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

- For clusters running Kubernetes 1.24 or earlier, use `bundle_pre_1_25.yaml`.

- For clusters running Kubernetes 1.25 or later, use `bundle_post_1_25.yaml`.

Before you begin

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

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

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

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

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

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

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

Steps

1. Create the resources and deploy the operator:

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

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

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



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

Step 4: Create the `TridentOrchestrator` and install Trident

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

```

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
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image:      netapp/trident-autosupport:23.07
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:              true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:         30
    Kubelet Dir:        /var/lib/kubelet
    Log Format:          text
    Silence Autosupport: false
    Trident Image:      netapp/trident:23.07.1
  Message:            Trident installed Namespace:
trident
  Status:              Installed
  Version:              v23.07.1
Events:
  Type Reason Age From Message ---- -
-----
Normal
Installing 74s trident-operator.netapp.io Installing Trident Normal
Installed 67s trident-operator.netapp.io Trident installed

```

Verify the installation

There are several ways to verify your installation.

Using `TridentOrchestrator` status

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

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

Using pod creation status

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

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

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

Using `tridentctl`

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

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

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 23.07.1        | 23.07.1        |
+-----+-----+
```

Manually deploy the Trident operator (Offline mode)

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

Critical information about Astra Trident 23.07

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.27 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in `multipath.conf` file.

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

Manually deploy the Trident operator and install Trident

Review [the installation overview](#) to ensure you've met installation prerequisites and selected the correct installation option for your environment.

Before you begin

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



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

1. Verify your Kubernetes version:

```
kubectl version
```

2. Verify cluster administrator privileges:

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

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

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

Step 1: Download the Trident installer package

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

```
wget https://github.com/NetApp/trident/releases/download/v23.07.1/trident-
installer-23.07.1.tar.gz
tar -xf trident-installer-23.07.1.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:23.07.1` 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:23.07.1` if your images are all located in one registry.

- `image: <your-registry>/netapp/trident-operator:23.07.1` if your Trident image is located in a different registry from your CSI images.

Step 4: Deploy the Trident operator

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

- For clusters running Kubernetes 1.24 or earlier, use `bundle_pre_1_25.yaml`.
- For clusters running Kubernetes 1.25 or later, use `bundle_post_1_25.yaml`.

Before you begin

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

```
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:23.07"
tridentImage: "<your-registry>/trident:23.07.1"
```

Images in different registries

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

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.07"
tridentImage: "<your-registry>/netapp/trident:23.07.1"
```

Step 6: Create the `TridentOrchestrator` and install Trident

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

```

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>/netapp/trident-autosupport:23.07
  Debug:              true
  Image Registry:    <your-registry>/sig-storage
  Namespace:         trident
  Trident Image:     <your-registry>/netapp/trident:23.07.1
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image: <your-registry>/netapp/trident-
autosupport:23.07
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:              true
    Http Request Timeout: 90s
    Image Pull Secrets:
    Image Registry:    <your-registry>/sig-storage
    k8sTimeout:        30
    Kubelet Dir:       /var/lib/kubelet
    Log Format:         text
    Probe Port:        17546
    Silence Autosupport: false
    Trident Image:     <your-registry>/netapp/trident:23.07.1
  Message:            Trident installed
  Namespace:          trident
  Status:              Installed
  Version:             v23.07.1
Events:
  Type Reason Age From Message ---- -
-----
Normal
Installing 74s trident-operator.netapp.io Installing Trident Normal
Installed 67s trident-operator.netapp.io Trident installed

```

Verify the installation

There are several ways to verify your installation.

Using `TridentOrchestrator` status

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

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

Using pod creation status

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

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

NAME	READY	STATUS	RESTARTS
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 Astra Trident installed.

```
./tridentctl -n trident version

+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 23.07.1       | 23.07.1       |
+-----+-----+
```

Deploy Trident operator using Helm (Standard mode)

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

Critical information about Astra Trident 23.07.1

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.27 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in `multipath.conf` file.

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

Deploy the Trident operator and install Astra Trident using Helm

Using the Trident [Helm Chart](#) you can deploy the Trident operator and install Trident in one step.

Review [the installation overview](#) to ensure you've met installation prerequisites and selected the correct installation option for your environment.

Before you begin

In addition to the [deployment prerequisites](#) you need [Helm version 3](#).

Steps

1. Add the Astra Trident Helm repository:

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

2. Use `helm install` and specify a name for your deployment as in the following example where `23.07.1` is the version of Astra Trident you are installing.

```
helm install <name> netapp-trident/trident-operator --version 23.07.1  
--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 `--set` command where `23.07.1` is the version of Astra Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 23.07.1  
--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	
<code>tolerations</code>	Tolerations for pod assignment	

Option	Description	Default
affinity	Affinity for pod assignment	
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.	
imageRegistry	Identifies the registry for the trident-operator, trident, and other images. Leave empty to accept the default.	""
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 Astra 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

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 Astra Trident periodic AutoSupport reporting.	false
tridentAutosupportImageTag	Allows overriding the tag of the image for Astra Trident AutoSupport container.	<version>
tridentAutosupportProxy	Allows Astra Trident AutoSupport container to phone home via an HTTP proxy.	""
tridentLogFormat	Sets the Astra Trident logging format (text or json).	"text"
tridentDisableAuditLog	Disables Astra Trident audit logger.	true
tridentLogLevel	Allows the log level of Astra Trident to be set to: trace, debug, info, warn, error, or fatal.	"info"
tridentDebug	Allows the log level of Astra Trident to be set to debug.	false
tridentLogWorkflows	Allows specific Astra Trident workflows to be enabled for trace logging or log suppression.	""
tridentLogLayers	Allows specific Astra Trident layers to be enabled for trace logging or log suppression.	""
tridentImage	Allows the complete override of the image for Astra Trident.	""
tridentImageTag	Allows overriding the tag of the image for Astra Trident.	""
tridentProbePort	Allows overriding the default port used for Kubernetes liveness/readiness probes.	""
windows	Allows Astra 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

Understanding controller pods and node pods

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

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

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

Deploy Trident operator using Helm (Offline mode)

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

Critical information about Astra Trident 23.07

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.27 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in `multipath.conf` file.

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

Deploy the Trident operator and install Astra Trident using Helm

Using the Trident [Helm Chart](#) you can deploy the Trident operator and install Trident in one step.

Review [the installation overview](#) to ensure you've met installation prerequisites and selected the correct installation option for your environment.

Before you begin

In addition to the [deployment prerequisites](#) you need [Helm version 3](#).

Steps

1. Add the Astra Trident Helm repository:

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

2. Use `helm install` and specify a name for your deployment 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, 23.07.1 is the version of Astra Trident you are installing.

Images in one registry

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

Images in different registries

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

```
helm install <name> netapp-trident/trident-operator --version
23.07.1 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.07.1 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.07 --set tridentImage=<your-
registry>/netapp/trident:23.07.1 --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 `--set` command where 23.07.1 is the version of Astra Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 23.07.1
--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	
<code>tolerations</code>	Tolerations for pod assignment	
<code>affinity</code>	Affinity for pod assignment	
<code>tridentControllerPluginNodeSelector</code>	Additional node selectors for pods. Refer to Understanding controller pods and node pods for details.	
<code>tridentControllerPluginTolerations</code>	Overrides Kubernetes tolerations for pods. Refer to Understanding controller pods and node pods for details.	
<code>tridentNodePluginNodeSelector</code>	Additional node selectors for pods. Refer to Understanding controller pods and node pods for details.	
<code>tridentNodePluginTolerations</code>	Overrides Kubernetes tolerations for pods. Refer to Understanding controller pods and node pods for details.	
<code>imageRegistry</code>	Identifies the registry for the <code>trident-operator</code> , <code>trident</code> , and other images. Leave empty to accept the default.	""
<code>imagePullPolicy</code>	Sets the image pull policy for the <code>trident-operator</code> .	IfNotPresent
<code>imagePullSecrets</code>	Sets the image pull secrets for the <code>trident-operator</code> , <code>trident</code> , and other images.	
<code>kubeletDir</code>	Allows overriding the host location of kubelet's internal state.	"/var/lib/kubelet"
<code>operatorLogLevel</code>	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"
<code>operatorDebug</code>	Allows the log level of the Trident operator to be set to <code>debug</code> .	true
<code>operatorImage</code>	Allows the complete override of the image for <code>trident-operator</code> .	""

Option	Description	Default
operatorImageTag	Allows overriding the tag of the <code>trident-operator</code> image.	""
tridentIPv6	Allows enabling Astra 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 Astra Trident periodic AutoSupport reporting.	false
tridentAutosupportImageTag	Allows overriding the tag of the image for Astra Trident AutoSupport container.	<version>
tridentAutosupportProxy	Allows Astra Trident AutoSupport container to phone home via an HTTP proxy.	""
tridentLogFormat	Sets the Astra Trident logging format (<code>text</code> or <code>json</code>).	"text"
tridentDisableAuditLog	Disables Astra Trident audit logger.	true
tridentLogLevel	Allows the log level of Astra Trident to be set to: <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> , or <code>fatal</code> .	"info"
tridentDebug	Allows the log level of Astra Trident to be set to <code>debug</code> .	false
tridentLogWorkflows	Allows specific Astra Trident workflows to be enabled for trace logging or log suppression.	""
tridentLogLayers	Allows specific Astra Trident layers to be enabled for trace logging or log suppression.	""
tridentImage	Allows the complete override of the image for Astra Trident.	""
tridentImageTag	Allows overriding the tag of the image for Astra Trident.	""
tridentProbePort	Allows overriding the default port used for Kubernetes liveness/readiness probes.	""

Option	Description	Default
<code>windows</code>	Allows Astra Trident to be installed on Windows worker node.	<code>false</code>
<code>enableForceDetach</code>	Allows enabling the force detach feature.	<code>false</code>
<code>excludePodSecurityPolicy</code>	Excludes the operator pod security policy from creation.	<code>false</code>

What's next

Customize Trident operator installation

The Trident operator allows you to customize Astra Trident installation using the attributes in the `TridentOrchestrator` spec. If you want to customize the installation beyond what `TridentOrchestrator` arguments allow, consider using `tridentctl` to generate custom YAML manifests to modify as needed.

Understanding controller pods and node pods

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

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

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

Configuration options



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

This table details `TridentOrchestrator` attributes.

Parameter	Description	Default
<code>namespace</code>	Namespace to install Astra Trident in	"default"
<code>debug</code>	Enable debugging for Astra Trident	<code>false</code>

Parameter	Description	Default
enableForceDetach	<p>ontap-san and ontap-san-economy 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> <p>This is an experimental feature in 23.04. Refer to Details about force detach for important details.</p>	false
windows	Setting to true enables installation on Windows worker nodes.	false
IPv6	Install Astra Trident over IPv6	false
k8sTimeout	Timeout for Kubernetes operations	30sec
silenceAutosupport	Don't send autosupport bundles to NetApp automatically	false
autosupportImage	The container image for Autosupport Telemetry	"netapp/trident-autosupport:23.07"
autosupportProxy	The address/port of a proxy for sending Autosupport Telemetry	"http://proxy.example.com:8888"
uninstall	A flag used to uninstall Astra Trident	false
logFormat	Astra Trident logging format to be used [text,json]	"text"
tridentImage	Astra Trident image to install	"netapp/trident:23.07"
imageRegistry	Path to internal registry, of the format <registry FQDN>[:port] [/subpath]	"k8s.gcr.io/sig-storage" (k8s 1.19+) or "quay.io/k8scsi"
kubeletDir	Path to the kubelet directory on the host	"/var/lib/kubelet"
wipeout	A list of resources to delete to perform a complete removal of Astra Trident	
imagePullSecrets	Secrets to pull images from an internal registry	
imagePullPolicy	<p>Sets the image pull policy for the the Trident operator. Valid values are:</p> <p>Always to always pull the image.</p> <p>IfNotPresent to pull the image only if it does not already exist on the node.</p> <p>Never to never pull the image.</p>	IfNotPresent
controllerPluginNodeSelector	Additional node selectors for pods. Follows same format as pod.spec.nodeSelector.	No default; optional

Parameter	Description	Default
<code>controllerPluginTolerations</code>	Overrides Kubernetes tolerations for pods. Follows the same format as <code>pod.spec.Tolerations</code> .	No default; optional
<code>nodePluginNodeSelector</code>	Additional node selectors for pods. Follows same format as <code>pod.spec.nodeSelector</code> .	No default; optional
<code>nodePluginTolerations</code>	Overrides Kubernetes tolerations for pods. Follows the same format as <code>pod.spec.Tolerations</code> .	No default; optional



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

Details about force detach

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



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

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

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



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

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

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

Sample configurations

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

Example 1: Basic custom configuration

This is an example for a basic custom configuration.

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

Example 2: Deploy with node selectors

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

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

Example 3: Deploy on Windows worker nodes

This example illustrates deployment on a Windows worker node.

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

Copyright information

Copyright © 2024 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.