NetApp

Install using Trident operator

Astra Trident

NetApp
April 05, 2023

Table of Contents

Install using Trident operator

Manually deploy the Trident operator (Standard mode)

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

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.26 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.

- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

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

Manually deploy the Trident operator and install Trident

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

Before you begin

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

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

1. Verify your Kubernetes version:

```
kubectl version
```

2. Verify cluster administrator privileges:

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

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

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

Step 1: Download the Trident installer package

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

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.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 lower, use `bundle_pre_1_25.yaml`.

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

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

Steps

1. Create the resources and deploy the operator:

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

> (i) To deploy the operator in a namespace other than the `trident` namespace, update `serviceaccount.yaml`, `clusterrolebinding.yaml` and `operator.yaml` and generate your bundle file using the `kustomization.yaml`:
>
> ```
> kubectl kustomize deploy/ > deploy/<bundle>.yaml
> ```

2. Verify the operator was deployed.

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

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

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

Step 4: Create the `TridentOrchestrator` and install Trident

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

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

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                     false
    Autosupport Hostname:
    Autosupport Image:        netapp/trident-autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                    true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:          30
    Kubelet Dir:         /var/lib/kubelet
    Log Format:          text
    Silence Autosupport: false
    Trident Image:       netapp/trident:23.01.1
  Message:                     Trident installed  Namespace:
trident
  Status:                      Installed
  Version:                     v23.01.1
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 `TridentOrchestrator` CR.
Installed	Astra Trident has successfully installed.
Uninstalling	The operator is uninstalling Astra Trident, because `spec.uninstall=true`.
Uninstalled	Astra Trident is uninstalled.
Failed	The operator could not install, patch, update or uninstall Astra Trident; the operator will automatically try to recover from this state. If this state persists you will require troubleshooting.
Updating	The operator is updating an existing installation.
Error	The `TridentOrchestrator` is not used. Another one already exists.

Using pod creation status

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

```
kubectl get pods -n trident

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

Using `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

What's next

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

Manually deploy the Trident operator (Offline mode)

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

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

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

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

Manually deploy the Trident operator and install Trident

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

Before you begin

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

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

1. Verify your Kubernetes version:

```
kubectl version
```

2. Verify cluster administrator privileges:

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

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

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

Step 1: Download the Trident installer package

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

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.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:23.01.1` 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:23.01.1` if your images are all located in one registry.

- **image: `<your-registry>/netapp/trident-operator:23.01.1`** if your Trident image is located in a different registry from your CSI images.

Step 4: Deploy the Trident operator

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

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

1. Create the resources and deploy the operator:

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

   > ⓘ The Astra Trident installer provides a bundle file that can be used to install the operator and create associated objects. The bundle file is an easy way to deploy the operator and install Astra Trident using a default configuration.
   >
   > - For clusters running Kubernetes 1.24 or lower, use `bundle_pre_1_25.yaml`.
   > - For clusters running Kubernetes 1.25 or higher, use `bundle_post_1_25.yaml`.

2. Verify the operator was deployed.

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

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

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

Step 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:23.01"
tridentImage: "<your-registry>/trident:23.01.1"
```

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:23.01"
tridentImage: "<your-registry>/netapp/trident:23.01.1"
```

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.

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

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/netapp/trident-autosupport:23.01
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.01.1
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         <your-registry>/netapp/trident-
autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Http Request Timeout:      90s
    Image Pull Secrets:
    Image Registry:      <your-registry>/sig-storage
    k8sTimeout:          30
    Kubelet Dir:         /var/lib/kubelet
    Log Format:          text
    Probe Port:          17546
    Silence Autosupport: false
    Trident Image:       <your-registry>/netapp/trident:23.01.1
  Message:               Trident installed
  Namespace:             trident
  Status:                Installed
  Version:               v23.01.1
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 `TridentOrchestrator` CR.
Installed	Astra Trident has successfully installed.
Uninstalling	The operator is uninstalling Astra Trident, because `spec.uninstall=true`.
Uninstalled	Astra Trident is uninstalled.
Failed	The operator could not install, patch, update or uninstall Astra Trident; the operator will automatically try to recover from this state. If this state persists you will require troubleshooting.
Updating	The operator is updating an existing installation.
Error	The `TridentOrchestrator` is not used. Another one already exists.

Using pod creation status

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

```
kubectl get pods -n trident

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

Using `tridentctl`

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

```
./tridentctl -n trident version

+---------------+---------------+
| SERVER VERSION | CLIENT VERSION |
+---------------+---------------+
| 23.01.1       | 23.01.1        |
+---------------+---------------+
```

What's next

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

Deploy Trident operator using Helm (Standard mode)

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

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.26 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.

- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

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

Deploy the Trident operator and install Astra Trident using Helm

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

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

Before you begin

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

Steps

1. Add the Astra Trident Helm repository:

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

2. Use `helm install` and specify a name for your deployment as in the following example where `23.01.1` is the version of Astra Trident you are installing.

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--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
`--values` (or `-f`)	Specify a YAML file with overrides. This can be specified multiple times and the rightmost file will take precedence.
`--set`	Specify overrides on the command line.

For example, to change the default value of `debug`, run the following `--set` command where `23.01.1` is the version of Astra Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

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

What's next

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

Deploy Trident operator using Helm (Offline mode)

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

Critical information about Astra Trident 23.01

You must read the following critical information about Astra Trident.

Critical information about Astra Trident

- Kubernetes 1.26 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.

- Astra Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

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

Deploy the Trident operator and install Astra Trident using Helm

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

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

Before you begin

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

Steps

1. Add the Astra Trident Helm repository:

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

2. Use `helm install` and specify a name for your deployment and image registry location. 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, `23.01.1` is the version of Astra Trident you are installing.

Images in one registry

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --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
23.01.1 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.01.1 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.01 --set tridentImage=<your-
registry>/netapp/trident:23.01.1 --create-namespace --namespace
<trident-namespace>
```

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

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

Pass configuration data during install

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

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

For example, to change the default value of `debug`, run the following `--set` command where `23.01.1` is the version of Astra Trident you are installing:

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

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

What's next

Now you can [create create a backend and storage class, provision a volume, and mount the volume in a pod](#).

Customize Trident operator installation

The Trident operator allows you to customize Astra Trident installation using the attributes in the `TridentOrchestrator` spec. If you want to customize the installation beyond what `TridentOrchestrator` arguments allow, 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
`namespace`	Namespace to install Astra Trident in	"default"
`debug`	Enable debugging for Astra Trident	false
`windows`	Setting to `true` enables installation on Windows worker nodes.	false
`IPv6`	Install Astra Trident over IPv6	false
`k8sTimeout`	Timeout for Kubernetes operations	30sec
`silenceAutosupport`	Don't send autosupport bundles to NetApp automatically	false

Parameter	Description	Default
`enableNodePrep`	Manage worker node dependencies automatically (**BETA**)	false
`autosupportImage`	The container image for Autosupport Telemetry	"netapp/trident-autosupport:23.01"
`autosupportProxy`	The address/port of a proxy for sending Autosupport Telemetry	"http://proxy.example.com:8888"
`uninstall`	A flag used to uninstall Astra Trident	false
`logFormat`	Astra Trident logging format to be used [text,json]	"text"
`tridentImage`	Astra Trident image to install	"netapp/trident:21.04"
`imageRegistry`	Path to internal registry, of the format `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage (k8s 1.19+) or quay.io/k8scsi"
`kubeletDir`	Path to the kubelet directory on the host	"/var/lib/kubelet"
`wipeout`	A list of resources to delete to perform a complete removal of Astra Trident	
`imagePullSecrets`	Secrets to pull images from an internal registry	
`imagePullPolicy`	Sets the image pull policy for the the Trident operator. Valid values are: `Always` to always pull the image. `IfNotPresent` to pull the image only if it does not already exist on the node. `Never` to never pull the image.	`IfNotPresent`
`controllerPluginNodeSelector`	Additional node selectors for pods. 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

Parameter	Description	Default
`nodePluginTolerations`	Overrides Kubernetes tolerations for pods. Follows the same format as pod.spec.Tolerations.	No default; optional

> (i) For more information on formatting pod parameters, see Assigning Pods to Nodes.

Sample configurations

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

Example 1: Basic custom configuration

This is an example for a basic custom configuration.

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

Example 2: Deploy with node selectors

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

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

Example 3: Deploy on Windows worker nodes

This example illustrates deployment on a Windows worker node.

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