Deploy with Trident operator

Astra Trident

NetApp
June 27, 2022

Table of Contents

Deploy with Trident operator

You can deploy Astra Trident with the Trident operator. You can deploy the Trident operator either manually or using Helm.

If you have not already familiarized yourself with the basic concepts, now is a great time to do that.

What you'll need

To deploy Astra Trident, the following prerequisites should be met:

- You have full privileges to a supported Kubernetes cluster running Kubernetes 1.17 and above.
- You have access to a supported NetApp storage system.
- You have the capability to mount volumes from all of the Kubernetes worker nodes.
- You have 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.
- You have set the `KUBECONFIG` environment variable to point to your Kubernetes cluster configuration.
- You have enabled the feature gates required by Astra Trident.
- If you are using Kubernetes with Docker Enterprise, follow their steps to enable CLI access.

Got all that? Great! Let's get started.

Deploy the Trident operator by using Helm

Perform the steps listed to deploy the Trident operator by using Helm.

What you'll need

In addition to the prerequisites listed above, to deploy Trident operator by using Helm, you need the following:

- Kubernetes 1.17 and later
- Helm version 3

Steps

1. Add Trident's Helm repository:

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

2. Use the `helm install` command and specify a name for your deployment.
   See the following example:

```
helm install <release-name> netapp-trident/trident-operator --version
22.1.0 --namespace <trident-namespace>
```

If you did not already create a namespace for Trident, you can add the `--create-namespace` parameter to the `helm install` command. Helm will then automatically create the namespace for you.

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

- `--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:

```
$ helm install <name> netapp-trident/trident-operator --version 22.1.0
--set tridentDebug=true
```

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

`helm list` shows you details about the installation, such as name, namespace, chart, status, app version, revision number, and so on.

Deploy the Trident operator manually

Perform the steps listed to manually deploy the Trident operator.

Step 1: Qualify your Kubernetes cluster

The first thing you need to do is log in to the Linux host and verify that it is managing a *working*, supported Kubernetes cluster that you have the necessary privileges to.

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`.

To see if your Kubernetes version is later than 1.17, run the following command:

```
kubectl version
```

To see if you have Kubernetes cluster administrator privileges, run the following command:

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

To verify if you can launch a pod that uses an image from Docker Hub and reach your storage system over the pod network, run the following command:

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

Step 2: Download and set up the operator

> Beginning with 21.01, the Trident operator is cluster scoped. Using the Trident operator to install Trident requires creating the `TridentOrchestrator` Custom Resource Definition (CRD) and defining other resources. You should perform these steps to set up the operator before you can install Astra Trident.

1. Download the latest version of the Trident installer bundle from the *Downloads* section and extract it.

   ```
   wget https://github.com/NetApp/trident/releases/download/v21.04/trident-installer-21.04.tar.gz
   tar -xf trident-installer-21.04.tar.gz
   cd trident-installer
   ```

2. Use the appropriate CRD manifest to create the `TridentOrchestrator` CRD. You then create a `TridentOrchestrator` Custom Resource later on to instantiate an installation by the operator.

   Run the following command:

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

3. After the `TridentOrchestrator` CRD is created, create the following resources required for the operator deployment:
   - A ServiceAccount for the operator
   - A ClusterRole and ClusterRoleBinding to the ServiceAccount
   - A dedicated PodSecurityPolicy
   - The operator itself

   The Trident installer contains manifests for defining these resources. By default, the operator is deployed in the `trident` namespace. If the `trident` namespace does not exist, use the following manifest to create one.

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

4. To deploy the operator in a namespace other than the default `trident` namespace, you should update the `serviceaccount.yaml`, `clusterrolebinding.yaml` and `operator.yaml` manifests and generate your `bundle.yaml`.

   Run the following command to update the YAML manifests and generate your `bundle.yaml` using the

`kustomization.yaml`:

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

Run the following command to create the resources and deploy the operator:

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

5. To verify the status of the operator after you have deployed, do the following:

```
$ kubectl get deployment -n <operator-namespace>
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m

$ kubectl get pods -n <operator-namespace>
NAME                                READY   STATUS              RESTARTS
AGE
trident-operator-54cb664d-lnjxh   1/1     Running             0
3m
```

The operator deployment successfully creates a pod running on one of the worker nodes in your cluster.

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

Step 3: Create `TridentOrchestrator` and install Trident

You are now ready to install Astra Trident using the operator! This will require creating `TridentOrchestrator`. The Trident installer comes with example definitions for creating `TridentOrchestrator`. This kicks off an installation in the `trident` namespace.

```
$ 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:21.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

The Trident operator enables you to customize the manner in which Astra Trident is installed by using the attributes in the `TridentOrchestrator` spec. See Customize your Trident deployment.

The Status of `TridentOrchestrator` indicates if the installation was successful and displays the version of Trident installed.

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.

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, you should check the logs of the operator. See the troubleshooting section.

You can confirm if the Astra Trident installation completed by taking a look at the pods that have been created:

```
$ kubectl get pod -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw        5/5     Running   0          1m
trident-csi-mr6zc                   2/2     Running   0          1m
trident-csi-xrp7w                   2/2     Running   0          1m
trident-csi-zh2jt                   2/2     Running   0          1m
trident-operator-766f7b8658-ldzsv   1/1     Running   0          3m
```

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

```
$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

Now you can go ahead and create a backend. See post-deployment tasks.

For troubleshooting issues during deployment, see the troubleshooting section.

Customize Trident operator deployment

The Trident operator enables you to customize the manner in which Astra Trident is installed by using the attributes in the `TridentOrchestrator` spec.

See the following table for the list of attributes:

Parameter	Description	Default
`namespace`	Namespace to install Astra Trident in	"default"
`debug`	Enable debugging for Astra Trident	false
`IPv6`	Install Astra Trident over IPv6	false
`k8sTimeout`	Timeout for Kubernetes operations	30sec
`silenceAutosupport`	Don't send autosupport bundles to NetApp automatically	false
`enableNodePrep`	Manage worker node dependencies automatically (**BETA**)	false
`autosupportImage`	The container image for Autosupport Telemetry	"netapp/trident-autosupport:21.04.0"
`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 1.17+) 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	
`controllerPluginNodeSelector`	Additional node selectors for pods running the Trident Controller CSI Plugin. Follows same format as pod.spec.nodeSelector.	No default; optional

Parameter	Description	Default
`controllerPluginTolerations`	Overrides tolerations for pods running the Trident Controller CSI Plugin. Follows the same format as pod.spec.Tolerations.	No default; optional
`nodePluginNodeSelector`	Additional node selectors for pods running the Trident Node CSI Plugin. Follows same format as pod.spec.nodeSelector.	No default; optional
`nodePluginTolerations`	Overrides tolerations for pods running the Trident Node CSI Plugin. Follows the same format as pod.spec.Tolerations.	No default; optional

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

Automatic worker node prep is a **beta feature** meant to be used in non-production environments only.

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

You can use the attributes mentioned above when defining `TridentOrchestrator` to customize your installation. Here's an example:

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

Here is another example that shows 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
```

If you are looking to customize the installation beyond what `TridentOrchestrator` arguments allow, you should consider using `tridentctl` to generate custom YAML manifests that you can modify as needed.