Deploy with tridentctl

Astra Trident

NetApp
January 03, 2023

Table of Contents

Deploy with `tridentctl`

You can deploy Astra Trident using `tridentctl`. It's a good idea to familiarize yourself with the basic concepts. To customize your `tridentctl` deployment, refer to Customize tridentctl deployment.

Critical information about Astra Trident 22.10

You must read the following critical information before upgrading to Astra Trident 22.10.

Critical information about Astra Trident 22.10

- Kubernetes 1.25 is now supported in Trident. You must upgrade to Astra Trident 22.10 prior to upgrading to Kubernetes 1.25.
- Astra Trident now 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.

Verify prerequisites

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

- Full privileges to a supported Kubernetes cluster.
- 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 points to your Kubernetes cluster configuration.
- The feature gates required by Astra Trident are enabled.
- If you are using Kubernetes with Docker Enterprise, follow their steps to enable CLI access.

Step 1: Qualify your Kubernetes cluster

Log into the Linux host and verify it is managing a working, supported Kubernetes cluster and you have the necessary privileges.

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

To check your Kubernetes version, run the following command:

```
kubectl version
```

To verify 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>
```

Identify your Kubernetes server version. You will use it when you install Astra Trident.

Step 2: Download and extract the installer

The Trident installer creates a Trident pod, configures the CRD objects that are used to maintain its state, and initializes the CSI sidecars that perform actions, such as provisioning and attaching volumes to the cluster hosts.

You can download and extract the latest version of the Trident installer bundle from the *Assets* section on GitHub.

For example, if the latest version is 22.10.0:

```
wget https://github.com/NetApp/trident/releases/download/v22.10.0/trident-
installer-22.10.0.tar.gz
tar -xf trident-installer-22.10.0.tar.gz
cd trident-installer
```

Step 3: Install Astra Trident

Install Astra Trident in the desired namespace by executing the `tridentctl install` command.

```
./tridentctl install -n trident
....
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-csi-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=22.10.0
INFO Trident installation succeeded.
....
```

To enable Astra Trident to run on Windows nodes, add the `--windows` flag to the install command: `$ ./tridentctl install --windows -n trident`.

Output similar to the following displays when the installer is complete. Depending on the number of nodes in your Kubernetes cluster, there might be more pods present:

```
kubectl get pod -n trident
NAME                           READY   STATUS    RESTARTS   AGE
trident-csi-679648bd45-cv2mx   4/4     Running   0          5m29s
trident-csi-vgc8n              2/2     Running   0          5m29s

./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.10.0        | 22.10.0        |
+----------------+----------------+
```

To complete Astra Trident configuration, continue to post-deployment tasks.

If the installer does not complete successfully or `trident-csi-<generated id>` does not have a **Running** status, the platform was not installed.

For troubleshooting issues during deployment, refer to troubleshooting.

Customize `tridentctl` deployment

You can use the Astra Trident installer to customize deployment.

Learn about the installer

The Astra 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]`.

If you are using a distribution of Kubernetes, where `kubelet` keeps its data on a path other than the usual `/var/lib/kubelet`, you can specify the alternate path by using `--kubelet-dir`.

If you need to customize the installation beyond what the installer's arguments allow, you can also customize the deployment files. Using the `--generate-custom-yaml` parameter creates the following YAML files in the installer's `setup` directory:

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`
- `trident-resourcequota.yaml`

After you have generated these files, you can modify them according to your needs and then use `--use-custom-yaml` to install your custom deployment.

```
./tridentctl install -n trident --use-custom-yaml
```

Copyright information

Trademark information