Deployment overview

Astra Trident

amitha
October 20, 2021

Table of Contents

Deployment overview

You can deploy Astra Trident using the Trident operator or with `tridentctl`.

Choose the deployment method

To determine which deployment method to use, consider the following:

Why should I use the Trident operator?

The Trident operator is a great way to dynamically manage Astra Trident resources and automate the setup phase. There are some prerequisites that must be satisfied. See the requirements.

The Trident operator provides several benefits as outlined below.

Self-healing capability

You can monitor an Astra Trident installation and actively take measures to address issues, such as when the deployment is deleted or if it is modified accidentally. When the operator is set up as a deployment, a `trident-operator-<generated-id>` pod is created. This pod associates a `TridentOrchestrator` CR with an Astra Trident installation and always ensures there is only one active `TridentOrchestrator`. In other words, the operator ensures that there is only one instance of Astra 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 Astra 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 triggers the creation of new pods to modify the installation accordingly.

Automatically handles Kubernetes upgrades

When the Kubernetes version of the cluster is upgraded to a supported version, the operator updates an existing Astra 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 Astra Trident. If Astra Trident has already been installed with the operator, a warning is displayed to indicate that Astra Trident is installed on an unsupported Kubernetes version.

Why should I use Helm?

If you have other applications that you are managing using Helm, starting with Astra Trident 21.01, you can manage your deployment also using Helm.

When should I use `tridenctl`?

If you have an existing deployment that must be upgraded to or if you are looking to highly customize your deployment, you should take a look at using tridentctl. This is the conventional method of deploying Astra Trident.

Considerations for moving between deployment methods

It is not hard to imagine a scenario where moving between deployment methods is desired. You should consider the following before attempting to move from a `tridentctl` deployment to an operator-based deployment, or vice-versa:

- Always use the same method for uninstalling Astra Trident. If you have deployed with `tridentctl`, you should use the appropriate version of the `tridentctl` binary to uninstall Astra Trident. Similarly, if you are deploying with the operator, you should edit the `TridentOrchestrator` CR and set `spec.uninstall=true` to uninstall Astra Trident.
- If you have an operator-based deployment that you want to remove and use `tridentctl` to deploy Astra Trident, you should first edit `TridentOrchestrator` and set `spec.uninstall=true` to uninstall Astra 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 do 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 use Helm-based deployment without running into issues.

Understand the deployment modes

There are three ways to deploy Astra Trident.

Standard deployment

Deploying Trident on a Kubernetes cluster results in the Astra Trident installer doing two things:

- Fetching the container images over the Internet
- Creating a deployment and/or node daemonset, which spins up Astra Trident pods on all the eligible nodes in the Kubernetes cluster.

A standard deployment such as this can be performed in two different ways:

- Using `tridentctl install`
- Using the Trident operator. You can deploy Trident operator either manually or by using Helm.

This mode of installing is the easiest way to install Astra Trident and works for most environments that do not

impose network restrictions.

Offline deployment

To perform an air-gapped deployment, you can use the `--image-registry` flag when invoking `tridentctl install` to point to a private image registry. If deploying with the Trident operator, you can alternatively specify `spec.imageRegistry` in your `TridentOrchestrator`. This registry should contain the Trident image, the Trident Autosupport image, and the CSI sidecar images as required by your Kubernetes version.

To customize your deployment, you can use `tridentctl` to generate the manifests for Trident's resources. This includes the deployment, daemonset, service account, and the cluster role that Astra Trident creates as part of its installation.

See these links for more information about customizing your deployment:

- Customize your operator-based deployment
- Customize your `tridentctl`-based deployment

> If you are using a private image repository, you should add `/k8scsi` for Kubernetes versions earlier than 1.17 or `/sig-storage` for Kubernetes versions later than 1.17 to the end of the private registry URL. When using a private registry for `tridentctl` deployment, you should use `--trident-image` and `--autosupport-image` in conjunction with `--image-registry`. If you are deploying Astra Trident by using the Trident operator, ensure that the orchestrator CR includes `tridentImage` and `autosupportImage` in the installation parameters.

Remote deployment

Here is a high-level overview of the remote deployment process:

- Deploy the appropriate version of `kubectl` on the remote machine from where you want to deploy Astra 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.