Manage Astra Trident

Astra Trident

NetApp
September 01, 2022

Table of Contents

Manage Astra Trident

Upgrade Astra Trident

Astra Trident follows a quarterly release cadence, delivering four major releases every calendar year. Each new release builds on top of the previous releases, providing new features and performance enhancements as well as bug fixes and improvements. You are encouraged to upgrade at least once a year to take advantage of the new features in Astra Trident.

Upgrading to a release that is five releases ahead will require you to perform a multistep upgrade.

Determine the version to upgrade to

- You can upgrade to the `YY.MM` release from the `YY-1.MM` release and any in-between releases. For example, you can perform a direct upgrade to 20.07 from 19.07 and later (including dot releases, such as 19.07.1).
- If you have an earlier release, you should perform a multistep upgrade. This requires you to first upgrade to the most recent release that fits your four-release window. For example, if you are running 18.07 and want to upgrade to the 20.07 release, then follow the multistep upgrade process as given below:
  - First upgrade from 18.07 to 19.07. See the documentation of the respective release to obtain specific instructions for upgrading.
  - Then upgrade from 19.07 to 20.07.

All upgrades for versions 19.04 and earlier require the migration of Astra Trident's metadata from it's own `etcd` to CRD objects. Ensure that you check the documentation of the release to understand how the upgrade works.

When upgrading, it is important you provide `parameter.fsType` in `StorageClasses` used by Astra Trident. You can delete and re-create `StorageClasses` without disrupting pre-existing volumes. This is a **requirement** for enforcing security contexts for SAN volumes. The sample input directory contains examples, such as `storage-class-basic.yaml.templ` and `storage-class-bronze-default.yaml`.
For more information, see Known Issues.

Which upgrade path should I choose?

You can upgrade by using one of the following paths:

- Using the Trident operator.
- Using `tridentctl`.

CSI Volume Snapshots is now a feature that is GA, beginning with Kubernetes 1.20. When upgrading Astra Trident, all previous alpha snapshot CRs and CRDs (Volume Snapshot Classes, Volume Snapshots and Volume Snapshot Contents) must be removed before the upgrade is performed. Refer to this blog to understand the steps involved in migrating alpha snapshots to the beta/GA spec.

You can use the Trident operator to upgrade if the following conditions are met:

- You are running CSI Trident (19.07 and later).
- You have a CRD-based Trident release (19.07 and later).
- You are **not** performing a customized install (using custom YAMLs).

> Do not use the operator to upgrade Trident if you are using an `etcd`-based Trident release (19.04 or earlier).

If you do not want to use the operator or you have a customized install that cannot be supported by the operator, you can upgrade by using `tridentctl`. This is the preferred method of upgrades for Trident releases 19.04 and earlier.

Changes to the operator

The 21.01 release of Astra Trident introduces some key architectural changes to the operator, namely the following:

- The operator is now **cluster-scoped**. Previous instances of the Trident operator (versions 20.04 through 20.10) were **namespace-scoped**. An operator that is cluster-scoped is advantageous for the following reasons:
  - Resource accountability: The operator now manages resources associated with an Astra Trident installation at the cluster level. As part of installing Astra Trident, the operator creates and maintains several resources by using `ownerReferences`. Maintaining `ownerReferences` on cluster-scoped resources can throw up errors on certain Kubernetes distributors such as OpenShift. This is mitigated with a cluster-scoped operator. For auto-healing and patching Trident resources, this is an essential requirement.
  - Cleaning up during uninstallation: A complete removal of Astra Trident would require all associated resources to be deleted. A namespace-scoped operator might experience issues with the removal of cluster-scoped resources (such as the clusterRole, ClusterRoleBinding and PodSecurityPolicy) and lead to an incomplete clean-up. A cluster-scoped operator eliminates this issue. Users can completely uninstall Astra Trident and install afresh if needed.
- `TridentProvisioner` is now replaced with `TridentOrchestrator` as the Custom Resource used to install and manage Astra Trident. In addition, a new field is introduced to the `TridentOrchestrator` spec. Users can specify that the namespace Trident must be installed/upgraded from using the `spec.namespace` field. You can take a look at an example here.

Find more information

- Upgrade by using the Trident operator
- Upgrade by using `tridentctl`

Upgrade with the operator

You can easily upgrade an existing Astra Trident installation using the operator.

What you'll need

To upgrade by using the operator, the following conditions should be met:

- You should have a CSI-based Astra Trident installation. To check if you are running CSI Trident, examine the pods in your Trident namespace. If they follow the `trident-csi-*` naming pattern, you are running CSI Trident.
- You should have a CRD-based Trident installation. This represents all releases from 19.07 and later. If you have a CSI-based installation, you most likely have a CRD-based installation.
- If you have uninstalled CSI Trident and the metadata from the installation persists, you can upgrade by using the operator.
- Only one Astra Trident installation should exist across all the namespaces in a given Kubernetes cluster.
- You should be using a Kubernetes cluster that runs version 1.19 -1.24.
- If alpha snapshot CRDs are present, you should remove them with `tridentctl obliviate alpha-snapshot-crd`. This deletes the CRDs for the alpha snapshot spec. For existing snapshots that should be deleted/migrated, see this blog.

When upgrading Trident by using the operator on OpenShift Container Platform, you should upgrade to Trident 21.01.1 or later. The Trident operator released with 21.01.0 contains a known issue that has been fixed in 21.01.1. For more details, see the issue details on GitHub.

Upgrade a cluster-scoped operator installation

To upgrade from **Trident 21.01 and later**, here is the set of steps to be followed.

Steps

1. Delete the Trident operator that was used to install the current Astra Trident instance. For example, if you are upgrading from 21.01, run the following command:

```
kubectl delete -f 21.01/trident-installer/deploy/bundle.yaml -n trident
```

2. (Optional) If you want to modify the installation parameters, edit the `TridentOrchestrator` object that you created when installing Trident.
   This can include changes, such as modifying the custom Trident image, private image registry to pull container images from, enabling debug logs, or specifying image pull secrets.
3. Install Astra Trident by using the `bundle.yaml` file that sets up the Trident operator for the new version. Run the following command:

```
kubectl create -f 21.10.0/trident-installer/deploy/bundle.yaml -n
trident
```

As part of this step, the 21.10.0 Trident operator will identify an existing Astra Trident installation and upgrade it to the same version as the operator.

Upgrade a namespace-scoped operator installation

To upgrade from an instance of Astra Trident installed using the namespace-scoped operator (versions 20.07 through 20.10), here is the set of steps to be followed:

Steps

1. Verify the status of the existing Trident installation. To do this, check the **Status** of `TridentProvisioner`. The status should be `Installed`.

```
$ kubectl describe tprov trident -n trident | grep Message: -A 3
Message:  Trident installed
Status:   Installed
Version:  v20.10.1
```

> If status shows `Updating`, ensure you resolve it before proceeding. For a list of possible status values, see here.

2. Create the `TridentOrchestrator` CRD by using the manifest provided with the Trident installer.

```
# Download the release required [21.01]
$ mkdir 21.07.1
$ cd 21.07.1
$ wget
https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
$ tar -xf trident-installer-21.07.1.tar.gz
$ cd trident-installer
$ kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Delete the namespace-scoped operator by using its manifest. To complete this step, you require the `bundle.yaml` file used to deploy the namespace-scoped operator. You can obtain `bundle.yaml` from the Trident repository. Make sure to use the appropriate branch.

> You should make the necessary changes to the Trident install parameters (for example, changing the values for `tridentImage`, `autosupportImage`, private image repository, and providing `imagePullSecrets`) after deleting the namespace-scoped operator and before installing the cluster-scoped operator. For a complete list of parameters that can be updated, see the list of parameters.

```
#Ensure you are in the right directory
$ pwd
$ /root/20.10.1/trident-installer

#Delete the namespace-scoped operator
$ kubectl delete -f deploy/bundle.yaml
serviceaccount "trident-operator" deleted
clusterrole.rbac.authorization.k8s.io "trident-operator" deleted
clusterrolebinding.rbac.authorization.k8s.io "trident-operator" deleted
deployment.apps "trident-operator" deleted
podsecuritypolicy.policy "tridentoperatorpods" deleted

#Confirm the Trident operator was removed
$ kubectl get all -n trident
NAME                                READY   STATUS    RESTARTS   AGE
pod/trident-csi-68d979fb85-dsrmn    6/6     Running   12         99d
pod/trident-csi-8jfhf               2/2     Running   6          105d
pod/trident-csi-jtnjz               2/2     Running   6          105d
pod/trident-csi-lcxvh               2/2     Running   8          105d

NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
service/trident-csi   ClusterIP   10.108.174.125   <none>
34571/TCP,9220/TCP   105d

NAME                           DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR                                        AGE
daemonset.apps/trident-csi   3         3         3       3            3
kubernetes.io/arch=amd64,kubernetes.io/os=linux   105d

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/trident-csi   1/1     1            1           105d

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/trident-csi-68d979fb85   1         1         1
105d
```

At this stage, the `trident-operator-xxxxxxxxxx-xxxxx` pod is deleted.

4. (Optional) If the install parameters need to be modified, update the `TridentProvisioner` spec. These could be changes such as modifying the private image registry to pull container images from, enabling debug logs, or specifying image pull secrets.

```
$  kubectl patch tprov <trident-provisioner-name> -n <trident-namespace>
--type=merge -p '{"spec":{"debug":true}}'
```

5. Install the cluster-scoped operator.

> Installing the cluster-scoped operator initiates the migration of `TridentProvisioner` objects to `TridentOrchestrator` objects, deletes `TridentProvisioner` objects and the `tridentprovisioner` CRD, and upgrades Astra Trident to the version of the cluster-scoped operator being used. In the example that follows, Trident is upgraded to 21.07.1.

> Upgrading Astra Trident by using the cluster-scoped operator results in the migration of `tridentProvisioner` to a `tridentOrchestrator` object with the same name. This is automatically handled by the operator. The upgrade will also have Astra Trident installed in the same namespace as before.

```
#Ensure you are in the correct directory
$ pwd
$ /root/21.07.1/trident-installer

#Install the cluster-scoped operator in the **same namespace**
$ kubectl create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#All tridentProvisioners will be removed, including the CRD itself
$ kubectl get tprov -n trident
Error from server (NotFound): Unable to list "trident.netapp.io/v1,
Resource=tridentprovisioners": the server could not find the requested
resource (get tridentprovisioners.trident.netapp.io)

#tridentProvisioners are replaced by tridentOrchestrator
$ kubectl get torc
NAME      AGE
trident   13s

#Examine Trident pods in the namespace
$ kubectl get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m41s
trident-csi-xrst8                   2/2     Running   0          1m41s
trident-operator-5574dbbc68-nthjv   1/1     Running   0          1m52s

#Confirm Trident has been updated to the desired version
$ kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v21.07.1
```

Upgrade a Helm-based operator installation

Perform the following steps to upgrade a Helm-based operator installation.

Steps

1. Download the latest Astra Trident release.
2. Use the `helm upgrade` command. See the following example:

```
$ helm upgrade <name> trident-operator-21.07.1.tgz
```

where `trident-operator-21.07.1.tgz` reflects the version that you want to upgrade to.

3. Run `helm list` to verify that the chart and app version have both been upgraded.

> To pass configuration data during the upgrade, use `--set`.

For example, to change the default value of `tridentDebug`, run the following command:

```
$ helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true
```

If you run `$ tridentctl logs`, you can see the debug messages.

> If you set any non-default options during the initial installation, ensure that the options are included in the upgrade command, or else, the values will be reset to their defaults.

Upgrade from a non-operator installation

If you have a CSI Trident instance that meets the prerequisites listed above, you can upgrade to the latest release of the Trident operator.

Steps

1. Download the latest Astra Trident release.

```
# Download the release required [21.07.1]
$ mkdir 21.07.1
$ cd 21.07.1
$ wget
https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
$ tar -xf trident-installer-21.07.1.tar.gz
$ cd trident-installer
```

2. Create the `tridentorchestrator` CRD from the manifest.

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

3. Deploy the operator.

```
#Install the cluster-scoped operator in the **same namespace**
$ kubectl create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          150d
trident-csi-xrst8                   2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv   1/1     Running   0          1m30s
```

4. Create a `TridentOrchestrator` CR for installing Astra Trident.

```
#Create a tridentOrchestrator to initate a Trident install
$ cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

$ kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s

#Confirm Trident was upgraded to the desired version
$ kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v21.07.1
```

The existing backends and PVCs are automatically available.

Upgrade with `tridentctl`

You can easily upgrade an existing Astra Trident installation by using `tridentctl`.

Considerations

When upgrading to the latest release of Astra Trident, consider the following:

- Starting with Trident 20.01, only the beta release of volume snapshots is supported. Kubernetes administrators should take care to safely back up or convert the alpha snapshot objects to beta to retain the legacy alpha snapshots.
- The beta release of volume snapshots introduces a modified set of CRDs and a snapshot controller, both of which should be set up before installing Astra Trident.

This blog discusses the steps involved in migrating alpha volume snapshots to the beta format.

About this task

Uninstalling and reinstalling Astra Trident acts as an upgrade. When you uninstall Trident, the Persistent Volume Claim (PVC) and Persistent Volume (PV) used by the Astra Trident deployment are not deleted. PVs that have already been provisioned will remain available while Astra Trident is offline, and Astra Trident will provision volumes for any PVCs that are created in the interim once it is back online.

When upgrading Astra Trident, do not interrupt the upgrade process. Ensure that the installer runs to completion.

Next steps after upgrade

To make use of the rich set of features that are available in newer Trident releases (such as, On-Demand Volume Snapshots), you can upgrade the volumes by using the `tridentctl upgrade` command.

If there are legacy volumes, you should upgrade them from a NFS/iSCSI type to the CSI type to be able to use the complete set of new features in Astra Trident. A legacy PV that has been provisioned by Trident supports the traditional set of features.

Consider the following when deciding to upgrade volumes to the CSI type:

- You might not need to upgrade all the volumes. Previously created volumes will continue to be accessible and function normally.
- A PV can be mounted as part of a deployment/StatefulSet when upgrading. It is not required to bring down the deployment/StatefulSet.
- You **cannot** attach a PV to a standalone pod when upgrading. You should shut down the pod before upgrading the volume.
- You can upgrade only a volume that is bound to a PVC. Volumes that are not bound to PVCs should be removed and imported before upgrading.

Volume upgrade example

Here is an example that shows how a volume upgrade is performed.

1. Run `kubectl get pv` to list the PVs.

```
$ kubectl get pv
NAME                        CAPACITY     ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                   STORAGECLASS    REASON   AGE
default-pvc-1-a8475         1073741824   RWO            Delete
Bound    default/pvc-1          standard                19h
default-pvc-2-a8486         1073741824   RWO            Delete
Bound    default/pvc-2          standard                19h
default-pvc-3-a849e         1073741824   RWO            Delete
Bound    default/pvc-3          standard                19h
default-pvc-4-a84de         1073741824   RWO            Delete
Bound    default/pvc-4          standard                19h
trident                     2Gi          RWO            Retain
Bound    trident/trident                                19h
```

There are currently four PVs that have been created by Trident 20.07, using the `netapp.io/trident` provisioner.

2. Run `kubectl describe pv` to get the details of the PV.

```
$ kubectl describe pv default-pvc-2-a8486

Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: netapp.io/trident
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:      NFS (an NFS mount that lasts the lifetime of a pod)
    Server:    10.xx.xx.xx
    Path:      /trid_1907_alpha_default_pvc_2_a8486
    ReadOnly:  false
```

The PV was created by using the `netapp.io/trident` provisioner and is of the type NFS. To support all the new features provided by Astra Trident, this PV should be upgraded to the CSI type.

3. Run the `tridenctl upgrade volume <name-of-trident-volume>` command to upgrade a legacy

Astra Trident volume to the CSI spec.

```
$ ./tridentctl get volumes -n trident
+--------------------+---------+--------------+----------
+--------------------------------------+--------+---------+
|            NAME     |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+--------------------+---------+--------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-3-a849e | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-1-a8475 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-4-a84de | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+--------------------+---------+--------------+----------
+--------------------------------------+--------+---------+

$ ./tridentctl upgrade volume default-pvc-2-a8486 -n trident
+--------------------+---------+--------------+----------
+--------------------------------------+--------+---------+
|            NAME     |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+--------------------+---------+--------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+--------------------+---------+--------------+----------
+--------------------------------------+--------+---------+
```

4. Run a `kubectl describe pv` to verify that the volume is a CSI volume.

```
$ kubectl describe pv default-pvc-2-a8486
Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume
source)
    Driver:            csi.trident.netapp.io
    VolumeHandle:      default-pvc-2-a8486
    ReadOnly:          false
    VolumeAttributes:      backendUUID=c5a6f6a4-b052-423b-80d4-
8fb491a14a22

internalName=trid_1907_alpha_default_pvc_2_a8486
                           name=default-pvc-2-a8486
                           protocol=file
Events:                <none>
```

In this manner, you can upgrade volumes of the NFS/iSCSI type that were created by Astra Trident to the CSI type, on a per-volume basis.

Uninstall Astra Trident

Depending on how Astra Trident is installed, there are multiple options to uninstall it.

Uninstall by using Helm

If you installed Astra Trident by using Helm, you can uninstall it by using `helm uninstall`.

```
#List the Helm release corresponding to the Astra Trident install.
$ helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
$ helm uninstall trident -n trident
release "trident" uninstalled
```

Uninstall by using the Trident operator

If you installed Astra Trident by using the operator, you can uninstall it by doing one of the following:

- **Edit `TridentOrchestrator` to set the uninstall flag:** You can edit `TridentOrchestrator` and set `spec.uninstall=true`. Edit the `TridentOrchestrator` CR and set the `uninstall` flag as shown below:

  ```
  $  kubectl patch torc <trident-orchestrator-name> --type=merge -p
  '{"spec":{"uninstall":true}}'
  ```

  When the `uninstall` flag is set to `true`, the Trident operator uninstalls Trident, but does not remove the TridentOrchestrator itself. You should clean up the TridentOrchestrator and create a new one if you want to install Trident again.

- **Delete `TridentOrchestrator`:** By removing the `TridentOrchestrator` CR that was used to deploy Astra Trident, you instruct the operator to uninstall Trident. The operator processes the removal of `TridentOrchestrator` and proceeds to remove the Astra Trident deployment and daemonset, deleting the Trident pods it had created as part of the installation.
  To completely remove Astra Trident (including the CRDs it creates) and effectively wipe the slate clean, you can edit `TridentOrchestrator` to pass the `wipeout` option. See the following example:

```
$ kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

This uninstalls Astra Trident completely and clears all metadata related to the backends and volumes it manages. Subsequent installations are treated as fresh installations.

> You should only consider wiping out the CRDs when performing a complete uninstallation. This cannot be undone. **Do not wipe out the CRDs unless you are looking to start over and create a fresh Astra Trident installation**.

Uninstall by using `tridentctl`

Run the `uninstall` command in `tridentctl` as follows to removes all of the resources associated with Astra Trident except for the CRDs and related objects, thereby making it easy to run the installer again to update to a more recent version.

```
./tridentctl uninstall -n <namespace>
```

To perform a complete removal of Astra Trident, you should remove the finalizers for the CRDs created by Astra Trident and delete the CRDs.

Downgrade Astra Trident

Learn about the steps involved in downgrading to an earlier version of Astra Trident.

You might consider downgrading for various reasons, such as the following:

- Contingency planning
- Immediate fix for bugs observed as a result of an upgrade
- Dependency issues, unsuccessful and incomplete upgrades

When to downgrade

You should consider a downgrade when moving to a Astra Trident release that uses CRDs. Because Astra Trident now uses CRDs for maintaining state, all storage entities created (backends, storage classes, PV, and volume snapshots) have associated CRD objects instead of data written into the `trident` PV (used by the earlier installed version of Astra Trident). Newly created PVs, backends, and storage classes are all maintained as CRD objects. If you need to downgrade, this should only be attempted for a version of Astra Trident that runs using CRDs (19.07 and later). This is to ensure that all the operations performed on the current Astra Trident release are visible after the downgrade occurs.

When not to downgrade

You should not downgrade to a release of Trident that uses `etcd` to maintain state (19.04 and earlier). All operations performed with the current Astra Trident release are not reflected after the downgrade. Newly created PVs are not usable when moving back to an earlier version. Changes made to objects such as backends, PVs, storage classes, and volume snapshots (created/updated/deleted) are not visible to Astra Trident when moving back to an earlier version. Going back to an earlier version does not disrupt access for PVs that were already created by using the older release, unless they have been upgraded.

Downgrade process when Astra Trident is installed by using the operator

For installations done using the Trident Operator, the downgrade process is different and does not require the use of `tridentctl`.

For installations done using the Trident operator, Astra Trident can be downgraded to either of the following:

- A version that is installed using the namespace-scoped operator (20.07 - 20.10).
- A version that is installed using the cluster-scoped operator (21.01 and later).

Downgrade to cluster-scoped operator

To downgrade Astra Trident to a release that uses the cluster-scoped operator, follow the steps mentioned below.

Steps

1. Uninstall Astra Trident. **Do not wipeout the CRDs unless you want to completely remove an existing installation.**
2. Delete the cluster-scoped operator. To do this, you will need the manifest used to deploy the operator. You can obtain it from the Trident GitHub repo. Make sure you switch to the required branch.
3. Continue downgrading by installing the desired version of Astra Trident. Follow the documentation for the desired release.

Downgrade to namespace-scoped operator

This section summarizes the steps involved in downgrading to an Astra Trident release that falls in the range 20.07 through 20.10, which will be installed using the namespace-scoped operator.

Steps

1. Uninstall Astra Trident. **Do not wipeout the CRDs unless you want to completely remove an existing installation.**
   Make sure the `tridentorchestrator` is deleted.

```
#Check to see if there are any tridentorchestrators present
$ kubectl get torc
NAME        AGE
trident     20h

#Looks like there is a tridentorchestrator that needs deleting
$ kubectl delete torc trident
tridentorchestrator.trident.netapp.io "trident" deleted
```

2. Delete the cluster-scoped operator. To do this, you will need the manifest used to deploy the operator. You can obtain it here from the Trident GitHub repo. Make sure you switch to the required branch.
3. Delete the `tridentorchestrator` CRD.

```
#Check to see if ``tridentorchestrators.trident.netapp.io`` CRD is
present and delete it.
$ kubectl get crd tridentorchestrators.trident.netapp.io
NAME                                        CREATED AT
tridentorchestrators.trident.netapp.io   2021-01-21T21:11:37Z
$ kubectl delete crd tridentorchestrators.trident.netapp.io
customresourcedefinition.apiextensions.k8s.io
"tridentorchestrators.trident.netapp.io" deleted
```

   Astra Trident has been uninstalled.

4. Continue downgrading by installing the desired version. Follow the documentation for the desired release.

Downgrade by using Helm

To downgrade, use the `helm rollback` command. See the following example:

```
$ helm rollback trident [revision #]
```

Downgrade process when Astra Trident is installed by using `tridentctl`

If you installed Astra Trident by using `tridentctl`, the downgrade process involves the following steps. This sequence walks you through the downgrade process to move from Astra Trident 21.07 to 20.07.

> Before beginning the downgrade, you should take a snapshot of your Kubernetes cluster's `etcd`. This enables you to back up the current state of Astra Trident's CRDs.

Steps

1. Make sure that Trident is installed by using `tridentctl`. If you are unsure about how Astra Trident is installed, run this simple test:
   a. List the pods present in the Trident namespace.
   b. Identify the version of Astra Trident running in your cluster. You can either use `tridentctl` or take a look at the image used in the Trident pods.
   c. If you **do not see** a `tridentOrchestrator`, (or) a `tridentprovisioner`, (or) a pod named `trident-operator-xxxxxxxxxx-xxxxx`, Astra Trident **is installed** with `tridentctl`.
2. Uninstall Astra Trident with the existing `tridentctl` binary. In this case, you will uninstall with the 21.07 binary.

```
$ tridentctl version -n trident
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.0        | 21.07.0        |
+----------------+----------------+

$ tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted Trident daemonset.
INFO Deleted Trident service.
INFO Deleted Trident secret.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Deleted pod security policy.
podSecurityPolicy=tridentpods
INFO The uninstaller did not delete Trident's namespace in case it is
going to be reused.
INFO Trident uninstallation succeeded.
```

3. After this is complete, obtain the Trident binary for the desired version (in this example, 20.07), and use it to install Astra Trident. You can generate custom YAMLs for a customized installation if needed.

```
$ cd 20.07/trident-installer/
$ ./tridentctl install -n trident-ns
INFO Created installer service account.
serviceaccount=trident-installer
INFO Created installer cluster role.               clusterrole=trident-
installer
INFO Created installer cluster role binding.
clusterrolebinding=trident-installer
INFO Created installer configmap.                  configmap=trident-
installer
...
...
INFO Deleted installer cluster role binding.
INFO Deleted installer cluster role.
INFO Deleted installer service account.
```

The downgrade process is complete.