Troubleshooting

Astra Trident

NetApp
January 28, 2023

Table of Contents

Troubleshooting

Use the pointers provided here for troubleshooting issues you might encounter while installing and using Astra Trident.

For help with Astra Trident, create a support bundle using `tridentctl logs -a -n trident` and send it to `NetApp Support <Getting Help>`.

For a comprehensive list of troubleshooting articles, see the NetApp Knowledgebase (login required). You can also find information about troubleshooting issues related to Astra here.

General troubleshooting

- If the Trident pod fails to come up properly (for example, when the Trident pod is stuck in the `ContainerCreating` phase with fewer than two ready containers), running `kubectl -n trident describe deployment trident` and `kubectl -n trident describe pod trident-**` can provide additional insights. Obtaining kubelet logs (for example, via `journalctl -xeu kubelet`) can also be helpful.
- If there is not enough information in the Trident logs, you can try enabling the debug mode for Trident by passing the `-d` flag to the install parameter based on your installation option.

  Then confirm debug is set using `./tridentctl logs -n trident` and searching for `level=debug msg` in the log.

  Installed with Operator

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

  This will restart all Trident pods, which can take several seconds. You can check this by observing the 'AGE' column in the output of `kubectl get pod -n trident`.

  For Astra Trident 20.07 and 20.10 use `tprov` in place of `torc`.

  Installed with Helm

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

  Installed with tridentctl

  ```
  ./tridentctl uninstall -n trident
  ./tridentctl install -d -n trident
  ```

- You can also obtain debug logs for each backend by including `debugTraceFlags` in your backend definition. For example, include `debugTraceFlags: {"api":true, "method":true,}` to obtain API

calls and method traversals in the Trident logs. Existing backends can have `debugTraceFlags` configured with a `tridentctl backend update`.

- When using RedHat CoreOS, ensure that `iscsid` is enabled on the worker nodes and started by default. This can be done using OpenShift MachineConfigs or by modifying the ignition templates.
- A common problem you could encounter when using Trident with Azure NetApp Files is when the tenant and client secrets come from an app registration with insufficient permissions. For a complete list of Trident requirements, see Azure NetApp Files configuration.
- If there are problems with mounting a PV to a container, ensure that `rpcbind` is installed and running. Use the required package manager for the host OS and check if `rpcbind` is running. You can check the status of the `rpcbind` service by running a `systemctl status rpcbind` or its equivalent.
- If a Trident backend reports that it is in the `failed` state despite having worked before, it is likely caused by changing the SVM/admin credentials associated with the backend. Updating the backend information using `tridentctl update backend` or bouncing the Trident pod will fix this issue.
- If you are upgrading your Kubernetes cluster and/or Trident to use beta Volume Snapshots, ensure that all the existing alpha snapshot CRs are completely removed. You can then use the `tridentctl obliviate alpha-snapshot-crd` command to delete alpha snapshot CRDs. See this blog to understand the steps involved in migrating alpha snapshots.
- If you encounter permission issues when installing Trident with Docker as the container runtime, attempt the installation of Trident with the `--in cluster=false` flag. This will not use an installer pod and avoid permission troubles seen due to the `trident-installer` user.
- Use the `uninstall parameter <Uninstalling Trident>` for cleaning up after a failed run. By default, the script does not remove the CRDs that have been created by Trident, making it safe to uninstall and install again even in a running deployment.
- If you are looking to downgrade to an earlier version of Trident, first run the `tridenctl uninstall` command to remove Trident. Download the desired Trident version and install using the `tridentctl install` command. Only consider a downgrade if there are no new PVs created and if no changes have been made to already existing PVs/backends/ storage classes. Since Trident now uses CRDs for maintaining state, all storage entities created (backends, storage classes, PVs and Volume Snapshots) have `associated CRD objects <Kubernetes CustomResourceDefinition Objects>` instead of data written into the PV that was used by the earlier installed version of Trident. **Newly created PVs will not be usable when moving back to an earlier version. Changes made to objects, such as backends, PVs, storage classes, and volume snapshots (created/updated/deleted) will not be visible to Trident when downgraded**. The PV that was used by the earlier version of Trident installed will still be visible to Trident. Going back to an earlier version will not disrupt access for PVs that were already created using the older release, unless they have been upgraded.
- To completely remove Trident, run the `tridentctl obliviate crd` command. This will remove all CRD objects and undefine the CRDs. Trident will no longer manage any PVs it had already provisioned.

  Trident will need to be reconfigured from scratch after this.

- After a successful install, if a PVC is stuck in the `Pending` phase, running `kubectl describe pvc` can provide additional information about why Trident failed to provision a PV for this PVC.

Troubleshooting an unsuccessful Trident deployment using the operator

If you are deploying Trident using the operator, 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 by running following command:

```
tridentctl logs -l trident-operator
```

Trailing the logs of the trident-operator container can point to where the problem lies. For example, one such issue could be the inability to pull the required container images from upstream registries in an airgapped environment.

To understand why the installation of Trident was unsuccessful, you should take a look at the `TridentOrchestrator` status.

```
kubectl describe torc trident-2
Name:         trident-2
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Trident is bound to another CR 'trident'
  Namespace:                    trident-2
  Status:                       Error
  Version:
Events:
  Type     Reason  Age                From                          Message
  ----     ------  ----               ----                          -------
  Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'
```

This error indicates that there already exists a `TridentOrchestrator` that was used to install Trident. Since each Kubernetes cluster can only have one instance of Trident, the operator ensures that at any given time there only exists one active `TridentOrchestrator` that it can create.

In addition, observing the status of the Trident pods can often indicate if something is not right.

```
kubectl get pods -n trident

NAME                                READY   STATUS             RESTARTS
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

You can clearly see that the pods are not able to initialize completely because one or more container images were not fetched.

To address the problem, you should edit the `TridentOrchestrator` CR. Alternatively, you can delete `TridentOrchestrator`, and create a new one with the modified and accurate definition.

Troubleshooting an unsuccessful Trident deployment using `tridentctl`

To help figure out what went wrong, you could run the installer again using the `-d` argument, which will turn on debug mode and help you understand what the problem is:

```
./tridentctl install -n trident -d
```

After addressing the problem, you can clean up the installation as follows, and then run the `tridentctl install` command again:

```
./tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Removed Trident user from security context constraint.
INFO Trident uninstallation succeeded.
```

Copyright information

Trademark information