



What's next?

Astra Trident

amitha
October 20, 2021

Table of Contents

- What's next? 1
- Step 1: Create a backend 1
- Step 2: Create a storage class 1
- Step 3: Provision your first volume 3
- Step 4: Mount the volumes in a pod 4

What's next?

After you deploy Astra Trident, you can proceed with creating a backend, creating a storage class, provisioning a volume, and mounting the volume in a pod.

Step 1: Create a backend

You can now go ahead and create a backend that will be used by Astra Trident to provision volumes. To do this, create a `backend.json` file that contains the necessary parameters. Sample configuration files for different backend types can be found in the `sample-input` directory.

See [here](#) for more details about how to configure the file for your backend type.

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

```
./tridentctl -n trident create backend -f backend.json
+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |          UUID          |
STATE | VOLUMES |
+-----+-----+-----+
+-----+-----+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |         0 |
+-----+-----+-----+
+-----+-----+
```

If the creation fails, something was wrong with the backend configuration. You can view the logs to determine the cause by running the following command:

```
./tridentctl -n trident logs
```

After you address the problem, simply go back to the beginning of this step and try again. For more troubleshooting tips, see [the troubleshooting](#) section.

Step 2: Create a storage class

Kubernetes users provision volumes by using persistent volume claims (PVCs) that specify a [storage class](#) by name. The details are hidden from the users, but a storage class identifies the provisioner that is used for that class (in this case, Trident), and what that class means to the provisioner.

Create a storage class Kubernetes users will specify when they want a volume. The configuration of the class needs to model the backend that you created in the previous step, so that Astra Trident will use it to provision new volumes.

The simplest storage class to start with is one based on the `sample-input/storage-class-csi.yaml.templ` file that comes with the installer, replacing `BACKEND_TYPE` with the storage driver name.

```
./tridentctl -n trident get backend
+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+
+-----+-----+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |         0 |
+-----+-----+-----+
+-----+-----+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

This is a Kubernetes object, so you use `kubectl` to create it in Kubernetes.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

You should now see a **basic-csi** storage class in both Kubernetes and Astra Trident, and Astra Trident should have discovered the pools on the backend.

```

kubect1 get sc basic-csi
NAME          PROVISIONER          AGE
basic-csi    csi.trident.netapp.io 15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}

```

Step 3: Provision your first volume

Now you are ready to dynamically provision your first volume. This is done by creating a Kubernetes [persistent volume claim](#) (PVC) object.

Create a PVC for a volume that uses the storage class that you just created.

See `sample-input/pvc-basic-csi.yaml` for an example. Make sure the storage class name matches the one that you created.

```
kubectl create -f sample-input/pvc-basic-csi.yaml
```

```
kubectl get pvc --watch
```

NAME	STATUS	VOLUME	CAPACITY
basic	Pending		
basic	1s		
basic	Pending	pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7	0
basic	5s		
basic	Bound	pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7	1Gi
RWO	basic	7s	

Step 4: Mount the volumes in a pod

Now let us mount the volume. We will launch an nginx pod that mounts the PV under `/usr/share/nginx/html`.

```
cat << EOF > task-pv-pod.yaml
```

```
kind: Pod
```

```
apiVersion: v1
```

```
metadata:
```

```
  name: task-pv-pod
```

```
spec:
```

```
  volumes:
```

```
    - name: task-pv-storage
```

```
      persistentVolumeClaim:
```

```
        claimName: basic
```

```
  containers:
```

```
    - name: task-pv-container
```

```
      image: nginx
```

```
      ports:
```

```
        - containerPort: 80
```

```
          name: "http-server"
```

```
      volumeMounts:
```

```
        - mountPath: "/usr/share/nginx/html"
```

```
          name: task-pv-storage
```

```
EOF
```

```
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

At this point, the pod (application) no longer exists but the volume is still there. You can use it from another pod if you want to.

To delete the volume, delete the claim:

```
kubectl delete pvc basic
```

You can now do additional tasks, such as the following:

- [Configure additional backends.](#)
- [Create additional storage classes.](#)

Copyright Information

Copyright © 2021 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means-graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system-without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

Trademark Information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.