



NetApp Astra Trident Overview

NetApp Solutions

NetApp
May 09, 2022

This PDF was generated from https://docs.netapp.com/us-en/netapp-solutions/containers/anthos-with-netapp/a-w-n_trident_ontap_nfs.html on May 09, 2022. Always check docs.netapp.com for the latest.

Table of Contents

- Astra Trident Overview 1
 - Download Astra Trident 1
 - Install the Trident Operator with Helm 2
 - Manually install the Trident Operator 4
 - Create storage-system backends 8
 - NetApp ONTAP NFS configuration: Anthos with NetApp 8
 - NetApp ONTAP iSCSI configuration: Anthos with NetApp 11
 - NetApp Element iSCSI configuration: Anthos with NetApp 13

Astra Trident Overview

Astra Trident is an open-source and fully supported storage orchestrator for containers and Kubernetes distributions, including Anthos. Trident works with the entire NetApp storage portfolio, including the NetApp ONTAP and Element storage systems, and it also supports NFS and iSCSI connections. Trident accelerates the DevOps workflow by allowing end users to provision and manage storage from their NetApp storage systems without requiring intervention from a storage administrator.

An administrator can configure a number of storage backends based on project needs and storage system models that enable advanced storage features, including compression, specific disk types, or QoS levels that guarantee a certain level of performance. After they are defined, these backends can be used by developers in their projects to create persistent volume claims (PVCs) and to attach persistent storage to their containers on demand.

[Error: Missing Graphic Image]

Astra Trident has a rapid development cycle and, like Kubernetes, is released four times a year.

The latest version of Astra Trident, 22.01, was released in January 2022. A support matrix for what version of Trident has been tested with which Kubernetes distribution can be found [here](#).

Starting with the 20.04 release, Trident setup is performed by the Trident operator. The operator makes large scale deployments easier and provides additional support including self healing for pods that are deployed as a part of the Trident install.

With the 21.01 release, a Helm chart was made available to ease the installation of the Trident Operator.

Download Astra Trident

To install Trident on the deployed user cluster and provision a persistent volume, complete the following steps:

1. Download the installation archive to the admin workstation and extract the contents. The current version of Trident is 22.01, which can be downloaded [here](#).

```
[ubuntu@gke-admin-ws-2021-07-15 ~]$ wget
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
--2021-05-06 15:17:30--
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
```

```

30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
[following]
--2021-05-06 15:17:30--  https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
Resolving github-releases.githubusercontent.com (github-
releases.githubusercontent.com)... 185.199.108.154, 185.199.109.154,
185.199.110.154, ...
Connecting to github-releases.githubusercontent.com (github-
releases.githubusercontent.com)|185.199.108.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38349341 (37M) [application/octet-stream]
Saving to: 'trident-installer-22.01.0.tar.gz'

100%[=====
=====>] 38,349,341  88.5MB/s
in 0.4s

2021-05-06 15:17:30 (88.5 MB/s) - 'trident-installer-22.01.0.tar.gz'
saved [38349341/38349341]

```

2. Extract the Trident install from the downloaded bundle.

```

[ubuntu@gke-admin-ws-2021-07-15 ~]$ tar -xzf trident-installer-
22.01.0.tar.gz
[ubuntu@gke-admin-ws-2021-07-15 ~]$ cd trident-installer/
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$

```

Install the Trident Operator with Helm



Helm is not installed by default on the GKE-Admin workstation. You can easily install it using the apt tool available in Ubuntu.

1. First, set the location of the user cluster's `kubeconfig` file as an environment variable so that you don't have to reference it, because Trident has no option to pass this file.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ export
KUBECONFIG=~/.user-cluster-1/user-cluster-1-kubeconfig
```

2. Run the Helm command to install the Trident operator from the tarball in the helm directory while creating the trident namespace in your user cluster.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ helm install trident
helm/trident-operator-22.01.0.tgz --create-namespace --namespace trident
NAME: trident
LAST DEPLOYED: Fri May  7 12:54:25 2021
NAMESPACE: trident
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing trident-operator, which will deploy and manage
NetApp's Trident CSI
storage provisioner for Kubernetes.

Your release is named 'trident' and is installed into the 'trident'
namespace.
Please note that there must be only one instance of Trident (and
trident-operator) in a Kubernetes cluster.

To configure Trident to manage storage resources, you will need a copy
of tridentctl, which is
available in pre-packaged Trident releases. You may find all Trident
releases and source code
online at https://github.com/NetApp/trident.

To learn more about the release, try:

$ helm status trident
$ helm get all trident
```

3. You can verify that Trident is successfully installed by checking the pods that are running in the namespace or by using the `tridentctl` binary to check the installed version.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ oc get pods -n
trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-5z45l                   1/2     Running   2           30s
trident-csi-696b685cf8-htdb2       6/6     Running   0           30s
trident-csi-b74p2                   2/2     Running   0           30s
trident-csi-lrw4n                   2/2     Running   0           30s
trident-operator-7c748d957-gr2gw    1/1     Running   0           36s
```

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ ./tridentctl -n
trident version
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 22.01.0        | 22.01.0        |
+-----+-----+
```



In some cases, customer environments might require the customization of the Trident deployment. In these cases, it is also possible to manually install the Trident operator and update the included manifests to customize the deployment.

Manually install the Trident Operator

1. First, set the location of the user cluster's `kubeconfig` file as an environment variable so that you don't have to reference it, because Trident has no option to pass this file.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ KUBECONFIG=~/.user-
cluster-1/user-cluster-1-kubeconfig
```

2. The `trident-installer` directory contains manifests for defining all the required resources. Using the appropriate manifests, create the `TridentOrchestrator` custom resource definition.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.tride
nt.netapp.io created
```

3. If one does not exist, create a Trident namespace in your cluster using the provided manifest.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl apply -f
deploy/namespace.yaml
namespace/trident created
```

4. Create the resources required for the Trident operator deployment, such as a ServiceAccount for the operator, a ClusterRole and ClusterRoleBinding to the ServiceAccount, a dedicated PodSecurityPolicy, or the operator itself.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ 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
```

5. You can check the status of the operator after it's deployed with the following commands:

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl get
deployment -n trident
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
trident-operator    1/1      1             1            23s
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl get pods -n
trident
NAME                                READY    STATUS    RESTARTS    AGE
trident-operator-66f48895cc-lzczk    1/1      Running   0           41s
```

6. With the operator deployed, we can now use it to install Trident. This requires creating a TridentOrchestrator.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl describe
torc trident
Name:                trident
Namespace:
Labels:              <none>
Annotations:         <none>
API Version:         trident.netapp.io/v1
Kind:                TridentOrchestrator
Metadata:
  Creation Timestamp:  2021-05-07T17:00:28Z
  Generation:         1
  Managed Fields:
    API Version:       trident.netapp.io/v1
    Fields Type:       FieldsV1
    fieldsV1:
```

```

f:spec:
  ..
  f:debug:
  f:namespace:
Manager:      kubect1-create
Operation:    Update
Time:         2021-05-07T17:00:28Z
API Version:  trident.netapp.io/v1
Fields Type:  FieldsV1
fieldsV1:
  f:status:
    ..
    f:currentInstallationParams:
      ..
      f:IPv6:
      f:autosupportHostname:
      f:autosupportImage:
      f:autosupportProxy:
      f:autosupportSerialNumber:
      f:debug:
      f:enableNodePrep:
      f:imagePullSecrets:
      f:imageRegistry:
      f:k8sTimeout:
      f:kubeletDir:
      f:logFormat:
      f:silenceAutosupport:
      f:tridentImage:
    f:message:
    f:namespace:
    f:status:
    f:version:
Manager:      trident-operator
Operation:    Update
Time:         2021-05-07T17:00:28Z
Resource Version:  931421
Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
UID:          8a26a7a6-dde8-4d55-9b66-a7126754d81f
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:          false
    Autosupport Hostname:

```



```

Autosupport Image:      netapp/trident-autosupport:21.01
Autosupport Proxy:
Autosupport Serial Number:
Debug:                  true
Enable Node Prep:       false
Image Pull Secrets:
Image Registry:
k8sTimeout:            30
Kubelet Dir:            /var/lib/kubelet
Log Format:             text
Silence Autosupport:    false
Trident Image:          netapp/trident:22.01.0
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v22.01.0
Events:
  Type    Reason      Age   From                                Message
  ----    -
  Normal  Installing  80s   trident-operator.netapp.io         Installing
Trident
  Normal  Installed   68s   trident-operator.netapp.io         Trident
installed

```

7. You can verify that Trident is successfully installed by checking the pods that are running in the namespace or by using the tridentctl binary to check the installed version.

```

[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl get pods -n
trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-bb64c6cb4-lmd6h         6/6     Running   0           82s
trident-csi-gn59q                   2/2     Running   0           82s
trident-csi-m4szj                   2/2     Running   0           82s
trident-csi-sb9k9                   2/2     Running   0           82s
trident-operator-66f48895cc-lzczk    1/1     Running   0           2m39s

[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ ./tridentctl -n
trident version
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 22.01.0       | 22.01.0       |
+-----+-----+

```

Create storage-system backends

After completing the Astra Trident Operator install, you must configure the backend for the specific NetApp storage platform you are using. Follow the links below in order to continue the setup and configuration of Astra Trident.

- [NetApp ONTAP NFS](#)
- [NetApp ONTAP iSCSI](#)
- [NetApp Element iSCSI](#)

Next: [Advanced Configuration Options: Anthos with NetApp](#).

NetApp ONTAP NFS configuration: Anthos with NetApp

To enable Trident integration with the NetApp ONTAP storage system, you must create a backend that enables communication with the storage system.

1. There are sample backend files available in the downloaded installation archive in the `sample-input` folder hierarchy. For NetApp ONTAP systems serving NFS, copy the `backend-ontap-nas.json` file to your working directory and edit the file.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ cp sample-  
input/backends-samples/ontap-nas/backend-ontap-nas.json ./  
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ vi backend-ontap-  
nas.json
```

2. Edit the `backendName`, `managementLIF`, `dataLIF`, `svm`, `username`, and `password` values in this file.

```
{  
  "version": 1,  
  "storageDriverName": "ontap-nas",  
  "backendName": "ontap-nas+10.61.181.221",  
  "managementLIF": "172.21.224.201",  
  "dataLIF": "10.61.181.221",  
  "svm": "trident_svm",  
  "username": "cluster-admin",  
  "password": "password"  
}
```



It is a best practice to define the custom `backendName` value as a combination of the `storageDriverName` and the `dataLIF` that is serving NFS for easy identification.

3. With this backend file in place, run the following command to create your first backend.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ ./tridentctl -n
trident create backend -f backend-ontap-nas.json
```

NAME	STATE	VOLUMES	STORAGE DRIVER	UUID
ontap-nas+10.61.181.221	online	0	ontap-nas	be7a619d-c81d-445c-b80c-5c87a73c5b1e

4. With the backend created, you must next create a storage class. Just as with the backend, there is a sample storage class file that can be edited for the environment available in the sample-inputs folder. Copy it to the working directory and make necessary edits to reflect the backend created.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ cp sample-
input/storage-class-samples/storage-class-csi.yaml.tmpl ./storage-
class-basic.yaml
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ vi storage-class-
basic.yaml
```

5. The only edit that must be made to this file is to define the `backendType` value to the name of the storage driver from the newly created backend. Also note the `name-field` value, which must be referenced in a later step.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
```



There is an optional field called `fsType` that is defined in this file. This line can be deleted in NFS backends.

6. Run the `kubectl` command to create the storage class.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl create -f
storage-class-basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. With the storage class created, you must then create the first persistent volume claim (PVC). There is a sample `pvc-basic.yaml` file that can be used to perform this action located in `sample-inputs` as well.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ cp sample-input/pvc-
samples/pvc-basic.yaml ./
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ vi pvc-basic.yaml
```

8. The only edit that must be made to this file is ensuring that the `storageClassName` field matches the one just created. The PVC definition can be further customized as required by the workload to be provisioned.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. Create the PVC by issuing the `kubectl` command. Creation can take some time depending on the size of the backing volume being created, so you can watch the process as it completes.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl create -f
pvc-basic.yaml
persistentvolumeclaim/basic created

[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl get pvc
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
basic      Bound       pvc-b4370d37-0fa4-4c17-bd86-94f96c94b42d  1Gi
RWO          basic-csi     7s
```

[Next: Advanced Configuration Options: Anthos with NetApp.](#)

NetApp ONTAP iSCSI configuration: Anthos with NetApp

To enable Trident integration with the NetApp ONTAP storage system, you must create a backend that enables communication with the storage system.

1. There are sample backend files available in the downloaded installation archive in the `sample-input` folder hierarchy. For NetApp ONTAP systems serving iSCSI, copy the `backend-ontap-san.json` file to your working directory and edit the file.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ cp sample-  
input/backends-samples/ontap-san/backend-ontap-san.json ./  
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ vi backend-ontap-  
san.json
```

2. Edit the `managementLIF`, `dataLIF`, `svm`, `username`, and `password` values in this file.

```
{  
  "version": 1,  
  "storageDriverName": "ontap-san",  
  "managementLIF": "172.21.224.201",  
  "dataLIF": "10.61.181.240",  
  "svm": "trident_svm",  
  "username": "admin",  
  "password": "password"  
}
```

3. With this backend file in place, run the following command to create your first backend.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ ./tridentctl -n  
trident create backend -f backend-ontap-san.json  
+-----+-----+  
+-----+-----+-----+-----+  
|          NAME          | STORAGE DRIVER |          UUID          |  
| STATE | VOLUMES |  
+-----+-----+  
+-----+-----+-----+-----+  
| ontapsan_10.61.181.241 | ontap-san      | 6788533c-7fea-4a35-b797-  
fb9bb3322b91 | online |      0 |  
+-----+-----+  
+-----+-----+-----+-----+
```

4. With the backend created, you must next create a storage class. Just as with the backend, there is a sample storage class file that can be edited for the environment available in the `sample-inputs` folder. Copy it to the working directory and make necessary edits to reflect the backend created.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ cp sample-  
input/storage-class-samples/storage-class-csi.yaml.templ ./storage-  
class-basic.yaml  
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ vi storage-class-  
basic.yaml
```

5. The only edit that must be made to this file is to define the `backendType` value to the name of the storage driver from the newly created backend. Also note the `name-field` value, which must be referenced in a later step.

```
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
  name: basic-csi  
provisioner: csi.trident.netapp.io  
parameters:  
  backendType: "ontap-san"
```



There is an optional field called `fsType` that is defined in this file. In iSCSI backends, this value can be set to a specific Linux filesystem type (XFS, ext4, and so on) or can be deleted to allow OpenShift to decide what filesystem to use.

6. Run the `kubectl` command to create the storage class.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl create -f  
storage-class-basic.yaml  
storageclass.storage.k8s.io/basic-csi created
```

7. With the storage class created, you must then create the first persistent volume claim (PVC). There is a sample `pvc-basic.yaml` file that can be used to perform this action located in `sample-inputs` as well.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ cp sample-input/pvc-  
samples/pvc-basic.yaml ./  
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ vi pvc-basic.yaml
```

8. The only edit that must be made to this file is ensuring that the `storageClassName` field matches the one just created. The PVC definition can be further customized as required by the workload to be provisioned.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. Create the PVC by issuing the `kubectl` command. Creation can take some time depending on the size of the backing volume being created, so you can watch the process as it completes.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl create -f
pvc-basic.yaml
persistentvolumeclaim/basic created

[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl get pvc
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
basic      Bound       pvc-7ceac1ba-0189-43c7-8f98-094719f7956c  1Gi
RWO                                     basic-csi      3s
```

[Next: Advanced Configuration Options: Anthos with NetApp.](#)

NetApp Element iSCSI configuration: Anthos with NetApp

To enable Trident integration with the NetApp Element storage system, you must create a backend that enables communication with the storage system using the iSCSI protocol.

1. There are sample backend files available in the downloaded installation archive in the `sample-input` folder hierarchy. For NetApp Element systems serving iSCSI, copy the `backend-solidfire.json` file to your working directory and edit the file.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ cp sample-
input/backends-samples/solidfire/backend-solidfire.json ./
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ vi ./backend-
solidfire.json
```

- a. Edit the user, password, and MVIP value on the `EndPoint` line.
- b. Edit the `SVIP` value.

```
{
  "version": 1,
  "storageDriverName": "solidfire-san",
  "Endpoint": "https://trident:password@172.21.224.150/json-
rpc/8.0",
  "SVIP": "10.61.180.200:3260",
  "TenantName": "trident",
  "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS":
2000, "burstIOPS": 4000}},
            {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS":
6000, "burstIOPS": 8000}},
            {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS":
8000, "burstIOPS": 10000}}]
}
```

2. With this back-end file in place, run the following command to create your first backend.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ ./tridentctl -n
trident create backend -f backend-solidfire.json
+-----+-----+
+-----+-----+-----+-----+
|          NAME          | STORAGE DRIVER |          UUID          |
| STATE | VOLUMES | |          |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| solidfire_10.61.180.200 | solidfire-san  | b90783ee-e0c9-49af-8d26-
3ea87ce2efdf | online |          0 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

3. With the backend created, you must next create a storage class. Just as with the backend, there is a sample storage class file that can be edited for the environment available in the sample-inputs folder. Copy it to the working directory and make necessary edits to reflect the backend created.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ cp sample-
input/storage-class-samples/storage-class-csi.yaml.templ ./storage-
class-basic.yaml
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ vi storage-class-
basic.yaml
```

4. The only edit that must be made to this file is to define the `backendType` value to the name of the storage driver from the newly created backend. Also note the `name-field` value, which must be referenced in a later step.


```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "solidfire-san"
```



There is an optional field called `fsType` that is defined in this file. In iSCSI backends, this value can be set to a specific Linux filesystem type (XFS, ext4, and so on), or it can be deleted to allow OpenShift to decide what filesystem to use.

5. Run the `kubectl` command to create the storage class.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl create -f
storage-class-basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

6. With the storage class created, you must then create the first persistent volume claim (PVC). There is a sample `pvc-basic.yaml` file that can be used to perform this action located in `sample-inputs` as well.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ cp sample-input/pvc-
samples/pvc-basic.yaml ./
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ vi pvc-basic.yaml
```

7. The only edit that must be made to this file is ensuring that the `storageClassName` field matches the one just created. The PVC definition can be further customized as required by the workload to be provisioned.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

8. Create the PVC by issuing the `kubectl` command. Creation can take some time depending on the size of the backing volume being created, so you can watch the process as it completes.

```
[ubuntu@gke-admin-ws-2021-07-15 trident-installer]$ kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY
ACCESS MODES		STORAGECLASS	AGE
basic	Bound	pvc-3445b5cc-df24-453d-a1e6-b484e874349d	1Gi
RWO		basic-csi	5s

Next: Advanced Configuration Options: Anthos with NetApp.

Copyright Information

Copyright © 2022 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.