



Google Cloud NetApp Volumes

Trident

NetApp
October 30, 2026

This PDF was generated from <https://docs.netapp.com/us-en/trident-2606/trident-use/gcnv.html> on October 30, 2026. Always check docs.netapp.com for the latest.

Table of Contents

Google Cloud NetApp Volumes	1
Configure Google Cloud NetApp Volumes	1
Overview	1
Prepare to configure	1
Configure NAS storage	2
Configure Google Cloud NetApp Volumes for SAN workloads	5
Overview	5
Flex Unified storage pools	5
Configure a Trident SAN backend	5
Create a StorageClass	6
Provision block volumes	6
Block volume behavior	8
Access modes	8
volumeMode behavior	8
Supported operations	9
Extra GiB overprovisioning behavior	9
Pod examples	9
Attach and mount behavior	10
Prepare to configure a Google Cloud NetApp Volumes backend	11
Prerequisites for NFS or SMB volumes	11
Google Cloud NetApp Volumes backend configuration options and examples	11
Backend configuration options	11
Volume provisioning options	13
Example configurations	13
What's next?	19
Storage class definitions	19
Configure auto-tiering for Google Cloud NetApp Volumes	22
Overview	22
Concepts	22
Configuration model	23
Supported functionality in Trident 26.06	23
Unsupported functionality in Trident 26.06	23
Backend configuration parameters	24
Volume-level overrides using PersistentVolumeClaim annotations	24
Behavior and limitations	24
Configure Google Cloud NetApp Volumes ONTAP-mode storage pools	25
How it works	25
Supported and unsupported scope	25
Prerequisites	26
Configure the backend	26

Google Cloud NetApp Volumes

Configure Google Cloud NetApp Volumes

You can configure Google Cloud NetApp Volumes as a backend for Trident to provision storage for Kubernetes workloads.

Overview

Trident supports Google Cloud NetApp Volumes for both NAS (NFS and SMB) and block (iSCSI) workloads.

- NAS workloads use the `google-cloud-netapp-volumes` backend
- Block (iSCSI) workloads use the `google-cloud-netapp-volumes-san` backend

NAS volumes provide file-based storage and are accessed using NFS or SMB protocols. These volumes support shared access across multiple pods or nodes.

Block volumes provide raw block storage and are accessed as iSCSI devices attached to Kubernetes nodes. These volumes are used when applications require block-level access.

This applies to the following environments:

- Trident 26.02 and later
- Google Kubernetes Engine (GKE) or Red Hat OpenShift
- Google Cloud NetApp Volumes storage pools

To configure block (iSCSI) storage, see [Configure block storage \(iSCSI\)](#).

Prepare to configure

Cloud identity enables Kubernetes workloads to access Google Cloud resources by authenticating as a workload identity instead of using static credentials.

To use cloud identity with Google Cloud NetApp Volumes, you must have:

- A Kubernetes cluster deployed using Google Kubernetes Engine (GKE)
- Workload identity enabled on the GKE cluster and the metadata server enabled on the node pools
- A Google Cloud service account with the Google Cloud NetApp Volumes Admin role (`roles/netapp.admin`) or an equivalent custom role
- Trident installed with the cloud provider set to `GCP` and the cloud identity annotation configured

Trident operator

To install Trident using the Trident operator, edit `tridentorchestrator_cr.yaml`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
```

```
name: trident
spec:
  namespace: trident
  cloudProvider: "GCP"
  cloudIdentity: "iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

Helm

Set the cloud provider and cloud identity when installing Trident with Helm:

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

tridentctl

Install Trident by specifying the cloud provider and cloud identity:

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
  -n trident
```

Configure NAS storage



For Google Cloud NetApp Volumes UNIFIED storage pools, Trident applies UNIFIED-specific naming and validation rules during volume operations.

When locating a volume, Trident can evaluate multiple compatible volume name variants (for example, hyphen and underscore formats) to improve import and discovery reliability.

Driver details

Trident provides the `google-cloud-netapp-volumes` driver to provision NAS storage from Google Cloud NetApp Volumes.

The driver supports the following access modes:

- `ReadWriteOnce (RWO)`
- `ReadOnlyMany (ROX)`
- `ReadWriteMany (RWX)`
- `ReadWriteOncePod (RWOP)`

Driver	Protocol	volumeMode	Access modes supported	File systems supported
google-cloud-netapp-volumes	NFS SMB	Filesystem	RWO, ROX, RWX, RWOP	nfs, smb

Configure a Trident NAS backend

```

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
    - labels:
        cloud: gcp
        network: "<vpc-network>"

```

Provision NAS volumes

NAS volumes are provisioned using the `google-cloud-netapp-volumes` backend and support NFS and SMB protocols.

StorageClass for NFS volumes

To provision NFS volumes, set `nasType` to `nfs`.

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: true

```

StorageClass for SMB volumes

To provision SMB volumes, set `nasType` to `smb` and provide credentials.

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
allowVolumeExpansion: true

```

PersistentVolumeClaim example (RWX)

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs

```

PersistentVolumeClaim example (RWO)

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs

```



NAS volumes use volumeMode: Filesystem.

Configure Google Cloud NetApp Volumes for SAN workloads

You can configure Trident to provision block storage volumes using the iSCSI protocol from Google Cloud NetApp Volumes. SAN volumes are provisioned from Flex Unified storage pools by using the `google-cloud-netapp-volumes-san` storage driver.



This driver is dedicated to block workloads and does not support NAS protocols.



The `google-cloud-netapp-volumes-san` backend is required to provision iSCSI block volumes. The `google-cloud-netapp-volumes` backend supports NAS protocols only and cannot be used for SAN workloads.

Overview

Trident supports Google Cloud NetApp Volumes SAN (iSCSI) workloads using the `google-cloud-netapp-volumes-san` driver.

SAN volumes are provisioned from Flex Unified storage pools and presented to Kubernetes nodes as iSCSI block devices.

This applies to the following environments:

- Trident 26.02 and later
- Google Kubernetes Engine (GKE) or Red Hat OpenShift
- Google Cloud NetApp Volumes Flex Unified storage pools
- iSCSI-based workloads

Flex Unified storage pools

Flex Unified storage pools provide block storage using the iSCSI protocol and are required for SAN provisioning:

- Flex Unified REGIONAL pools are supported.
- Flex Unified ZONAL pools are supported starting with Trident 26.02.1.
- Only the **Flex** service level is supported for SAN workloads.

Configure a Trident SAN backend

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
```

```

storageDriverName: google-cloud-netapp-volumes-san
projectNumber: "<project-number>"
location: "<region>"
sdkTimeout: "600"
storage:
- labels:
    cloud: gcp
    performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex

```

Create a StorageClass

After configuring the SAN backend, create a StorageClass that references the google-cloud-netapp-volumes-san driver.

The filesystem type is defined in the StorageClass, not in the backend.

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true

```

Supported filesystem types:

- ext4 (default)
- ext3
- xfs



The SAN driver supports only the Flex service level and does not use NAS-specific backend parameters such as `exportRule`, `unixPermissions`, `nasType`, `snapshotDir`, `nfsMountOptions`, or tiering-related settings.

Provision block volumes

ReadWriteOnce (RWO)

```

apiVersion: v1
kind: PersistentVolumeClaim

```

```

metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san

```

ReadWriteOncePod (RWOP)

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san

```

ReadOnlyMany (ROX)

A common pattern for ROX is to clone an existing ReadWriteOnce volume and mount the clone as read-only.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo

```

ReadWriteMany (RWX) — raw block only

ReadWriteMany is supported only when `volumeMode: Block`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

Block volume behavior

Block volumes are provisioned as iSCSI LUNs and presented to Kubernetes nodes as block devices.

Block volumes:

- Use the iSCSI protocol
- Support filesystem and raw block presentation
- Are attached and managed by Trident
- Support multiple Kubernetes access modes

Access modes

Block volumes provisioned by Trident support the following access modes:

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)
- ReadWriteOncePod (RWOP)
- ReadWriteMany (RWX), supported only when `volumeMode: Block`

volumeMode behavior

The `volumeMode` field controls how a block volume is exposed:

- Filesystem
Trident formats and mounts the volume.
- Block
Trident attaches the device and exposes it as a raw block device.

Supported operations

Block volumes provisioned using the `google-cloud-netapp-volumes-san` driver support:

- Create
- Delete
- Clone
- Snapshot
- Resize
- Import

Extra GiB overprovisioning behavior

Google Cloud NetApp Volumes block volumes include internal metadata overhead. This overhead reduces the kernel-visible device size compared to the provisioned capacity.

Testing shows:

- Approximately 300 KiB overhead on initial creation
- Up to approximately 107 MiB overhead after a resize

Because Google Cloud NetApp Volumes accepts only whole-GiB allocations, Trident ensures that the usable device size always meets or exceeds the PVC request by:

- Rounding the requested size up to the next whole GiB
- Adding an additional 1 GiB buffer

Example:

- PVC request: 100 GiB
- Provisioned size in Google Cloud NetApp Volumes: 101 GiB
- Usable space visible to the application: at least 100 GiB

Pod examples

Filesystem-mounted block volume (RWO)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
```

```

- name: data
  mountPath: /mnt/data
volumes:
- name: data
  persistentVolumeClaim:
    claimName: gcnv-san-rwo

```

Raw block device (RWX)

```

apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx

```

Attach and mount behavior

For SAN volumes provisioned from Google Cloud NetApp Volumes:

- Trident creates a Logical Unit Number (LUN) in a Flex Unified storage pool.
- During publish, Trident maps the LUN to a per-node host group.
- During node staging, Trident:
 - Logs in to the iSCSI target
 - Discovers the LUN
 - Configures multipath
- If `volumeMode: Filesystem`, Trident formats the device if required and mounts it.
- If `volumeMode: Block`, Trident attaches the device and exposes it directly to the pod without formatting or mounting.



SAN block volumes do not provide distributed locking or write coordination. When a block volume is accessed by multiple nodes (ReadWriteMany with `volumeMode: Block`), the application or filesystem must manage concurrency.

Prepare to configure a Google Cloud NetApp Volumes backend

Before you can configure your Google Cloud NetApp Volumes backend, you need to ensure the following requirements are met.

Prerequisites for NFS or SMB volumes

If you are using Google Cloud NetApp Volumes for the first time or in a new location, some initial configuration is required to set up Google Cloud NetApp Volumes and create an NFS or SMB volume. Refer to [Before you begin](#).

Ensure that you have the following before configuring Google Cloud NetApp Volumes backend:

- A Google Cloud account configured with Google Cloud NetApp Volumes service. Refer to [Google Cloud NetApp Volumes](#).
- Project number of your Google Cloud account. Refer to [Identifying projects](#).
- A Google Cloud service account with the NetApp Volumes Admin (`roles/netapp.admin`) role. Refer to [Identity and Access Management roles and permissions](#).
- API key file for your GCNV account. Refer to [Create a service account key](#)
- A storage pool. Refer to [Storage pools overview](#) .

For more information about how to set up access to Google Cloud NetApp Volumes, refer to [Set up access to Google Cloud NetApp Volumes](#).

Google Cloud NetApp Volumes backend configuration options and examples

Learn about backend configuration options for Google Cloud NetApp Volumes and review configuration examples.

Backend configuration options

Each backend provisions volumes in a single Google Cloud region. To create volumes in other regions, you can define additional backends.

Parameter	Description	Default
version		Always 1
storageDriverName	Name of the storage driver	The value of <code>storageDriverName</code> must be specified as "google-cloud-netapp-volumes".
backendName	(Optional) Custom name of the storage backend	Driver name + "_" + part of API key

Parameter	Description	Default
<code>storagePools</code>	Optional parameter used to specify storage pools for volume creation.	
<code>projectNumber</code>	Google Cloud account project number. The value is found on the Google Cloud portal home page.	
<code>location</code>	<p>The Google Cloud location where Trident creates GCNV volumes. When creating cross-region Kubernetes clusters, volumes created in a <code>location</code> can be used in workloads scheduled on nodes across multiple Google Cloud regions.</p> <p>Cross-region traffic incurs an additional cost.</p>	
<code>apiKey</code>	<p>API key for the Google Cloud service account with the <code>netapp.admin</code> role.</p> <p>It includes the JSON-formatted contents of a Google Cloud service account's private key file (copied verbatim into the backend configuration file).</p> <p>The <code>apiKey</code> must include key-value pairs for the following keys: <code>type</code>, <code>project_id</code>, <code>client_email</code>, <code>client_id</code>, <code>auth_uri</code>, <code>token_uri</code>, <code>auth_provider_x509_cert_url</code>, and <code>client_x509_cert_url</code>.</p>	
<code>nfsMountOptions</code>	Fine-grained control of NFS mount options.	"nfsvers=3"
<code>limitVolumeSize</code>	Fail provisioning if the requested volume size is above this value.	"" (not enforced by default)
<code>serviceLevel</code>	The service level of a storage pool and its volumes. The values are <code>flex</code> , <code>standard</code> , <code>premium</code> , or <code>extreme</code> .	
<code>labels</code>	Set of arbitrary JSON-formatted labels to apply on volumes	""
<code>network</code>	Google Cloud network used for GCNV volumes.	
<code>debugTraceFlags</code>	<p>Debug flags to use when troubleshooting. Example, <code>{"api":false, "method":true}</code>.</p> <p>Do not use this unless you are troubleshooting and require a detailed log dump.</p>	null
<code>nasType</code>	<p>Configure NFS or SMB volumes creation.</p> <p>Options are <code>nfs</code>, <code>smb</code> or <code>null</code>. Setting to <code>null</code> defaults to NFS volumes.</p>	<code>nfs</code>

Parameter	Description	Default
supportedTopologies	Represents a list of regions and zones that are supported by this backend. For more information, refer to Use CSI Topology . For example: supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a	

Volume provisioning options

You can control default volume provisioning in the `defaults` section of the configuration file.

Parameter	Description	Default
exportRule	The export rules for new volumes. Must be a comma-separated list of any combination of IPv4 addresses.	"0.0.0.0/0"
snapshotDir	Access to the <code>.snapshot</code> directory	true, false (default behavior might vary. Set explicitly) "false" for NFSv3
snapshotReserve	Percentage of volume reserved for snapshots	"" (accept default of 0)
unixPermissions	The unix permissions of new volumes (4 octal digits).	""

Example configurations

The following examples show basic configurations that leave most parameters to default. This is the easiest way to define a backend.

▼ Minimal configuration

This is the absolute minimum backend configuration. With this configuration, Trident discovers all of your storage pools delegated to Google Cloud NetApp Volumes in the configured location, and places new volumes on one of those pools randomly. Because `nasType` is omitted, the `nfs` default applies and the backend will provision for NFS volumes.

This configuration is ideal when you are just getting started with Google Cloud NetApp Volumes and trying things out, but in practice you might need to provide additional scoping for the volumes you provision.



Replace `<id_value>` and `<key_value>` with your service account credentials.

```
---
apiVersion: v1
kind: Secret
```

```

metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret

```

▼ Configuration for SMB volumes

```

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"

```

```

location: asia-east1
serviceLevel: flex
nasType: smb
apiKey:
  type: service_account
  project_id: cloud-native-data
  client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
  client_id: "123456789737813416734"
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret

```

▼ Configuration with StoragePools filter

```

---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium

```

```

storagePools:
  - premium-pool1-europe-west6
  - premium-pool2-europe-west6
apiKey:
  type: service_account
  project_id: my-gcnv-project
  client_email: myproject-prod@my-gcnv-project.iam.gserviceaccount.com
  client_id: "103346282737811234567"
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret

```

▼ Virtual pool configuration

This backend configuration defines multiple virtual pools in a single file. Virtual pools are defined in the `storage` section. They are useful when you have multiple storage pools supporting different service levels and you want to create storage classes in Kubernetes that represent those. Virtual pool labels are used to differentiate the pools. For instance, in the example below `performance` label and `serviceLevel` type is used to differentiate virtual pools.

You can also set some default values to be applicable to all virtual pools, and overwrite the default values for individual virtual pools. In the following example, `snapshotReserve` and `exportRule` serve as defaults for all virtual pools.

For more information, refer to [Virtual pools](#).

```

---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig

```

```

metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
        serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
        serviceLevel: premium
    - labels:
        performance: standard
        serviceLevel: standard

```

▼ Cloud identity for GKE

When you install Trident with a GKE cloud identity, backends authenticate as a workload identity and omit the `apiKey` block. Cloud identity is configured the same way for both Google Cloud NetApp Volumes drivers. The only difference is the `storageDriverName` value: use `google-cloud-netapp-volumes` for NAS (NFS or SMB) volumes and `google-cloud-netapp-volumes-san` for SAN (iSCSI block) volumes.



Cloud identity support for the `google-cloud-netapp-volumes-san` driver requires

Trident 26.06 or later. SAN volumes are provisioned from Flex Unified storage pools, so set `serviceLevel` to `flex`.

NAS driver (google-cloud-netapp-volumes)

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

SAN driver (google-cloud-netapp-volumes-san)

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv-san
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: flex
  storagePool: pool-flex1
```

▼ Supported topologies configuration

Trident facilitates provisioning of volumes for workloads based on regions and availability zones. The `supportedTopologies` block in this backend configuration is used to provide a list of regions and zones per backend. The region and zone values specified here must match the region and zone values from the labels on each Kubernetes cluster node. These regions and zones represent the list of permissible values that can be provided in a storage class. For storage classes that contain a subset of the regions and zones provided in a backend, Trident creates volumes in the mentioned region and zone.

For more information, refer to [Use CSI Topology](#).

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

```
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

What's next?

After you create the backend configuration file, run the following command:

```
kubectl create -f <backend-file>
```

To verify that the backend is successfully created, run the following command:

```
kubectl get tridentbackendconfig
```

NAME	BACKEND NAME	BACKEND UUID
PHASE STATUS		
backend-tbc-gcnv	backend-tbc-gcnv	b2fd1ff9-b234-477e-88fd-713913294f65
Bound Success		

If the backend creation fails, something is wrong with the backend configuration. You can describe the backend using the `kubectl get tridentbackendconfig <backend-name>` command or view the logs to determine the cause by running the following command:

```
tridentctl logs
```

After you identify and correct the problem with the configuration file, you can delete the backend and run the create command again.

Storage class definitions

The following is a basic `StorageClass` definition that refers to the backend above.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
```

```
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

Example definitions using the `parameter.selector` field:

Using `parameter.selector` you can specify for each `StorageClass` the [virtual pool](#) that is used to host a volume. The volume will have the aspects defined in the chosen pool.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

For more details on storage classes, refer to [Create a storage class](#).

Example definitions for SMB volumes

Using `nasType`, `node-stage-secret-name`, and `node-stage-secret-namespace`, you can specify an SMB volume and provide the required Active Directory credentials. Any Active Directory user/password with any/no permissions can be used for the node stage secret.

▼ Basic configuration on default namespace

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

▼ Using different secrets per namespace

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "${pvc.namespace}"
```

▼ Using different secrets per volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "${pvc.name}"
  csi.storage.k8s.io/node-stage-secret-namespace: "${pvc.namespace}"
```



nasType: smb filters for pools which support SMB volumes. nasType: nfs or nasType: null filters for NFS pools.

PVC definition example

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc

```

To verify if the PVC is bound, run the following command:

```

kubectl get pvc gcnv-nfs-pvc

```

NAME	STATUS	VOLUME	CAPACITY
ACCESS MODES	STORAGECLASS	AGE	
gcnv-nfs-pvc	Bound	pvc-b00f2414-e229-40e6-9b16-ee03eb79a213	100Gi
RWX	gcnv-nfs-sc	1m	

Configure auto-tiering for Google Cloud NetApp Volumes

Auto-tiering is configured through Trident backend parameters and PersistentVolumeClaim annotations during volume provisioning. You can configure auto-tiering for Google Cloud NetApp Volumes using Trident.

Overview

Auto-tiering allows Trident to provision volumes that automatically move inactive data from a performance tier to a capacity tier.

This reduces storage cost while preserving performance for frequently accessed data.

Trident applies auto-tiering settings only at volume creation time. Post-provisioning changes are not supported in Trident 26.06.

Concepts

Auto-tiering

Auto-tiering moves infrequently accessed data from a performance tier to a capacity tier based on access patterns.

Data movement occurs asynchronously and is not immediate.

Tiering policy

The tiering policy determines whether auto-tiering is enabled for a volume.

The following policies are supported:

- * `auto`: Enables automatic tiering based on access patterns
- * `none`: Disables auto-tiering

Cooling days

Cooling days specify the minimum number of days a block of data must remain inactive before it becomes eligible for tiering.

Cooling days apply only when the tiering policy is set to `auto`.

Configuration model

Configuration scopes

Auto-tiering can be configured at multiple scopes:

- **Storage pool scope**
Applies to all volumes provisioned from the pool.
- **Volume scope**
Applies to a single volume through `PersistentVolumeClaim` annotations.

Trident determines the effective configuration based on where each setting is defined.

Configuration precedence

When the same setting is defined at multiple scopes, Trident applies the following precedence order:

1. `PersistentVolumeClaim` annotations
2. Trident backend configuration
3. Storage pool defaults

Settings defined at a higher precedence override lower-level values.

Supported functionality in Trident 26.06

Trident 26.06 supports the following auto-tiering capabilities for Google Cloud NetApp Volumes:

- Enabling or disabling auto-tiering during volume provisioning
- Defining a tiering policy in the Trident backend configuration
- Overriding the tiering policy and cooling days per volume using PVC annotations
- Configuring cooling days for volumes with auto-tiering enabled

Unsupported functionality in Trident 26.06

The following operations are not supported:

- Modifying auto-tiering settings after volume creation

- Changing tiering policies on existing volumes using Kubernetes updates
- Applying auto-tiering settings outside of Trident-managed provisioning workflows

Backend configuration parameters

The following parameters control auto-tiering behavior when defined in the Trident backend configuration:

Parameter	Required	Description
tieringPolicy	No	Tiering policy for volumes (<code>auto</code> or <code>none</code>)
tieringMinimumCoolingDays	No	Number of inactive days before data is tiered (range: 2–183, default: 31)

Volume-level overrides using PersistentVolumeClaim annotations

Supported annotations

PersistentVolumeClaim annotations allow per-volume overrides of auto-tiering settings.

Annotation	Description
<code>trident.netapp.io/tieringPolicy</code>	Overrides the tiering policy for the volume
<code>trident.netapp.io/tieringMinimumCoolingDays</code>	Overrides the cooling days value for the volume

Example: PersistentVolumeClaim with auto-tiering overrides

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

Behavior and limitations

Provisioning behavior

- Auto-tiering settings are evaluated and applied only at volume creation time.

- Trident does not reconcile tiering configuration after provisioning.
- Cooling days are ignored when the tiering policy is set to `none`.

Platform limitations

- Auto-tiering is supported only for NAS volumes (NFS and SMB).
- Block volumes (iSCSI) do not support auto-tiering.
- The Google Cloud NetApp Volumes storage pool must have auto-tiering enabled in Google Cloud.

Supported values

- Valid range for `tieringMinimumCoolingDays`: 2 to 183
- Default value: 31

Configure Google Cloud NetApp Volumes ONTAP-mode storage pools

Beginning with Trident 26.06, you can configure existing `ontap-san` and `ontap-nas` backends to provision storage on Google Cloud NetApp Volumes (GCNV) ONTAP-mode storage pools. This feature is a Tech Preview.



This feature is a Tech Preview in Trident 26.06. Tech Preview features are not supported for production use. The functionality, configuration fields, and supported matrix can change before general availability (GA).

How it works

Trident reuses the existing `ontap-san` and `ontap-nas` driver logic and routes ONTAP REST calls through a GCNV proxy endpoint. This approach lets you provision and manage storage on GCNV-hosted ONTAP clusters while keeping the same operational workflows that you use for a direct ONTAP backend. ONTAP-mode does not introduce a new storage driver. You enable it per backend by adding a `gcnv` configuration block to an `ontap-san` or `ontap-nas` backend.

The provisioning path is as follows:

`PersistentVolumeClaim` → Trident (`ontap-san` or `ontap-nas`) → ONTAP REST client → GCNV proxy → ONTAP cluster in the GCNV pool

Supported and unsupported scope

This Tech Preview supports the following:

- Drivers: `ontap-san` (iSCSI) and `ontap-nas` (NFS or SMB).
- Lifecycle operations through the proxy: backend initialization, volume creation and deletion, and related ONTAP REST flows.
- Authentication models: Workload Identity Pool (WIP), service account key, and Application Default Credentials (ADC) fallback.

The following are out of scope for this Tech Preview:

- The `ontap-san-economy`, `ontap-nas-economy`, and `ontap-nas-flexgroup` drivers.
- The ASA r2 personality path.
- ZAPI fallback. ONTAP-mode uses ONTAP REST only.

Prerequisites

Before you configure an ONTAP-mode backend, ensure that you have the following:

- Trident 26.06 or later.
- A GCNV ONTAP-mode storage pool in your target project and location.
- GCP IAM permissions for proxy operations, scoped appropriately for your environment.
- A backend configuration that includes `proxyURL`, `projectNumber`, `location`, and `poolID`.
- For iSCSI SAN workflows, the node-side iSCSI and multipath prerequisites for your platform. Refer to [Prepare the worker node](#).

Configure the backend

A `TridentBackendConfig` requires credentials. For ONTAP-mode, that secret holds GCP proxy credentials—not ONTAP managementLIF / username / password. `svm` is optional: each GCNV ONTAP-mode pool currently has one SVM, and Trident derives it from the pool when omitted.

Use the same service account pattern as native GCNV: non-sensitive fields in `gcnv.apiKey`, `private_key` and `private_key_id` in the secret. See [Google Cloud NetApp Volumes configuration examples](#).

ontap-san backend

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-san
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  backendName: gcnv-ontap-san
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<region-or-zone>"
    poolID: "<pool-id>"
    svm: "<svm-name>"
```

ontap-nas backend

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<region-or-zone>"
    poolID: "<pool-id>"
    apiKey:
      type: service_account
      project_id: "<project-id>"
      client_email: "<service-account-email>"
      client_id: "<client-id>"
      auth_uri: "https://accounts.google.com/o/oauth2/auth"
      token_uri: "https://oauth2.googleapis.com/token"
      auth_provider_x509_cert_url:
"https://www.googleapis.com/oauth2/v1/certs"
      client_x509_cert_url: "<client-x509-cert-url>"
```

Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-sa-secret
  namespace: trident
type: Opaque
stringData:
  private_key_id: "<private-key-id>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <private-key>
    -----END PRIVATE KEY-----
```



Do not put `private_key` or `private_key_id` in the `TridentBackendConfig spec`—the CRD rejects them. Use `ontap-san` with the same `gcnv` and credential pattern for block storage.

```
svm: "<svm-name>"
```

== Authentication

Trident resolves credentials for proxy access in the following order:

- . Workload Identity Pool (WIP)
- . Service account key
- . Application Default Credentials (ADC)

For this Tech Preview, use the following practices:

- * Prefer Workload Identity Pool where it is available.
- * Do not embed raw private keys in version-controlled backend files. Store credentials in a Kubernetes secret.
- * Apply least-privilege IAM scoping to the service account.

== Validation and initialization behavior

Trident validates an ONTAP-mode backend during initialization and fails fast rather than partially initializing.

Backend initialization fails if a required ``gcnv`` field is missing, the driver is unsupported for ONTAP-mode, or credential resolution fails.

== Known limitations

- * This feature is a Tech Preview. Behavior and the supported matrix can change before GA.
- * ZAPI fallback is not used in ONTAP-mode.
- * Protocol and node-readiness requirements still apply. For example, iSCSI workflows require the node-side prerequisites described in [xref:{relative_path}../trident-use/worker-node-prep.html](#) [Prepare the worker node].
- * Existing ONTAP and GCNV environment constraints still apply.

== Upgrade and compatibility

ONTAP-mode is opt-in per backend through the ``gcnv`` configuration block:

- * Existing non-GCNV ONTAP backends are unaffected.

* Mixed deployments that combine direct ONTAP backends and ONTAP-mode proxy backends are supported through backend-level configuration. Each backend is configured independently.

* To stop using ONTAP-mode, remove or replace the affected backends. No global switch is required.

== What's next?

Apply secret before TBC; use `kubectl apply` and `-n trident` to match examples.

:leveloffset: -1

:leveloffset: -1

<<<

Copyright information

Copyright © 2026 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.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b) (3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

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