Get started

Trident

NetApp
June 30, 2026

Table of Contents

Get started

Learn about Trident

Learn about Trident

Trident is a fully-supported open source project maintained by NetApp. It has been designed to help you meet your containerized application's persistence demands using industry-standard interfaces, such as the Container Storage Interface (CSI).

What is Trident?

Netapp Trident enables consumption and management of storage resources across all popular NetApp storage platforms, in the public cloud or on premises, including on-premises ONTAP clusters (AFF, FAS, and ASA), ONTAP Select, Cloud Volumes ONTAP, Element software (NetApp HCI, SolidFire), Azure NetApp Files, Amazon FSx for NetApp ONTAP, and Google Cloud NetApp Volumes.

Trident is a Container Storage Interface (CSI) compliant dynamic storage orchestrator that natively integrates with Kubernetes. Trident runs as a single Controller Pod plus a Node Pod on each worker node in the cluster. Refer to Trident architecture for details.

Trident also provides direct integration with the Docker ecosystem for NetApp storage platforms. The NetApp Docker Volume Plugin (nDVP) supports the provisioning and management of storage resources from the storage platform to Docker hosts. Refer to Deploy Trident for Docker for details.

If this is your first time using Kubernetes, you should familiarize yourself with the Kubernetes concepts and tools.

Supported Kubernetes platforms

Trident is supported on a range of Kubernetes distributions and platforms.

Supported platforms include:
* Upstream Kubernetes
* Red Hat OpenShift
* SUSE Harvester 1.7.0 (ONTAP iSCSI)

Kubernetes integration with NetApp products

The NetApp portfolio of storage products integrates with many aspects of a Kubernetes cluster, providing advanced data management capabilities, which enhance the functionality, capability, performance, and availability of the Kubernetes deployment.

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP is a fully managed AWS service that lets you launch and run file systems powered by the NetApp ONTAP storage operating system.

Azure NetApp Files

Azure NetApp Files is an enterprise-grade Azure file share service, powered by NetApp. You can run your most demanding file-based workloads in Azure natively, with the performance and rich data management you expect from NetApp.

Cloud Volumes ONTAP

Cloud Volumes ONTAP is a software-only storage appliance that runs the ONTAP data management software in the cloud.

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes is a fully managed file storage service in Google Cloud that provides high-performance, enterprise-grade file storage.

Element software

Element enables the storage administrator to consolidate workloads by guaranteeing performance and enabling a simplified and streamlined storage footprint.

NetApp HCI

NetApp HCI simplifies the management and scale of the datacenter by automating routine tasks and enabling infrastructure administrators to focus on more important functions.

Trident can provision and manage storage devices for containerized applications directly against the underlying NetApp HCI storage platform.

NetApp ONTAP

NetApp ONTAP is the NetApp multiprotocol, unified storage operating system that provides advanced data management capabilities for any application.

ONTAP systems have all-flash, hybrid, or all-HDD configurations and offer many different deployment models: on-premises FAS, AFA, and ASA clusters, ONTAP Select, and Cloud Volumes ONTAP. Trident supports these ONTAP deployment models.

Trident architecture

Trident runs as a single Controller Pod plus a Node Pod on each worker node in the cluster. The node pod must be running on any host where you want to potentially mount a Trident volume.

Understanding controller pods and node pods

Trident deploys as a single Trident Controller Pod and one or more Trident Node Pods on the Kubernetes

cluster and uses standard Kubernetes *CSI Sidecar Containers* to simplify the deployment of CSI plugins. Kubernetes CSI Sidecar Containers are maintained by the Kubernetes Storage community.

Kubernetes node selectors and tolerations and taints are used to constrain a pod to run on a specific or preferred node. You can configure node selectors and tolerations for controller and node pods during Trident installation.

- The controller plugin handles volume provisioning and management, such as snapshots and resizing.
- The node plugin handles attaching the storage to the node.

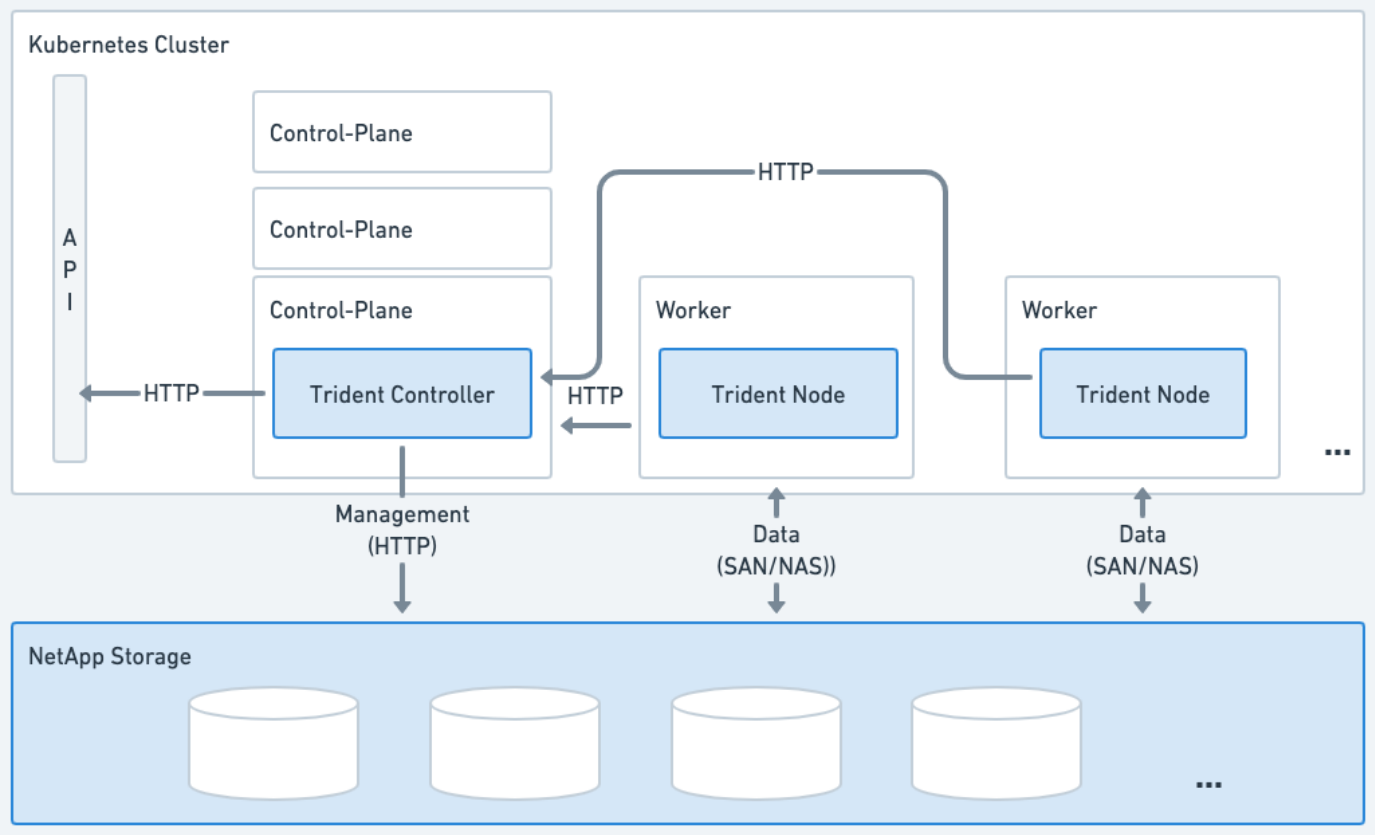


Figure 1. Trident deployed on the Kubernetes cluster

Trident Controller Pod

The Trident Controller Pod is a single Pod running the CSI Controller plugin.

- Responsible for provisioning and managing volumes in NetApp storage
- Managed by a Kubernetes Deployment
- Can run on the control-plane or worker nodes, depending on installation parameters.

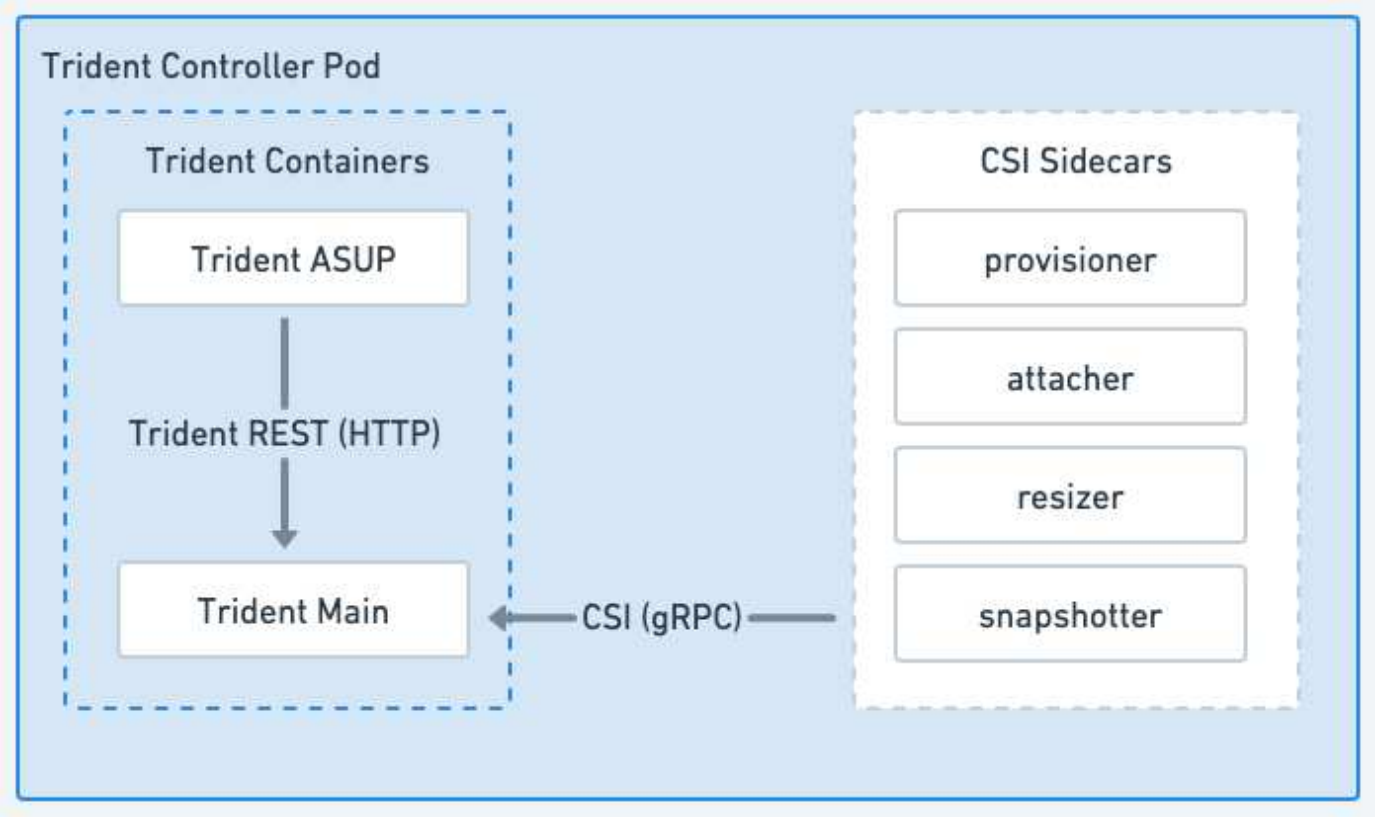


Figure 2. Trident Controller Pod diagram

Trident Node Pods

Trident Node Pods are privileged Pods running the CSI Node plugin.

- Responsible for mounting and unmounting storage for Pods running on the host
- Managed by a Kubernetes DaemonSet
- Must run on any node that will mount NetApp storage

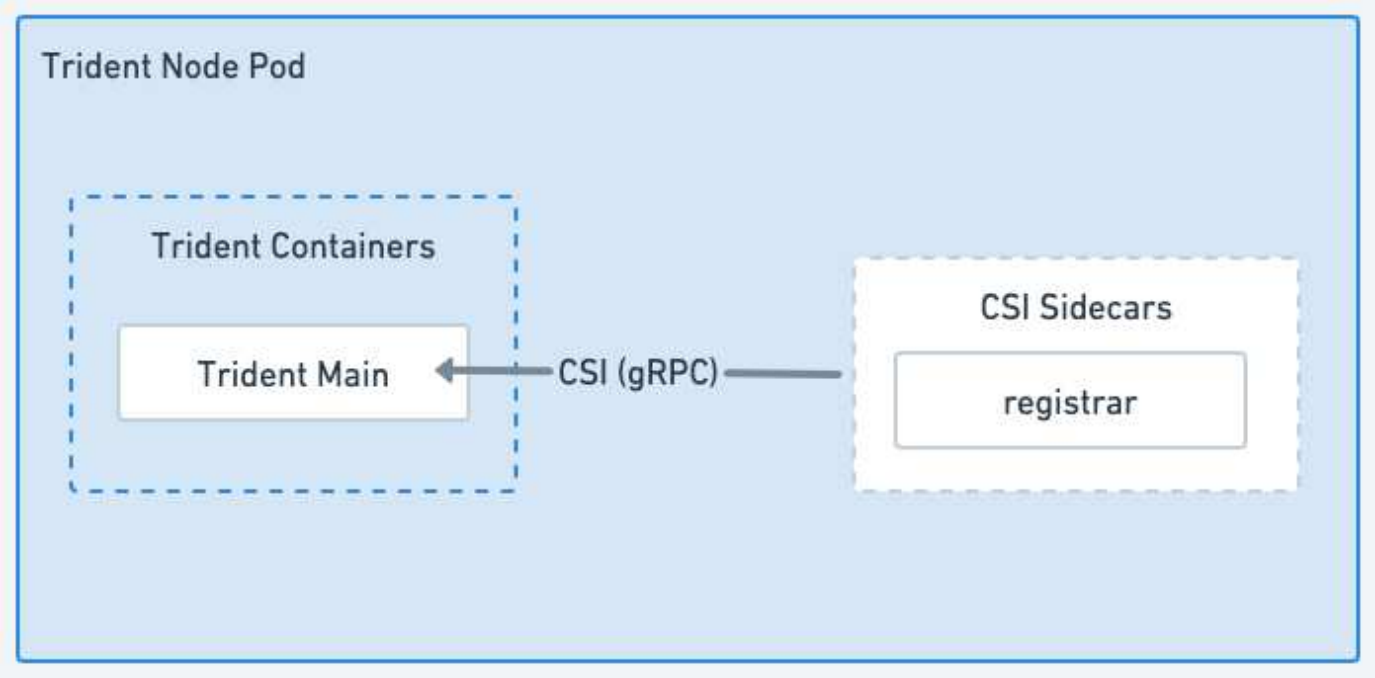


Figure 3. Trident Node Pod diagram

Supported Kubernetes cluster architectures

Trident is supported with the following Kubernetes architectures:

Kubernetes cluster architectures	Supported	Default install
Single master, compute	Yes	Yes
Multiple master, compute	Yes	Yes
Master, `etcd`, compute	Yes	Yes
Master, infrastructure, compute	Yes	Yes

Concepts

Provisioning

Provisioning in Trident has two primary phases. The first phase associates a storage class with the set of suitable backend storage pools and occurs as a necessary preparation before provisioning. The second phase includes the volume creation itself and requires choosing a storage pool from those associated with the pending volume's storage class.

Storage class association

Associating backend storage pools with a storage class relies on both the storage class's requested attributes and its `storagePools`, `additionalStoragePools`, and `excludeStoragePools` lists. When you create a storage class, Trident compares the attributes and pools offered by each of its backends to those requested by

the storage class. If a storage pool's attributes and name match all of the requested attributes and pool names, Trident adds that storage pool to the set of suitable storage pools for that storage class. In addition, Trident adds all storage pools listed in the `additionalStoragePools` list to that set, even if their attributes do not fulfill all or any of the storage class's requested attributes. You should use the `excludeStoragePools` list to override and remove storage pools from use for a storage class. Trident performs a similar process every time you add a new backend, checking whether its storage pools satisfy those of the existing storage classes and removing any that have been marked as excluded.

Volume creation

Trident then uses the associations between storage classes and storage pools to determine where to provision volumes. When you create a volume, Trident first gets the set of storage pools for that volume's storage class, and, if you specify a protocol for the volume, Trident removes those storage pools that cannot provide the requested protocol (for example, a NetApp HCI/SolidFire backend cannot provide a file-based volume while an ONTAP NAS backend cannot provide a block-based volume). Trident randomizes the order of this resulting set, to facilitate an even distribution of volumes, and then iterates through it, attempting to provision the volume on each storage pool in turn. If it succeeds on one, it returns successfully, logging any failures encountered in the process. Trident returns a failure **only if** it fails to provision on **all** the storage pools available for the requested storage class and protocol.

Volume snapshots

Learn more about how Trident handles the creation of volume snapshots for its drivers.

Learn about volume snapshot creation

- For the `ontap-nas`, `ontap-san`, and `azure-netapp-files` drivers, each Persistent Volume (PV) maps to a FlexVol volume As a result, volume snapshots are created as NetApp snapshots. NetApp snapshot technology delivers more stability, scalability, recoverability, and performance than competing snapshot technologies. These snapshot copies are extremely efficient both in the time needed to create them and in storage space.
- For the `ontap-nas-flexgroup` driver, each Persistent Volume (PV) maps to a FlexGroup. As a result, volume snapshots are created as NetApp FlexGroup snapshots. NetApp snapshot technology delivers more stability, scalability, recoverability, and performance than competing snapshot technologies. These snapshot copies are extremely efficient both in the time needed to create them and in storage space.
- For the `ontap-san-economy` driver, PVs map to LUNs created on shared FlexVol volumes VolumeSnapshots of PVs are achieved by performing FlexClones of the associated LUN. ONTAP FlexClone technology makes it possible to create copies of even the largest datasets almost instantaneously. Copies share data blocks with their parents, consuming no storage except what is required for metadata.
- For the `solidfire-san` driver, each PV maps to a LUN created on the NetApp Element software/NetApp HCI cluster. VolumeSnapshots are represented by Element snapshots of the underlying LUN. These snapshots are point-in-time copies and only take up a small amount of system resources and space.
- When working with the `ontap-nas` and `ontap-san` drivers, ONTAP snapshots are point-in-time copies of the FlexVol and consume space on the FlexVol itself. This can result in the amount of writable space in the volume to reduce with time as snapshots are created/scheduled. One simple way of addressing this is to grow the volume by resizing through Kubernetes. Another option is to delete snapshots that are no longer required. When a VolumeSnapshot created through Kubernetes is deleted, Trident will delete the associated ONTAP snapshot. ONTAP snapshots that were not created through Kubernetes can also be deleted.

With Trident, you can use VolumeSnapshots to create new PVs from them. Creating PVs from these snapshots

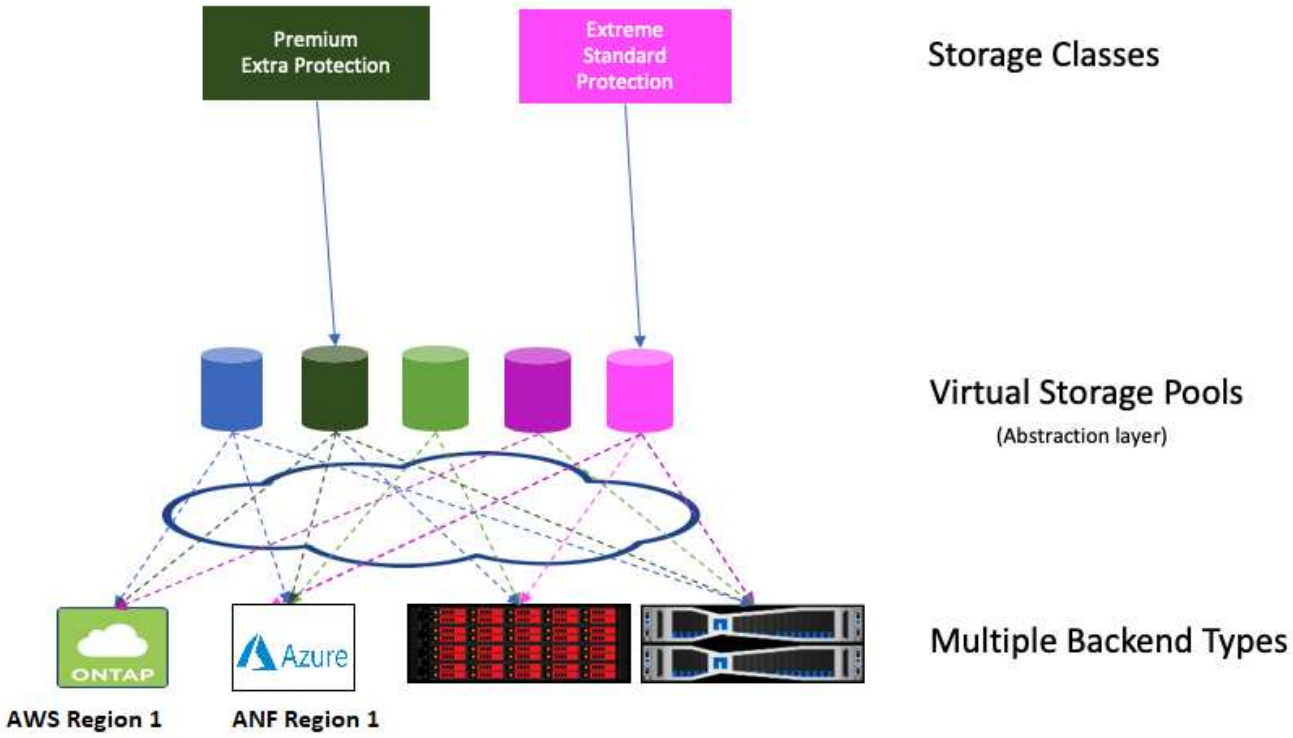
is performed by using the FlexClone technology for supported ONTAP backends. When creating a PV from a snapshot, the backing volume is a FlexClone of the snapshot's parent volume. The `solidfire-san` driver uses Element software volume clones to create PVs from snapshots. Here it creates a clone from the Element snapshot.

Virtual pools

Virtual pools provide a layer of abstraction between Trident storage backends and Kubernetes `StorageClasses`. They allow an administrator to define aspects, such as location, performance, and protection for each backend in a common, backend-agnostic way without making a `StorageClass` specify which physical backend, backend pool, or backend type to use to meet desired criteria.

Learn about virtual pools

The storage administrator can define virtual pools on any of the Trident backends in a JSON or YAML definition file.



Any aspect specified outside the virtual pools list is global to the backend and will apply to all the virtual pools, while each virtual pool might specify one or more aspects individually (overriding any backend-global aspects).

- When defining virtual pools, do not attempt to rearrange the order of existing virtual pools in a backend definition.
- We advise against modifying attributes for an existing virtual pool. You should define a new virtual pool to make changes.

Most aspects are specified in backend-specific terms. Crucially, the aspect values are not exposed outside the

backend's driver and are not available for matching in `StorageClasses`. Instead, the administrator defines one or more labels for each virtual pool. Each label is a key:value pair, and labels might be common across unique backends. Like aspects, labels can be specified per-pool or global to the backend. Unlike aspects, which have predefined names and values, the administrator has full discretion to define label keys and values as needed. For convenience, storage administrators can define labels per virtual pool and group volumes by label.

The virtual pool labels can be defied using these characters:

- uppercase letters `A-Z`
- lowercase letters `a-z`
- numbers `0-9`
- underscores _
- hyphens -

A `StorageClass` identifies which virtual pool to use by referencing the labels within a selector parameter. Virtual pool selectors support the following operators:

Operator	Example	A pool's label value must:
`=`	performance=premium	Match
`!=`	performance!=extreme	Not match
`in`	location in (east, west)	Be in the set of values
`notin`	performance notin (silver, bronze)	Not be in the set of values
`<key>`	protection	Exist with any value
`!<key>`	!protection	Not exist

Volume access groups

Learn more about how Trident uses volume access groups.

> Ignore this section if you are using CHAP, which is recommended to simplify management and avoid the scaling limit described below. In addition, if you are using Trident in CSI mode, you can ignore this section. Trident uses CHAP when installed as an enhanced CSI provisioner.

Learn about volume access groups

Trident can use volume access groups to control access to the volumes that it provisions. If CHAP is disabled, it expects to find an access group called `trident` unless you specify one or more access group IDs in the configuration.

While Trident associates new volumes with the configured access groups, it does not create or otherwise manage access groups themselves. The access groups must exist before the storage backend is added to Trident, and they need to contain the iSCSI IQNs from every node in the Kubernetes cluster that could potentially mount the volumes provisioned by that backend. In most installations, that includes every worker node in the cluster.

For Kubernetes clusters with more than 64 nodes, you should use multiple access groups. Each access group

may contain up to 64 IQNs, and each volume can belong to four access groups. With the maximum four access groups configured, any node in a cluster up to 256 nodes in size will be able to access any volume. For latest limits on volume access groups, refer to here.

If you're modifying the configuration from one that is using the default `trident` access group to one that uses others as well, include the ID for the `trident` access group in the list.

Quick start for Trident

You can install Trident and start managing storage resources in a few steps. Before getting started, review Trident requirements.

For Docker, refer to Trident for Docker.

1 Prepare the worker node

All worker nodes in the Kubernetes cluster must be able to mount the volumes you have provisioned for your pods.

Prepare the worker node

2 Install Trident

Trident offers several installation methods and modes optimized for a variety of environments and organizations.

Install Trident

3 Create a backend

A backend defines the relationship between Trident and a storage system. It tells Trident how to communicate with that storage system and how Trident should provision volumes from it.

Configure a backend for your storage system

4 Create a Kubernetes StorageClass

The Kubernetes StorageClass object specifies Trident as the provisioner and allows you to create a storage class to provision volumes with customizable attributes. Trident creates a matching storage class for Kubernetes objects that specify the Trident provisioner.

Create a storage class

5 Provision a volume

A *PersistentVolume* (PV) is a physical storage resource provisioned by the cluster administrator on a Kubernetes cluster. The *PersistentVolumeClaim* (PVC) is a request for access to the PersistentVolume on the cluster.

Create a PersistentVolume (PV) and a PersistentVolumeClaim (PVC) that uses the configured Kubernetes StorageClass to request access to the PV. You can then mount the PV to a pod.

Provision a volume

What's next?

You can now add additional backends, manage storage classes, manage backends, and perform volume operations.

Requirements

Before installing Trident you should review these general system requirements. Specific backends might have additional requirements.

Critical information about Trident

You must read the following critical information about Trident.

Critical information about Trident

- Kubernetes 1.36 is now supported in Trident. Upgrade Trident prior to upgrading Kubernetes.
- Trident strictly enforces the use of multipathing configuration in SAN environments, with a recommended value of `find_multipaths: no` in multipath.conf file.

  Use of non-multipathing configuration or use of `find_multipaths: yes` or `find_multipaths: smart` value in multipath.conf file will result in mount failures. Trident has recommended the use of `find_multipaths: no` since the 21.07 release.

Supported frontends (orchestrators)

Trident supports multiple container engines and orchestrators, including the following:

- Anthos On-Prem (VMware) and Anthos on bare metal 1.16
- Kubernetes 1.27 - 1.36
- OpenShift 4.12, 4.14 - 4.21 (If you plan to use iSCSI node preparation with OpenShift 4.19, the minimum Trident version supported is 25.06.1.)

  Trident continues to support older OpenShift versions in alignment with the Red Hat Extended Update Support (EUS) release lifecycle, even if they rely on Kubernetes versions that are no longer officially supported upstream. While installing Trident in such cases, you can safely ignore any warning messages about the Kubernetes version.

- Rancher Kubernetes Engine 2 (RKE2) v1.28.x - 1.36.x

Trident also works with a host of other fully-managed and self-managed Kubernetes offerings, including Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service

(AKS), Mirantis Kubernetes Engine (MKE), and VMWare Tanzu Portfolio.

Trident and ONTAP can be used as a storage provider for KubeVirt.

> Before upgrading a Kubernetes cluster from 1.25 to 1.26 or later that has Trident installed, refer to Upgrade a Helm installation.

Supported backends (storage)

To use Trident, you need one or more of the following supported backends:

- Amazon FSx for NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp Volumes
- NetApp All SAN Array (ASA)
- On-premises FAS, AFF, or ASA r2 (iSCSI, NVMe/TCP, and FC) running ONTAP versions under NetApp full or limited support. See Software Version Support.
- NetApp HCI/Element software 11 or above

Trident support for KubeVirt and OpenShift Virtualization

Storage drivers supported:

Trident supports the following ONTAP drivers for KubeVirt and OpenShift Virtualization:

- ontap-nas
- ontap-san (iSCSI, FCP, NVMe over TCP)
- ontap-san-economy (iSCSI Only)

Points to consider:

- Update storage class to have the `fsType` parameter (for example: `fsType: "ext4"`) in OpenShift Virtualization environment. If needed, set the volume mode to block explicitly using the `volumeMode=Block` parameter in the `dataVolumeTemplates` to notify CDI to create Block data volumes.
- *RWX access mode for block-storage drivers*: ontap-san (iSCSI, NVMe/TCP, FC) and ontap-san-economy (iSCSI) drivers are supported only with "volumeMode: Block" (raw device). For these drivers, the `fstype` parameter cannot be used because the volumes are provided in raw device mode.
- For live-migration workflow/s where RWX access mode is required, these combinations are supported:
  - NFS + `volumeMode=Filesystem`
  - iSCSI + `volumeMode=Block` (raw device)
  - NVMe/TCP + `volumeMode=Block` (raw device)
  - FC + `volumeMode=Block` (raw device)

Feature requirements

The table below summarizes the features available with this release of Trident and the versions of Kubernetes it supports.

Feature	Kubernetes version	Feature gates required?
Trident	1.27 - 1.36	No
Volume Snapshots	1.27 - 1.36	No
PVC from Volume Snapshots	1.27 - 1.36	No
iSCSI PV resize	1.27 - 1.36	No
ONTAP Bidirectional CHAP	1.27 - 1.36	No
Dynamic Export Policies	1.27 - 1.36	No
Trident Operator	1.27 - 1.36	No
CSI Topology	1.27 - 1.36	No

Tested host operating systems

Though Trident does not officially support specific operating systems, the following are known to work:

- Red Hat Enterprise Linux CoreOS (RHCOS) versions supported by OpenShift Container Platform on AMD64 and ARM64
- Red Hat Enterprise Linux (RHEL) 8 or later on AMD64 and ARM64

  NVMe/TCP requires RHEL 9 or later.

- Ubuntu 22.04 LTS or later on AMD64 and ARM64
- Windows Server 2022
- SUSE Linux Enterprise Server (SLES) 15 or later

By default, Trident runs in a container and will, therefore, run on any Linux worker. However, those workers need to be able to mount the volumes that Trident provides using the standard NFS client or iSCSI initiator, depending on the backends you are using.

The `tridentctl` utility also runs on any of these distributions of Linux.

Host configuration

All worker nodes in the Kubernetes cluster must be able to mount the volumes you have provisioned for your pods. To prepare the worker nodes, you must install NFS, iSCSI, or NVMe tools based on your driver selection.

Prepare the worker node

Storage system configuration

Trident might require changes to a storage system before a backend configuration can use it.

Configure backends

Trident ports

Trident requires access to specific ports for communication.

Trident ports

Container images and corresponding Kubernetes versions

For air-gapped installations, the following list is a reference of container images needed to install Trident. Use the `tridentctl images` command to verify the list of needed container images.

Container images required for Trident 26.02

Kubernetes versions	Container image
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0, v1.36.0	• docker.io/netapp/trident:26.02.0 • docker.io/netapp/trident-autosupport:26.02 • registry.k8s.io/sig-storage/csi-provisioner:v6.1.0 • registry.k8s.io/sig-storage/csi-attacher:v4.10.0 • registry.k8s.io/sig-storage/csi-resizer:v2.0.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.5.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.15.0 • docker.io/netapp/trident-operator:26.02.0 (optional)

Copyright information

Trademark information