Install Trident Protect

Trident

NetApp

February 02, 2026

Table of Contents

Install Trident Protect

Trident Protect requirements

Get started by verifying the readiness of your operational environment, application clusters, applications, and licenses. Ensure that your environment meets these requirements to deploy and operate Trident Protect.

Trident Protect Kubernetes cluster compatibility

Trident Protect is compatible with a wide range of fully managed and self-managed Kubernetes offerings, including:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Rancher
- VMware Tanzu Portfolio
- Upstream Kubernetes

> - Trident Protect backups are supported on Linux compute nodes only. Windows compute nodes are not supported for backup operations.
> - Ensure that the cluster on which you install Trident Protect is configured with a running snapshot controller and the related CRDs. To install a snapshot controller, refer to these instructions.
> - Ensure that at least one VolumeSnapshotClass exists. For more information, refer to VolumeSnapshotClass.

Trident Protect storage backend compatibility

Trident Protect supports the following storage backends:

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- ONTAP storage arrays
- Google Cloud NetApp Volumes
- Azure NetApp Files

Ensure that your storage backend meets the following requirements:

- Ensure that NetApp storage connected to the cluster is using Trident 24.02 or newer (Trident 24.10 is recommended).

- Ensure that you have a NetApp ONTAP storage backend.

- Ensure that you have configured an object storage bucket for storing backups.
- Create any application namespaces that you plan to use for applications or application data management operations. Trident Protect does not create these namespaces for you; if you specify a nonexistent namespace in a custom resource, the operation will fail.

Requirements for nas-economy volumes

Trident Protect supports backup and restore operations to nas-economy volumes. Snapshots, clones, and SnapMirror replication to nas-economy volumes are not currently supported. You need to enable a snapshot directory for each nas-economy volume you plan to use with Trident Protect.

Some applications are not compatible with volumes that use a snapshot directory. For these applications, you need to hide the snapshot directory by running the following command on the ONTAP storage system:

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

You can enable the snapshot directory by running the following command for each nas-economy volume, replacing `<volume-UUID>` with the UUID of the volume you want to change:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

You can enable snapshot directories by default for new volumes by setting the Trident backend configuration option `snapshotDir` to `true`. Existing volumes are not affected.

Protecting data with KubeVirt VMs

Trident Protect provides filesystem freeze and unfreeze capabilities for KubeVirt virtual machines during data protection operations to ensure data consistency. The configuration method and default behavior for VM freeze operations varies across Trident Protect versions, with newer releases offering simplified configuration through Helm chart parameters.

During restore operations, any `VirtualMachineSnapshots` created for a virtual machine (VM) are not restored.

Trident Protect 25.10 and newer

Trident Protect automatically freezes and unfreezes KubeVirt filesystems during data protection operations to ensure consistency. Beginning with Trident Protect 25.10, you can disable this behavior using the `vm.freeze` parameter during Helm chart installation. The parameter is enabled by default.

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 to 25.06

Beginning with Trident Protect 24.10.1, Trident Protect automatically freezes and unfreezes KubeVirt filesystems during data protection operations. Optionally, you can disable this automatic behavior using the following command:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

Trident Protect 24.10

Trident Protect 24.10 does not automatically ensure a consistent state for KubeVirt VM filesystems during data protection operations. If you want to protect your KubeVirt VM data using Trident Protect 24.10, you need to manually enable the freeze/unfreeze functionality for the filesystems before the data protection operation. This ensures that the filesystems are in a consistent state.

You can configure Trident Protect 24.10 to manage the freezing and unfreezing of the VM filesystem during data protection operations by configuring virtualization and then using the following command:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Requirements for SnapMirror replication

NetApp SnapMirror replication is available for use with Trident Protect for the following ONTAP solutions:

- On-premises NetApp FAS, AFF, and ASA clusters
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

ONTAP cluster requirements for SnapMirror replication

Ensure your ONTAP cluster meets the following requirements if you plan to use SnapMirror replication:

- **NetApp Trident**: NetApp Trident must exist on both the source and destination Kubernetes clusters that utilize ONTAP as a backend. Trident Protect supports replication with NetApp SnapMirror technology using storage classes backed by the following drivers:
  - `ontap-nas`: NFS
  - `ontap-san`: iSCSI
  - `ontap-san`: FC
  - `ontap-san`: NVMe/TCP (requires minimum ONTAP version 9.15.1)
- **Licenses**: ONTAP SnapMirror asynchronous licenses using the Data Protection bundle must be enabled on both the source and destination ONTAP clusters. Refer to SnapMirror licensing overview in ONTAP for

more information.

Beginning with ONTAP 9.10.1, all licenses are delivered as a NetApp license file (NLF), which is a single file that enables multiple features. Refer to Licenses included with ONTAP One for more information.

Only SnapMirror asynchronous protection is supported.

Peering considerations for SnapMirror replication

Ensure your environment meets the following requirements if you plan to use storage backend peering:

- **Cluster and SVM**: The ONTAP storage backends must be peered. Refer to Cluster and SVM peering overview for more information.

  Ensure that the SVM names used in the replication relationship between two ONTAP clusters are unique.

- **NetApp Trident and SVM**: The peered remote SVMs must be available to NetApp Trident on the destination cluster.
- **Managed backends**: You need to add and manage ONTAP storage backends in Trident Protect to create a replication relationship.

Trident / ONTAP configuration for SnapMirror replication

Trident Protect requires that you configure at least one storage backend that supports replication for both the source and destination clusters. If the source and destination clusters are the same, the destination application should use a different storage backend than the source application for the best resiliency.

Kubernetes cluster requirements for SnapMirror replication

Ensure your Kubernetes clusters meet the following requirements:

- **AppVault accessibility**: Both source and destination clusters must have network access to read from and write to the AppVault for application object replication.
- **Network connectivity**: Configure firewall rules, bucket permissions, and IP allowlists to enable communication between both clusters and the AppVault across WANs.

Many enterprise environments implement strict firewall policies across WAN connections. Verify these network requirements with your infrastructure team before configuring replication.

Install and configure Trident Protect

If your environment meets the requirements for Trident Protect, you can follow these steps to install Trident Protect on your cluster. You can obtain Trident Protect from NetApp, or install it from your own private registry. Installing from a private registry is helpful if your cluster cannot access the Internet.

Install Trident Protect

Install Trident Protect from NetApp

Steps

1. Add the Trident Helm repository:

```
helm repo add netapp-trident-protect
https://netapp.github.io/trident-protect-helm-chart
```

2. Use Helm to install Trident Protect. Replace `<name-of-cluster>` with a cluster name, which will be assigned to the cluster and used to identify the cluster's backups and snapshots:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect
```

3. Optionally, to enable debug logging (recommended for troubleshooting), use:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect
```

Debug logging helps NetApp support troubleshoot issues without requiring log level changes or problem reproduction.

Install Trident Protect from a private registry

You can install Trident Protect from a private image registry if your Kubernetes cluster is unable to access the Internet. In these examples, replace values in brackets with information from your environment:

Steps

1. Pull the following images to your local machine, update the tags, and then push them to your private registry:

```
docker.io/netapp/controller:25.10.0
docker.io/netapp/restic:25.10.0
docker.io/netapp/kopia:25.10.0
docker.io/netapp/kopiablockrestore:25.10.0
docker.io/netapp/trident-autosupport:25.10.0
docker.io/netapp/exechook:25.10.0
docker.io/netapp/resourcebackup:25.10.0
docker.io/netapp/resourcerestore:25.10.0
docker.io/netapp/resourcedelete:25.10.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

For example:

```
docker pull docker.io/netapp/controller:25.10.0
```

```
docker tag docker.io/netapp/controller:25.10.0 <private-registry-
url>/controller:25.10.0
```

```
docker push <private-registry-url>/controller:25.10.0
```

ⓘ To obtain the Helm chart, first download the Helm chart on a machine with internet access using `helm pull trident-protect --version 100.2510.0 --repo https://netapp.github.io/trident-protect-helm-chart`, then copy the resulting `trident-protect-100.2510.0.tgz` file to your offline environment and install using `helm install trident-protect ./trident-protect-100.2510.0.tgz` instead of the repository reference in the final step.

2. Create the Trident Protect system namespace:

```
kubectl create ns trident-protect
```

3. Log in to the registry:

```
helm registry login <private-registry-url> -u <account-id> -p <api-
token>
```

4. Create a pull secret to use for private registry authentication:

```
kubectl create secret docker-registry regcred --docker
-username=<registry-username> --docker-password=<api-token> -n
trident-protect --docker-server=<private-registry-url>
```

5. Add the Trident Helm repository:

```
helm repo add netapp-trident-protect
https://netapp.github.io/trident-protect-helm-chart
```

6. Create a file named `protectValues.yaml`. Ensure that it contains the following Trident Protect settings:

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

The `imageRegistry` and `imagePullSecrets` values apply to all component images including `resourcebackup` and `resourcerestore`. If you push images to a specific repository path within your registry (for example, `example.com:443/my-repo`), include the full path in the registry field. This will ensure that all images are pulled from `<private-registry-url>/<image-name>:<tag>`.

7. Use Helm to install Trident Protect. Replace `<name_of_cluster>` with a cluster name, which will be assigned to the cluster and used to identify the cluster's backups and snapshots:

   ```
   helm install trident-protect netapp-trident-protect/trident-protect
   --set clusterName=<name_of_cluster> --version 100.2510.0 --create
   -namespace --namespace trident-protect -f protectValues.yaml
   ```

8. Optionally, to enable debug logging (recommended for troubleshooting), use:

   ```
   helm install trident-protect netapp-trident-protect/trident-protect
   --set clusterName=<name-of-cluster> --set logLevel=debug --version
   100.2510.0 --create-namespace --namespace trident-protect -f
   protectValues.yaml
   ```

   Debug logging helps NetApp support troubleshoot issues without requiring log level changes or problem reproduction.

For additional Helm chart configuration options, including AutoSupport settings and namespace filtering, refer to Customize Trident Protect installation.

Install the Trident Protect CLI plugin

You can use the Trident Protect command line plugin, which is an extension of the Trident `tridentctl` utility, to create and interact with Trident Protect custom resources (CRs).

Install the Trident Protect CLI plugin

Before using the command line utility, you need to install it on the machine you use to access your cluster. Follow these steps, depending on if your machine uses an x64 or ARM CPU.

Download plugin for Linux AMD64 CPUs

Steps

1. Download the Trident Protect CLI plugin:

   ```
   curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-
   protect/releases/download/25.10.0/tridentctl-protect-linux-amd64
   ```

Download plugin for Linux ARM64 CPUs

Steps

1. Download the Trident Protect CLI plugin:

   ```
   curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-
   protect/releases/download/25.10.0/tridentctl-protect-linux-arm64
   ```

Download plugin for Mac AMD64 CPUs

Steps

1. Download the Trident Protect CLI plugin:

   ```
   curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-
   protect/releases/download/25.10.0/tridentctl-protect-macos-amd64
   ```

Download plugin for Mac ARM64 CPUs

Steps

1. Download the Trident Protect CLI plugin:

   ```
   curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-
   protect/releases/download/25.10.0/tridentctl-protect-macos-arm64
   ```

2. Enable execute permissions for the plugin binary:

   ```
   chmod +x tridentctl-protect
   ```

3. Copy the plugin binary to a location that is defined in your PATH variable. For example, `/usr/bin` or `/usr/local/bin` (you might need elevated privileges):

   ```
   cp ./tridentctl-protect /usr/local/bin/
   ```

4. Optionally, you can copy the plugin binary to a location in your home directory. In this case, it is recommended to ensure the location is part of your PATH variable:

```
cp ./tridentctl-protect ~/bin/
```

Copying the plugin to a location in your PATH variable enables you to use the plugin by typing `tridentctl-protect` or `tridentctl protect` from any location.

View Trident CLI plugin help

You can use the built-in plugin help features to get detailed help on the capabilities of the plugin:

Steps

1. Use the help function to view usage guidance:

```
tridentctl-protect help
```

Enable command auto-completion

After you have installed the Trident Protect CLI plugin, you can enable auto-completion for certain commands.

Enable auto-completion for the Bash shell

Steps

1. Create the completion script:

   ```
   tridentctl-protect completion bash > tridentctl-completion.bash
   ```

2. Make a new directory in your home directory to contain the script:

   ```
   mkdir -p ~/.bash/completions
   ```

3. Move the downloaded script to the `~/.bash/completions` directory:

   ```
   mv tridentctl-completion.bash ~/.bash/completions/
   ```

4. Add the following line to the `~/.bashrc` file in your home directory:

   ```
   source ~/.bash/completions/tridentctl-completion.bash
   ```

Enable auto-completion for the Z shell

Steps

1. Create the completion script:

   ```
   tridentctl-protect completion zsh > tridentctl-completion.zsh
   ```

2. Make a new directory in your home directory to contain the script:

   ```
   mkdir -p ~/.zsh/completions
   ```

3. Move the downloaded script to the `~/.zsh/completions` directory:

   ```
   mv tridentctl-completion.zsh ~/.zsh/completions/
   ```

4. Add the following line to the `~/.zprofile` file in your home directory:

   ```
   source ~/.zsh/completions/tridentctl-completion.zsh
   ```

Result

Upon your next shell login, you can use command auto-completion with the tridentctl-protect plugin.

Customize Trident Protect installation

You can customize the default configuration of Trident Protect to meet the specific requirements of your environment.

Specify Trident Protect container resource limits

You can use a configuration file to specify resource limits for Trident Protect containers after you install Trident Protect. Setting resource limits enables you to control how much of the cluster's resources are consumed by Trident Protect operations.

Steps

1. Create a file named `resourceLimits.yaml`.
2. Populate the file with resource limit options for Trident Protect containers according to the needs of your environment.

   The following example configuration file shows the available settings and contains the default values for each resource limit:

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. Apply the values from the `resourceLimits.yaml` file:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

Customize security context constraints

You can use a configuration file to modify OpenShift security context constraint (SCCs) for Trident Protect containers after you install Trident Protect. These constraints define security restrictions for pods in a Red Hat OpenShift cluster.

Steps

1. Create a file named `sccconfig.yaml`.
2. Add the SCC option to the file and modify the parameters according to the needs of your environment.

   The following example shows the default values of the parameters for the SCC option:

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

This table describes the parameters for the SCC option:

Parameter	Description	Default
create	Determines whether an SCC resource can be created. An SCC resource will be created only if `scc.create` is set to `true` and the Helm installation process identifies an OpenShift environment. If not operating on OpenShift, or if `scc.create` is set to `false`, no SCC resource will be created.	true
name	Specifies the name of the SCC.	trident-protect-job
priority	Defines the priority of the SCC. SCCs with higher priority values are assessed before those with lower values.	1

3. Apply the values from the `sccconfig.yaml` file:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

This will replace the default values with those specified in the `sccconfig.yaml` file.

Configure additional Trident Protect helm chart settings

You can customize AutoSupport settings and namespace filtering to meet your specific requirements. The following table describes the available configuration parameters:

Parameter	Type	Description
autoSupport.proxy	string	Configures a proxy URL for NetApp AutoSupport connections. Use this to route support bundle uploads through a proxy server. Example: `http://my.proxy.url`.
autoSupport.insecure	boolean	Skips TLS verification for AutoSupport proxy connections when set to `true`. Use only for insecure proxy connections. (default: `false`)

Parameter	Type	Description
autoSupport.enabled	boolean	Enables or disables daily Trident Protect AutoSupport bundle uploads. When set to `false`, scheduled daily uploads are disabled, but you can still manually generate support bundles. (default: `true`)
restoreSkipNamespaceAnnotations	string	Comma-separated list of namespace annotations to exclude from backup and restore operations. Allows you to filter namespaces based on annotations.
restoreSkipNamespaceLabels	string	Comma-separated list of namespace labels to exclude from backup and restore operations. Allows you to filter namespaces based on labels.

You can configure these options using either a YAML configuration file or command-line flags:

Use YAML file

Steps

1. Create a configuration file and name it `values.yaml`.
2. In the file you created, add the configuration options you want to customize.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. After you populate the `values.yaml` file with the correct values, apply the configuration file:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

Use CLI flag

Steps

1. Use the following command with the `--set` flag to specify individual parameters:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --reuse-values
```

Restrict Trident Protect pods to specific nodes

You can use the Kubernetes nodeSelector node selection constraint to control which of your nodes are eligible to run Trident Protect pods, based on node labels. By default, Trident Protect is restricted to nodes that are running Linux. You can further customize these constraints depending on your needs.

Steps

1. Create a file named `nodeSelectorConfig.yaml`.
2. Add the nodeSelector option to the file and modify the file to add or change node labels to restrict according to the needs of your environment. For example, the following file contains the default OS restriction, but also targets a specific region and app name:

```yaml
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Apply the values from the `nodeSelectorConfig.yaml` file:

   ```
   helm upgrade trident-protect -n trident-protect netapp-trident-
   protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
   ```

   This replaces the default restrictions with those you specified in the `nodeSelectorConfig.yaml` file.