Protect Kubernetes workloads (Preview)

NetApp Backup and Recovery

NetApp
February 11, 2026

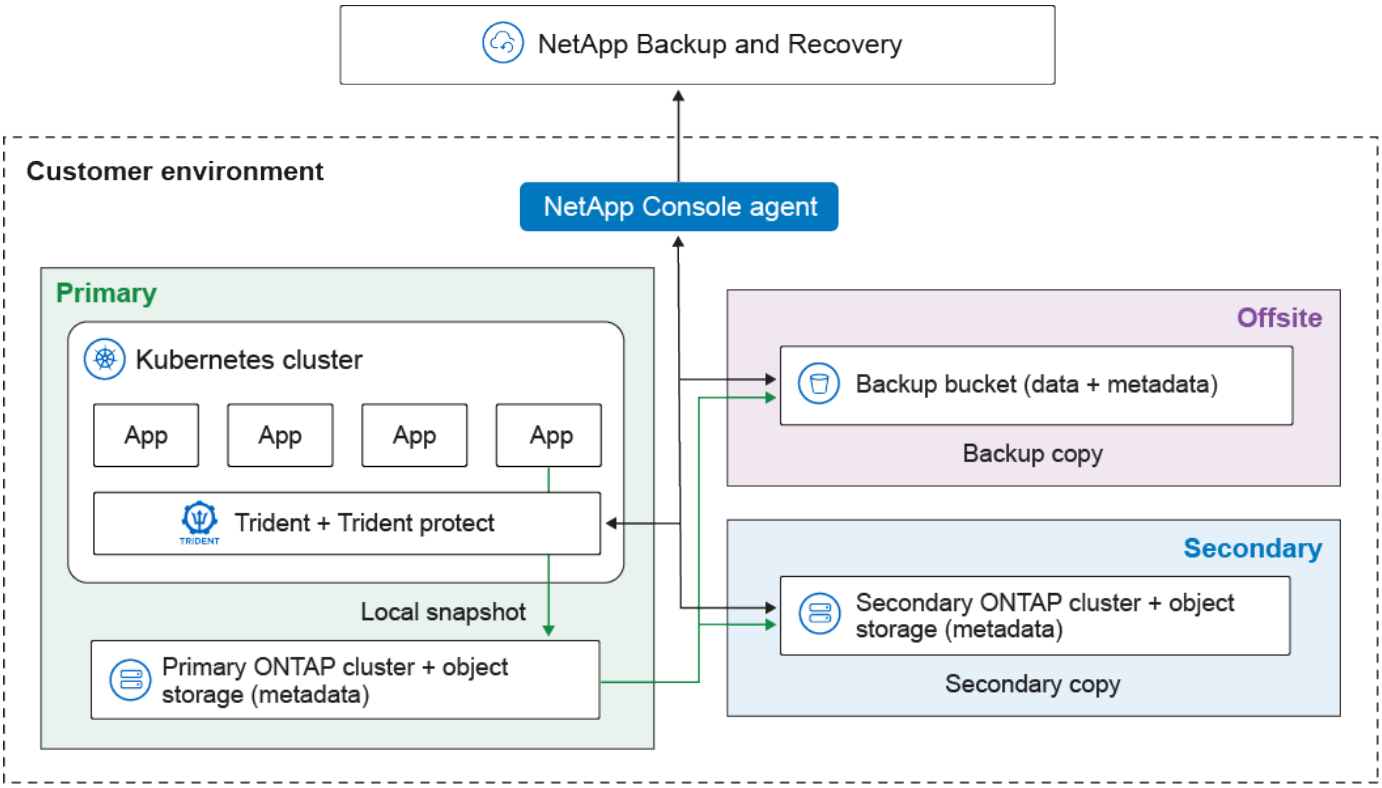
Table of Contents

Protect Kubernetes workloads (Preview)

Manage Kubernetes workloads overview

Managing Kubernetes workloads in NetApp Backup and Recovery enables you to discover, manage, and protect your Kubernetes clusters and applications all in one place. You can manage resources and applications hosted on your Kubernetes clusters. You can also create and associate protection policies with your Kubernetes workloads, all using a single interface.

The following diagram shows the components and basic architecture of backup and recovery for Kubernetes workloads and how different copies of your data can be stored in different locations:



NetApp Backup and Recovery provides the following benefits for managing Kubernetes workloads:

- A single control plane for protecting applications running across multiple Kubernetes clusters. These applications can include containers or virtual machines running on your Kubernetes clusters.
- Native integration with NetApp SnapMirror, enabling storage offloading capabilities for all backup and recovery workflows.
- Incremental forever backups for Kubernetes applications, translating to lower Recovery Point Objectives (RPOs) and Recovery Time Objectives (RTOs).

This documentation is provided as a technology preview. During the preview, Kubernetes functionality is not recommended for production workloads. With this preview offering, NetApp reserves the right to modify offering details, contents, and timeline before General Availability.

You can accomplish the following tasks related to managing Kubernetes workloads:

- Discover Kubernetes workloads.
- Manage Kubernetes clusters.
- Add and protect Kubernetes applications.
- Manage Kubernetes applications.
- Restore Kubernetes applications.

Discover Kubernetes workloads in NetApp Backup and Recovery

NetApp Backup and Recovery needs to discover Kubernetes workloads before protecting them.

Required NetApp Console role
Backup and Recovery super admin. Learn about Backup and recovery roles and privileges. Learn about NetApp Console access roles for all services.

Discover Kubernetes workloads

In Backup and Recovery inventory, discover Kubernetes workloads in your environment. Adding a workload adds a Kubernetes cluster to NetApp Backup and Recovery. You can then add applications and protect cluster resources.

> When you discover a cluster that is currently protected with Trident Protect, any backup schedules that were used with Trident Protect are disabled during the discovery process (Trident Protect backup schedules are not compatible with Backup and Recovery). To protect the cluster's applications, create a new protection policy or associate the applications with an existing policy. You can then remove the Trident Protect backup schedules if needed.

Steps

1. Do one of the following:
   - If you are discovering Kubernetes workloads for the first time, in NetApp Backup and Recovery, under **Workloads**, select the **Kubernetes** tile.
   - If you have already discovered Kubernetes workloads, in NetApp Backup and Recovery, select **Inventory** > **Workloads** and then select **Discover resources**.
2. Select the **Kubernetes** workload type.
3. Enter a cluster name and choose a connector to use with the cluster.
4. Follow the command line instructions that appear:
   - Create a Trident Protect namespace
   - Create a Kubernetes secret
   - Add a Helm repository
   - Install or upgrade Trident Protect and the Trident Protect connector

     These steps ensure that NetApp Backup and Recovery can interact with the cluster.
5. After you complete the steps, select **Discover**.

The cluster is added to the inventory.

6. Select **View** in the associated Kubernetes workload to see the list of applications, clusters, and namespaces for that workload.

Continue to the NetApp Backup and Recovery Dashboard

Follow these steps to view the NetApp Backup and Recovery Dashboard.

1. From the NetApp Console menu, select **Protection** > **Backup and recovery**.
2. Select a workload tile (for example, Microsoft SQL Server).
3. From the Backup and Recovery menu, select **Dashboard**.
4. Review the health of data protection. The number of at risk or protected workloads increases based on the newly discovered, protected, and backed up workloads.

   Learn what the Dashboard shows you.

Add and protect Kubernetes applications

Add and protect Kubernetes applications

NetApp Backup and Recovery enables you to easily discover your Kubernetes clusters, without generating and uploading kubeconfig files. You can connect Kubernetes clusters and install the required software using simple commands copied from the NetApp Console user interface.

Required NetApp Console role

Organization admin or SnapCenter admin. Learn about NetApp Backup and Recovery access roles. Learn about NetApp Console access roles for all services.

Add and protect a new Kubernetes application

The first step in protecting Kubernetes applications is to create an application within NetApp Backup and Recovery. When you create an application, you make the Console aware of the running application on the Kubernetes cluster.

Before you begin

Before you can add and protect a Kubernetes application, you need to discover Kubernetes workloads.

Add an application using the web UI

Steps

1. In NetApp Backup and Recovery, select **Inventory**.
2. Choose a Kubernetes instance, and select **View** to view the resources associated with that instance.
3. Select the **Applications** tab.
4. Select **Create application**.
5. Enter a name for the application.
6. Optionally, choose any of the following fields to search for the resources you want to protect:
   - Associated cluster
   - Associated namespaces
   - Resource types
   - Label selectors
7. Optionally, select **Cluster Scoped Resources** to choose any resources that are scoped at the cluster level. If you include them, they are added to the application when you create it.
8. Optionally, select **Search** to find the resources based on your search criteria.

   The Console does not store the search parameters or results; the parameters are used to search the selected Kubernetes cluster for resources that can be included in the application.

9. The Console displays a list of resources that match your search criteria.
10. If the list contains the resources you want to protect, select **Next**.
11. Optionally, in the **Policy** area, choose an existing protection policy to protect the application or create a new policy. If you don't select a policy, the application is created without a protection policy. You can add a protection policy later.
12. In the **Prescripts and postscripts** area, enable and configure any prescript or postscript execution hooks that you want to run before or after backup operations. To enable prescripts or postscripts, you must have already created at least one execution hook template.
13. Select **Create**.

Result

The application is created and appears in the list of applications in the **Applications** tab of the Kubernetes inventory. The NetApp Console enables protection for the application based on your settings, and you can monitor the progress in the **Monitoring** area of backup and recovery.

Add an application using a CR

Steps

1. Create the destination application CR file:
   a. Create the custom resource (CR) file and name it (for example, `my-app-name.yaml`).
   b. Configure the following attributes:
      - **metadata.name**: (*Required*) The name of the application custom resource. Note the name you choose because other CR files needed for protection operations refer to this value.
      - **spec.includedNamespaces**: (*Required*) Use namespace and label selector to specify the

namespaces and resources that the application uses. The application namespace must be part of this list. The label selector is optional and can be used to filter resources within each specified namespace.

- **spec.includedClusterScopedResources**: (*Optional*) Use this attribute to specify cluster-scoped resources to be included in the application definition. This attribute allows you to select these resources based on their group, version, kind, and labels.
  - **groupVersionKind**: (*Required*) Specifies the API group, version, and kind of the cluster-scoped resource.
  - **labelSelector**: (*Optional*) Filters the cluster-scoped resources based on their labels.

c. Configure the following annotations, if needed:

- **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Optional*) This annotation is only applicable to applications defined from virtual machines, such as in KubeVirt environments, where filesystem freezes occur before snapshots. Specify whether this application can write to the filesystem during a snapshot. If set to true, the application ignores the global setting and can write to the filesystem during a snapshot. If set to false, the application ignores the global setting and the filesystem is frozen during a snapshot. If specified but the application has no virtual machines in the application definition, the annotation is ignored. If not specified, the application follows the global filesystem freeze setting.
- **protect.trident.netapp.io/protection-command**: (*Optional*) Use this annotation to instruct Backup and Recovery to protect or stop protecting the application. The possible values are `protect` or `unprotect`.
- **protect.trident.netapp.io/protection-policy-name**: (*Optional*) Use this annotation to specify the name of the Backup and Recovery protection policy that you want to use to protect this application. This protection policy must already exist in Backup and Recovery.

If you need to apply this annotation after an application has already been created, you can use the following command:

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

Example YAML:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
    protect.trident.netapp.io/protection-command: "protect"
    protect.trident.netapp.io/protection-policy-name: "policy-
name"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
          app: another-example-app
  includedClusterScopedResources:
    - groupVersionKind:
        group: rbac.authorization.k8s.io
        kind: ClusterRole
        version: v1
      labelSelector:
        matchLabels:
          mylabel: test
```

2. (*Optional*) Add filtering that includes or excludes resources marked with particular labels:
   - **resourceFilter.resourceSelectionCriteria**: (Required for filtering) Use `Include` or `Exclude` to include or exclude a resource defined in resourceMatchers. Add the following resourceMatchers parameters to define the resources to be included or excluded:
     - **resourceFilter.resourceMatchers**: An array of resourceMatcher objects. If you define multiple elements in this array, they match as an OR operation, and the fields inside each element (group, kind, version) match as an AND operation.
       - **resourceMatchers[].group**: (*Optional*) Group of the resource to be filtered.
       - **resourceMatchers[].kind**: (*Optional*) Kind of the resource to be filtered.
       - **resourceMatchers[].version**: (*Optional*) Version of the resource to be filtered.
       - **resourceMatchers[].names**: (*Optional*) Names in the Kubernetes metadata.name field of the resource to be filtered.
       - **resourceMatchers[].namespaces**: (*Optional*) Namespaces in the Kubernetes metadata.name field of the resource to be filtered.
       - **resourceMatchers[].labelSelectors**: (*Optional*) Label selector string in the Kubernetes

metadata.name field of the resource as defined in the Kubernetes documentation. For example: `"trident.netapp.io/os=linux"`.

> When both `resourceFilter` and `labelSelector` are used, `resourceFilter` runs first, and then `labelSelector` is applied to the resulting resources.

For example:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

3. After you create the application CR to match your environment, apply the CR. For example:

```
kubectl apply -f my-app-name.yaml
```

Back up Kubernetes applications now using the Backup and Recovery web UI

NetApp Backup and Recovery enables you to manually back up Kubernetes applications using the web interface.

Required NetApp Console role

Organization admin or SnapCenter admin. Learn about NetApp Backup and Recovery access roles. Learn about NetApp Console access roles for all services.

Back up a Kubernetes application now using the web UI

Manually create a backup of a Kubernetes application to establish a baseline for future backups and snapshots, or to ensure the most recent data is protected.

Steps

1. In NetApp Backup and Recovery, select **Inventory**.
2. Choose a Kubernetes instance, and select **View** to view the resources associated with that instance.
3. Select the **Applications** tab.
4. In the list of applications, choose an application you want to back up and select the associated Actions menu.
5. Select **Backup now**.
6. Ensure the correct application name is selected.
7. Select **Back up**.

Result

The Console creates a backup of the application and displays the progress in the **Monitoring** area of Backup and Recovery. The backup is created based on the protection policy associated with the application.

Back up Kubernetes applications now using custom resources in Backup and Recovery

NetApp Backup and Recovery enables you to manually back up Kubernetes applications using custom resources (CRs).

Back up a Kubernetes application now using custom resources

Manually create a backup of a Kubernetes application to establish a baseline for future backups and snapshots, or to ensure the most recent data is protected.

> Cluster-scoped resources are included in a backup, snapshot, or clone if they are explicitly referenced in the application definition or if they have references to any of the application namespaces.

Unresolved directive in br-use-back-up-now-kubernetes-applications-cr.adoc - include::../_include/backup-include-sessiontoken-note.adoc[]

Create a local snapshot using a custom resource

To create a snapshot of your Kubernetes application and store it locally, use the Snapshot custom resource with specific attributes.

Steps

1. Create the custom resource (CR) file and name it `local-snapshot-cr.yaml`.
2. In the file you created, configure the following attributes:
   - **metadata.name**: (*Required*) The name of this custom resource; choose a unique and sensible name for your environment.
   - **spec.applicationRef**: The Kubernetes name of the application to snapshot.
   - **spec.appVaultRef**: (*Required*) The name of the AppVault where the snapshot contents (metadata) should be stored.
   - **spec.reclaimPolicy**: (*Optional*) Defines what happens to the AppArchive of a snapshot when the snapshot CR is deleted. This means that even when set to `Retain`, the snapshot will be deleted. Valid options:

- `Retain` (default)
- `Delete`

```
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: local-snapshot-cr
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Retain
```

3. After you populate the `local-snapshot-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -f local-snapshot-cr.yaml
```

Back up an application to an object store using a custom resource

Create a Backup CR with specific attributes to back up your application to an object store.

Steps

1. Create the custom resource (CR) file and name it `object-store-backup-cr.yaml`.
2. In the file you created, configure the following attributes:
   - **metadata.name**: (*Required*) The name of this custom resource; choose a unique and sensible name for your environment.
   - **spec.applicationRef**: (*Required*) The Kubernetes name of the application to back up.
   - **spec.appVaultRef**: (*Required, mutually exclusive with spec.appVaultTargetsRef*) If you use the same bucket to store the snapshot and backup, this is the name of the AppVault where the backup contents should be stored.
   - **spec.appVaultTargetsRef**: (*Required, mutually exclusive with spec.appVaultRef*) If you use different buckets to store the snapshot and backup, this is the name of the AppVault where the backup contents should be stored.
   - **spec.dataMover**: (*Optional*) A string indicating which backup tool to use for the backup operation. The value is case sensitive and must be `CBS`.
   - **spec.reclaimPolicy**: (*Optional*) Defines what happens to the backup contents (metadata/volume data) when the Backup CR is deleted. Possible values:
     - `Delete`
     - `Retain` (default)
   - **spec.cleanupSnapshot**: (*Required*) Ensures that the temporary snapshot created by the Backup CR is not deleted after the backup operation completes. Recommended value: `false`.

     Example YAML when using the same bucket to store the snapshot and backup:

```
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: CBS
  reclaimPolicy: Retain
  cleanupSnapshot: false
```

Example YAML when using different buckets to store the snapshot and backup:

```
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: object-store-backup-cr
spec:
  applicationRef: my-application
  appVaultTargetsRef: appvault-targets-name
  dataMover: CBS
  reclaimPolicy: Retain
  cleanupSnapshot: false
```

3. After you populate the `object-store-backup-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -f object-store-backup-cr.yaml
```

Create a 3-2-1 fanout backup using a custom resource

Backing up using a 3-2-1 fanout architecture copies a backup to secondary storage as well as to an object store. To create a 3-2-1 fanout backup, create a Backup CR with specific attributes.

Steps

1. Create the custom resource (CR) file and name it `3-2-1-fanout-backup-cr.yaml`.
2. In the file you created, configure the following attributes:
   - **metadata.name**: (*Required*) The name of this custom resource; choose a unique and sensible name for your environment.
   - **spec.applicationRef**: (*Required*) The Kubernetes name of the application to back up.
   - **spec.appVaultTargetsRef**: (*Required*) The name of the AppVault where the backup contents should be stored.

- **spec.dataMover**: (*Optional*) A string indicating which backup tool to use for the backup operation. The value is case sensitive and must be `CBS`.
- **spec.reclaimPolicy**: (*Optional*) Defines what happens to the backup contents (metadata/volume data) when the Backup CR is deleted. Possible values:
  - `Delete`
  - `Retain` (default)
- **spec.cleanupSnapshot**: (*Required*) Ensures that the temporary snapshot created by the Backup CR is not deleted after the backup operation completes. Recommended value: `false`.
- **spec.replicateSnapshot**: (*Required*) Instructs Backup and Recovery to replicate the snapshot to secondary storage. Required value: `true`.
- **spec.replicateSnapshotReclaimPolicy**: (*Optional*) Defines what happens to the replicated snapshot when it is deleted. Possible values:
  - `Delete`
  - `Retain` (default)

  Example YAML:

```yaml
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: 3-2-1-fanout-backup-cr
spec:
  applicationRef: my-application
  appVaultTargetsRef: appvault-targets-name
  dataMover: CBS
  reclaimPolicy: Retain
  cleanupSnapshot: false
  replicateSnapshot: true
  replicateSnapshotReclaimPolicy: Retain
```

3. After you populate the `3-2-1-fanout-backup-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -f 3-2-1-fanout-backup-cr.yaml
```

Supported backup annotations

The following table describes the annotations you can use when creating a backup CR.

Annotation	Type	Description	Default value
protect.trident.netapp.io/full-backup	string	Specifies whether a backup should be non-incremental. Set to `true` to create a non-incremental backup. It is best practice to perform a full backup periodically and then perform incremental backups in between full backups to minimize the risk associated with restores.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	string	The maximum time allowed for the overall snapshot operation to complete.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	string	The maximum time allowed for volume snapshots to reach the ready-to-use state.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	string	The maximum time allowed for volume snapshots to be created.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	string	Maximum time (in seconds) to wait for any newly created PersistentVolumeClaims (PVCs) to reach the `Bound` phase before the operations fails.	"1200" (20 minutes)

Restore Kubernetes applications

Restore Kubernetes applications using the web UI

NetApp Backup and Recovery enables you to restore applications that you have protected with a protection policy. To restore an application, an application needs to have at least one restore point available. A restore point consists of either the local snapshot or the backup to the object store (or both). You can restore an application using the local, secondary, or object store archive.

Before you begin

If you are restoring an application that was backed up using Trident Protect, ensure that Trident Protect is installed on both the source and destination clusters.

Required NetApp Console role

Organization admin or SnapCenter admin. Learn about NetApp Backup and Recovery access roles. Learn about NetApp Console access roles for all services.

Steps

1. In the NetApp Backup and Recovery menu, select **Restore**.
2. Choose a Kubernetes application from the list, and select **View and Restore** for that application.

   The list of restore points appears.

3. Select the **Restore** button for the restore point you want to use.

General settings

1. Choose the source location to restore from.
2. Choose the destination cluster from the **Cluster** list.

   Restoring a local snapshot created by Trident Protect to a different cluster is not supported at this time.

3. Choose to restore to the original namespaces or new namespaces.
4. If you chose to restore to new namespaces, enter the destination namespace or namespaces to use.
5. Select **Next**.

Resource selection

1. Choose whether you want to restore all resources associated with the application or use a filter to select specific resources to restore:

Restore all resources

a. Select **Restore all resources**.

b. Select **Next**.

Restore specific resources

a. Select **Selective resources**.

b. Choose the behavior of the resource filter. If you choose **Include**, the resources you select are restored. If you choose **Exclude**, the resources you select are not restored.

c. Select **Add rules** to add rules that define filters for selecting resources. You need at least one rule to filter resources.

   Each rule can filter on criteria such as the resource namespace, labels, group, version, and kind.

d. Select **Save** to save each rule.

e. When you have added all the rules you need, select **Search** to see the resources available in the backup archive that match your filter criteria.

   The resources shown are the resources that currently exist on the cluster.

f. When satisfied with the results, select **Next**.

Destination settings

1. Expand the **Destination settings** section and choose to restore either to the default storage class, a different storage class, or if you are restoring to a different cluster, to map the storage classes to the destination cluster.
2. If you chose to restore to a different storage class, select a destination storage class to match each source storage class.
3. Optionally, if you are restoring a backup or snapshot that was made using Trident Protect, view the details of the AppVault used as the storage bucket for the restore operation. If there is a change in your environment or the AppVault status, select **Sync App Vault** to refresh the details.

If you need to create an AppVault on a Kubernetes cluster to facilitate restoring a backup or snapshot created using Trident Protect, refer to Use Trident Protect AppVault objects to manage buckets.

4. Optionally, expand the **Restore scripts** section and enable the **Postscript** option to choose an execution hook template that will run after the restore operation is complete. If needed, enter any arguments that the script needs and add label selectors to filter resources based on resource labels.
5. Select **Restore**.

Restore Kubernetes applications using a custom resource

You can use custom resources to restore your applications from a snapshot or backup. Restoring from an existing snapshot will be faster when restoring the application to the same cluster.

- When you restore an application, all execution hooks configured for the application are restored with the app. If a post-restore execution hook is present, it runs automatically as part of the restore operation.
- Restoring from a backup to a different namespace or to the original namespace is supported for qtree volumes. However, restoring from a snapshot to a different namespace or to the original namespace is not supported for qtree volumes.
- You can use advanced settings to customize restore operations. To learn more, refer to Use advanced custom resource restore settings.

Restore a backup to a different namespace

When you restore a backup to a different namespace using a BackupRestore CR, Backup and Recovery restores the application in a new namespace and creates an application CR for the restored application. To protect the restored application, create on-demand backups or snapshots, or establish a protection schedule.

- Restoring a backup to a different namespace with existing resources will not alter any resources that share names with those in the backup. To restore all resources in the backup, either delete and re-create the target namespace, or restore the backup to a new namespace.
- When using a CR to restore to a new namespace, you must manually create the destination namespace before applying the CR. Backup and Recovery automatically creates namespaces only when using the CLI.

Unresolved directive in br-use-restore-kubernetes-applications-cr.adoc - include::../_include/restore-include-sessiontoken-note.adoc[]

When you restore backups using Kopia as the data mover, you can optionally specify annotations in the CR to control the behavior of the temporary storage used by Kopia. Refer to the Kopia documentation for more information about the options you can configure.

Steps

1. Create the custom resource (CR) file and name it `trident-protect-backup-restore-cr.yaml`.
2. In the file you created, configure the following attributes:

- **metadata.name**: (*Required*) The name of this custom resource; choose a unique and sensible name for your environment.
- **spec.appArchivePath**: The path inside AppVault where the backup contents are stored. You can use the following command to find this path:

```
kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath
='{.status.appArchivePath}'
```

- **spec.appVaultRef**: (*Required*) The name of the AppVault where the backup contents are stored.
- **spec.namespaceMapping**: The mapping of the source namespace of the restore operation to the destination namespace. Replace `my-source-namespace` and `my-destination-namespace` with information from your environment.

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
  namespaceMapping: [{"source": "my-source-namespace", "destination":
"my-destination-namespace"}]
```

Unresolved directive in br-use-restore-kubernetes-applications-cr.adoc - include::../_include/restore-include-selective-restore.adoc[]

3. After you populate the `trident-protect-backup-restore-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Restore a backup to the original namespace

You can restore a backup to the original namespace at any time.

Unresolved directive in br-use-restore-kubernetes-applications-cr.adoc - include::../_include/restore-include-sessiontoken-note.adoc[]

> When you restore backups using Kopia as the data mover, you can optionally specify annotations in the CR to control the behavior of the temporary storage used by Kopia. Refer to the Kopia documentation for more information about the options you can configure.

Steps

1. Create the custom resource (CR) file and name it `trident-protect-backup-ipr-cr.yaml`.

2. In the file you created, configure the following attributes:
   - **metadata.name**: (*Required*) The name of this custom resource; choose a unique and sensible name for your environment.
   - **spec.appArchivePath**: The path inside AppVault where the backup contents are stored. You can use the following command to find this path:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath
     ='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Required*) The name of the AppVault where the backup contents are stored.

     For example:

     ```
     apiVersion: protect.trident.netapp.io/v1
     kind: BackupInplaceRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appArchivePath: my-backup-path
       appVaultRef: appvault-name
     ```

     Unresolved directive in br-use-restore-kubernetes-applications-cr.adoc - include::../_include/restore-include-selective-restore.adoc[]

3. After you populate the `trident-protect-backup-ipr-cr.yaml` file with the correct values, apply the CR:

   ```
   kubectl apply -f trident-protect-backup-ipr-cr.yaml
   ```

Restore a backup to a different cluster

You can restore a backup to a different cluster if there is an issue with the original cluster.

- When you restore backups using Kopia as the data mover, you can optionally specify annotations in the CR to control the behavior of the temporary storage used by Kopia. Refer to the Kopia documentation for more information about the options you can configure.
- When using a CR to restore to a new namespace, you must manually create the destination namespace before applying the CR.

Before you begin

Ensure the following prerequisites are met:

- The destination cluster has Trident Protect installed.

- The destination cluster has access to the bucket path of the same AppVault as the source cluster, where the backup is stored.
- Ensure that the AWS session token expiration is sufficient for any long-running restore operations. If the token expires during the restore operation, the operation can fail.
  - Refer to the AWS API documentation for more information about checking the current session token expiration.
  - Refer to the AWS documentation for more information about credentials with AWS resources.

Steps

1. Check the availability of the AppVault CR on the destination cluster using Trident Protect CLI plugin:

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

> Ensure that the namespace intended for the application restore exists on the destination cluster.

2. View the backup contents of the available AppVault from the destination cluster:

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

Running this command displays the available backups in the AppVault, including their originating clusters, corresponding application names, timestamps, and archive paths.

Example output:

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   CLUSTER   |    APP    |  TYPE  |      NAME       |        TIMESTAMP
|    PATH     |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
| production1 | wordpress | backup | wordpress-bkup-1| 2024-10-30
08:37:40 (UTC)| backuppath1 |
| production1 | wordpress | backup | wordpress-bkup-2| 2024-10-30
08:37:40 (UTC)| backuppath2 |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
```

3. Restore the application to the destination cluster using the AppVault name and archive path:

4. Create the custom resource (CR) file and name it `trident-protect-backup-restore-cr.yaml`.

5. In the file you created, configure the following attributes:
   - **metadata.name**: (*Required*) The name of this custom resource; choose a unique and sensible name for your environment.
   - **spec.appVaultRef**: (*Required*) The name of the AppVault where the backup contents are stored.
   - **spec.appArchivePath**: The path inside AppVault where the backup contents are stored. You can use the following command to find this path:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath
     ='{.status.appArchivePath}'
     ```

     If BackupRestore CR is not available, you can use the command mentioned in step 2 to view the backup contents.

   - **spec.namespaceMapping**: The mapping of the source namespace of the restore operation to the destination namespace. Replace `my-source-namespace` and `my-destination-namespace` with information from your environment.

     For example:

     ```
     apiVersion: protect.trident.netapp.io/v1
     kind: BackupRestore
     metadata:
       name: my-cr-name
       namespace: my-destination-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-backup-path
       namespaceMapping: [{"source": "my-source-namespace", "destination":
     "my-destination-namespace"}]
     ```

6. After you populate the `trident-protect-backup-restore-cr.yaml` file with the correct values, apply the CR:

   ```
   kubectl apply -f trident-protect-backup-restore-cr.yaml
   ```

Restore a snapshot to a different namespace

You can restore data from a snapshot using a custom resource (CR) file either to a different namespace or the original source namespace. When you restore a snapshot to a different namespace using a SnapshotRestore CR, Backup and Recovery restores the application in a new namespace and creates an application CR for the restored application. To protect the restored application, create on-demand backups or snapshots, or establish a protection schedule.

- SnapshotRestore supports the `spec.storageClassMapping` attribute, but only when the source and destination storage classes use the same storage backend. If you attempt to restore to a `StorageClass` that uses a different storage backend, the restore operation will fail.
- When using a CR to restore to a new namespace, you must manually create the destination namespace before applying the CR.

Unresolved directive in br-use-restore-kubernetes-applications-cr.adoc - include::../_include/restore-include-sessiontoken-note.adoc[]

Steps

1. Create the custom resource (CR) file and name it `trident-protect-snapshot-restore-cr.yaml`.
2. In the file you created, configure the following attributes:
   - **metadata.name**: (*Required*) The name of this custom resource; choose a unique and sensible name for your environment.
   - **spec.appVaultRef**: (*Required*) The name of the AppVault where the snapshot contents are stored.
   - **spec.appArchivePath**: The path inside AppVault where the snapshot contents are stored. You can use the following command to find this path:

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o jsonpath
     ='{.status.appArchivePath}'
     ```

   - **spec.namespaceMapping**: The mapping of the source namespace of the restore operation to the destination namespace. Replace `my-source-namespace` and `my-destination-namespace` with information from your environment.

     ```
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace", "destination":
     "my-destination-namespace"}]
     ```

     Unresolved directive in br-use-restore-kubernetes-applications-cr.adoc - include::../_include/restore-include-selective-restore.adoc[]

3. After you populate the `trident-protect-snapshot-restore-cr.yaml` file with the correct values, apply the CR:

   ```
   kubectl apply -f trident-protect-snapshot-restore-cr.yaml
   ```

Restore a snapshot to the original namespace

You can restore a snapshot to the original namespace at any time.

Unresolved directive in br-use-restore-kubernetes-applications-cr.adoc - include::../_include/restore-include-sessiontoken-note.adoc[]

Steps

1. Create the custom resource (CR) file and name it `trident-protect-snapshot-ipr-cr.yaml`.
2. In the file you created, configure the following attributes:
   - **metadata.name**: (*Required*) The name of this custom resource; choose a unique and sensible name for your environment.
   - **spec.appVaultRef**: (*Required*) The name of the AppVault where the snapshot contents are stored.
   - **spec.appArchivePath**: The path inside AppVault where the snapshot contents are stored. You can use the following command to find this path:

     ```
     kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

     ```
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotInplaceRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
     ```

     Unresolved directive in br-use-restore-kubernetes-applications-cr.adoc - include::../_include/restore-include-selective-restore.adoc[]
3. After you populate the `trident-protect-snapshot-ipr-cr.yaml` file with the correct values, apply the CR:

   ```
   kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
   ```

Use advanced custom resource restore settings

You can customize restore operations using advanced settings such as annotations, namespace settings, and storage options to meet your specific requirements.

Unresolved directive in br-use-kubernetes-advanced-restore-settings.adoc - include::../_include/namespace-anno-labels.adoc[]

Supported fields

This section describes additional fields available for restore operations.

Storage class mapping

The `spec.storageClassMapping` attribute defines a mapping from a storage class present in the source application to a new storage class on the target cluster. You can use this when migrating applications between clusters with different storage classes or when changing the storage backend for BackupRestore operations.

Example:

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

Supported annotations

This section lists the supported annotations for configuring various behaviors in the system. If an annotation is not explicitly set by the user, the system will use the default value.

Annotation	Type	Description	Default value
protect.trident.netapp.io/data-mover-timeout-sec	string	The maximum time (in seconds) allowed for data mover operation to be stalled.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	string	The maximum size limit (in megabytes) for the Kopia content cache.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	string	Maximum time (in seconds) to wait for any newly created PersistentVolumeClaims (PVCs) to reach the `Bound` phase before the operations fails. Applies to all restore CR types (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Use a higher value if your storage backend or cluster often requires more time.	"1200" (20 minutes)

Manage Kubernetes clusters

NetApp Backup and Recovery enables you to discover and manage your Kubernetes clusters so that you can protect resources hosted by the clusters.

Required NetApp Console role

Organization admin or SnapCenter admin. Learn about NetApp Backup and Recovery access roles. Learn about NetApp Console access roles for all services.

To discover Kubernetes clusters, refer to Discover Kubernetes workloads.

Edit Kubernetes cluster information

You can edit a cluster if you need to change its name.

Steps

1. In NetApp Backup and Recovery, select **Inventory** > **Clusters**.
2. In the list of clusters, choose a cluster you want to edit and select the associated Actions menu.
3. Select **Edit cluster**.
4. Make any required changes to the cluster name. The cluster name needs to match the name that you used with the Helm command during the discovery process.
5. Select **Done**.

Remove a Kubernetes cluster

To stop protecting a Kubernetes cluster, disable protection and delete associated applications, then remove the cluster from NetApp Backup and Recovery. NetApp Backup and Recovery does not delete the cluster or its resources; it only removes the cluster from the NetApp Console inventory.

Steps

1. In NetApp Backup and Recovery, select **Inventory** > **Clusters**.
2. In the list of clusters, choose a cluster you want to edit and select the associated Actions menu.
3. Select **Remove cluster**.
4. Review the information in the confirmation dialog box, and select **Remove**.

Manage Kubernetes applications

NetApp Backup and Recovery enables you to unprotect and delete your Kubernetes applications and associated resources.

Required NetApp Console role

Organization admin or SnapCenter admin. Learn about NetApp Backup and Recovery access roles. Learn about NetApp Console access roles for all services.

Unprotect a Kubernetes application

You can unprotect an application if you no longer want to protect it. When you unprotect an application, NetApp Backup and Recovery stops protecting the application but keeps all associated backups and snapshots.

You cannot unprotect an application while protection operations are still running for it. Either wait for the operation to finish, or as a workaround, remove the restore point the running protection operation is using. You can then unprotect the application.

Steps

1. In NetApp Backup and Recovery, select **Inventory**.
2. Choose a Kubernetes instance, and select **View** to view the resources associated with that instance.

3. Select the **Applications** tab.
4. In the list of applications, choose an application you want to unprotect and select the associated Actions menu.
5. Select **Unprotect**.
6. Read the notice, and when ready, select **Unprotect**.

Delete a Kubernetes application

Delete an application you no longer need. NetApp Backup and Recovery stops protection and removes all backups and snapshots for deleted applications.

Steps

1. In NetApp Backup and Recovery, select **Inventory**.
2. Choose a Kubernetes instance, and select **View** to view the resources associated with that instance.
3. Select the **Applications** tab.
4. In the list of applications, choose an application you want to delete and select the associated Actions menu.
5. Select **Delete**.
6. Enable **Delete snapshots and backups** to remove all snapshots and backups of the application.

   You will no longer be able to restore the application using these snapshots and backups.

7. Confirm the action and select **Delete**.

Remove a restore point for a Kubernetes application

You might need to remove a restore point for an application if you need to unprotect it and protection operations are currently running.

Steps

1. In the NetApp Backup and Recovery menu, select **Restore**.
2. Choose a Kubernetes application from the list, and select **View and Restore** for that application.

   The list of restore points appears.

3. Choose the recovery point you need to delete and select the Actions icon ••• > **Delete recovery point** to delete it.

Manage NetApp Backup and Recovery execution hook templates for Kubernetes workloads

An execution hook is a custom action that runs with a data protection operation in a managed Kubernetes application. For example, create application-consistent snapshots by using an execution hook to pause database transactions before a snapshot and resume them after. When you create an execution hook template, specify the hook type, the script to run, and filters for target containers. Use the template to link execution hooks

to your applications.

NetApp Backup and Recovery freezes and unfreezes filesystems for applications like KubeVirt during data protection. You can disable this behavior globally or for specific applications using the Trident Protect documentation:

- To disable this behavior for all applications, refer to Protecting data with KubeVirt VMs.
- To disable this behavior for a specific application, refer to Define an application.

Required NetApp Console role

Organization admin or SnapCenter admin. Learn about NetApp Backup and Recovery access roles. Learn about NetApp Console access roles for all services.

Types of execution hooks

NetApp Backup and Recovery supports the following types of execution hooks, based on when they can be run:

- Pre-snapshot
- Post-snapshot
- Pre-backup
- Post-backup
- Post-restore

Order of execution

When a data protection operation is run, execution hook events take place in the following order:

1. Any applicable custom pre-operation execution hooks are run on the appropriate containers. You can create multiple custom pre-operation hooks, but their execution order is not guaranteed or configurable.
2. Filesystem freezes occur, if applicable.
3. The data protection operation is performed.
4. Frozen filesystems are unfrozen, if applicable.
5. NetApp Backup and Recovery runs any applicable custom pre-operation execution hooks on the appropriate containers. You can create multiple custom post-operation hooks, but their execution order is not guaranteed or configurable.

If you create multiple hooks of the same type, their execution order is not guaranteed. Hooks of different types always run in the specified order. For example, the following is the order of execution of a configuration that has all of the different types of hooks:

1. Pre-snapshot hooks executed
2. Post-snapshot hooks executed
3. Pre-backup hooks executed
4. Post-backup hooks executed

Test execution hook scripts before enabling them in production. Use 'kubectl exec' to test scripts, then verify snapshots and backups by cloning the app to a temporary namespace and restoring.

If a pre-snapshot execution hook adds, changes, or removes Kubernetes resources, those changes are included in the snapshot or backup and in any subsequent restore operation.

Important notes about custom execution hooks

Consider the following when planning execution hooks for your apps.

- An execution hook must use a script to perform actions. Many execution hooks can reference the same script.
- Execution hooks need to be written in the format of executable shell scripts.
- Script size is limited to 96KB.
- Execution hook settings and any matching criteria are used to determine which hooks are applicable to a snapshot, backup, or restore operation.

Execution hooks can reduce or disable application functionality. Make your custom hooks run as quickly as possible. If you start a backup or snapshot operation with associated execution hooks but then cancel it, the hooks are still allowed to run if the backup or snapshot operation has already begun. This means that the logic used in a post-backup execution hook cannot assume that the backup was completed.

Execution hook filters

When you add or edit an execution hook for an application, you can add filters to the execution hook to manage which containers the hook will match. Filters are useful for applications that use the same container image on all containers, but might use each image for a different purpose (such as Elasticsearch). Filters allow you to create scenarios where execution hooks run on some but not necessarily all identical containers. If you create multiple filters for a single execution hook, they are combined with a logical AND operator. You can have up to 10 active filters per execution hook.

Each filter you add to an execution hook uses a regular expression to match containers in your cluster. When a hook matches a container, the hook will run its associated script on that container. Regular expressions for filters use the Regular Expression 2 (RE2) syntax, which does not support creating a filter that excludes containers from the list of matches. For information on the syntax that NetApp Backup and Recovery supports for regular expressions in execution hook filters, see Regular Expression 2 (RE2) syntax support.

If you add a namespace filter to an execution hook that runs after a restore or clone operation and the restore or clone source and destination are in different namespaces, the namespace filter is only applied to the destination namespace.

Execution hook examples

Visit the NetApp Verda GitHub project to download real execution hooks for popular apps such as Apache Cassandra and Elasticsearch. You can also see examples and get ideas for structuring your own custom execution hooks.

Create an execution hook template

You can create a custom execution hook template that you can use to perform actions before or after a data protection operation on an application.

Templates that you create here are only usable when protecting Kubernetes workloads.

Steps

1. In the Console, go to **Protection** > **Backup and recovery**.
2. Select the **Settings** tab.
3. Expand the **Execution hook template** section.
4. Select **Create execution hook template**.
5. Enter a name for the execution hook.
6. Optionally, choose a type of hook. For example, a post-restore hook is run after the restore operation is complete.
7. In the **Script** text box, enter the executable shell script that you want to run as part of the execution hook template. Optionally, you can select **Upload script** to upload a script file instead.
8. Select **Create**.

   After you create the template, it appears in the list of templates in the **Execution hook template** section.

Copyright information

Trademark information