Add and protect Kubernetes applications

NetApp Backup and Recovery

NetApp
February 11, 2026

Table of Contents

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)

Copyright information

Trademark information