Gestiona y protege las aplicaciones

Trident

NetApp
October 29, 2026

This PDF was generated from https://docs.netapp.com/es-es/trident-2606/trident-protect/trident-protect-appvault-custom-resources.html on October 29, 2026. Always check docs.netapp.com for the latest.

Tabla de contenidos

Gestiona y protege las aplicaciones . . . 1
- Usa los objetos de Trident Protect AppVault para gestionar buckets . . . 1
  - Configura la autenticación y las contraseñas de AppVault . . . 1
  - Ejemplos de creación de AppVault . . . 5
  - Ver información de AppVault . . . 10
  - Eliminar un AppVault . . . 11
- Define una aplicación para gestión con Trident Protect . . . 12
  - Crea un CR de AppVault . . . 12
  - Define una aplicación . . . 12
- Protege las aplicaciones usando Trident Protect . . . 17
  - Crea una instantánea a pedido . . . 17
  - Crear un backup a pedido . . . 18
  - Crea un cronograma de protección de datos . . . 21
  - Borrar una instantánea . . . 25
  - Eliminar un backup . . . 25
  - Comprobar el estado de una operación de backup . . . 26
  - Habilita el backup y la restauración para las operaciones de azure-netapp-files (ANF) . . . 26
- Restaurar aplicaciones . . . 27
  - Restaura aplicaciones usando Trident Protect . . . 27
  - Usa la configuración avanzada de restauración de Trident Protect . . . 41
  - Modifica los recursos a medida que los restauras usando recursos personalizados . . . 44
- Replica aplicaciones usando NetApp SnapMirror y Trident Protect . . . 49
  - Anotaciones y etiquetas de namespace durante las operaciones de restauración y conmutación por error . . . 49
  - Ganchos de ejecución durante la conmutación por error y las operaciones inversas . . . 51
  - Configura una relación de replicación . . . 51
  - Invertir la dirección de replicación de la aplicación . . . 60
- Migra aplicaciones usando Trident Protect . . . 63
  - Operaciones de backup y restauración . . . 63
  - Migra aplicaciones de una clase de almacenamiento a otra clase de almacenamiento . . . 64
- Protege las máquinas virtuales de KubeVirt con Trident Protect . . . 67
  - KubeVirt modos de protección . . . 67
  - Recursos recopilados durante la copia de seguridad y la instantánea . . . 67
  - Protege máquinas virtuales usando definiciones de aplicaciones basadas en espacios de nombres . . . 69
  - Protección dinámica de máquinas virtuales con selectores de etiquetas . . . 70
  - Protege máquinas virtuales concretas con includedVirtualMachines . . . 72
  - Restaurar máquinas virtuales específicas de KubeVirt . . . 73
  - Crear una aplicación con ámbito de máquina virtual mediante la CLI . . . 78
  - Información relacionada . . . 78
- Administra los hooks de ejecución de Trident Protect . . . 78
  - Tipos de hooks de ejecución . . . 79
  - Notas importantes sobre los ganchos de ejecución personalizados . . . 80
  - Filtros de ejecución hook . . . 80

Ejemplos de execution hook 80
Crear un gancho de ejecución 81
Ejecuta manualmente un gancho de ejecución 83

Gestiona y protege las aplicaciones

Usa los objetos de Trident Protect AppVault para gestionar buckets

El recurso personalizado (CR) del bucket para Trident Protect se conoce como un AppVault. Los objetos AppVault son la representación declarativa del flujo de trabajo de Kubernetes de un bucket de almacenamiento. Un AppVault CR contiene las configuraciones necesarias para que un bucket se pueda usar en operaciones de protección, como copias de seguridad, instantáneas, operaciones de restauración y replicación SnapMirror. Solo los administradores pueden crear AppVaults.

Necesitas crear una CR de AppVault manualmente o desde la línea de comandos cuando realizas operaciones de protección de datos en una aplicación. La CR de AppVault es específica de tu entorno y puedes usar los ejemplos en esta página como guía cuando crees CR de AppVault.

> Asegúrate de que el CR AppVault está en el clúster donde está instalado Trident Protect. Si el CR AppVault no existe o no puedes acceder a él, la línea de comandos muestra un error.

Configura la autenticación y las contraseñas de AppVault

Antes de crear un AppVault CR, asegúrate de que el AppVault y el data mover que elijas puedan autenticarse con el proveedor y cualquier recurso relacionado.

Contraseñas del repositorio de data mover

Cuando creas objetos AppVault usando CRs o el complemento de línea de comandos (CLI) de Trident Protect, puedes especificar un secreto de Kubernetes con contraseñas personalizadas para el cifrado de Restic y Kopia. Si no especificas un secreto, Trident Protect usa una contraseña predeterminada.

- Al crear manualmente AppVault CRs, usa el campo **spec.dataMoverPasswordSecretRef** para especificar el secreto.
- Al crear objetos AppVault usando la CLI de Trident Protect, usa el argumento `--data-mover-password-secret-ref` para especificar el secreto.

Crea una contraseña secreta para el repositorio de data mover

Usa los siguientes ejemplos para crear el secreto de la contraseña. Cuando crees objetos AppVault, puedes indicarle a Trident Protect que use este secreto para autenticarse con el repositorio de data mover.

> - En función del motor de datos que utilices, solo tienes que incluir la contraseña correspondiente a ese motor de datos. Por ejemplo, si usas Restic y no planeas usar Kopia en el futuro, puedes incluir solo la contraseña de Restic cuando crees el secreto.
> - Guarda la contraseña en un lugar seguro. La vas a necesitar para restaurar datos en el mismo clúster o en uno diferente. Si se elimina el clúster o el `trident-protect` namespace, no podrás restaurar tus backups o snapshots sin la contraseña.

Usa un CR

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

Usa la CLI

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

Permisos IAM de almacenamiento compatible con S3

Cuando accedes a almacenamiento compatible con S3, como Amazon S3, Generic S3, "StorageGrid S3" o "ONTAP S3" usando Trident Protect, necesitas asegurarte de que las credenciales de usuario que proporciones tengan los permisos necesarios para acceder al bucket. El siguiente es un ejemplo de una política que otorga los permisos mínimos requeridos para el acceso con Trident Protect. Puedes aplicar esta política al usuario que administra las políticas de buckets compatibles con S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

Para obtener más información sobre las políticas de Amazon S3, consulta los ejemplos en "Documentación de Amazon S3".

EKS Pod Identity para Amazon S3 (AWS) authentication

Trident Protect es compatible con EKS Pod Identity para operaciones de traslado de datos de Kopia. Esta función permite el acceso seguro a los buckets de S3 sin almacenar credenciales de AWS en los secretos de Kubernetes.

Requisitos para EKS Pod Identity con Trident Protect

Antes de usar EKS Pod Identity con Trident Protect, asegúrate de lo siguiente:

- Tu clúster EKS tiene activado Pod Identity.
- Has creado una función de IAM con los permisos necesarios para el bucket de S3. Para obtener más información, consulta "Permisos IAM de almacenamiento compatible con S3".
- La función IAM está asociada con las siguientes cuentas de servicio de Trident Protect:
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Para obtener instrucciones detalladas sobre cómo habilitar Pod Identity y asociar roles IAM con cuentas de servicio, consulta la "Documentación de AWS EKS Pod Identity".

Configuración de AppVault Cuando uses EKS Pod Identity, configura tu CR de AppVault con la `useIAM: true` flag en vez de credenciales explícitas:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

AppVault ejemplos de generación de claves para proveedores de nube

Al definir un CR de AppVault, necesitas incluir credenciales para acceder a los recursos alojados por el proveedor, a menos que estés usando autenticación IAM. Cómo generes las claves para las credenciales dependerá del proveedor. A continuación tienes ejemplos de generación de claves por línea de comandos para varios proveedores. Puedes usar los siguientes ejemplos para crear claves para las credenciales de cada

proveedor de nube.

Google Cloud

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

Amazon S3 (AWS)

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

Microsoft Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

S3 genérico

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

ONTAP S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

StorageGrid S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src
-bucket-secret> \
```

```
-n trident-protect
```

Ejemplos de creación de AppVault

Los siguientes son ejemplos de definiciones de AppVault para cada proveedor.

AppVault ejemplos de CR

Puedes usar los siguientes ejemplos de CR para crear objetos AppVault para cada proveedor de nube.

- Puedes especificar opcionalmente un secreto de Kubernetes que contenga contraseñas personalizadas para el cifrado de los repositorios Restic y Kopia. Consulta Contraseñas del repositorio de data mover para obtener más información.
- Para los objetos de Amazon S3 (AWS) AppVault, puedes especificar opcionalmente un sessionToken, que es útil si usas el inicio de sesión único (SSO) para la autenticación. Este token se crea cuando generas claves para el proveedor en AppVault ejemplos de generación de claves para proveedores de nube.
- Para los objetos S3 AppVault, puedes especificar opcionalmente una URL de proxy de salida para el tráfico S3 saliente usando la clave `spec.providerConfig.S3.proxyURL`.

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3 (AWS)

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
```

```
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

Para entornos EKS que usan Pod Identity con Kopia data mover, puedes eliminar la sección `providerCredentials` y agregar `useIAM: true` dentro de la configuración `s3` en su lugar.

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
```

```
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

S3 genérico

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
```

```
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

StorageGrid S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

Ejemplos de creación de AppVault usando la CLI de Trident Protect

Puedes usar los siguientes ejemplos de comandos de la CLI para crear AppVault CRs para cada proveedor.

- Puedes especificar opcionalmente un secreto de Kubernetes que contenga contraseñas personalizadas para el cifrado de los repositorios Restic y Kopia. Consulta Contraseñas del repositorio de data mover para obtener más información.
- Para los objetos S3 AppVault, puedes especificar opcionalmente una URL de proxy de salida para el tráfico S3 saliente usando el argumento `--proxy-url <ip_address:port>`.

Google Cloud

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Microsoft Azure

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

S3 genérico

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

StorageGrid S3

```
tridentctl-protect create vault StorageGridS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Opciones de configuración compatibles `providerConfig.s3`

Consulta la siguiente tabla para ver las opciones de configuración del proveedor S3:

Parámetro	Descripción	Predeterminado	Ejemplo
`providerConfig.s3.skipCertValidation`	Desactiva la verificación de certificados SSL/TLS.	false	"true", "false"
`providerConfig.s3.secure`	Habilita la comunicación HTTPS segura con el endpoint S3.	verdadero	"true", "false"
`providerConfig.s3.proxyURL`	Especifica la URL del servidor proxy usado para conectarte a S3.	Ninguno	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Proporciona un certificado CA raíz personalizado para la verificación SSL/TLS.	Ninguno	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	Habilita la autenticación IAM para acceder a los buckets S3. Aplicable para EKS Pod Identity.	false	verdadero, falso

Ver información de AppVault

Puedes usar el complemento CLI de Trident Protect para ver información sobre los objetos AppVault que has creado en el clúster.

Pasos

1. Ver el contenido de un AppVault objeto:

   ```
   tridentctl-protect get appvaultcontent gcp-vault \
   --show-resources all \
   -n trident-protect
   ```

   Ejemplo de salida:

```
+-------------+-------+----------+-----------------------------
+--------------------------+
|   CLUSTER   |  APP  |   TYPE   |            NAME             |
TIMESTAMP         |
+-------------+-------+----------+-----------------------------
+--------------------------+
|             | mysql | snapshot | mysnap                      | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup   | mybackup5                   | 2024-
08-09 22:25:13 (UTC) |
|             | mysql | backup   | mybackup                    | 2024-
08-09 21:02:52 (UTC) |
+-------------+-------+----------+-----------------------------
+--------------------------+
```

2. Opcionalmente, para ver el AppVaultPath para cada recurso, usa la bandera `--show-paths`.

   El nombre del clúster en la primera columna de la tabla solo está disponible si se especificó un nombre de clúster en la instalación de Trident Protect helm. Por ejemplo: `--set clusterName=production1`.

Eliminar un AppVault

Puedes eliminar un objeto AppVault en cualquier momento.

> No elimines la clave `finalizers` en el CR AppVault antes de borrar el objeto AppVault. Si lo haces, pueden quedar datos residuales en el bucket AppVault y recursos huérfanos en el clúster.

Antes de empezar

Asegúrate de que has eliminado todos los CR de instantáneas y copias de seguridad que está usando el AppVault que quieres eliminar.

Eliminar un AppVault usando la CLI de Kubernetes

1. Elimina el objeto AppVault, reemplazando `appvault-name` por el nombre del objeto AppVault que quieres eliminar:

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Elimina un AppVault usando la CLI de Trident Protect

1. Elimina el objeto AppVault, reemplazando `appvault-name` por el nombre del objeto AppVault que quieres eliminar:

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Define una aplicación para gestión con Trident Protect

Puedes definir una aplicación que quieras gestionar con Trident Protect creando una CR de aplicación y una CR asociada AppVault.

Crea un CR de AppVault

Necesitas crear un CR de AppVault que se usará cuando realices operaciones de protección de datos en la aplicación, y el CR de AppVault debe estar en el clúster donde está instalado Trident Protect. El CR de AppVault es específico para tu entorno; para ejemplos de CR de AppVault, consulta "AppVault recursos personalizados."

Define una aplicación

Necesitas definir cada aplicación que quieras gestionar con Trident Protect. Puedes definir una aplicación para gestionarla creando manualmente un CR de aplicación o usando la CLI de Trident Protect.

> Para definir una aplicación con ámbito de máquina virtual (VM) que proteja máquinas virtuales (VM) específicas de KubeVirt por su nombre en lugar de espacios de nombres completos, usa el campo `spec.includedVirtualMachines` en el Application CR o la bandera `--virtual-machines` en la CLI. Para detalles, referencia de campos y ejemplos en YAML, consulta "Protege las máquinas virtuales de KubeVirt con Trident Protect".

Agrega una aplicación usando un CR

Pasos

1. Crea el archivo CR de la aplicación de destino:
   a. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `maria-app.yaml`).
   b. Configura los siguientes atributos:

- **metadata.name**: (*Obligatorio*) El nombre del recurso personalizado de la aplicación. Ten en cuenta el nombre que elijas porque otros archivos CR necesarios para las operaciones de protección hacen referencia a este valor.
- **spec.includedNamespaces**: (*Requerido*) Usa el selector de espacio de nombres y de etiqueta para especificar los espacios de nombres y recursos que utiliza la aplicación. El espacio de nombres de la aplicación debe formar parte de esta lista. El selector de etiqueta es opcional y puedes usarlo para filtrar recursos dentro de cada espacio de nombres especificado.
- **spec.includedClusterScopedResources**: (*Opcional*) Usa este atributo para especificar los recursos de ámbito clúster que se incluirán en la definición de la aplicación. Este atributo te permite seleccionar estos recursos según su grupo, versión, tipo y etiquetas.
  - **groupVersionKind**: (*Obligatorio*) especifica el grupo de API, la versión y el tipo del recurso con alcance de clúster.
  - **labelSelector**: (*Opcional*) Filtra los recursos con ámbito de clúster según sus etiquetas.
- **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Opcional*) Esta anotación solo es aplicable a aplicaciones definidas a partir de máquinas virtuales, como en entornos KubeVirt, donde se producen congelaciones del sistema de archivos antes de las instantáneas. Especifica si esta aplicación puede escribir en el sistema de archivos durante una instantánea. Si se establece en true, la aplicación ignora la configuración global y puede escribir en el sistema de archivos durante una instantánea. Si se establece en false, la aplicación ignora la configuración global y el sistema de archivos se congela durante una instantánea. Si se especifica pero la aplicación no tiene máquinas virtuales en la definición de la aplicación, se ignora la anotación. Si no se especifica, la aplicación sigue la "ajuste global de congelación de Trident Protect".

Si necesitas aplicar esta anotación después de que ya se haya creado una aplicación, puedes usar el siguiente comando:

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

+
Ejemplo de YAML:

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  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
```

1. (*Opcional*) Si lo necesitas, puedes añadir filtrado de recursos al mismo CR para incluir o excluir recursos específicos:

   - **Ejemplo de filtro genérico:**

     - **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `Include` o `Exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros de resourceMatchers para definir los recursos que se incluirán o excluirán:

       - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines varios elementos en este array, se combinan como una operación OR, y los campos dentro de cada elemento (group, kind, version) se combinan como una operación AND.
       - **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.
       - **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.
       - **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
       - **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
       - **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

       > Cuando se utilizan tanto `resourceFilter` como `labelSelector`, `resourceFilter` se ejecuta primero y luego `labelSelector` se aplica a los recursos resultantes.

     Por ejemplo:

```
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"]
```

- **Ejemplo de filtro solo de PVC:**

  Para definir una aplicación sólo para PVC, también debes incluir `PersistentVolume` y `VolumeSnapshotClass` en el filtro de recursos. Las operaciones de instantáneas y copias de seguridad dependen de `PersistentVolume` (el volumen de ámbito de clúster vinculado a cada PVC) y `VolumeSnapshotClass` (el controlador de instantáneas), y fallarán si no se incluyen. Por ejemplo:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-pvc-app
  namespace: my-app-namespace
spec:
  includedNamespaces:
  - namespace: my-app-namespace
  resourceFilter:
    resourceMatchers:
    - kind: PersistentVolumeClaim
      version: v1
    - kind: PersistentVolume
      version: v1
    - kind: VolumeSnapshotClass
      version: v1
    resourceSelectionCriteria: Include
```

2. Después de crear la CR de la aplicación para que coincida con tu entorno, aplica la CR. Por ejemplo:

```
kubectl apply -f maria-app.yaml
```

Pasos

1. Crea y aplica la definición de la aplicación usando uno de los siguientes ejemplos, reemplazando los valores entre corchetes por información de tu entorno. Puedes incluir espacios de nombres y recursos en la definición de la aplicación usando listas separadas por comas con los argumentos que se muestran en los ejemplos.

Puedes usar opcionalmente una anotación cuando creas una app para especificar si la aplicación puede escribir en el sistema de archivos durante una snapshot. Esto solo es aplicable a aplicaciones definidas desde máquinas virtuales, como en entornos KubeVirt, donde los congelamientos del sistema de archivos ocurren antes de las snapshots. Si estableces la anotación en `true`, la aplicación ignora la configuración global y puede escribir en el sistema de archivos durante una snapshot. Si la estableces en `false`, la aplicación ignora la configuración global y el sistema de archivos se congela durante una snapshot. Si usas la anotación pero la aplicación no tiene máquinas virtuales en la definición de la app, la anotación se ignora. Si no usas la anotación, la aplicación sigue la "ajuste global de congelación de Trident Protect".

+ Para especificar la anotación cuando usas la CLI para crear una aplicación, puedes usar la bandera `--annotation`.

+ * Crea la aplicación y utiliza la configuración global para el comportamiento de congelación del sistema de archivos:

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace>
```

+ * Crea la aplicación y configura la configuración local de la aplicación para el comportamiento de congelación del sistema de archivos:

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace> --annotation
protect.trident.netapp.io/skip-vm-freeze=<"true"|"false">
```

+ * Puedes usar `--resource-filter-include` y `--resource-filter-exclude` para incluir o excluir recursos según `resourceSelectionCriteria` como grupo, tipo, versión, etiquetas, nombres y namespaces, como se muestra en el siguiente ejemplo:

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
```

```
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace> --resource-filter-include
'[{"Group":"apps","Kind":"Deployment","Version":"v1","Names":["my-
deployment"],"Namespaces":["my-namespace"],"LabelSelectors":["app=my-
app"]}]'
```

+ * Para definir una aplicación que utilice únicamente PVC, también debes incluir `PersistentVolume` y `VolumeSnapshotClass` en el filtro de recursos. Las operaciones de instantáneas y copias de seguridad dependen de `PersistentVolume` (el volumen de ámbito de clúster vinculado a cada PVC) y `VolumeSnapshotClass` (el controlador de instantáneas), y fallarán si no se incluyen. Por ejemplo:

+

```
tridentctl-protect create app my-pvc-app --namespaces <my-app-
namespace> --resource-filter-include
'[{"Kind":"PersistentVolumeClaim","Version":"v1"},{"Kind":"PersistentVo
lume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Version":"v1"}]'
-n <my-app-namespace>
```

Protege las aplicaciones usando Trident Protect

Puedes proteger todas las apps gestionadas por Trident Protect tomando instantáneas y backups usando una política de protección automatizada o de forma ad hoc.

Puedes configurar Trident Protect para congelar y descongelar sistemas de archivos durante las operaciones de protección de datos. "Conoce más sobre cómo configurar la congelación del sistema de archivos con Trident Protect".

Crea una instantánea a pedido

Puedes crear una instantánea a pedido en cualquier momento.

Los recursos con ámbito de clúster se incluyen en un backup, snapshot o clon si se hace referencia a ellos explícitamente en la definición de la aplicación o si tienen referencias a cualquiera de los espacios de nombres de la aplicación.

Crear una instantánea usando un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-snapshot-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.

- **spec.applicationRef**: el nombre de Kubernetes de la aplicación para hacer snapshot.
- **spec.appVaultRef**: (*Required*) El nombre de AppVault donde se debe almacenar el contenido de la instantánea (metadatos).
- **spec.reclaimPolicy**: (*opcional*) Define lo que pasa con la AppArchive de una instantánea cuando se elimina el CR de la instantánea. Esto significa que incluso cuando se establece en `Retain`, la instantánea se eliminará. Opciones válidas:
  - `Retain` (predeterminado)
  - `Delete`

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

3. Después de que completes el archivo `trident-protect-snapshot-cr.yaml` con los valores correctos, aplica el CR:

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

Crea una instantánea usando la CLI

Pasos

1. Crea la instantánea, reemplazando los valores entre corchetes con información de tu entorno. Por ejemplo:

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> -n
<application_namespace>
```

Crear un backup a pedido

Puedes hacer un backup de una app en cualquier momento.

> Los recursos con ámbito de clúster se incluyen en un backup, snapshot o clon si se hace referencia a ellos explícitamente en la definición de la aplicación o si tienen referencias a cualquiera de los espacios de nombres de la aplicación.

Antes de empezar

Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de

backup s3 de larga duración. Si el token caduca durante la operación de backup, la operación puede fallar.

- Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
- Consulta "Documentación de AWS IAM" para más información sobre las credenciales con los recursos de AWS.

Crea un backup usando una CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-backup-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.applicationRef**: (*Required*) el nombre de Kubernetes de la aplicación que quieres respaldar.
   - **spec.appVaultRef**: (*Obligatorio*) El nombre del AppVault donde se debe almacenar el contenido de la copia de seguridad.
   - **spec.dataMover**: (*Opcional*) Una cadena que indica qué herramienta de backup se usará para la operación de backup. Valores posibles (distingue mayúsculas de minúsculas):
     - `Restic`
     - `Kopia` (predeterminado)
   - **spec.reclaimPolicy**: (*Opcional*) Define qué sucede con una copia de seguridad al liberarla de su reclamación. Valores posibles:
     - `Delete`
     - `Retain` (predeterminado)
   - **spec.snapshotRef**: (*Opcional*): Nombre de la instantánea que se usará como origen del backup. Si no se proporciona, se creará una instantánea temporal y se hará un backup.

     Ejemplo de YAML:

```yaml
---
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: Kopia
```

3. Después de que completes el archivo `trident-protect-backup-cr.yaml` con los valores

correctos, aplica el CR:

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

Crea un backup usando la CLI

Pasos

1. Crea el backup, reemplazando los valores entre corchetes con información de tu entorno. Por ejemplo:

```
tridentctl-protect create backup <my_backup_name> --appvault <my-
vault-name> --app <name_of_app_to_back_up> --data-mover
<Kopia_or_Restic> -n <application_namespace>
```

   Opcionalmente, puedes usar la `--full-backup` flag para especificar si un backup debe ser no incremental. Por defecto, todos los backups son incrementales. Cuando usas esta flag, el backup se vuelve no incremental. Lo mejor es hacer un backup completo de vez en cuando y luego backups incrementales entre los backups completos para minimizar el riesgo asociado con las restauraciones.

Anotaciones de backup compatibles

La siguiente tabla describe las anotaciones que puedes usar al crear una CR de backup:

Anotación	Tipo	Descripción	Valor predeterminado
protect.trident.netapp.io/backup-completo	cadena	Especifica si una copia de seguridad debe ser no incremental. Establece en `true` para crear una copia de seguridad no incremental. Lo mejor es hacer un backup completo de vez en cuando y luego backups incrementales entre los backups completos para minimizar el riesgo asociado con las restauraciones.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	cadena	El tiempo máximo permitido para que se complete toda la operación de instantánea.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	cadena	El tiempo máximo permitido para que las instantáneas de volumen alcancen el estado listo para usar.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	cadena	El tiempo máximo permitido para que se creen instantáneas de volumen.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadena	Tiempo máximo (en segundos) de espera para que cualquier PersistentVolumeClaims (PVCs) recién creado alcance la `Bound` fase antes de que la operación falle.	"1200" (20 minutos)

Crea un cronograma de protección de datos

Una política de protección protege una app creando snapshots, backups o ambos en una programación definida. Puedes elegir crear snapshots y backups cada hora, día, semana y mes, y puedes especificar la cantidad de copias que quieres conservar. Puedes programar un backup completo no incremental usando la anotación full-backup-rule. Por defecto, todos los backups son incrementales. Realizar un backup completo periódicamente, junto con backups incrementales entre medio, ayuda a reducir el riesgo asociado con las restauraciones.

- Puedes crear programaciones solo para instantáneas configurando `backupRetention` en cero y `snapshotRetention` en un valor mayor que cero. Configurar `snapshotRetention` en cero significa que cualquier backup programado seguirá creando instantáneas, pero estas son temporales y se eliminan inmediatamente después de que el backup se complete.
- Los recursos con ámbito de clúster se incluyen en un backup, snapshot o clon si se hace referencia a ellos explícitamente en la definición de la aplicación o si tienen referencias a cualquiera de los espacios de nombres de la aplicación.

Crea un cronograma usando un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-schedule-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.dataMover**: (*Opcional*) Una cadena que indica qué herramienta de backup se usará para la operación de backup. Valores posibles (distingue mayúsculas de minúsculas):
     - `Restic`
     - `Kopia` (predeterminado)
   - **spec.applicationRef**: el nombre de Kubernetes de la aplicación que se va a respaldar.
   - **spec.appVaultRef**: (*Obligatorio*) El nombre del AppVault donde se debe almacenar el contenido de la copia de seguridad.
   - **spec.backupRetention**: (*Obligatorio*) El número de backups que se conservarán. Cero indica que no se deben crear backups (solo instantáneas).
   - **spec.backupReclaimPolicy**: (*Opcional*) Determina qué sucede con un backup si el backup CR se elimina durante su periodo de retención. Después del periodo de retención, los backups siempre se eliminan. Valores posibles (distingue mayúsculas de minúsculas):
     - `Retain` (predeterminado)
     - `Delete`
   - **spec.snapshotRetention**: (*Obligatorio*) El número de instantáneas que se conservarán. Cero indica que no se deben crear instantáneas.
   - **spec.snapshotReclaimPolicy**: (*Opcional*) Determina qué sucede con una instantánea si el CR de la instantánea se elimina durante su periodo de retención. Después del periodo de retención, las instantáneas siempre se eliminan. Valores posibles (distingue mayúsculas de minúsculas):

- `Retain`
- `Delete` (predeterminado)

- **spec.granularity**: la frecuencia con la que debe ejecutarse la programación. Valores posibles, junto con los campos asociados obligatorios:
  - `Hourly` (requiere que especifiques `spec.minute`)
  - `Daily` (requiere que especifiques `spec.minute` y `spec.hour`)
  - `Weekly` (requiere que especifiques `spec.minute, spec.hour`, y `spec.dayOfWeek`)
  - `Monthly` (requiere que especifiques `spec.minute, spec.hour`, y `spec.dayOfMonth`)
  - `Custom`
- **spec.dayOfMonth**: (*Opcional*) El día del mes (1 - 31) en que debe ejecutarse la programación. Este campo es obligatorio si la granularidad está establecida en `Monthly`. El valor debe proporcionarse como una cadena.
- **spec.dayOfWeek**: (*Opcional*) El día de la semana (0 - 7) en que debe ejecutarse la programación. Los valores 0 o 7 indican domingo. Este campo es obligatorio si la granularidad está establecida en `Weekly`. El valor debe proporcionarse como una cadena.
- **spec.hour**: (*Opcional*) La hora del día (0 - 23) en que debe ejecutarse la programación. Este campo es obligatorio si la granularidad se establece en `Daily`, `Weekly` o `Monthly`. El valor debe proporcionarse como una cadena.
- **spec.minute**: (*Opcional*) El minuto de la hora (0 - 59) en que debe ejecutarse la programación. Este campo es obligatorio si la granularidad se establece en `Hourly`, `Daily`, `Weekly` o `Monthly`. El valor debe proporcionarse como una cadena.
- **spec.runImmediately**: (*Opcional*) Establece en `true` para activar una ejecución de línea de base única e inmediata (copia de seguridad y/o instantánea según la configuración de retención) cuando se crea el programa. Por defecto es `false`. Esto no modifica la recurrencia posterior.

  Ejemplo de YAML para programación de backup y snapshots:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"
```

Ejemplo de YAML para la programación solo de instantáneas:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

Ejemplo YAML para programación con ejecución inmediata:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-daily-schedule-run-immediately
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "7"
  snapshotRetention: "7"
  granularity: Daily
  hour: "3"
  minute: "0"
  runImmediately: true
```

3. Después de que completes el archivo `trident-protect-schedule-cr.yaml` con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

Crea un horario usando la CLI

Pasos

1. Crea el programa de protección, sustituyendo los valores entre corchetes por información de tu entorno. Por ejemplo:

Puedes usar `tridentctl-protect create schedule --help` para ver información de ayuda detallada para este comando.

```
tridentctl-protect create schedule <my_schedule_name> \
  --appvault <my_appvault_name> \
  --app <name_of_app_to_snapshot> \
  --backup-retention <how_many_backups_to_retain> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_to_run_schedule> \
  --granularity <frequency_to_run> \
  --hour <hour_of_day_to_run> \
  --minute <minute_of_hour_to_run> \
  --recurrence-rule <recurrence> \
  --snapshot-retention <how_many_snapshots_to_retain> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

Las siguientes banderas te dan un control adicional sobre tu horario:

- **Programación de backup completo**: Usa la bandera `--full-backup-rule` para programar backups completos no incrementales. Esta bandera solo funciona con `--granularity Daily`. Valores posibles:
  - `Always`: crea un backup completo cada día.
  - Días específicos de la semana: especifica uno o varios días separados por comas (por ejemplo, `"Monday,Thursday"`). Valores válidos: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

    La `--full-backup-rule` flag no funciona con granularidad horaria, semanal o mensual.

- **Protección inicial inmediata**: Usa `--run-immediately true` para crear una copia de seguridad o instantánea inicial inmediatamente cuando se crea la programación, en lugar de esperar a la primera hora de ejecución programada. El valor predeterminado es `false`.
- **Planificaciones de solo instantáneas**: establece `--backup-retention 0` y especifica un valor mayor que cero para `--snapshot-retention`.

Anotaciones de horario compatibles

En la tabla siguiente se describen las anotaciones que puedes usar al crear un CR de planificación:

Anotación	Tipo	Descripción	Valor predeterminado
protect.trident.netapp.io/full-backup-rule	cadena	Especifica la regla para programar backups completos. Puedes configurarla en `Always` para un backup completo constante o personalizarla según tus necesidades. Por ejemplo, si eliges granularidad diaria, puedes especificar los días de la semana en los que debe hacerse el backup completo (por ejemplo, `"Monday,Thursday"`). Los valores válidos de días de la semana son: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. Ten en cuenta que esta anotación solo se puede usar con programaciones que tengan `granularity` configurado en `Daily`.	No configurado (todos los backups son incrementales)
protect.trident.netapp.io/snapshot-completion-timeout	cadena	El tiempo máximo permitido para que se complete toda la operación de instantánea.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	cadena	El tiempo máximo permitido para que las instantáneas de volumen alcancen el estado listo para usar.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	cadena	El tiempo máximo permitido para que se creen instantáneas de volumen.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadena	Tiempo máximo (en segundos) de espera para que cualquier PersistentVolumeClaims (PVCs) recién creado alcance la `Bound` fase antes de que la operación falle.	"1200" (20 minutos)

Borrar una instantánea

Elimina las instantáneas programadas o bajo demanda que ya no necesitas.

Pasos

1. Elimina el snapshot CR asociado con el snapshot:

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

Eliminar un backup

Elimina los backups programados o bajo demanda que ya no necesites.

Asegúrate de que la política de recuperación esté configurada en `Delete` para eliminar todos los datos de backup del almacenamiento de objetos. La configuración predeterminada de la política es `Retain` para evitar la pérdida accidental de datos. Si la política no se cambia a `Delete`, los datos de backup permanecerán en el almacenamiento de objetos y tendrás que eliminarlos manualmente.

Pasos

1. Elimina el backup CR asociado al backup:

```
kubectl delete backup <backup_name> -n my-app-namespace
```

Comprobar el estado de una operación de backup

Puedes usar la línea de comandos para comprobar el estado de un proceso de backup que está en curso, ha terminado o ha fallado.

Pasos

1. Usa el siguiente comando para recuperar el estado de la operación de backup, sustituyendo los valores entre corchetes por la información de tu entorno:

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Habilita el backup y la restauración para las operaciones de azure-netapp-files (ANF)

Si has instalado Trident Protect, puedes activar la funcionalidad de backup y restauración con gestión eficiente del espacio para backends de almacenamiento que usan la clase de almacenamiento azure-netapp-files y que fueron creados antes de Trident 24.06. Esta funcionalidad funciona con volúmenes NFSv4 y no consume espacio adicional del capacity pool.

Antes de empezar

Asegúrate de lo siguiente:

- Has instalado Trident Protect.
- Has definido una aplicación en Trident Protect. Esta aplicación tendrá una funcionalidad de protección limitada hasta que completes este procedimiento.
- Has `azure-netapp-files` seleccionado como la clase de almacenamiento por defecto para tu backend de almacenamiento.

▼ Expandir para pasos de configuración

1. Haz lo siguiente en Trident si el volumen ANF se creó antes de actualizar a Trident 24.10:

   a. Habilita el directorio de instantáneas para cada PV que esté basado en azure-netapp-files y asociado con la aplicación:

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. Confirma que se ha habilitado el directorio de instantáneas para cada PV asociado:

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   ```

```
snapshotDir
```

Respuesta:

```
snapshotDirectory: "true"
```

+
Cuando el directorio de instantáneas no está habilitado, Trident Protect elige la funcionalidad de backup normal, que consume temporalmente espacio en el pool de capacidad durante el proceso de backup. En este caso, asegúrate de que hay suficiente espacio disponible en el pool de capacidad para crear un volumen temporal del tamaño del volumen que se está respaldando.

Resultado

La aplicación está lista para backup y restauración usando Trident Protect. Cada PVC también está disponible para que otras aplicaciones lo usen en backups y restauraciones.

Restaurar aplicaciones

Restaura aplicaciones usando Trident Protect

Puedes usar Trident Protect para restaurar tu aplicación desde una instantánea o backup. Restaurar desde una instantánea existente será más rápido cuando restaures la aplicación en el mismo clúster.

- Cuando restauras una aplicación, todos los ganchos de ejecución configurados para la aplicación se restauran junto con la app. Si hay un gancho de ejecución posterior a la restauración, se ejecuta automáticamente como parte de la operación de restauración.
- La restauración desde una copia de seguridad a un espacio de nombres diferente o al espacio de nombres original es compatible para los volúmenes qtree. Sin embargo, restaurar desde una instantánea a un espacio de nombres diferente o al espacio de nombres original no es compatible para los volúmenes qtree.
- Puedes usar la configuración avanzada para personalizar las operaciones de restauración. Para obtener más información, consulta "Usa la configuración avanzada de restauración de Trident Protect".
- Para restaurar máquinas virtuales (VM) específicas de KubeVirt desde un archivo sin restaurar todos los recursos del espacio de nombres, usa el campo `includedVirtualMachines` en un CR de restauración. Para CR compatibles, referencia de campos y ejemplos en YAML, consulta "Protege las máquinas virtuales de KubeVirt con Trident Protect".

Restaura desde una copia de seguridad a un espacio de nombres diferente

Cuando restauras una copia de seguridad en un espacio de nombres diferente usando un CR de BackupRestore, Trident Protect restaura la aplicación en un nuevo espacio de nombres y crea un CR de aplicación para la aplicación restaurada. Para proteger la aplicación restaurada, crea copias de seguridad o instantáneas bajo demanda, o establece un programa de protección.

- Restaurar una copia de seguridad en un espacio de nombres diferente con recursos

existentes no alterará ningún recurso que comparta nombres con los de la copia de seguridad. Para restaurar todos los recursos de la copia de seguridad, elimina y vuelve a crear el espacio de nombres de destino o restaura la copia de seguridad en un nuevo espacio de nombres.

- Cuando usas un CR para restaurar a un nuevo espacio de nombres, debes crear manualmente el espacio de nombres de destino antes de aplicar el CR. Trident Protect crea espacios de nombres automáticamente solo cuando usas la CLI.

Antes de empezar

Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de restauración s3 de larga duración. Si el token caduca durante la operación de restauración, la operación puede fallar.

- Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
- Consulta "Documentación de AWS IAM" para más información sobre las credenciales con los recursos de AWS.

Cuando restauras copias de seguridad usando Kopia como el transportador de datos, puedes especificar opcionalmente anotaciones en el CR o usando la CLI para controlar el comportamiento del almacenamiento temporal que usa Kopia. Consulta la "Documentación de Kopia" para más información sobre las opciones que puedes configurar. Usa el comando `tridentctl-protect create --help` para más información sobre cómo especificar anotaciones con la Trident Protect CLI.

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-backup-restore-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la copia de seguridad. Puedes usar el siguiente comando para encontrar esta ruta:

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

   - **spec.appVaultRef**: (*Required*) el nombre del AppVault donde se almacena el contenido de la copia de seguridad.
   - **spec.destinationApplicationName**: (*opcional*) el nombre para la aplicación restaurada. Si se proporciona, la aplicación restaurada usa este nombre. Si no se proporciona, la aplicación restaurada usa el nombre de la aplicación de origen.
   - **spec.namespaceMapping**: la asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplaza `my-source-namespace` y `my-destination-namespace` con información de tu entorno.

```
---
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
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace",
"destination": "my-destination-namespace"}]
```

3. (*Opcional*) Si necesitas seleccionar solo determinados recursos de la aplicación para restaurarlos, añade un filtro que incluya o excluya los recursos marcados con etiquetas concretas:

> Trident Protect selecciona algunos recursos automáticamente debido a su relación con los recursos que tú seleccionas. Por ejemplo, si seleccionas un recurso de reclamación de volumen persistente y este tiene un pod asociado, Trident Protect también restaurará el pod asociado.

- **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `Include` o `Exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros de resourceMatchers para definir los recursos que se incluirán o excluirán:
  - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines varios elementos en este array, se combinan como una operación OR, y los campos dentro de cada elemento (group, kind, version) se combinan como una operación AND.
    - **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
    - **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.
    - **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.
    - **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
    - **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
    - **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

      Por ejemplo:

```
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"]
```

4. Después de que completes el archivo `trident-protect-backup-restore-cr.yaml` con los valores correctos, aplica el CR:

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

Usa la CLI

Pasos

1. Restaura la copia de seguridad en un espacio de nombres diferente, reemplazando los valores entre corchetes por información de tu entorno. El argumento `namespace-mapping` usa espacios de nombres separados por dos puntos para mapear los espacios de nombres de origen a los espacios de nombres de destino correctos en el formato `source1:dest1,source2:dest2`. Por ejemplo:

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--destination-app-name<custom_app_name>\
-n <application_namespace>
```

Restaura desde una copia de seguridad al espacio de nombres original

Puedes restaurar una copia de seguridad en el espacio de nombres original en cualquier momento. Cuando realizas una restaurar sin movimiento, Trident Protect gestiona automáticamente los programas de protección y las operaciones en curso para evitar puntos de recuperación no válidos:

- Todos los programas de protección activados para la aplicación se desactivan antes de que comience la restauración. Esto evita que se ejecuten copias de seguridad o instantáneas programadas mientras se restauran los recursos de la aplicación.
- Después de que la restauración se complete correctamente, solo se vuelven a habilitar las programaciones que estaban habilitadas antes de la restauración. Las programaciones que ya estaban deshabilitadas permanecen deshabilitadas.
- Cualquier operación de copia de seguridad o instantánea en curso se cancela antes de que comience la restauración. Si una operación no se cancela en 5 minutos, la restauración continúa y registra una advertencia en el estado de restauración CR.

Antes de empezar

Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de restauración s3 de larga duración. Si el token caduca durante la operación de restauración, la operación puede fallar.

- Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
- Consulta "Documentación de AWS IAM" para más información sobre las credenciales con los recursos de AWS.

> Cuando restauras copias de seguridad usando Kopia como el transportador de datos, puedes especificar opcionalmente anotaciones en el CR o usando la CLI para controlar el comportamiento del almacenamiento temporal que usa Kopia. Consulta la "Documentación de Kopia" para más información sobre las opciones que puedes configurar. Usa el comando `tridentctl-protect create --help` para más información sobre cómo especificar anotaciones con la Trident Protect CLI.

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-backup-ipr-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la copia de seguridad. Puedes usar el siguiente comando para encontrar esta ruta:

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

   - **spec.appVaultRef**: (*Required*) el nombre del AppVault donde se almacena el contenido de la copia de seguridad.

     Por ejemplo:

```
---
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
```

3. (*Opcional*) Si necesitas seleccionar solo determinados recursos de la aplicación para restaurarlos, añade un filtro que incluya o excluya los recursos marcados con etiquetas concretas:

   > Trident Protect selecciona algunos recursos automáticamente debido a su relación con los recursos que tú seleccionas. Por ejemplo, si seleccionas un recurso de reclamación de volumen persistente y este tiene un pod asociado, Trident Protect también restaurará el pod asociado.

   - **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `Include` o `Exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros de resourceMatchers para definir los recursos que se incluirán o excluirán:
     - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines varios elementos en este array, se combinan como una operación OR, y los campos dentro de cada elemento (group, kind, version) se combinan como una operación AND.
       - **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.
       - **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.
       - **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
       - **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
       - **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

         Por ejemplo:

```
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"]
```

4. Después de que completes el archivo `trident-protect-backup-ipr-cr.yaml` con los valores correctos, aplica el CR:

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

Usa la CLI

Pasos

1. Restaura la copia de seguridad en el espacio de nombres original, sustituyendo los valores entre corchetes por información de tu entorno. El argumento `backup` utiliza un espacio de nombres y un nombre de copia de seguridad en el formato `<namespace>/<name>`. Por ejemplo:

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

Restaurar desde una copia de seguridad en un clúster diferente

Puedes restaurar una copia de seguridad en un clúster diferente si hay algún problema con el clúster original.

- Cuando restauras copias de seguridad usando Kopia como el transportador de datos, puedes especificar opcionalmente anotaciones en el CR o usando la CLI para controlar el comportamiento del almacenamiento temporal que usa Kopia. Consulta la "Documentación de Kopia" para más información sobre las opciones que puedes configurar. Usa el comando `tridentctl-protect create --help` para más información sobre cómo especificar anotaciones con la Trident Protect CLI.
- Cuando usas un CR para restaurar a un nuevo espacio de nombres, debes crear manualmente el espacio de nombres de destino antes de aplicar el CR. Trident Protect crea espacios de nombres automáticamente solo cuando usas la CLI.

Antes de empezar

Asegúrate de que se cumplen los siguientes requisitos previos:

- El clúster de destino tiene Trident Protect instalado.
- El clúster de destino tiene acceso a la ruta del bucket de la misma AppVault que el clúster de origen, donde se almacena la copia de seguridad.
- Asegúrate de que tu entorno local puede conectarse al bucket de almacenamiento de objetos definido en el AppVault CR cuando ejecutes el comando `tridentctl-protect get appvaultcontent`. Si las restricciones de red impiden el acceso, ejecuta la CLI de Trident Protect desde un pod en el clúster de destino en su lugar.
- Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de restauración de larga duración. Si el token caduca durante la operación de restauración, la operación puede fallar.
  - Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
  - Consulta "Documentación de AWS" para más información sobre las credenciales con los recursos de AWS.

Pasos

1. Verifica que el AppVault CR existe en el clúster de destino usando el complemento CLI de Trident Protect:

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

Si el AppVault CR no existe en el clúster de destino, créalo siguiendo los pasos en "Usa los objetos de Trident Protect AppVault para gestionar buckets".

2. Visualiza el contenido de la copia de seguridad disponible en AppVault en el clúster de destino y anota `appArchivePath` de la copia de seguridad que quieres restaurar:

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

Al ejecutar este comando se muestran las copias de seguridad disponibles en el AppVault, incluidos sus clústeres de origen, los nombres de las aplicaciones correspondientes, las marcas de tiempo y las rutas de archivo.

Ejemplo de salida:

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   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. Restaura la aplicación en el clúster de destino usando el nombre AppVault y la ruta de archivo:

Cuando utilices un CR, asegúrate de que el espacio de nombres previsto para la restauración de la aplicación existe en el clúster de destino.

Usa un CR

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-backup-restore-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:

- **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
- **spec.appVaultRef**: (*Required*) el nombre del AppVault donde se almacena el contenido de la copia de seguridad.
- **spec.appArchivePath**: (*Obligatorio*) La ruta dentro de AppVault donde se almacena el contenido de la copia de seguridad. Usa el comando del paso 2 para ver el contenido de la copia de seguridad y encontrar `appArchivePath` la copia de seguridad que quieres restaurar.
- **spec.destinationApplicationName**: (*opcional*) el nombre para la aplicación restaurada. Si se proporciona, la aplicación restaurada usa este nombre. Si no se proporciona, la aplicación restaurada usa el nombre de la aplicación de origen.
- **spec.namespaceMapping**: la asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplaza `my-source-namespace` y `my-destination-namespace` con información de tu entorno.

  Por ejemplo:

```
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
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. Después de que completes el archivo `trident-protect-backup-restore-cr.yaml` con los valores correctos, aplica el CR:

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

Usa la CLI

1. Usa el siguiente comando para restaurar la aplicación, reemplazando los valores entre corchetes por la información de tu entorno. El argumento namespace-mapping utiliza espacios de nombres separados por dos puntos para asignar los espacios de nombres de origen a los espacios de nombres de destino correctos en el formato source1:dest1,source2:dest2. Por ejemplo:

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--destination-app-name <custom_app_name> \
--context <destination_cluster_name> \
```

```
-n <application_namespace>
```

Restaurar desde una snapshot a un namespace diferente

Puedes restaurar datos de una instantánea usando un archivo de recurso personalizado (CR) ya sea en un espacio de nombres diferente o en el espacio de nombres de origen. Cuando restauras una instantánea en un espacio de nombres diferente usando un CR de SnapshotRestore, Trident Protect restaura la aplicación en un nuevo espacio de nombres y crea un CR de aplicación para la aplicación restaurada. Para proteger la aplicación restaurada, crea copias de seguridad o instantáneas bajo demanda, o establece un programa de protección.

- SnapshotRestore admite el atributo `spec.storageClassMapping`, pero solo cuando las clases de almacenamiento de origen y destino usan el mismo backend de almacenamiento. Si intentas restaurar en un `StorageClass` que usa un backend de almacenamiento diferente, la operación de restauración fallará.
- Cuando usas un CR para restaurar a un nuevo espacio de nombres, debes crear manualmente el espacio de nombres de destino antes de aplicar el CR. Trident Protect crea espacios de nombres automáticamente solo cuando usas la CLI.

Antes de empezar

Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de restauración s3 de larga duración. Si el token caduca durante la operación de restauración, la operación puede fallar.

- Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
- Consulta "Documentación de AWS IAM" para más información sobre las credenciales con los recursos de AWS.

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-snapshot-restore-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.appVaultRef**: (*Required*) El nombre de AppVault donde se almacenan los contenidos de la instantánea.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la instantánea. Puedes usar el siguiente comando para encontrar esta ruta:

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

   - **spec.destinationApplicationName**: (*opcional*) el nombre para la aplicación restaurada. Si se

proporciona, la aplicación restaurada usa este nombre. Si no se proporciona, la aplicación restaurada usa el nombre de la aplicación de origen.

- **spec.namespaceMapping**: la asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplaza `my-source-namespace` y `my-destination-namespace` con información de tu entorno.

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

3. (*Opcional*) Si necesitas seleccionar solo determinados recursos de la aplicación para restaurarlos, añade un filtro que incluya o excluya los recursos marcados con etiquetas concretas:

> Trident Protect selecciona algunos recursos automáticamente debido a su relación con los recursos que tú seleccionas. Por ejemplo, si seleccionas un recurso de reclamación de volumen persistente y este tiene un pod asociado, Trident Protect también restaurará el pod asociado.

- **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `Include` o `Exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros de resourceMatchers para definir los recursos que se incluirán o excluirán:
  - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines varios elementos en este array, se combinan como una operación OR, y los campos dentro de cada elemento (group, kind, version) se combinan como una operación AND.
    - **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
    - **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.
    - **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.
    - **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
    - **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
    - **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

      Por ejemplo:

```
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"]
```

4. Después de que completes el archivo `trident-protect-snapshot-restore-cr.yaml` con los valores correctos, aplica el CR:

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

Usa la CLI

Pasos

1. Restaura la instantánea en un espacio de nombres diferente, reemplazando los valores entre corchetes con información de tu entorno.

   - El `snapshot`argumento usa un espacio de nombres y un nombre de instantánea en el formato `<namespace>/<name>.
   - El argumento `namespace-mapping` usa espacios de nombres separados por dos puntos para mapear los espacios de nombres de origen a los espacios de nombres de destino correctos en el formato `source1:dest1,source2:dest2`.

     Por ejemplo:

```
tridentctl-protect create snapshotrestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--destination-app-name <custom_app_name> \
-n <application_namespace>
```

Restaurar desde una snapshot al namespace original

Puedes restaurar una instantánea en el espacio de nombres original en cualquier momento. Cuando realizas

una restaurar sin movimiento, Trident Protect gestiona automáticamente los programas de protección y las operaciones en curso para evitar puntos de recuperación no válidos:

- Todos los programas de protección activados para la aplicación se desactivan antes de que comience la restauración. Esto evita que se ejecuten copias de seguridad o instantáneas programadas mientras se restauran los recursos de la aplicación.
- Después de que la restauración se complete correctamente, solo se vuelven a habilitar las programaciones que estaban habilitadas antes de la restauración. Las programaciones que ya estaban deshabilitadas permanecen deshabilitadas.
- Cualquier operación de copia de seguridad o instantánea en curso se cancela antes de que comience la restauración. Si una operación no se cancela en 5 minutos, la restauración continúa y registra una advertencia en el estado de restauración CR.

Antes de empezar

Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de restauración s3 de larga duración. Si el token caduca durante la operación de restauración, la operación puede fallar.

- Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
- Consulta "Documentación de AWS IAM" para más información sobre las credenciales con los recursos de AWS.

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-snapshot-ipr-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.appVaultRef**: (*Required*) El nombre de AppVault donde se almacenan los contenidos de la instantánea.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la instantánea. Puedes usar el siguiente comando para encontrar esta ruta:

     ```
     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
```

+ . (*Opcional*) Si necesitas seleccionar solo determinados recursos de la aplicación para restaurarlos, añade un filtro que incluya o excluya los recursos marcados con etiquetas concretas:

+ NOTA: Trident Protect selecciona automáticamente algunos recursos debido a su relación con los recursos que tú seleccionas. Por ejemplo, si seleccionas un recurso de reclamación de volumen persistente y este tiene un pod asociado, Trident Protect también restaurará el pod asociado.

+ * **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Utiliza `Include` o `Exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir: **resourceFilter.resourceMatchers: Una matriz de objetos resourceMatcher. Si defines varios elementos en este array, se combinan como una operación OR, y los campos dentro de cada elemento (group, kind, version) se combinan como una operación AND.** * **resourceMatchers[].group**: (*Opcional*) Grupo del recurso que se va a filtrar. ***** **resourceMatchers[].kind: (*Opcional*) Tipo del recurso que se va a filtrar.** * **resourceMatchers[].version**: (*Opcional*) Versión del recurso que se va a filtrar. ***** **resourceMatchers[].names: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.** * **resourceMatchers[].namespaces**: (*Opcional*) Namespaces en el campo metadata.name de Kubernetes del recurso que se va a filtrar. *** resourceMatchers[].labelSelectors**: (*Opcional*) Cadena de selector de etiquetas en el campo metadata.name de Kubernetes del recurso, tal como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

+ Por ejemplo:

+

```
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"]
```

+ . Después de que completes el archivo `trident-protect-snapshot-ipr-cr.yaml` con los valores correctos, aplica el CR:

+

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

Usa la CLI

Pasos

1. Restaura la instantánea en el espacio de nombres original, reemplazando los valores entre corchetes por información de tu entorno. Por ejemplo:

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

Verifica el estado de una operación de restauración

Puedes usar la línea de comandos para comprobar el estado de una operación de restauración que está en curso, ha finalizado o ha fallado.

Pasos

1. Usa el siguiente comando para obtener el estado de la operación de restauración, reemplazando los valores entre corchetes por la información de tu entorno:

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

Usa la configuración avanzada de restauración de Trident Protect

Puedes personalizar las operaciones de restauración usando configuraciones avanzadas como anotaciones, configuración del espacio de nombres y opciones de almacenamiento para cumplir con tus requisitos específicos.

Anotaciones y etiquetas de namespace durante las operaciones de restauración y conmutación por error

Durante las operaciones de restauración y conmutación por error, las etiquetas y anotaciones en el espacio de nombres de destino se hacen coincidir con las etiquetas y anotaciones en el espacio de nombres de origen. Las etiquetas o anotaciones del espacio de nombres de origen que no existen en el espacio de nombres de destino se añaden, y cualquier etiqueta o anotación que ya exista se sobrescribe para que coincida con el valor del espacio de nombres de origen. Las etiquetas o anotaciones que existen solo en el espacio de nombres de destino permanecen sin cambios.

Si usas Red Hat OpenShift, es importante que notes el papel fundamental de las anotaciones de espacios de nombres en entornos OpenShift. Las anotaciones de espacios de nombres aseguran que los pods restaurados sigan los permisos y configuraciones de seguridad apropiados definidos por las restricciones de contexto de seguridad (SCC) de OpenShift y puedan acceder a los volúmenes sin problemas de permisos. Para obtener más información,

consulta "OpenShift documentación de restricciones de contexto de seguridad".

Puedes evitar que se sobrescriban anotaciones específicas en el espacio de nombres de destino configurando la variable de entorno de Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de realizar la restauración o la conmutación por error. Por ejemplo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

Al realizar una operación de restauración o conmutación por error, las anotaciones y etiquetas de espacios de nombres especificadas en `restoreSkipNamespaceAnnotations` y `restoreSkipNamespaceLabels` quedan excluidas de la operación de restauración o conmutación por error. Asegúrate de que estos ajustes se configuren durante la instalación inicial de Helm. Para obtener más información, consulta "Configura los ajustes adicionales del helm chart de Trident Protect".

Si instalaste la aplicación de origen usando Helm con la `--create-namespace` flag, se da un tratamiento especial a la clave de etiqueta `name`. Durante el proceso de restauración o conmutación por error, Trident Protect copia esta etiqueta al espacio de nombres de destino, pero actualiza el valor al valor del espacio de nombres de destino si el valor del origen coincide con el espacio de nombres de origen. Si este valor no coincide con el espacio de nombres de origen, se copia al espacio de nombres de destino sin cambios.

Ejemplo

El siguiente ejemplo presenta un espacio de nombres de origen y uno de destino, cada uno con anotaciones y etiquetas diferentes. Puedes ver el estado del espacio de nombres de destino antes y después de la operación, y cómo se combinan o sobrescriben las anotaciones y etiquetas en el espacio de nombres de destino.

Antes de la operación de restauración o conmutación por error

La siguiente tabla ilustra el estado de los espacios de nombres de origen y destino de ejemplo antes de la operación de restauración o conmutación por error:

Espacio de nombres	Anotaciones	Etiquetas
Namespace ns-1 (fuente)	• annotation.one/key: "valoractualizado" • annotation.two/key: "true"	• entorno=producción • compliance=hipaa • name=ns-1
Namespace ns-2 (destino)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

Después de la operación de restauración

La siguiente tabla ilustra el estado del espacio de nombres de destino de ejemplo después de la operación de restauración o conmutación por error. Se han añadido algunas claves, se han sobrescrito otras y la etiqueta `name` se ha actualizado para que coincida con el espacio de nombres de destino:

Espacio de nombres	Anotaciones	Etiquetas
Namespace ns-2 (destino)	• annotation.one/key: "valoractualizado" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • compliance=hipaa • entorno=producción • role=database

Campos compatibles

Esta sección describe los campos adicionales disponibles para las operaciones de restauración.

Asignación de clases de almacenamiento

El atributo `spec.storageClassMapping` define una asignación de una clase de almacenamiento presente en la aplicación de origen a una nueva clase de almacenamiento en el clúster de destino. Puedes usar esto cuando migres aplicaciones entre clústeres con diferentes clases de almacenamiento o cuando cambies el backend de almacenamiento para operaciones de BackupRestore.

Ejemplo:

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

Anotaciones compatibles

En esta sección se enumeran las anotaciones compatibles para configurar diversos comportamientos del sistema. Si el usuario no establece explícitamente una anotación, el sistema usará el valor predeterminado.

Anotación	Tipo	Descripción	Valor predeterminado
protect.trident.netapp.io/data-mover-timeout-sec	cadena	El tiempo máximo (en segundos) permitido para que la operación de movimiento de datos se quede en pausa.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	cadena	El límite de tamaño máximo (en megabytes) para la caché de contenido de Kopia.	"1000"

Anotación	Tipo	Descripción	Valor predeterminado
protect.trident.netapp.io/pvc-bind-timeout-sec	cadena	Tiempo máximo (en segundos) de espera para que cualquier PersistentVolumeClaims (PVCs) recién creado alcance la `Bound` fase antes de que la operación falle. Se aplica a todos los tipos de restauración de CR (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Usa un valor más alto si tu backend de almacenamiento o tu clúster suelen necesitar más tiempo.	"1200" (20 minutos)

Modifica los recursos a medida que los restauras usando recursos personalizados

Las transformaciones de recursos te permiten modificar un recurso mientras se está restaurando. Esto resulta útil cuando la versión restaurada necesita ser diferente de la original, por ejemplo, al actualizar un nombre de host en un ConfigMap al restaurar una aplicación en un clúster diferente.

Cómo funciona la modificación de recursos

El campo `transformations` en `SnapshotRestore`, `BackupRestore`, `AppMirrorRelationship` y otros recursos de restauración te permite modificar los recursos de Kubernetes durante el proceso de restauración. Esto es útil cuando la aplicación restaurada necesita configurarse de manera diferente a la original, por ejemplo, cambiando nombres de host, URLs de registros, límites de recursos o variables de entorno.

Las transformaciones de recursos usan operaciones "RFC 6902 JSON Patch" y rutas "RFC 6901 JSON Pointer" para seleccionar y modificar campos específicos dentro de los recursos de Kubernetes.

A continuación se muestra la estructura básica de un objeto de restaurar sin movimiento que contiene transformaciones de recursos:

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-restore
  namespace: target-namespace
spec:
  appVaultRef: my-vault
  appArchivePath: /path/to/snapshot
  namespaceMapping:
    - source: source-ns
      destination: target-ns
  transformations:
    - resource:
        kind: ConfigMap       # Required: resource kind
        group: ""             # Optional: API group (empty for core
resources)
```

```
    version: ""           # Optional: API version
    name: ""              # Optional: specific resource name
  operations:
    - op: replace         # Operation type
      path: "/data/key"   # JSON Pointer path
      value: "new-value"  # New value (for add/replace/test)
```

Recursos compatibles

Puedes usar transformaciones de recursos con recursos que cumplan los siguientes criterios:

- **kind** (obligatorio): el tipo de recurso de Kubernetes (por ejemplo, `ConfigMap`, `Deployment`, `Pod`)
- **grupo** (opcional): el grupo de la API (por ejemplo, `apps`, `route.openshift.io`) — omítelo para los recursos principales
- **versión** (opcional): la versión de la API (por ejemplo, `v1`, `v1beta1`)
- **nombre** (opcional): aplica solo a un recurso específico por su nombre

Actualmente no es posible modificar PersistentVolumeClaims ni los espacios de nombres.

Operaciones compatibles

Puedes utilizar las siguientes operaciones para modificar recursos: `add`, `copy`, `move`, `remove`, `replace`, y `test`.

Agrega un valor a un recurso

Usa la operación `add` para añadir un nuevo campo o valor en la ruta especificada. Puedes añadir datos a objetos o arrays.

Usa un CR

El siguiente ejemplo añade un selector de nodo a un recurso `Deployment`:

```
transformations:
  - resource:
      kind: Deployment
    operations:
      - op: add
        path: "/spec/template/spec/nodeSelector"
        value:
          "topology.kubernetes.io/zone": "us-east-1a"
          disktype: "ssd"
```

Usa la CLI

Usa el siguiente comando para ejecutar esta transformación usando la línea de comandos:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'apps,v1,Deployment:add{"path":"/spec/template/spec/nodeSelector","valu
e":{"topology.kubernetes.io/zone":"us-east-1a","disktype":"ssd"}}'
```

Copia un valor dentro de un recurso

Usa la operación `copy` para copiar un valor de una ruta a otra dentro del mismo recurso. El origen no se modifica.

Usa un CR

El siguiente ejemplo duplica una clave de datos para un objeto `ConfigMap`:

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: copy
        from: "/data/source-key"
        path: "/data/backup-key"
```

Usa la CLI

Usa el siguiente comando para ejecutar esta transformación usando la línea de comandos:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation ',v1,ConfigMap:copy{"from":"/data/source-
key","path":"/data/backup-key"}'
```

Mover un valor dentro de un recurso

Usa la operación `move` para mover un valor de una ruta a otra dentro del mismo recurso. El origen se elimina y el valor se coloca en el destino.

Usa un CR

El siguiente ejemplo cambia el nombre de una clave de datos para un objeto `ConfigMap`:

```
transformations:
```

```
  - resource:
      kind: ConfigMap
    operations:
      - op: move
        from: "/data/OLD_KEY"
        path: "/data/NEW_KEY"
```

Usa la CLI

Usa el siguiente comando para ejecutar esta transformación usando la línea de comandos:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:move{"from":"/data/OLD_KEY","path":"/data/NEW_KEY"}'
```

Eliminar un valor de un recurso

Usa la operación `remove` para eliminar un campo o valor en la ruta especificada.

Usa un CR

El siguiente ejemplo elimina una anotación de un recurso `ConfigMap`:

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: remove
        path: "/metadata/annotations/kubectl.kubernetes.io~1last-
applied-configuration"
```

En la ruta del ejemplo anterior, `~1` es la secuencia de escape del puntero JSON para `/`.

Usa la CLI

Usa el siguiente comando para ejecutar esta transformación usando la línea de comandos:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:remove{"path":"/metadata/annotations/kubectl.kubernetes.
```

```
io~1last-applied-configuration"}'
```

Sustituir un valor dentro de un recurso

Utiliza la operación `replace` para sustituir un valor existente dentro de un recurso en la ruta especificada. La ruta JSON ya debe existir.

Usa un CR

El siguiente ejemplo cambia el nombre de host de un objeto `Route`:

```
transformations:
  - resource:
      kind: Route
      group: route.openshift.io
    operations:
      - op: replace
        path: "/spec/host"
        value: "prod.example.com"
```

Usa la CLI

Usa el siguiente comando para ejecutar esta transformación usando la línea de comandos:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'route.openshift.io,v1,Route:replace{"path":"/spec/host","value":"prod.
example.com"}'
```

Probar la modificación de los recursos

Utiliza la operación `test` para comprobar que el valor de una ruta coincide con el valor esperado antes de que se apliquen las demás operaciones. Si la comprobación falla, se revierte toda la modificación.

Usa un CR

El siguiente ejemplo actualiza `database-host` solo si `environment` está establecido en `staging`:

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
```

```
      - op: test
        path: "/data/environment"
        value: "staging"
      - op: replace
        path: "/data/database-host"
        value: "prod-db.example.com"
```

Usa la CLI

Usa el siguiente comando para ejecutar esta transformación usando la línea de comandos:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:test{"path":"/data/environment","value":"staging"},repla
ce{"path":"/data/database-host","value":"prod-db.example.com"}'
```

Replica aplicaciones usando NetApp SnapMirror y Trident Protect

Con Trident Protect, puedes usar las capacidades de replicación asíncrona de NetApp SnapMirror para replicar datos y cambios de aplicaciones de un backend de almacenamiento a otro, en el mismo clúster o entre diferentes clústeres.

Anotaciones y etiquetas de namespace durante las operaciones de restauración y conmutación por error

Durante las operaciones de restauración y conmutación por error, las etiquetas y anotaciones en el espacio de nombres de destino se hacen coincidir con las etiquetas y anotaciones en el espacio de nombres de origen. Las etiquetas o anotaciones del espacio de nombres de origen que no existen en el espacio de nombres de destino se añaden, y cualquier etiqueta o anotación que ya exista se sobrescribe para que coincida con el valor del espacio de nombres de origen. Las etiquetas o anotaciones que existen solo en el espacio de nombres de destino permanecen sin cambios.

> Si usas Red Hat OpenShift, es importante que notes el papel fundamental de las anotaciones de espacios de nombres en entornos OpenShift. Las anotaciones de espacios de nombres aseguran que los pods restaurados sigan los permisos y configuraciones de seguridad apropiados definidos por las restricciones de contexto de seguridad (SCC) de OpenShift y puedan acceder a los volúmenes sin problemas de permisos. Para obtener más información, consulta "OpenShift documentación de restricciones de contexto de seguridad".

Puedes evitar que se sobrescriban anotaciones específicas en el espacio de nombres de destino configurando la variable de entorno de Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de realizar la restauración o la conmutación por error. Por ejemplo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

Al realizar una operación de restauración o conmutación por error, las anotaciones y etiquetas de espacios de nombres especificadas en `restoreSkipNamespaceAnnotations` y `restoreSkipNamespaceLabels` quedan excluidas de la operación de restauración o conmutación por error. Asegúrate de que estos ajustes se configuren durante la instalación inicial de Helm. Para obtener más información, consulta "Configura los ajustes adicionales del helm chart de Trident Protect".

Si instalaste la aplicación de origen usando Helm con la `--create-namespace` flag, se da un tratamiento especial a la clave de etiqueta `name`. Durante el proceso de restauración o conmutación por error, Trident Protect copia esta etiqueta al espacio de nombres de destino, pero actualiza el valor al valor del espacio de nombres de destino si el valor del origen coincide con el espacio de nombres de origen. Si este valor no coincide con el espacio de nombres de origen, se copia al espacio de nombres de destino sin cambios.

Ejemplo

El siguiente ejemplo presenta un espacio de nombres de origen y uno de destino, cada uno con anotaciones y etiquetas diferentes. Puedes ver el estado del espacio de nombres de destino antes y después de la operación, y cómo se combinan o sobrescriben las anotaciones y etiquetas en el espacio de nombres de destino.

Antes de la operación de restauración o conmutación por error

La siguiente tabla ilustra el estado de los espacios de nombres de origen y destino de ejemplo antes de la operación de restauración o conmutación por error:

Espacio de nombres	Anotaciones	Etiquetas
Namespace ns-1 (fuente)	• annotation.one/key: "valoractualizado" • annotation.two/key: "true"	• entorno=producción • compliance=hipaa • name=ns-1
Namespace ns-2 (destino)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

Después de la operación de restauración

La siguiente tabla ilustra el estado del espacio de nombres de destino de ejemplo después de la operación de restauración o conmutación por error. Se han añadido algunas claves, se han sobrescrito otras y la etiqueta `name` se ha actualizado para que coincida con el espacio de nombres de destino:

Espacio de nombres	Anotaciones	Etiquetas
Namespace ns-2 (destino)	• annotation.one/key: "valoractualizado" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • compliance=hipaa • entorno=producción • role=database

Puedes configurar Trident Protect para congelar y descongelar sistemas de archivos durante las operaciones de protección de datos. "Conoce más sobre cómo configurar la congelación del sistema de archivos con Trident Protect".

Ganchos de ejecución durante la conmutación por error y las operaciones inversas

Cuando uses la relación AppMirror para proteger tu aplicación, hay comportamientos específicos relacionados con los hooks de ejecución que deberías conocer durante las operaciones de conmutación por error y reversa.

- Durante la conmutación por error, los ganchos de ejecución se copian automáticamente del clúster de origen al clúster de destino. No necesitas volver a crearlos manualmente. Después de la conmutación por error, los ganchos de ejecución están presentes en la aplicación y se ejecutarán durante cualquier acción relevante.
- Durante la resincronización inversa o inversa, se eliminan todos los execution hooks existentes en la aplicación. Cuando la source application se convierte en la destination application, estos execution hooks no son válidos y se eliminan para evitar su ejecución.

Para obtener más información sobre los ganchos de ejecución, consulta "Administra los hooks de ejecución de Trident Protect".

Configura una relación de replicación

Configurar una relación de replicación implica lo siguiente:

- Elegir con qué frecuencia quieres que Trident Protect tome una instantánea de la app (que incluye los recursos de Kubernetes de la app, así como las instantáneas de volumen para cada uno de los volúmenes de la app)
- Elegir el calendario de replicación (incluye los recursos de Kubernetes y también los datos de volumen persistente)
- Configurar la hora para que se tome la instantánea

Pasos

1. En el clúster de origen, crea un AppVault para la aplicación de origen. Dependiendo de tu proveedor de almacenamiento, modifica un ejemplo en "AppVault recursos personalizados" para adaptarlo a tu entorno:

   Crea un AppVault usando un CR

   a. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-appvault-primary-source.yaml`).

   b. Configura los siguientes atributos:

      - **metadata.name**: (*Required*) El nombre del recurso personalizado AppVault. Toma nota del

nombre que elijas, porque otros archivos CR necesarios para una relación de replicación hacen referencia a este valor.

- **spec.providerConfig**: (*Obligatorio*) Almacena la configuración necesaria para acceder a AppVault usando el proveedor especificado. Elige un bucketName y cualquier otro detalle necesario para tu proveedor. Toma nota de los valores que elijas, porque otros archivos CR necesarios para una relación de replicación hacen referencia a estos valores. Consulta "AppVault recursos personalizados" para ver ejemplos de CR de AppVault con otros proveedores.
- **spec.providerCredentials**: (*Obligatorio*) Almacena referencias a cualquier credencial necesaria para acceder a AppVault utilizando el proveedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*Required*) Indica que el valor de la credencial debe venir de un secreto.
    - **key**: (*Obligatorio*) La clave válida del secreto a seleccionar.
    - **nombre**: (*Required*) Nombre del secreto que contiene el valor para este campo. Debe estar en el mismo espacio de nombres.
  - **spec.providerCredentials.secretAccessKey**: (*Obligatorio*) La clave de acceso utilizada para acceder al proveedor. El **name** debe coincidir con **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obligatorio*) determina qué proporciona la copia de seguridad; por ejemplo, NetApp ONTAP S3, S3 genérico, Google Cloud o Microsoft Azure. Valores posibles:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Después de que completes el archivo `trident-protect-appvault-primary-source.yaml` con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

Crea un AppVault usando la CLI

a. Crea el AppVault, reemplazando los valores entre corchetes con información de tu entorno:

```
tridentctl-protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name> -n
trident-protect
```

2. En el clúster de origen, crea la CR de la aplicación de origen:

Crea la aplicación fuente usando una CR

a. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-app-source.yaml`).

b. Configura los siguientes atributos:

- **metadata.name**: (*Obligatorio*) El nombre del recurso personalizado de la aplicación. Toma nota del nombre que elijas, porque otros archivos CR necesarios para una relación de replicación hacen referencia a este valor.
- **spec.includedNamespaces**: (*Obligatorio*) Una matriz de espacios de nombres y etiquetas asociadas. Usa los nombres de los espacios de nombres y, si quieres, limita el alcance de los espacios de nombres con etiquetas para especificar los recursos que existen en los espacios de nombres que aparecen aquí. El espacio de nombres de la aplicación debe formar parte de esta matriz.

  Ejemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      labelSelector: {}
```

c. Después de que completes el archivo `trident-protect-app-source.yaml` con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

Crea la aplicación de origen usando la CLI

a. Crea la aplicación fuente. Por ejemplo:

```
tridentctl-protect create app <my-app-name> --namespaces <namespaces-to-be-included> -n <my-app-namespace>
```

3. Opcionalmente, en el clúster de origen, toma una instantánea de la aplicación de origen. Esta instantánea se usa como base para la aplicación en el clúster de destino. Si te saltas este paso, tendrás que esperar a que se ejecute la siguiente instantánea programada para tener una instantánea reciente. Para crear una instantánea bajo demanda, consulta "Crea una instantánea a pedido".

4. En el clúster de origen, crea la programación de replicación CR:

> Además del programa que se proporciona a continuación, se recomienda crear un programa de instantáneas diarias independiente con un período de retención de 7 días para mantener una instantánea común entre clústeres de ONTAP emparejados. Esto garantiza que las instantáneas estén disponibles hasta por 7 días, pero el período de retención se puede personalizar según las necesidades del usuario.
>
> Si se produce una conmutación por error, el sistema puede usar estas instantáneas hasta por 7 días para operaciones inversas. Este enfoque hace que el proceso inverso sea más rápido y eficiente porque solo se transferirán los cambios hechos desde la última instantánea, no todos los datos.
>
> Si un programa existente para la aplicación ya cumple con los requisitos de retención deseados, no se requieren programaciones adicionales.

Crea el programa de replicación usando una CR

a. Crea un programa de replicación para la aplicación de origen:

   i. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-schedule.yaml`).

   ii. Configura los siguientes atributos:

   - **metadata.name**: (*Obligatorio*) El nombre del recurso personalizado de schedule.
   - **spec.appVaultRef**: (*Obligatorio*) Este valor debe coincidir con el campo metadata.name de la AppVault para la aplicación de origen.
   - **spec.applicationRef**: (*Obligatorio*) Este valor debe coincidir con el campo metadata.name del CR de la aplicación de origen.
   - **spec.backupRetention**: (*Obligatorio*) Este campo es obligatorio y el valor debe establecerse en 0.
   - **spec.enabled**: debe establecerse en true.
   - **spec.granularity**: debe establecerse en `Custom`.
   - **spec.recurrenceRule**: define una fecha de inicio en tiempo UTC y un intervalo de recurrencia.
   - **spec.snapshotRetention**: debe establecerse en 2.

   Ejemplo de YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
```

```
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. Después de que completes el archivo `trident-protect-schedule.yaml` con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-namespace
```

Crea la programación de replicación usando la CLI

a. Crea el programa de replicación, reemplazando los valores entre corchetes con información de tu entorno:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

Ejemplo:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. En el clúster de destino, crea una CR de aplicación de origen AppVault que sea idéntica a la CR AppVault que aplicaste en el clúster de origen y asígnale un nombre (por ejemplo, `trident-protect-appvault-primary-destination.yaml`).
6. Aplica la CR:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n trident-protect
```

7. Crea un CR de AppVault de destino para la aplicación de destino en el clúster de destino. Dependiendo de tu proveedor de almacenamiento, modifica un ejemplo en "AppVault recursos personalizados" para adaptarlo a tu entorno:

a. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-appvault-secondary-destination.yaml`).

b. Configura los siguientes atributos:

- **metadata.name**: (*Required*) El nombre del recurso personalizado AppVault. Toma nota del nombre que elijas, porque otros archivos CR necesarios para una relación de replicación hacen referencia a este valor.
- **spec.providerConfig**: (*Obligatorio*) Almacena la configuración necesaria para acceder a AppVault usando el proveedor especificado. Elige un `bucketName` y cualquier otro detalle necesario para tu proveedor. Toma nota de los valores que elijas, porque otros archivos CR necesarios para una relación de replicación hacen referencia a estos valores. Consulta "AppVault recursos personalizados" para ver ejemplos de CR de AppVault con otros proveedores.
- **spec.providerCredentials**: (*Obligatorio*) Almacena referencias a cualquier credencial necesaria para acceder a AppVault utilizando el proveedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*Required*) Indica que el valor de la credencial debe venir de un secreto.
    - **key**: (*Obligatorio*) La clave válida del secreto a seleccionar.
    - **nombre**: (*Required*) Nombre del secreto que contiene el valor para este campo. Debe estar en el mismo espacio de nombres.
  - **spec.providerCredentials.secretAccessKey**: (*Obligatorio*) La clave de acceso utilizada para acceder al proveedor. El **name** debe coincidir con **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obligatorio*) determina qué proporciona la copia de seguridad; por ejemplo, NetApp ONTAP S3, S3 genérico, Google Cloud o Microsoft Azure. Valores posibles:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Después de que completes el archivo `trident-protect-appvault-secondary-destination.yaml` con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. En el clúster de destino, crea un archivo CR de AppMirrorRelationship.

> Al usar una CR, crea manualmente el espacio de nombres de destino antes de aplicarla. Trident Protect crea espacios de nombres automáticamente solo cuando usas la CLI.

Crea un AppMirrorRelationship usando un CR

a. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-relationship.yaml`).

b. Configura los siguientes atributos:

- **metadata.name:** (Obligatorio) El nombre del recurso personalizado AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*Obligatorio*) Este valor debe coincidir con el nombre de AppVault para la aplicación de destino en el clúster de destino.
- **spec.namespaceMapping**: (*Obligatorio*) Los espacios de nombres de destino y de origen deben coincidir con el espacio de nombres de la aplicación definido en el CR de la aplicación respectiva.
- **spec.sourceAppVaultRef**: (*Obligatorio*) Este valor debe coincidir con el nombre de AppVault para la aplicación de origen.
- **spec.sourceApplicationName**: (*Obligatorio*) Este valor debe coincidir con el nombre de la aplicación de origen que definiste en el CR de la aplicación de origen.
- **spec.sourceApplicationUID**: (Obligatorio) Este valor debe coincidir con el UID de la aplicación de origen que definiste en el CR de la aplicación de origen.
- **spec.storageClassName**: (*opcional*) elige el nombre de una clase de almacenamiento válida en el clúster. La clase de almacenamiento debe estar vinculada a una máquina virtual de almacenamiento ONTAP que esté emparejada con el entorno de origen. Si no se proporciona la clase de almacenamiento, se usará la clase de almacenamiento predeterminada del clúster.
- **spec.storageClassMapping**: (*Opcional*) Asigna las clases de almacenamiento de origen a las de destino para PVC individuales dentro del mismo AMR. Cada entrada requiere un `source` y un nombre de clase de almacenamiento `destination`. Cuando se configura, `storageClassMapping` anula por completo `storageClassName`; los PVC que no tengan una entrada de origen coincidente usan la clase de almacenamiento predeterminada del clúster, no `storageClassName`.

  Al usar `storageClassMapping` en un CR de AppMirrorRelationship, ten en cuenta las siguientes restricciones:

  - `storageClassMapping` anula `storageClassName` por completo. Cuando ambos campos están configurados, `storageClassName` se ignora.
  - Los PVC con clases de almacenamiento de origen que no tienen una entrada correspondiente en `storageClassMapping` usan la clase de almacenamiento predeterminada del clúster, no `storageClassName`.
  - Cada entrada requiere tanto un `source` como un `destination` valor.
  - Cambiar la clase de almacenamiento de destino puede provocar la eliminación y recreación de los PVC de destino existentes.
- **spec.recurrenceRule**: define una fecha de inicio en tiempo UTC y un intervalo de recurrencia.

  Ejemplo de YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
```

```
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-2
```

c. Después de que completes el archivo `trident-protect-relationship.yaml` con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

Crea un AppMirrorRelationship usando la CLI

a. Crea y aplica el objeto AppMirrorRelationship, reemplazando los valores entre corchetes con información de tu entorno:

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> --storageclass-mapping
<src_class:dst_class> -n <application_namespace>
```

Usa `--storageclass-mapping` para asignar clases de almacenamiento de origen individuales a clases de almacenamiento de destino. Especifica uno o más pares `source:destination` separados por comas (por ejemplo, `"class-a:class-b,class-c:class-d"`). Usa `*:destination-class` como comodín para asignar todos los PVC sin asignar a una sola clase de almacenamiento de destino. Cuando `--storageclass-mapping` está configurado, `--storage-class` se ignora.

Ejemplo con una única clase de almacenamiento de destino (`--storage-class`):

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. (*Opcional*) En el clúster de destino, revisa el estado y el estatus de la relación de replicación:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Conmutar al clúster de destino

Con Trident Protect, puedes conmutar por error las aplicaciones replicadas a un clúster de destino. Este procedimiento detiene la relación de replicación y pone la app en línea en el clúster de destino. Trident Protect no detiene la app en el clúster de origen si estaba operativa.

Pasos

1. En el clúster de destino, edita el archivo CR AppMirrorRelationship (por ejemplo, `trident-protect-relationship.yaml`) y cambia el valor de **spec.desiredState** a `Promoted`.
2. Guarda el archivo CR.
3. Aplica la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (*Opcional*) Crea cualquier programación de protección que necesites en la aplicación conmutada por error.
5. (*Opcional*) Verifica el estado y el estatus de la relación de replicación:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Resincronizar una relación de replicación conmutada por error

La operación de resincronización restablece la relación de replicación. Después de realizar una operación de resincronización, la aplicación de origen original se convierte en la aplicación en ejecución y cualquier cambio hecho en la aplicación en ejecución en el clúster de destino se descarta.

El proceso detiene la app en el clúster de destino antes de restablecer la replicación.

Cualquier dato escrito en la aplicación de destino durante la conmutación por error se perderá.

Pasos

1. Opcional: en el clúster de origen, crea una instantánea de la aplicación de origen. Esto asegura que se capturen los últimos cambios del clúster de origen.
2. En el clúster de destino, edita el archivo CR de AppMirrorRelationship (por ejemplo, `trident-protect-relationship.yaml`) y cambia el valor de spec.desiredState a `Established`.
3. Guarda el archivo CR.
4. Aplica la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Si creaste alguna programación de protección en el clúster de destino para proteger la aplicación que se transfirió tras el fallo, elimínala. Cualquier programación que quede provoca fallos en las instantáneas del volumen.

Resincroniza de forma inversa una relación de replicación que ha fallado

Cuando haces una resincronización inversa de una relación de replicación fallida, la aplicación de destino se convierte en la aplicación de origen y la de origen en la de destino. Los cambios hechos en la aplicación de destino durante el failover se mantienen.

Pasos

1. En el clúster de destino original, elimina el CR AppMirrorRelationship. Esto hace que el destino se convierta en el origen. Si quedan programas de protección en el nuevo clúster de destino, elimínalos.
2. Configura una relación de replicación aplicando los archivos CR que usaste originalmente para configurar la relación en los clústeres opuestos.
3. Asegúrate de que el nuevo destino (clúster de origen original) esté configurado con ambos CR de AppVault.
4. Configura una relación de replicación en el clúster opuesto, configurando los valores para la dirección inversa.

Invertir la dirección de replicación de la aplicación

Cuando inviertes la dirección de replicación, Trident Protect mueve la aplicación al backend de almacenamiento de destino mientras sigue replicando de vuelta al backend de almacenamiento de origen original. Trident Protect detiene la aplicación de origen y replica los datos al destino antes de hacer el failover a la app de destino.

En esta situación, estás intercambiando el clúster de origen y el clúster de destino.

Pasos

1. En el clúster de origen, crea una instantánea de apagado:

Crea una instantánea de apagado usando un CR

a. Desactiva las programaciones de la política de protección para la aplicación de origen.

b. Crea un archivo ShutdownSnapshot CR:

i. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-shutdownsnapshot.yaml`).

ii. Configura los siguientes atributos:

- **metadata.name**: (*Requerido*) El nombre del recurso personalizado.
- **spec.AppVaultRef**: (*Required*) Este valor debe coincidir con el campo metadata.name del AppVault para la aplicación de origen.
- **spec.ApplicationRef**: (*Required*) Este valor debe coincidir con el campo metadata.name del archivo CR de la aplicación de origen.

Ejemplo de YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-c08a5dbe844e
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
```

c. Después de que completes el archivo `trident-protect-shutdownsnapshot.yaml` con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-namespace
```

Crea una instantánea de apagado usando la CLI

a. Crea la instantánea de apagado, reemplazando los valores entre corchetes por información de tu entorno. Por ejemplo:

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot> --appvault <my_vault> --app <app_to_snapshot> -n <application_namespace>
```

2. En el clúster de origen, después de que se complete la instantánea de apagado, obtén el estado de la instantánea de apagado:

```
kubectl get shutdownsnapshot -n my-app-namespace
```

```
<shutdown_snapshot_name> -o yaml
```

3. En el clúster de origen, encuentra el valor de **shutdownsnapshot.status.appArchivePath** usando el siguiente comando y apunta la última parte de la ruta del archivo (también llamada basename; esto será todo lo que esté después de la última barra):

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. Realiza una conmutación por error del nuevo clúster de destino al nuevo clúster de origen, con el siguiente cambio:

> En el paso 2 del procedimiento de conmutación por error, incluye el campo `spec.promotedSnapshot` en el archivo AppMirrorRelationship CR y pon su valor en el nombre base que anotaste en el paso 3 de arriba.

5. Realiza los pasos de resincronización inversa en Resincroniza de forma inversa una relación de replicación que ha fallado.
6. Habilita los calendarios de protección en el nuevo clúster de origen.

Resultado

Las siguientes acciones ocurren debido a la replicación inversa:

- Se toma una instantánea de los recursos de Kubernetes de la aplicación fuente original.
- Los pods de la app fuente original se detienen de forma gradual eliminando los recursos de Kubernetes de la app (dejando los PVCs y PVs en su lugar).
- Después de que los pods se apagan, se toman instantáneas de los volúmenes de la app y se replican.
- Las relaciones de SnapMirror se rompen, lo que deja los volúmenes de destino listos para lectura/escritura.
- Los recursos de Kubernetes de la app se restauran desde la instantánea previa al apagado, usando los datos de volumen replicados después de que la app de origen original se apagara.
- La replicación se restablece en sentido inverso.

Devuelve las aplicaciones al clúster de origen

Con Trident Protect, puedes lograr la "recuperación" después de una operación de conmutación por error usando la siguiente secuencia de operaciones. En este flujo de trabajo para restaurar la dirección de replicación original, Trident Protect replica (resincroniza) cualquier cambio de la aplicación de vuelta a la aplicación de origen original antes de invertir la dirección de replicación.

Este proceso parte de una relación que ha completado una conmutación por error a un destino e implica los siguientes pasos:

- Empieza con un estado conmutado por error.
- Resincroniza de forma inversa la relación de replicación.

> No realices una operación de resincronización normal, ya que esto descartará los datos

escritos en el clúster de destino durante el procedimiento de conmutación por error.

- Invierte la dirección de replicación.

Pasos

1. Realiza los pasos de Resincroniza de forma inversa una relación de replicación que ha fallado.
2. Realiza los pasos de Invertir la dirección de replicación de la aplicación.

Eliminar una relación de replicación

Puedes eliminar una relación de replicación en cualquier momento. Cuando eliminas la relación de replicación de aplicaciones, el resultado son dos aplicaciones separadas sin relación entre ellas.

Pasos

1. En el clúster de destino actual, elimina el CR AppMirrorRelationship:

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Migra aplicaciones usando Trident Protect

Puedes migrar tus aplicaciones entre clústeres o a diferentes clases de almacenamiento restaurando los datos de backup.

Cuando migras una aplicación, todos los ganchos de ejecución configurados para la aplicación se migran con la app. Si hay un gancho de ejecución posterior a la restauración, se ejecuta automáticamente como parte de la operación de restauración.

Operaciones de backup y restauración

Para realizar operaciones de backup y restauración en los siguientes escenarios, puedes automatizar tareas específicas de backup y restauración.

Clonar en el mismo clúster

Para clonar una aplicación en el mismo clúster, crea una instantánea o backup y restaura los datos en el mismo clúster.

Pasos

1. Debe realizar una de las siguientes acciones:
   a. "Crear una instantánea".
   b. "Crear un backup".
2. En el mismo clúster, haz una de las siguientes cosas, dependiendo de si creaste una instantánea o un backup:
   a. "Restaura tus datos desde la instantánea".
   b. "Restaura tus datos desde el backup".

Clonar a clúster diferente

Para clonar una aplicación en un clúster diferente (realizar un clon entre clústeres), crea un backup en el clúster de origen y luego restaura el backup en un clúster diferente. Asegúrate de que Trident Protect está instalado en el clúster de destino.

Puedes replicar una aplicación entre diferentes clústeres usando "Replicación de SnapMirror".

Pasos

1. "Crear un backup".
2. Asegúrate de que el AppVault CR para el bucket de almacenamiento de objetos que contiene el backup se ha configurado en el clúster de destino.
3. En el clúster de destino, "restaura tus datos desde el backup".

Migra aplicaciones de una clase de almacenamiento a otra clase de almacenamiento

Puedes migrar aplicaciones de una clase de almacenamiento a otra diferente restaurando un backup en la clase de almacenamiento de destino.

Por ejemplo (excluyendo los secretos del restore CR):

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

Restaura la instantánea usando una CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-snapshot-restore-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:

- **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
- **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la instantánea. Puedes usar el siguiente comando para encontrar esta ruta:

```
kubectl get snapshots <my-snapshot-name> -n trident-protect -o
jsonpath='{.status.appArchivePath}'
```

- **spec.appVaultRef**: (*Required*) El nombre de AppVault donde se almacenan los contenidos de la instantánea.
- **spec.namespaceMapping**: la asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplaza `my-source-namespace` y `my-destination-namespace` con información de tu entorno.

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

3. Opcionalmente, si necesitas seleccionar solo ciertos recursos de la aplicación para restaurar, agrega un filtrado que incluya o excluya recursos marcados con etiquetas particulares:

- **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `include or exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros de resourceMatchers para definir los recursos que se incluirán o excluirán:
  - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines varios elementos en este array, se combinan como una operación OR, y los campos dentro de cada elemento (group, kind, version) se combinan como una operación AND.
    - **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
    - **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.
    - **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.
    - **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
    - **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
    - **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de

Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

Por ejemplo:

```
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"]
```

4. Después de que completes el archivo `trident-protect-snapshot-restore-cr.yaml` con los valores correctos, aplica el CR:

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

Restaura la instantánea usando la CLI

Pasos

1. Restaura la instantánea en un espacio de nombres diferente, reemplazando los valores entre corchetes con información de tu entorno.

   - El `snapshot`argumento usa un espacio de nombres y un nombre de instantánea en el formato `<namespace>/<name>.
   - El argumento `namespace-mapping` usa espacios de nombres separados por dos puntos para mapear los espacios de nombres de origen a los espacios de nombres de destino correctos en el formato `source1:dest1,source2:dest2`.

   Por ejemplo:

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Protege las máquinas virtuales de KubeVirt con Trident Protect

Trident Protect te permite hacer copias de seguridad y restaurar máquinas virtuales (VM) de KubeVirt que se ejecutan en OpenShift Virtualization. Puedes proteger todas las VMs en uno o varios espacios de nombres a la vez, o usar el campo `includedVirtualMachines` para seleccionar VMs específicas por nombre. Trident Protect recopila automáticamente todos los recursos dependientes de las VMs durante la copia de seguridad y, al restaurar, puedes recuperar todo el archivo o restaurar de forma selectiva VMs individuales.

KubeVirt modos de protección

Trident Protect admite dos formas de definir qué máquinas virtuales de KubeVirt se deben proteger. Elige uno de estos modos al definir una aplicación:

Modo	Campo	Descripción
Basado en namespace	`includedNamespaces`	Protege todos los recursos coincidentes en uno o varios espacios de nombres. Usa un `labelSelector` en la entrada del espacio de nombres para controlar qué máquinas virtuales se incluyen dinámicamente.
De ámbito de la máquina virtual	`includedVirtualMachines`	Protege máquinas virtuales específicas de KubeVirt por espacio de nombres y nombre. Utiliza este modo cuando las máquinas virtuales estén repartidas por distintos espacios de nombres o cuando no desees proteger todos los recursos del espacio de nombres.

Debes especificar exactamente un campo (`includedNamespaces` o `includedVirtualMachines`) en el `ApplicationSpec`. No puedes utilizar ambos en la misma aplicación.

Ambos modos admiten filtros adicionales:

- Utiliza `resourceFilter` para incluir o excluir recursos específicos.
- Utiliza `includedClusterScopedResources` para incluir recursos compatibles con el ámbito del clúster, como objetos `StorageClass`.

Trident Protect puede congelar y descongelar los sistemas de archivos de las máquinas virtuales de KubeVirt durante las operaciones de protección de datos para garantizar la coherencia. Para obtener detalles de configuración, consulta "Protegiendo datos con KubeVirt VMs".

Recursos recopilados durante la copia de seguridad y la instantánea

Cuando creas una copia de seguridad o una instantánea para una aplicación que incluye KubeVirt VMs, Trident Protect recopila los recursos de las VMs seleccionadas y descubre automáticamente sus recursos dependientes.

El comportamiento de la recopilación se aplica en dos casos:

- Aplicaciones con ámbito de máquina virtual que utilizan `includedVirtualMachines`.
- Aplicaciones basadas en espacios de nombres que contienen recursos de KubeVirt `VirtualMachine`.

Recopilación con ámbito de máquina virtual

En el caso de las aplicaciones con ámbito de máquina virtual, Trident Protect comienza con cada `VirtualMachine` listado en `includedVirtualMachines` y descubre los recursos dependientes necesarios para restaurar esa máquina virtual.

Tipo de recurso	Ámbito	Cómo se descubre
`VirtualMachine`	Con espacio de nombres	Coincidencia directa de la `includedVirtualMachines` lista.
`VirtualMachineInstance`	Con espacio de nombres	El mismo espacio de nombres y nombre que la máquina virtual, si la máquina virtual está en ejecución.
`VirtualMachineInstancetype`	Con espacio de nombres	Referenciado por `spec.instancetype` cuando `kind: VirtualMachineInstancetype`.
`VirtualMachineClusterInstancetype`	Clúster	Referenciado por `spec.instancetype` cuando `kind` no está definido o `kind: VirtualMachineClusterInstancetype`.
`VirtualMachinePreference`	Con espacio de nombres	Referenciado por `spec.preference` cuando `kind: VirtualMachinePreference`.
`VirtualMachineClusterPreference`	Clúster	Referenciado por `spec.preference` cuando `kind` no está definido o `kind: VirtualMachineClusterPreference`.
`DataVolume`	Con espacio de nombres	Citado en `spec.template.spec.volumes[].dataVolume.name`.
`PersistentVolumeClaim`	Con espacio de nombres	Descubierto a partir de PVCs respaldados por DataVolume, que usan el mismo nombre que el DataVolume; referencias directas a PVC en `persistentVolumeClaim.claimName`; PVCs de volcado de memoria en `memoryDump.claimName`; y PVCs efímeros en `ephemeral.persistentVolumeClaim.claimName`.
`PersistentVolume`	Clúster	Detectado a partir del PV de respaldo para cada PVC recopilado usando `spec.volumeName`.
`Secret`	Con espacio de nombres	Descubiertos a partir de los secretos de volumen en `secret.secretName`; los secretos de descarga de imágenes en containerDisk; las referencias a secretos de datos de usuario y de red en CloudInitNoCloud y CloudInitConfigDrive; los secretos de Sysprep; los secretos de descarga de imágenes de contenedores en KernelBoot; y los secretos de SSH o contraseñas en AccessCredentials.
`ConfigMap`	Con espacio de nombres	Detectado desde el volumen ConfigMaps en `configMap.name` y Sysprep ConfigMaps.

Tipo de recurso	Ámbito	Cómo se descubre
`ServiceAccount`	Con espacio de nombres	Descubierto a partir de referencias de volúmenes `serviceAccount.serviceAccountName`.
`NetworkAttachmentDefinition`	Con espacio de nombres	Se detectan a partir de las redes Multus en `spec.template.spec.networks[].multus.networkName` cuando el nombre de la red no contiene `/`. Solo se recogen las referencias de NetworkAttachmentDefinition que pertenecen al mismo espacio de nombres. Las referencias entre espacios de nombres se omiten.
`Pod`	Con espacio de nombres	Detectado para `virt-launcher` Pods usando el selector de etiquetas `kubevirt.io=virt-launcher,vm.kubevirt.io/name=<vmName>`.
Clase de almacenamiento, volúmenes Trident y backends Trident	Clúster	Se recopilan de la misma forma que las aplicaciones basadas en espacios de nombres, a partir de los PVC detectados para la VM.

Una vez recopilados todos los recursos con espacio de nombres de ámbito de máquina virtual, Trident Protect aplica la estándar `resourceFilter`, si está configurada. Luego, la recopilación de recursos con ámbito de clúster continúa con normalidad.

Colección basada en espacios de nombres

En el caso de las aplicaciones basadas en espacios de nombres, Trident Protect analiza los espacios de nombres seleccionados y recopila todos los recursos que coinciden. Si se encuentran objetos KubeVirt `VirtualMachine`, Trident Protect recopila automáticamente sus recursos dependientes, incluidos los tipos de instancia, las preferencias y los Pods `virt-launcher`, del mismo modo que lo hace con las aplicaciones con ámbito de máquina virtual.

Esto significa que los recursos específicos de la máquina virtual se incluyen en tu copia de seguridad o instantánea, incluso cuando la aplicación se define con `includedNamespaces`.

Protege máquinas virtuales usando definiciones de aplicaciones basadas en espacios de nombres

Al crear una aplicación basada en espacios de nombres, Trident Protect analiza los espacios de nombres seleccionados y recopila todos los recursos, incluidas las máquinas virtuales de KubeVirt. Para cada máquina virtual encontrada, Trident Protect también recopila los recursos dependientes necesarios para realizar la copia de seguridad y la restauración de dicha máquina virtual.

Ejemplo de CR de aplicación basada en espacios de nombres:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
```

```
  includedNamespaces:
    - namespace: <vm_namespace>
```

Después de crear la aplicación, utiliza los flujos de trabajo estándar de Trident Protect para las copias de seguridad y las instantáneas:

- Para ver ejemplos de archivos YAML de instantáneas, consulta "Crea una instantánea a pedido".
- Para ver ejemplos de archivos YAML de copia de seguridad, consulta "Crear un backup a pedido".

Comportamiento de restauración para aplicaciones de máquinas virtuales basadas en espacios de nombres

Dado que las aplicaciones basadas en espacios de nombres recopilan recursos de los espacios de nombres seleccionados, la copia de seguridad o la instantánea pueden incluir objetos de tiempo de ejecución de KubeVirt y metadatos gestionados por KubeVirt. Algunos de estos objetos no deben restaurarse directamente porque representan el estado de tiempo de ejecución del clúster de origen.

Durante la restauración, Trident Protect aplica transformaciones compatibles con KubeVirt para eliminar los metadatos del clúster de origen y omitir los recursos transitorios de KubeVirt. Esto ayuda a KubeVirt a recrear la máquina virtual de forma limpia en el entorno restaurado.

Recurso	Transformar
VirtualMachine	Elimina las etiquetas y anotaciones que empiezan con `kubevirt.io/`.
VirtualMachineInstance	Elimina las etiquetas y anotaciones que empiezan con `kubevirt.io/`.
VirtualMachineInstanceMigration	Ignorado y no restaurado.
VirtualMachineSnapshot	Ignorado y no restaurado.
VirtualMachineSnapshotContent	Ignorado y no restaurado.
Pod (`virt-launcher`)	Se omite cuando el pod tiene la etiqueta `kubevirt.io=virt-launcher`.

Trident Protect restaura los recursos de KubeVirt en un orden definido para evitar problemas de dependencia. CDI DataVolumes (`cdi.kubevirt.io`) se restauran primero para que los datos del disco de la máquina virtual estén disponibles antes de que se creen los objetos de la máquina virtual. Después de eso, VirtualMachines (`kubevirt.io`) se restauran antes que VirtualMachineInstances.

Protección dinámica de máquinas virtuales con selectores de etiquetas

Si quieres proteger VMs de forma dinámica sin tener que especificar el nombre de cada VM, utiliza definiciones de aplicaciones basadas en espacios de nombres con selectores de etiquetas.

Solo el CR VirtualMachine necesita etiquetas. Los recursos dependientes como DataVolumes, PVC, secretos y ConfigMaps se recopilan automáticamente.

Pasos

1. Etiqueta los CR de las máquinas virtuales con una etiqueta compartida:

```
kubectl label virtualmachine <vm_name 1> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_2> app=<label_value> -n
<vm_namespace>
```

2. Crea un CR de aplicación con `includedNamespaces` y `labelSelector`:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
      labelSelector:
        matchLabels:
          app: <label_value>
```

3. Crea copias de seguridad y restauraciones usando el flujo de trabajo estándar de Trident Protect.

Más detalles:

- Puedes añadir o eliminar máquinas virtuales de la protección en cualquier momento cambiando las etiquetas:

```
kubectl label virtualmachine <vm_name_to_add> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_to_remove> app
=<different_label_value> -n <vm_namespace>
```

- También puedes utilizar selectores de etiquetas en varios espacios de nombres:

```
spec:
  includedNamespaces:
    - namespace: <vm_namespace_1>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
    - namespace: <vm_namespace_2>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
```

Protege máquinas virtuales concretas con includedVirtualMachines

Utiliza `includedVirtualMachines` cuando quieras proteger un conjunto nombrado de VirtualMachines en uno o varios espacios de nombres y restaurar de forma selectiva VMs específicas desde un archivo.

Tanto los CR de aplicación como los de restauración utilizan el mismo formato de selector `includedVirtualMachines`.

Utiliza este selector en diferentes contextos:

- En una solicitud CR, define qué máquinas virtuales están protegidas.
- En una solicitud de restaurar sin movimiento (CR), se define qué máquinas virtuales se restauran desde la copia de seguridad o el archivo de instantánea.

> En las restauraciones de CR, `includedVirtualMachines` es opcional. Si no incluyes este campo, se restaura todo el archivo. Cuando lo especificas, solo se restauran las máquinas virtuales enumeradas y sus recursos dependientes.

El YAML que aparece a continuación muestra la estructura del selector compartido:

```
includedVirtualMachines:
  - namespace: <vm_namespace_1>
    names:
      - <vm_name_1>
      - <vm_name_2>
  - namespace: <vm_namespace_2>
    names:
      - <vm_name_3>
```

> Varias entradas en `includedVirtualMachines` pueden hacer referencia a diferentes espacios de nombres, lo que permite definir y restaurar aplicaciones compatibles con máquinas virtuales en varios espacios de nombres.

Campo	Tipo	Descripción
espacio de nombres	cadena	Espacio de nombres de Kubernetes que contiene los objetos de máquina virtual de destino.
nombres	lista de cadenas	Uno o varios nombres de máquinas virtuales en ese espacio de nombres.

Ejemplo de aplicación basada en máquinas virtuales:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
```

```
  includedVirtualMachines:
    - namespace: <vm_namespace_1>
      names:
        - <vm_name_1>
        - <vm_name_2>
    - namespace: <vm_namespace_2>
      names:
        - <vm_name_3>
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

Restaurar máquinas virtuales específicas de KubeVirt

Puedes usar `includedVirtualMachines` en los CR de restauración para restaurar solo las máquinas virtuales seleccionadas de KubeVirt desde un backup, una instantánea o un archivo de instantáneas replicadas.

Cuando `includedVirtualMachines` se especifica en un CR de restauración, Trident Protect restaura solo las VirtualMachines indicadas y sus recursos dependientes, como `VirtualMachineInstance`, `DataVolume`, `PersistentVolumeClaim`, `PersistentVolume`, `Secret`, `ConfigMap`, `ServiceAccount` y otros objetos relacionados con las máquinas virtuales.

Si no especificas `includedVirtualMachines` en la CR de restauración, Trident Protect restaura el archivo completo según la configuración de la CR de restauración.

CR compatibles con la restauración y comandos CLI

La siguiente tabla muestra los CR de restauración que son compatibles con `includedVirtualMachines` y sus comandos CLI correspondientes.

Restaurar CR	Comando de la CLI	Descripción
`BackupRestore`	`tridentctl-protect create backuprestore`	Restaura desde una copia de seguridad a asignaciones de espacios de nombres nuevos o diferentes. Admite `namespaceMapping` y `storageClassMapping`.
`BackupInplaceRestore`	`tridentctl-protect create backupinplacerestore`	Restaura desde una copia de seguridad en el espacio de nombres original o en los espacios de nombres originales.
`SnapshotRestore`	`tridentctl-protect create snapshotrestore`	Restaura desde una instantánea a asignaciones de espacios de nombres nuevos o diferentes. Admite `namespaceMapping` y `storageClassMapping`.
`SnapshotInplaceRestore`	`tridentctl-protect create snapshotinplacerestore`	Restaura desde una instantánea en el espacio de nombres original o en los espacios de nombres originales.

Restaurar CR	Comando de la CLI	Descripción
`ReplicateSnapshotRestore`	Solo YAML, o utiliza el comando CLI compatible si está disponible en tu entorno	Restaura a partir de una instantánea replicada. Admite `namespaceMapping`, `storageClassMapping` y restaurar sin movimiento usando el indicador `inPlaceRestore`.

> La `tridentctl-protect` CLI no tiene un flag para `includedVirtualMachines` en los comandos de restaurar. Para restaurar máquinas virtuales específicas, debes aplicar directamente un archivo YAML de CR de restaurar usando `kubectl apply`. Puedes escribir el YAML desde cero o usar la CLI con `--dry-run --output yaml` para generar un archivo inicial, agregar el campo `includedVirtualMachines` bajo `spec` y luego aplicarlo.

Comportamiento de la restauración en el ámbito de la máquina virtual

El `includedVirtualMachines` campo de un CR de restauración utiliza el mismo formato de selector que el CR de la aplicación:

```
includedVirtualMachines:
  - namespace: <source_vm_namespace>
    names:
      - <vm_name_1>
      - <vm_name_2>
```

Para restaurar CR, los valores de los espacios de nombres en `includedVirtualMachines` se refieren a los espacios de nombres de origen tal como existen en la copia de seguridad, la instantánea o el archivo de instantánea replicada.

Si utilizas `namespaceMapping`, Trident Protect selecciona primero las máquinas virtuales solicitadas del archivo de origen y luego aplica las asignaciones de espacios de nombres durante la restauración. Si utilizas `storageClassMapping`, Trident Protect aplica las asignaciones de clases de almacenamiento durante la fase de transformación de la restauración.

Genera el archivo YAML de restauración con la CLI

Puedes utilizar `tridentctl-protect` para generar un CR de restauración básica y luego editar el archivo YAML para añadir `includedVirtualMachines`.

```
tridentctl-protect create <restore_command> <restore_name> \
  <restore_options> \
  --dry-run --output yaml > <restore_name>.yaml
```

Edita el archivo YAML generado y añade el campo `includedVirtualMachines` debajo de `spec`, luego aplica el archivo:

```
kubectl apply -f <restore_name>.yaml
```

Ejemplo: definir una aplicación basada en una máquina virtual

La siguiente aplicación protege tres máquinas virtuales concretas repartidas en dos espacios de nombres.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: production-vms
  namespace: prod
spec:
  includedVirtualMachines:
    - namespace: prod
      names:
        - app-server
        - database-server
    - namespace: monitoring
      names:
        - prometheus-vm
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

Ejemplo: crear una copia de seguridad

Crear una copia de seguridad para una aplicación basada en VM usa el CR de copia de seguridad estándar. No se requieren campos específicos de VM en el CR de copia de seguridad.

```
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  name: daily-backup
  namespace: prod
spec:
  applicationRef: production-vms
  appVaultRef: s3-vault
```

Ejemplo: restaurar una máquina virtual a partir de una copia de seguridad

En el siguiente ejemplo solo se restaura `database-server` desde el archivo de copia de seguridad. Las demás máquinas virtuales del archivo, como `app-server` y `prometheus-vm`, se omiten.

```
apiVersion: protect.trident.netapp.io/v1
```

```
kind: BackupRestore
metadata:
  name: restore-db-only
  namespace: prod-vms-restore
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  namespaceMapping:
    - source: prod
      destination: prod-dr
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f restore-db-only.yaml
```

Solo `database-server` y sus recursos dependientes, como DataVolumes, PVC, PV, Secrets, ConfigMaps y los recursos relacionados de KubeVirt, se restauran.

Ejemplo: restaurar una única máquina virtual sin movimiento desde una copia de seguridad

El siguiente ejemplo restaura únicamente `database-server` en su espacio de nombres original desde el archivo de backup.

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: ipr-db-only
  namespace: prod
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f ipr-db-only.yaml
```

Ejemplo: restaurar una máquina virtual a partir de una instantánea

En el siguiente ejemplo solo se restaura `ubuntu-vm-blue-a627be38` de un archivo de instantánea.

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: blue-vm-snap-restore
  namespace: blue
spec:
  appVaultRef: my-appvault
  appArchivePath: snapshots/protectctl-blue-vm/snap-1
  includedVirtualMachines:
    - namespace: blue
      names:
        - ubuntu-vm-blue-a627be38
```

```
kubectl apply -f blue-vm-snap-restore.yaml
```

Combina la restauración de máquinas virtuales con otros filtros

Puedes utilizar `includedVirtualMachines` junto con `resourceFilter`, `namespaceMapping` y `storageClassMapping` en el mismo CR de restaurar sin movimiento para afinar qué se restaura.

Cuando añades un `resourceFilter`, Trident Protect primero selecciona las máquinas virtuales listadas en `includedVirtualMachines` y sus recursos dependientes. Luego aplica el filtro de recursos para acotar aún más lo que se incluye o excluye de esos resultados.

```
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace>
      names:
        - <vm_name>
  resourceFilter:
    resourceSelectionCriteria: Include
    resourceMatchers:
      - kinds: ["PersistentVolumeClaim"]
        names: ["<pvc_name>"]
```

En el caso de `BackupRestore`, `SnapshotRestore` y `ReplicateSnapshotRestore`, `namespaceMapping` y `storageClassMapping` funcionan junto con `includedVirtualMachines`. Las referencias a los espacios de nombres en `includedVirtualMachines` se corresponden con los espacios de nombres de origen tal y como aparecen en el archivo. Las asignaciones de espacios de nombres y clases de almacenamiento se aplican durante la fase de transformación de la restauración, una vez finalizado el filtrado.

Para obtener más información sobre el comportamiento de la restauración, consulta "Restaura aplicaciones usando Trident Protect".

Crear una aplicación con ámbito de máquina virtual mediante la CLI

Usa la `--virtual-machines` flag para limitar una aplicación a máquinas virtuales específicas de KubeVirt en vez de a espacios de nombres completos. La `--virtual-machines` flag es mutuamente excluyente con `--namespaces`.

Pasos

1. Crea la aplicación ejecutando uno de los siguientes comandos y sustituyendo los valores entre corchetes por la información de tu entorno. El formato para la `--virtual-machines` opción es `<namespace>(<vm_name>)`. Para varias máquinas virtuales, usa nombres separados por comas dentro de los paréntesis:
   - Una sola máquina virtual en un espacio de nombres:

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace>(<vm_name>)"
     ```

   - Varias máquinas virtuales en distintos espacios de nombres:

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace_1>(<vm_name_1>,<vm_name_2>
     ),<vm_namespace_2>(<vm_name_3>)"
     ```

Información relacionada

- "Define una aplicación para la gestión"
- "Protege las aplicaciones"
- "Restaura aplicaciones usando Trident Protect"
- "Usa opciones avanzadas de restauración"
- "Instala el plugin CLI de Trident Protect"
- "Usa objetos de AppVault para gestionar los buckets"

Administra los hooks de ejecución de Trident Protect

Un gancho de ejecución es una acción personalizada que puedes configurar para que se ejecute junto con una operación de protección de datos de una app gestionada. Por ejemplo, si tienes una app de base de datos, puedes usar un gancho de ejecución para pausar todas las transacciones de la base de datos antes de una instantánea y reanudar las transacciones después de que la instantánea esté completa. Esto garantiza instantáneas consistentes con las aplicaciones.

Tipos de hooks de ejecución

Trident Protect admite los siguientes tipos de ganchos de ejecución, según cuándo pueden ejecutarse:

- Previa a la instantánea
- Posterior a la instantánea
- Previa a la copia de seguridad
- Posterior a la copia de seguridad
- Post-restauración
- Después de la conmutación por error

Orden de ejecución

Cuando se ejecuta una operación de protección de datos, los eventos de hook de ejecución ocurren en el siguiente orden:

1. Los hooks personalizados previos a la operación que sean aplicables se ejecutan en los contenedores correspondientes. Puedes crear y ejecutar tantos ganchos personalizados de ejecución previa a la operación como necesites, pero el orden de ejecución de estos ganchos antes de la operación no está garantizado ni es configurable.
2. Ocurre la congelación del sistema de archivos, si aplica. "Conoce más sobre cómo configurar la congelación del sistema de archivos con Trident Protect".
3. Se realiza la operación de protección de datos.
4. Los sistemas de archivos congelados se descongelan, si aplica.
5. Todos los hooks personalizados de ejecución post-operación aplicables se ejecutan en los contenedores apropiados. Puedes crear y ejecutar tantos hooks personalizados de ejecución post-operación como necesites, pero el orden de ejecución de estos hooks después de la operación no está garantizado ni es configurable.

Si creas varios ganchos de ejecución del mismo tipo (por ejemplo, pre-snapshot), el orden de ejecución de esos ganchos no está garantizado. Sin embargo, el orden de ejecución de ganchos de diferentes tipos sí está garantizado. Por ejemplo, este es el orden de ejecución de una configuración que tiene todos los diferentes tipos de ganchos:

1. Ganchos previos a la instantánea ejecutados
2. Ganchos post-snapshot ejecutados
3. Ganchos previos a la copia de seguridad ejecutados
4. Ganchos posteriores a la copia de seguridad ejecutados

El ejemplo de orden anterior solo se aplica cuando ejecutas una copia de seguridad que no usa una instantánea existente.

Siempre debes probar tus scripts de ganchos de ejecución antes de habilitarlos en un entorno de producción. Puedes usar el comando 'kubectl exec' para probar cómodamente los scripts. Después de habilitar los ganchos de ejecución en un entorno de producción, prueba las instantáneas y copias de seguridad resultantes para asegurarte de que son consistentes. Puedes hacer esto clonando la app en un espacio de nombres temporal, restaurando la instantánea o copia de seguridad y luego probando la app.

Si un hook de ejecución previo a la instantánea añade, cambia o elimina recursos de Kubernetes, esos cambios se incluyen en la instantánea o backup y en cualquier operación de restauración posterior.

Notas importantes sobre los ganchos de ejecución personalizados

Ten en cuenta lo siguiente al planificar los hooks de ejecución para tus apps.

- Un gancho de ejecución debe usar un script para realizar acciones. Muchos ganchos de ejecución pueden hacer referencia al mismo script.
- Trident Protect requiere que los scripts que usan los hooks de ejecución estén escritos en formato de scripts de shell ejecutables.
- El tamaño del script está limitado a 96KB.
- Trident Protect utiliza la configuración de los execution hook y cualquier criterio de coincidencia para determinar qué hooks son aplicables a una operación de snapshot, backup o restore.

Debido a que los ganchos de ejecución a menudo reducen o deshabilitan por completo la funcionalidad de la aplicación contra la que se ejecutan, siempre debes intentar minimizar el tiempo que tardan en ejecutarse tus ganchos de ejecución personalizados. Si inicias una operación de copia de seguridad o instantánea con ganchos de ejecución asociados pero luego la cancelas, los ganchos aún pueden ejecutarse si la operación de copia de seguridad o instantánea ya ha comenzado. Esto significa que la lógica utilizada en un gancho de ejecución posterior a la copia de seguridad no puede asumir que la copia de seguridad se completó.

Filtros de ejecución hook

Cuando añades o editas un gancho de ejecución para una aplicación, puedes añadir filtros al gancho de ejecución para gestionar en qué contenedores coincidirá el gancho. Los filtros son útiles para aplicaciones que usan la misma imagen de contenedor en todos los contenedores, pero pueden usar cada imagen para un propósito diferente (como Elasticsearch). Los filtros te permiten crear escenarios en los que los ganchos de ejecución se ejecutan en algunos, pero no necesariamente en todos los contenedores idénticos. Si creas varios filtros para un solo gancho de ejecución, se combinan con un operador lógico AND. Puedes tener hasta 10 filtros activos por gancho de ejecución.

Cada filtro que añadas a un gancho de ejecución utiliza una expresión regular para identificar contenedores en tu clúster. Cuando un gancho coincide con un contenedor, el gancho ejecutará su script asociado en ese contenedor. Las expresiones regulares para los filtros utilizan la sintaxis de Regular Expression 2 (RE2), que no admite la creación de filtros que excluyan contenedores de la lista de coincidencias. Para información sobre la sintaxis que Trident Protect admite para expresiones regulares en los filtros de ganchos de ejecución, consulta "Compatibilidad con la sintaxis de Regular Expression 2 (RE2)".

Si añades un filtro de espacio de nombres a un gancho de ejecución que se ejecuta después de una operación de restauración o clonación, y el origen y el destino de la restauración o clonación están en espacios de nombres diferentes, el filtro de espacio de nombres solo se aplica al espacio de nombres de destino.

Ejemplos de execution hook

Visita el "Proyecto NetApp Verda GitHub" para descargar ganchos de ejecución reales para apps populares como Apache Cassandra y Elasticsearch. También puedes ver ejemplos y obtener ideas para estructurar tus propios ganchos de ejecución personalizados.

Los scripts del repositorio de Verda se proporcionan «tal cual» y no cuentan con soporte oficial por parte de NetApp. NetApp solo abarca el framework de hooks de ejecución dentro de Trident Protect.

Crear un gancho de ejecución

Puedes crear un gancho de ejecución personalizado para una app usando Trident Protect. Necesitas tener permisos de Owner, Admin o Member para crear ganchos de ejecución.

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-hook.yaml`.
2. Configura los siguientes atributos para que coincidan con tu entorno de Trident Protect y la configuración del clúster:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.applicationRef**: (*Required*) el nombre de Kubernetes de la aplicación para la que quieres ejecutar el gancho de ejecución.
   - **spec.stage**: (*Required*) Una cadena que indica en qué etapa de la acción debe ejecutarse el gancho de ejecución. Valores posibles:
     - Pre
     - Publicar
   - **spec.action**: (*Required*) Una cadena que indica qué acción realizará el gancho de ejecución, suponiendo que coincida cualquier filtro de gancho de ejecución especificado. Valores posibles:
     - Snapshot
     - Copia de seguridad
     - Restaurar
     - Conmutación por error
   - **spec.enabled**: (*Opcional*) Indica si este execution hook está habilitado o deshabilitado. Si no se especifica, el valor predeterminado es true.
   - **spec.hookSource**: (*Required*) Una cadena que contiene el script de hook codificado en base64.
   - **spec.timeout**: (*Opcional*) Un número que define cuánto tiempo en minutos se permite que se ejecute el hook de ejecución. El valor mínimo es 1 minuto y el valor por defecto es 25 minutos si no se especifica.
   - **spec.arguments**: (*Opcional*) Una lista YAML de argumentos que puedes especificar para el gancho de ejecución.
   - **spec.matchingCriteria**: (*Opcional*) Una lista opcional de pares clave-valor de criterios, cada par constituye un filtro de gancho de ejecución. Puedes añadir hasta 10 filtros por gancho de ejecución.
   - **spec.matchingCriteria.type**: (*Opcional*) una cadena que identifica el tipo de filtro de gancho de ejecución. Valores posibles:
     - ContainerImage

        - ContainerName
        - PodName
        - PodLabel
        - NamespaceName
    - **spec.matchingCriteria.value**: (*Opcional*) Una cadena o expresión regular que identifica el valor del filtro de ejecución.

    Ejemplo de YAML:

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

3. Después de rellenar el archivo CR con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-hook.yaml
```

Usa la CLI

Pasos

1. Crea el gancho de ejecución, sustituyendo los valores entre corchetes por información de tu entorno. Por ejemplo:

```
tridentctl-protect create exechook <my_exec_hook_name> --action
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>
--source-file <script-file> -n <application_namespace>
```

Ejecuta manualmente un gancho de ejecución

Puedes ejecutar manualmente un gancho de ejecución con fines de prueba o si necesitas volver a ejecutar el gancho manualmente después de un fallo. Necesitas tener permisos de Owner, Admin o Member para ejecutar manualmente ganchos de ejecución.

Ejecutar manualmente un hook de ejecución consiste en dos pasos básicos:

1. Crea una copia de seguridad de recursos, que recoge los recursos y crea una copia de seguridad de ellos, determinando dónde se ejecutará el hook
2. Ejecuta el hook de ejecución contra la copia de seguridad

▼ **Paso 1: crea una copia de seguridad de los recursos**

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-resource-backup.yaml`.
2. Configura los siguientes atributos para que coincidan con tu entorno de Trident Protect y la configuración del clúster:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.applicationRef**: (*Required*) El nombre de Kubernetes de la aplicación para la que crear la copia de seguridad de recursos.
   - **spec.appVaultRef**: (*Required*) el nombre del AppVault donde se almacena el contenido de la copia de seguridad.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la copia de seguridad. Puedes usar el siguiente comando para encontrar esta ruta:

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

Ejemplo de YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ResourceBackup
metadata:
  name: example-resource-backup
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
```

3. Después de rellenar el archivo CR con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-resource-backup.yaml
```

Usa la CLI

Pasos

1. Crea el backup, reemplazando los valores entre corchetes con información de tu entorno. Por ejemplo:

```
tridentctl protect create resourcebackup <my_backup_name> --app
<my_app_name> --appvault <my_appvault_name> -n <my_app_namespace>
--app-archive-path <app_archive_path>
```

2. Consulta el estado de la copia de seguridad. Puedes usar este comando de ejemplo repetidamente hasta que la operación termine:

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. Verifica que la copia de seguridad se realizó correctamente:

```
kubectl describe resourcebackup <my_backup_name>
```

▼ Paso 2: ejecuta el gancho de ejecución

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-hook-run.yaml`.

2. Configura los siguientes atributos para que coincidan con tu entorno de Trident Protect y la configuración del clúster:

   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.applicationRef**: (*Obligatorio*) asegúrate de que este valor coincide con el nombre de la aplicación del CR ResourceBackup que creaste en el paso 1.
   - **spec.appVaultRef**: (*Obligatorio*) Asegúrate de que este valor coincide con el appVaultRef del ResourceBackup CR que creaste en el paso 1.
   - **spec.appArchivePath**: asegúrate de que este valor coincide con el appArchivePath del CR ResourceBackup que creaste en el paso 1.

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

   - **spec.action**: (*Required*) Una cadena que indica qué acción realizará el gancho de ejecución, suponiendo que coincida cualquier filtro de gancho de ejecución especificado. Valores posibles:
     - Snapshot
     - Copia de seguridad
     - Restaurar
     - Conmutación por error
   - **spec.stage**: (*Required*) Una cadena que indica en qué etapa de la acción debe ejecutarse el gancho de ejecución. Esta ejecución de gancho no ejecutará ganchos en ninguna otra etapa. Valores posibles:
     - Pre
     - Publicar

     Ejemplo de YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover
```

3. Después de rellenar el archivo CR con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-hook-run.yaml
```

Usa la CLI

Pasos

1. Crea la solicitud para ejecutar manualmente el gancho:

```
tridentctl protect create exechooksrun <my_exec_hook_run_name> -n
<my_app_namespace> --action snapshot --stage <pre_or_post> --app
<my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>
```

2. Comprueba el estado de la ejecución del hook de ejecución. Puedes ejecutar este comando repetidamente hasta que se complete la operación:

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. Describe el objeto exechooksrun para ver los detalles finales y el estado:

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Información de copyright

Copyright © 2026 NetApp, Inc. Todos los derechos reservados. Imprimido en EE. UU. No se puede reproducir este documento protegido por copyright ni parte del mismo de ninguna forma ni por ningún medio (gráfico, electrónico o mecánico, incluidas fotocopias, grabaciones o almacenamiento en un sistema de recuperación electrónico) sin la autorización previa y por escrito del propietario del copyright.

El software derivado del material de NetApp con copyright está sujeto a la siguiente licencia y exención de responsabilidad:

ESTE SOFTWARE LO PROPORCIONA NETAPP «TAL CUAL» Y SIN NINGUNA GARANTÍA EXPRESA O IMPLÍCITA, INCLUYENDO, SIN LIMITAR, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO, CUYA RESPONSABILIDAD QUEDA EXIMIDA POR EL PRESENTE DOCUMENTO. EN NINGÚN CASO NETAPP SERÁ RESPONSABLE DE NINGÚN DAÑO DIRECTO, INDIRECTO, ESPECIAL, EJEMPLAR O RESULTANTE (INCLUYENDO, ENTRE OTROS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTIVOS, PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS, O INTERRUPCIÓN DE LA ACTIVIDAD EMPRESARIAL) CUALQUIERA SEA EL MODO EN EL QUE SE PRODUJERON Y LA TEORÍA DE RESPONSABILIDAD QUE SE APLIQUE, YA SEA EN CONTRATO, RESPONSABILIDAD OBJETIVA O AGRAVIO (INCLUIDA LA NEGLIGENCIA U OTRO TIPO), QUE SURJAN DE ALGÚN MODO DEL USO DE ESTE SOFTWARE, INCLUSO SI HUBIEREN SIDO ADVERTIDOS DE LA POSIBILIDAD DE TALES DAÑOS.

NetApp se reserva el derecho de modificar cualquiera de los productos aquí descritos en cualquier momento y sin aviso previo. NetApp no asume ningún tipo de responsabilidad que surja del uso de los productos aquí descritos, excepto aquello expresamente acordado por escrito por parte de NetApp. El uso o adquisición de este producto no lleva implícita ninguna licencia con derechos de patente, de marcas comerciales o cualquier otro derecho de propiedad intelectual de NetApp.

Es posible que el producto que se describe en este manual esté protegido por una o más patentes de EE. UU., patentes extranjeras o solicitudes pendientes.

LEYENDA DE DERECHOS LIMITADOS: el uso, la copia o la divulgación por parte del gobierno están sujetos a las restricciones establecidas en el subpárrafo (b)(3) de los derechos de datos técnicos y productos no comerciales de DFARS 252.227-7013 (FEB de 2014) y FAR 52.227-19 (DIC de 2007).

Los datos aquí contenidos pertenecen a un producto comercial o servicio comercial (como se define en FAR 2.101) y son propiedad de NetApp, Inc. Todos los datos técnicos y el software informático de NetApp que se proporcionan en este Acuerdo tienen una naturaleza comercial y se han desarrollado exclusivamente con fondos privados. El Gobierno de EE. UU. tiene una licencia limitada, irrevocable, no exclusiva, no transferible, no sublicenciable y de alcance mundial para utilizar los Datos en relación con el contrato del Gobierno de los Estados Unidos bajo el cual se proporcionaron los Datos. Excepto que aquí se disponga lo contrario, los Datos no se pueden utilizar, desvelar, reproducir, modificar, interpretar o mostrar sin la previa aprobación por escrito de NetApp, Inc. Los derechos de licencia del Gobierno de los Estados Unidos de América y su Departamento de Defensa se limitan a los derechos identificados en la cláusula 252.227-7015(b) de la sección DFARS (FEB de 2014).

Información de la marca comercial

NETAPP, el logotipo de NETAPP y las marcas que constan en http://www.netapp.com/TM son marcas comerciales de NetApp, Inc. El resto de nombres de empresa y de producto pueden ser marcas comerciales de sus respectivos propietarios.