Gestione y proteja aplicaciones

Trident

NetApp
February 20, 2026

Tabla de contenidos

Gestione y proteja aplicaciones

Utilice objetos Trident Protect AppVault para administrar depósitos

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

Debe crear una CR de AppVault manualmente o desde la línea de comandos al realizar operaciones de protección de datos en una aplicación. La CR de AppVault es específica de su entorno, y puede usar los ejemplos de esta página como guía al crearlas.

> Asegúrese de que AppVault CR esté en el clúster donde está instalado Trident Protect. Si el CR de AppVault no existe o no puede acceder a él, la línea de comandos mostrará un error.

Configurar la autenticación y las contraseñas de AppVault

Antes de crear un CR de AppVault, asegúrese de que AppVault y el transportador de datos que elija puedan autenticarse con el proveedor y cualquier recurso relacionado.

Contraseñas del repositorio de Data Mover

Cuando crea objetos de AppVault utilizando CR o el complemento CLI de Trident Protect, puede especificar un secreto de Kubernetes con contraseñas personalizadas para el cifrado de Restic y Kopia. Si no especifica un secreto, Trident Protect utiliza una contraseña predeterminada.

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

Cree un secreto de contraseña del repositorio de Data Mover

Utilice los siguientes ejemplos para crear la contraseña secreta. Cuando crea objetos de AppVault, puede indicarle a Trident Protect que use este secreto para autenticarse con el repositorio de transferencia de datos.

> - Dependiendo de qué transmisor de datos esté utilizando, solo necesita incluir la contraseña correspondiente para ese transmisor de datos. Por ejemplo, si está utilizando Restic y no planea usar KOPIA en el futuro, puede incluir solo la contraseña de Restic cuando cree el secreto.
> - Guarde la contraseña en un lugar seguro. La necesitará para restaurar datos en el mismo clúster o en uno diferente. Si el clúster o el… `trident-protect` Si se elimina el espacio de nombres, no podrá restaurar sus copias de seguridad o instantáneas sin la contraseña.

Utilice 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
```

Utilice 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 de IAM de almacenamiento compatibles con S3

Cuando accede a almacenamiento compatible con S3, como Amazon S3, Generic S3, "StorageGRID S3" , o "ONTAP S3" Al utilizar Trident Protect, debe asegurarse de que las credenciales de usuario que proporcione tengan los permisos necesarios para acceder al depósito. El siguiente es un ejemplo de una política que otorga los permisos mínimos requeridos para acceder con Trident Protect. Puede aplicar esta política al usuario que administra las políticas de bucket 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, consulte los ejemplos en la "Documentación de Amazon S3" .

Identidad de pod EKS para la autenticación de Amazon S3 (AWS)

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

Requisitos para la identidad del pod EKS con Trident Protect

Antes de utilizar EKS Pod Identity con Trident Protect, asegúrese de lo siguiente:

- Su clúster EKS tiene Identidad de Pod habilitada.
- Ha creado una función de IAM con los permisos de bucket S3 necesarios. Para obtener más información, consulte"Permisos de IAM de almacenamiento compatibles 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 de IAM con cuentas de servicio, consulte la "Documentación de identidad de pod de AWS EKS" .

Configuración de AppVault Al usar EKS Pod Identity, configure su AppVault CR con la `useIAM: true` bandera en lugar 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
```

Ejemplos de generación de claves de AppVault para proveedores de cloud

Al definir un CR de AppVault, debe incluir credenciales para acceder a los recursos alojados por el proveedor, a menos que utilice la autenticación IAM. La forma de generar las claves para las credenciales variará según el proveedor. Los siguientes son ejemplos de generación de claves de línea de comandos para varios proveedores. Puede utilizar 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
```

Genérico S3

```
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

A continuación se muestran ejemplos de definiciones de AppVault para cada proveedor.

Ejemplos de AppVault CR

Puede utilizar los siguientes ejemplos de CR para crear objetos de AppVault para cada proveedor de cloud.

- Opcionalmente, puede especificar un secreto de Kubernetes que contenga contraseñas personalizadas para el cifrado del repositorio de Restic y KOPIA. Consulte Contraseñas del repositorio de Data Mover si desea obtener más información.
- Para los objetos AppVault de Amazon S3 (AWS), puede especificar opcionalmente un sessionToken, lo que resulta útil si utiliza el inicio de sesión único (SSO) para la autenticación. Este token se crea cuando se generan claves para el proveedor en Ejemplos de generación de claves de AppVault para proveedores de cloud.
- Para los objetos S3 AppVault, puede especificar opcionalmente una URL de proxy de salida para el tráfico S3 saliente mediante la `spec.providerConfig.S3.proxyURL` clave.

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)

```yaml
---
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 utilizan Pod Identity con el transportador de datos Kopia, puede eliminar el `providerCredentials` sección y agregar `useIAM: true` bajo el `s3` configuración 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
```

Genérico S3

```
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 mediante la CLI de Trident Protect

Puede utilizar los siguientes ejemplos de comandos CLI para crear AppVault CRS para cada proveedor.

- Opcionalmente, puede especificar un secreto de Kubernetes que contenga contraseñas personalizadas para el cifrado del repositorio de Restic y KOPIA. Consulte Contraseñas del repositorio de Data Mover si desea obtener más información.
- Para los objetos S3 AppVault, puede especificar opcionalmente una URL de proxy de salida para el tráfico S3 saliente mediante el `--proxy-url <ip_address:port>` argumento.

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
```

Genérico S3

```
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
```

Apoyado `providerConfig.s3` opciones de configuración

Consulte la siguiente tabla para conocer las opciones de configuración del proveedor S3:

Parámetro	Descripción	Predeterminado	Ejemplo
`providerConfig.s3.skipCertValidation`	Deshabilitar la verificación del certificado SSL/TLS.	falso	"verdadero", "falso"
`providerConfig.s3.secure`	Habilite la comunicación HTTPS segura con el punto final S3.	verdadero	"verdadero", "falso"
`providerConfig.s3.proxyURL`	Especifique la URL del servidor proxy utilizado para conectarse a S3.	Ninguno	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Proporcione un certificado CA raíz personalizado para la verificación SSL/TLS.	Ninguno	"CN=MiCAPersonalizada"
`providerConfig.s3.useIAM`	Habilite la autenticación IAM para acceder a los buckets S3. Aplicable para la identidad del pod EKS.	falso	verdadero, falso

Ver información de AppVault

Puede utilizar el complemento CLI de Trident Protect para ver información sobre los objetos de AppVault que ha creado en el clúster.

Pasos

1. Ver el contenido de un objeto AppVault:

```
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 AppVaultPath para cada recurso, utilice el indicador `--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

Puede eliminar un objeto AppVault en cualquier momento.

No elimine `finalizers` la clave de AppVault CR antes de eliminar el objeto AppVault. Si lo hace, puede dar como resultado datos residuales en el bucket de AppVault y recursos huérfanos en el cluster.

Antes de empezar

Asegúrese de haber eliminado todos los CRS de instantánea y copia de seguridad que utiliza el AppVault que desea eliminar.

Quite un AppVault con la CLI de Kubernetes

1. Elimine el objeto AppVault, sustituyéndolo `appvault-name` por el nombre del objeto AppVault que desea eliminar:

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

Eliminar un AppVault mediante la CLI de Trident Protect

1. Elimine el objeto AppVault, sustituyéndolo `appvault-name` por el nombre del objeto AppVault que desea eliminar:

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

Definir una aplicación para la gestión con Trident Protect

Puede definir una aplicación que desee administrar con Trident Protect creando un CR de aplicación y un CR de AppVault asociado.

Cree un CR de AppVault

Debe crear un CR de AppVault que se utilizará al realizar operaciones de protección de datos en la aplicación, y el CR de AppVault debe residir en el clúster donde está instalado Trident Protect. La solicitud de cambio (CR) de AppVault es específica de su entorno; para ver ejemplos de solicitudes de cambio de cambio de AppVault, consulte"Recursos personalizados de AppVault."

Defina una aplicación

Debe definir cada aplicación que desee administrar con Trident Protect. Puede definir una aplicación para su administración ya sea creando manualmente un CR de aplicación o utilizando la CLI de Trident Protect.

Agregar una aplicación mediante un CR

Pasos

1. Cree el archivo CR de la aplicación de destino:
   a. Cree el archivo de recursos personalizados (CR) y asígnele un nombre (por ejemplo, `maria-app.yaml`).
   b. Configure los siguientes atributos:
      - **metadata.name**: (*required*) El nombre del recurso personalizado de la aplicación. Tenga en cuenta el nombre que elija porque otros archivos CR necesarios para las operaciones de protección hacen referencia a este valor.
      - **spec.includedNamespaces**: (*required*) Utilice el espacio de nombres y el selector de etiquetas 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 etiquetas es opcional y se puede utilizar para filtrar recursos dentro de cada espacio de nombres especificado.
      - **spec.includedClusterScopedResources**: (*Optional*) Utilice este atributo para especificar los recursos de ámbito de cluster que se incluirán en la definición de la aplicación. Este atributo le permite seleccionar estos recursos en función de su grupo, versión, tipo y etiquetas.
        - **GroupVersionKind**: (*required*) Especifica el grupo API, la versión y el tipo del recurso de ámbito de cluster.
        - **LabelSelector**: (*Optional*) Filtra los recursos de ámbito de cluster en función de sus etiquetas.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Opcional*) Esta anotación solo es aplicable a aplicaciones definidas desde máquinas virtuales, como en entornos KubeVirt, donde las congelaciones del sistema de archivos ocurren antes de las instantáneas. Especifique si esta aplicación puede escribir en el sistema de archivos durante una instantánea. Si se establece en verdadero, la aplicación ignora la configuración global y puede escribir en el sistema de archivos durante una instantánea. Si se establece en falso, 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, la anotación se ignora. Si no se especifica, la aplicación sigue el procedimiento establecido."Configuración de congelación global de Trident Protect" .

Si necesita aplicar esta anotación después de crear una aplicación, puede utilizar el siguiente comando:

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

+
Ejemplo 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*) Agregue filtros que incluyan o excluyan recursos marcados con etiquetas particulares:

   - **ResourceFilter.resourceSelectionCriteria**: (Requerido para filtrar) Usar `Include` o `Exclude` incluir o excluir un recurso definido en resourceMatchers. Agregue los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:

     - **ResourceFilter.resourceMatchers**: Una matriz de objetos resourceMatcher. Si define varios elementos en esta matriz, coinciden como una OPERACIÓN OR y los campos dentro de cada elemento (grupo, tipo, versión) coinciden como una operación AND.
       - **ResourceMatchers[].group**: (*Optional*) Grupo del recurso a filtrar.
       - **ResourceMatchers[].kind**: (*Optional*) Tipo de recurso a filtrar.
       - **ResourceMatchers[].version**: (*Optional*) Versión del recurso que se va a filtrar.
       - **ResourceMatchers[].names**: (*Optional*) Nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.

- **ResourceMatchers[].namespaces**: (*Optional*) Espacios de nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.
- **ResourceMatchers[].labelSelectors**: (*Optional*) Cadena de selector de etiquetas en el campo Kubernetes metadata.name del recurso tal como se define en el "Documentación de Kubernetes". Por ejemplo `"trident.netapp.io/os=linux"`: .

> Cuando ambos `resourceFilter` y `labelSelector` se utilizan, `resourceFilter` corre 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"]
```

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

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

Pasos

1. Cree y aplique la definición de la aplicación utilizando uno de los siguientes ejemplos, sustituyendo valores entre paréntesis por información de su entorno. Puede incluir espacios de nombres y recursos en la definición de la aplicación mediante listas separadas por comas con los argumentos que se muestran en los ejemplos.

   Opcionalmente, puede usar una anotación al crear una aplicación para especificar si la aplicación puede escribir en el sistema de archivos durante una instantánea. Esto solo es aplicable a aplicaciones definidas desde máquinas virtuales, como en entornos KubeVirt, donde se producen congelaciones del sistema de archivos antes de las instantáneas. Si configura la anotación a `true` La aplicación ignora la configuración global y puede escribir en el sistema de archivos durante una instantánea. Si lo configuras para `false` La aplicación ignora la configuración global y el sistema de

archivos se congela durante la toma de una instantánea. Si utiliza la anotación pero la aplicación no tiene máquinas virtuales en la definición de la aplicación, la anotación se ignora. Si no utilizas la anotación, la aplicación sigue el comportamiento esperado."Configuración de congelación global de Trident Protect" .

Para especificar la anotación al utilizar la CLI para crear una aplicación, puede utilizar el `--annotation` indicador.

- Cree la aplicación y utilice 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>
```

- Cree la aplicación y configure la configuración de la aplicación local 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 utilizar `--resource-filter-include` y `--resource-filter-exclude` banderas para incluir o excluir recursos según `resourceSelectionCriteria` como grupo, tipo, versión, etiquetas, nombres y espacios de nombres, 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"]}]'
```

Proteja las aplicaciones con Trident Protect

Puede proteger todas las aplicaciones administradas por Trident Protect tomando instantáneas y copias de seguridad mediante una política de protección automatizada o de forma ad hoc.

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

Crear una snapshot bajo demanda

Puede crear una snapshot bajo demanda en cualquier momento.

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

Cree una instantánea con un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-snapshot-cr.yaml`.
2. En el archivo creado, configure los siguientes atributos:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.applicationRef**: El nombre de Kubernetes de la aplicación a la instantánea.
   - **Spec.appVaultRef**: (*required*) El nombre del AppVault donde se debe almacenar el contenido de la instantánea (metadatos).
   - **Spec.reclaimer Policy**: (*Optional*) define lo que sucede con el AppArchive de una instantánea cuando se elimina el CR de la instantánea. Esto significa que incluso cuando se define en `Retain`, la instantánea se suprimirá. 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 rellenar `trident-protect-snapshot-cr.yaml` el archivo con los valores correctos, aplique el CR:

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

Cree una copia Snapshot mediante la CLI

Pasos

1. Cree la instantánea, reemplazando valores entre paréntesis con información de su entorno. Por ejemplo:

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

Cree un backup bajo demanda

Puede realizar una copia de seguridad de una aplicación en cualquier momento.

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

Antes de empezar

Asegúrese de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de copia de seguridad de S3 que se ejecute durante mucho tiempo. Si el token caduca durante la operación de copia de seguridad, la operación puede fallar.

- Consulte el "Documentación de la API de AWS" para obtener más información sobre la comprobación de la caducidad del token de sesión actual.
- Consulte el documento para "Documentación de AWS IAM" obtener más información acerca de las credenciales con recursos de AWS.

Cree una copia de seguridad con un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-backup-cr.yaml`.
2. En el archivo creado, configure los siguientes atributos:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.applicationRef**: (*required*) El nombre de Kubernetes de la aplicación para realizar una copia de seguridad.
   - **Spec.appVaultRef**: (*required*) El nombre del AppVault donde se debe almacenar el contenido de la copia de seguridad.
   - **SPEC.DATAMOVER**: (*Optional*) Una cadena que indica qué herramienta de copia de seguridad usar para la operación de copia de seguridad. Valores posibles (distingue mayúsculas de minúsculas):
     - `Restic`
     - `Kopia` (predeterminado)
   - **Spec.reclaimer Policy**: (*Optional*) define lo que sucede con una copia de seguridad cuando se libera de su reclamación. Los posibles valores son los siguientes:
     - `Delete`
     - `Retain` (predeterminado)
   - **spec.snapshotRef**: (*Opcional*): Nombre de la instantánea que se utilizará como origen de la copia de seguridad. Si no se proporciona, se creará una instantánea temporal y se realizará una copia de seguridad.

     Ejemplo 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 rellenar `trident-protect-backup-cr.yaml` el archivo con los valores correctos, aplique el CR:

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

Cree un backup con la interfaz de línea de comandos

Pasos

1. Cree el backup sustituyendo valores entre paréntesis con información de su 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, puede utilizar `--full-backup` el indicador para especificar si una copia de seguridad debe ser no incremental. Por defecto, todos los backups son incrementales. Cuando se utiliza este indicador, la copia de seguridad pasa a ser no incremental. Es mejor realizar backups completos de forma periódica y después realizar backups incrementales entre backups completos para minimizar el riesgo asociado a las restauraciones.

Anotaciones de respaldo admitidas

La siguiente tabla describe las anotaciones que puede utilizar al crear una copia de seguridad de CR:

Anotación	Tipo	Descripción	Valor predeterminado
proteger.trident.netapp.io/copia-de-seguridad-completa	cadena	Especifica si una copia de seguridad debe ser no incremental. Empezar a `true` para crear una copia de seguridad no incremental. La mejor práctica es realizar una copia de seguridad completa periódicamente y luego realizar copias de seguridad incrementales entre las copias de seguridad completas para minimizar el riesgo asociado con las restauraciones.	"falso"
protect.trident.netapp.io/snapshot-complete-timeout	cadena	Tiempo máximo permitido para que se complete la operación de instantánea general.	"60 m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	cadena	Tiempo máximo permitido para que las instantáneas de volumen alcancen el estado listo para usar.	"30 m"
protect.trident.netapp.io/volume-snapshots-created-timeout	cadena	Tiempo máximo permitido para la creación de instantáneas de volumen.	"5 m"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadena	Tiempo máximo (en segundos) que se debe esperar para que cualquier PersistentVolumeClaim (PVC) recién creado llegue al `Bound` fase antes de que fallen las operaciones.	"1200" (20 minutos)

Cree un programa de protección de datos

Una política de protección protege una aplicación mediante la creación de instantáneas, copias de seguridad o ambas según un cronograma definido. Puede elegir crear instantáneas y copias de seguridad cada hora, día,

semana o mes, y puede especificar la cantidad de copias que desea conservar. Puede programar una copia de seguridad completa no incremental mediante la anotación full-backup-rule. De forma predeterminada, todas las copias de seguridad son incrementales. Realizar una copia de seguridad completa periódicamente, junto con copias de seguridad incrementales entre ellas, ayuda a reducir el riesgo asociado con las restauraciones.

- Puede crear programaciones para instantáneas únicamente configurando `backupRetention` a cero y `snapshotRetention` a un valor mayor que cero. Configuración `snapshotRetention` Poner a cero significa que cualquier copia de seguridad programada seguirá creando instantáneas, pero éstas serán temporales y se eliminarán inmediatamente después de que se complete la copia de seguridad.
- Los recursos de ámbito de clúster se incluyen en un backup, una copia de Snapshot o un clon si se hace referencia explícitamente a estos en la definición de la aplicación o si tienen referencias a cualquiera de los espacios de nombres de la aplicación.

Crear un horario mediante un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-schedule-cr.yaml`.
2. En el archivo creado, configure los siguientes atributos:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **SPEC.DATAMOVER**: (*Optional*) Una cadena que indica qué herramienta de copia de seguridad usar para la operación de copia de seguridad. Valores posibles (distingue mayúsculas de minúsculas):
     - `Restic`
     - `Kopia` (predeterminado)
   - **Spec.applicationRef**: El nombre de Kubernetes de la aplicación para realizar una copia de seguridad.
   - **Spec.appVaultRef**: (*required*) El nombre del AppVault donde se debe almacenar el contenido de la copia de seguridad.
   - **spec.backupRetention**: (*Obligatorio*) La cantidad de copias de seguridad a conservar. Cero indica que no se deben crear copias de seguridad (solo instantáneas).
   - **spec.backupReclaimPolicy**: (*Opcional*) Determina qué sucede con una copia de seguridad si el CR de la copia de seguridad se elimina durante su período de retención. Después del período de retención, las copias de seguridad siempre se eliminan. Valores posibles (distingue entre mayúsculas y minúsculas):
     - `Retain` (predeterminado)
     - `Delete`
   - **spec.snapshotRetention**: (*Obligatorio*) La cantidad de instantáneas a conservar. El valor 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 período de retención. Después del período de retención, las instantáneas siempre se eliminan. Valores posibles (distingue entre mayúsculas y minúsculas):
     - `Retain`
     - `Delete`(por defecto)
   - **spec.granularity**: La frecuencia con la que debe ejecutarse el horario. Los posibles valores, junto con los campos asociados necesarios:
     - `Hourly`(requiere que usted especifique `spec.minute`)
     - `Daily`(requiere que usted especifique `spec.minute` y `spec.hour`)
     - `Weekly`(requiere que usted especifique `spec.minute, spec.hour` , y `spec.dayOfWeek`)
     - `Monthly`(requiere que usted especifique `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 de 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 el programa. Este campo es obligatorio si la granularidad está establecida 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 el programa. Este campo es obligatorio si la granularidad está establecida en `Hourly` , `Daily` , `Weekly` , o `Monthly` . El valor debe proporcionarse como una cadena.

Ejemplo de YAML para programación de copias de seguridad y instantáneas:

```
---
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 una programación de solo 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"
```

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

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

Cree una programación con la CLI

Pasos

1. Cree el programa de protección, reemplazando los valores entre paréntesis con información de su entorno. Por ejemplo:

> Puede usar `tridentctl-protect create schedule --help` para ver información de ayuda detallada de 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 proporcionan control adicional sobre su programación:

- **Programación de copia de seguridad completa**: utilice la `--full-backup-rule` bandera para programar copias de seguridad completas no incrementales. Esta bandera sólo funciona con `--granularity Daily`. Valores posibles:
  - `Always`:Cree una copia de seguridad completa todos los días.
  - Días de la semana específicos: especifique uno o más días separados por comas (por ejemplo, `"Monday,Thursday"`). Valores válidos: lunes, martes, miércoles, jueves, viernes, sábado, domingo.

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

- **Horarios solo de instantáneas**: Establecer `--backup-retention 0` y especifique un valor mayor que cero para `--snapshot-retention`.

Anotaciones de programación admitidas

La siguiente tabla describe las anotaciones que puede utilizar al crear una solicitud de cambio de programación:

Anotación	Tipo	Descripción	Valor predeterminado
proteger.trident.netapp.io/regla-de-copia-de-seguridad-completa	cadena	Especifica la regla para programar copias de seguridad completas. Puedes configurarlo para `Always` para realizar copias de seguridad completas y constantes o personalizarlas según sus necesidades. Por ejemplo, si elige granularidad diaria, puede especificar los días de la semana en los que debe realizarse la copia de seguridad completa (por ejemplo, `"Monday,Thursday"`). Los valores válidos para días de la semana son: lunes, martes, miércoles, jueves, viernes, sábado y domingo. Tenga en cuenta que esta anotación solo se puede utilizar con programaciones que tengan `granularity` empezar a `Daily`.	No configurado (todas las copias de seguridad son incrementales)
protect.trident.netapp.io/snapshot-complete-timeout	cadena	Tiempo máximo permitido para que se complete la operación de instantánea general.	"60 m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	cadena	Tiempo máximo permitido para que las instantáneas de volumen alcancen el estado listo para usar.	"30 m"
protect.trident.netapp.io/volume-snapshots-created-timeout	cadena	Tiempo máximo permitido para la creación de instantáneas de volumen.	"5 m"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadena	Tiempo máximo (en segundos) que se debe esperar para que cualquier PersistentVolumeClaim (PVC) recién creado llegue al `Bound` fase antes de que fallen las operaciones.	"1200" (20 minutos)

Eliminar una copia de Snapshot

Elimine las snapshots programadas o bajo demanda que ya no necesite.

Pasos

1. Elimine el CR de instantánea asociado a la instantánea:

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

Eliminar una copia de seguridad

Elimine los backups programados o bajo demanda que ya no necesita.

Asegúrese de que la política de recuperación esté configurada en `Delete` Para eliminar todos los datos de respaldo del almacenamiento de objetos. La configuración predeterminada de la política es `Retain` Para evitar la pérdida accidental de datos. Si la política no se modifica a `Delete` Los datos de respaldo permanecerán en el almacenamiento de objetos y será necesario eliminarlos manualmente.

Pasos

1. Elimine el CR de backup asociado con el backup:

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

Compruebe el estado de una operación de backup

Puede usar la línea de comandos para comprobar el estado de una operación de backup que está en curso, se completa o tiene errores.

Pasos

1. Utilice el siguiente comando para recuperar el estado de la operación de copia de seguridad, sustituyendo los valores entre corchetes por información de su entorno:

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

Permita el backup y la restauración para las operaciones de azure-NetApp-files (ANF)

Si ha instalado Trident Protect, puede habilitar la funcionalidad de copia de seguridad y restauración que ahorra espacio para los backends de almacenamiento que usan la clase de almacenamiento azure-netapp-files y se crearon antes de Trident 24.06. Esta funcionalidad funciona con volúmenes NFSv4 y no consume espacio adicional del grupo de capacidad.

Antes de empezar

Asegúrese de lo siguiente:

- Ha instalado Trident Protect.
- Ha definido una aplicación en Trident Protect. Esta aplicación tendrá una funcionalidad de protección limitada hasta que complete este procedimiento.
- `azure-netapp-files`Seleccionó como clase de almacenamiento predeterminada para el back-end de almacenamiento.

Expanda para obtener pasos de configuración

1. Haga lo siguiente en Trident si el volumen ANF se creó antes de actualizar a Trident 24,10:

   a. Habilite el directorio de instantáneas para cada VP basado en azure-NetApp-files y asociado con la aplicación:

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

   b. Confirme que el directorio de snapshots se haya habilitado para cada VP 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 copia de seguridad regular, que consume temporalmente espacio en el grupo de capacidad durante el proceso de copia de seguridad. En este caso, asegúrese de que haya suficiente espacio disponible en el grupo de capacidad para crear un volumen temporal del tamaño del volumen del que se está realizando la copia de seguridad.

Resultado

La aplicación está lista para realizar copias de seguridad y restaurar mediante Trident Protect. Cada PVC también está disponible para ser utilizado por otras aplicaciones para copias de seguridad y restauraciones.

Restaure las aplicaciones

Restaurar aplicaciones usando Trident Protect

Puede utilizar Trident Protect para restaurar su aplicación desde una instantánea o una copia de seguridad. Restaurar desde una instantánea existente será más rápido al restaurar la aplicación en el mismo clúster.

- Al restaurar una aplicación, todos los ganchos de ejecución configurados para la aplicación se restauran con la aplicación. Si hay un enlace de ejecución posterior a la restauración, se ejecuta automáticamente como parte de la operación de restauración.
- Se permite restaurar desde una copia de seguridad a un espacio de nombres diferente o al original en volúmenes qtree. Sin embargo, no se permite restaurar desde una instantánea a un espacio de nombres diferente o al original en volúmenes qtree.
- Puede utilizar la configuración avanzada para personalizar las operaciones de restauración. Para obtener más información, consulte "Utilice la configuración de restauración avanzada de Trident Protect".

Restauración desde un backup a un espacio de nombres diferente

Cuando restaura una copia de seguridad en un espacio de nombres diferente mediante 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, cree copias de seguridad o instantáneas bajo demanda, o establezca un programa de protección.

- Al restaurar un backup en un espacio de nombres diferente con los recursos existentes, no se alterará ningún recurso que comparta los nombres con los que aparecen en el backup. Para restaurar todos los recursos del backup, elimine y vuelva a crear el espacio de nombres objetivo, o restaure el backup en un nuevo espacio de nombres.
- Al utilizar una CR para restaurar a un nuevo espacio de nombres, debe crear manualmente el espacio de nombres de destino antes de aplicar la CR. Trident Protect crea espacios de nombres automáticamente solo cuando se usa la CLI.

Antes de empezar

Asegúrese de que la caducidad del token de sesión de AWS sea suficiente para las operaciones de restauración de S3 que se ejecuten durante mucho tiempo. Si el token caduca durante la operación de restauración, puede fallar la operación.

- Consulte el "Documentación de la API de AWS" para obtener más información sobre la comprobación de la caducidad del token de sesión actual.
- Consulte el documento para "Documentación de AWS IAM" obtener más información acerca de las credenciales con recursos de AWS.

Al restaurar copias de seguridad utilizando Kopia como transportador de datos, puede especificar opcionalmente anotaciones en la CR o usar la CLI para controlar el comportamiento del almacenamiento temporal utilizado por Kopia. Consulte el "Documentación de KOPIA" Para obtener más información sobre las opciones que puede configurar. Utilice el `tridentctl-protect create --help` Comando para obtener más información sobre cómo especificar anotaciones con la CLI de Trident Protect.

Utilice un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-backup-restore-cr.yaml`.
2. En el archivo creado, configure los siguientes atributos:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.appArchivePath**: La ruta dentro de AppVault donde se almacena el contenido de la copia de seguridad. Puede utilizar el siguiente comando para buscar 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.namespaceMapping**: La asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplace `my-source-namespace` y `my-destination-namespace` con la información de su 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
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Optional*) Si necesita seleccionar solo ciertos recursos de la aplicación para restaurar, agregue filtros que incluyan o excluyan recursos marcados con etiquetas particulares:

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

   - **ResourceFilter.resourceSelectionCriteria**: (Requerido para filtrar) Usar `Include` o `Exclude` incluir o excluir un recurso definido en resourceMatchers. Agregue los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:
     - **ResourceFilter.resourceMatchers**: Una matriz de objetos resourceMatcher. Si define varios elementos en esta matriz, coinciden como una OPERACIÓN OR y los campos dentro de

cada elemento (grupo, tipo, versión) coinciden como una operación AND.

- **ResourceMatchers[].group**: (*Optional*) Grupo del recurso a filtrar.
- **ResourceMatchers[].kind**: (*Optional*) Tipo de recurso a filtrar.
- **ResourceMatchers[].version**: (*Optional*) Versión del recurso que se va a filtrar.
- **ResourceMatchers[].names**: (*Optional*) Nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.
- **ResourceMatchers[].namespaces**: (*Optional*) Espacios de nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.
- **ResourceMatchers[].labelSelectors**: (*Optional*) Cadena de selector de etiquetas en el campo Kubernetes metadata.name del recurso tal como se define en el "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 rellenar `trident-protect-backup-restore-cr.yaml` el archivo con los valores correctos, aplique el CR:

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

Utilice la CLI

Pasos

1. Restaure la copia de seguridad en un espacio de nombres diferente, sustituyendo valores entre paréntesis por información de su entorno. El `namespace-mapping` argumento utiliza espacios de nombres separados por dos puntos para asignar 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> \
-n <application_namespace>
```

Restaure desde un backup al espacio de nombres original

Es posible restaurar un backup en el espacio de nombres original en cualquier momento.

Antes de empezar

Asegúrese de que la caducidad del token de sesión de AWS sea suficiente para las operaciones de restauración de S3 que se ejecuten durante mucho tiempo. Si el token caduca durante la operación de restauración, puede fallar la operación.

- Consulte el "Documentación de la API de AWS" para obtener más información sobre la comprobación de la caducidad del token de sesión actual.
- Consulte el documento para "Documentación de AWS IAM" obtener más información acerca de las credenciales con recursos de AWS.

> Al restaurar copias de seguridad utilizando Kopia como transportador de datos, puede especificar opcionalmente anotaciones en la CR o usar la CLI para controlar el comportamiento del almacenamiento temporal utilizado por Kopia. Consulte el "Documentación de KOPIA" Para obtener más información sobre las opciones que puede configurar. Utilice el `tridentctl-protect create --help` Comando para obtener más información sobre cómo especificar anotaciones con la CLI de Trident Protect.

Utilice un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-backup-ipr-cr.yaml`.
2. En el archivo creado, configure los siguientes atributos:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.appArchivePath**: La ruta dentro de AppVault donde se almacena el contenido de la copia de seguridad. Puede utilizar el siguiente comando para buscar 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. (*Optional*) Si necesita seleccionar solo ciertos recursos de la aplicación para restaurar, agregue filtros que incluyan o excluyan recursos marcados con etiquetas particulares:

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

   - **ResourceFilter.resourceSelectionCriteria**: (Requerido para filtrar) Usar `Include` o `Exclude` incluir o excluir un recurso definido en resourceMatchers. Agregue los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:
     - **ResourceFilter.resourceMatchers**: Una matriz de objetos resourceMatcher. Si define varios elementos en esta matriz, coinciden como una OPERACIÓN OR y los campos dentro de cada elemento (grupo, tipo, versión) coinciden como una operación AND.
       - **ResourceMatchers[].group**: (*Optional*) Grupo del recurso a filtrar.
       - **ResourceMatchers[].kind**: (*Optional*) Tipo de recurso a filtrar.

- **ResourceMatchers[].version**: (*Optional*) Versión del recurso que se va a filtrar.
- **ResourceMatchers[].names**: (*Optional*) Nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.
- **ResourceMatchers[].namespaces**: (*Optional*) Espacios de nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.
- **ResourceMatchers[].labelSelectors**: (*Optional*) Cadena de selector de etiquetas en el campo Kubernetes metadata.name del recurso tal como se define en el "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 rellenar `trident-protect-backup-ipr-cr.yaml` el archivo con los valores correctos, aplique el CR:

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

Utilice la CLI

Pasos

1. Restaure la copia de seguridad en el espacio de nombres original, sustituyendo valores entre paréntesis por información de su entorno. El `backup` argumento 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>
```

Restauración desde un backup en otro clúster

Puede restaurar un backup a otro clúster si hay un problema con el clúster original.

- Al restaurar copias de seguridad utilizando Kopia como transportador de datos, puede especificar opcionalmente anotaciones en la CR o usar la CLI para controlar el comportamiento del almacenamiento temporal utilizado por Kopia. Consulte el "Documentación de KOPIA" Para obtener más información sobre las opciones que puede configurar. Utilice el `tridentctl-protect create --help` Comando para obtener más información sobre cómo especificar anotaciones con la CLI de Trident Protect.
- Al utilizar una CR para restaurar a un nuevo espacio de nombres, debe crear manualmente el espacio de nombres de destino antes de aplicar la CR. Trident Protect crea espacios de nombres automáticamente solo cuando se usa la CLI.

Antes de empezar

Asegúrese 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 de bloque de la misma AppVault que el clúster de origen, en la que se almacena el backup.
- Asegúrese de que su entorno local pueda conectarse al bucket de almacenamiento de objetos definido en el CR de AppVault al ejecutar el proceso. `tridentctl-protect get appvaultcontent` dominio. Si las restricciones de red impiden el acceso, ejecute la CLI de Trident Protect desde dentro de un pod en el clúster de destino.
- Asegúrese de que la caducidad del token de sesión de AWS sea suficiente para las operaciones de restauración que se ejecuten durante mucho tiempo. Si el token caduca durante la operación de restauración, puede fallar la operación.
  - Consulte el "Documentación de la API de AWS" para obtener más información sobre la comprobación de la caducidad del token de sesión actual.
  - Consulte el documento para "Documentación de AWS" obtener más información acerca de las credenciales con recursos de AWS.

Pasos

1. Verifique la disponibilidad de AppVault CR en el clúster de destino mediante el complemento CLI de Trident Protect:

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

   Asegúrese de que el espacio de nombres destinado para la restauración de la aplicación exista en el clúster de destino.

2. Visualice el contenido de las copias de seguridad del AppVault disponible desde el clúster de destino:

```
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 AppVault, incluidos sus clústeres de origen, los nombres de 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. Restaure la aplicación en el clúster de destino mediante el nombre de AppVault y la ruta de archivo:

Utilice un CR

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-backup-restore-cr.yaml`.
2. En el archivo creado, configure los siguientes atributos:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **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. Puede utilizar el siguiente comando para buscar esta ruta:

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

     Si BackupRestore CR no está disponible, puede usar el comando mencionado en el paso 2 para ver el contenido de la copia de seguridad.

   - **spec.namespaceMapping**: La asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplace `my-source-namespace` y `my-destination-namespace` con la información de su 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
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

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

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

Utilice la CLI

1. Utilice el siguiente comando para restaurar la aplicación, sustituyendo valores entre paréntesis por información de su entorno. El argumento de asignación de espacio de nombres utiliza espacios de

nombres separados por dos puntos para asignar espacios de nombres de origen a los espacios de nombres de destino correctos con 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> \
--context <destination_cluster_name> \
-n <application_namespace>
```

Restauración desde una copia snapshot a un espacio de nombres diferente

Puede restaurar datos desde una instantánea utilizando un archivo de recursos personalizados (CR) ya sea en un espacio de nombres diferente o en el espacio de nombres de origen original. Cuando restaura una instantánea a un espacio de nombres diferente mediante 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, cree copias de seguridad o instantáneas bajo demanda, o establezca un programa de protección.

- SnapshotRestore admite el `spec.storageClassMapping` atributo, pero solo cuando las clases de almacenamiento de origen y destino utilizan el mismo backend de almacenamiento. Si intenta restaurar a un `StorageClass` que utiliza un backend de almacenamiento diferente, la operación de restauración fallará.
- Al utilizar una CR para restaurar a un nuevo espacio de nombres, debe crear manualmente el espacio de nombres de destino antes de aplicar la CR. Trident Protect crea espacios de nombres automáticamente solo cuando se usa la CLI.

Antes de empezar

Asegúrese de que la caducidad del token de sesión de AWS sea suficiente para las operaciones de restauración de S3 que se ejecuten durante mucho tiempo. Si el token caduca durante la operación de restauración, puede fallar la operación.

- Consulte el "Documentación de la API de AWS" para obtener más información sobre la comprobación de la caducidad del token de sesión actual.
- Consulte el documento para "Documentación de AWS IAM" obtener más información acerca de las credenciales con recursos de AWS.

Utilice un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-snapshot-restore-cr.yaml`.
2. En el archivo creado, configure los siguientes atributos:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.appVaultRef**: (*required*) El nombre del AppVault donde se almacena el contenido de la instantánea.
   - **Spec.appArchivePath**: La ruta dentro de AppVault donde se almacena el contenido de la instantánea. Puede utilizar el siguiente comando para buscar esta ruta:

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

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

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

3. (*Optional*) Si necesita seleccionar solo ciertos recursos de la aplicación para restaurar, agregue filtros que incluyan o excluyan recursos marcados con etiquetas particulares:

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

   - **ResourceFilter.resourceSelectionCriteria**: (Requerido para filtrar) Usar `Include` o `Exclude` incluir o excluir un recurso definido en resourceMatchers. Agregue los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:
     - **ResourceFilter.resourceMatchers**: Una matriz de objetos resourceMatcher. Si define varios elementos en esta matriz, coinciden como una OPERACIÓN OR y los campos dentro de

cada elemento (grupo, tipo, versión) coinciden como una operación AND.

- **ResourceMatchers[].group**: (*Optional*) Grupo del recurso a filtrar.
- **ResourceMatchers[].kind**: (*Optional*) Tipo de recurso a filtrar.
- **ResourceMatchers[].version**: (*Optional*) Versión del recurso que se va a filtrar.
- **ResourceMatchers[].names**: (*Optional*) Nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.
- **ResourceMatchers[].namespaces**: (*Optional*) Espacios de nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.
- **ResourceMatchers[].labelSelectors**: (*Optional*) Cadena de selector de etiquetas en el campo Kubernetes metadata.name del recurso tal como se define en el "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 rellenar `trident-protect-snapshot-restore-cr.yaml` el archivo con los valores correctos, aplique el CR:

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

Utilice la CLI

Pasos

1. Restaure la instantánea en un espacio de nombres diferente, reemplazando los valores entre paréntesis por información de su entorno.

   - El `snapshot` argumento utiliza un espacio de nombres y un nombre de instantánea en el formato `<namespace>/<name>`.
   - El `namespace-mapping` argumento utiliza espacios de nombres separados por dos puntos para

asignar 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> \
-n <application_namespace>
```

Restauración desde una copia Snapshot al espacio de nombres original

Es posible restaurar una copia de Snapshot en el espacio de nombres original en cualquier momento.

Antes de empezar

Asegúrese de que la caducidad del token de sesión de AWS sea suficiente para las operaciones de restauración de S3 que se ejecuten durante mucho tiempo. Si el token caduca durante la operación de restauración, puede fallar la operación.

- Consulte el "Documentación de la API de AWS" para obtener más información sobre la comprobación de la caducidad del token de sesión actual.
- Consulte el documento para "Documentación de AWS IAM" obtener más información acerca de las credenciales con recursos de AWS.

Utilice un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-snapshot-ipr-cr.yaml`.
2. En el archivo creado, configure los siguientes atributos:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.appVaultRef**: (*required*) El nombre del AppVault donde se almacena el contenido de la instantánea.
   - **Spec.appArchivePath**: La ruta dentro de AppVault donde se almacena el contenido de la instantánea. Puede utilizar el siguiente comando para buscar 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
```

3. (*Optional*) Si necesita seleccionar solo ciertos recursos de la aplicación para restaurar, agregue filtros que incluyan o excluyan recursos marcados con etiquetas particulares:

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

   - **ResourceFilter.resourceSelectionCriteria**: (Requerido para filtrar) Usar `Include` o `Exclude` incluir o excluir un recurso definido en resourceMatchers. Agregue los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:
     - **ResourceFilter.resourceMatchers**: Una matriz de objetos resourceMatcher. Si define varios elementos en esta matriz, coinciden como una OPERACIÓN OR y los campos dentro de cada elemento (grupo, tipo, versión) coinciden como una operación AND.
       - **ResourceMatchers[].group**: (*Optional*) Grupo del recurso a filtrar.
       - **ResourceMatchers[].kind**: (*Optional*) Tipo de recurso a filtrar.
       - **ResourceMatchers[].version**: (*Optional*) Versión del recurso que se va a filtrar.
       - **ResourceMatchers[].names**: (*Optional*) Nombres en el campo Kubernetes

metadata.name del recurso que se va a filtrar.

- **ResourceMatchers[].namespaces**: (*Optional*) Espacios de nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.
- **ResourceMatchers[].labelSelectors**: (*Optional*) Cadena de selector de etiquetas en el campo Kubernetes metadata.name del recurso tal como se define en el "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 rellenar `trident-protect-snapshot-ipr-cr.yaml` el archivo con los valores correctos, aplique el CR:

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

Utilice la CLI

Pasos

1. Restaure la instantánea en el espacio de nombres original, reemplazando los valores entre paréntesis por información de su entorno. Por ejemplo:

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

Compruebe el estado de una operación de restauración

Puede usar la línea de comandos para comprobar el estado de una operación de restauración en curso, que se completó o con errores.

Pasos

1. Utilice el siguiente comando para recuperar el estado de la operación de restauración, sustituyendo valores de entre corchetes con información de su entorno:

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

Utilice la configuración de restauración avanzada de Trident Protect

Puede personalizar las operaciones de restauración utilizando configuraciones avanzadas, como anotaciones, configuraciones de espacios de nombres y opciones de almacenamiento para satisfacer sus requisitos específicos.

Etiquetas y anotaciones del espacio de nombres durante las operaciones de restauración y conmutación al nodo de respaldo

Durante las operaciones de restauración y conmutación al nodo de respaldo, se realizan etiquetas y anotaciones en el espacio de nombres de destino que coincidan con las etiquetas y anotaciones en el espacio de nombres de origen. Se añaden etiquetas o anotaciones del espacio de nombres origen que no existen en el espacio de nombres destino, y las etiquetas o anotaciones que ya existan se sobrescriben para que coincidan con el valor del espacio de nombres origen. Las etiquetas o anotaciones que sólo existen en el espacio de nombres de destino permanecen sin cambios.

> Si utiliza Red Hat OpenShift, es importante tener en cuenta el papel fundamental que desempeñan las anotaciones de espacios de nombres en los entornos OpenShift. Las anotaciones de espacio de nombres garantizan que los pods restaurados cumplan con los permisos y las configuraciones de seguridad adecuados 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, consulte la"Documentación de restricciones de contexto de seguridad de OpenShift" .

Puede evitar que se sobrescriban anotaciones específicas en el espacio de nombres de destino mediante el establecimiento de la variable de entorno de Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de llevar a cabo la operación de restauración o 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, se tendrán en cuenta las anotaciones y etiquetas de espacio de nombres especificadas en `restoreSkipNamespaceAnnotations` y `restoreSkipNamespaceLabels` quedan excluidos de la operación de restauración o conmutación por error. Asegúrese de que estos ajustes se configuren durante la instalación inicial de Helm. Para obtener más información, consulte "Configurar ajustes adicionales del gráfico de timón Trident Protect".

Si instalaste la aplicación de origen usando Helm con el `--create-namespace` bandera, se le da un trato especial a la `name` Clave de etiqueta. Durante el proceso de restauración o conmutación por error, Trident Protect copia esta etiqueta en el 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 destino, cada uno con anotaciones y etiquetas diferentes. Puede ver el estado del espacio de nombres de destino antes y después de la operación, así como cómo las anotaciones y etiquetas se combinan o sobrescriben en el espacio de nombres de destino.

Antes de la operación de restauración o conmutación por error

En la siguiente tabla se muestra el estado del ejemplo de espacios de nombres de origen y destino antes de la operación de restauración o conmutación por error:

Espacio de nombres	Anotaciones	Etiquetas
Espacio de nombres ns-1 (origen)	• anotación.uno/clave: “updatedvalue” • anotación.dos/clave: “verdadero”	• entorno=producción • cumplimiento=hipaa • name=ns-1
Espacio de nombres ns-2 (destino)	• anotación.uno/tecla: “verdadero” • anotación.tres/clave: “falso”	• role=base de datos

Después de la operación de restauración

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

Espacio de nombres	Anotaciones	Etiquetas
Espacio de nombres ns-2 (destino)	• anotación.uno/clave: “updatedvalue” • anotación.dos/clave: “verdadero” • anotación.tres/clave: “falso”	• name=ns-2 • cumplimiento=hipaa • entorno=producción • role=base de datos

Campos admitidos

Esta sección describe campos adicionales disponibles para operaciones de restauración.

Mapeo de clases de almacenamiento

El `spec.storageClassMapping` El atributo 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. Puede usar esto al migrar aplicaciones entre clústeres con diferentes clases de almacenamiento o al cambiar el backend de almacenamiento para operaciones de BackupRestore.

Ejemplo:

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

Anotaciones admitidas

Esta sección enumera las anotaciones compatibles para configurar diversos comportamientos en el sistema. Si el usuario no configura una anotación explícitamente, el sistema utilizará el valor predeterminado.

Anotación	Tipo	Descripción	Valor predeterminado
protect.trident.netapp.io/tiempo-de-espera-del-transportador-de-datos-sec	cadena	El tiempo máximo (en segundos) permitido para que la operación de transferencia de datos permanezca detenida.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	cadena	El límite de tamaño máximo (en megabytes) para el caché de contenido de Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadena	Tiempo máximo (en segundos) que se debe esperar para que cualquier PersistentVolumeClaim (PVC) recién creado llegue al `Bound` fase antes de que fallen las operaciones. Se aplica a todos los tipos de restauración de CR (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Utilice un valor más alto si su backend de almacenamiento o clúster requiere con frecuencia más tiempo.	"1200" (20 minutos)

Replicar aplicaciones utilizando NetApp SnapMirror y Trident Protect

Con Trident Protect, puede utilizar las capacidades de replicación asincrónica de la tecnología SnapMirror de NetApp para replicar cambios de datos y aplicaciones de un backend de almacenamiento a otro, en el mismo clúster o entre diferentes clústeres.

Etiquetas y anotaciones del espacio de nombres durante las operaciones de restauración y conmutación al nodo de respaldo

Durante las operaciones de restauración y conmutación al nodo de respaldo, se realizan etiquetas y anotaciones en el espacio de nombres de destino que coincidan con las etiquetas y anotaciones en el espacio de nombres de origen. Se añaden etiquetas o anotaciones del espacio de nombres origen que no existen en el espacio de nombres destino, y las etiquetas o anotaciones que ya existan se sobrescriben para que coincidan con el valor del espacio de nombres origen. Las etiquetas o anotaciones que sólo existen en el espacio de nombres de destino permanecen sin cambios.

> Si utiliza Red Hat OpenShift, es importante tener en cuenta el papel fundamental que desempeñan las anotaciones de espacios de nombres en los entornos OpenShift. Las anotaciones de espacio de nombres garantizan que los pods restaurados cumplan con los permisos y las configuraciones de seguridad adecuados 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, consulte la"Documentación de restricciones de contexto de seguridad de OpenShift" .

Puede evitar que se sobrescriban anotaciones específicas en el espacio de nombres de destino mediante el establecimiento de la variable de entorno de Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de llevar a cabo la operación de restauración o 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, se tendrán en cuenta las anotaciones y etiquetas de espacio de nombres especificadas en `restoreSkipNamespaceAnnotations` y `restoreSkipNamespaceLabels` quedan excluidos de la operación de restauración o conmutación por error. Asegúrese de que estos ajustes se configuren durante la instalación inicial de Helm. Para obtener más información, consulte "Configurar ajustes adicionales del gráfico de timón Trident Protect".

Si instalaste la aplicación de origen usando Helm con el `--create-namespace` bandera, se le da un trato especial a la `name` Clave de etiqueta. Durante el proceso de restauración o conmutación por error, Trident Protect copia esta etiqueta en el 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 destino, cada uno con anotaciones y etiquetas diferentes. Puede ver el estado del espacio de nombres de destino antes y después de la operación, así como cómo las anotaciones y etiquetas se combinan o sobrescriben en el espacio de nombres de destino.

Antes de la operación de restauración o conmutación por error

En la siguiente tabla se muestra el estado del ejemplo de espacios de nombres de origen y destino antes de la operación de restauración o conmutación por error:

Espacio de nombres	Anotaciones	Etiquetas
Espacio de nombres ns-1 (origen)	• anotación.uno/clave: “updatedvalue” • anotación.dos/clave: “verdadero”	• entorno=producción • cumplimiento=hipaa • name=ns-1
Espacio de nombres ns-2 (destino)	• anotación.uno/tecla: “verdadero” • anotación.tres/clave: “falso”	• role=base de datos

Después de la operación de restauración

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

Espacio de nombres	Anotaciones	Etiquetas
Espacio de nombres ns-2 (destino)	• anotación.uno/clave: “updatedvalue” • anotación.dos/clave: “verdadero” • anotación.tres/clave: “falso”	• name=ns-2 • cumplimiento=hipaa • entorno=producción • role=base de datos

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

Ganchos de ejecución durante operaciones de conmutación por error e inversas

Al usar la relación AppMirror para proteger su aplicación, hay comportamientos específicos relacionados con los ganchos de ejecución que debe tener en cuenta durante las operaciones de conmutación por error y de reversión.

- Durante la conmutación por error, los ganchos de ejecución se copian automáticamente del clúster de origen al de destino. No es necesario volver a crearlos manualmente. Tras la conmutación por error, los ganchos de ejecución permanecen en la aplicación y se ejecutarán durante cualquier acción relevante.

- Durante la resincronización inversa, se eliminan todos los ganchos de ejecución existentes en la aplicación. Cuando la aplicación de origen se convierte en la aplicación de destino, estos ganchos de ejecución dejan de ser válidos y se eliminan para impedir su ejecución.

Para obtener más información sobre los ganchos de ejecución, consulte"Administrar los ganchos de ejecución de Trident Protect" .

Configurar una relación de replicación

La configuración de una relación de replicación implica lo siguiente:

- Elegir la frecuencia con la que desea que Trident Protect tome una instantánea de la aplicación (que incluye los recursos de Kubernetes de la aplicación, así como las instantáneas de volumen para cada uno de los volúmenes de la aplicación)
- Elegir el programa de replicación (incluye recursos de Kubernetes, así como datos de volumen persistentes)
- Establecer la hora para que se realice la snapshot

Pasos

1. En el clúster de origen, cree un AppVault para la aplicación de origen. Según su proveedor de almacenamiento, modifique un ejemplo en "Recursos personalizados de AppVault" Para adaptarse a su entorno:

Cree un AppVault con un CR

a. Cree el archivo de recursos personalizados (CR) y asígnele un nombre (por ejemplo, `trident-protect-appvault-primary-source.yaml`).

b. Configure los siguientes atributos:

- **metadata.name**: (*required*) El nombre del recurso personalizado de AppVault. Tome nota del nombre que elija, ya que otros archivos CR necesarios para una relación de replicación hacen referencia a este valor.
- **spec.providerConfig**: (*required*) almacena la configuración necesaria para acceder a AppVault utilizando el proveedor especificado. Elija un nombre de bucketName y cualquier otro detalle necesario para su proveedor. Tome nota de los valores que elija, ya que otros archivos CR necesarios para una relación de replicación hacen referencia a estos valores. Consulte "Recursos personalizados de AppVault"para ver ejemplos de CRS de AppVault con otros proveedores.
- **spec.providerCredentials**: (*required*) 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 provenir de un secreto.
    - **KEY**: (*REQUIRED*) La clave válida del secreto para seleccionar.
    - **Name**: (*required*) Nombre del secreto que contiene el valor para este campo. Debe estar en el mismo espacio de nombres.
  - **spec.providerCredentials.secretAccessKey**: (*required*) La clave de acceso utilizada para acceder al proveedor. El **name** debe coincidir con **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*required*) determina qué proporciona la copia de seguridad; por ejemplo, NetApp ONTAP S3, S3 genérico, Google Cloud o Microsoft Azure. Los posibles valores son los siguientes:
  - aws
  - azure
  - gcp
  - genérico-s3
  - ONTAP-s3
  - StorageGRID-s3

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

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

Cree un AppVault con la CLI

a. Cree AppVault, reemplazando los valores entre paréntesis con información de su 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, cree la aplicación de origen CR:

Cree la aplicación de origen mediante un CR

a. Cree el archivo de recursos personalizados (CR) y asígnele un nombre (por ejemplo, `trident-protect-app-source.yaml`).

b. Configure los siguientes atributos:

- **metadata.name**: (*required*) El nombre del recurso personalizado de la aplicación. Tome nota del nombre que elija, ya que otros archivos CR necesarios para una relación de replicación hacen referencia a este valor.
- **spec.includedNamespaces**: (*required*) Una matriz de espacios de nombres y etiquetas asociadas. Utilice nombres de espacio de nombres y, opcionalmente, reduzca el ámbito de los espacios de nombres con etiquetas para especificar los recursos que existen en los espacios de nombres que se muestran aquí. El espacio de nombres de la aplicación debe formar parte de esta cabina.

  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 rellenar `trident-protect-app-source.yaml` el archivo con los valores correctos, aplique el CR:

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

Cree la aplicación de origen mediante la CLI

a. Cree la aplicación de origen. 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, tome una instantánea de la aplicación de origen. Esta instantánea se utiliza como base para la aplicación en el clúster de destino. Si omite este paso, deberá esperar a que se ejecute la siguiente instantánea programada para tener una instantánea reciente. Para crear una instantánea bajo demanda, consulte "Crear una snapshot bajo demanda".

4. En el clúster de origen, cree 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 los 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 agiliza y optimiza el proceso, ya que solo se transfieren los cambios realizados 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 programas adicionales.

Cree la programación de replicación utilizando un CR.

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

i. Cree el archivo de recursos personalizados (CR) y asígnele un nombre (por ejemplo, `trident-protect-schedule.yaml`).

ii. Configure los siguientes atributos:

- **metadata.name**: (*required*) El nombre del recurso personalizado de horario.
- **spec.appVaultRef**: (*Obligatorio*) Este valor debe coincidir con el campo metadata.name del 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**: (*required*) Este campo es obligatorio y el valor debe establecerse en 0.
- **Spec.enabled**: Debe establecerse en true.
- **spec.granularity**: debe ser establecido en `Custom`.
- **Spec.recurrenceRule**: Define una fecha de inicio en la hora UTC y un intervalo de recurrencia.
- **Spec.snapshotRetention**: Debe establecerse en 2.

Ejemplo 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 rellenar `trident-protect-schedule.yaml` el archivo con los valores correctos, aplique el CR:

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

Cree la programación de replicación mediante la CLI.

a. Cree la planificación de replicación, reemplazando los valores entre corchetes con información de su 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, cree una aplicación de origen AppVault CR idéntica a la aplicación AppVault CR que aplicó en el clúster de origen y asígnele el nombre (por ejemplo, `trident-protect-appvault-primary-destination.yaml`).
6. Aplicar el CR:

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

7. Cree un AppVault CR de destino para la aplicación de destino en el clúster de destino. Según su proveedor de almacenamiento, modifique un ejemplo en "Recursos personalizados de AppVault" Para adaptarse a su entorno:
   a. Cree el archivo de recursos personalizados (CR) y asígnele un nombre (por ejemplo, `trident-protect-appvault-secondary-destination.yaml`).
   b. Configure los siguientes atributos:
      - **metadata.name**: (*required*) El nombre del recurso personalizado de AppVault. Tome nota del nombre que elija, ya que otros archivos CR necesarios para una relación de replicación hacen referencia a este valor.
      - **spec.providerConfig**: (*required*) almacena la configuración necesaria para acceder a AppVault utilizando el proveedor especificado. Elija un `bucketName` y cualquier otro detalle necesario para su proveedor. Tome nota de los valores que elija, ya que otros archivos CR necesarios para una relación de replicación hacen referencia a estos valores. Consulte "Recursos personalizados de

AppVault"para ver ejemplos de CRS de AppVault con otros proveedores.

- **spec.providerCredentials**: (*required*) 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 provenir de un secreto.
    - **KEY**: (*REQUIRED*) La clave válida del secreto para seleccionar.
    - **Name**: (*required*) Nombre del secreto que contiene el valor para este campo. Debe estar en el mismo espacio de nombres.
  - **spec.providerCredentials.secretAccessKey**: (*required*) La clave de acceso utilizada para acceder al proveedor. El **name** debe coincidir con **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*required*) determina qué proporciona la copia de seguridad; por ejemplo, NetApp ONTAP S3, S3 genérico, Google Cloud o Microsoft Azure. Los posibles valores son los siguientes:
  - aws
  - azure
  - gcp
  - genérico-s3
  - ONTAP-s3
  - StorageGRID-s3

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

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

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

> Al utilizar una CR, cree manualmente el espacio de nombres de destino antes de aplicar la CR. Trident Protect crea espacios de nombres automáticamente solo cuando se usa la CLI.

Cree una AppMirrorRelationship con un CR

a. Cree el archivo de recursos personalizados (CR) y asígnele un nombre (por ejemplo, `trident-protect-relationship.yaml`).

b. Configure los siguientes atributos:

- **metadata.name:** (requerido) El nombre del recurso personalizado AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*required*) Este valor debe coincidir con el nombre de AppVault para la aplicación de destino en el clúster de destino.
- **spec.namespaceMapping**: (*required*) Los espacios de nombres de destino y origen deben coincidir con el espacio de nombres de aplicación definido en la aplicación CR respectiva.
- **Spec.sourceAppVaultRef**: (*required*) Este valor debe coincidir con el nombre de AppVault para la aplicación de origen.
- **Spec.sourceApplicationName**: (*required*) Este valor debe coincidir con el nombre de la aplicación de origen que definió en la aplicación de origen CR.
- **spec.sourceApplicationUID**: (Obligatorio) Este valor debe coincidir con el UID de la aplicación de origen que definió en el CR de la aplicación de origen.
- **spec.storageClassName**: (*Opcional*) Elija 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 utilizará por defecto la clase de almacenamiento predeterminada del clúster.
- **Spec.recurrenceRule**: Define una fecha de inicio en la hora UTC y un intervalo de recurrencia.

  Ejemplo 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 rellenar `trident-protect-relationship.yaml` el archivo con los valores correctos, aplique el CR:

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

Cree una AppMirrorRelationship con 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> -n <application_namespace>
```

Ejemplo:

```
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. (*Optional*) En el clúster de destino, compruebe el estado y el estado de la relación de replicación:

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

Conmutación por error al clúster de destino

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

Pasos

1. En el clúster de destino, edite el archivo CR de AppMirrorRelationship (por ejemplo, `trident-protect-relationship.yaml`) y cambie el valor de **spec.desiredState** a `Promoted`.
2. Guarde el archivo CR.
3. Aplicar el CR:

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

4. (*Optional*) Cree cualquier programación de protección que necesite en la aplicación con fallos.
5. (*Optional*) Compruebe el estado y el estado de la relación de replicación:

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

Resincronizar una relación de replicación con fallo

La operación de resincronización vuelve a establecer la relación de replicación. Después de realizar una operación de resincronización, la aplicación de origen original se convierte en la aplicación en ejecución y se descartan todos los cambios realizados en la aplicación en ejecución en el clúster de destino.

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

Se perderán todos los datos escritos en la aplicación de destino durante la conmutación al respaldo.

Pasos

1. Opcional: En el clúster de origen, cree una copia Snapshot de la aplicación de origen. De esta forma se garantiza que se capturen los cambios más recientes del clúster de origen.
2. En el clúster de destino, edite el archivo CR de AppMirrorRelationship (por ejemplo, `trident-protect-relationship.yaml`) y cambie el valor de spec.desiredState a `Established`.
3. Guarde el archivo CR.
4. Aplicar el CR:

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

5. Si ha creado cualquier programación de protección en el clúster de destino para proteger la aplicación con errores, elimínela. Cualquier programación que permanezca provoca errores de snapshots de volumen.

Resincronización inversa de una relación de replicación fallida

Cuando se realiza una resincronización inversa de una relación de replicación fallida, la aplicación de destino se convierte en la aplicación de origen y el origen se convierte en el destino. Se mantienen los cambios realizados en la aplicación de destino durante la conmutación por error.

Pasos

1. En el clúster de destino original, elimine el CR de AppMirrorRelationship. Esto hace que el destino se convierta en el origen. Si queda alguna programación de protección en el nuevo clúster de destino, elimínela.
2. Configure una relación de replicación aplicando los archivos CR que utilizó originalmente para configurar la relación con los clusters opuestos.
3. Asegúrese de que el nuevo destino (cluster de origen original) está configurado con los CRS de AppVault.
4. Configure una relación de replicación en el cluster opuesto, configurando valores para la dirección inversa.

Invertir dirección de replicación de aplicaciones

Cuando invierte la dirección de replicación, Trident Protect mueve la aplicación al backend de almacenamiento de destino mientras continúa replicando al backend de almacenamiento de origen original. Trident Protect detiene la aplicación de origen y replica los datos en el destino antes de conmutar por error a la aplicación de destino.

En esta situación, está intercambiando el origen y el destino.

Pasos

1. En el clúster de origen, cree una snapshot de apagado:

Cree una instantánea de cierre con un CR

a. Desactive las programaciones de políticas de protección para la aplicación de origen.

b. Crear un archivo CR de ShutdownSnapshot:

i. Cree el archivo de recursos personalizados (CR) y asígnele un nombre (por ejemplo, `trident-protect-shutdownsnapshot.yaml`).

ii. Configure los siguientes atributos:

- **metadata.name**: (*required*) 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 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 rellenar `trident-protect-shutdownsnapshot.yaml` el archivo con los valores correctos, aplique el CR:

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

Cree una snapshot apagada con la CLI

a. Cree la instantánea de cierre, reemplazando valores entre paréntesis con información de su 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, cuando se complete la snapshot de apagado, obtenga el estado de la snapshot de apagado:

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. En el clúster de origen, busque el valor de **shutdownsnapshot.status.appArchivePath** usando el siguiente comando, y registre la última parte de la ruta del archivo (también llamada nombre base; esto será todo después de la última barra diagonal):

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

4. Realice 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, incluya el `spec.promotedSnapshot` campo en el archivo AppMirrorRelationship CR y establezca su valor en el nombre base que registró en el paso 3 anterior.

5. Realice los pasos de resincronización inversa en Resincronización inversa de una relación de replicación fallida.
6. Habilite las programaciones de protección en el nuevo clúster de origen.

Resultado

Las siguientes acciones se producen debido a la replicación inversa:

- Se toma una instantánea de los recursos de Kubernetes de la aplicación de origen original.
- Los pods de la aplicación de origen originales se detienen con dignidad al eliminar los recursos de Kubernetes de la aplicación (dejando las RVP y los VP en funcionamiento).
- Después de que los pods se cierran, se toman y replican instantáneas de los volúmenes de la aplicación.
- Las relaciones de SnapMirror se rompen, lo que hace que los volúmenes de destino estén listos para la lectura/escritura.
- Los recursos de Kubernetes de la aplicación se restauran a partir de la instantánea previa al cierre, utilizando los datos del volumen replicados después de que se cerró la aplicación de origen original.
- La replicación se restablece en la dirección inversa.

Conmutación tras error de las aplicaciones al clúster de origen original

Al utilizar Trident Protect, puede lograr una "conmutación por error" después de una operación de conmutación por error mediante la siguiente secuencia de operaciones. En este flujo de trabajo para restaurar la dirección de replicación original, Trident Protect replica (resincroniza) cualquier cambio de aplicación a la aplicación de origen original antes de revertir la dirección de replicación.

Este proceso se inicia desde una relación que ha completado una conmutación al nodo de respaldo a un destino e implica los siguientes pasos:

- Comience con un estado de conmutación al respaldo.
- Vuelva a sincronizar la relación de replicación.

> No realice una operación de resincronización normal, ya que esto descartará los datos escritos en el clúster de destino durante el procedimiento de conmutación por error.

- Invierta la dirección de replicación.

Pasos

1. Realice Resincronización inversa de una relación de replicación fallidalos pasos.
2. Realice Invertir dirección de replicación de aplicacioneslos pasos.

Eliminar una relación de replicación

Puede eliminar una relación de replicación en cualquier momento. Al eliminar la relación de replicación de la aplicación, se crean dos aplicaciones independientes sin relación entre ellas.

Pasos

1. En el clúster de eliminación actual, elimine el CR de AppMirrorRelationship:

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

Migrar aplicaciones usando Trident Protect

Puede migrar sus aplicaciones entre clústeres o a diferentes clases de almacenamiento restaurando datos de respaldo.

> Al migrar una aplicación, todos los ganchos de ejecución configurados para la aplicación se migran con la aplicación. Si hay un enlace 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 para las siguientes situaciones, se pueden automatizar tareas de backup y restauración específicas.

Clone en el mismo clúster

Para clonar una aplicación en el mismo clúster, cree una copia Snapshot o un backup y restaure los datos en el mismo clúster.

Pasos

1. Debe realizar una de las siguientes acciones:
   a. "Crear una copia de Snapshot".
   b. "Cree un backup".
2. En el mismo clúster, realice una de las siguientes acciones, según si se ha creado una snapshot o un backup:

a. "Restaure los datos desde la copia Snapshot".

b. "Restaure los datos del backup".

Clone en otro clúster

Para clonar una aplicación en un clúster diferente (realizar una clonación entre clústeres), cree una copia de seguridad en el clúster de origen y, a continuación, restaure la copia de seguridad en un clúster diferente. Asegúrese de que Trident Protect esté instalado en el clúster de destino.

Puede replicar una aplicación entre diferentes clusters mediante "Replicación de SnapMirror".

Pasos

1. "Cree un backup".
2. Asegúrese de que el CR de AppVault para el depósito de almacenamiento de objetos que contiene la copia de seguridad se haya configurado en el clúster de destino.
3. En el clúster de destino, "restaure los datos del backup".

Migre aplicaciones de una clase de almacenamiento a otra clase de almacenamiento

Puede migrar aplicaciones de una clase de almacenamiento a una clase de almacenamiento diferente restaurando una copia de seguridad en la clase de almacenamiento de destino.

Por ejemplo (excluyendo los secretos de la CR de restauración):

```
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
```

Restaure la instantánea con un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-snapshot-restore-cr.yaml`.
2. En el archivo creado, configure los siguientes atributos:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.appArchivePath**: La ruta dentro de AppVault donde se almacena el contenido de la instantánea. Puede utilizar el siguiente comando para buscar esta ruta:

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

   - **Spec.appVaultRef**: (*required*) El nombre del AppVault donde se almacena el contenido de la instantánea.
   - **spec.namespaceMapping**: La asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplace `my-source-namespace` y `my-destination-namespace` con la información de su 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 necesita seleccionar solo ciertos recursos de la aplicación para restaurar, agregue filtros que incluyan o excluyan recursos marcados con etiquetas concretas:
   - **ResourceFilter.resourceSelectionCriteria**: (Requerido para filtrar) Usa `include or exclude` para incluir o excluir un recurso definido en resourceMatchers. Agregue los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:
     - **ResourceFilter.resourceMatchers**: Una matriz de objetos resourceMatcher. Si define varios elementos en esta matriz, coinciden como una OPERACIÓN OR y los campos dentro de cada elemento (grupo, tipo, versión) coinciden como una operación AND.
       - **ResourceMatchers[].group**: (*Optional*) Grupo del recurso a filtrar.
       - **ResourceMatchers[].kind**: (*Optional*) Tipo de recurso a filtrar.
       - **ResourceMatchers[].version**: (*Optional*) Versión del recurso que se va a filtrar.

- **ResourceMatchers[].names**: (*Optional*) Nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.
- **ResourceMatchers[].namespaces**: (*Optional*) Espacios de nombres en el campo Kubernetes metadata.name del recurso que se va a filtrar.
- **ResourceMatchers[].labelSelectors**: (*Optional*) Cadena de selector de etiquetas en el campo Kubernetes metadata.name del recurso tal como se define en el "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 rellenar `trident-protect-snapshot-restore-cr.yaml` el archivo con los valores correctos, aplique el CR:

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

Restaure la instantánea mediante la interfaz de línea de comandos

Pasos

1. Restaure la instantánea en un espacio de nombres diferente, reemplazando los valores entre paréntesis por información de su entorno.

   - El `snapshot` argumento utiliza un espacio de nombres y un nombre de instantánea en el formato `<namespace>/<name>`.
   - El `namespace-mapping` argumento utiliza espacios de nombres separados por dos puntos para asignar 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>
```

Administrar los ganchos de ejecución de Trident Protect

Un enlace de ejecución es una acción personalizada que puede configurar para que se ejecute junto con una operación de protección de datos de una aplicación gestionada. Por ejemplo, si dispone de una aplicación de base de datos, puede utilizar un enlace de ejecución para pausar todas las transacciones de la base de datos antes de realizar una instantánea y reanudar las transacciones una vez completada la instantánea. De este modo se garantiza la creación de instantáneas coherentes con la aplicación.

Tipos de enlaces de ejecución

Trident Protect admite los siguientes tipos de ganchos de ejecución, según cuándo se puedan ejecutar:

- Copia previa de Snapshot
- Possnapshot
- Previo al backup
- Después del backup
- Después de la restauración
- Después de la conmutación al respaldo

Orden de ejecución

Cuando se ejecuta una operación de protección de datos, los eventos de enlace de ejecución tienen lugar en el siguiente orden:

1. Los ganchos de ejecución de preoperación personalizados aplicables se ejecutan en los contenedores adecuados. Puede crear y ejecutar tantos ganchos de prefuncionamiento personalizados como necesite, pero el orden de ejecución de estos enlaces antes de la operación no está garantizado ni configurable.
2. Si corresponde, se producen bloqueos del sistema de archivos. "Obtenga más información 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 corresponde.
5. Los enlaces de ejecución de post-operación personalizados aplicables se ejecutan en los contenedores adecuados. Puede crear y ejecutar tantos enlaces de post-operación personalizados como necesite, pero el orden de ejecución de estos enlaces después de la operación no está garantizado ni configurable.

Si crea varios enlaces de ejecución del mismo tipo (por ejemplo, presnapshot), no se garantiza el orden de ejecución de esos enlaces. Sin embargo, el orden de ejecución de ganchos de diferentes tipos está garantizado. Por ejemplo, el siguiente es el orden de ejecución de una configuración que tiene todos los diferentes tipos de ganchos:

1. Ganchos presnapshot ejecutados
2. Ganchos posteriores a la instantánea ejecutados
3. Ganchos de precopia de seguridad ejecutados
4. Se han ejecutado los enlaces posteriores a la copia de seguridad

El ejemplo de orden anterior solo se aplica cuando se ejecuta un backup que no utiliza una snapshot existente.

Siempre debe probar sus secuencias de comandos de ejecución de enlace antes de habilitarlas en un entorno de producción. Puede utilizar el comando 'kubectl exec' para probar cómodamente los scripts. Después de habilitar los enlaces de ejecución en un entorno de producción, pruebe las copias Snapshot y backups resultantes para garantizar que sean coherentes. Para ello, puede clonar la aplicación en un espacio de nombres temporal, restaurar la instantánea o la copia de seguridad y, a continuación, probar la aplicación.

Si un gancho de ejecución previo a la instantánea agrega, cambia o elimina recursos de Kubernetes, esos cambios se incluyen en la instantánea o la copia de seguridad y en cualquier operación de restauración posterior.

Notas importantes sobre los enlaces de ejecución personalizados

Tenga en cuenta lo siguiente al planificar enlaces de ejecución para sus aplicaciones.

- Un enlace de ejecución debe utilizar una secuencia de comandos para realizar acciones. Muchos enlaces de ejecución pueden hacer referencia al mismo script.
- Trident Protect requiere que los scripts que utilizan los ganchos de ejecución estén escritos en el formato de scripts de shell ejecutables.
- El tamaño del script está limitado a 96 KB.
- Trident Protect utiliza configuraciones de gancho de ejecución y cualquier criterio coincidente para determinar qué ganchos son aplicables a una operación de instantánea, copia de seguridad o restauración.

Debido a que los enlaces de ejecución a menudo reducen o desactivan por completo la funcionalidad de la aplicación con la que se ejecutan, siempre debe intentar minimizar el tiempo que tardan los enlaces de ejecución personalizados. Si inicia una operación de copia de seguridad o de instantánea con los enlaces de ejecución asociados pero, a continuación, la cancela, los ganchos pueden ejecutarse si ya se ha iniciado la operación de copia de seguridad o de Snapshot. Esto significa que la lógica utilizada en un enlace de ejecución posterior a la copia de seguridad no puede suponer que la copia de seguridad se ha completado.

Filtros de gancho de ejecución

Al agregar o editar un enlace de ejecución para una aplicación, puede agregar filtros al enlace de ejecución para gestionar qué contenedores coincidirá el enlace. 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 le permiten crear escenarios donde los enlaces de ejecución se ejecutan en algunos, pero no necesariamente todos los contenedores idénticos. Si crea varios filtros para un único enlace de ejecución, se combinan con un operador y lógico. Puede tener hasta 10 filtros activos por gancho de ejecución.

Cada filtro que agrega a un gancho de ejecución utiliza una expresión regular para que coincida con los contenedores en su clúster. Cuando un gancho coincide con un contenedor, el gancho ejecutará su script asociado en ese contenedor. Las expresiones regulares para filtros utilizan la sintaxis de Expresión regular 2 (RE2), que no admite la creación de un filtro que excluya contenedores de la lista de coincidencias. Para obtener información sobre la sintaxis que Trident Protect admite para expresiones regulares en filtros de gancho de ejecución, consulte "Soporte de sintaxis de expresión regular 2 (RE2)" .

> Si se agrega un filtro de espacio de nombres a un enlace de ejecución que se ejecuta después de una operación de restauración o clonado y el origen y destino de la restauración o clonado se encuentran en diferentes espacios de nombres, el filtro de espacio de nombres solo se aplica al espacio de nombres de destino.

Ejemplos de gancho de ejecución

Visite el "Proyecto Verda GitHub de NetApp" para descargar ganchos de ejecución reales para aplicaciones populares como Apache Cassandra y Elasticsearch. También puede ver ejemplos y obtener ideas para estructurar sus propios enlaces de ejecución personalizados.

Cree un enlace de ejecución

Puede crear un gancho de ejecución personalizado para una aplicación usando Trident Protect. Necesitas tener permisos de Propietario, Administrador o Miembro para crear ganchos de ejecución.

Utilice un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-hook.yaml`.
2. Configure los siguientes atributos para que coincidan con su entorno de Trident Protect y la configuración del clúster:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.applicationRef**: (*required*) El nombre de Kubernetes de la aplicación para la que ejecutar el hook de ejecución.
   - **Spec.stage**: (*required*) Una cadena que indica qué etapa durante la acción debe ejecutarse el gancho de ejecución. Los posibles valores son los siguientes:
     - Pre
     - Publicación
   - **Spec.action**: (*required*) Una cadena que indica qué acción tomará el gancho de ejecución, asumiendo que los filtros de enlace de ejecución especificados coinciden. Los posibles valores son los siguientes:
     - Snapshot
     - Backup
     - Restaurar
     - Conmutación al respaldo
   - **Spec.enabled**: (*Optional*) Indica si este enlace de ejecución está habilitado o desactivado. Si no se especifica, el valor predeterminado es TRUE.
   - **Spec.hookSource**: (*required*) Una cadena que contiene el script hook codificado en base64.
   - **SPEC.TIMEOUT**: (*Optional*) Un número que define cuánto tiempo en minutos se permite ejecutar el gancho de ejecución. El valor mínimo es 1 minuto y el valor predeterminado es 25 minutos si no se especifica.
   - **Spec.arguments**: (*Optional*) Una lista YAML de argumentos que puede especificar para el enlace de ejecución.
   - **Spec.matchingCriteria**: (*Optional*) Una lista opcional de pares de valores clave de criterios, cada par que forma un filtro de enlace de ejecución. Puede agregar hasta 10 filtros por gancho de ejecución.
   - **Spec.matchingCriteria.type**: (*Optional*) Una cadena que identifica el tipo de filtro de gancho de ejecución. Los posibles valores son los siguientes:
     - ConteneerImage
     - Nombre del contenedor
     - PodName
     - PodLabel
     - Nombre del espacio de nombre
   - **Spec.matchingCriteria.value**: (*Optional*) Una cadena o expresión regular que identifica el valor del filtro de enlace de ejecución.

Ejemplo 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, aplique el CR:

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

Utilice la CLI

Pasos

1. Cree el enlace de ejecución, sustituyendo los valores entre paréntesis por información de su 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>
```

Ejecute manualmente un enlace de ejecución

Puede ejecutar manualmente un enlace de ejecución para fines de prueba o si necesita volver a ejecutar el enlace manualmente después de un fallo. Debe tener permisos de Propietario, Administrador o Miembro para ejecutar manualmente los ganchos de ejecución.

La ejecución manual de un gancho de ejecución consta de dos pasos básicos:

1. Cree un backup de recursos, que recopila recursos y crea un backup de ellos, determinando dónde se ejecutará el enlace
2. Ejecute el enlace de ejecución en la copia de seguridad

Paso 1: Crear una copia de seguridad de recursos

Utilice un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-resource-backup.yaml`.
2. Configure los siguientes atributos para que coincidan con su entorno de Trident Protect y la configuración del clúster:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.applicationRef**: (*required*) El nombre de Kubernetes de la aplicación para la que crear la copia de seguridad del recurso.
   - **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. Puede utilizar el siguiente comando para buscar esta ruta:

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

     Ejemplo 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, aplique el CR:

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

Utilice la CLI

Pasos

1. Cree el backup sustituyendo valores entre paréntesis con información de su 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. Vea el estado del backup. Puede usar este comando de ejemplo varias veces hasta que se complete la operación:

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

3. Compruebe que el backup se ha realizado correctamente:

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

Paso 2: Ejecute el enlace de ejecución

Utilice un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre `trident-protect-hook-run.yaml`.
2. Configure los siguientes atributos para que coincidan con su entorno de Trident Protect y la configuración del clúster:

   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.applicationRef**: (*required*) Asegúrese de que este valor coincida con el nombre de la aplicación de ResourceBackup CR que creó en el paso 1.
   - **Spec.appVaultRef**: (*required*) Asegúrese de que este valor coincida con appVaultRef del ResourceBackup CR que creó en el paso 1.
   - **Spec.appArchivePath**: Asegúrate de que este valor coincida con el appArchivePath del ResourceBackup CR 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 tomará el gancho de ejecución, asumiendo que los filtros de enlace de ejecución especificados coinciden. Los posibles valores son los siguientes:
     - Snapshot
     - Backup
     - Restaurar
     - Conmutación al respaldo
   - **Spec.stage**: (*required*) Una cadena que indica qué etapa durante la acción debe ejecutarse el gancho de ejecución. Esta secuencia de gancho no ejecutará ganchos en ninguna otra etapa. Los posibles valores son los siguientes:
     - Pre
     - Publicación

     Ejemplo 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, aplique el CR:

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

Utilice la CLI

Pasos

1. Cree la solicitud de ejecución de enlace de ejecución manual:

```
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. Compruebe el estado de la ejecución del enlace de ejecución. Este comando se puede ejecutar varias veces hasta que se complete la operación:

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

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

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