Protege aplicaciones con Trident Protect

Trident

NetApp
September 26, 2025

Tabla de contenidos

Protege aplicaciones con Trident Protect

Más información sobre Trident Protect

NetApp Trident Protect proporciona funcionalidades avanzadas de gestión de datos de aplicaciones que mejoran la funcionalidad y la disponibilidad de aplicaciones de Kubernetes con estado respaldadas por los sistemas de almacenamiento de NetApp ONTAP y el aprovisionador de almacenamiento CSI de NetApp Trident. Trident Protect simplifica la gestión, la protección y el movimiento de cargas de trabajo en contenedores entre clouds públicos y entornos en las instalaciones. Además, ofrece funciones de automatización mediante su API y la CLI.

Puede proteger aplicaciones con Trident Protect creando recursos personalizados (CRS) o mediante la interfaz de línea de comandos Trident Protect.

El futuro

Puede obtener más información sobre los requisitos de Trident Protect antes de instalarlos:

- "Los requisitos de Trident protegen"

Instale Trident Protect

Los requisitos de Trident protegen

Comience verificando la preparación de su entorno operativo, clústeres de aplicaciones, aplicaciones y licencias. Asegúrese de que su entorno cumpla los siguientes requisitos para poner en marcha y operar Trident Protect.

La compatibilidad de clústeres de Kubernetes de Trident protege

Trident Protect es compatible con una amplia gama de ofertas de Kubernetes totalmente gestionadas y autogestionadas, entre las que se incluyen:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Rancher
- Cartera de VMware Tanzania
- Subida de Kubernetes

Asegúrese de que el clúster en el que instala Trident Protect está configurado con un controlador de instantáneas en ejecución y los CRD relacionados. Para instalar un controlador de instantánea, consulte "estas instrucciones" .

La compatibilidad del back-end de almacenamiento con Trident protege

Trident Protect es compatible con los siguientes back-ends de almacenamiento:

- Amazon FSX para ONTAP de NetApp
- Cloud Volumes ONTAP
- Cabinas de almacenamiento ONTAP de NetApp
- NetApp Volumes para Google Cloud
- Azure NetApp Files

Asegúrese de que el back-end de almacenamiento cumple los siguientes requisitos:

- Compruebe que el almacenamiento de NetApp conectado al clúster utilice Astra Trident 24,02 o una versión posterior (se recomienda Trident 24,10).
  - Si Astra Trident es anterior a la versión 24.06.1 y tienes pensado utilizar la funcionalidad de recuperación ante desastres de NetApp SnapMirror, debe habilitar manualmente el aprovisionador de Astra Control.
- Asegúrese de tener el aprovisionador de control de Astra más reciente (instalado y habilitado de forma predeterminada a partir de Astra Trident 24.06.1).
- Asegúrese de tener un back-end de almacenamiento NetApp ONTAP.
- Asegúrese de haber configurado un depósito de almacenamiento de objetos para almacenar backups.
- Cree los espacios de nombres de aplicaciones que desee utilizar para las aplicaciones o las operaciones de gestión de datos de aplicaciones. Trident Protect no crea estos espacios de nombres; si especifica un espacio de nombres no existente en un recurso personalizado, se producirá un error en la operación.

Requisitos para volúmenes de economía nas

Trident Protect admite las operaciones de backup y restauración en los volúmenes de economía nas. Actualmente no se admiten copias Snapshot, clones y replicación de SnapMirror en volúmenes económicos de nas. Debe habilitar un directorio snapshot para cada volumen económico nas que vaya a utilizar con Trident Protect.

Algunas aplicaciones no son compatibles con volúmenes que usan un directorio Snapshot. Para estas aplicaciones, debe ocultar el directorio Snapshot mediante la ejecución del siguiente comando en el sistema de almacenamiento de ONTAP:

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

Para habilitar el directorio snapshot, ejecute el siguiente comando para cada volumen nas-económico, sustituyéndolo `<volume-UUID>` por el UUID del volumen que desea cambiar:

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

Es posible habilitar los directorios de snapshots de forma predeterminada para volúmenes nuevos si se configura la opción Trident backend configuration `snapshotDir` en `true`. Los volúmenes existentes no se ven afectados.

Protección de datos con máquinas virtuales de KubeVirt

Trident Protect 24,10 y 24.10.1 y versiones posteriores tienen un comportamiento distinto al proteger las aplicaciones que se ejecutan en máquinas virtuales de KubeVirt. En ambas versiones, puede habilitar o deshabilitar la congelación y descongelación del sistema de archivos durante las operaciones de protección de datos.

Para todas las versiones de Trident Protect, para activar o desactivar la funcionalidad de congelación automática en entornos de OpenShift, es posible que deba otorgar permisos con privilegios al espacio de nombres de la aplicación. Por ejemplo:

```
oc adm policy add-scc-to-user privileged -z default -n
<application-namespace>
```

Trident Protect 24,10

Trident Protect 24,10 no garantiza automáticamente un estado coherente para los sistemas de archivos de máquinas virtuales KubeVirt durante las operaciones de protección de datos. Si desea proteger los datos de las máquinas virtuales KubeVirt con Trident Protect 24,10, debe habilitar manualmente la funcionalidad de congelación/descongelación para los sistemas de archivos antes de la operación de protección de datos. Esto garantiza que los sistemas de archivos estén en un estado consistente.

Puede configurar Trident Protect 24,10 para gestionar la congelación y descongelación del sistema de archivos de la máquina virtual durante las operaciones de protección de datos, mediante "configurar la virtualización"el siguiente comando:

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

Trident Protect 24.10.1 y posterior

A partir de Trident Protect 24.10.1, Trident Protect congela y descongela automáticamente los sistemas de archivos KubeVirt durante las operaciones de protección de datos. De manera opcional, puede deshabilitar este comportamiento automático mediante el siguiente comando:

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

Requisitos para la replicación de SnapMirror

NetApp SnapMirror está disponible para usar con Trident Protect para las siguientes soluciones de ONTAP:

- ASA de NetApp
- AFF de NetApp

- FAS de NetApp
- ONTAP Select de NetApp
- Cloud Volumes ONTAP de NetApp
- Amazon FSX para ONTAP de NetApp

Requisitos de clústeres de ONTAP para la replicación de SnapMirror

Asegúrese de que el clúster de ONTAP cumple los siguientes requisitos si tiene pensado utilizar la replicación de SnapMirror:

- **El aprovisionador de control Astra o Trident**: El aprovisionador de control Astra o Trident deben existir en los clústeres de Kubernetes de origen y de destino que utilizan ONTAP como backend. Trident Protect admite la replicación con tecnología de NetApp SnapMirror mediante clases de almacenamiento respaldadas por los controladores siguientes:
  - `ontap-nas`
  - `ontap-san`
- **Licencias**: Las licencias asíncronas de SnapMirror de ONTAP que utilizan el paquete de protección de datos deben estar habilitadas en los clústeres de ONTAP de origen y de destino. Consulte "Información general sobre las licencias de SnapMirror en ONTAP" si desea obtener más información.

Consideraciones sobre la relación de paridad para la replicación de SnapMirror

Compruebe que el entorno cumple los siguientes requisitos si piensa utilizar la paridad de back-end de almacenamiento:

- **Cluster y SVM**: Los back-ends de almacenamiento ONTAP deben ser peered. Consulte "Información general sobre relaciones entre iguales de clústeres y SVM" si desea obtener más información.

  > Compruebe que los nombres de las SVM utilizados en la relación de replicación entre dos clústeres de ONTAP sean únicos.

- **Trident y SVM**: Las SVM remotas entre iguales deben estar disponibles para el aprovisionador de control de Astra o Trident en el clúster de destino.
- **Backends administrados**: Necesitas agregar y administrar backends de almacenamiento ONTAP en Trident Protect para crear una relación de replicación.
- **NVMe sobre TCP**: Trident Protect no admite la replicación de NetApp SnapMirror para los back-ends de almacenamiento que están utilizando el protocolo NVMe sobre TCP.

Configuración de Trident/ONTAP para la replicación de SnapMirror

Trident Protect requiere que configure al menos un back-end de almacenamiento que admita la replicación para los clústeres de origen y destino. Si los clústeres de origen y destino son los mismos, la aplicación de destino debe usar un back-end de almacenamiento diferente al de la aplicación de origen para obtener la mejor resiliencia.

Instalar y configurar Trident Protect

Si su entorno cumple los requisitos de protección Trident, puede seguir estos pasos para instalar Trident Protect en el clúster. Puede obtener Trident Protect de NetApp o instalarlo

desde su propio registro privado. La instalación desde un registro privado es útil si su clúster no puede acceder a Internet.

> De forma predeterminada, Trident Protect recopila información de soporte que ayuda con cualquier caso de soporte de NetApp que pueda abrir, incluidos registros, métricas e información de topología sobre clústeres y aplicaciones gestionadas. Trident Protect envía estos paquetes de soporte a NetApp a diario. Opcionalmente, puede deshabilitar esta recogida de bundles de soporte al instalar Trident Protect. Puede hacerlo manualmente "generar un bundle de soporte"en cualquier momento.

Instale Trident Protect

Instale Trident Protect de NetApp

Pasos

1. Añada el repositorio Helm de Trident:

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

2. Instale los CRD de Trident Protect:

```
helm install trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.1 --create-namespace --namespace
trident-protect
```

3. Utilice Helm para instalar Trident Protect mediante uno de los siguientes comandos. Sustituya `<name_of_cluster>` por un nombre de clúster, que se asignará al clúster y se utilizará para identificar los backups y las snapshots del clúster:
   - Instale Trident Protect normalmente:

```
helm install trident-protect netapp-trident-protect/trident-
protect --set clusterName=<name_of_cluster> --version 100.2410.1
--create-namespace --namespace trident-protect
```

   - Instale Trident Protect y deshabilite las cargas programadas diarias del bundle de soporte de Trident Protect AutoSupport:

```
helm install trident-protect netapp-trident-protect/trident-
protect --set autoSupport.enabled=false --set
clusterName=<name_of_cluster> --version 100.2410.1 --create
-namespace --namespace trident-protect
```

Instale Trident Protect desde un registro privado

Puede instalar Trident Protect desde un registro de imágenes privado si su clúster de Kubernetes no puede acceder a Internet. En estos ejemplos, reemplace los valores entre paréntesis por información de su entorno:

Pasos

1. Tire de las siguientes imágenes a su máquina local, actualice las etiquetas y, a continuación, empújelas en su registro privado:

```
netapp/controller:24.10.1
netapp/restic:24.10.1
netapp/kopia:24.10.1
netapp/trident-autosupport:24.10.0
netapp/exechook:24.10.1
netapp/resourcebackup:24.10.1
netapp/resourcerestore:24.10.1
netapp/resourcedelete:24.10.1
bitnami/kubectl:1.30.2
kubebuilder/kube-rbac-proxy:v0.16.0
```

Por ejemplo:

```
docker pull netapp/controller:24.10.1
```

```
docker tag netapp/controller:24.10.1 <private-registry-
url>/controller:24.10.1
```

```
docker push <private-registry-url>/controller:24.10.1
```

2. Cree el espacio de nombres del sistema Trident Protect:

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

3. Inicie sesión en el Registro:

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

4. Cree un secreto de extracción para utilizarlo en la autenticación del registro privado:

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

5. Añada el repositorio Helm de Trident:

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

6. Crear un archivo llamado `protectValues.yaml`. Asegúrese de que contiene las siguientes configuraciones de Trident Protect:

```
---
image:
  registry: <private-registry-url>
imagePullSecrets:
  - name: regcred
controller:
  image:
    registry: <private-registry-url>
rbacProxy:
  image:
    registry: <private-registry-url>
crCleanup:
  imagePullSecrets:
    - name: regcred
webhooksCleanup:
  imagePullSecrets:
    - name: regcred
```

7. Instale los CRD de Trident Protect:

```
helm install trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.1 --create-namespace --namespace
trident-protect
```

8. Utilice Helm para instalar Trident Protect mediante uno de los siguientes comandos. Sustituya `<name_of_cluster>` por un nombre de clúster, que se asignará al clúster y se utilizará para identificar los backups y las snapshots del clúster:
   - Instale Trident Protect normalmente:

```
helm install trident-protect netapp-trident-protect/trident-
protect --set clusterName=<name_of_cluster> --version 100.2410.1
--create-namespace --namespace trident-protect -f
protectValues.yaml
```

   - Instale Trident Protect y deshabilite las cargas programadas diarias del bundle de soporte de Trident Protect AutoSupport:

```
helm install trident-protect netapp-trident-protect/trident-
protect --set autoSupport.enabled=false --set
clusterName=<name_of_cluster> --version 100.2410.1 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

Especifique los límites de recursos del contenedor Trident Protect

Puede utilizar un archivo de configuración para especificar límites de recursos para contenedores Trident Protect después de instalar Trident Protect. La configuración de límites de recursos permite controlar cuántos recursos del clúster consumen las operaciones de Trident Protect.

Pasos

1. Crear un archivo llamado `resourceLimits.yaml`.
2. Rellene el archivo con opciones de límite de recursos para contenedores Trident Protect según las necesidades de su entorno.

   El siguiente archivo de configuración de ejemplo muestra la configuración disponible y contiene los vaules predeterminados para cada límite de recursos:

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

```
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. Aplique los valores del `resourceLimits.yaml` archivo:

```
helm upgrade trident-protect -n trident-protect -f <resourceLimits.yaml>
--reuse-values
```

Instale el complemento de la CLI Trident Protect

Puede utilizar el plugin de línea de comandos Trident Protect, que es una extensión de la utilidad Trident `tridentctl`, para crear e interactuar con los recursos personalizados de Trident Protect (CRS).

Instale el complemento de la CLI Trident Protect

Antes de utilizar la utilidad de línea de comandos, debe instalarla en la máquina que utiliza para acceder al clúster. Siga estos pasos, dependiendo de si su máquina utiliza una CPU x64 o ARM.

Descargar plugin para CPU Linux AMD64

Pasos

1. Descargue el complemento de la CLI de Trident Protect:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-linux-amd64
```

Descargar plugin para CPU Linux ARM64

Pasos

1. Descargue el complemento de la CLI de Trident Protect:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-linux-arm64
```

Descargar plugin para CPU Mac AMD64

Pasos

1. Descargue el complemento de la CLI de Trident Protect:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-macos-amd64
```

Descargar plugin para CPU Mac ARM64

Pasos

1. Descargue el complemento de la CLI de Trident Protect:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-macos-arm64
```

1. Active los permisos de ejecución para el binario del plugin:

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

2. Copie el binario del plugin a una ubicación definida en su variable PATH. Por ejemplo, `/usr/bin` o `/usr/local/bin` (puede que necesite Privileges elevado):

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

3. Opcionalmente, puede copiar el binario del plugin a una ubicación en su directorio principal. En este caso, se recomienda asegurarse de que la ubicación forma parte de la variable PATH:

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

Copiar el plugin a una ubicación en su variable PATH le permite usar el plugin escribiendo `tridentctl-protect` o `tridentctl protect` desde cualquier ubicación.

Consulte la ayuda del complemento de la CLI de Trident

Puede utilizar las funciones de ayuda del plugin incorporado para obtener ayuda detallada sobre las capacidades del plugin:

Pasos

1. Utilice la función de ayuda para ver la guía de uso:

```
tridentctl-protect help
```

Habilite el autocompletado de comandos

Después de instalar el complemento de CLI Trident Protect, puede habilitar la finalización automática para ciertos comandos.

Active la finalización automática del shell Bash

Pasos

1. Descargue el script de finalización:

```
curl -L -O https://github.com/NetApp/tridentctl-
protect/releases/download/24.10.1/tridentctl-completion.bash
```

2. Cree un nuevo directorio en el directorio principal para que contenga el script:

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

3. Mueva el script descargado al `~/.bash/completions` directorio:

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

4. Añada la siguiente línea al `~/.bashrc` archivo en su directorio principal:

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

Active la finalización automática del shell Z

Pasos

1. Descargue el script de finalización:

```
curl -L -O https://github.com/NetApp/tridentctl-
protect/releases/download/24.10.1/tridentctl-completion.zsh
```

2. Cree un nuevo directorio en el directorio principal para que contenga el script:

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

3. Mueva el script descargado al `~/.zsh/completions` directorio:

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

4. Añada la siguiente línea al `~/.zprofile` archivo en su directorio principal:

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

Resultado

En su próximo inicio de sesión en el shell, puede utilizar el comando auto-completado con el plugin tridentctl-Protect.

Gestione Trident Protect

Gestione la autorización y el control de acceso de Trident Protect

Trident Protect utiliza el modelo de Kubernetes de control de acceso basado en roles (RBAC). De forma predeterminada, Trident Protect proporciona un único espacio de nombres del sistema y su cuenta de servicio predeterminada asociada. Si cuenta con una organización con muchos usuarios o con necesidades de seguridad específicas, puede utilizar las funciones de control de acceso basado en roles de Trident Protect para obtener un control más granular sobre el acceso a los recursos y los espacios de nombres.

El administrador de clúster siempre tiene acceso a los recursos del espacio de nombres predeterminado `trident-protect` y también puede acceder a los recursos en el resto de espacios de nombres. Para controlar el acceso a recursos y aplicaciones, es necesario crear espacios de nombres adicionales y agregar recursos y aplicaciones a esos espacios de nombres.

Tenga en cuenta que ningún usuario puede crear CRS de gestión de datos de aplicaciones en el espacio de nombres predeterminado `trident-protect`. Debe crear CRS de gestión de datos de aplicaciones en un espacio de nombres de aplicaciones (como práctica recomendada, crear CRS de gestión de datos de aplicaciones en el mismo espacio de nombres que la aplicación asociada).

> Sólo los administradores deben tener acceso a los objetos de recursos personalizados Privileged Trident Protect, que incluyen:
>
> - **AppVault**: Requiere datos de credenciales de bucket
> - **Paquete de Protección**: Recopila métricas, registros y otros datos sensibles de Trident
> - **BundleSchedule**: Gestiona los horarios de recopilación de registros
>
> Como práctica recomendada, use RBAC para restringir el acceso a los objetos con privilegios a los administradores.

Para obtener más información sobre cómo el RBAC regula el acceso a los recursos y espacios de nombres, consulte la "Documentación de RBAC de Kubernetes".

Para obtener información sobre las cuentas de servicio, consulte la "Documentación de la cuenta de servicio de Kubernetes".

Ejemplo: Administrar el acceso para dos grupos de usuarios

Por ejemplo, una organización tiene un administrador de clústeres, un grupo de usuarios de ingeniería y un grupo de usuarios de marketing. El administrador del clúster debe realizar las siguientes tareas para crear un entorno en el que el grupo de ingeniería y el grupo de marketing tengan acceso solo a los recursos asignados a sus respectivos espacios de nombres.

Paso 1: Crear un espacio de nombres para contener recursos para cada grupo

La creación de un espacio de nombres permite separar los recursos de forma lógica y controlar mejor quién tiene acceso a dichos recursos.

Pasos

1. Cree un espacio de nombres para el grupo de ingeniería:

```
kubectl create ns engineering-ns
```

2. Cree un espacio de nombres para el grupo de marketing:

```
kubectl create ns marketing-ns
```

Paso 2: Crear nuevas cuentas de servicio para interactuar con los recursos de cada espacio de nombres

Cada nuevo espacio de nombres que cree viene con una cuenta de servicio predeterminada, pero debe crear una cuenta de servicio para cada grupo de usuarios para que pueda dividir aún más Privileges entre grupos en el futuro si es necesario.

Pasos

1. Cree una cuenta de servicio para el grupo de ingeniería:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Cree una cuenta de servicio para el grupo de marketing:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

Paso 3: Crear un secreto para cada nueva cuenta de servicio

Un secreto de cuenta de servicio se utiliza para autenticarse con la cuenta de servicio, y se puede eliminar y volver a crear fácilmente si está comprometido.

Pasos

1. Cree un secreto para la cuenta de servicio de ingeniería:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. Cree un secreto para la cuenta de servicio de marketing:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
type: kubernetes.io/service-account-token
```

Paso 4: Cree un objeto RoleBinding para enlazar el objeto ClusterRole a cada nueva cuenta de servicio

Al instalar Trident Protect, se crea un objeto ClusterRole predeterminado. Puede enlazar este ClusterRole a la cuenta de servicio creando y aplicando un objeto RoleBinding.

Pasos

1. Enlazar ClusterRole a la cuenta de servicio de ingeniería:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. Enlazar ClusterRole a la cuenta de servicio de marketing:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

Paso 5: Probar permisos

Compruebe que los permisos son correctos.

Pasos

1. Confirme que los usuarios de ingeniería pueden acceder a los recursos de ingeniería:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Confirme que los usuarios de ingeniería no pueden acceder a los recursos de marketing:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

Paso 6: Otorgar acceso a los objetos de AppVault

Para realizar tareas de gestión de datos, como backups e instantáneas, el administrador de clúster debe conceder acceso a los objetos de AppVault a usuarios individuales.

Pasos

1. Cree y aplique un archivo YAML de combinación secreta y AppVault que otorgue acceso a un usuario a un AppVault. Por ejemplo, el siguiente CR otorga acceso a un AppVault al usuario `eng-user`:

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. Cree y aplique un CR de rol para permitir que los administradores del cluster concedan acceso a recursos específicos en un espacio de nombres. Por ejemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. Cree y aplique un CR de RoleBinding para enlazar los permisos al usuario eng-user. Por ejemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. Compruebe que los permisos son correctos.

   a. Se ha intentado recuperar la información del objeto AppVault para todos los espacios de nombres:

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   Debería ver una salida similar a la siguiente:

```
Error from server (Forbidden): appvaults.protect.trident.netapp.io is
forbidden: User "system:serviceaccount:engineering-ns:eng-user"
cannot list resource "appvaults" in API group
"protect.trident.netapp.io" in the namespace "trident-protect"
```

b. Prueba para ver si el usuario puede obtener la información de AppVault a la que ahora tiene permiso para acceder:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
trident-protect
```

Debería ver una salida similar a la siguiente:

```
yes
```

Resultado

Los usuarios a los que ha otorgado permisos de AppVault deben poder usar objetos de AppVault autorizados para operaciones de gestión de datos de aplicaciones y no deben poder acceder a ningún recurso fuera de los espacios de nombres asignados ni crear nuevos recursos a los que no tengan acceso.

Generar un bundle de soporte de Trident Protect

Trident Protect permite a los administradores generar paquetes que incluyen información útil para el soporte de NetApp, incluidos registros, métricas e información de topología sobre los clústeres y las aplicaciones que se están gestionando. Si está conectado a Internet, puede cargar paquetes de soporte en el sitio de soporte de NetApp (NSS) mediante un archivo de recursos personalizados (CR).

Cree un paquete de soporte mediante un CR

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre (por ejemplo, `trident-protect-support-bundle.yaml`).
2. Configure los siguientes atributos:
   - **metadata.name**: (*required*) El nombre de este recurso personalizado; elija un nombre único y sensible para su entorno.
   - **Spec.triggerType**: (*required*) Determina si el paquete de soporte se genera inmediatamente o se programa. La generación de paquetes programada se tiene lugar a LAS 12am UTC. Los posibles valores son los siguientes:
     - Programado
     - Manual
   - **SPEC.uploadEnabled**: (*Opcional*) Controla si el paquete de soporte debe cargarse en el sitio de soporte de NetApp después de que se genere. Si no se especifica, el valor por defecto es `false`. Los posibles valores son los siguientes:
     - verdadero
     - false (predeterminado)
   - **Spec.dataWindowStart**: (*Optional*) Una cadena de fecha en formato RFC 3339 que especifica la fecha y la hora en que debe comenzar la ventana de datos incluidos en el paquete de soporte. Si no se especifica, el valor predeterminado es hace 24 horas. La fecha de ventana más antigua que puede especificar es hace 7 días.

     Ejemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. Después de rellenar `astra-support-bundle.yaml` el archivo con los valores correctos, aplique el CR:

```
kubectl apply -f trident-protect-support-bundle.yaml
```

Cree un bundle de soporte mediante la CLI

Pasos

1. Cree el paquete de soporte, reemplazando valores entre paréntesis con la información del entorno.

`trigger-type`Determina si el grupo se crea inmediatamente o si la hora de creación está determinada por la programación, y puede ser `Manual` o. `Scheduled` El valor predeterminado es `Manual`.

Por ejemplo:

```
tridentctl-protect create autosupportbundle <my_bundle_name>
--trigger-type <trigger_type>
```

Actualice Trident Protect

Puede actualizar Trident Protect a la última versión para aprovechar las nuevas funciones o correcciones de errores.

Para actualizar Trident Protect, realice los siguientes pasos.

Pasos

1. Actualice el repositorio de Trident Helm:

```
helm repo update
```

2. Actualice los CRD de Trident Protect:

```
helm upgrade trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.1  --namespace trident-protect
```

3. Actualizar Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2410.1 --namespace trident-protect
```

Gestione y proteja aplicaciones

Utilice los objetos de Trident Protect AppVault para administrar buckets

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

Ejemplos de generación de claves y definición de AppVault

Al definir un CR de AppVault, debe incluir credenciales para acceder a los recursos alojados por el proveedor. La forma en que se generan las claves para las credenciales variará según el proveedor. A continuación se muestran ejemplos de generación de claves de línea de comandos para varios proveedores, seguidos de definiciones de AppVault de ejemplo para cada proveedor.

Ejemplos de generación de claves

Puede utilizar los siguientes ejemplos para crear claves para las credenciales de cada proveedor de cloud.

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=<generic-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=<generic-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
=<generic-s3-trident-protect-src-bucket-secret> -n trident-protect
```

Ejemplos de AppVault CR

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

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3 (AWS)

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  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-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

StorageGRID S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-
971f-ac4a83621922
  namespace: trident-protect
spec:
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

Ejemplos de creación de AppVault con la CLI de Trident Protect

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

Google Cloud

```
tridentctl-protect create vault GCP my-new-vault --bucket mybucket
--project my-gcp-project --secret <gcp-creds>/<credentials>
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> --bucket <bucket-name>
--secret  <secret-name>  --endpoint <s3-endpoint>
```

Microsoft Azure

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

Genérico S3

```
tridentctl-protect create vault GenericS3 <vault-name> --bucket
<bucket-name> --secret  <secret-name>  --endpoint <s3-endpoint>
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> --bucket <bucket-
name> --secret  <secret-name>  --endpoint <s3-endpoint>
```

StorageGRID S3

```
tridentctl-protect create vault StorageGridS3 s3vault --bucket <bucket-
name> --secret  <secret-name>  --endpoint <s3-endpoint>
```

Utilice el explorador AppVault para ver la información de AppVault

Puede usar el complemento de la CLI de Trident Protect para ver información sobre los objetos de AppVault que se han creado en el clúster.

Pasos

1. Ver el contenido de un objeto AppVault:

   ```
   tridentctl-protect get appvaultcontent gcp-vault --show-resources all
   ```

   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 sólo está disponible si se ha especificado un nombre de clúster en la instalación del sistema Trident Protect. 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 todas las copias de Snapshot y las copias de seguridad almacenadas en el bloque asociado.

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
```

Elimine un AppVault con 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
```

Defina una aplicación para administrar 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. AppVault CR es específico de su entorno; para ver ejemplos de AppVault CRS, consulte "Recursos personalizados de AppVault."

Defina una aplicación

Debe definir cada aplicación que desee gestionar con Trident Protect. Puede definir una aplicación para la gestión creando manualmente una CR de aplicación o mediante el uso de 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 etiquetas de espacio de nombres o un nombre de espacio de nombres para especificar espacios de nombres en los que existen los recursos de la aplicación. El espacio de nombres de la aplicación debe formar parte de esta lista.

        Ejemplo YAML:

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

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
```

Agregue una aplicación mediante la CLI

Pasos

1. Cree y aplique la definición de la aplicación, sustituyendo los valores entre paréntesis por la 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 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>
```

Proteja las aplicaciones con Trident Protect

Puede proteger todas las aplicaciones gestionadas por Trident Protect al tomar copias Snapshot y backups usando una política de protección automatizada o ad hoc.

Puede configurar Trident Protect para congelar y descongelar sistemas de archivos durante las 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.

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.

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**: (*Optional*): 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.

```
---
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> -n
<application_namespace>
```

Cree un programa de protección de datos

La política de protección protege una aplicación mediante la creación de snapshots, backups o ambos con una programación definida. Puede optar por crear snapshots y backups por hora, día, semana y mes, y especificar la cantidad de copias que desea retener.

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**: El número de copias de seguridad a retener. Cero indica que no se debe crear ningún backup.
   - **Spec.snapshotRetention**: El número de instantáneas a retener. Cero indica que no se debe crear ninguna instantánea.
   - **spec.granularity**: La frecuencia con la que debe ejecutarse el horario. Los posibles valores, junto con los campos asociados necesarios:
     - `hourly` (requiere que especifique `spec.minute`)
     - `daily` (requiere que especifique `spec.minute` y `spec.hour`)
     - `weekly` (requiere que especifique `spec.minute, spec.hour`, y `spec.dayOfWeek`)
     - `monthly` (requiere que especifique `spec.minute, spec.hour`, y `spec.dayOfMonth`)
   - **Spec.dayOfMonth**: (*Optional*) El día del mes (1 - 31) en el que se debe ejecutar el horario. Este campo es necesario si la granularidad se define en `monthly`.
   - **SPEC.DayOfWeek**: (*Optional*) El día de la semana (0 - 7) en el que se debe ejecutar el horario. Los valores de 0 o 7 indican el domingo. Este campo es necesario si la granularidad se define en `weekly`.
   - **SPEC.HOUR**: (*Opcional*) La hora del día (0 - 23) que debe ejecutarse el horario. Este campo es necesario si la granularidad se define en `daily`, `weekly`o `monthly`.
   - **Spec.minute**: (*Optional*) El minuto de la hora (0 - 59) que debe ejecutarse el horario. Este campo es necesario si la granularidad se define en `hourly`, , , `daily` `weekly`o `monthly`.

```
---
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: <monthly>
  dayOfMonth: "1"
  dayOfWeek: "0"
  hour: "0"
  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> --data-mover
<kopia_or_restic> --day-of-month <day_of_month_to_run_schedule>
--day-of-week <day_of_month_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> -n <application_namespace>
```

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.

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 backup y restauración con gestión eficiente del espacio para back-ends de almacenamiento que utilizan 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 pool de capacidad.

Antes de empezar

Asegúrese de lo siguiente:

- Ha instalado Trident Protect.
- Debe haber definido una aplicación en Trident Protect. Esta aplicación tendrá 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 no se habilita el directorio Snapshot, Trident Protect elige la funcionalidad normal de backup, que consume temporalmente el espacio del pool de capacidad durante el proceso de backup. En este caso, asegúrese de que haya espacio suficiente disponible en el pool de capacidad para crear un volumen temporal del tamaño del volumen del que se va a realizar el backup.

Resultado

La aplicación está lista para backup y restauración con Trident Protect. Otras aplicaciones también pueden utilizar cada RVP para realizar backups y restauraciones de datos.

Restaure aplicaciones mediante Trident Protect

Es posible usar Trident Protect para restaurar la aplicación desde una snapshot o un backup. La restauración a partir de una snapshot existente será más rápida cuando se restaure 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.

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 RedHat OpenShift, es importante tener en cuenta el papel fundamental de las anotaciones de espacio de nombres en entornos OpenShift. Las anotaciones del espacio de nombres garantizan que los pods restaurados cumplan los permisos y las configuraciones de seguridad adecuados definidos por las restricciones de contexto de seguridad (SCCs) de OpenShift y puedan acceder a los volúmenes sin problemas de permiso. 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:

```
kubectl set env -n trident-protect deploy/trident-protect-controller-
manager
RESTORE_SKIP_NAMESPACE_ANNOTATIONS=<annotation_key_to_skip_1>,<annotation_
key_to_skip_2>
```

Si instaló la aplicación de origen con Helm con el `--create-namespace` indicador, se le dará un tratamiento especial a la `name` clave de etiqueta. Durante el proceso de restauración o recuperación tras fallos, Trident protege esta etiqueta al espacio de nombres de destino, pero actualiza el valor al valor del espacio de nombres de destino si el valor del origen coincide con el espacio de nombres de origen. Si este valor no coincide con el espacio de nombres de origen, se copia al espacio de nombres de destino sin cambios.

Ejemplo

El siguiente ejemplo presenta un espacio de nombres de origen y 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

Restauración desde un backup a un espacio de nombres diferente

Cuando se restaura un backup en un espacio de nombres diferente con BackupRestore CR, Trident Protect restaura la aplicación en un espacio de nombres nuevo y crea un CR de aplicación para la aplicación restaurada. Para proteger la aplicación restaurada, cree backups o snapshots bajo demanda, o establezca una programación 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.

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.
   - **Spec.storageClassMapping**: La asignación de la clase de almacenamiento de origen de la operación de restauración a la clase de almacenamiento de destino. Reemplace `destinationStorageClass` y `sourceStorageClass` 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"}]
       storageClassMapping:
         destination: "${destinationStorageClass}"
         source: "${sourceStorageClass}"
     ```

3. (*Optional*) Si necesita seleccionar solo ciertos recursos de la aplicación para restaurar, 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"`: .

    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.

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:

   - **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.

Antes de empezar

Asegúrese de que se cumplen los siguientes requisitos previos:

- El clúster de destino tiene instalado Trident Protect.
- 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.

Pasos

1. Compruebe la disponibilidad de AppVault CR en el clúster de destino mediante el complemento de 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> -n <application_namespace>
--context <destination_cluster_name>
```

Restauración desde una copia snapshot a un espacio de nombres diferente

Puede restaurar datos desde una copia Snapshot con un archivo de recurso personalizado (CR) en un espacio de nombres diferente o en el espacio de nombres de origen original. Al restaurar una snapshot en un espacio de nombres diferente con SnapshotRestore CR, Trident Protect restaura la aplicación en un espacio de nombres nuevo y crea un CR de aplicación para la aplicación restaurada. Para proteger la aplicación restaurada, cree backups o snapshots bajo demanda, o establezca una programación de protección.

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.
   - **Spec.storageClassMapping**: La asignación de la clase de almacenamiento de origen de la operación de restauración a la clase de almacenamiento de destino. Reemplace `destinationStorageClass` y `sourceStorageClass` 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"}]
       storageClassMapping:
         destination: "${destinationStorageClass}"
         source: "${sourceStorageClass}"
     ```

3. (*Optional*) Si necesita seleccionar solo ciertos recursos de la aplicación para restaurar, 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"`: .

    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.

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:
   - **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 <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}'
```

Replicar aplicaciones con NetApp SnapMirror y Trident Protect

Con Trident Protect, puede usar las funcionalidades de replicación asíncrona de la tecnología NetApp SnapMirror para replicar datos y cambios de aplicaciones de un back-end 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 RedHat OpenShift, es importante tener en cuenta el papel fundamental de las anotaciones de espacio de nombres en entornos OpenShift. Las anotaciones del espacio de nombres garantizan que los pods restaurados cumplan los permisos y las configuraciones de seguridad adecuados definidos por las restricciones de contexto de seguridad (SCCs) de OpenShift y puedan acceder a los volúmenes sin problemas de permiso. 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:

```
kubectl set env -n trident-protect deploy/trident-protect-controller-
manager
RESTORE_SKIP_NAMESPACE_ANNOTATIONS=<annotation_key_to_skip_1>,<annotation_
key_to_skip_2>
```

Si instaló la aplicación de origen con Helm con el `--create-namespace` indicador, se le dará un tratamiento especial a la `name` clave de etiqueta. Durante el proceso de restauración o recuperación tras fallos, Trident protege esta etiqueta al espacio de nombres de destino, pero actualiza el valor al valor del espacio de nombres de destino si el valor del origen coincide con el espacio de nombres de origen. Si este valor no coincide con el espacio de nombres de origen, se copia al espacio de nombres de destino sin cambios.

Ejemplo

El siguiente ejemplo presenta un espacio de nombres de origen y 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 las operaciones de protección de datos. "Obtenga más información sobre cómo configurar la congelación del sistema de archivos con Trident Protect".

Configurar una relación de replicación

La configuración de una relación de replicación implica lo siguiente:

- Elegir con qué frecuencia quieres que Trident Protect tome una instantánea de la aplicación (que incluye los recursos de Kubernetes de la aplicación, así como las instantáneas de volumen de 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. Cree un AppVault para la aplicación de origen en el clúster 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>
```

2. 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. Si lo desea, realice una instantánea de la aplicación de origen. Esta copia Snapshot se utiliza como base de la aplicación en el clúster de destino. Si omite este paso, deberá esperar a que se ejecute la siguiente instantánea programada para que tenga una instantánea reciente.

Tome una instantánea con 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**: (*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 de la aplicación de origen CR.
- **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-0e1f88ab-f013-4bce-8ae9-6afed9df59a1
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  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
```

Tome una instantánea utilizando la CLI

a. 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>
```

4. Cree una aplicación de origen AppVault CR en el clúster de destino que sea 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`).
5. Aplicar el CR:

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

6. Cree un AppVault 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 my-app-namespace
```

7. Cree un archivo CR de AppMirrorRelationship:

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.storageClassName**: (*required*) Elija el nombre de una clase de almacenamiento válida en el clúster. La clase de almacenamiento debe estar vinculada a un equipo virtual de almacenamiento ONTAP que esté relacionado con el entorno de origen.
- **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. Cree y aplique el objeto AppMirrorRelationship, reemplazando los valores entre paréntesis con información de su entorno. Por ejemplo:

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --recurrence-rule <rule> --source-app
<my_source_app> --source-app-vault <my_source_app_vault> -n
<application_namespace>
```

8. (*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
```

Conmutación por error al clúster de destino

Con Trident Protect, puede conmutar al respaldo de aplicaciones replicadas a un clúster de destino. Este procedimiento detiene la relación de replicación y conecta la aplicación en el clúster de destino. Trident Protect no detiene la aplicación en el clúster de origen si estaba operativa.

Pasos

1. Abra el archivo AppMirrorRelationship CR (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. Crear una instantánea de la aplicación de origen.
2. Abra el archivo AppMirrorRelationship CR (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. Elimine el AppMirrorRelationship CR en el clúster de destino original. 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 los CRS de AppVault están listos en cada clúster.
4. Configure una relación de replicación en el cluster opuesto, configurando valores para la dirección inversa.

Invertir dirección de replicación de aplicaciones

Al invertir la dirección de replicación, Trident Protect mueve la aplicación al back-end del almacenamiento de destino, a la vez que continúa replicando al back-end del 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. Crear una instantánea de cierre:

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. Cuando se complete la copia de Snapshot, se debe obtener el estado de la copia de Snapshot:

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

3. 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 clúster de destino al 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

Con Trident Protect, puede obtener un «fallo» tras una operación de recuperación tras fallos utilizando la siguiente secuencia de operaciones. En este flujo de trabajo para restaurar la dirección de replicación original, Trident protege replica (resincroniza) cualquier cambio de aplicación de nuevo en 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. Elimine el CR de AppMirrorRelationship:

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

Migrar aplicaciones con Trident Protect

Puede migrar sus aplicaciones entre clústeres o clases de almacenamiento si restaura los datos de backup o Snapshot a otro clúster o tipo de almacenamiento.

> 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 un clon entre clústeres), crear un backup en el clúster de origen y, a continuación, restaurar el backup en un clúster diferente. Asegúrese de que Trident Protect está 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 otra clase de almacenamiento restaurando una instantánea a otra 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>
```

Gestionar los enlaces 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 enlaces de ejecución, según cuándo se pueden 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 congelan los sistemas 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.

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 los enlaces de ejecución utilizan se escriban en formato de scripts de shell ejecutables.
- El tamaño del script está limitado a 96 KB.
- Trident Protect utiliza la configuración de enlace de ejecución y cualquier criterio de coincidencia para determinar qué enlaces se aplican 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 agregue a un enlace de ejecución utiliza una expresión regular para hacer coincidir los contenedores del clúster. Cuando un gancho coincide con un contenedor, el gancho ejecutará su script asociado en ese contenedor. Las expresiones regulares para los filtros utilizan la sintaxis expresión regular 2 (RE2), que no admite la creación de un filtro que excluye contenedores de la lista de coincidencias. Para obtener información sobre la sintaxis que soporta Trident Protect para expresiones regulares en filtros de

enlace 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 mediante Trident Protect. Necesita tener permisos de propietario, administrador o miembro para crear enlaces 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 Trident Protect y la configuración de 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>
```

Desinstale Trident Protect

Es posible que necesite eliminar los componentes de Trident Protect si va a actualizar de una versión de prueba a una versión completa del producto.

Para quitar Trident Protect, realice los siguientes pasos.

Pasos

1. Elimine los archivos CR de Trident Protect:

```
helm uninstall -n trident-protect trident-protect-crds
```

2. Quitar Trident Protect:

```
helm uninstall -n trident-protect trident-protect
```

3. Elimine el espacio de nombres Trident Protect:

```
kubectl delete ns trident-protect
```