Gérer et protéger les applications

Trident

NetApp
March 05, 2026

Sommaire

Gérer et protéger les applications

Utilisez les objets Trident Protect AppVault pour gérer les compartiments.

La ressource personnalisée (CR) du compartiment pour Trident Protect est connue sous le nom d'AppVault. Les objets AppVault sont la représentation déclarative du flux de travail Kubernetes d'un bucket de stockage. Un AppVault CR contient les configurations nécessaires pour qu'un bucket soit utilisé dans les opérations de protection, telles que les sauvegardes, les snapshots, les opérations de restauration et la réplication SnapMirror . Seuls les administrateurs peuvent créer des AppVaults.

Vous devez créer une CR AppVault manuellement ou depuis la ligne de commande lorsque vous effectuez des opérations de protection des données sur une application. La CR AppVault est spécifique à votre environnement et vous pouvez vous inspirer des exemples de cette page pour créer des CR AppVault.

> Assurez-vous que le fichier CR d'AppVault se trouve sur le cluster où Trident Protect est installé. Si le CR AppVault n'existe pas ou si vous ne pouvez pas y accéder, la ligne de commande affiche une erreur.

Configurer l'authentification et les mots de passe AppVault

Avant de créer un AppVault CR, assurez-vous que l'AppVault et le dispositif de transfert de données que vous choisissez peuvent s'authentifier auprès du fournisseur et de toutes les ressources associées.

Mots de passe du référentiel de déplacement de données

Lorsque vous créez des objets AppVault à l'aide de CR ou du plugin CLI Trident Protect, vous pouvez spécifier un secret Kubernetes avec des mots de passe personnalisés pour le chiffrement Restic et Kopia. Si vous ne spécifiez pas de mot de passe secret, Trident Protect utilise un mot de passe par défaut.

- Lors de la création manuelle de demandes de changement AppVault, utilisez le champ **spec.dataMoverPasswordSecretRef** pour spécifier le secret.
- Lors de la création d'objets AppVault à l'aide de l'interface de ligne de commande Trident Protect, utilisez le `--data-mover-password-secret-ref` argument permettant de préciser le secret.

Créer un mot de passe secret pour le référentiel de déplacement de données

Utilisez les exemples suivants pour créer le mot de passe secret. Lorsque vous créez des objets AppVault, vous pouvez indiquer à Trident Protect d'utiliser ce secret pour s'authentifier auprès du référentiel de transfert de données.

- Selon le logiciel de transfert de données que vous utilisez, il vous suffit d'indiquer le mot de passe correspondant. Par exemple, si vous utilisez Restic et que vous ne prévoyez pas d'utiliser Kopia à l'avenir, vous pouvez inclure uniquement le mot de passe Restic lors de la création du secret.
- Conservez le mot de passe en lieu sûr. Vous en aurez besoin pour restaurer les données sur le même cluster ou sur un autre. Si le cluster ou le `trident-protect` L'espace de noms a été supprimé ; vous ne pourrez pas restaurer vos sauvegardes ou vos instantanés sans le mot de passe.

Utilisez un CR

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

Utiliser la CLI

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

Autorisations IAM de stockage compatibles S3

Lorsque vous accédez à un stockage compatible S3 tel qu'Amazon S3, Generic S3, "StorageGrid S3" , ou "ONTAP S3" Lors de l'utilisation de Trident Protect, vous devez vous assurer que les informations d'identification de l'utilisateur que vous fournissez disposent des autorisations nécessaires pour accéder au compartiment. Voici un exemple de politique accordant les autorisations minimales requises pour l'accès avec Trident Protect. Vous pouvez appliquer cette politique à l'utilisateur qui gère les politiques de compartiment compatibles S3.

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

Pour plus d'informations sur les politiques Amazon S3, reportez-vous aux exemples dans la documentation. "Documentation Amazon S3" .

Identité du pod EKS pour l'authentification Amazon S3 (AWS)

Trident Protect prend en charge EKS Pod Identity pour les opérations de déplacement de données Kopia. Cette fonctionnalité permet un accès sécurisé aux buckets S3 sans stocker les informations d'identification AWS dans les secrets Kubernetes.

Configuration requise pour l'identification du pod EKS avec Trident Protect

Avant d'utiliser EKS Pod Identity avec Trident Protect, assurez-vous des points suivants :

- L'identité de pod est activée sur votre cluster EKS.
- Vous avez créé un rôle IAM avec les autorisations de compartiment S3 nécessaires. Pour en savoir plus, consultez "Autorisations IAM de stockage compatibles S3".
- Le rôle IAM est associé aux comptes de service Trident Protect suivants :
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Pour obtenir des instructions détaillées sur l'activation de Pod Identity et l'association des rôles IAM aux comptes de service, veuillez consulter la documentation. "Documentation sur l'identité des pods AWS EKS" .

Configuration d'AppVault Lorsque vous utilisez EKS Pod Identity, configurez votre ressource personnalisée AppVault avec le `useIAM: true` drapeau au lieu d'identifiants explicites :

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

Exemples de génération de clés AppVault pour les fournisseurs de cloud

Lors de la définition d'un AppVault CR, vous devez inclure des informations d'identification pour accéder aux ressources hébergées par le fournisseur, sauf si vous utilisez l'authentification IAM. La manière dont vous générez les clés pour les informations d'identification varie selon le fournisseur. Voici des exemples de génération de clés en ligne de commande pour plusieurs fournisseurs. Vous pouvez utiliser les exemples suivants pour créer des clés pour les informations d'identification de chaque fournisseur 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=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

Microsoft Azure

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

S3 générique

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

ONTAP S3

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

StorageGrid S3

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

Exemples de création d'AppVault

Vous trouverez ci-dessous des exemples de définitions AppVault pour chaque fournisseur.

Exemples de CR AppVault

Vous pouvez utiliser les exemples de CR suivants pour créer des objets AppVault pour chaque fournisseur de cloud.

- Vous pouvez éventuellement spécifier un secret Kubernetes contenant des mots de passe personnalisés pour le chiffrement des référentiels Restic et Kopia. Se référer àMots de passe du référentiel de déplacement de données pour plus d'informations.
- Pour les objets Amazon S3 (AWS) AppVault, vous pouvez éventuellement spécifier un jeton de session, ce qui est utile si vous utilisez l'authentification unique (SSO). Ce jeton est créé lorsque vous générez des clés pour le fournisseur dansExemples de génération de clés AppVault pour les fournisseurs de cloud .
- Pour les objets S3 AppVault, vous pouvez éventuellement spécifier une URL de proxy de sortie pour le trafic S3 sortant à l'aide de `spec.providerConfig.S3.proxyURL` clé.

Google Cloud

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

Amazon S3 (AWS)

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

Pour les environnements EKS utilisant Pod Identity avec Kopia Data Mover, vous pouvez supprimer le `providerCredentials` section et ajouter `useIAM: true` sous le `s3` plutôt une configuration.

Microsoft Azure

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

S3 générique

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

ONTAP S3

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

StorageGrid S3

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

Exemples de création d'AppVault à l'aide de l'interface de ligne de commande Trident Protect

Vous pouvez utiliser les exemples de commandes CLI suivants pour créer des demandes de changement AppVault pour chaque fournisseur.

- Vous pouvez éventuellement spécifier un secret Kubernetes contenant des mots de passe personnalisés pour le chiffrement des référentiels Restic et Kopia. Se référer àMots de passe du référentiel de déplacement de données pour plus d'informations.
- Pour les objets S3 AppVault, vous pouvez éventuellement spécifier une URL de proxy de sortie pour le trafic S3 sortant à l'aide de `--proxy-url <ip_address:port>` argument.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

S3 générique

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

ONTAP S3

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

StorageGrid S3

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

Consultez les informations d'AppVault

Vous pouvez utiliser le plugin CLI Trident Protect pour consulter les informations relatives aux objets AppVault que vous avez créés sur le cluster.

Étapes

1. Afficher le contenu d'un objet AppVault :

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

Exemple de résultat :

```
+-------------+-------+----------+-----------------------------
+--------------------------+
|   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. Pour afficher le chemin AppVaultPath de chaque ressource, vous pouvez également utiliser l'indicateur. `--show-paths` .

   Le nom du cluster dans la première colonne du tableau n'est disponible que si un nom de cluster a été spécifié dans l'installation Helm de Trident Protect. Par exemple: `--set clusterName=production1` .

Supprimer un AppVault

Vous pouvez supprimer un objet AppVault à tout moment.

> Ne retirez pas le `finalizers` Saisissez la clé dans la ressource personnalisée AppVault avant de supprimer l'objet AppVault. Si vous procédez ainsi, cela peut entraîner la présence de données résiduelles dans le compartiment AppVault et de ressources orphelines dans le cluster.

Avant de commencer

Assurez-vous d'avoir supprimé toutes les ressources de configuration (CR) de snapshot et de sauvegarde utilisées par l'AppVault que vous souhaitez supprimer.

Supprimer un AppVault à l'aide de l'interface de ligne de commande Kubernetes

1. Supprimez l'objet AppVault, en le remplaçant `appvault-name` avec le nom de l'objet AppVault à supprimer :

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

Supprimer un AppVault à l'aide de l'interface de ligne de commande Trident Protect

1. Supprimez l'objet AppVault, en le remplaçant `appvault-name` avec le nom de l'objet AppVault à supprimer :

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

Définissez une application de gestion avec Trident Protect

Vous pouvez définir une application que vous souhaitez gérer avec Trident Protect en créant une demande de changement d'application et une demande de changement AppVault associée.

Créer une demande de changement AppVault

Vous devez créer une ressource personnalisée AppVault qui sera utilisée lors des opérations de protection des données sur l'application, et cette ressource personnalisée AppVault doit résider sur le cluster où Trident Protect est installé. La demande de changement (CR) AppVault est spécifique à votre environnement ; pour des exemples de CR AppVault, reportez-vous à"Ressources personnalisées AppVault."

Définir une application

Vous devez définir chaque application que vous souhaitez gérer avec Trident Protect. Vous pouvez définir une application de gestion soit en créant manuellement une demande de changement d'application, soit en utilisant l'interface de ligne de commande (CLI) de Trident Protect.

Ajouter une application utilisant un CR

Étapes

1. Créez le fichier CR de l'application de destination :
   a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `maria-app.yaml`).
   b. Configurez les attributs suivants :
      - **metadata.name**: (*Obligatoire*) Le nom de la ressource personnalisée de l'application. Notez le nom que vous choisissez, car d'autres fichiers CR nécessaires aux opérations de protection font référence à cette valeur.
      - **spec.includedNamespaces**: (*Obligatoire*) Utilisez le sélecteur d'espace de noms et d'étiquette pour spécifier les espaces de noms et les ressources utilisés par l'application. L'espace de noms de l'application doit figurer dans cette liste. Le sélecteur d'étiquettes est facultatif et peut être utilisé pour filtrer les ressources au sein de chaque espace de noms spécifié.
      - **spec.includedClusterScopedResources**: (*Optionnel*) Utilisez cet attribut pour spécifier les ressources à portée de cluster à inclure dans la définition de l'application. Cet attribut vous permet de sélectionner ces ressources en fonction de leur groupe, de leur version, de leur type et de leurs étiquettes.
        - **groupVersionKind**: (*Obligatoire*) Spécifie le groupe d'API, la version et le type de la ressource à portée de cluster.
        - **labelSelector**: (*Optionnel*) Filtre les ressources à portée de cluster en fonction de leurs étiquettes.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Optional*) Cette annotation s'applique uniquement aux applications définies à partir de machines virtuelles, comme dans les environnements KubeVirt, où les gels du système de fichiers se produisent avant les instantanés. Indiquez si cette application peut écrire sur le système de fichiers lors d'un instantané. Si cette option est activée (true), l'application ignore le paramètre global et peut écrire sur le système de fichiers pendant la création d'un instantané. Si cette option est désactivée, l'application ignore le paramètre global et le système de fichiers est figé lors de la création d'un instantané. Si cette annotation est spécifiée mais que l'application ne comporte aucune machine virtuelle dans sa définition, elle est ignorée. Sauf indication contraire, la demande suit la procédure"Paramètre de gel global Trident Protect" .

Si vous devez appliquer cette annotation après la création d'une application, vous pouvez utiliser la commande suivante :

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

+
Exemple de YAML :

+

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

1. (*Facultatif*) Ajouter un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :

   - **resourceFilter.resourceSelectionCriteria** : (Requis pour le filtrage) Utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :

     - **resourceFilter.resourceMatchers** : Un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent selon une opération OU, et les champs à l'intérieur de chaque élément (groupe, type, version) correspondent selon une opération ET.
       - **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
       - **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.
       - **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.
       - **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.

- **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.
- **resourceMatchers[].labelSelectors** : (*Facultatif*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes, telle que définie dans le "Documentation Kubernetes" . Par exemple: `"trident.netapp.io/os=linux"` .

> ⓘ Lorsque les deux `resourceFilter` et `labelSelector` sont utilisés, `resourceFilter` court d'abord, puis ensuite `labelSelector` est appliquée aux ressources résultantes.

Par exemple:

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

2. Après avoir créé la demande de changement (CR) d'application correspondant à votre environnement, appliquez-la. Par exemple:

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

Étapes

1. Créez et appliquez la définition de l'application en utilisant l'un des exemples suivants, en remplaçant les valeurs entre crochets par les informations de votre environnement. Vous pouvez inclure des espaces de noms et des ressources dans la définition de l'application en utilisant des listes séparées par des virgules avec les arguments indiqués dans les exemples.

   Vous pouvez, si vous le souhaitez, utiliser une annotation lors de la création d'une application pour spécifier si celle-ci peut écrire sur le système de fichiers pendant la prise d'un instantané. Ceci ne s'applique qu'aux applications définies à partir de machines virtuelles, comme dans les environnements KubeVirt, où des blocages du système de fichiers se produisent avant la création des instantanés. Si vous définissez l'annotation sur `true` L'application ignore le paramètre global et peut écrire sur le système de fichiers pendant la création d'un instantané. Si vous le configurez à `false`

L'application ignore alors le paramètre global et le système de fichiers est figé lors de la prise d'un instantané. Si vous utilisez l'annotation mais que l'application ne comporte aucune machine virtuelle dans sa définition, l'annotation est ignorée. Si vous n'utilisez pas l'annotation, l'application suit le comportement attendu."Paramètre de gel global Trident Protect" .

Pour spécifier l'annotation lorsque vous utilisez l'interface de ligne de commande pour créer une application, vous pouvez utiliser la `--annotation` drapeau.

- Créez l'application et utilisez le paramètre global pour le comportement de gel du système de fichiers :

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

- Créez l'application et configurez les paramètres locaux de l'application pour le comportement de gel du système de fichiers :

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

Vous pouvez utiliser `--resource-filter-include` et `--resource-filter-exclude` indicateurs permettant d'inclure ou d'exclure des ressources en fonction de `resourceSelectionCriteria` tels que le groupe, le type, la version, les étiquettes, les noms et les espaces de noms, comme illustré dans l'exemple suivant :

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

Protégez les applications à l'aide de Trident Protect

Vous pouvez protéger toutes les applications gérées par Trident Protect en prenant des instantanés et des sauvegardes à l'aide d'une politique de protection automatisée ou de manière ponctuelle.

Vous pouvez configurer Trident Protect pour geler et dégeler les systèmes de fichiers pendant les opérations de protection des données. "Apprenez-en davantage sur la configuration du gel du système de fichiers avec Trident Protect.".

Créer un instantané à la demande

Vous pouvez créer un instantané à la demande à tout moment.

Les ressources à portée de cluster sont incluses dans une sauvegarde, un instantané ou un clone si elles sont explicitement référencées dans la définition de l'application ou si elles font référence à l'un des espaces de noms de l'application.

Créez un instantané à l'aide d'une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-snapshot-cr.yaml` .
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.applicationRef** : Le nom Kubernetes de l'application à capturer.
   - **spec.appVaultRef**: (*Obligatoire*) Le nom de l'AppVault où le contenu de l'instantané (métadonnées) doit être stocké.
   - **spec.reclaimPolicy**: (*Optionnel*) Définit ce qui arrive à l'AppArchive d'un instantané lorsque le CR de l'instantané est supprimé. Cela signifie que même lorsqu'il est réglé sur `Retain` , le cliché sera supprimé. Options valides :
     - `Retain`(défaut)
     - `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. Après avoir rempli le `trident-protect-snapshot-cr.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Créez un instantané à l'aide de l'interface de ligne de commande (CLI).

Étapes

1. Créez l'instantané en remplaçant les valeurs entre crochets par les informations provenant de votre environnement. Par exemple:

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

Créer une sauvegarde à la demande

Vous pouvez sauvegarder une application à tout moment.

> Les ressources à portée de cluster sont incluses dans une sauvegarde, un instantané ou un clone si elles sont explicitement référencées dans la définition de l'application ou si elles font référence à l'un des espaces de noms de l'application.

Avant de commencer

Assurez-vous que la durée de validité du jeton de session AWS est suffisante pour toute opération de sauvegarde S3 de longue durée. Si le jeton expire pendant l'opération de sauvegarde, l'opération peut échouer.

- Se référer à "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Se référer à "Documentation AWS IAM" pour plus d'informations sur les identifiants avec les ressources AWS.

Créez une sauvegarde à l'aide d'une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-backup-cr.yaml` .
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.applicationRef**: (*Obligatoire*) Le nom Kubernetes de l'application à sauvegarder.
   - **spec.appVaultRef**: (*Obligatoire*) Le nom de l'AppVault où le contenu de la sauvegarde doit être stocké.
   - **spec.dataMover**: (*Optionnel*) Une chaîne de caractères indiquant l'outil de sauvegarde à utiliser pour l'opération de sauvegarde. Valeurs possibles (sensible à la casse) :
     - `Restic`
     - `Kopia`(défaut)
   - **spec.reclaimPolicy**: (*Optional*) Définit ce qui arrive à une sauvegarde lorsqu'elle est libérée de sa revendication. Valeurs possibles :
     - `Delete`
     - `Retain`(défaut)
   - **spec.snapshotRef**: (*Optionnel*): Nom du snapshot à utiliser comme source de la sauvegarde. Si aucune information n'est fournie, un instantané temporaire sera créé et sauvegardé.
   - **metadata.annotations.protect.trident.netapp.io/full-backup** : (*Optionnel*) Cette annotation est utilisée pour spécifier si une sauvegarde doit être non incrémentale. Par défaut, toutes les sauvegardes sont incrémentielles. Toutefois, si cette annotation est définie sur `true` , la sauvegarde devient non incrémentale. Si aucune option n'est spécifiée, la sauvegarde suit le paramètre de sauvegarde incrémentielle par défaut. Il est recommandé d'effectuer périodiquement une sauvegarde complète, puis d'effectuer des sauvegardes incrémentales entre les sauvegardes complètes afin de minimiser les risques liés aux restaurations.

     Exemple de YAML :

```yaml
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
  annotations:
    protect.trident.netapp.io/full-backup: "true"
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. Après avoir rempli le `trident-protect-backup-cr.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Créez une sauvegarde à l'aide de l'interface de ligne de commande (CLI).

Étapes

1. Créez la sauvegarde en remplaçant les valeurs entre crochets par les informations provenant de votre environnement. Par exemple:

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

   Vous pouvez utiliser, si vous le souhaitez, le `--full-backup` indicateur permettant de spécifier si une sauvegarde doit être non incrémentale. Par défaut, toutes les sauvegardes sont incrémentielles. Lorsque cette option est utilisée, la sauvegarde devient non incrémentale. Il est recommandé d'effectuer périodiquement une sauvegarde complète, puis d'effectuer des sauvegardes incrémentales entre les sauvegardes complètes afin de minimiser les risques liés aux restaurations.

Élaborer un calendrier de protection des données

Une politique de protection protège une application en créant des instantanés, des sauvegardes ou les deux selon un calendrier défini. Vous pouvez choisir de créer des instantanés et des sauvegardes toutes les heures, tous les jours, toutes les semaines et tous les mois, et vous pouvez spécifier le nombre de copies à conserver. Vous pouvez planifier une sauvegarde complète non incrémentielle en utilisant l'annotation full-backup-rule. Par défaut, toutes les sauvegardes sont incrémentielles. L'exécution périodique d'une sauvegarde complète, ainsi que de sauvegardes incrémentielles entre les deux, permet de réduire le risque associé aux restaurations.

- Vous pouvez créer des planifications pour les instantanés uniquement en configurant `backupRetention` à zéro et `snapshotRetention` à une valeur supérieure à zéro. Paramètre `snapshotRetention` La valeur zéro signifie que les sauvegardes planifiées continueront à créer des instantanés, mais ceux-ci seront temporaires et supprimés immédiatement une fois la sauvegarde terminée.
- Les ressources à portée de cluster sont incluses dans une sauvegarde, un instantané ou un clone si elles sont explicitement référencées dans la définition de l'application ou si elles font référence à l'un des espaces de noms de l'application.

Créez un planning à l'aide d'une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-schedule-cr.yaml` .
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.dataMover**: (*Optionnel*) Une chaîne de caractères indiquant l'outil de sauvegarde à utiliser pour l'opération de sauvegarde. Valeurs possibles (sensible à la casse) :
     - `Restic`
     - `Kopia`(défaut)
   - **spec.applicationRef** : Nom Kubernetes de l'application à sauvegarder.
   - **spec.appVaultRef**: (*Obligatoire*) Le nom de l'AppVault où le contenu de la sauvegarde doit être stocké.
   - **spec.backupRetention**: Le nombre de sauvegardes à conserver. Zéro indique qu'aucune sauvegarde ne doit être créée (instantanés uniquement).
   - **spec.snapshotRetention** : Le nombre d'instantanés à conserver. La valeur zéro indique qu'aucun instantané ne doit être créé.
   - **spec.granularity** : La fréquence à laquelle la planification doit s'exécuter. Valeurs possibles, ainsi que les champs associés obligatoires :
     - `Hourly`(nécessite que vous précisiez `spec.minute`)
     - `Daily`(nécessite que vous précisiez `spec.minute` et `spec.hour`)
     - `Weekly`(nécessite que vous précisiez `spec.minute, spec.hour` , et `spec.dayOfWeek`)
     - `Monthly`(nécessite que vous précisiez `spec.minute, spec.hour` , et `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (*Facultatif*) Le jour du mois (1 - 31) auquel la planification doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `Monthly` . La valeur doit être fournie sous forme de chaîne.
   - **spec.dayOfWeek**: (*Facultatif*) Le jour de la semaine (0 - 7) pendant lequel la planification doit s'exécuter. Les valeurs de 0 ou 7 indiquent dimanche. Ce champ est obligatoire si la granularité est définie sur `Weekly` . La valeur doit être fournie sous forme de chaîne.
   - **spec.hour**: (*Facultatif*) L'heure de la journée (0 - 23) à laquelle le programme doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `Daily` , `Weekly` , ou `Monthly` . La valeur doit être fournie sous forme de chaîne.
   - **spec.minute**: (*Facultatif*) La minute de l'heure (0 - 59) à laquelle la planification doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `Hourly` , `Daily` , `Weekly` , ou `Monthly` . La valeur doit être fournie sous forme de chaîne.
   - **metadata.annotations.protect.trident.netapp.io/full-backup-rule**: (*Optionnel*) Cette annotation est utilisée pour spécifier la règle de planification de la sauvegarde complète. Vous pouvez le paramétrer pour `always` pour une sauvegarde complète et permanente ou personnalisez-la en fonction de vos besoins. Par exemple, si vous choisissez une granularité quotidienne, vous

pouvez spécifier les jours de la semaine où la sauvegarde complète doit avoir lieu.

Exemple de YAML pour la planification de sauvegarde et de snapshot :

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

Exemple de YAML pour la planification des instantanés uniquement :

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

3. Après avoir rempli le `trident-protect-schedule-cr.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Créez une planification à l'aide de l'interface de ligne de commande (CLI).

Étapes

1. Créez le calendrier de protection en remplaçant les valeurs entre parenthèses par les informations provenant de votre environnement. Par exemple:

   Vous pouvez utiliser `tridentctl-protect create schedule --help` pour afficher des informations d'aide détaillées sur cette commande.

   ```
   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>
   --full-backup-rule <string>
   ```

   Vous pouvez paramétrer le `--full-backup-rule` drapeau à `always` pour une sauvegarde complète et permanente ou personnalisez-la en fonction de vos besoins. Par exemple, si vous choisissez une granularité quotidienne, vous pouvez spécifier les jours de la semaine où la sauvegarde complète doit avoir lieu. Par exemple, utilisez `--full-backup-rule "Monday,Thursday"` programmer une sauvegarde complète les lundis et jeudis.

   Pour les planifications ne prenant en compte que les instantanés, définissez `--backup-retention 0` et spécifiez une valeur supérieure à 0 pour `--snapshot-retention` .

Supprimer un instantané

Supprimez les instantanés planifiés ou à la demande dont vous n'avez plus besoin.

Étapes

1. Supprimez la ressource personnalisée (CR) associée à l'instantané :

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

Supprimer une sauvegarde

Supprimez les sauvegardes planifiées ou à la demande dont vous n'avez plus besoin.

Assurez-vous que la politique de réclamation est configurée pour `Delete` supprimer toutes les données de sauvegarde du stockage d'objets. Le paramètre par défaut de la politique est `Retain` pour éviter toute perte accidentelle de données. Si la politique n'est pas modifiée à `Delete` Les données de sauvegarde resteront stockées dans l'objet et devront être supprimées manuellement.

Étapes

1. Supprimez la CR de sauvegarde associée à la sauvegarde :

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

Vérifier l'état d'une opération de sauvegarde

Vous pouvez utiliser la ligne de commande pour vérifier l'état d'une opération de sauvegarde : en cours, terminée ou ayant échoué.

Étapes

1. Utilisez la commande suivante pour récupérer l'état de l'opération de sauvegarde, en remplaçant les valeurs entre parenthèses par les informations provenant de votre environnement :

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

Activer la sauvegarde et la restauration des opérations azure-netapp-files (ANF)

Si vous avez installé Trident Protect, vous pouvez activer une fonctionnalité de sauvegarde et de restauration économe en espace pour les backends de stockage qui utilisent la classe de stockage azure-netapp-files et qui ont été créés avant Trident 24.06. Cette fonctionnalité est compatible avec les volumes NFSv4 et ne consomme pas d'espace supplémentaire du pool de capacité.

Avant de commencer

Assurez-vous de ce qui suit :

- Vous avez installé Trident Protect.
- Vous avez défini une application dans Trident Protect. Cette application offrira des fonctionnalités de protection limitées jusqu'à ce que vous ayez terminé cette procédure.
- Tu as `azure-netapp-files` sélectionné comme classe de stockage par défaut pour votre système de stockage dorsal.

Développez pour afficher les étapes de configuration

1. Procédez comme suit dans Trident si le volume ANF a été créé avant la mise à niveau vers Trident 24.10 :

   a. Activez le répertoire de snapshots pour chaque PV basé sur azure-netapp-files et associé à l'application :

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

   b. Vérifiez que le répertoire des instantanés a été activé pour chaque PV associé :

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

   Réponse:

   ```
   snapshotDirectory: "true"
   ```

   +
   Lorsque le répertoire de snapshots n'est pas activé, le système choisit la fonctionnalité de sauvegarde standard, qui consomme temporairement de l'espace dans le pool de capacité pendant le processus de sauvegarde. Dans ce cas, assurez-vous de disposer d'un espace suffisant dans le pool de capacité pour créer un volume temporaire de la taille du volume sauvegardé.

Résultat

L'application est prête pour la sauvegarde et la restauration à l'aide de Trident Protect. Chaque PVC peut également être utilisé par d'autres applications pour les sauvegardes et les restaurations.

Restaurer les applications

Restaurez les applications à l'aide de Trident Protect

Vous pouvez utiliser Trident Protect pour restaurer votre application à partir d'un instantané ou d'une sauvegarde. La restauration à partir d'un instantané existant sera plus rapide lors de la restauration de l'application sur le même cluster.

- Lorsque vous restaurez une application, tous les points d'exécution configurés pour celle-ci sont restaurés avec elle. Si un point d'entrée d'exécution post-restauration est présent, il s'exécute automatiquement dans le cadre de l'opération de restauration.
- La restauration à partir d'une sauvegarde vers un espace de noms différent ou vers l'espace de noms d'origine est prise en charge pour les volumes qtree. Cependant, la restauration à partir d'un instantané vers un espace de noms différent ou vers l'espace de noms d'origine n'est pas prise en charge pour les volumes qtree.
- Vous pouvez utiliser des paramètres avancés pour personnaliser les opérations de restauration. Pour en savoir plus, consultez "Utilisez les paramètres de restauration avancés de Trident Protect".

Restaurer à partir d'une sauvegarde vers un espace de noms différent

Lorsque vous restaurez une sauvegarde dans un espace de noms différent à l'aide d'une ressource personnalisée BackupRestore, Trident Protect restaure l'application dans un nouvel espace de noms et crée une ressource personnalisée d'application pour l'application restaurée. Pour protéger l'application restaurée, créez des sauvegardes ou des instantanés à la demande, ou établissez un calendrier de protection.

La restauration d'une sauvegarde dans un espace de noms différent contenant des ressources existantes ne modifiera pas les ressources qui partagent les mêmes noms que celles de la sauvegarde. Pour restaurer toutes les ressources de la sauvegarde, supprimez et recréez l'espace de noms cible, ou restaurez la sauvegarde dans un nouvel espace de noms.

Avant de commencer

Assurez-vous que la durée d'expiration du jeton de session AWS est suffisante pour toute opération de restauration S3 de longue durée. Si le jeton expire pendant l'opération de restauration, celle-ci peut échouer.

- Se référer à "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Se référer à "Documentation AWS IAM" pour plus d'informations sur les identifiants avec les ressources AWS.

Lorsque vous restaurez des sauvegardes à l'aide de Kopia comme outil de transfert de données, vous pouvez éventuellement spécifier des annotations dans le CR ou utiliser l'interface de ligne de commande pour contrôler le comportement du stockage temporaire utilisé par Kopia. Se référer à "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez le `tridentctl-protect create --help` commande pour plus d'informations sur la spécification des annotations avec l'interface de ligne de commande Trident Protect.

Utilisez un CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-backup-restore-cr.yaml` .
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appArchivePath** : Chemin d'accès dans AppVault où sont stockés les contenus de sauvegarde. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

   - **spec.appVaultRef**: (*Obligatoire*) Le nom de l'AppVault où sont stockés les contenus de sauvegarde.
   - **spec.namespaceMapping** : Le mappage de l'espace de noms source de l'opération de restauration vers l'espace de noms de destination. Remplacer `my-source-namespace` et `my-destination-namespace` avec des informations provenant de votre environnement.

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

3. (*Facultatif*) Si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées par des étiquettes particulières :

   > Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

   - **resourceFilter.resourceSelectionCriteria** : (Requis pour le filtrage) Utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
     - **resourceFilter.resourceMatchers** : Un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent selon une opération OU, et les champs

à l'intérieur de chaque élément (groupe, type, version) correspondent selon une opération ET.

- **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
- **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.
- **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.
- **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.
- **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.
- **resourceMatchers[].labelSelectors** : (*Facultatif*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes, telle que définie dans le "Documentation Kubernetes" . Par exemple: `"trident.netapp.io/os=linux"` .

  Par exemple:

```
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. Après avoir rempli le `trident-protect-backup-restore-cr.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Utiliser la CLI

Étapes

1. Restaurez la sauvegarde dans un espace de noms différent, en remplaçant les valeurs entre crochets par les informations de votre environnement. Le `namespace-mapping` L'argument utilise des espaces de noms séparés par deux-points pour associer les espaces de noms source aux espaces de noms de destination corrects au format `source1:dest1,source2:dest2` . Par exemple:

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

Restaurer à partir d'une sauvegarde vers l'espace de noms d'origine

Vous pouvez restaurer une sauvegarde dans l'espace de noms d'origine à tout moment.

Avant de commencer

Assurez-vous que la durée d'expiration du jeton de session AWS est suffisante pour toute opération de restauration S3 de longue durée. Si le jeton expire pendant l'opération de restauration, celle-ci peut échouer.

- Se référer à "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Se référer à "Documentation AWS IAM" pour plus d'informations sur les identifiants avec les ressources AWS.

> Lorsque vous restaurez des sauvegardes à l'aide de Kopia comme outil de transfert de données, vous pouvez éventuellement spécifier des annotations dans le CR ou utiliser l'interface de ligne de commande pour contrôler le comportement du stockage temporaire utilisé par Kopia. Se référer à "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez le `tridentctl-protect create --help` commande pour plus d'informations sur la spécification des annotations avec l'interface de ligne de commande Trident Protect.

Utilisez un CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-backup-ipr-cr.yaml` .
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appArchivePath** : Chemin d'accès dans AppVault où sont stockés les contenus de sauvegarde. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

   - **spec.appVaultRef**: (*Obligatoire*) Le nom de l'AppVault où sont stockés les contenus de sauvegarde.

     Par exemple:

```
---
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. (*Facultatif*) Si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées par des étiquettes particulières :

   Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

   - **resourceFilter.resourceSelectionCriteria** : (Requis pour le filtrage) Utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
     - **resourceFilter.resourceMatchers** : Un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent selon une opération OU, et les champs à l'intérieur de chaque élément (groupe, type, version) correspondent selon une opération ET.
       - **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
       - **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.

- **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.
- **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.
- **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.
- **resourceMatchers[].labelSelectors** : (*Facultatif*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes, telle que définie dans le "Documentation Kubernetes" . Par exemple: `"trident.netapp.io/os=linux"` .

  Par exemple:

```
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. Après avoir rempli le `trident-protect-backup-ipr-cr.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Utiliser la CLI

Étapes

1. Restaurez la sauvegarde dans l'espace de noms d'origine, en remplaçant les valeurs entre crochets par les informations de votre environnement. Le `backup` L'argument utilise un espace de noms et un nom de sauvegarde au format `<namespace>/<name>` . Par exemple:

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

Restaurer à partir d'une sauvegarde vers un cluster différent

Vous pouvez restaurer une sauvegarde sur un autre cluster en cas de problème avec le cluster d'origine.

> Lorsque vous restaurez des sauvegardes à l'aide de Kopia comme outil de transfert de données, vous pouvez éventuellement spécifier des annotations dans le CR ou utiliser l'interface de ligne de commande pour contrôler le comportement du stockage temporaire utilisé par Kopia. Se référer à "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez le `tridentctl-protect create --help` commande pour plus d'informations sur la spécification des annotations avec l'interface de ligne de commande Trident Protect.

Avant de commencer

Assurez-vous que les conditions préalables suivantes sont remplies :

- Le cluster de destination possède Trident Protect installé.
- Le cluster de destination a accès au chemin du compartiment du même AppVault que le cluster source, où la sauvegarde est stockée.
- Assurez-vous que votre environnement local peut se connecter au compartiment de stockage d'objets défini dans la ressource personnalisée AppVault lors de l'exécution. `tridentctl-protect get appvaultcontent` commande. Si des restrictions réseau empêchent l'accès, exécutez plutôt l'interface de ligne de commande Trident Protect depuis un pod sur le cluster de destination.
- Assurez-vous que la durée de validité du jeton de session AWS est suffisante pour toute opération de restauration de longue durée. Si le jeton expire pendant l'opération de restauration, celle-ci peut échouer.
  - Se référer à "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
  - Se référer à "Documentation AWS" pour plus d'informations sur les identifiants avec les ressources AWS.

Étapes

1. Vérifiez la disponibilité de la ressource personnalisée AppVault sur le cluster de destination à l'aide du plugin CLI Trident Protect :

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

   > Assurez-vous que l'espace de noms destiné à la restauration de l'application existe sur le cluster de destination.

2. Consultez le contenu de la sauvegarde de l'AppVault disponible sur le cluster de destination :

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

   L'exécution de cette commande affiche les sauvegardes disponibles dans AppVault, y compris leurs clusters d'origine, les noms des applications correspondantes, les horodatages et les chemins d'accès aux

archives.

Exemple de résultat :

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   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. Restaurez l’application sur le cluster de destination en utilisant le nom AppVault et le chemin d’accès à l’archive :

Utilisez un CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-backup-restore-cr.yaml` .
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appVaultRef**: (*Obligatoire*) Le nom de l'AppVault où sont stockés les contenus de sauvegarde.
   - **spec.appArchivePath** : Chemin d'accès dans AppVault où sont stockés les contenus de sauvegarde. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

     Si BackupRestore CR n'est pas disponible, vous pouvez utiliser la commande mentionnée à l'étape 2 pour afficher le contenu de la sauvegarde.

   - **spec.namespaceMapping** : Le mappage de l'espace de noms source de l'opération de restauration vers l'espace de noms de destination. Remplacer `my-source-namespace` et `my-destination-namespace` avec des informations provenant de votre environnement.

     Par exemple:

```
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. Après avoir rempli le `trident-protect-backup-restore-cr.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Utiliser la CLI

1. Utilisez la commande suivante pour restaurer l'application, en remplaçant les valeurs entre crochets par les informations provenant de votre environnement. L'argument namespace-mapping utilise des

espaces de noms séparés par deux-points pour mapper les espaces de noms sources aux espaces de noms de destination corrects au format source1:dest1,source2:dest2. Par exemple:

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--context <destination_cluster_name> \
-n <application_namespace>
```

Restaurer à partir d'un instantané vers un espace de noms différent

Vous pouvez restaurer des données à partir d'un instantané à l'aide d'un fichier de ressources personnalisé (CR), soit dans un espace de noms différent, soit dans l'espace de noms source d'origine. Lorsque vous restaurez un instantané dans un espace de noms différent à l'aide d'une ressource personnalisée SnapshotRestore, Trident Protect restaure l'application dans un nouvel espace de noms et crée une ressource personnalisée d'application pour l'application restaurée. Pour protéger l'application restaurée, créez des sauvegardes ou des instantanés à la demande, ou établissez un calendrier de protection.

SnapshotRestore prend en charge `spec.storageClassMapping` cet attribut est valable uniquement lorsque les classes de stockage source et de destination utilisent le même système de stockage dorsal. Si vous tentez de restaurer un `StorageClass` Si le système de stockage utilisé est différent, l'opération de restauration échouera.

Avant de commencer

Assurez-vous que la durée d'expiration du jeton de session AWS est suffisante pour toute opération de restauration S3 de longue durée. Si le jeton expire pendant l'opération de restauration, celle-ci peut échouer.

- Se référer à "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Se référer à "Documentation AWS IAM" pour plus d'informations sur les identifiants avec les ressources AWS.

Utilisez un CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-snapshot-restore-cr.yaml` .
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appVaultRef**: (*Obligatoire*) Le nom de l'AppVault où le contenu de l'instantané est stocké.
   - **spec.appArchivePath** : Le chemin d'accès dans AppVault où le contenu des instantanés est stocké. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

   - **spec.namespaceMapping** : Le mappage de l'espace de noms source de l'opération de restauration vers l'espace de noms de destination. Remplacer `my-source-namespace` et `my-destination-namespace` avec des informations provenant de votre environnement.

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

3. (*Facultatif*) Si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées par des étiquettes particulières :

   > Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

   - **resourceFilter.resourceSelectionCriteria** : (Requis pour le filtrage) Utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
     - **resourceFilter.resourceMatchers** : Un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent selon une opération OU, et les champs à l'intérieur de chaque élément (groupe, type, version) correspondent selon une opération ET.

- **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
- **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.
- **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.
- **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.
- **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.
- **resourceMatchers[].labelSelectors** : (*Facultatif*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes, telle que définie dans le "Documentation Kubernetes" . Par exemple: `"trident.netapp.io/os=linux"` .

  Par exemple:

```
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. Après avoir rempli le `trident-protect-snapshot-restore-cr.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Utiliser la CLI

Étapes

1. Restaurez l'instantané dans un espace de noms différent, en remplaçant les valeurs entre crochets par les informations de votre environnement.

   - Le `snapshot` L'argument utilise un espace de noms et un nom d'instantané au format `<namespace>/<name>` .
   - Le `namespace-mapping` L'argument utilise des espaces de noms séparés par deux-points pour associer les espaces de noms source aux espaces de noms de destination corrects au format

`source1:dest1,source2:dest2`.

Par exemple:

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

Restaurer à partir d'un instantané vers l'espace de noms d'origine

Vous pouvez restaurer un instantané dans l'espace de noms d'origine à tout moment.

> Si votre application utilise plusieurs espaces de noms et que ces espaces de noms contiennent des PVC portant le même nom, les opérations de restauration d'instantané (sur place et vers un nouvel espace de noms) ne fonctionneront pas correctement. Tous les volumes restaurés auront les mêmes données au lieu des données correctes pour chaque espace de noms. Utilisez la restauration de sauvegarde plutôt que la restauration d'instantané, ou effectuez une mise à niveau vers la version 26.02 ou ultérieure qui corrige ce problème.

Avant de commencer

Assurez-vous que la durée d'expiration du jeton de session AWS est suffisante pour toute opération de restauration S3 de longue durée. Si le jeton expire pendant l'opération de restauration, celle-ci peut échouer.

- Se référer à "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Se référer à "Documentation AWS IAM" pour plus d'informations sur les identifiants avec les ressources AWS.

Utilisez un CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-snapshot-ipr-cr.yaml` .
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appVaultRef**: (*Obligatoire*) Le nom de l'AppVault où le contenu de l'instantané est stocké.
   - **spec.appArchivePath** : Le chemin d'accès dans AppVault où le contenu des instantanés est stocké. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

     ```
     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. (*Facultatif*) Si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées par des étiquettes particulières :

   > Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

   - **resourceFilter.resourceSelectionCriteria** : (Requis pour le filtrage) Utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
     - **resourceFilter.resourceMatchers** : Un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent selon une opération OU, et les champs à l'intérieur de chaque élément (groupe, type, version) correspondent selon une opération ET.
       - **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
       - **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.
       - **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.
       - **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.

- **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.
- **resourceMatchers[].labelSelectors** : (*Facultatif*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes, telle que définie dans le "Documentation Kubernetes" . Par exemple: `"trident.netapp.io/os=linux"` .

  Par exemple:

```
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. Après avoir rempli le `trident-protect-snapshot-ipr-cr.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Utiliser la CLI

Étapes

1. Restaurez l'instantané dans l'espace de noms d'origine, en remplaçant les valeurs entre crochets par les informations de votre environnement. Par exemple:

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

Vérifier l'état d'une opération de restauration

Vous pouvez utiliser la ligne de commande pour vérifier l'état d'une opération de restauration : en cours, terminée ou ayant échoué.

Étapes

1. Utilisez la commande suivante pour récupérer l'état de l'opération de restauration, en remplaçant les valeurs entre parenthèses par les informations provenant de votre environnement :

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

Utilisez les paramètres de restauration avancés de Trident Protect

Vous pouvez personnaliser les opérations de restauration à l'aide de paramètres avancés tels que les annotations, les paramètres d'espace de noms et les options de stockage pour répondre à vos besoins spécifiques.

Annotations et étiquettes d'espace de noms lors des opérations de restauration et de basculement

Lors des opérations de restauration et de basculement, les étiquettes et annotations de l'espace de noms de destination sont mises en correspondance avec celles de l'espace de noms source. Les étiquettes ou annotations de l'espace de noms source qui n'existent pas dans l'espace de noms de destination sont ajoutées, et toutes les étiquettes ou annotations existantes sont écrasées pour correspondre à la valeur de l'espace de noms source. Les étiquettes ou annotations qui existent uniquement dans l'espace de noms de destination restent inchangées.

> Si vous utilisez Red Hat OpenShift, il est important de noter le rôle essentiel des annotations d'espace de noms dans les environnements OpenShift. Les annotations d'espace de noms garantissent que les pods restaurés adhèrent aux autorisations et aux configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (SCC) OpenShift et peuvent accéder aux volumes sans problèmes d'autorisation. Pour plus d'informations, veuillez consulter le "Documentation sur les contraintes de contexte de sécurité d'OpenShift" .

Vous pouvez empêcher l'écrasement de certaines annotations dans l'espace de noms de destination en définissant la variable d'environnement Kubernetes. `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` avant d'effectuer l'opération de restauration ou de basculement. Par exemple:

```
helm upgrade trident-protect --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

> Lors d'une opération de restauration ou de basculement, toutes les annotations et étiquettes d'espace de noms spécifiées dans `restoreSkipNamespaceAnnotations` et `restoreSkipNamespaceLabels` sont exclues de l'opération de restauration ou de basculement. Assurez-vous que ces paramètres sont configurés lors de l'installation initiale de Helm. Pour en savoir plus, consultez "Configurer les options de filtrage AutoSupport et d'espace de noms".

Si vous avez installé l'application source à l'aide de Helm avec le `--create-namespace` Le drapeau, un traitement spécial est accordé au `name` Légende. Lors du processus de restauration ou de basculement, Trident Protect copie cette étiquette dans l'espace de noms de destination, mais met à jour la valeur avec la

valeur de l'espace de noms de destination si la valeur de la source correspond à l'espace de noms source. Si cette valeur ne correspond pas à l'espace de noms source, elle est copiée dans l'espace de noms de destination sans modification.

Exemple

L'exemple suivant présente un espace de noms source et un espace de noms de destination, chacun avec des annotations et des étiquettes différentes. Vous pouvez voir l'état de l'espace de noms de destination avant et après l'opération, et comment les annotations et les étiquettes sont combinées ou écrasées dans l'espace de noms de destination.

Avant l'opération de restauration ou de basculement

Le tableau suivant illustre l'état des espaces de noms source et de destination de l'exemple avant l'opération de restauration ou de basculement :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-1 (source)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environnement=production • conformité=hippaa • nom=ns-1
Espace de noms ns-2 (destination)	• annotation.one/key: "true" • annotation.three/key: "false"	• rôle=base de données

Après l'opération de restauration

Le tableau suivant illustre l'état de l'espace de noms de destination d'exemple après l'opération de restauration ou de basculement. Des touches ont été ajoutées, d'autres ont été écrasées, et le `name` L'étiquette a été mise à jour pour correspondre à l'espace de noms de destination :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-2 (destination)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• nom=ns-2 • conformité=hippaa • environnement=production • rôle=base de données

Champs pris en charge

Cette section décrit les champs supplémentaires disponibles pour les opérations de restauration.

Mappage des classes de stockage

Le `spec.storageClassMapping` L'attribut définit un mappage entre une classe de stockage présente dans l'application source et une nouvelle classe de stockage sur le cluster cible. Vous pouvez l'utiliser lors de la migration d'applications entre des clusters avec différentes classes de stockage ou lors du changement du backend de stockage pour les opérations BackupRestore.

Exemple:

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

Annotations prises en charge

Cette section répertorie les annotations prises en charge pour configurer différents comportements du système. Si aucune annotation n'est explicitement définie par l'utilisateur, le système utilisera la valeur par défaut.

Annotation	Type	Description	Valeur par défaut
protect.trident.netapp.io/data-mover-timeout-sec	chaîne	Le temps maximal (en secondes) pendant lequel l'opération de déplacement de données peut être bloquée.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	chaîne	La limite de taille maximale (en mégaoctets) pour le cache de contenu Kopia.	"1000"

Répliquez les applications à l'aide de NetApp SnapMirror et Trident Protect.

Avec Trident Protect, vous pouvez utiliser les capacités de réplication asynchrone de la technologie NetApp SnapMirror pour répliquer les données et les modifications d'applications d'un système de stockage à un autre, sur le même cluster ou entre différents clusters.

Annotations et étiquettes d'espace de noms lors des opérations de restauration et de basculement

Lors des opérations de restauration et de basculement, les étiquettes et annotations de l'espace de noms de destination sont mises en correspondance avec celles de l'espace de noms source. Les étiquettes ou annotations de l'espace de noms source qui n'existent pas dans l'espace de noms de destination sont ajoutées, et toutes les étiquettes ou annotations existantes sont écrasées pour correspondre à la valeur de l'espace de noms source. Les étiquettes ou annotations qui existent uniquement dans l'espace de noms de destination restent inchangées.

Si vous utilisez Red Hat OpenShift, il est important de noter le rôle essentiel des annotations d'espace de noms dans les environnements OpenShift. Les annotations d'espace de noms garantissent que les pods restaurés adhèrent aux autorisations et aux configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (SCC) OpenShift et peuvent accéder aux volumes sans problèmes d'autorisation. Pour plus d'informations, veuillez consulter le "Documentation sur les contraintes de contexte de sécurité d'OpenShift" .

Vous pouvez empêcher l'écrasement de certaines annotations dans l'espace de noms de destination en définissant la variable d'environnement Kubernetes. `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` avant d'effectuer l'opération de restauration ou de basculement. Par exemple:

```
helm upgrade trident-protect --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

> Lors d'une opération de restauration ou de basculement, toutes les annotations et étiquettes d'espace de noms spécifiées dans `restoreSkipNamespaceAnnotations` et `restoreSkipNamespaceLabels` sont exclues de l'opération de restauration ou de basculement. Assurez-vous que ces paramètres sont configurés lors de l'installation initiale de Helm. Pour en savoir plus, consultez "Configurer les options de filtrage AutoSupport et d'espace de noms".

Si vous avez installé l'application source à l'aide de Helm avec le `--create-namespace` Le drapeau, un traitement spécial est accordé au `name` Légende. Lors du processus de restauration ou de basculement, Trident Protect copie cette étiquette dans l'espace de noms de destination, mais met à jour la valeur avec la valeur de l'espace de noms de destination si la valeur de la source correspond à l'espace de noms source. Si cette valeur ne correspond pas à l'espace de noms source, elle est copiée dans l'espace de noms de destination sans modification.

Exemple

L'exemple suivant présente un espace de noms source et un espace de noms de destination, chacun avec des annotations et des étiquettes différentes. Vous pouvez voir l'état de l'espace de noms de destination avant et après l'opération, et comment les annotations et les étiquettes sont combinées ou écrasées dans l'espace de noms de destination.

Avant l'opération de restauration ou de basculement

Le tableau suivant illustre l'état des espaces de noms source et de destination de l'exemple avant l'opération de restauration ou de basculement :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-1 (source)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environnement=production • conformité=hippaa • nom=ns-1
Espace de noms ns-2 (destination)	• annotation.one/key: "true" • annotation.three/key: "false"	• rôle=base de données

Après l'opération de restauration

Le tableau suivant illustre l'état de l'espace de noms de destination d'exemple après l'opération de restauration ou de basculement. Des touches ont été ajoutées, d'autres ont été écrasées, et le `name` L'étiquette a été mise à jour pour correspondre à l'espace de noms de destination :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-2 (destination)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• nom=ns-2 • conformité=hippaa • environnement=production • rôle=base de données

Vous pouvez configurer Trident Protect pour geler et dégeler les systèmes de fichiers pendant les opérations de protection des données. "Apprenez-en davantage sur la configuration du gel du système de fichiers avec Trident Protect.".

Points d'exécution lors des opérations de basculement et d'inversion

Lorsque vous utilisez la relation AppMirror pour protéger votre application, il existe des comportements spécifiques liés aux points d'exécution dont vous devez tenir compte lors des opérations de basculement et de restauration.

- Lors d'un basculement, les points d'entrée d'exécution sont automatiquement copiés du cluster source vers le cluster de destination. Vous n'avez pas besoin de les recréer manuellement. Après le basculement, des points d'entrée d'exécution sont présents dans l'application et s'exécuteront lors de toute action pertinente.
- Lors d'une opération inverse ou d'une resynchronisation inverse, tous les points d'entrée d'exécution existants sur l'application sont supprimés. Lorsque l'application source devient l'application de destination, ces points d'ancrage d'exécution ne sont plus valides et sont supprimés pour empêcher leur exécution.

Pour en savoir plus sur les hooks d'exécution, consultez"Gérer les hooks d'exécution de Trident Protect" .

Établir une relation de réplication

La mise en place d'une relation de réplication implique les éléments suivants :

- Choisir la fréquence à laquelle Trident Protect doit prendre un instantané de l'application (qui inclut les ressources Kubernetes de l'application ainsi que les instantanés de volume pour chacun des volumes de l'application).
- Choix du calendrier de réplication (incluant les ressources Kubernetes ainsi que les données de volume persistant)
- Définir l'heure de la prise de vue

Étapes

1. Sur le cluster source, créez un AppVault pour l'application source. Selon votre fournisseur de stockage, modifiez un exemple dans"Ressources personnalisées AppVault" pour s'adapter à votre environnement :

Créez un AppVault à l'aide d'une demande de changement.

a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-appvault-primary-source.yaml`).

b. Configurez les attributs suivants :

- **metadata.name**: (*Obligatoire*) Le nom de la ressource personnalisée AppVault. Notez bien le nom que vous choisissez, car d'autres fichiers CR nécessaires à une relation de réplication font référence à cette valeur.
- **spec.providerConfig**: (*Obligatoire*) Stocke la configuration nécessaire pour accéder à AppVault à l'aide du fournisseur spécifié. Choisissez un nom de compartiment et toutes autres informations nécessaires pour votre fournisseur. Notez les valeurs que vous choisissez, car d'autres fichiers CR nécessaires à une relation de réplication font référence à ces valeurs. Se référer à"Ressources personnalisées AppVault" pour des exemples de demandes de changement AppVault avec d'autres fournisseurs.
- **spec.providerCredentials**: (*Obligatoire*) Stocke les références à toutes les informations d'identification requises pour accéder à AppVault à l'aide du fournisseur spécifié.
  - **spec.providerCredentials.valueFromSecret**: (*Obligatoire*) Indique que la valeur d'identification doit provenir d'un secret.
    - **clé**: (*Obligatoire*) La clé valide du secret à sélectionner.
    - **nom** : (*Obligatoire*) Nom du secret contenant la valeur de ce champ. Doit se trouver dans le même espace de noms.
  - **spec.providerCredentials.secretAccessKey**: (*Obligatoire*) La clé d'accès utilisée pour accéder au fournisseur. Le **nom** doit correspondre à **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obligatoire*) Détermine ce qui fournit la sauvegarde ; par exemple, NetApp ONTAP S3, S3 générique, Google Cloud ou Microsoft Azure. Valeurs possibles :
  - aws
  - azuré
  - GCP
  - générique-s3
  - ontap-s3
  - storagegrid-s3

c. Après avoir rempli le `trident-protect-appvault-primary-source.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Créez un AppVault à l'aide de l'interface de ligne de commande (CLI).

a. Créez l'AppVault en remplaçant les valeurs entre crochets par les informations provenant de votre environnement :

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

2. Sur le cluster source, créez l'application source CR :

Créez l'application source à l'aide d'une CR

a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-app-source.yaml`).

b. Configurez les attributs suivants :

- **metadata.name**: (*Obligatoire*) Le nom de la ressource personnalisée de l'application. Notez bien le nom que vous choisissez, car d'autres fichiers CR nécessaires à une relation de réplication font référence à cette valeur.
- **spec.includedNamespaces**: (*Obligatoire*) Un tableau d'espaces de noms et d'étiquettes associées. Utilisez les noms d'espaces de noms et, éventuellement, restreignez la portée des espaces de noms à l'aide d'étiquettes pour spécifier les ressources qui existent dans les espaces de noms listés ici. L'espace de noms de l'application doit faire partie de ce tableau.

  Exemple de fichier 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. Après avoir rempli le `trident-protect-app-source.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Créez l'application source à l'aide de l'interface de ligne de commande (CLI).

a. Créez l'application source. Par exemple:

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

3. Vous pouvez également, si vous le souhaitez, prendre un instantané de l'application source sur le cluster source. Cet instantané sert de base à l'application sur le cluster de destination. Si vous ignorez cette étape, vous devrez attendre la prochaine capture d'écran planifiée pour obtenir une version récente. Pour créer un instantané à la demande, reportez-vous à "Créer un instantané à la demande".

4. Sur le cluster source, créez la ressource personnalisée (CR) de planification de réplication :

En plus du calendrier fourni ci-dessous, il est recommandé de créer un calendrier de capture instantanée quotidienne distinct avec une période de conservation de 7 jours afin de maintenir une capture instantanée commune entre les clusters ONTAP appariés. Cela garantit la disponibilité des instantanés pendant une durée maximale de 7 jours, mais cette période de conservation peut être personnalisée en fonction des besoins de l'utilisateur.

En cas de basculement, le système peut utiliser ces instantanés pendant une durée maximale de 7 jours pour les opérations inverses. Cette approche rend le processus inverse plus rapide et plus efficace car seules les modifications apportées depuis le dernier instantané seront transférées, et non la totalité des données.

Si un calendrier existant pour la demande répond déjà aux exigences de conservation souhaitées, aucun calendrier supplémentaire n'est requis.

Créez la planification de réplication à l'aide d'une ressource personnalisée (CR).

a. Créez une planification de réplication pour l'application source :

i. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-schedule.yaml`).

ii. Configurez les attributs suivants :

- **metadata.name**: (*Obligatoire*) Le nom de la ressource personnalisée de planification.
- **spec.appVaultRef** : (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name de l'AppVault pour l'application source.
- **spec.applicationRef**: (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name de la ressource personnalisée (CR) de l'application source.
- **spec.backupRetention**: (*Obligatoire*) Ce champ est obligatoire et sa valeur doit être définie sur 0.
- **spec.enabled** : Doit être défini sur true.
- **spec.granularity** : Doit être défini sur `Custom` .
- **spec.recurrenceRule** : Définissez une date de début en temps UTC et un intervalle de récurrence.
- **spec.snapshotRetention** : Doit être défini sur 2.

Exemple de YAML :

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. Après avoir rempli le `trident-protect-schedule.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Créez la planification de réplication à l'aide de l'interface de ligne de commande (CLI).

a. Créez la planification de réplication en remplaçant les valeurs entre crochets par les informations provenant de votre environnement :

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

Exemple:

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

5. Sur le cluster de destination, créez une demande de changement (CR) AppVault d'application source identique à celle que vous avez appliquée sur le cluster source et nommez-la (par exemple, `trident-protect-appvault-primary-destination.yaml`).
6. Appliquer le CR :

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

7. Créez une ressource personnalisée AppVault de destination pour l'application de destination sur le cluster de destination. Selon votre fournisseur de stockage, modifiez un exemple dans"Ressources personnalisées AppVault" pour s'adapter à votre environnement :
   a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-appvault-secondary-destination.yaml`).
   b. Configurez les attributs suivants :
      - **metadata.name**: (*Obligatoire*) Le nom de la ressource personnalisée AppVault. Notez bien le nom que vous choisissez, car d'autres fichiers CR nécessaires à une relation de réplication font référence à cette valeur.
      - **spec.providerConfig**: (*Obligatoire*) Stocke la configuration nécessaire pour accéder à AppVault à l'aide du fournisseur spécifié. Choisissez un `bucketName` et toute autre information nécessaire à votre fournisseur. Notez les valeurs que vous choisissez, car d'autres fichiers CR nécessaires à une relation de réplication font référence à ces valeurs. Se référer à"Ressources personnalisées

AppVault" pour des exemples de demandes de changement AppVault avec d'autres fournisseurs.

- **spec.providerCredentials**: (*Obligatoire*) Stocke les références à toutes les informations d'identification requises pour accéder à AppVault à l'aide du fournisseur spécifié.
  - **spec.providerCredentials.valueFromSecret**: (*Obligatoire*) Indique que la valeur d'identification doit provenir d'un secret.
    - **clé**: (*Obligatoire*) La clé valide du secret à sélectionner.
    - **nom** : (*Obligatoire*) Nom du secret contenant la valeur de ce champ. Doit se trouver dans le même espace de noms.
  - **spec.providerCredentials.secretAccessKey**: (*Obligatoire*) La clé d'accès utilisée pour accéder au fournisseur. Le **nom** doit correspondre à **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obligatoire*) Détermine ce qui fournit la sauvegarde ; par exemple, NetApp ONTAP S3, S3 générique, Google Cloud ou Microsoft Azure. Valeurs possibles :
  - aws
  - azuré
  - GCP
  - générique-s3
  - ontap-s3
  - storagegrid-s3

c. Après avoir rempli le `trident-protect-appvault-secondary-destination.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

8. Sur le cluster de destination, créez un fichier CR AppMirrorRelationship :

Créez une relation AppMirror à l'aide d'un CR

a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-relationship.yaml`).

b. Configurez les attributs suivants :

- **metadata.name:** (Obligatoire) Le nom de la ressource personnalisée AppMirrorRelationship.
- **spec.destinationAppVaultRef** : (*Obligatoire*) Cette valeur doit correspondre au nom de l'AppVault pour l'application de destination sur le cluster de destination.
- **spec.namespaceMapping**: (*Obligatoire*) Les espaces de noms de destination et source doivent correspondre à l'espace de noms de l'application défini dans la CR de l'application respective.
- **spec.sourceAppVaultRef** : (*Obligatoire*) Cette valeur doit correspondre au nom de l'AppVault pour l'application source.
- **spec.sourceApplicationName** : (*Obligatoire*) Cette valeur doit correspondre au nom de l'application source que vous avez définie dans la ressource personnalisée de l'application source.
- **spec.sourceApplicationUID** : (Obligatoire) Cette valeur doit correspondre à l'UID de l'application source que vous avez définie dans la CR de l'application source.
- **spec.storageClassName**: (*Optionnel*) Choisissez le nom d'une classe de stockage valide sur le cluster. La classe de stockage doit être liée à une machine virtuelle de stockage ONTAP qui est appariée avec l'environnement source. Si la classe de stockage n'est pas spécifiée, la classe de stockage par défaut du cluster sera utilisée.
- **spec.recurrenceRule** : Définissez une date de début en temps UTC et un intervalle de récurrence.

  Exemple de YAML :

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-2
```

c. Après avoir rempli le `trident-protect-relationship.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Créez une relation AppMirror à l'aide de l'interface de ligne de commande (CLI).

a. Créez et appliquez l'objet AppMirrorRelationship en remplaçant les valeurs entre crochets par les informations provenant de votre environnement :

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

Exemple:

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

9. (*Facultatif*) Sur le cluster de destination, vérifiez l'état et le statut de la relation de réplication :

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

Basculement vers le cluster de destination

Avec Trident Protect, vous pouvez basculer les applications répliquées vers un cluster de destination. Cette procédure interrompt la relation de réplication et met l'application en ligne sur le cluster de destination. Trident Protect n'arrête pas l'application sur le cluster source si elle était opérationnelle.

Étapes

1. Sur le cluster de destination, modifiez le fichier CR AppMirrorRelationship (par exemple, `trident-protect-relationship.yaml`) et modifiez la valeur de **spec.desiredState** en `Promoted` .
2. Enregistrez le fichier CR.
3. Appliquer le CR :

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

4. (*Facultatif*) Créez les plans de protection nécessaires sur l'application basculée.
5. (*Facultatif*) Vérifiez l'état et le statut de la relation de réplication :

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

Resynchroniser une relation de réplication ayant échoué

L'opération de resynchronisation rétablit la relation de réplication. Après une opération de resynchronisation, l'application source d'origine devient l'application en cours d'exécution, et toutes les modifications apportées à l'application en cours d'exécution sur le cluster de destination sont annulées.

Le processus interrompt l'application sur le cluster de destination avant de rétablir la réplication.

Toutes les données écrites dans l'application de destination pendant le basculement seront perdues.

Étapes

1. Facultatif : sur le cluster source, créez un instantané de l'application source. Cela permet de garantir que les dernières modifications provenant du cluster source sont prises en compte.
2. Sur le cluster de destination, modifiez le fichier CR AppMirrorRelationship (par exemple, `trident-protect-relationship.yaml`) et modifiez la valeur de spec.desiredState en `Established` .
3. Enregistrez le fichier CR.
4. Appliquer le CR :

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

5. Si vous avez créé des plans de protection sur le cluster de destination pour protéger l'application basculée, supprimez-les. Toute planification restante entraîne des échecs de snapshots de volume.

Resynchronisation inverse d'une relation de réplication ayant échoué

Lors d'une resynchronisation inverse d'une relation de réplication ayant basculé, l'application de destination devient l'application source et la source devient la destination. Les modifications apportées à l'application de destination pendant le basculement sont conservées.

Étapes

1. Sur le cluster de destination d'origine, supprimez la ressource personnalisée AppMirrorRelationship. Cela a pour conséquence que la destination devienne la source. S'il reste des plans de protection sur le nouveau cluster de destination, supprimez-les.
2. Établissez une relation de réplication en appliquant les fichiers CR que vous avez initialement utilisés pour établir la relation aux clusters opposés.
3. Assurez-vous que la nouvelle destination (cluster source d'origine) est configurée avec les deux CR AppVault.
4. Configurez une relation de réplication sur le cluster opposé, en configurant les valeurs pour la direction inverse.

sens de réplication de l'application inverse

Lorsque vous inversez le sens de la réplication, Trident Protect déplace l'application vers le système de stockage de destination tout en continuant à répliquer vers le système de stockage source d'origine. Trident Protect arrête l'application source et réplique les données vers la destination avant de basculer vers l'application de destination.

Dans ce cas, vous inversez la source et la destination.

Étapes

1. Sur le cluster source, créez un instantané d'arrêt :

Créez un instantané d'arrêt à l'aide d'une CR

a. Désactivez les calendriers de stratégie de protection pour l'application source.

b. Créer un fichier CR ShutdownSnapshot :

i. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-shutdownsnapshot.yaml`).

ii. Configurez les attributs suivants :

- **metadata.name**: (*Obligatoire*) Le nom de la ressource personnalisée.
- **spec.AppVaultRef** : (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name de l'AppVault pour l'application source.
- **spec.ApplicationRef**: (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name du fichier CR de l'application source.

Exemple de YAML :

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

c. Après avoir rempli le `trident-protect-shutdownsnapshot.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Créez un instantané d'arrêt à l'aide de l'interface de ligne de commande (CLI).

a. Créez un instantané d'arrêt en remplaçant les valeurs entre crochets par les informations de votre environnement. Par exemple:

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

2. Sur le cluster source, une fois la capture instantanée de l'arrêt terminée, obtenez l'état de cette capture :

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

3. Sur le cluster source, recherchez la valeur de **shutdownsnapshot.status.appArchivePath** à l'aide de la commande suivante et notez la dernière partie du chemin d'accès au fichier (également appelée nom de base ; il s'agit de tout ce qui suit la dernière barre oblique) :

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

4. Effectuez un basculement du nouveau cluster de destination vers le nouveau cluster source, avec la modification suivante :

> ⓘ À l'étape 2 de la procédure de basculement, incluez le `spec.promotedSnapshot` champ dans le fichier CR AppMirrorRelationship, et définissez sa valeur sur le nom de base que vous avez enregistré à l'étape 3 ci-dessus.

5. Effectuez les étapes de resynchronisation inverses dansResynchronisation inverse d'une relation de réplication ayant échoué .
6. Activez les plans de protection sur le nouveau cluster source.

Résultat

Les actions suivantes se produisent en raison de la réplication inverse :

- Une capture instantanée des ressources Kubernetes de l'application source d'origine est effectuée.
- Les pods de l'application source d'origine sont arrêtés en douceur en supprimant les ressources Kubernetes de l'application (en laissant les PVC et les PV en place).
- Une fois les pods arrêtés, des instantanés des volumes de l'application sont pris et répliqués.
- Les relations SnapMirror sont rompues, ce qui rend les volumes de destination prêts pour la lecture/écriture.
- Les ressources Kubernetes de l'application sont restaurées à partir de l'instantané antérieur à l'arrêt, en utilisant les données de volume répliquées après l'arrêt de l'application source d'origine.
- La réplication est rétablie dans le sens inverse.

Rétablir les applications sur le cluster source d'origine

Avec Trident Protect, vous pouvez effectuer un « retour en arrière » après une opération de basculement en utilisant la séquence d'opérations suivante. Dans ce flux de travail visant à rétablir le sens de réplication d'origine, Trident Protect réplique (resynchronise) toutes les modifications apportées à l'application vers l'application source d'origine avant d'inverser le sens de réplication.

Ce processus débute à partir d'une relation ayant effectué un basculement vers une destination et comprend les étapes suivantes :

- Commencez par un état de basculement.
- Inverser la resynchronisation de la relation de réplication.

> N'effectuez pas d'opération de resynchronisation normale, car cela supprimerait les données écrites sur le cluster de destination pendant la procédure de basculement.

- Inverser le sens de la réplication.

Étapes

1. Effectuez leResynchronisation inverse d'une relation de réplication ayant échoué mesures.
2. Effectuez lesens de réplication de l'application inverse mesures.

Supprimer une relation de réplication

Vous pouvez supprimer une relation de réplication à tout moment. Lorsque vous supprimez la relation de réplication d'applications, cela crée deux applications distinctes sans aucune relation entre elles.

Étapes

1. Sur le cluster de destination actuel, supprimez la ressource personnalisée AppMirrorRelationship :

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

Migrer les applications à l'aide de Trident Protect

Vous pouvez migrer vos applications entre clusters ou vers différentes classes de stockage en restaurant les données de sauvegarde.

> Lors de la migration d'une application, tous les points d'exécution configurés pour celle-ci sont migrés avec elle. Si un point d'entrée d'exécution post-restauration est présent, il s'exécute automatiquement dans le cadre de l'opération de restauration.

opérations de sauvegarde et de restauration

Pour effectuer des opérations de sauvegarde et de restauration dans les scénarios suivants, vous pouvez automatiser des tâches spécifiques de sauvegarde et de restauration.

Cloner sur le même cluster

Pour cloner une application sur le même cluster, créez un instantané ou une sauvegarde et restaurez les données sur le même cluster.

Étapes

1. Effectuez l'une des opérations suivantes :
   a. "Créer un instantané".
   b. "Créer une sauvegarde".
2. Sur le même cluster, effectuez l'une des opérations suivantes, selon que vous ayez créé un instantané ou une sauvegarde :

a. "Restaurez vos données à partir de l'instantané.".

b. "Restaurez vos données à partir de la sauvegarde".

Cloner sur un cluster différent

Pour cloner une application sur un cluster différent (effectuer un clonage inter-clusters), créez une sauvegarde sur le cluster source, puis restaurez la sauvegarde sur un cluster différent. Assurez-vous que Trident Protect est installé sur le cluster de destination.

> Vous pouvez répliquer une application entre différents clusters en utilisant"Réplication SnapMirror" .

Étapes

1. "Créer une sauvegarde".
2. Assurez-vous que la ressource personnalisée AppVault pour le compartiment de stockage d'objets contenant la sauvegarde a été configurée sur le cluster de destination.
3. Sur le cluster de destination,"restaurez vos données à partir de la sauvegarde" .

Migrer des applications d'une classe de stockage à une autre classe de stockage

Vous pouvez migrer des applications d'une classe de stockage vers une autre classe de stockage en restaurant une sauvegarde vers la classe de stockage de destination.

Par exemple (en excluant les secrets de la restauration CR) :

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

Restaurez l'instantané à l'aide d'une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-snapshot-restore-cr.yaml` .
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appArchivePath** : Le chemin d'accès dans AppVault où le contenu des instantanés est stocké. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

   - **spec.appVaultRef**: (*Obligatoire*) Le nom de l'AppVault où le contenu de l'instantané est stocké.
   - **spec.namespaceMapping** : Le mappage de l'espace de noms source de l'opération de restauration vers l'espace de noms de destination. Remplacer `my-source-namespace` et `my-destination-namespace` avec des informations provenant de votre environnement.

     ```
     ---
     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. Si vous souhaitez sélectionner uniquement certaines ressources de l'application à restaurer, vous pouvez ajouter un filtrage qui inclut ou exclut les ressources marquées d'étiquettes particulières :
   - **resourceFilter.resourceSelectionCriteria** : (Requis pour le filtrage) Utiliser `include or exclude` inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
     - **resourceFilter.resourceMatchers** : Un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent selon une opération OU, et les champs à l'intérieur de chaque élément (groupe, type, version) correspondent selon une opération ET.
       - **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
       - **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.
       - **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.
       - **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ metadata.name de

Kubernetes de la ressource à filtrer.

- **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes de la ressource à filtrer.
- **resourceMatchers[].labelSelectors** : (*Facultatif*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes, telle que définie dans le "Documentation Kubernetes" . Par exemple: `"trident.netapp.io/os=linux"` .

  Par exemple:

```
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. Après avoir rempli le `trident-protect-snapshot-restore-cr.yaml` fichier contenant les valeurs correctes, appliquer le CR :

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

Restaurez l'instantané à l'aide de l'interface de ligne de commande (CLI).

Étapes

1. Restaurez l'instantané dans un espace de noms différent, en remplaçant les valeurs entre crochets par les informations de votre environnement.

   - Le `snapshot` L'argument utilise un espace de noms et un nom d'instantané au format `<namespace>/<name>` .
   - Le `namespace-mapping` L'argument utilise des espaces de noms séparés par deux-points pour associer les espaces de noms source aux espaces de noms de destination corrects au format `source1:dest1,source2:dest2` .

     Par exemple:

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

Gérer les hooks d'exécution de Trident Protect

Un hook d'exécution est une action personnalisée que vous pouvez configurer pour qu'elle s'exécute conjointement avec une opération de protection des données d'une application gérée. Par exemple, si vous disposez d'une application de base de données, vous pouvez utiliser un hook d'exécution pour suspendre toutes les transactions de base de données avant un instantané et reprendre les transactions une fois l'instantané terminé. Cela garantit des instantanés cohérents avec les applications.

Types de hooks d'exécution

Trident Protect prend en charge les types de hooks d'exécution suivants, en fonction du moment où ils peuvent être exécutés :

- Pré-instantané
- Post-instantané
- Pré-sauvegarde
- Post-sauvegarde
- Post-restauration
- Après le basculement

Ordre d'exécution

Lorsqu'une opération de protection des données est exécutée, les événements de hook d'exécution se produisent dans l'ordre suivant :

1. Tous les hooks d'exécution de pré-opération personnalisés applicables sont exécutés sur les conteneurs appropriés. Vous pouvez créer et exécuter autant de hooks de pré-opération personnalisés que vous le souhaitez, mais l'ordre d'exécution de ces hooks avant l'opération n'est ni garanti ni configurable.
2. Des blocages du système de fichiers se produisent, le cas échéant. "Apprenez-en davantage sur la configuration du gel du système de fichiers avec Trident Protect.".
3. L'opération de protection des données est effectuée.
4. Les systèmes de fichiers gelés sont dégelés, le cas échéant.
5. Tous les hooks d'exécution post-opération personnalisés applicables sont exécutés sur les conteneurs appropriés. Vous pouvez créer et exécuter autant de hooks post-opération personnalisés que vous le souhaitez, mais l'ordre d'exécution de ces hooks après l'opération n'est ni garanti ni configurable.

Si vous créez plusieurs hooks d'exécution du même type (par exemple, pré-snapshot), l'ordre d'exécution de ces hooks n'est pas garanti. Cependant, l'ordre d'exécution des hooks de différents types est garanti. Par exemple, voici l'ordre d'exécution d'une configuration qui possède tous les différents types de hooks :

1. Hooks pré-instantanés exécutés
2. Hooks post-instantanés exécutés
3. Hooks de pré-sauvegarde exécutés
4. Hooks post-sauvegarde exécutés

L'exemple de commande précédent ne s'applique que lorsque vous exécutez une sauvegarde qui n'utilise pas d'instantané existant.

Vous devez toujours tester vos scripts d'exécution avant de les activer dans un environnement de production. Vous pouvez utiliser la commande « kubectl exec » pour tester facilement les scripts. Après avoir activé les hooks d'exécution dans un environnement de production, testez les snapshots et les sauvegardes résultants pour vous assurer qu'ils sont cohérents. Vous pouvez le faire en clonant l'application dans un espace de noms temporaire, en restaurant l'instantané ou la sauvegarde, puis en testant l'application.

Si un hook d'exécution pré-snapshot ajoute, modifie ou supprime des ressources Kubernetes, ces modifications sont incluses dans le snapshot ou la sauvegarde et dans toute opération de restauration ultérieure.

Remarques importantes sur les hooks d'exécution personnalisés

Tenez compte des éléments suivants lors de la planification des hooks d'exécution pour vos applications.

- Un hook d'exécution doit utiliser un script pour effectuer des actions. De nombreux hooks d'exécution peuvent référencer le même script.
- Trident Protect exige que les scripts utilisés par les hooks d'exécution soient écrits au format de scripts shell exécutables.
- La taille du script est limitée à 96 Ko.
- Trident Protect utilise les paramètres d'exécution et tous les critères correspondants pour déterminer quels hooks sont applicables à une opération de snapshot, de sauvegarde ou de restauration.

Étant donné que les hooks d'exécution réduisent ou désactivent souvent complètement les fonctionnalités de l'application sur laquelle ils s'exécutent, vous devez toujours essayer de minimiser le temps d'exécution de vos hooks d'exécution personnalisés. Si vous démarrez une opération de sauvegarde ou de snapshot avec des hooks d'exécution associés, mais que vous l'annulez ensuite, les hooks sont toujours autorisés à s'exécuter si l'opération de sauvegarde ou de snapshot a déjà commencé. Cela signifie que la logique utilisée dans un hook d'exécution post-sauvegarde ne peut pas supposer que la sauvegarde a été terminée.

Filtres de crochet d'exécution

Lorsque vous ajoutez ou modifiez un hook d'exécution pour une application, vous pouvez ajouter des filtres au hook d'exécution pour gérer les conteneurs auxquels le hook correspondra. Les filtres sont utiles pour les applications qui utilisent la même image de conteneur sur tous les conteneurs, mais peuvent utiliser chaque image à des fins différentes (comme Elasticsearch). Les filtres vous permettent de créer des scénarios dans lesquels les hooks d'exécution s'exécutent sur certains conteneurs identiques, mais pas nécessairement sur tous. Si vous créez plusieurs filtres pour un seul hook d'exécution, ils sont combinés avec un opérateur AND logique. Vous pouvez avoir jusqu'à 10 filtres actifs par hook d'exécution.

Chaque filtre que vous ajoutez à un hook d'exécution utilise une expression régulière pour faire correspondre les conteneurs de votre cluster. Lorsqu'un hook correspond à un conteneur, le hook exécutera son script associé sur ce conteneur. Les expressions régulières pour les filtres utilisent la syntaxe d'expression régulière 2 (RE2), qui ne prend pas en charge la création d'un filtre excluant les conteneurs de la liste des correspondances. Pour plus d'informations sur la syntaxe prise en charge par Trident Protect pour les expressions régulières dans les filtres de hook d'exécution, consultez "Prise en charge de la syntaxe des expressions régulières 2 (RE2)" .

Si vous ajoutez un filtre d'espace de noms à un hook d'exécution qui s'exécute après une opération de restauration ou de clonage et que la source et la destination de restauration ou de clonage se trouvent dans des espaces de noms différents, le filtre d'espace de noms est appliqué uniquement à l'espace de noms de destination.

Exemples de crochets d'exécution

Visitez le "Projet GitHub NetApp Verda" pour télécharger de véritables hooks d'exécution pour des applications populaires telles qu'Apache Cassandra et Elasticsearch. Vous pouvez également voir des exemples et obtenir des idées pour structurer vos propres hooks d'exécution personnalisés.

Créer un point d'accroche d'exécution

Vous pouvez créer un hook d'exécution personnalisé pour une application en utilisant . Vous devez disposer des autorisations de propriétaire, d'administrateur ou de membre pour créer des points d'exécution.

Utilisez un CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-hook.yaml` .
2. Configurez les attributs suivants pour qu'ils correspondent à votre environnement Trident Protect et à la configuration de votre cluster :

   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.applicationRef**: (*Obligatoire*) Le nom Kubernetes de l'application pour laquelle exécuter le hook d'exécution.
   - **spec.stage**: (*Obligatoire*) Une chaîne de caractères indiquant à quelle étape de l'action le hook d'exécution doit s'exécuter. Valeurs possibles :
     - Pré
     - Poste
   - **spec.action**: (*Obligatoire*) Une chaîne indiquant l'action que le hook d'exécution entreprendra, en supposant que tous les filtres de hook d'exécution spécifiés correspondent. Valeurs possibles :
     - Instantané
     - Sauvegarde
     - Restaurer
     - Basculement
   - **spec.enabled**: (*Optionnel*) Indique si ce point d'accroche d'exécution est activé ou désactivé. Si aucune valeur n'est spécifiée, la valeur par défaut est « vrai ».
   - **spec.hookSource**: (*Obligatoire*) Une chaîne contenant le script de hook encodé en base64.
   - **spec.timeout**: (*Optionnel*) Un nombre définissant la durée en minutes pendant laquelle le hook d'exécution est autorisé à s'exécuter. La valeur minimale est de 1 minute, et la valeur par défaut est de 25 minutes si elle n'est pas spécifiée.
   - **spec.arguments**: (*Optionnel*) Une liste YAML d'arguments que vous pouvez spécifier pour le hook d'exécution.
   - **spec.matchingCriteria**: (*Optionnel*) Une liste facultative de paires clé-valeur de critères, chaque paire constituant un filtre de crochet d'exécution. Vous pouvez ajouter jusqu'à 10 filtres par point d'exécution.
   - **spec.matchingCriteria.type**: (*Optionnel*) Une chaîne identifiant le type de filtre de crochet d'exécution. Valeurs possibles :
     - Image conteneur
     - Nom du conteneur
     - Nom du pod
     - Étiquette de podcast
     - Nom de l'espace de noms
   - **spec.matchingCriteria.value**: (*Optionnel*) Une chaîne de caractères ou une expression régulière identifiant la valeur du filtre du point d'exécution.

     Exemple de YAML :

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

3. Après avoir rempli le fichier CR avec les valeurs correctes, appliquez le CR :

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

Utiliser la CLI

Étapes

1. Créez le point d'exécution en remplaçant les valeurs entre crochets par les informations provenant de votre environnement. Par exemple:

```
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>
```

Exécuter manuellement un hook d'exécution

Vous pouvez exécuter manuellement un hook d'exécution à des fins de test ou si vous devez le réexécuter manuellement après un échec. Vous devez disposer des autorisations de propriétaire, d'administrateur ou de membre pour exécuter manuellement les hooks d'exécution.

L'exécution manuelle d'un hook se compose de deux étapes de base :

1. Créez une sauvegarde des ressources, qui collecte les ressources et en crée une copie, déterminant ainsi où le hook s'exécutera.
2. Exécutez le script d'exécution sur la sauvegarde

Étape 1 : Créer une sauvegarde des ressources

Utilisez un CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-resource-backup.yaml` .
2. Configurez les attributs suivants pour qu'ils correspondent à votre environnement Trident Protect et à la configuration de votre cluster :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.applicationRef**: (*Obligatoire*) Le nom Kubernetes de l'application pour laquelle créer la sauvegarde de ressource.
   - **spec.appVaultRef**: (*Obligatoire*) Le nom de l'AppVault où sont stockés les contenus de sauvegarde.
   - **spec.appArchivePath** : Chemin d'accès dans AppVault où sont stockés les contenus de sauvegarde. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

     Exemple de YAML :

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

3. Après avoir rempli le fichier CR avec les valeurs correctes, appliquez le CR :

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

Utiliser la CLI

Étapes

1. Créez la sauvegarde en remplaçant les valeurs entre crochets par les informations provenant de votre environnement. Par exemple:

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

2. Consultez l'état de la sauvegarde. Vous pouvez utiliser cette commande d'exemple à plusieurs reprises jusqu'à ce que l'opération soit terminée :

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

3. Vérifiez que la sauvegarde a réussi :

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

Étape 2 : Exécuter le hook d’exécution

Utilisez un CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le. `trident-protect-hook-run.yaml` .
2. Configurez les attributs suivants pour qu'ils correspondent à votre environnement Trident Protect et à la configuration de votre cluster :
   - **metadata.name** : (*Obligatoire*) Le nom de cette ressource personnalisée ; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.applicationRef** : (*Obligatoire*) Assurez-vous que cette valeur corresponde au nom de l'application de la ressource de sauvegarde que vous avez créée à l'étape 1.
   - **spec.appVaultRef** : (*Obligatoire*) Assurez-vous que cette valeur correspond à l'appVaultRef de la ressource CR ResourceBackup que vous avez créée à l'étape 1.
   - **spec.appArchivePath** : Assurez-vous que cette valeur correspond à appArchivePath de la ressource personnalisée ResourceBackup que vous avez créée à l'étape 1.

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

   - **spec.action**: (*Obligatoire*) Une chaîne indiquant l'action que le hook d'exécution entreprendra, en supposant que tous les filtres de hook d'exécution spécifiés correspondent. Valeurs possibles :
     - Instantané
     - Sauvegarde
     - Restaurer
     - Basculement
   - **spec.stage**: (*Obligatoire*) Une chaîne de caractères indiquant à quelle étape de l'action le hook d'exécution doit s'exécuter. Cette opération d'accrochage ne permettra pas d'accrocher les hameçons à aucune autre étape. Valeurs possibles :
     - Pré
     - Poste

     Exemple de YAML :

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

3. Après avoir rempli le fichier CR avec les valeurs correctes, appliquez le CR :

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

Utiliser la CLI

Étapes

1. Créer la requête d'exécution manuelle du hook :

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

2. Vérifiez l'état d'exécution du hook. Vous pouvez exécuter cette commande à plusieurs reprises jusqu'à ce que l'opération soit terminée :

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

3. Décrivez l'objet exechooksrun pour voir les détails et l'état finaux :

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