Gérer et protéger les applications

Trident

NetApp
July 01, 2026

Sommaire

Gérer et protéger les applications

Utilisez les objets Trident Protect AppVault pour gérer les compartiments

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

Vous devez créer une AppVault CR manuellement ou à partir de la ligne de commandes lorsque vous effectuez des opérations de protection des données sur une application. Le AppVault CR est spécifique à votre environnement, et vous pouvez utiliser les exemples de cette page comme guide lors de la création de AppVault CR.

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

Configurer l'authentification et les mots de passe AppVault

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

Mots de passe du référentiel de data mover

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

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

Créer un secret de mot de passe 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 demander à Trident Protect d'utiliser ce secret pour s'authentifier auprès du référentiel du data mover.

- Selon le data mover que vous utilisez, il vous suffit d'inclure le mot de passe correspondant pour ce data mover. 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 un autre. Si le cluster ou le `trident-protect` namespace est supprimé, vous ne pourrez pas restaurer vos sauvegardes ou 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
```

Utilisez la ligne de commandes (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, S3 générique, "StorageGrid S3", ou "ONTAP S3" en utilisant Trident Protect, vous devez vous assurer que les informations d'identification de l'utilisateur disposent des autorisations nécessaires pour accéder au compartiment. Voici un exemple de stratégie accordant les autorisations minimales requises pour l'accès avec Trident Protect. Vous pouvez appliquer cette stratégie à l'utilisateur qui gère les stratégies des compartiments 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 le "Documentation Amazon S3".

EKS Pod Identity 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 compartiments S3 sans stocker les informations d'identification AWS dans les secrets Kubernetes.

Exigences pour EKS Pod Identity 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 disposant des autorisations nécessaires pour le compartiment S3. 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, reportez-vous à la "Documentation AWS EKS Pod Identity".

AppVault Configuration Lors de l'utilisation d'EKS Pod Identity, configurez votre AppVault CR avec le `useIAM: true` indicateur au lieu d'informations d'identification 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
```

AppVault exemples de génération de clés pour les fournisseurs de cloud

Lors de la définition d'une AppVault CR, vous devez inclure les informations d'identification permettant d'accéder aux ressources hébergées par le fournisseur, sauf si vous utilisez l'authentification IAM. La façon dont vous générez les clés pour les informations d'identification varie selon le fournisseur. Voici des exemples de génération de clés en ligne de commandes pour plusieurs fournisseurs. Vous pouvez utiliser les exemples suivants pour créer les 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
```

AppVault creation exemples

Les exemples suivants sont des définitions AppVault pour chaque fournisseur.

AppVault CR exemples

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

- Vous pouvez spécifier, si vous le souhaitez, un secret Kubernetes contenant des mots de passe personnalisés pour le chiffrement du référentiel Restic et Kopia. Consultez Mots de passe du référentiel de data mover pour plus d'informations.
- Pour les objets AppVault Amazon S3 (AWS), vous pouvez éventuellement spécifier un sessionToken, ce qui est utile si vous utilisez l'authentification par connexion unique (SSO). Ce jeton est créé lorsque vous générez des clés pour le fournisseur dans AppVault exemples de génération de clés 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 la clé `spec.providerConfig.S3.proxyURL`.

Google Cloud

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

Amazon S3 (AWS)

```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 la `providerCredentials` section et ajouter `useIAM: true` sous la configuration `s3` à la place.

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
```

AppVault Exemples de création à l'aide de la ligne de commandes Trident Protect

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

- Vous pouvez spécifier, si vous le souhaitez, un secret Kubernetes contenant des mots de passe personnalisés pour le chiffrement du référentiel Restic et Kopia. Consultez Mots de passe du référentiel de data mover 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 l'argument `--proxy-url <ip_address:port>`.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

S3 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
```

Options de configuration `providerConfig.s3` prises en charge

Consultez le tableau suivant pour connaître les options de configuration du fournisseur S3 :

Paramètre	Description	Défaut	Exemple
`providerConfig.s3.skipCertValidation`	Désactivez la vérification du certificat SSL/TLS.	false	"true", "false"
`providerConfig.s3.secure`	Activez la communication HTTPS sécurisée avec le point de terminaison S3.	true	"true", "false"
`providerConfig.s3.proxyURL`	Spécifiez l'URL du serveur proxy utilisé pour se connecter à S3.	Aucune	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Fournissez un certificat d'autorité de certification racine personnalisé pour la vérification SSL/TLS.	Aucune	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	Activez l'authentification IAM pour accéder aux compartiments S3. Applicable à EKS Pod Identity.	false	vrai, faux

Afficher les informations AppVault

Vous pouvez utiliser le plugin CLI Trident Protect pour consulter les informations sur les 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 éventuellement le AppVaultPath pour chaque ressource, utilisez le drapeau `--show-paths`.

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

Retirer un AppVault

Vous pouvez supprimer un AppVault objet à tout moment.

Ne supprimez pas la `finalizers` clé dans le AppVault CR avant de supprimer l'objet AppVault. Si vous le faites, cela peut entraîner des données résiduelles dans le compartiment AppVault et des ressources orphelines dans le cluster.

Avant de commencer

Assurez-vous d'avoir supprimé toutes les CR de snapshot et de sauvegarde utilisées par le AppVault que vous souhaitez supprimer.

Supprimez un AppVault à l'aide de la CLI Kubernetes

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

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

Supprimez un AppVault à l'aide de la ligne de commandes Trident Protect

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

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

Définir une application pour la gestion avec Trident Protect

Vous pouvez définir une application que vous souhaitez gérer avec Trident Protect en créant un CR d'application et un CR AppVault associé.

Créer un AppVault CR

Vous devez créer une AppVault CR qui sera utilisée lors des opérations de protection des données sur l'application, et la AppVault CR doit résider sur le cluster où Trident Protect est installé. La AppVault CR est spécifique à votre environnement ; pour des exemples de AppVault CR, consultez "AppVault ressources personnalisées."

Définir une application

Vous devez définir chaque application que vous souhaitez gérer avec Trident Protect. Vous pouvez définir une application à gérer soit en créant manuellement un CR d'application, soit en utilisant le CLI de Trident Protect.

Ajouter une application à l'aide d'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 un espace de noms et un sélecteur 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'étiquette est facultatif et peut être utilisé pour filtrer les ressources au sein de chaque espace de noms spécifié.
      - **spec.includedClusterScopedResources**: (*Facultatif*) Utilisez cet attribut pour spécifier les ressources de portée cluster à inclure dans la définition de l'application. Cet attribut vous permet de sélectionner ces ressources en fonction de leur groupe, version, type et 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 du cluster en fonction de leurs étiquettes.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze** : (*Optionnel*) Cette annotation s'applique uniquement aux applications définies à partir de machines virtuelles, telles que dans les environnements KubeVirt, où le gel du système de fichiers intervient avant la création d'instantanés. Indiquez si cette application peut écrire sur le système de fichiers pendant la création d'un instantané. Si la valeur est true, l'application ignore le paramètre global et peut écrire sur le système de fichiers pendant la création d'un instantané. Si la valeur est false, l'application ignore le paramètre global et le système de fichiers est gelé pendant 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. Si elle n'est pas spécifiée, l'application suit le "paramètre de gel global Trident Protect".

Si vous devez appliquer cette annotation après qu'une application a déjà été créée, 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 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*) Si nécessaire, vous pouvez ajouter un filtrage des ressources à la même CR pour inclure ou exclure des ressources spécifiques :

   - **Exemple de filtre générique :**

     - **resourceFilter.resourceSelectionCriteria** : (Obligatoire pour le filtrage) Utilisez `Include` ou `Exclude` pour 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 (group, kind, 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 Kubernetes metadata.name 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** : (*Optionnel*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes tel que défini dans le "Documentation Kubernetes". Par exemple : `"trident.netapp.io/os=linux"`.

> ⓘ Lorsque `resourceFilter` et `labelSelector` sont utilisés, `resourceFilter` s'exécute en premier, puis `labelSelector` est appliqué 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"]
```

◦ **Exemple de filtre uniquement PVC :**

Pour définir une application utilisant uniquement des PVC, vous devez également inclure `PersistentVolume` et `VolumeSnapshotClass` dans le filtre de ressources. Les opérations de snapshot et de sauvegarde dépendent de `PersistentVolume` (le volume à portée du cluster lié à chaque PVC) et `VolumeSnapshotClass` (le pilote de snapshot), et échoueront sans eux. Par exemple :

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

2. Après avoir créé la CR d'application adaptée à votre environnement, appliquez la CR. Par exemple :

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

Étapes

1. Créez et appliquez la définition d'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 d'application en utilisant des listes séparées par des virgules avec les arguments indiqués dans les exemples.

   Vous pouvez éventuellement utiliser une annotation lors de la création d'une application pour spécifier si l'application peut écrire sur le système de fichiers pendant un instantané. Ceci s'applique uniquement aux applications définies à partir de machines virtuelles, comme dans les environnements KubeVirt, où le système de fichiers est gelé avant les instantanés. Si vous définissez l'annotation sur `true`, l'application ignore le paramètre global et peut écrire sur le système de fichiers pendant un instantané. Si vous la définissez sur `false`, l'application ignore le paramètre global et le système de fichiers est gelé pendant un instantané. Si vous utilisez l'annotation mais que l'application ne comporte aucune machine virtuelle dans la définition de l'application, l'annotation est ignorée. Si vous n'utilisez pas l'annotation, l'application suit le "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 le `--annotation` indicateur.

   - 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 le paramètre local 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` pour inclure ou 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"]}]'
```

- Pour définir une application utilisant uniquement des PVC, vous devez également inclure `PersistentVolume` et `VolumeSnapshotClass` dans le filtre de ressources. Les opérations de snapshot et de sauvegarde dépendent de `PersistentVolume` (le volume à portée du cluster lié à chaque PVC) et `VolumeSnapshotClass` (le pilote de snapshot), et échoueront sans eux. Par exemple :

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

Protégez les applications à l'aide de Trident Protect

Vous pouvez protéger toutes les applications gérées par Trident Protect en prenant des

snapshots 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. "En savoir plus 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 ont des références 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 défini sur `Retain`, l'instantané 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 fichier `trident-protect-snapshot-cr.yaml` avec les valeurs correctes, appliquez 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 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 app 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 ont des références 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.

- Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Consultez la "Documentation AWS IAM" pour plus d'informations sur les identifiants relatifs aux 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 du AppVault où le contenu de la sauvegarde doit être stocké.
   - **spec.dataMover** : (*Facultatif*) Chaîne de caractères indiquant l'outil de sauvegarde à utiliser pour l'opération de sauvegarde. Valeurs possibles (sensibles à la casse) :
     - `Restic`
     - `Kopia` (défaut)
   - **spec.reclaimPolicy** : (*Optionnel*) Définit ce qui se passe pour une sauvegarde lorsqu'elle est libérée de sa revendication. Valeurs possibles :
     - `Delete`
     - `Retain` (défaut)
   - **spec.snapshotRef** : (*Facultatif*) : Nom de l'instantané à utiliser comme source de la sauvegarde. Si cette information n'est pas fournie, un instantané temporaire sera créé et sauvegardé.

     Exemple YAML :

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

3. Après avoir rempli le fichier `trident-protect-backup-cr.yaml` avec les valeurs correctes, appliquez 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 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 éventuellement utiliser le `--full-backup` flag pour spécifier si une sauvegarde doit être non incrémentielle. Par défaut, toutes les sauvegardes sont incrémentielles. Lorsque ce flag est utilisé, la sauvegarde devient non incrémentielle. Il est bonne pratique d'effectuer une sauvegarde complète périodiquement, puis d'effectuer des sauvegardes incrémentielles entre les sauvegardes complètes afin de minimiser le risque associé aux restaurations.

Annotations de sauvegarde prises en charge

Le tableau suivant décrit les annotations que vous pouvez utiliser lors de la création d'un CR de sauvegarde :

Annotation	Type	Description	valeur par défaut
protect.trident.netapp.io/full-backup	chaîne	Indique si une sauvegarde doit être non incrémentielle. Définissez sur `true` pour créer une sauvegarde non incrémentielle. Il est bonne pratique d'effectuer une sauvegarde complète périodiquement, puis d'effectuer des sauvegardes incrémentielles entre les sauvegardes complètes afin de minimiser le risque associé aux restaurations.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	chaîne	Le temps maximal autorisé pour que l'opération globale de capture d'instantané soit terminée.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	chaîne	Le temps maximal autorisé pour que les instantanés de volume atteignent l'état prêt à l'emploi.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	chaîne	Durée maximale autorisée pour la création d'instantanés de volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	chaîne	Temps maximal (en secondes) d'attente pour que tout nouveau PersistentVolumeClaims (PVC) atteigne la phase `Bound` avant que l'opération n'échoue.	"1200" (20 minutes)

Créer un calendrier de protection des données

Une politique de protection protège une app en créant des instantanés, des sauvegardes, ou les deux selon une planification définie. 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 à l'aide de l'annotation full-backup-rule. Par défaut, toutes les sauvegardes sont incrémentielles. Effectuer une sauvegarde complète périodiquement,

ainsi que des sauvegardes incrémentielles entre les deux, aide à réduire le risque associé aux restaurations.

- Vous pouvez créer des planifications pour les instantanés uniquement en définissant `backupRetention` à zéro et `snapshotRetention` à une valeur supérieure à zéro. Définir `snapshotRetention` à zéro signifie que toute sauvegarde planifiée créera toujours des instantanés, mais ceux-ci seront temporaires et supprimés immédiatement après la fin de la sauvegarde.
- 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 ont des références 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** : (*Facultatif*) Chaîne de caractères indiquant l'outil de sauvegarde à utiliser pour l'opération de sauvegarde. Valeurs possibles (sensibles à la casse) :
     - `Restic`
     - `Kopia` (défaut)
   - **spec.applicationRef** : Le nom Kubernetes de l'application à sauvegarder.
   - **spec.appVaultRef** : (*Obligatoire*) Le nom du AppVault où le contenu de la sauvegarde doit être stocké.
   - **spec.backupRetention**: (*Obligatoire*) Le nombre de sauvegardes à conserver. Zéro indique qu'aucune sauvegarde ne doit être créée (instantanés uniquement).
   - **spec.backupReclaimPolicy** : (*Optionnel*) Détermine ce qui se passe pour une sauvegarde si le CR de sauvegarde est supprimé pendant sa période de rétention. Après la période de rétention, les sauvegardes sont toujours supprimées. Valeurs possibles (sensibles à la casse) :
     - `Retain` (défaut)
     - `Delete`
   - **spec.snapshotRetention** : (*Obligatoire*) Le nombre d'instantanés à conserver. Zéro indique qu'aucun instantané ne doit être créé.
   - **spec.snapshotReclaimPolicy** : (*Optionnel*) Détermine ce qui se passe pour un instantané si la ressource personnalisée (CR) de l'instantané est supprimée pendant sa période de rétention. Après la période de rétention, les instantanés sont toujours supprimés. Valeurs possibles (sensibles à la casse) :
     - `Retain`
     - `Delete` (défaut)
   - **spec.granularity** : La fréquence à laquelle la planification doit s'exécuter. Valeurs possibles, ainsi que les champs associés obligatoires :
     - `Hourly` (nécessite que vous spécifiiez `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 requis 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) auquel la planification doit s'exécuter. Les valeurs 0 ou 7 indiquent le dimanche. Ce champ est requis 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 la planification 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.
- **spec.runImmediately** : (*Optionnel*) Définissez sur `true` pour déclencher une exécution de base unique et immédiate (sauvegarde et/ou snapshot selon les paramètres de rétention) lors de la création de la planification. La valeur par défaut est `false`. Cela ne modifie pas la récurrence ultérieure.

  Exemple YAML pour la planification des sauvegardes et des instantanés :

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

  Exemple YAML pour une planification uniquement par instantané :

```
---
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"
```

Exemple YAML pour une planification avec exécution immédiate :

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

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

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

Créez une planification à l'aide de la CLI

Étapes

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

Vous pouvez utiliser `tridentctl-protect create schedule --help` pour consulter des informations d'aide détaillées pour 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> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_to_run_schedule> \
  --granularity <frequency_to_run> \
  --hour <hour_of_day_to_run> \
  --minute <minute_of_hour_to_run> \
  --recurrence-rule <recurrence> \
  --snapshot-retention <how_many_snapshots_to_retain> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

Les options suivantes offrent un contrôle supplémentaire sur votre emploi du temps :

- **Planification de sauvegarde complète** : Utilisez le `--full-backup-rule` indicateur pour planifier des sauvegardes complètes non incrémentielles. Cet indicateur fonctionne uniquement avec `--granularity Daily`. Valeurs possibles :
  - `Always`: Créez une sauvegarde complète chaque jour.
  - Jours de la semaine spécifiques : Indiquez un ou plusieurs jours séparés par des virgules (par exemple, `"Monday,Thursday"`). Valeurs valides : Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

    Le `--full-backup-rule`paramètre ne fonctionne pas avec la granularité Horaire, Hebdomadaire ou Mensuelle.

- **Protection immédiate de la base de référence** : Utilisez `--run-immediately true` pour créer une sauvegarde ou un instantané initial immédiatement lors de la création de la planification, plutôt que d'attendre la première exécution planifiée. La valeur par défaut est `false`.
- **Planifications d'instantané uniquement** : Définissez `--backup-retention 0` et spécifiez une valeur supérieure à zéro pour `--snapshot-retention`.

Annotations de planification prises en charge

Le tableau suivant décrit les annotations que vous pouvez utiliser lors de la création d'un schedule CR :

Annotation	Type	Description	valeur par défaut
protect.trident.netapp.io/full-backup-rule	chaîne	Spécifie la règle de planification des sauvegardes complètes. Vous pouvez la configurer sur `Always` pour une sauvegarde complète constante ou la personnaliser selon 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, `"Monday,Thursday"`). Les valeurs valides pour les jours de la semaine sont : Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. Notez que cette annotation ne peut être utilisée qu'avec les planifications qui ont `granularity` défini sur `Daily`.	Non défini (toutes les sauvegardes sont incrémentales)
protect.trident.netapp.io/snapshot-completion-timeout	chaîne	Le temps maximal autorisé pour que l'opération globale de capture d'instantané soit terminée.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	chaîne	Le temps maximal autorisé pour que les instantanés de volume atteignent l'état prêt à l'emploi.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	chaîne	Durée maximale autorisée pour la création d'instantanés de volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	chaîne	Temps maximal (en secondes) d'attente pour que tout nouveau PersistentVolumeClaims (PVC) atteigne la phase `Bound` avant que l'opération n'échoue.	"1200" (20 minutes)

Supprimer un instantané

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

Étapes

1. Supprimez le snapshot CR associé 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 stratégie de récupération est configurée sur `Delete` pour supprimer toutes les données de sauvegarde du stockage objet. Le paramètre par défaut de la stratégie est `Retain` pour éviter toute perte de données. Si la stratégie n'est pas modifiée sur `Delete`, les données de sauvegarde resteront dans le stockage objet et devront être supprimées manuellement.

Étapes

1. Supprimez le CR de sauvegarde associé à la sauvegarde :

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

Vérifiez l'état d'une opération de sauvegarde

Vous pouvez utiliser la ligne de commandes pour vérifier l'état d'une opération de sauvegarde qui est en cours, terminée ou a échoué.

Étapes

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

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

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

Si vous avez installé Trident Protect, vous pouvez activer la fonctionnalité de sauvegarde et de restauration efficace en termes d'espace pour les systèmes de stockage utilisant la classe de stockage azure-netapp-files et créés avant Trident 24.06. Cette fonctionnalité fonctionne 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 disposera d'une fonctionnalité de protection limitée jusqu'à ce que vous ayez terminé cette procédure.
- Vous avez `azure-netapp-files` sélectionné comme classe de stockage par défaut pour votre storage backend.

Développez pour afficher les étapes de configuration

1. Effectuez les opérations suivantes 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 qui est 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, Trident Protect choisit la fonctionnalité de sauvegarde régulière, 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 à sauvegarder.

Résultat

L'application est prête pour la sauvegarde et la restauration à l'aide de Trident Protect. Chaque PVC est également disponible pour 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.

- Lors de la restauration d'une application, tous les points d'exécution configurés pour l'application sont restaurés avec l'application. Si un point 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 les 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 BackupRestore CR, Trident Protect restaure l'application dans un nouvel espace de noms et crée un CR d'application pour l'application restaurée. Pour protéger l'application restaurée, créez des sauvegardes ou des instantanés à la demande, ou définissez une planification de protection.

- La restauration d'une sauvegarde dans un espace de noms différent contenant des ressources existantes ne modifiera pas les ressources portant le même nom 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.
- Lors de l'utilisation d'un CR pour restaurer dans un nouvel espace de noms, vous devez créer manuellement l'espace de noms de destination avant d'appliquer le CR. Trident Protect crée automatiquement les espaces de noms uniquement lors de l'utilisation du CLI.

Avant de commencer

Assurez-vous que la durée de validité 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, l'opération peut échouer.

- Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Consultez la "Documentation AWS IAM" pour plus d'informations sur les identifiants relatifs aux ressources AWS.

Lorsque vous restaurez des sauvegardes en utilisant Kopia comme moteur de déplacement de données, vous pouvez éventuellement spécifier des annotations dans le CR ou en utilisant la CLI pour contrôler le comportement du stockage temporaire utilisé par Kopia. Reportez-vous à l' "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez la commande `tridentctl-protect create --help` pour plus d'informations sur la spécification des annotations avec la CLI 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** : Le chemin à l'intérieur de AppVault où le contenu de la sauvegarde est stocké. 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 du AppVault où les contenus de sauvegarde sont stockés.
   - **spec.destinationApplicationName** : (*Facultatif*) Le nom de l'application restaurée. Si ce nom est fourni, l'application restaurée utilise ce nom. Si ce nom n'est pas fourni, l'application restaurée utilise le nom de l'application source.
   - **spec.namespaceMapping** : La correspondance de l'espace de noms source de l'opération de restauration avec l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par les informations 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
       destinationApplicationName: my-new-app-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 avec des étiquettes particulières :

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

   - **resourceFilter.resourceSelectionCriteria** : (Obligatoire pour le filtrage) Utilisez `Include` ou

`Exclude` pour 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 (group, kind, 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 Kubernetes metadata.name 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** : (*Optionnel*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes tel que défini 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 fichier `trident-protect-backup-restore-cr.yaml` avec les valeurs correctes, appliquez le CR :

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

Utilisez la ligne de commandes (CLI)

Étapes

1. Restaurez la sauvegarde dans un espace de noms différent, en remplaçant les valeurs entre crochets par les informations de votre environnement. L'`namespace-mapping`argument utilise des

espaces de noms séparés par des deux-points pour faire correspondre 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> \
--destination-app-name<custom_app_name>\
-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. Lorsque vous effectuez une restauration sur place, Trident Protect gère automatiquement les planifications de protection et les opérations en cours afin d'éviter les points de récupération invalides :

- Tous les plans de protection activés pour l'application sont désactivés avant le début de la restauration. Cela empêche les sauvegardes planifiées ou les instantanés de s'exécuter pendant la restauration des ressources de l'application.
- Après la restauration réussie, seules les planifications qui étaient activées avant la restauration sont réactivées. Les planifications qui étaient déjà désactivées restent désactivées.
- Toute opération de sauvegarde ou de capture instantanée en cours est annulée avant le début de la restauration. Si une opération n'est pas annulée dans les 5 minutes, la restauration se poursuit et un avertissement est consigné dans le statut du CR de restauration.

Avant de commencer

Assurez-vous que la durée de validité 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, l'opération peut échouer.

- Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Consultez la "Documentation AWS IAM" pour plus d'informations sur les identifiants relatifs aux ressources AWS.

> Lorsque vous restaurez des sauvegardes en utilisant Kopia comme moteur de déplacement de données, vous pouvez éventuellement spécifier des annotations dans le CR ou en utilisant la CLI pour contrôler le comportement du stockage temporaire utilisé par Kopia. Reportez-vous à l' "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez la commande `tridentctl-protect create --help` pour plus d'informations sur la spécification des annotations avec la CLI 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** : Le chemin à l'intérieur de AppVault où le contenu de la sauvegarde est stocké. 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 du AppVault où les contenus de sauvegarde sont stockés.

     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 avec des étiquettes particulières :

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

   - **resourceFilter.resourceSelectionCriteria** : (Obligatoire pour le filtrage) Utilisez `Include` ou `Exclude` pour 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 (group, kind, 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 Kubernetes metadata.name 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** : (*Optionnel*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes tel que défini 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 fichier `trident-protect-backup-ipr-cr.yaml` avec les valeurs correctes, appliquez le CR :

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

Utilisez la ligne de commandes (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. L'`backup`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 cluster différent en cas de problème avec le cluster d'origine.

- Lorsque vous restaurez des sauvegardes en utilisant Kopia comme moteur de déplacement de données, vous pouvez éventuellement spécifier des annotations dans le CR ou en utilisant la CLI pour contrôler le comportement du stockage temporaire utilisé par Kopia. Reportez-vous à l' "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez la commande `tridentctl-protect create --help` pour plus d'informations sur la spécification des annotations avec la CLI Trident Protect.
- Lors de l'utilisation d'un CR pour restaurer dans un nouvel espace de noms, vous devez créer manuellement l'espace de noms de destination avant d'appliquer le CR. Trident Protect crée automatiquement les espaces de noms uniquement lors de l'utilisation du CLI.

Avant de commencer

Assurez-vous que les conditions préalables suivantes sont remplies :

- Le cluster de destination a 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 le CR AppVault lors de l'exécution de la commande `tridentctl-protect get appvaultcontent`. Si des restrictions réseau empêchent l'accès, exécutez la CLI Trident Protect depuis un pod sur le cluster de destination à la place.
- Assurez-vous que la durée de validité du jeton de session AWS soit suffisante pour toute opération de restauration de longue durée. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.
  - Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
  - Consultez la "Documentation AWS" pour plus d'informations sur les identifiants relatifs aux ressources AWS.

Étapes

1. Vérifiez que la AppVault CR existe sur le cluster de destination à l'aide du plugin CLI Trident Protect :

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

   Si la AppVault CR n'existe pas sur le cluster de destination, créez-la en suivant les étapes dans "Utilisez les objets Trident Protect AppVault pour gérer les compartiments".

2. Consultez le contenu de la sauvegarde disponible de AppVault sur le cluster de destination, et notez `appArchivePath` de la sauvegarde que vous souhaitez restaurer :

```
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 le AppVault, y compris leurs clusters d'origine, les noms des applications correspondantes, les horodatages et les chemins d'accès aux archives.

Exemple de sortie :

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   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 :

> Lors de l'utilisation d'une CR, assurez-vous que l'espace de noms destiné à la restauration de l'application existe sur le cluster de destination.

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 du AppVault où les contenus de sauvegarde sont stockés.
   - **spec.appArchivePath** : (Obligatoire) Le chemin à l'intérieur de AppVault où le contenu de la sauvegarde est stocké. Utilisez la commande de l'étape 2 pour afficher le contenu de la sauvegarde et trouver `appArchivePath` la sauvegarde que vous souhaitez restaurer.
   - **spec.destinationApplicationName** : (*Facultatif*) Le nom de l'application restaurée. Si ce nom est fourni, l'application restaurée utilise ce nom. Si ce nom n'est pas fourni, l'application restaurée utilise le nom de l'application source.
   - **spec.namespaceMapping** : La correspondance de l'espace de noms source de l'opération de restauration avec l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par les informations 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
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

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

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

Utilisez la ligne de commandes (CLI)

1. Utilisez la commande suivante pour restaurer l'application, en remplaçant les valeurs entre crochets par les informations de votre environnement. L'argument namespace-mapping utilise des espaces de noms séparés par des deux-points pour associer les espaces de noms source aux espaces de noms de destination correspondants, 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> \
--destination-app-name <custom_app_name> \
--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 ressource personnalisé (CR), soit vers un autre espace de noms, soit vers l'espace de noms source d'origine. Lorsque vous restaurez un instantané vers un autre espace de noms à l'aide d'un SnapshotRestore CR, Trident Protect restaure l'application dans un nouvel espace de noms et crée un CR d'application pour l'application restaurée. Pour protéger l'application restaurée, créez des sauvegardes ou des instantanés à la demande, ou définissez une planification de protection.

- SnapshotRestore prend en charge l' `spec.storageClassMapping`attribut, mais uniquement lorsque les classes de stockage source et de destination utilisent le même système de stockage. Si vous tentez de restaurer vers une classe de stockage `StorageClass` qui utilise un système de stockage différent, l'opération de restauration échouera.
- Lors de l'utilisation d'un CR pour restaurer dans un nouvel espace de noms, vous devez créer manuellement l'espace de noms de destination avant d'appliquer le CR. Trident Protect crée automatiquement les espaces de noms uniquement lors de l'utilisation du CLI.

Avant de commencer

Assurez-vous que la durée de validité 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, l'opération peut échouer.

- Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Consultez la "Documentation AWS IAM" pour plus d'informations sur les identifiants relatifs aux 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 du AppVault où le contenu de l'instantané est stocké.
   - **spec.appArchivePath** : Le chemin à l'intérieur de AppVault où les contenus de l'instantané sont stockés. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

   - **spec.destinationApplicationName** : (*Facultatif*) Le nom de l'application restaurée. Si ce nom est fourni, l'application restaurée utilise ce nom. Si ce nom n'est pas fourni, l'application restaurée utilise le nom de l'application source.
   - **spec.namespaceMapping** : La correspondance de l'espace de noms source de l'opération de restauration avec l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par les informations 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 avec des étiquettes particulières :

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

   - **resourceFilter.resourceSelectionCriteria** : (Obligatoire pour le filtrage) Utilisez `Include` ou `Exclude` pour 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 (group, kind, 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 Kubernetes metadata.name 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** : (*Optionnel*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes tel que défini 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 fichier `trident-protect-snapshot-restore-cr.yaml` avec les valeurs correctes, appliquez le CR :

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

Utilisez la ligne de commandes (CLI)

Étapes

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

   ◦ L'`snapshot`argument utilise un espace de noms et un nom d'instantané au

format `<namespace>/<name>`.

- L'`namespace-mapping`argument utilise des espaces de noms séparés par des deux-points pour faire correspondre 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> \
--destination-app-name <custom_app_name> \
-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. Lorsque vous effectuez une restauration sur place, Trident Protect gère automatiquement les planifications de protection et les opérations en cours afin d'éviter les points de récupération invalides :

- Tous les plans de protection activés pour l'application sont désactivés avant le début de la restauration. Cela empêche les sauvegardes planifiées ou les instantanés de s'exécuter pendant la restauration des ressources de l'application.
- Après la restauration réussie, seules les planifications qui étaient activées avant la restauration sont réactivées. Les planifications qui étaient déjà désactivées restent désactivées.
- Toute opération de sauvegarde ou de capture instantanée en cours est annulée avant le début de la restauration. Si une opération n'est pas annulée dans les 5 minutes, la restauration se poursuit et un avertissement est consigné dans le statut du CR de restauration.

Avant de commencer

Assurez-vous que la durée de validité 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, l'opération peut échouer.

- Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Consultez la "Documentation AWS IAM" pour plus d'informations sur les identifiants relatifs aux 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 du AppVault où le contenu de l'instantané est stocké.
   - **spec.appArchivePath** : Le chemin à l'intérieur de AppVault où les contenus de l'instantané sont stockés. 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 avec des étiquettes particulières :

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

   - **resourceFilter.resourceSelectionCriteria** : (Obligatoire pour le filtrage) Utilisez `Include` ou `Exclude` pour 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 (group, kind, 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 Kubernetes metadata.name 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** : (*Optionnel*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes tel que défini 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 fichier `trident-protect-snapshot-ipr-cr.yaml` avec les valeurs correctes, appliquez le CR :

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

Utilisez la ligne de commandes (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 <namespace/snapshot_to_restore> \
-n <application_namespace>
```

Vérifiez l'état d'une opération de restauration

Vous pouvez utiliser la ligne de commandes pour vérifier l'état d'une opération de restauration qui est 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 crochets par les informations 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 de namespace 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 à jour pour correspondre aux étiquettes et annotations 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 déjà présentes sont remplacées 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 crucial des annotations d'espace de noms dans les environnements OpenShift. Les annotations d'espace de noms garantissent que les pods restaurés respectent les permissions et les configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (OpenShift SCC) et peuvent accéder aux volumes sans problème de permissions. Pour plus d'informations, consultez la "Documentation des contraintes de contexte de sécurité 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 -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

> Lors d'une opération de restauration ou de basculement, 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 des paramètres supplémentaires du chart Helm Trident Protect".

Si vous avez installé l'application source avec Helm avec le `--create-namespace` flag, un traitement spécial est appliqué à la clé de label `name`. Lors du processus de restauration ou de basculement, Trident Protect copie ce label dans l'espace de noms de destination, mais met à jour la valeur avec celle de l'espace de noms de destination si la valeur provenant 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 possédant 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"	• environment=production • conformité=hipaa • name=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 après l'opération de restauration ou de basculement. Certaines clés ont été ajoutées, d'autres ont été écrasées, et l' `name`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"	• name=ns-2 • conformité=hipaa • environment=production • rôle=base de données

Champs pris en charge

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

Correspondance des classes de stockage

L' `spec.storageClassMapping`attribut définit une correspondance 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 des classes de stockage différentes ou lors du changement de

système 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 une annotation n'est pas 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) autorisé pour que l'opération de déplacement de données soit 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"
protect.trident.netapp.io/pvc-bind-timeout-sec	chaîne	Temps maximal (en secondes) d'attente pour que tout nouveau PersistentVolumeClaims (PVC) atteigne la phase `Bound` avant que l'opération n'échoue. S'applique à tous les types de CR de restauration (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Utilisez une valeur plus élevée si votre backend de stockage ou votre cluster nécessite souvent plus de temps.	"1200" (20 minutes)

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 à jour pour correspondre aux étiquettes et annotations 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 déjà présentes sont remplacées 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 crucial des annotations d'espace de noms dans les environnements OpenShift. Les annotations d'espace de noms garantissent que les pods restaurés respectent les permissions et les configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (OpenShift SCC) et peuvent accéder aux volumes sans problème de permissions. Pour plus d'informations, consultez la "Documentation des contraintes de contexte de sécurité 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 -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

> Lors d'une opération de restauration ou de basculement, 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 des paramètres supplémentaires du chart Helm Trident Protect".

Si vous avez installé l'application source avec Helm avec le `--create-namespace` flag, un traitement spécial est appliqué à la clé de label `name`. Lors du processus de restauration ou de basculement, Trident Protect copie ce label dans l'espace de noms de destination, mais met à jour la valeur avec celle de l'espace de noms de destination si la valeur provenant 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 possédant 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"	• environment=production • conformité=hipaa • name=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 après l'opération de restauration ou de basculement. Certaines clés ont été ajoutées, d'autres ont été écrasées, et l' `name`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"	• name=ns-2 • conformité=hipaa • environment=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. "En savoir plus sur la configuration du gel du système de fichiers avec Trident Protect".

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

Lorsque vous utilisez une relation AppMirror pour protéger votre application, il existe des comportements spécifiques liés aux hooks d'exécution dont vous devez être conscient lors des opérations de basculement et d'inversion.

- Lors d'un basculement, les hooks d'exécution sont automatiquement copiés du cluster source vers le cluster de destination. Il n'est pas nécessaire de les recréer manuellement. Après le basculement, les hooks d'exécution sont présents sur l'application et s'exécuteront lors de toute action concernée.
- Lors d'une opération inverse ou d'une resynchronisation inverse, tous les hooks d'exécution existants sur l'application sont supprimés. Lorsque l'application source devient l'application de destination, ces hooks d'exécution ne sont plus valides et sont supprimés afin d'empêcher leur exécution.

Pour en savoir plus sur les hooks d'exécution, reportez-vous à "Gérer les hooks d'exécution de Trident Protect".

Configurer une relation de réplication

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

- Choisir la fréquence à laquelle vous souhaitez que Trident Protect prenne 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 (inclut les ressources Kubernetes ainsi que les données de volume persistant)
- Définition de l’heure à laquelle le snapshot sera pris

Étapes

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

Créez un AppVault en utilisant un CR

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 le nom que vous choisissez, car d'autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
- **spec.providerConfig** : (*Obligatoire*) Stocke la configuration nécessaire pour accéder à AppVault en utilisant le fournisseur spécifié. Choisissez un bucketName et tout autre détail nécessaire pour votre fournisseur. Notez les valeurs que vous choisissez, car d'autres fichiers CR nécessaires pour une relation de réplication font référence à ces valeurs. Reportez-vous à "Ressources personnalisées AppVault" pour des exemples de CR AppVault avec d'autres fournisseurs.
- **spec.providerCredentials** : (*Obligatoire*) Stocke les références à tout identifiant requis pour accéder à AppVault en utilisant le fournisseur spécifié.
  - **spec.providerCredentials.valueFromSecret** : (*Obligatoire*) Indique que la valeur de l'identifiant 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
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

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

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

Créez un AppVault à l'aide de la CLI

a. Créez le 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 le CR d'application source :

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 le nom que vous choisissez, car d'autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
- **spec.includedNamespaces** : (*Obligatoire*) Un tableau des espaces de noms et des étiquettes associées. Utilisez les noms des espaces de noms et, éventuellement, restreignez la portée des espaces de noms avec des 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 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 fichier `trident-protect-app-source.yaml` avec les valeurs correctes, appliquez le CR :

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

Créez l'application source à l'aide de la 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. Optionnellement, sur le cluster source, prenez un instantané de l'application source. Cet instantané est utilisé comme base pour l'application sur le cluster de destination. Si vous ignorez cette étape, vous devrez attendre la prochaine exécution de l'instantané planifié afin d'avoir un instantané récent. Pour créer un instantané à la demande, consultez "Créer un instantané à la demande".

4. Sur le cluster source, créez la CR de planification de réplication :

En complément du calendrier ci-dessous, il est recommandé de créer un calendrier de snapshot quotidien distinct, avec une période de conservation de 7 jours, afin de maintenir un snapshot commun entre les clusters ONTAP appariés. Cela garantit que les snapshots sont disponibles pendant jusqu'à 7 jours, mais la période de conservation peut être personnalisée en fonction des besoins de l'utilisateur.

En cas de basculement, le système peut utiliser ces instantanés pendant 7 jours maximum pour les opérations de restauration. Cette approche rend le processus de restauration plus rapide et plus efficace, car seules les modifications apportées depuis le dernier instantané seront transférées, pas toutes les données.

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

Créez la planification de réplication à l'aide d'un 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.
- **spéc.appVaultRef** : (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name de AppVault pour l'application source.
- **spec.applicationRef**: (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name de l'application source CR.
- **spec.backupRetention** : (*Obligatoire*) Ce champ est obligatoire et la valeur doit être définie sur 0.
- **spec.enabled** : doit être défini sur true.
- **spec.granularité** : Doit être définie sur `Custom`.
- **spec.recurrenceRule** : Définir une date de début en temps UTC et un intervalle de récurrence.
- **spec.snapshotRetention** : Doit être réglé sur 2.

Exemple 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 fichier `trident-protect-schedule.yaml` avec les valeurs correctes, appliquez 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 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 AppVault CR d'application source identique à la AppVault CR que vous avez appliquée sur le cluster source et nommez-la (par exemple, `trident-protect-appvault-primary-destination.yaml`).
6. Appliquez le CR :

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

7. Créez une ressource personnalisée AppVault CR de destination pour l'application de destination sur le cluster de destination. En fonction de votre fournisseur de stockage, modifiez un exemple dans "Ressources personnalisées AppVault" pour l'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 le nom que vous choisissez, car d'autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
      - **spec.providerConfig** : (*Obligatoire*) Stocke la configuration nécessaire pour accéder à AppVault en utilisant le fournisseur spécifié. Choisissez un `bucketName` et tous les autres détails nécessaires pour votre fournisseur. Notez les valeurs que vous choisissez, car d'autres fichiers CR nécessaires pour une relation de réplication font référence à ces valeurs. Reportez-vous à

"Ressources personnalisées AppVault" pour des exemples de CR AppVault avec d'autres fournisseurs.

- **spec.providerCredentials** : (*Obligatoire*) Stocke les références à tout identifiant requis pour accéder à AppVault en utilisant le fournisseur spécifié.
  - **spec.providerCredentials.valueFromSecret** : (*Obligatoire*) Indique que la valeur de l'identifiant 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
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Après avoir rempli le fichier `trident-protect-appvault-secondary-destination.yaml` avec les valeurs correctes, appliquez 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.

> Lors de l'utilisation d'un CR, créez manuellement l'espace de noms de destination avant d'appliquer le CR. Trident Protect crée automatiquement les espaces de noms uniquement lors de l'utilisation du CLI.

Créez un AppMirrorRelationship 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 AppVault pour l'application de destination sur le cluster de destination.
- **spec.namespaceMapping**: (*Obligatoire*) Les espaces de noms de destination et de 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 AppVault pour l'application source.
- **spec.sourceApplicationName**: (*Obligatoire*) Cette valeur doit correspondre au nom de l'application source que vous avez définie dans la CR 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** : (*Facultatif*) 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 appairée avec l'environnement source. Si la classe de stockage n'est pas fournie, la classe de stockage par défaut du cluster sera utilisée par défaut.
- **spec.recurrenceRule** : Définir une date de début en temps UTC et un intervalle de récurrence.

  Exemple 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 fichier `trident-protect-relationship.yaml` avec les valeurs correctes, appliquez le CR :

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

Créez un AppMirrorRelationship à l'aide de la CLI

a. Créez et appliquez l'objet AppMirrorRelationship, en remplaçant les valeurs entre crochets par les informations 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 ne met pas l'application hors service 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 changez la valeur de **spec.desiredState** en `Promoted`.
2. Enregistrez le fichier CR.
3. Appliquez le CR :

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

4. (*Facultatif*) Créez tous les plans de protection dont vous avez besoin 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 basculée

L'opération de resynchronisation rétablit la relation de réplication. Après avoir effectué 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 arrête 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 garantit que les dernières modifications 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 changez la valeur de spec.desiredState à `Established`.
3. Enregistrez le fichier CR.
4. Appliquez le CR :

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

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

Resynchroniser à l'inverse une relation de réplication ayant échoué

Lorsque vous effectuez 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 le CR AppMirrorRelationship. Cela fait en sorte que la destination devienne la source. S'il reste des planifications 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 configurer la relation aux clusters opposés.
3. Assurez-vous que la nouvelle destination (cluster source d'origine) est configurée avec les deux AppVault CR.
4. Configurez une relation de réplication sur le cluster opposé, en configurant les valeurs pour la direction inverse.

Inverser le sens de réplication de l'application

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 cette situation, vous échangez 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 plannings de la stratégie de protection pour l'application source.

b. Créez 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 AppVault pour l'application source.
   - **spec.ApplicationRef** : (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name du fichier CR de l'application source.

     Exemple 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 fichier `trident-protect-shutdownsnapshot.yaml` avec les valeurs correctes, appliquez le CR :

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

Créez un instantané d'arrêt à l'aide de la CLI

a. Créez l'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, après la fin de la capture instantanée de l'arrêt, obtenez l'état de la capture instantanée de l'arrêt :

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

3. Sur le cluster source, trouvez 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 le basename ; cela correspond à 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 champ `spec.promotedSnapshot` 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 dans Resynchroniser à l'inverse une relation de réplication ayant échoué.
6. Activez les plannings de protection sur le nouveau cluster source.

Résultat

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

- Une capture instantanée est prise des ressources Kubernetes de l'application source d'origine.
- 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'app 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 pris avant 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 arrière » après une opération de basculement en utilisant la séquence d'opérations suivante. Dans ce workflow pour restaurer la direction de réplication d'origine, Trident Protect réplique (resynchronise) toutes les modifications de l'application vers l'application source d'origine avant d'inverser la direction de la 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 échoué.
- Inversez 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.

- Inversez le sens de la réplication.

Étapes

1. Effectuez les Resynchroniser à l'inverse une relation de réplication ayant échoué étapes.
2. Effectuez les Inverser le sens de réplication de l'application étapes.

Supprimer une relation de réplication

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

Étapes

1. Sur le cluster de destination actuel, supprimez le CR 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 des données de sauvegarde.

> Lors de la migration d'une application, tous les points d'exécution configurés pour l'application sont migrés avec l'application. Si un point 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 pour les scénarios suivants, vous pouvez automatiser des tâches spécifiques de sauvegarde et de restauration.

Cloner vers 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 vers un cluster différent

Pour cloner une application sur un autre cluster (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 AppVault CR du 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 à une autre en restaurant une sauvegarde dans la classe de stockage de destination.

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

```
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 à l'intérieur de AppVault où les contenus de l'instantané sont stockés. 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 du AppVault où le contenu de l'instantané est stocké.
   - **spec.namespaceMapping** : La correspondance de l'espace de noms source de l'opération de restauration avec l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par les informations 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. Optionnellement, si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :
   - **resourceFilter.resourceSelectionCriteria** : (Obligatoire pour le filtrage) Utilisez `include or exclude` pour 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 (group, kind, 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 Kubernetes metadata.name 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** : (*Optionnel*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes tel que défini 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 fichier `trident-protect-snapshot-restore-cr.yaml` avec les valeurs correctes, appliquez 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.

   - L'`snapshot`argument utilise un espace de noms et un nom d'instantané au format `<namespace>/<name>.
   - L'`namespace-mapping`argument utilise des espaces de noms séparés par des deux-points pour faire correspondre 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 point 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 avez une application de base de données, vous pouvez utiliser un point d'exécution pour suspendre toutes les transactions de la base de données avant un instantané, puis reprendre les transactions après la fin de l'instantané. Cela garantit des instantanés cohérents au niveau des 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
- Après l'instantané
- Pré-sauvegarde
- Post-sauvegarde
- Après restauration
- Après le basculement

Ordre d'exécution

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

1. Les hooks d'exécution personnalisés applicables avant l'opération sont exécutés sur les conteneurs appropriés. Vous pouvez créer et exécuter autant de hooks d'exécution personnalisés que nécessaire, mais l'ordre d'exécution de ces hooks avant l'opération n'est ni garanti ni configurable.
2. Des blocages du système de fichiers peuvent survenir, le cas échéant. "En savoir plus 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. 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 nécessaire, 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, pre-snapshot), l'ordre d'exécution de ces hooks n'est pas garanti. Cependant, l'ordre d'exécution des hooks de types différents est garanti. Par exemple, voici l'ordre d'exécution d'une configuration qui comporte tous les différents types de hooks :

1. Hooks de pré-snapshot exécutés
2. Hooks post-snapshot 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 hook 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 le snapshot 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 concernant les hooks d'exécution personnalisés

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

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

Parce que les hooks d'exécution réduisent souvent ou désactivent complètement la fonctionnalité de l'application sur laquelle ils s'exécutent, vous devez toujours essayer de minimiser le temps que vos hooks d'exécution personnalisés prennent pour s'exécuter. Si vous lancez une opération de sauvegarde ou de snapshot avec des hooks d'exécution associés, puis l'annulez, 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 complétée.

Filtres de hooks d'exécution

Lorsque vous ajoutez ou modifiez un hook d'exécution pour une application, vous pouvez ajouter des filtres au hook d'exécution afin de gérer les conteneurs auxquels le hook correspond. 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 où les hooks d'exécution s'exécutent sur certains, mais pas nécessairement tous, les conteneurs identiques. Si vous créez plusieurs filtres pour un seul hook d'exécution, ils sont combinés avec un opérateur logique AND. 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 dans 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 Regular Expression 2 (RE2), qui ne prend pas en charge la création d'un filtre excluant des conteneurs de la liste des correspondances. Pour obtenir des informations sur la syntaxe que Trident Protect prend en charge pour les expressions régulières dans les filtres de hooks d'exécution, consultez "Prise en charge de la syntaxe Regular Expression 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 la restauration ou du clonage se trouvent dans des espaces de noms différents, le filtre d'espace de noms ne s'applique qu'à l'espace de noms de destination.

Exemples de hooks d'exécution

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

Créer un hook d'exécution

Vous pouvez créer un hook d'exécution personnalisé pour une application à l'aide de Trident Protect. Vous devez disposer des autorisations Owner, Admin ou Member pour créer des hooks 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*) Chaîne de caractères indiquant à quelle étape de l'action le hook d'exécution doit s'exécuter. Valeurs possibles :
     - Pré
     - Publier
   - **spec.action** : (*Obligatoire*) Une chaîne indiquant l'action que le hook d'exécution effectuera, en supposant que les filtres de hook d'exécution spécifiés correspondent. Valeurs possibles :
     - Instantané
     - Sauvegarde
     - Restaurer
     - Basculement
   - **spec.enabled** : (*Optionnel*) Indique si ce hook d'exécution est activé ou désactivé. Si non spécifié, la valeur par défaut est true.
   - **spec.hookSource**: (*Obligatoire*) Une chaîne contenant le script de hook encodé en base64.
   - **spec.timeout** : (*Optionnel*) Un nombre définissant combien de temps, en minutes, 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** : (*Facultatif*) Liste facultative de paires clé-valeur de critères, chaque paire constituant un filtre d'exécution hook. Vous pouvez ajouter jusqu'à 10 filtres par exécution hook.
   - **spec.matchingCriteria.type** : (*Optionnel*) Une chaîne identifiant le type de filtre du hook d'exécution. Valeurs possibles :
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (*Optionnel*) Une chaîne de caractères ou une expression régulière identifiant la valeur du filtre du hook d'exécution.

     Exemple 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
```

Utilisez la ligne de commandes (CLI)

Étapes

1. Créez le hook d'exécution en remplaçant les valeurs entre crochets par les informations 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 script d'exécution à des fins de test ou si vous devez le relancer manuellement après un échec. Vous devez disposer des autorisations de propriétaire, d'administrateur ou de membre pour exécuter manuellement des scripts d'exécution.

L'exécution manuelle d'un hook consiste en deux étapes de base :

1. Créez une sauvegarde de ressources, qui collecte les ressources et crée une sauvegarde de celles-ci, déterminant où le hook s'exécutera
2. Exécutez le hook 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 ressources.
   - **spec.appVaultRef** : (*Obligatoire*) Le nom du AppVault où les contenus de sauvegarde sont stockés.
   - **spec.appArchivePath** : Le chemin à l'intérieur de AppVault où le contenu de la sauvegarde est stocké. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

     Exemple 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
```

Utilisez la ligne de commandes (CLI)

Étapes

1. Créez la sauvegarde en remplaçant les valeurs entre crochets par les informations 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 cet exemple de commande à plusieurs reprises jusqu'à la fin de l'opération :

```
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 ResourceBackup CR que vous avez créée à l'étape 1.
   - **spec.appVaultRef** : (*Obligatoire*) Assurez-vous que cette valeur corresponde à la appVaultRef du ResourceBackup CR que vous avez créée à l'étape 1.
   - **spec.appArchivePath** : Assurez-vous que cette valeur corresponde à la appArchivePath du CR ResourceBackup que vous avez créé 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 effectuera, en supposant que les filtres de hook d'exécution spécifiés correspondent. Valeurs possibles :
     - Instantané
     - Sauvegarde
     - Restaurer
     - Basculement
   - **spec.stage** : (*Obligatoire*) Chaîne de caractères indiquant à quelle étape de l'action le hook d'exécution doit s'exécuter. Ce hook run n'exécutera aucun hook à une autre étape. Valeurs possibles :
     - Pré
     - Publier

     Exemple 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
```

Utilisez la ligne de commandes (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 de l'exécution du hook. Vous pouvez exécuter cette commande plusieurs fois jusqu'à la fin de l'opération :

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

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

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

Informations sur le copyright

Informations sur les marques commerciales