管理及保護應用程式
Trident

NetApp
March 05, 2026

This PDF was generated from https://docs.netapp.com/zh-tw/trident-2410/trident-protect/trident-protect-appvault-custom-resources.html on March 05, 2026. Always check docs.netapp.com for the latest.

目錄

管理及保護應用程式 1
使用Trident Protect AppVault 物件來管理儲存桶。 1
金鑰產生與 AppVault 定義範例 1
使用 AppVault 瀏覽器檢視 AppVault 資訊 7
移除 AppVault 8
使用Trident Protect 定義管理應用程式 9
建立 AppVault CR 9
定義應用程式 9
使用Trident Protect 保護應用程式 10
建立隨需快照 11
建立隨選備份 13
建立資料保護排程 15
刪除快照 17
刪除備份 18
檢查備份作業的狀態 18
啟用 NetApp 檔案（ anf ）作業的備份與還原 18
使用Trident Protect 恢復應用程式 19
還原和容錯移轉作業期間的命名空間註釋和標籤 19
從備份還原至不同的命名空間 21
從備份還原至原始命名空間 24
從備份還原至不同的叢集 26
從快照還原至不同的命名空間 29
從快照還原至原始命名空間 32
檢查還原作業的狀態 34
使用NetApp SnapMirror和Trident Protect 複製應用程式 35
還原和容錯移轉作業期間的命名空間註釋和標籤 35
設定複寫關係 36
反轉應用程式複寫方向 44
使用Trident Protect 遷移應用程式 47
備份與還原作業 47
將應用程式從一個儲存類別移轉至另一個儲存類別 48
管理Trident Protect 執行鉤子 51
執行掛勾的類型 51
關於自訂執行掛勾的重要注意事項 52
執行攔截篩選器 52
執行攔截範例 52
建立執行掛鉤 52

管理及保護應用程式

使用**Trident Protect AppVault** 物件來管理儲存桶。

Trident Protect 的儲存桶自訂資源 (CR) 稱為 AppVault。AppVault 物件是儲存桶的聲明性 Kubernetes 工作流程表示。AppVault CR 包含儲存桶在保護作業（例如備份、快照、復原作業和SnapMirror複製）中所使用的必要配置。只有管理員才能建立應用保險庫。

金鑰產生與 **AppVault** 定義範例

定義 AppVault CR 時，您需要加入認證，才能存取供應商託管的資源。您產生認證金鑰的方式會因供應商而異。以下是數個提供者的命令列金鑰產生範例，接著是每個提供者的 AppVault 定義範例。

關鍵創造範例

您可以使用下列範例來建立每個雲端供應商認證的金鑰。

Google Cloud

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

Amazon S3 （AWS）

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

Microsoft Azure

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

一般S3

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

ONTAP S3

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

StorageGRID S3

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

AppVault CR 範例

您可以使用下列 CR 範例，為每個雲端供應商建立 AppVault 物件。

Google Cloud

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

Amazon S3 （AWS）

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

Microsoft Azure

```yaml
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

一般S3

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

ONTAP S3

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

StorageGRID S3

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

使用**Trident Protect CLI** 建立 **AppVault** 的範例

您可以使用下列 CLI 命令範例，為每個供應商建立 AppVault CRS 。

Google Cloud

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

Amazon S3 （AWS）

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

Microsoft Azure

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

一般S3

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

ONTAP S3

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

StorageGRID S3

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

使用 AppVault 瀏覽器檢視 AppVault 資訊

您可以使用Trident Protect CLI 外掛程式查看有關叢集上建立的 AppVault 物件的資訊。

步驟

1. 檢視 AppVault 物件的內容：

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

   - 輸出範例 * ：

```
+-------------+-------+----------+----------------------------
+---------------------------+
|   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. （可選）要查看每個資源的 AppVaultPath ，請使用標誌 `--show-paths`。

   只有在Trident Protect helm 安裝中指定了叢集名稱時，表格第一列中的叢集名稱才可用。例如： `--set clusterName=production1` 。

移除 AppVault

您可以隨時移除 AppVault 物件。

> 刪除 AppVault 物件之前，請勿移除 `finalizers` AppVault CR 中的機碼。如果您這麼做，可能會導致 AppVault 貯體中的剩餘資料，以及叢集中的孤立資源。

開始之前

確保您已刪除儲存在相關儲存庫中的所有快照和備份。

使用 **Kubernetes CLI** 移除 **AppVault**

1. 移除 AppVault 物件，以要移除的 AppVault 物件名稱取代 `appvault_name`：

```
kubectl delete appvault <appvault_name> -n trident-protect
```

使用**Trident Protect CLI** 刪除 **AppVault**

1. 移除 AppVault 物件，以要移除的 AppVault 物件名稱取代 `appvault_name`：

```
tridentctl-protect delete appvault <appvault_name> -n trident-
protect
```

使用**Trident Protect** 定義管理應用程式

您可以透過建立應用程式 CR 和關聯的 AppVault CR 來定義要使用Trident Protect 管理的應用程式。

建立 **AppVault CR**

您需要建立一個 AppVault CR，該 CR 將在對應用程式執行資料保護操作時使用，並且 AppVault CR 需要位於安裝了Trident Protect 的叢集上。AppVault CR 是針對您的特定環境的；有關 AppVault CR 的範例，請參閱："AppVault 自訂資源。"

定義應用程式

您需要定義要使用Trident Protect 管理的每個應用程式。您可以透過手動建立應用程式 CR 或使用Trident Protect CLI 來定義要管理的應用程式。

使用 **CR** 新增應用程式

步驟

1. 建立目的地應用程式 CR 檔案：

   a. 建立自訂資源（ CR ）檔案並命名（例如 `maria-app.yaml`）。

   b. 設定下列屬性：

      - * metadata.name*: （ _required ）應用程式自訂資源的名稱。請注意您選擇的名稱，因為保護作業所需的其他 CR 檔案都會參照此值。
      - * spec.includedNamespaces*: （ _required _ ）使用命名空間標籤或命名空間名稱來指定應用程式資源所在的命名空間。應用程式命名空間必須是此清單的一部分。

        YAML 範例：

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

2. 建立應用程式 CR 以符合您的環境之後，請套用 CR 。例如：

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

使用 **CLI** 新增應用程式

步驟

1. 建立並套用應用程式定義，以環境資訊取代方括號中的值。您可以使用以逗號分隔的清單，將命名空間和資源包含在應用程式定義中，並附上下列範例所示的引數：

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

使用Trident Protect 保護應用程式

您可以使用自動保護策略或臨時保護策略，透過拍攝快照和備份來保護Trident Protect 管

理的所有應用程式。

> 您可以設定Trident Protect 在資料保護作業期間凍結和解凍檔案系統。"了解更多關於使用Trident Protect 設定檔系統凍結的信息"。

建立隨需快照

您可以隨時建立隨需快照。

使用 **CR** 建立快照

步驟

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-snapshot-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
   - * metadata.name*: （_required）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - **SPEC.applicationRef** ：要快照的應用程式的 Kubernetes 名稱。
   - **spec.appVaultRef** ：（_required _）應儲存快照內容（中繼資料）的 AppVault 名稱。
   - **spec.relaimersPolicy** ：（*Optional*）定義刪除快照 CR 時，應用程式歸檔會發生什麼情況。這表示即使設定為，快照也 `Retain`會被刪除。有效選項：
     - `Retain` （預設）
     - `Delete`

```yaml
---
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. 在您以正確的值填入檔案之後 `trident-protect-snapshot-cr.yaml` 、請套用 CR ：

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

使用 **CLI** 建立快照

步驟

1. 建立快照，以您環境的資訊取代方括號中的值。例如：

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

建立隨選備份

您可以隨時備份應用程式。

使用 **CR** 建立備份

步驟

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-backup-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
   - * metadata.name*: （_required）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - **SPEC.applicationRef** ：（_required _）要備份的應用程式 Kubernetes 名稱。
   - **spec.appVaultRef** ：（_required _）應儲存備份內容的 AppVault 名稱。
   - *spec.dataMover * ：（*Optional*）字串，指出備份作業所使用的備份工具。可能的值（區分大小寫）：
     - `Restic`
     - `Kopia` （預設）
   - **spec.reClaimPolicy** ：（*Optional*）定義備份從宣告中釋出時會發生什麼情況。可能值：
     - `Delete`
     - `Retain` （預設）
   - **Spec.snapshotRef** ：（*Optional*）：用於備份來源的快照名稱。如果未提供，將會建立並備份暫存快照。

```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. 在您以正確的值填入檔案之後 `trident-protect-backup-cr.yaml` 、請套用 CR ：

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

使用 **CLI** 建立備份

步驟

1. 建立備份，以您環境的資訊取代括號中的值。例如：

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

建立資料保護排程

保護原則可在已定義的排程中建立快照、備份或兩者、以保護應用程式。您可以選擇每小時、每天、每週和每月建立快照和備份、也可以指定要保留的複本數量。

使用 **CR** 建立排程

步驟

1. 建立自訂資源（ CR ）檔案並命名為 `trident-protect-schedule-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
   - * metadata.name*: （ _required ）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - *spec.dataMover * ：（ *Optional* ）字串，指出備份作業所使用的備份工具。可能的值（區分大小寫）：
     - `Restic`
     - `Kopia` （預設）
   - **SPEC.applicationRef** ：要備份之應用程式的 Kubernetes 名稱。
   - **spec.appVaultRef** ：（ _required _ ）應儲存備份內容的 AppVault 名稱。
   - *SPEC.BackupRetention * ：要保留的備份數量。零表示不應建立備份。
   - *spec.snapshotRetention * ：要保留的快照數。零表示不應建立任何快照。
   - * spec.granularity*: 執行排程的頻率。可能的值、以及必要的相關欄位：
     - `hourly` （要求您指定 `spec.minute`）
     - `daily` （要求您指定 `spec.minute` 和 `spec.hour`）
     - `weekly`（要求您指定 `spec.minute, spec.hour`，和 `spec.dayOfWeek`）
     - `monthly`（要求您指定 `spec.minute, spec.hour`，和 `spec.dayOfMonth`）
   - **spec.dayOfMontth** ：（ *Optional* ）排程應執行的月份日期（ 1 - 31 ）。如果精細度設為、則此欄位為必 `monthly`填。
   - *spec.dayOfWeek * ：（ _Optional _ ）排程應執行的一週中的一天（ 0 - 7 ）。0 或 7 的值表示星期日。如果精細度設為、則此欄位為必 `weekly`填。
   - *spec.hour * ：（ _Optional _ ）排程應執行的一天中的小時（ 0 - 23 ）。如果精細度設置爲、或，則此字段爲必填字段 `daily` `weekly` `monthly`。
   - * 規格分鐘 * ：（ _ 選用 _ ）排程應執行的小時（ 0 - 59 ）分鐘。如果精細度設置爲、、或，則此字段爲必填字段 `hourly` `daily` `weekly` `monthly`。

```yaml
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: <monthly>
  dayOfMonth: "1"
  dayOfWeek: "0"
  hour: "0"
  minute: "0"
```

3. 在您以正確的值填入檔案之後 `trident-protect-schedule-cr.yaml` 、請套用 CR ：

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

使用 **CLI** 建立排程

步驟

1. 建立保護排程，以環境資訊取代方括號中的值。例如：

> 您可以使用 `tridentctl-protect create schedule --help`來檢視此命令的詳細說明資訊。

```
tridentctl-protect create schedule <my_schedule_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> --backup
-retention <how_many_backups_to_retain> --data-mover
<kopia_or_restic> --day-of-month <day_of_month_to_run_schedule>
--day-of-week <day_of_month_to_run_schedule> --granularity
<frequency_to_run> --hour <hour_of_day_to_run> --minute
<minute_of_hour_to_run> --recurrence-rule <recurrence> --snapshot
-retention <how_many_snapshots_to_retain> -n <application_namespace>
```

刪除快照

刪除不再需要的排程或隨需快照。

步驟

1. 移除與快照相關的 Snapshot CR ：

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

刪除備份

刪除不再需要的排程或隨需備份。

步驟

1. 移除與備份相關的備份 CR ：

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

檢查備份作業的狀態

您可以使用命令列來檢查正在進行，已完成或已失敗的備份作業狀態。

步驟

1. 使用下列命令可擷取備份作業的狀態，以環境中的資訊取代方括號中的值：

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

啟用 NetApp 檔案（ anf ）作業的備份與還原

如果您已安裝Trident Protect，則可以為使用 azure-netapp-files 儲存類別且在Trident 24.06 之前建立的儲存後端啟用節省空間的備份和還原功能。此功能適用於 NFSv4 卷，並且不會佔用容量池中的額外空間。

開始之前

請確認下列事項：

- 您已安裝Trident Protect。
- 您已在Trident Protect中定義了一個應用程式。在您完成此步驟之前，此應用程式的保護功能將受到限制。
- 您已 `azure-netapp-files` 選擇儲存後端的預設儲存類別。

展開以進行組態步驟

1. 如果 anf Volume 是在升級至 Trident 24.10 之前建立的，請在 Trident 中執行下列動作：

   a. 針對每個以 NetApp 檔案為基礎且與應用程式相關的 PV ，啟用快照目錄：

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

   b. 確認已為每個相關的 PV 啟用快照目錄：

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

   回應：

   ```
   snapshotDirectory: "true"
   ```

   +
   如果未啟用快照目錄， Trident Protect 將選擇常規備份功能，該功能會在備份過程中暫時佔用容量池中的空間。在這種情況下，請確保容量池中有足夠的空間來建立與被備份磁碟區大小相同的臨時磁碟區。

結果

該應用程式已準備好使用Trident Protect 進行備份和還原。每個 PVC 也可供其他應用程式用於備份和還原。

使用Trident Protect 恢復應用程式

您可以使用Trident Protect 從快照或備份中還原您的應用程式。將應用程式還原到同一群集時，從現有快照恢復速度會更快。

> ⓘ 當您還原應用程式時，為應用程式設定的所有執行掛鉤都會隨應用程式一起還原。如果存在還原後執行掛鉤，則會在還原作業中自動執行。

還原和容錯移轉作業期間的命名空間註釋和標籤

在還原和容錯移轉作業期間，目的地命名空間中的標籤和註釋會與來源命名空間中的標籤和註釋相符。會新增來源命名空間中不存在的標籤或註釋，並覆寫已存在的任何標籤或註釋，以符合來源命名空間的值。只存在於目的地命名空間上的標籤或註釋會保持不變。

> ⓘ 如果您使用 RedHat OpenShift ，請務必注意 OpenShift 環境中命名空間註釋的關鍵作用。命名空間註釋可確保還原的 Pod 符合 OpenShift 安全性內容限制（ SCC ）所定義的適當權限和安全性組態，而且無需權限問題即可存取磁碟區。如需詳細資訊，請 "OpenShift 安全性內容限制文件"參閱。

您可以在執行還原或容錯移轉作業之前，先設定 Kubernetes 環境變數，以避免覆寫目的地命名空間中的特定註釋 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`。例如：

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

如果您使用 Helm 安裝了來源應用程序， `--create-namespace`國旗，給予特殊待遇 `name`標籤鍵。在復原或故障轉移過程中， Trident Protect 會將此標籤複製到目標命名空間，但如果來源命名空間的值與來源命名空間的值匹配，則會將值更新為目標命名空間的值。如果此值與來源命名空間不匹配，則會將其複製到目標命名空間，而不做任何變更。

範例

以下範例提供來源和目的地命名空間，每個命名空間都有不同的註釋和標籤。您可以查看作業前後目的地命名空間的狀態，以及註釋和標籤在目的地命名空間中的組合或覆寫方式。

還原或容錯移轉作業之前

下表說明還原或容錯移轉作業之前的範例來源和目的地命名空間狀態：

命名空間	註釋	標籤
命名空間 nS-1 （來源）	• annotation.one / 機碼：「 updatedvalue 」 • annotation.b2/key ：「 true 」	• 環境 = 正式作業 • Compliance = HIPAA • NAME=nS-1
命名空間 nS-2 （目的地）	• annotation.one / 機碼：「 true 」 • annotation.the/key ：「 FALSE 」	• role = 資料庫

還原作業之後

下表說明還原或容錯移轉作業之後範例目的地命名空間的狀態。某些金鑰已新增，部分已覆寫， `name`標籤已更新以符合目的地命名空間：

命名空間	註釋	標籤
命名空間 nS-2 （目的地）	• annotation.one / 機碼：「 updatedvalue 」 • annotation.b2/key ：「 true 」 • annotation.the/key ：「 FALSE 」	• NAME=nS-2 • Compliance = HIPAA • 環境 = 正式作業 • role = 資料庫

從備份還原至不同的命名空間

當您使用 BackupRestore CR 將備份還原到不同的命名空間時， Trident Protect 會在新的命名空間中還原應用程序，並為還原的應用程式建立一個應用程式 CR。為了保護已還原的應用程序，可以建立按需備份或快照，或製定保護計劃。

> 將備份還原至具有現有資源的不同命名空間，並不會改變任何與備份中共用名稱的資源。若要還原備份中的所有資源，請刪除並重新建立目標命名空間，或將備份還原至新的命名空間。

使用 **CR**

步驟

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-backup-restore-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
   - * metadata.name*: （_required）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - **spec.appArchivePath** ：儲存備份內容的 AppVault 內部路徑。您可以使用下列命令來尋找此路徑：

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

   - **spec.appVaultRef** ：（_required _）儲存備份內容的 AppVault 名稱。
   - **spec.namespaceMapping**: 將還原作業的來源命名空間對應至目的地命名空間。以環境中的資訊取代 `my-source-namespace`和 `my-destination-namespace`。
   - **spec.storageClassMapping** ：將還原作業的來源儲存類別對應至目的地儲存類別。以環境中的資訊取代 `destinationStorageClass`和 `sourceStorageClass`。

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

3. （*Optional*）如果您只需要選取應用程式的某些資源來還原，請新增篩選功能，以包含或排除標記有特定標籤的資源：
   - **resourceFilter.resourceSelectionCriteria** ：（篩選所需）使用 `Include`或包含或 `Exclude`排除在 resourceMatchers 中定義的資源。新增下列資源配置工具參數、以定義要納入或排除的資源：
     - **resourceFilter.resourceMatchers** ：一組 resourceMatcher 物件。如果您在此陣列中定義多個元素，它們會比對為 OR 作業，而每個元素（群組，種類，版本）內的欄位會比對為 AND 作業。
       - **resourceMatchers[].group** ：（*Optional*）要篩選的資源群組。

- **resourceMatchers[].cher** ：（ *Optional* ）要篩選的資源種類。
- **resourceMatchers[].version** ：（ *Optional* ）要篩選的資源版本。
- 要篩選之資源的 Kubernetes metadata.name 欄位中的 * resourceMatchers[].names* ：（ *Optional* ）名稱。
- 要篩選之資源的 Kubernetes metadata.name 欄位中的 * resourceMatchers[].names* ：（ *Optional* ）命名空間。
- 資源的 Kubernetes metadata.name 欄位中的 *resourceMatchers[].labelSelectors * ：（ *Optional* ） Label 選取器字串，如中所定義 "Kubernetes文件"。例如 `"trident.netapp.io/os=linux"`：。

  例如：

```
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. 在您以正確的值填入檔案之後 `trident-protect-backup-restore-cr.yaml` 、請套用 CR ：

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

使用**CLI**

步驟

1. 將備份還原至不同的命名空間，以環境中的資訊取代括弧中的值。此 `namespace-mapping`引數使用以冒號分隔的命名空間，以格式將來源命名空間對應至正確的目的地命名空間 `source1:dest1,source2:dest2`。例如：

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

從備份還原至原始命名空間

您可以隨時將備份還原至原始命名空間。

使用 **CR**

步驟

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-backup-ipr-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
   - * metadata.name*: （_required）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - **spec.appArchivePath** ：儲存備份內容的 AppVault 內部路徑。您可以使用下列命令來尋找此路徑：

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

   - **spec.appVaultRef** ：（_required _）儲存備份內容的 AppVault 名稱。

     例如：

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
```

3. （*Optional*）如果您只需要選取應用程式的某些資源來還原，請新增篩選功能，以包含或排除標記有特定標籤的資源：
   - **resourceFilter.resourceSelectionCriteria** ：（篩選所需）使用 `Include`或包含或 `Exclude`排除在 resourceMatchers 中定義的資源。新增下列資源配置工具參數、以定義要納入或排除的資源：
     - **resourceFilter.resourceMatchers** ：一組 resourceMatcher 物件。如果您在此陣列中定義多個元素，它們會比對為 OR 作業，而每個元素（群組，種類，版本）內的欄位會比對為 AND 作業。
       - **resourceMatchers[].group** ：（*Optional*）要篩選的資源群組。
       - **resourceMatchers[].cher** ：（*Optional*）要篩選的資源種類。
       - **resourceMatchers[].version** ：（*Optional*）要篩選的資源版本。
       - 要篩選之資源的 Kubernetes metadata.name 欄位中的 * resourceMatchers[].names* ：（*Optional*）名稱。
       - 要篩選之資源的 Kubernetes metadata.name 欄位中的 * resourceMatchers[].names* ：（*Optional*）命名空間。
       - 資源的 Kubernetes metadata.name 欄位中的 *resourceMatchers[].labelSelectors * ：（

Optional） Label 選取器字串，如中所定義 "Kubernetes文件"。例如 `"trident.netapp.io/os=linux"`：。

例如：

```
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. 在您以正確的值填入檔案之後 `trident-protect-backup-ipr-cr.yaml` 、請套用 CR ：

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

使用**CLI**

步驟

1. 將備份還原至原始命名空間，以環境中的資訊取代括弧中的值。 `backup`引數使用的名稱空間和備份名稱格式為 `<namespace>/<name>`。例如：

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

從備份還原至不同的叢集

如果原始叢集發生問題，您可以將備份還原至不同的叢集。

開始之前

確保符合下列先決條件：

- 目標叢集已安裝Trident Protect。
- 目的地叢集可存取與儲存備份的來源叢集相同 AppVault 的儲存區路徑。

步驟

1. 使用Trident Protect CLI 外掛程式檢查目標叢集上 AppVault CR 的可用性：

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

> 確保目的地叢集上存在用於應用程式還原的命名空間。

2. 從目的地叢集檢視可用 AppVault 的備份內容：

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

執行此命令會顯示 AppVault 中的可用備份，包括其原始叢集，對應的應用程式名稱，時間戳記和歸檔路徑。

  - 輸出範例：*

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   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. 使用 AppVault 名稱和歸檔路徑將應用程式還原至目的地叢集：

使用 **CR**

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-backup-restore-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
   - * metadata.name*: （_required）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - **spec.appVaultRef** ：（_required _）儲存備份內容的 AppVault 名稱。
   - **spec.appArchivePath** ：儲存備份內容的 AppVault 內部路徑。您可以使用下列命令來尋找此路徑：

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

     > 如果無法使用 BackupRestore CR ，您可以使用步驟 2 所述的命令來檢視備份內容。

   - **spec.namespaceMapping**: 將還原作業的來源命名空間對應至目的地命名空間。以環境中的資訊取代 `my-source-namespace`和 `my-destination-namespace`。

     例如：

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

3. 在您以正確的值填入檔案之後 `trident-protect-backup-restore-cr.yaml` 、請套用 CR ：

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

使用**CLI**

1. 使用下列命令還原應用程式，將方括號中的值取代為您環境中的資訊。命名空間對應引數使用以冒號分隔的命名空間，將來源命名空間對應到正確的目的地命名空間，格式為 source1:dest1 ，source2:dest2 。例如：

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

從快照還原至不同的命名空間

您可以使用自訂資源 (CR) 檔案從快照還原數據，還原到不同的命名空間或原始來源命名空間。當您使用 SnapshotRestore CR 將快照還原到不同的命名空間時， Trident Protect 會在新的命名空間中還原應用程序，並為還原的應用程式建立應用程式 CR。為了保護已還原的應用程序，可以建立按需備份或快照，或製定保護計劃。

使用 **CR**

步驟

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-snapshot-restore-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
   - * metadata.name*: （_required）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - **spec.appVaultRef** ：（_required _）儲存快照內容的 AppVault 名稱。
   - **spec.appArchivePath** ：在 AppVault 中儲存快照內容的路徑。您可以使用下列命令來尋找此路徑：

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

   - **spec.namespaceMapping**: 將還原作業的來源命名空間對應至目的地命名空間。以環境中的資訊取代 `my-source-namespace`和 `my-destination-namespace`。
   - **spec.storageClassMapping** ：將還原作業的來源儲存類別對應至目的地儲存類別。以環境中的資訊取代 `destinationStorageClass`和 `sourceStorageClass`。

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

3. （*Optional*）如果您只需要選取應用程式的某些資源來還原，請新增篩選功能，以包含或排除標記有特定標籤的資源：
   - **resourceFilter.resourceSelectionCriteria** ：（篩選所需）使用 `Include`或包含或 `Exclude`排除在 resourceMatchers 中定義的資源。新增下列資源配置工具參數、以定義要納入或排除的資源：
     - **resourceFilter.resourceMatchers** ：一組 resourceMatcher 物件。如果您在此陣列中定義多個元素，它們會比對為 OR 作業，而每個元素（群組，種類，版本）內的欄位會比對為 AND 作業。
       - **resourceMatchers[].group** ：（*Optional*）要篩選的資源群組。

- **resourceMatchers[].cher** ：（ *Optional* ）要篩選的資源種類。
- **resourceMatchers[].version** ：（ *Optional* ）要篩選的資源版本。
- 要篩選之資源的 Kubernetes metadata.name 欄位中的 * resourceMatchers[].names* ：（ *Optional* ）名稱。
- 要篩選之資源的 Kubernetes metadata.name 欄位中的 * resourceMatchers[].names* ：（ *Optional* ）命名空間。
- 資源的 Kubernetes metadata.name 欄位中的 *resourceMatchers[].labelSelectors * ：（ *Optional* ） Label 選取器字串，如中所定義 "Kubernetes文件"。例如 `"trident.netapp.io/os=linux"`：。

  例如：

```
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. 在您以正確的值填入檔案之後 `trident-protect-snapshot-restore-cr.yaml` 、請套用 CR ：

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

使用**CLI**

步驟

1. 將快照還原至不同的命名空間，以環境中的資訊取代方括號中的值。
   - `snapshot`引數使用格式的命名空間和快照名稱 `<namespace>/<name>`。
   - 此 `namespace-mapping`引數使用以冒號分隔的命名空間，以格式將來源命名空間對應至正確的目的地命名空間 `source1:dest1,source2:dest2`。

     例如：

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

從快照還原至原始命名空間

您可以隨時將快照還原至原始命名空間。

> 如果您的應用程式使用多個命名空間，且這些命名空間中存在同名的 PVC，則快照還原作業（無論是就地還原或還原到新命名空間）都將無法正常運作。所有還原的磁碟區將包含相同的資料，而不是每個命名空間對應的正確資料。請使用備份還原代替快照還原，或升級到 26.02 或更高版本以修復此問題。

使用 **CR**

步驟

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-snapshot-ipr-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
   - * metadata.name*: （_required）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - **spec.appVaultRef** ：（_required _）儲存快照內容的 AppVault 名稱。
   - **spec.appArchivePath** ：在 AppVault 中儲存快照內容的路徑。您可以使用下列命令來尋找此路徑：

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

```
---
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appVaultRef: appvault-name
    appArchivePath: my-snapshot-path
```

3. （*Optional*）如果您只需要選取應用程式的某些資源來還原，請新增篩選功能，以包含或排除標記有特定標籤的資源：
   - **resourceFilter.resourceSelectionCriteria** ：（篩選所需）使用 `Include`或包含或 `Exclude`排除在 resourceMatchers 中定義的資源。新增下列資源配置工具參數、以定義要納入或排除的資源：
     - **resourceFilter.resourceMatchers** ：一組 resourceMatcher 物件。如果您在此陣列中定義多個元素，它們會比對為 OR 作業，而每個元素（群組，種類，版本）內的欄位會比對為 AND 作業。
       - **resourceMatchers[].group** ：（*Optional*）要篩選的資源群組。
       - **resourceMatchers[].cher** ：（*Optional*）要篩選的資源種類。
       - **resourceMatchers[].version** ：（*Optional*）要篩選的資源版本。
       - 要篩選之資源的 Kubernetes metadata.name 欄位中的 * resourceMatchers[].names* ：（*Optional*）名稱。
       - 要篩選之資源的 Kubernetes metadata.name 欄位中的 * resourceMatchers[].names* ：（*Optional*）命名空間。
       - 資源的 Kubernetes metadata.name 欄位中的 *resourceMatchers[].labelSelectors * ：（*Optional*） Label 選取器字串，如中所定義 "Kubernetes文件"。例如 `"trident.netapp.io/os=linux"`：。

例如：

```
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. 在您以正確的值填入檔案之後 `trident-protect-snapshot-ipr-cr.yaml` 、請套用 CR ：

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

使用CLI

步驟

1. 將快照還原至原始命名空間，以環境中的資訊取代方括號中的值。例如：

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

檢查還原作業的狀態

您可以使用命令列來檢查進行中，已完成或已失敗的還原作業狀態。

步驟

1. 使用下列命令可擷取還原作業的狀態，以環境中的資訊取代方括號中的值：

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

使用NetApp SnapMirror和Trident Protect 複製應用程式

使用Trident Protect，您可以利用NetApp SnapMirror技術的非同步複製功能，將資料和應用程式變更從一個儲存後端複製到另一個儲存後端，無論是在同一叢集內還是在不同叢集之間。

還原和容錯移轉作業期間的命名空間註釋和標籤

在還原和容錯移轉作業期間，目的地命名空間中的標籤和註釋會與來源命名空間中的標籤和註釋相符。會新增來源命名空間中不存在的標籤或註釋，並覆寫已存在的任何標籤或註釋，以符合來源命名空間的值。只存在於目的地命名空間上的標籤或註釋會保持不變。

> 如果您使用 RedHat OpenShift ，請務必注意 OpenShift 環境中命名空間註釋的關鍵作用。命名空間註釋可確保還原的 Pod 符合 OpenShift 安全性內容限制（ SCC ）所定義的適當權限和安全性組態，而且無需權限問題即可存取磁碟區。如需詳細資訊，請 "OpenShift 安全性內容限制文件"參閱。

您可以在執行還原或容錯移轉作業之前，先設定 Kubernetes 環境變數，以避免覆寫目的地命名空間中的特定註釋 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`。例如：

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

如果您使用 Helm 安裝了來源應用程序， `--create-namespace`國旗，給予特殊待遇 `name`標籤鍵。在復原或故障轉移過程中， Trident Protect 會將此標籤複製到目標命名空間，但如果來源命名空間的值與來源命名空間的值匹配，則會將值更新為目標命名空間的值。如果此值與來源命名空間不匹配，則會將其複製到目標命名空間，而不做任何變更。

範例

以下範例提供來源和目的地命名空間，每個命名空間都有不同的註釋和標籤。您可以查看作業前後目的地命名空間的狀態，以及註釋和標籤在目的地命名空間中的組合或覆寫方式。

還原或容錯移轉作業之前

下表說明還原或容錯移轉作業之前的範例來源和目的地命名空間狀態：

命名空間	註釋	標籤
命名空間 nS-1 （來源）	• annotation.one / 機碼：「 updatedvalue 」 • annotation.b2/key ：「 true 」	• 環境 = 正式作業 • Compliance = HIPAA • NAME=nS-1

命名空間	註釋	標籤
命名空間 nS-2 （目的地）	• annotation.one / 機碼：「 true 」 • annotation.the/key ：「 FALSE 」	• role = 資料庫

還原作業之後

下表說明還原或容錯移轉作業之後範例目的地命名空間的狀態。某些金鑰已新增，部分已覆寫， `name`標籤已更新以符合目的地命名空間：

命名空間	註釋	標籤
命名空間 nS-2 （目的地）	• annotation.one / 機碼：「 updatedvalue 」 • annotation.b2/key ：「 true 」 • annotation.the/key ：「 FALSE 」	• NAME=nS-2 • Compliance = HIPAA • 環境 = 正式作業 • role = 資料庫

您可以設定Trident Protect 在資料保護作業期間凍結和解凍檔案系統。"了解更多關於使用Trident Protect 設定檔系統凍結的信息"。

設定複寫關係

設定複寫關係涉及下列事項：

- 選擇Trident Protect 拍攝應用程式快照的頻率（包括應用程式的 Kubernetes 資源以及應用程式每個磁碟區的磁碟區快照）。
- 選擇複寫排程（包括 Kubernetes 資源及持續磁碟區資料）
- 設定拍攝快照的時間

步驟

1. 為來源叢集上的來源應用程式建立 AppVault 。視您的儲存供應商而定，請修改中的範例"AppVault 自訂資源"以符合您的環境：

使用 **CR** 建立 **AppVault**

a. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-appvault-primary-source.yaml`）。

b. 設定下列屬性：

- * metadata.name*: （_required） AppVault 自訂資源的名稱。請記下您選擇的名稱，因為複寫關係所需的其他 CR 檔案會參照此值。
- * spec.providerConfig*: （_required）儲存使用指定供應商存取 AppVault 所需的組態。請為您的供應商選擇一個「鎖釦名稱」和任何其他必要的詳細資料。請記下您選擇的值，因為複寫關係所需的其他 CR 檔案會參照這些值。如需 AppVault CRS 與其他供應商的範例，請參閱"AppVault 自訂資源"。
- * spec.providerCredentials*: （_required _）會儲存使用指定提供者存取 AppVault 所需之任何認證的參考資料。
  - * spec.providerCredentials.valueFromSecret*: （_required _）表示認證值應來自機密。
    - **key** ：（_required _）要從中選擇的密碼的有效金鑰。
    - * 名稱 * ：（_ 必要 _）包含此欄位值的機密名稱。必須位於相同的命名空間中。
  - * spec.providerCredentials.secretAccessKey*: （_required _）存取提供者所用的存取金鑰。* 名稱 * 應與 * spec.providerCredentials.valueFromSecret.name* 相符。
- * spec.providerType*: （_required _）決定提供備份的內容，例如 NetApp ONTAP S3 ，一般 S3 ， Google Cloud 或 Microsoft Azure 。可能值：
  - AWS
  - Azure
  - GCP
  - generic-S3
  - ONTAP S3
  - StorageGRID S3

c. 在您以正確的值填入檔案之後 `trident-protect-appvault-primary-source.yaml` 、請套用 CR ：

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

使用 **CLI** 建立 **AppVault**

a. 建立 AppVault ，以環境資訊取代方括號中的值：

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

2. 建立來源應用程式 CR ：

使用 **CR** 建立來源應用程式

a. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-app-source.yaml`）。

b. 設定下列屬性：

- * metadata.name*: （_required）應用程式自訂資源的名稱。請記下您選擇的名稱，因為複寫關係所需的其他 CR 檔案會參照此值。
- * spec.includedNamespaces*: （_required）一組命名空間和相關標籤。使用命名空間名稱，並選擇性地使用標籤來縮小命名空間的範圍，以指定此處列出的命名空間中存在的資源。應用程式命名空間必須是此陣列的一部分。
  - YAML* 範例：

```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. 在您以正確的值填入檔案之後 `trident-protect-app-source.yaml` 、請套用 CR ：

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

使用 **CLI** 建立來源應用程式

a. 建立來源應用程式。例如：

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

3. 您也可以選擇拍攝來源應用程式的快照。此快照是作為目的地叢集上應用程式的基礎。如果您略過此步驟，則需要等待下一個排定的快照執行，以便擁有最近的快照。

使用 **CR** 拍攝快照

a. 建立來源應用程式的複寫排程：

   i. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-schedule.yaml`）。

   ii. 設定下列屬性：

      - * metadata.name*: （_required）排程自訂資源的名稱。
      - **spec.AppVaultRef** ：（_required _）此值必須符合來源應用程式的 AppVault metadata.name 欄位。
      - **spec.ApplicationRef** ：（_required _）此值必須符合來源應用程式 CR 的 metadata.name 欄位。
      - *spec.backupRetention * ：（_required _）此欄位為必填欄位，且值必須設為 0 。
      - **spec.enabled** ：必須設置爲 true 。
      - * spec.granularity*: 必須設定為 `Custom`。
      - **spec.recurrenceRule** ：以 UTC 時間和循環時間間隔定義開始日期。
      - *spec.snapshotRetention * ：必須設定為 2 。

      YAML 範例：

```yaml
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule-0e1f88ab-f013-4bce-8ae9-6afed9df59a1
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

   i. 在您以正確的值填入檔案之後 `trident-protect-schedule.yaml` 、請套用 CR ：

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

使用 **CLI** 拍攝快照

a. 建立快照，以您環境的資訊取代方括號中的值。例如：

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

4. 在目的地叢集上建立與您在來源叢集上套用的 AppVault CR 相同的來源應用程式 AppVault CR ，並命名該應用程式（例如 `trident-protect-appvault-primary-destination.yaml`）。

5. 套用 CR ：

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

6. 為目的地叢集上的目的地應用程式建立 AppVault 。視您的儲存供應商而定，請修改中的範例"AppVault 自訂資源"以符合您的環境：

   a. 建立自訂資源（ CR ）檔案並命名（例如 `trident-protect-appvault-secondary-destination.yaml`）。

   b. 設定下列屬性：

      - * metadata.name*: （ _required ） AppVault 自訂資源的名稱。請記下您選擇的名稱，因為複寫關係所需的其他 CR 檔案會參照此值。
      - * spec.providerConfig*: （ _required ）儲存使用指定供應商存取 AppVault 所需的組態。請為您的供應商選擇 `bucketName`和任何其他必要詳細資料。請記下您選擇的值，因為複寫關係所需的其他 CR 檔案會參照這些值。如需 AppVault CRS 與其他供應商的範例，請參閱"AppVault 自訂資源"。
      - * spec.providerCredentials*: （ _required _ ）會儲存使用指定提供者存取 AppVault 所需之任何認證的參考資料。
        - * spec.providerCredentials.valueFromSecret*: （ _required _ ）表示認證值應來自機密。
          - **key** ：（ _required _ ）要從中選擇的密碼的有效金鑰。
          - * 名稱 * ：（ _ 必要 _ ）包含此欄位值的機密名稱。必須位於相同的命名空間中。
        - * spec.providerCredentials.secretAccessKey*: （ _required _ ）存取提供者所用的存取金鑰。* 名稱 * 應與 * spec.providerCredentials.valueFromSecret.name* 相符。
      - * spec.providerType*: （ _required _ ）決定提供備份的內容，例如 NetApp ONTAP S3 ，一般 S3 ， Google Cloud 或 Microsoft Azure 。可能值：
        - AWS
        - Azure
        - GCP
        - generic-S3
        - ONTAP S3

    - StorageGRID S3

c. 在您以正確的值填入檔案之後 `trident-protect-appvault-secondary-destination.yaml` 、請套用 CR ：

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

7. 建立 AppMirrorRelationship CR 檔案：

使用 **CR** 建立 **AppMirrorRelationship**

a. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-relationship.yaml`）。

b. 設定下列屬性：

- * metadata.name:* （必要） AppMirrorRelationship 自訂資源的名稱。
- * spec.destinationAppVaultRef*: （_required_）此值必須符合目的地叢集上目的地應用程式的 AppVault 名稱。
- * spec.namespaceMapping*: （_required_）目的地和來源命名空間必須符合各自應用程式 CR 中定義的應用程式命名空間。
- **spec.sourceAppVaultRef** ：（_required_）此值必須符合來源應用程式的 AppVault 名稱。
- **spec.sourceApplicationName** ：（_required_）此值必須符合您在來源應用程式 CR 中定義的來源應用程式名稱。
- **spec.storageClassName** ：（_required_）選擇叢集上有效儲存類別的名稱。儲存類別必須連結至與來源環境對等的 ONTAP 儲存 VM 。
- **spec.recurrenceRule** ：以 UTC 時間和循環時間間隔定義開始日期。

  YAML 範例：

```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. 在您以正確的值填入檔案之後 `trident-protect-relationship.yaml` 、請套用 CR ：

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

使用 **CLI** 建立 **AppMirrorRelationship**

a. 建立並套用 AppMirrorRelationship 物件，以環境資訊取代方括號中的值。例如：

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

8. （ *Optional* ）檢查複寫關係的狀態和狀態：

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

容錯移轉至目的地叢集

使用Trident Protect，您可以將複製的應用程式故障轉移到目標叢集。此過程會停止複製關係，並將應用程式在目標叢集上連線。如果來源叢集上的應用程式正在運行， Trident Protect 不會停止該應用程式。

步驟

1. 開啟 AppMirrorRelationship CR 檔案（例如 `trident-protect-relationship.yaml`），並將 * spec.desiredState* 的值變更為 `Promoted`。
2. 儲存 CR 檔案。
3. 套用 CR ：

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

4. （ *Optional* ）在容錯移轉應用程式上建立所需的任何保護排程。
5. （ *Optional* ）檢查複寫關係的狀態和狀態：

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

重新同步容錯移轉複寫關係

重新同步作業會重新建立複寫關係。執行重新同步作業後，原始來源應用程式即成為執行中的應用程式，而對目的地叢集上執行中的應用程式所做的任何變更都會被捨棄。

此程序會在重新建立複寫之前，停止目的地叢集上的應用程式。

> 在容錯移轉期間寫入目的地應用程式的任何資料都會遺失。

步驟

1. 建立來源應用程式的快照。
2. 打開 AppMirrorRelationship CR 文件（例如 `trident-protect-relationship.yaml`），然後將 spec.desiredState 的值更改爲 `Established`。
3. 儲存 CR 檔案。
4. 套用 CR ：

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

5. 如果您在目的地叢集上建立任何保護排程來保護容錯移轉應用程式，請將其移除。任何仍會導致磁碟區快照失敗的排程。

反轉重新同步容錯移轉複寫關係

當您反向重新同步容錯移轉複寫關係時，目的地應用程式會變成來源應用程式，來源會變成目的地。在容錯移轉期間對目的地應用程式所做的變更會保留下來。

步驟

1. 刪除原始目的地叢集上的 AppMirrorRelationship CR 。這會導致目的地成為來源。如果新的目的地叢集上還有任何保護排程，請將其移除。
2. 套用原先用來設定與相對叢集關係的 CR 檔案，以設定複寫關係。
3. 確保每個叢集上的 AppVault CRS 均已就緒。
4. 在相對的叢集上設定複寫關係，設定反轉方向的值。

反轉應用程式複寫方向

當您反轉複製方向時， Trident Protect 會將應用程式移至目標儲存後端，同時繼續複製回原始來源儲存後端。Trident Protect 會停止來源應用程式並將資料複製到目標位置，然後再故障轉移到目標應用程式。

在這種情況下、您要交換來源和目的地。

步驟

1. 建立關機快照：

使用 **CR** 建立關機快照

a. 停用來源應用程式的保護原則排程。

b. 建立 ShutdownSnapshot CR 檔案：

   i. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-shutdownsnapshot.yaml`）。

   ii. 設定下列屬性：

      - * metadata.name*: （_required）自訂資源的名稱。
      - **spec.AppVaultRef**：（_required _）此值必須符合來源應用程式的 AppVault metadata.name 欄位。
      - **spec.ApplicationRef**：（_required _）此值必須符合來源應用程式 CR 檔案的 metadata.name 欄位。

        YAML 範例：

```
---
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. 在您以正確的值填入檔案之後 `trident-protect-shutdownsnapshot.yaml`、請套用 CR：

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

使用 **CLI** 建立關機快照

a. 建立關機快照，以環境資訊取代方括號中的值。例如：

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

2. 快照完成後，取得快照的狀態：

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

3. 使用下列命令尋找 * shutdownsnapshot .status.appArchivePath* 的值，並記錄檔案路徑的最後一部分（也稱為 basename ；這將是最後一個斜線之後的所有項目）：

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

4. 執行容錯移轉，從目的地叢集移轉至來源叢集，並進行下列變更：

> 在容錯移轉程序的步驟 2 中，將欄位包含在 `spec.promotedSnapshot` AppMirrorRelationship CR 檔案中，並將其值設為您在上述步驟 3 中記錄的基礎名稱。

5. 執行中的反向重新同步步驟[反轉重新同步容錯移轉複寫關係]。
6. 在新的來源叢集上啟用保護排程。

結果

由於反向複寫，因此會發生下列動作：

- 原始來源應用程式的 Kubernetes 資源會擷取快照。
- 刪除應用程式的Kubernetes資源（保留PVCS和PVs）、即可順利停止原始來源應用程式的Pod。
- 當 Pod 關機之後、應用程式的磁碟區快照就會被擷取和複寫。
- SnapMirror關係中斷、使目的地磁碟區準備好進行讀寫。
- 應用程式的 Kubernetes 資源會從關機前快照還原、並使用原始來源應用程式關機後複寫的 Volume 資料。
- 複寫會以相反方向重新建立。

將應用程式容錯移轉至原始來源叢集

使用Trident Protect，您可以透過以下步驟序列在故障轉移作業後實現「故障復原」。在此恢復原始複製方向的工作流程中， Trident Protect 會將任何應用程式變更複製（重新同步）回原始來源應用程序，然後再反轉複製方向。

此程序從已完成容錯移轉至目的地的關係開始、並涉及下列步驟：

- 從容錯移轉狀態開始。
- 反向重新同步複寫關係。

> 請勿執行正常的重新同步作業，因為這會捨棄在容錯移轉程序期間寫入目的地叢集的資料。

- 反轉複寫方向。

步驟

1. 執行[反轉重新同步容錯移轉複寫關係]步驟。
2. 執行[反轉應用程式複寫方向]步驟。

刪除複寫關係

您可以隨時刪除複寫關係。當您刪除應用程式複寫關係時，會產生兩個獨立的應用程式，兩者之間沒有任何關係。

步驟

1. 刪除 AppMirrorRelationship CR ：

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

使用Trident Protect 遷移應用程式

您可以將備份或快照資料還原至不同的叢集或儲存類別，在叢集或儲存類別之間移轉應用程式。

> 當您移轉應用程式時，為應用程式設定的所有執行掛鉤都會隨應用程式一起移轉。如果存在還原後執行掛鉤，則會在還原作業中自動執行。

備份與還原作業

若要針對下列案例執行備份與還原作業，您可以自動化特定的備份與還原工作。

複製到同一個叢集

若要將應用程式複製到同一個叢集，請建立快照或備份，然後將資料還原到同一個叢集。

步驟

1. 執行下列其中一項：
   a. "建立快照"。
   b. "建立備份"。
2. 在同一個叢集上，視您建立的是快照或備份而定，請執行下列其中一項：
   a. "從快照還原資料"。
   b. "從備份還原資料"。

複製到不同叢集

若要將應用程式複製到不同的叢集（執行跨叢集克隆），請在來源叢集上建立備份，然後將備份還原到不同的叢集。請確保目標叢集上已安裝Trident Protect。

> 您可以使用在不同叢集之間複寫應用程式"SnapMirror 複寫"。

步驟

1. "建立備份"。
2. 請確定已在目的地叢集上設定包含備份之物件儲存貯體的 AppVault CR 。
3. 在目的地叢集上"從備份還原資料"，。

將應用程式從一個儲存類別移轉至另一個儲存類別

您可以將快照還原至不同的目的地儲存類別，將應用程式從一個儲存類別移轉至不同的儲存類別。

例如（從還原 CR 中排除機密）：

```
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
```

使用 **CR** 還原快照

步驟

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-snapshot-restore-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
   - * metadata.name*: （_required）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - **spec.appArchivePath** ：在 AppVault 中儲存快照內容的路徑。您可以使用下列命令來尋找此路徑：

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

   - **spec.appVaultRef** ：（_required _）儲存快照內容的 AppVault 名稱。
   - **spec.namespaceMapping**: 將還原作業的來源命名空間對應至目的地命名空間。以環境中的資訊取代 `my-source-namespace`和 `my-destination-namespace`。

     ```
     ---
     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. 或者，如果您只需要選取要還原的應用程式特定資源，請新增篩選功能，以包含或排除標記有特定標籤的資源：
   - *resourceFilter.resourceSelectionCriteria ：（篩選所需）用於 `include or exclude`包含或排除在 resourceMatchers 中定義的資源。新增下列資源配置工具參數、以定義要納入或排除的資源：
     - **resourceFilter.resourceMatchers** ：一組 resourceMatcher 物件。如果您在此陣列中定義多個元素，它們會比對為 OR 作業，而每個元素（群組，種類，版本）內的欄位會比對為 AND 作業。
       - **resourceMatchers[].group** ：（*Optional*）要篩選的資源群組。
       - **resourceMatchers[].cher** ：（*Optional*）要篩選的資源種類。
       - **resourceMatchers[].version** ：（*Optional*）要篩選的資源版本。
       - 要篩選之資源的 Kubernetes metadata.name 欄位中的 * resourceMatchers[].names* ：（*Optional*）名稱。
       - 要篩選之資源的 Kubernetes metadata.name 欄位中的 * resourceMatchers[].names* ：（

Optional）命名空間。

- 資源的 Kubernetes metadata.name 欄位中的 *resourceMatchers[].labelSelectors *：（*Optional*） Label 選取器字串，如中所定義 "Kubernetes文件"。例如 `"trident.netapp.io/os=linux"`：。

  例如：

```
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. 在您以正確的值填入檔案之後 `trident-protect-snapshot-restore-cr.yaml`、請套用 CR：

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

使用 **CLI** 還原快照

步驟

1. 將快照還原至不同的命名空間，以環境中的資訊取代方括號中的值。

   - `snapshot`引數使用格式的命名空間和快照名稱 `<namespace>/<name>`。
   - 此 `namespace-mapping`引數使用以冒號分隔的命名空間，以格式將來源命名空間對應至正確的目的地命名空間 `source1:dest1,source2:dest2`。

     例如：

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

管理Trident Protect 執行鉤子

執行攔截是一種自訂動作、可設定搭配託管應用程式的資料保護作業一起執行。例如、如果您有資料庫應用程式、您可以使用執行掛勾來暫停快照之前的所有資料庫交易、並在快照完成後繼續交易。如此可確保應用程式一致的快照。

執行掛勾的類型

Trident Protect 支援以下幾種執行鉤子類型，取決於它們的運行時機：

- 快照前
- 快照後
- 預先備份
- 備份後
- 還原後
- 容錯移轉後

執行順序

執行資料保護作業時、執行掛機事件會依照下列順序發生：

1. 任何適用的自訂操作前執行掛勾都會在適當的容器上執行。您可以視需要建立及執行任意數量的自訂操作前掛勾、但在作業之前執行這些掛勾的順序既不保證也無法設定。
2. 如果適用，則會發生檔案系統凍結。"了解更多關於使用Trident Protect 設定檔系統凍結的信息"。
3. 執行資料保護作業。
4. 凍結的檔案系統會在適用的情況下解除凍結。
5. 任何適用的自訂操作後執行掛勾都會在適當的容器上執行。您可以視需要建立及執行任意數量的自訂後置作業掛勾、但在作業後執行這些掛勾的順序並不保證也無法設定。

如果您建立同一類型的多個執行掛勾（例如預先快照）、則無法保證這些掛勾的執行順序。不過、不同類型的掛勾的執行順序也有保證。例如，以下是具有所有不同類型勾點的組態執行順序：

1. 執行快照前掛勾
2. 快照後掛勾已執行
3. 執行備份前掛勾
4. 執行備份後掛勾

上述順序範例僅適用於執行不使用現有快照的備份時。

在正式作業環境中啟用執行攔截指令碼之前、請務必先進行測試。您可以使用'kubecl exec'命令來方便地測試指令碼。在正式作業環境中啟用執行掛勾之後、請測試所產生的快照和備份、以確保它們一致。您可以將應用程式複製到暫用命名空間、還原快照或備份、然後測試應用程式、藉此完成此作業。

關於自訂執行掛勾的重要注意事項

規劃應用程式的執行掛勾時、請考量下列事項。

- 執行攔截必須使用指令碼來執行動作。許多執行掛勾可以參照相同的指令碼。
- Trident Protect 要求執行鉤子使用的腳本以可執行 shell 腳本的格式編寫。
- 指令碼大小上限為96KB。
- Trident Protect 使用執行鉤子設定和任何符合條件來決定哪些鉤子適用於快照、備份或還原作業。

> 由於執行掛勾通常會減少或完全停用執行中應用程式的功能、因此您應該一律盡量縮短自訂執行掛勾執行所需的時間。如果您以相關的執行掛勾開始備份或快照作業、但隨後取消它、則如果備份或快照作業已經開始、仍允許掛勾執行。這表示備份後執行掛勾中使用的邏輯無法假設備份已完成。

執行攔截篩選器

當您新增或編輯應用程式的執行掛鉤時，您可以將篩選器新增至執行掛鉤，以管理掛鉤將符合的容器。篩選器對於在所有容器上使用相同容器映像的應用程式來說非常實用、但可能會將每個映像用於不同的用途（例如Elasticsearch）。篩選器可讓您建立執行攔截器在某些容器上執行的案例、但不一定所有容器都相同。如果您為單一執行掛勾建立多個篩選器、這些篩選器會與邏輯和運算子結合使用。每個執行掛機最多可有10個作用中篩選器。

新增到執行掛鉤的每個過濾器都使用正規表示式來匹配叢集中的容器。當鉤子與容器匹配時，鉤子將在該容器上運行其關聯的腳本。過濾器的正規表示式使用正規表示式 2 (RE2) 語法，該語法不支援建立從符合清單中排除容器的過濾器。有關Trident Protect 在執行鉤子過濾器中支援的正規表示式語法的詳細信息，請參閱 "規則運算式2（RE2）語法支援"。

> 如果您將命名空間篩選器新增至執行掛勾、而執行還原或複製作業之後執行、且還原或複製來源與目的地位於不同的命名空間、則命名空間篩選器只會套用至目的地命名空間。

執行攔截範例

請造訪 "NetApp Verda GitHub專案" 下載熱門應用程式（例如 Apache Cassandra 和 Elasticsearch ）的實際執行連結。您也可以查看範例、瞭解如何建構您自己的自訂執行掛勾。

建立執行掛鉤

您可以使用Trident Protect 為應用程式建立自訂執行鉤子。您需要擁有所有者、管理員或成員權限才能建立執行鉤子。

使用 **CR**

步驟

1. 建立自訂資源（ CR ）檔案並命名為 `trident-protect-hook.yaml`。
2. 設定以下屬性以符合您的Trident Protect 環境和叢集設定：
   - * metadata.name*: （ _required ）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - **SPEC.applicationRef** ：（ _required _ ）要執行執行攔截的應用程式 Kubernetes 名稱。
   - *spec.Stage * ：（ _required _ ）一個字串，指出執行掛鉤應在動作期間執行的階段。可能值：
     - 準備
     - 貼文
   - **spec.ACTION** ：（ _required _ ）字串，表示執行攔截將採取的行動，前提是指定的任何執行攔截篩選條件都已相符。可能值：
     - Snapshot
     - 備份
     - 還原
     - 容錯移轉
   - **spec.enabled** ：（ *Optional* ）表示此執行掛鉤是否已啟用或停用。如果未指定，則預設值為 true 。
   - **spec.hookSource** ：（ _required _ ）包含 base64 編碼 hook 指令碼的字串。
   - **spec.timeout** ：（ *Optional* ）一個數字，定義允許執行掛鉤執行的時間（以分鐘為單位）。最小值為 1 分鐘，如果未指定，預設值為 25 分鐘。
   - **spec.arguments** ：（ *Optional* ） YAML 引數清單，您可以為執行攔截器指定。
   - *spec.mmatchingCriteria ：（ *Optional* ）選擇性的條件金鑰值配對清單，每個配對組成執行掛機篩選器。每個執行掛鉤最多可新增 10 個篩選器。
   - **spec.matchingCriteria.type** ：（ *Optional* ）識別執行掛鉤篩選器類型的字串。可能值：
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value** ：（ *Optional* ）識別執行掛鉤篩選值的字串或規則運算式。

     YAML 範例：

```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. 在您以正確的值填入 CR 檔案之後，請套用 CR ：

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

使用**CLI**

步驟

1. 建立執行掛鉤，以環境資訊取代方括號中的值。例如：

```
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>
```

版權資訊

Copyright © 2026 NetApp, Inc. 版權所有。台灣印製。非經版權所有人事先書面同意，不得將本受版權保護文件的任何部分以任何形式或任何方法（圖形、電子或機械）重製，包括影印、錄影、錄音或儲存至電子檢索系統中。

由 NetApp 版權資料衍伸之軟體必須遵守下列授權和免責聲明：

此軟體以 NETAPP「原樣」提供，不含任何明示或暗示的擔保，包括但不限於有關適售性或特定目的適用性之擔保，特此聲明。於任何情況下，就任何已造成或基於任何理論上責任之直接性、間接性、附隨性、特殊性、懲罰性或衍生性損害（包括但不限於替代商品或服務之採購；使用、資料或利潤上的損失；或企業營運中斷），無論是在使用此軟體時以任何方式所產生的契約、嚴格責任或侵權行為（包括疏忽或其他）等方面，NetApp 概不負責，即使已被告知有前述損害存在之可能性亦然。

NetApp 保留隨時變更本文所述之任何產品的權利，恕不另行通知。NetApp 不承擔因使用本文所述之產品而產生的責任或義務，除非明確經過 NetApp 書面同意。使用或購買此產品並不會在依據任何專利權、商標權或任何其他 NetApp 智慧財產權的情況下轉讓授權。

本手冊所述之產品受到一項（含）以上的美國專利、國外專利或申請中專利所保障。

有限權利說明：政府機關的使用、複製或公開揭露須受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中的「技術資料權利 - 非商業項目」條款 (b)(3) 小段所述之限制。

此處所含屬於商業產品和 / 或商業服務（如 FAR 2.101 所定義）的資料均為 NetApp, Inc. 所有。根據本協議提供的所有 NetApp 技術資料和電腦軟體皆屬於商業性質，並且完全由私人出資開發。 美國政府對於該資料具有非專屬、非轉讓、非轉授權、全球性、有限且不可撤銷的使用權限，僅限於美國政府為傳輸此資料所訂合約所允許之範圍，並基於履行該合約之目的方可使用。除非本文另有規定，否則未經 NetApp Inc. 事前書面許可，不得逕行使用、揭露、重製、修改、履行或展示該資料。美國政府授予國防部之許可權利，僅適用於 DFARS 條款 252.227-7015(b)（2014 年 2 月）所述權利。

商標資訊

NETAPP、NETAPP 標誌及 http://www.netapp.com/TM 所列之標章均為 NetApp, Inc. 的商標。文中所涉及的所有其他公司或產品名稱，均為其各自所有者的商標，不得侵犯。