



還原應用程式 Trident

NetApp
October 30, 2026

目錄

還原應用程式	1
使用 Trident Protect 恢復應用程式	1
從備份還原到不同的命名空間	1
從備份還原到原始命名空間	4
從備份還原到不同的叢集	6
從快照還原到不同的命名空間	8
從快照還原到原始命名空間	11
檢查還原作業的狀態	13
使用進階 Trident Protect 還原設定	13
還原和容錯移轉作業期間的命名空間註釋和標籤	13
支援的欄位	15
支援的註釋	15
使用自訂資源還原資源時修改資源	16
資源修改的運作方式	16
支援的資源	17
支援的作業	17

還原應用程式

使用 Trident Protect 恢復應用程式

您可以使用 Trident Protect 從快照或備份還原應用程式。如果要將應用程式還原到同一個叢集，從現有快照還原速度會更快。



- 還原應用程式時，所有為該應用程式配置的執行鉤子都會隨應用程式一起還原。如果存在復原後執行鉤子，它會在復原操作過程中自動執行。
- qtree 磁碟區支援從備份還原到不同的命名空間或原始命名空間。但是，qtree 磁碟區不支援從快照還原到不同的命名空間或原始命名空間。
- 您可以使用進階設定來自訂還原操作。若要深入瞭解，請參閱 ["使用進階 Trident Protect 還原設定"](#)。
- 若要從歸檔中復原特定的 KubeVirt 虛擬機器 (VM) 而不還原所有命名空間資源，請使用復原 CR 上的 `includedVirtualMachines` 欄位。有關支援的 CR、欄位參考和 YAML 範例，請參閱 ["使用 Trident Protect 保護 KubeVirt 虛擬機器"](#)。

從備份還原到不同的命名空間

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式，並為還原後的應用程式建立一個應用程式 CR。若要保護還原的應用程式，您可以建立隨需備份或快照，或建立保護排程。



- 將備份還原到具有現有資源的不同命名空間不會變更與備份中資源同名的任何資源。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，或將備份還原到新的命名空間。
- 使用 CR 還原到新命名空間時，必須先手動建立目標命名空間，然後再套用 CR。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

開始之前

請確保 AWS 工作階段權杖的有效期限足以應付任何長時間執行的 s3 還原作業。如果權杖在還原作業期間過期、作業可能會失敗。

- 有關檢查當前會話令牌過期時間的更多資訊，請參閱 ["AWS API 說明文件"](#)。
- 如需 AWS 資源憑證的詳細資訊，請參閱 ["AWS IAM 說明文件"](#)。



當您使用 Kopia 作為資料移動工具還原備份時，您可以選擇在 CR 中或使用 CLI 指定註釋，以控制 Kopia 使用的暫存的行為。有關可配置選項的更多資訊，請參閱 ["Kopia 說明文件"](#)。使用 `tridentctl-protect create --help` 命令以取得有關使用 Trident Protect CLI 指定註釋的更多資訊。

使用 CR

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-backup-restore-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：

- **metadata.name**：（必填）此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
- **spec.appArchivePath**：AppVault 內儲存備份內容的路徑。您可以使用以下命令來尋找此路徑：

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

- **spec.appVaultRef**：（必填）儲存備份內容的 AppVault 名稱。
- **spec.destinationApplicationName**：（選用）還原應用程式的名稱。如果提供、還原的應用程式會使用此名稱。如果未提供、還原的應用程式會使用來源應用程式名稱。
- **spec.namespaceMapping**：復原作業的來源命名空間到目標命名空間的對應。請將 `my-source-namespace` 和 `my-destination-namespace` 替換為您環境中的資訊。

```
---
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. （可選）如果您只需要選擇應用程式的某些資源進行恢復，請新增篩選條件，以包含或排除帶有特定標籤的資源：



Trident Protect 會自動選擇一些資源，因為它們與您選擇的資源有關聯。例如，如果您選擇了持續性磁碟區宣告資源，且該資源關聯了一個 Pod，則 Trident Protect 也會還原該關聯 Pod。

- **resourceFilter.resourceSelectionCriteria**：（篩選必需）使用 `Include` 或 `Exclude` 來包含或排除在 resourceMatchers 中定義的資源。新增以下 resourceMatchers 參數以定義要包含或排除的資源：
 - **resourceFilter.resourceMatchers**：resourceMatcher 物件的陣列。如果在此陣列中定義多個元素，它們之間將按 OR 運算匹配，每個元素內的欄位（group、kind、version）之間將按 AND 運算匹配。
 - **resourceMatchers[].group**：（可選）要篩選的資源群組。
 - **resourceMatchers[].kind**：（可選）要篩選的資源類型。
 - **resourceMatchers[].version**：（可選）要篩選的資源版本。
 - **resourceMatchers[].names**：（可選）要過濾的資源的 Kubernetes metadata.name 欄位中的名稱。

- **resourceMatchers[].namespaces** : (可選) 要篩選的資源的 Kubernetes metadata.name 欄位中的命名空間。
- **resourceMatchers[].labelSelectors** : (可選) 資源在 Kubernetes metadata.name 欄位中定義的標籤選擇器字串 "[Kubernetes 說明文件](#)"。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 trident-protect-backup-restore-cr.yaml 檔案中填入正確的值後，套用 CR：

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

使用 CLI

步驟

1. 將備份還原到不同的命名空間，並將方括號中的值替換為您環境中的資訊。namespace-mapping`引數使用冒號分隔的命名空間，將來源命名空間對應到正確的目標命名空間，格式為`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> \
--destination-app-name<custom_app_name>\
-n <application_namespace>
```

從備份還原到原始命名空間

您可以隨時將備份還原到原始命名空間。執行就地還原時，Trident Protect 會自動管理保護排程和進行中的作業，以防止無效的還原點：

- 在還原開始之前，應用程式的所有已啟用保護排程都會停用。這可防止在還原應用程式資源時執行排程的備份或快照。
- 恢復成功完成後，只有恢復前已啟用的排程會重新啟用。先前已停用的排程仍保持停用狀態。
- 任何正在進行的備份或快照操作都會在還原開始前取消。如果操作在 5 分鐘內未取消，則還原將繼續進行，並在還原 CR 狀態中記錄警告。

開始之前

請確保 AWS 工作階段權杖的有效期限足以應付任何長時間執行的 s3 還原作業。如果權杖在還原作業期間過期、作業可能會失敗。

- 有關檢查當前會話令牌過期時間的更多資訊，請參閱 ["AWS API 說明文件"](#)。
- 如需 AWS 資源憑證的詳細資訊，請參閱 ["AWS IAM 說明文件"](#)。



當您使用 Kopia 作為資料移動工具還原備份時，您可以選擇在 CR 中或使用 CLI 指定註釋，以控制 Kopia 使用的暫存的行為。有關可配置選項的更多資訊，請參閱 ["Kopia 說明文件"](#)。使用 ``tridentctl-protect create --help`` 命令以取得有關使用 Trident Protect CLI 指定註釋的更多資訊。

使用 CR

步驟

1. 建立自訂資源（CR）檔案並將其命名為 `trident-protect-backup-ipr-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
 - **metadata.name**：（必填）此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
 - **spec.appArchivePath**：AppVault 內儲存備份內容的路徑。您可以使用以下命令來尋找此路徑：

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

- **spec.appVaultRef**：（必填）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. (可選) 如果您只需要選擇應用程式的某些資源進行恢復，請新增篩選條件，以包含或排除帶有特定標籤的資源：



Trident Protect 會自動選擇一些資源，因為它們與您選擇的資源有關聯。例如，如果您選擇了持續性磁碟區宣告資源，且該資源關聯了一個 Pod，則 Trident Protect 也會還原該關聯 Pod。

- **resourceFilter.resourceSelectionCriteria**：(篩選必需) 使用 `Include` 或 `Exclude` 來包含或排除在 resourceMatchers 中定義的資源。新增以下 resourceMatchers 參數以定義要包含或排除的資源：
 - **resourceFilter.resourceMatchers**：resourceMatcher 物件的陣列。如果在此陣列中定義多個元素，它們之間將按 OR 運算匹配，每個元素內的欄位 (group、kind、version) 之間將按 AND 運算匹配。
 - **resourceMatchers[].group**：(可選) 要篩選的資源群組。
 - **resourceMatchers[].kind**：(可選) 要篩選的資源類型。
 - **resourceMatchers[].version**：(可選) 要篩選的資源版本。
 - **resourceMatchers[].names**：(可選) 要過濾的資源的 Kubernetes metadata.name 欄位中的名稱。
 - **resourceMatchers[].namespaces**：(可選) 要篩選的資源的 Kubernetes metadata.name 欄位中的命名空間。
 - **resourceMatchers[].labelSelectors**：(可選) 資源在 Kubernetes metadata.name 欄位中定義的標籤選擇器字串 "[Kubernetes 說明文件](#)"。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 trident-protect-backup-ipr-cr.yaml 檔案中填入正確的值後，套用 CR：

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

使用 CLI

步驟

1. 將備份還原到原始命名空間，並將方括號中的值替換為您環境中的資訊。backup 參數使用命名空間和備份名稱，格式為 <namespace>/<name>。例如：

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

從備份還原到不同的叢集

如果原始叢集出現問題、您可以將備份還原至不同的叢集。



- 當您使用 Kopia 作為資料移動工具還原備份時，您可以選擇在 CR 中或使用 CLI 指定註釋，以控制 Kopia 使用的暫存的行為。有關可配置選項的更多資訊，請參閱 "[Kopia 說明文件](#)"。使用 `tridentctl-protect create --help` 命令以取得有關使用 Trident Protect CLI 指定註釋的更多資訊。
- 使用 CR 還原到新命名空間時，必須先手動建立目標命名空間，然後再套用 CR。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

開始之前

請確保符合下列先決條件：

- 目標叢集已安裝 Trident Protect。
- 目標叢集可以存取與來源叢集相同的 AppVault 儲存桶路徑，備份檔案就儲存在該路徑中。
- 執行 `tridentctl-protect get appvaultcontent` 命令時，請確保本機環境可以連接到 AppVault CR 中定義的物件儲存桶。如果網路限制導致無法存取，請改為在目標叢集的 Pod 內執行 Trident Protect CLI。
- 請確保 AWS 工作階段權杖的有效期限足以應付任何長時間執行的還原作業。如果權杖在還原作業期間過期、作業可能會失敗。
 - 有關檢查當前會話令牌過期時間的更多資訊，請參閱 "[AWS API 說明文件](#)"。
 - 如需 AWS 資源憑證的詳細資訊，請參閱 "[AWS 文件](#)"。

步驟

1. 使用 Trident Protect CLI 外掛程式驗證 AppVault CR 是否存在於目標叢集上：

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



如果目標叢集上不存在 AppVault CR，請按照 ["使用 Trident Protect AppVault 物件來管理儲存桶"](#) 中的步驟建立它。

2. 查看目標叢集上可用 AppVault 的備份內容，並記下 `appArchivePath` 要還原的備份：

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

執行此命令將顯示 AppVault 中的可用備份，包括其來源叢集、相應的應用程式名稱、時間戳記和歸檔路徑。

範例輸出：

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 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到目標叢集：



使用 CR 時，請確保用於應用程式還原的命名空間存在於目的地叢集上。

使用 CR

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-backup-restore-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
 - **metadata.name**：(必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
 - **spec.appVaultRef**：(必填) 儲存備份內容的 AppVault 名稱。
 - **spec.appArchivePath**：(必填) AppVault 內儲存備份內容的路徑。使用步驟 2 中的指令查看備份內容，並找到 `appArchivePath` 要還原的備份。

- **spec.destinationApplicationName**：（選用）還原應用程式的名稱。如果提供、還原的應用程式會使用此名稱。如果未提供、還原的應用程式會使用來源應用程式名稱。
- **spec.namespaceMapping**：復原作業的來源命名空間到目標命名空間的對應。請將 `my-source-namespace` 和 `my-destination-namespace` 替換為您環境中的資訊。

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
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. 在 trident-protect-backup-restore-cr.yaml 檔案中填入正確的值後，套用 CR：

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

使用 CLI

1. 使用以下命令恢復應用程式，並將方括號中的值替換為您環境中的資訊。命名空間映射參數使用冒號分隔的命名空間，將來源命名空間對應到正確的目標命名空間，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> \
--destination-app-name <custom_app_name> \
--context <destination_cluster_name> \
-n <application_namespace>
```

從快照還原到不同的命名空間

您可以使用自訂資源（CR）檔案從快照還原資料，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式，並為還原的應用程式建立一個應用程式 CR。若要保護還原的應用程式，您可以建立隨需備份或快照，或建立保護排程。



- SnapshotRestore 支援 `spec.storageClassMapping` 屬性，但僅當來源儲存類別和目標儲存類別使用相同的儲存後端時才支援。如果嘗試還原到使用不同儲存後端的 `StorageClass`，則還原操作將會失敗。
- 使用 CR 還原到新命名空間時，必須先手動建立目標命名空間，然後再套用 CR。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

開始之前

請確保 AWS 工作階段權杖的有效期限足以應付任何長時間執行的 s3 還原作業。如果權杖在還原作業期間過期、作業可能會失敗。

- 有關檢查當前會話令牌過期時間的更多資訊，請參閱 ["AWS API 說明文件"](#)。
- 如需 AWS 資源憑證的詳細資訊，請參閱 ["AWS IAM 說明文件"](#)。

使用 CR

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-snapshot-restore-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
 - **metadata.name**：(必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
 - **spec.appVaultRef**：(必填) 儲存快照內容的 AppVault 名稱。
 - **spec.appArchivePath**：AppVault 內儲存快照內容的路徑。您可以使用以下命令來尋找此路徑：

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

- **spec.destinationApplicationName**：(選用) 還原應用程式的名稱。如果提供、還原的應用程式會使用此名稱。如果未提供、還原的應用程式會使用來源應用程式名稱。
- **spec.namespaceMapping**：復原作業的來源命名空間到目標命名空間的對應。請將 ``my-source-namespace`` 和 ``my-destination-namespace`` 替換為您環境中的資訊。

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

3. (可選) 如果您只需要選擇應用程式的某些資源進行恢復，請新增篩選條件，以包含或排除帶有特定標籤的資源：



Trident Protect 會自動選擇一些資源，因為它們與您選擇的資源有關聯。例如，如果您選擇了持續性磁碟區宣告資源，且該資源關聯了一個 Pod，則 Trident Protect 也會還原該關聯 Pod。

- **resourceFilter.resourceSelectionCriteria**：(篩選必需) 使用 `Include` 或 `Exclude` 來包含或排除在 resourceMatchers 中定義的資源。新增以下 resourceMatchers 參數以定義要包含或排除的資源：
 - **resourceFilter.resourceMatchers**：resourceMatcher 物件的陣列。如果在此陣列中定義多個元素，它們之間將按 OR 運算匹配，每個元素內的欄位 (group、kind、version) 之間將按 AND 運算匹配。
 - **resourceMatchers[].group**：(可選) 要篩選的資源群組。
 - **resourceMatchers[].kind**：(可選) 要篩選的資源類型。
 - **resourceMatchers[].version**：(可選) 要篩選的資源版本。
 - **resourceMatchers[].names**：(可選) 要過濾的資源的 Kubernetes metadata.name 欄位中的名稱。
 - **resourceMatchers[].namespaces**：(可選) 要篩選的資源的 Kubernetes metadata.name 欄位中的命名空間。
 - **resourceMatchers[].labelSelectors**：(可選) 資源在 Kubernetes metadata.name 欄位中定義的標籤選擇器字串 "[Kubernetes 說明文件](#)"。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 trident-protect-snapshot-restore-cr.yaml 檔案中填入正確的值後，套用 CR：

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

使用 CLI

步驟

1. 將快照還原到不同的命名空間，並將括號中的值替換為您環境中的資訊。

- `snapshot` 引數使用命名空間和快照名稱，格式為 `<namespace>/<name>`。
- `namespace-mapping` 引數使用冒號分隔的命名空間，將來源命名空間對應到正確的目標命名空間，格式為 `<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> \
--destination-app-name <custom_app_name> \
-n <application_namespace>
```

從快照還原到原始命名空間

您可以隨時將快照還原到原始命名空間。執行就地還原時，Trident Protect 會自動管理保護排程和進行中的作業，以防止無效的還原點：

- 在還原開始之前，應用程式的所有已啟用保護排程都會停用。這可防止在還原應用程式資源時執行排程的備份或快照。
- 恢復成功完成後，只有恢復前已啟用的排程會重新啟用。先前已停用的排程仍保持停用狀態。
- 任何正在進行的備份或快照操作都會在還原開始前取消。如果操作在 5 分鐘內未取消，則還原將繼續進行，並在還原 CR 狀態中記錄警告。

開始之前

請確保 AWS 工作階段權杖的有效期限足以應付任何長時間執行的 s3 還原作業。如果權杖在還原作業期間過期、作業可能會失敗。

- 有關檢查當前會話令牌過期時間的更多資訊，請參閱 ["AWS API 說明文件"](#)。
- 如需 AWS 資源憑證的詳細資訊，請參閱 ["AWS IAM 說明文件"](#)。

使用 CR

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-snapshot-ipr-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
 - **metadata.name**：(必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
 - **spec.appVaultRef**：(必填) 儲存快照內容的 AppVault 名稱。

- **spec.archivePath**：AppVault 內儲存快照內容的路徑。您可以使用以下命令來尋找此路徑：

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

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

+ . (可選) 如果您只需要選擇應用程式的某些資源進行恢復，請新增篩選條件，以包含或排除帶有特定標籤的資源：

+ 注意：Trident Protect 會自動選擇某些資源，因為它們與您選擇的資源有關聯。例如，如果您選擇了持續性磁碟區宣告資源，且該資源關聯了一個 Pod，則 Trident Protect 也會還原該關聯 Pod。

+ * **resourceFilter.resourceSelectionCriteria**: (過濾必要) 使用 `Include` 或 `Exclude` 來包含或排除在 `resourceMatchers` 中定義的資源。新增下列 `resourceMatchers` 參數來定義要包含或排除的資源：**resourceFilter.resourceMatchers:resourceMatcher** 物件的陣列。如果在此陣列中定義多個元素，它們之間將按 `OR` 運算匹配，每個元素內的欄位 (`group`、`kind`、`version`) 之間將按 `AND` 運算匹配。* **resourceMatchers[].group**: (可選) 要篩選的資源群組。* **resourceMatchers[].kind**: (可選) 要篩選的資源種類。* **resourceMatchers[].version**: (可選) 要篩選的資源版本。* **resourceMatchers[].names**: (可選) 要篩選的資源的 `Kubernetes metadata.name` 欄位中的名稱。* **resourceMatchers[].namespaces**: (可選) 要篩選的資源的 `Kubernetes metadata.name` 欄位中的命名空間。* **resourceMatchers[].labelSelectors**: (可選) 資源的 `Kubernetes metadata.name` 欄位中的標籤選擇器字串，如 "[Kubernetes 說明文件](#)"中所定義。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"]

```

+ 在 trident-protect-snapshot-ipr-cr.yaml 檔案中填入正確的值後，套用 CR：

+

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

使用 CLI

步驟

1. 將快照還原到原始命名空間，並將方括號中的值替換為您環境中的資訊。例如：

```

tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/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正在進行、已完成或已失敗的還原作業狀態。

步驟

1. 使用以下命令檢索還原作業的狀態，將方括號中的值替換為您環境中的資訊：

```

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

```

使用進階 Trident Protect 還原設定

您可以利用註釋、命名空間設定和儲存選項等進階設定來自訂還原作業，以滿足您的特定需求。

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。來源命名

空間中存在但目的地命名空間中不存在的標籤或註釋會被新增，而任何已存在的標籤或註釋則會被覆寫以符合來源命名空間中的值。僅存在於目的地命名空間中的標籤或註釋則保持不變。



如果您使用 Red Hat OpenShift，請務必注意命名空間註解在 OpenShift 環境中的關鍵作用。命名空間註解可確保復原的 Pod 遵循 OpenShift 安全上下文約束（SCC）定義的相應權限和安全性配置，並能無權限問題地存取磁碟區。如需詳細資訊，請參閱 ["OpenShift 安全上下文約束文檔"](#)。

您可以透過在執行復原或故障轉移操作之前設定 Kubernetes 環境變數

RESTORE_SKIP_NAMESPACE_ANNOTATIONS，來防止目標命名空間中的特定註解被覆寫。例如：

```
helm upgrade trident-protect -n trident-protect netapp-trident-protect/trident-protect \
  --set-string
  restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_key_to_skip_2>}" \
  --reuse-values
```



執行還原或容錯移轉作業時，restoreSkipNamespaceAnnotations 和 restoreSkipNamespaceLabels 中指定的任何命名空間註釋和標籤都會從還原或容錯移轉作業中排除。請確保在初始 Helm 安裝期間配置這些設定。若要深入瞭解，請參閱 ["設定其他 Trident Protect Helm Chart 設定"](#)。

如果您使用 Helm 並帶有 --create-namespace 標誌安裝了來源應用程式，則會對 name 標籤鍵進行特殊處理。在還原或容錯移轉過程中，Trident Protect 會將此標籤複製到目的地命名空間，但如果來源的值與來源命名空間相符，則會將值更新為目的地命名空間值。如果此值與來源命名空間不相符，則會將其複製到目的地命名空間而不做任何變更。

範例

以下範例展示了來源命名空間和目標命名空間，它們各自具有不同的註解和標籤。您可以查看操作前後目標命名空間的狀態，以及註解和標籤在目標命名空間中是如何組合或覆蓋的。

在還原或容錯移轉作業之前

下表說明了復原或容錯移轉作業之前範例來源命名空間和目標命名空間的狀態：

命名空間	註解	標籤
命名空間 ns-1（來源）	<ul style="list-style-type: none">• annotation.one/key: 「updatedvalue」• annotation.two/key: 「true」	<ul style="list-style-type: none">• environment=production• 合規性=hipaa• 名稱=ns-1
命名空間 ns-2（目標）	<ul style="list-style-type: none">• annotation.one/key: 「true」• annotation.three/key: 「false」	<ul style="list-style-type: none">• 角色=資料庫

下表展示了復原或故障轉移作業後範例目標命名空間的狀態。一些鍵已被添加，一些鍵已被覆蓋，並且 name 標籤已更新以匹配目標命名空間：

命名空間	註解	標籤
命名空間 ns-2 (目標)	<ul style="list-style-type: none"> • annotation.one/key: 「updatedvalue」 • annotation.two/key: 「true」 • annotation.three/key: 「false」 	<ul style="list-style-type: none"> • 名稱=ns-2 • 合規性=hipaa • environment=production • 角色=資料庫

支援的欄位

本節說明可用於還原作業的其他欄位。

儲存類別對應

此 `spec.storageClassMapping` 屬性定義了從來源應用程式中的儲存類別到目標叢集上新儲存類別的對應。在將應用程式遷移到具有不同儲存類別的叢集之間，或變更 BackupRestore 操作的儲存後端時，可以使用此屬性。

- 範例：*

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

支援的註釋

本節列出了系統中用於設定各種行為的支援註解。如果使用者未明確設定註解，系統將使用預設值。

註解	類型	說明	預設值
protect.trident.netapp.io/data-mover-timeout-sec	字串	資料移動器操作允許停止的最長時間（以秒為單位）。	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	字串	Kopia 內容快取的大小上限（以 MB 為單位）。	"1000"

註解	類型	說明	預設值
protect.trident.netapp.io/pvc-bind-timeout-sec	字串	等待所有新建立的 PersistentVolumeClaims (PVC) 達到 `Bound` 階段的最長時間（以秒為單位），超過此時間操作將會失敗。適用於所有還原 CR 類型（BackupRestore、BackupInplaceRestore、SnapshotRestore、SnapshotInplaceRestore）。如果您的儲存設備後端或叢集通常需要更多時間，請使用更高的值。	"1200"（20 分鐘）

使用自訂資源還原資源時修改資源

資源轉換功能可讓您在復原資源的過程中進行修改。當復原後的版本需要與原始版本有所不同時，此功能非常有用，例如，在將應用程式還原到不同的叢集時，更新 ConfigMap 中的主機名稱。

資源修改的運作方式

The transformations 欄位位於 SnapshotRestore、BackupRestore、AppMirrorRelationship 及其他復原資源中，可讓您在復原過程中修改 Kubernetes 資源。當復原後的應用程式需要與原始應用程式採用不同的組態時，例如變更主機名稱、登錄 URL、資源限制或環境變數，此功能非常實用。

資源轉換使用 ["RFC 6902 JSON Patch"](#) 操作和 ["RFC 6901 JSON Pointer"](#) 路徑來定位和修改 Kubernetes 資源中的特定欄位。

下面展示了包含資源轉換的還原物件的基本結構：

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-restore
  namespace: target-namespace
spec:
  appVaultRef: my-vault
  appArchivePath: /path/to/snapshot
  namespaceMapping:
    - source: source-ns
      destination: target-ns
  transformations:
    - resource:
        kind: ConfigMap           # Required: resource kind
        group: ""                # Optional: API group (empty for core
resources)
        version: ""              # Optional: API version
        name: ""                 # Optional: specific resource name
      operations:
        - op: replace            # Operation type
          path: "/data/key"      # JSON Pointer path
```

```
value: "new-value" # New value (for add/replace/test)
```

支援的資源

您可以將資源轉換套用至符合下列條件的資源：

- **kind** (必填)：Kubernetes 資源類型 (例如，ConfigMap，Deployment，Pod)
- **group** (可選)：API 群組 (例如，apps，route.openshift.io) — 對於核心資源，請省略
- 版本 (可選)：API 版本 (例如，v1，v1beta1)
- 名稱 (可選)：僅套用於指定名稱的資源



目前不支援修改 PersistentVolumeClaims 和命名空間。

支援的作業

您可以使用以下操作來修改資源：add，copy，move，remove，replace，以及 test。

為資源新增值

使用 add 操作可在指定路徑新增欄位或值。您可以為物件或陣列新增資料。

使用 CR

以下範例向 Deployment 資源新增節點選擇器：

```
transformations:
- resource:
  kind: Deployment
  operations:
  - op: add
    path: "/spec/template/spec/nodeSelector"
    value:
      "topology.kubernetes.io/zone": "us-east-1a"
    disktype: "ssd"
```

使用 CLI

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \
--snapshot <namespace/snapshot-name> \
--namespace-mapping <source-ns>:<dest-ns> \
--transformation
'apps,v1,Deployment:add{"path":"/spec/template/spec/nodeSelector","valu
```

```
e":{"topology.kubernetes.io/zone":"us-east-1a","disktype":"ssd"}}'
```

複製資源中的值

使用 `copy` 操作可將相同資源內的值從一個路徑複製到另一個路徑。來源值不會改變。

使用 CR

以下範例複製 ConfigMap 物件的資料鍵：

```
transformations:
- resource:
  kind: ConfigMap
operations:
- op: copy
  from: "/data/source-key"
  path: "/data/backup-key"
```

使用 CLI

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \
--snapshot <namespace/snapshot-name> \
--namespace-mapping <source-ns>:<dest-ns> \
--transformation ',v1,ConfigMap:copy{"from":"/data/source-key","path":"/data/backup-key"}'
```

在資源內移動值

使用 `move` 操作可將相同資源中的值從一個路徑移至另一個路徑。來源路徑將被移除，值將被放置在目標路徑中。

使用 CR

以下範例重新命名 ConfigMap 物件的資料鍵：

```
transformations:
- resource:
  kind: ConfigMap
operations:
- op: move
  from: "/data/OLD_KEY"
```

```
path: "/data/NEW_KEY"
```

使用 CLI

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \  
  --snapshot <namespace/snapshot-name> \  
  --namespace-mapping <source-ns>:<dest-ns> \  
  --transformation  
' ,v1,ConfigMap:move{"from":"/data/OLD_KEY","path":"/data/NEW_KEY"} '
```

從資源中移除值

使用 `remove` 操作可刪除指定路徑中的欄位或值。

使用 CR

以下範例從 `ConfigMap` 資源中刪除註解：

```
transformations:  
  - resource:  
      kind: ConfigMap  
    operations:  
      - op: remove  
        path: "/metadata/annotations/kubectl.kubernetes.io~1last-  
applied-configuration"
```



在上述範例的路徑中、~1 是 / 的 JSON Pointer 轉義序列。

使用 CLI

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \  
  --snapshot <namespace/snapshot-name> \  
  --namespace-mapping <source-ns>:<dest-ns> \  
  --transformation  
' ,v1,ConfigMap:remove{"path":"/metadata/annotations/kubectl.kubernetes.  
io~1last-applied-configuration"} '
```

取代資源中的值

使用此 `replace` 操作可取代指定路徑下資源中的現有值。JSON 路徑必須已存在。

使用 CR

以下範例變更 Route 物件的主機名稱：

```
transformations:
- resource:
  kind: Route
  group: route.openshift.io
  operations:
  - op: replace
    path: "/spec/host"
    value: "prod.example.com"
```

使用 CLI

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \
--snapshot <namespace/snapshot-name> \
--namespace-mapping <source-ns>:<dest-ns> \
--transformation
'route.openshift.io,v1,Route:replace{"path":"/spec/host","value":"prod.
example.com"}'
```

測試資源修改

使用 test 操作，在執行其他操作之前驗證路徑處的值是否與預期值相符。如果測試失敗，整個修改將會回滾。

使用 CR

以下範例僅在 environment 設定為 staging 時，才會更新 database-host：

```
transformations:
- resource:
  kind: ConfigMap
  operations:
  - op: test
    path: "/data/environment"
    value: "staging"
  - op: replace
    path: "/data/database-host"
```

```
value: "prod-db.example.com"
```

使用 CLI

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \  
  --snapshot <namespace/snapshot-name> \  
  --namespace-mapping <source-ns>:<dest-ns> \  
  --transformation  
' ,v1,ConfigMap:test{"path":"/data/environment","value":"staging"},repla  
ce{"path":"/data/database-host","value":"prod-db.example.com"} '
```

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。