



アプリケーションをリストアする Trident

NetApp
October 29, 2026

目次

アプリケーションをリストアする	1
Trident Protectを使用してアプリケーションを復元する	1
バックアップから別の名前空間に復元する	1
バックアップから元の名前空間に復元する	4
バックアップから別のクラスタにリストアする	6
スナップショットから別の namespace にリストアする	9
スナップショットから元の名前空間に復元する	12
リストア処理のステータスを確認する	14
高度な Trident Protect リストア設定を使用する	15
リストアおよびフェイルオーバー処理中のネームスペースのアノテーションとラベル	15
サポートされているフィールド	16
サポートされているアノテーション	17
カスタムリソースを使用して復元する際にリソースを変更します	17
リソース変更の仕組み	17
サポートされているリソース	18
サポートされている操作	18

アプリケーションをリストアする

Trident Protectを使用してアプリケーションを復元する

Trident Protectを使用して、スナップショットまたはバックアップからアプリケーションを復元できます。アプリケーションを同じクラスターに復元する場合、既存のスナップショットからの復元の方が高速になります。



- アプリケーションを復元すると、アプリケーションに対して設定されているすべての実行フックもアプリとともに復元されます。リストア後の実行フックが存在する場合、リストア処理の一部として自動的に実行されます。
- バックアップから別の名前空間または元の名前空間への復元は、qtree ボリュームでサポートされています。ただし、スナップショットから別の名前空間または元の名前空間への復元は、qtree ボリュームではサポートされていません。
- 詳細設定を使用して復元操作をカスタマイズできます。詳細については、"[高度な Trident Protect リストア設定を使用する](#)"を参照してください。
- アーカイブからすべての名前空間リソースを復元せずに特定のKubeVirt仮想マシン（VM）を復元するには、復元CR の `includedVirtualMachines` フィールドを使用します。サポートされているCR、フィールドリファレンス、およびYAMLの例については、"[KubeVirt 仮想マシンを Trident Protect で保護する](#)"を参照してください。

バックアップから別の名前空間に復元する

BackupRestore CR を使用して異なる名前空間にバックアップを復元すると、Trident Protect は、アプリケーションを新しい名前空間に復元し、復元されたアプリケーションのアプリケーション CR を作成します。復元されたアプリケーションを保護するには、オンデマンド バックアップまたはスナップショットを作成するか、保護スケジュールを設定します。



- 既存のリソースを含む別の名前空間にバックアップをリストアしても、バックアップ内のリソースと名前を共有するリソースは変更されません。バックアップ内のすべてのリソースをリストアするには、ターゲット名前空間を削除して再作成するか、バックアップを新しい名前空間にリストアします。
- CR を使用して新しい名前空間に復元する場合は、CR を適用する前に、宛先名前空間を手動で作成する必要があります。Trident Protect は、CLI を使用する場合にのみネームスペースを自動的に作成します。

開始する前に

AWS セッショントークンの有効期限が、長時間実行される s3 復元処理に十分であることを確認します。復元操作中にトークンの有効期限が切れると、操作が失敗する可能性があります。

- 現在のセッショントークンの有効期限を確認する方法の詳細については、"[AWS API ドキュメント](#)"を参照してください。
- AWS リソースの認証情報の詳細については、"[AWS IAM ドキュメント](#)"を参照してください。



Kopia をデータ ムーバーとして使用してバックアップを復元する場合、オプションで CR に注釈を指定するか、CLI を使用して Kopia が使用する一時ストレージの動作を制御できます。設

定できるオプションの詳細については、"[Kopiaのドキュメント](#)"を参照してください。Trident Protect CLI で注釈を指定する方法の詳細については、`tridentctl-protect create --help` コマンドを使用してください。

CRを使用する

手順

1. カスタムリソース（CR）ファイルを作成し、`trident-protect-backup-restore-cr.yaml`という名前を付けます。
2. 作成したファイルで、次の属性を設定します：

- **metadata.name**：（必須）このカスタム リソースの名前。環境に合わせて一意かつ適切な名前を選択してください。
- **spec.appArchivePath**：AppVault内でバックアップコンテンツが保存されるパス。このパスを見つけるには、次のコマンドを使用できます：

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

- **spec.appVaultRef**：（必須）バックアップコンテンツが保存されるAppVaultの名前。
- **spec.destinationApplicationName**：（オプション）復元されたアプリケーションの名前。指定した場合、復元されたアプリケーションはこの名前を使用します。指定しない場合、復元されたアプリケーションは元のアプリケーション名を使用します。
- **spec.namespaceMapping**：リストア処理のソースネームスペースからデスティネーションネームスペースへのマッピング。`my-source-namespace`と`my-destination-namespace`を環境の情報に置き換えます。

```
---  
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. （オプション）アプリケーションの特定のリソースのみを選択して復元する必要がある場合は、特定のラベルが付いたリソースを含めるか除外するかを指定するフィルタリングを追加します。



Trident Protect は、選択したリソースとの関係に基づいて、一部のリソースを自動的に選択します。例えば、永続ボリューム要求リソースを選択し、それに関連付けられ

たポッドがある場合、Trident Protect は、関連付けられたポッドも復元します。

- **resourceFilter.resourceSelectionCriteria** : (フィルタリングに必須) `Include`または`Exclude`を使用して、resourceMatchersで定義されたリソースを含めるか除外します。以下のresourceMatchersパラメータを追加して、含めるまたは除外するリソースを定義します:
 - **resourceFilter.resourceMatchers** : resourceMatcherオブジェクトの配列。この配列に複数の要素を定義した場合、それらはOR演算で一致し、各要素内のフィールド (group、kind、version) はAND演算で一致します。
 - **resourceMatchers[].group** : (オプション) フィルタリングするリソースのグループ。
 - **resourceMatchers[].kind** : (オプション) フィルタリングするリソースの種類。
 - **resourceMatchers[].version** : (オプション) フィルタリングするリソースのバージョン。
 - **resourceMatchers[].names** : (オプション) フィルタリングするリソースのKubernetes metadata.name フィールド内の名前。
 - **resourceMatchers[].namespaces** : (オプション) フィルタリングするリソースのKubernetes metadata.name フィールド内の名前空間。
 - **resourceMatchers[].labelSelectors** : (オプション) "[Kubernetesドキュメント](#)"で定義されているリソースのKubernetesメタデータ.nameフィールドのラベルセクタ文字列。
例: "trident.netapp.io/os=linux"

次に例を示します。

```
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. `trident-protect-backup-restore-cr.yaml` ファイルに正しい値を入力したら、CRを適用します。

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

CLI を使用する

手順

1. 括弧内の値を環境の情報に置き換えて、バックアップを別の名前空間に復元します。 namespace-mapping` 引数はコロンで区切られた名前空間を使用して、ソース名前空間を正しいデスティネーション名前空間にマッピングします（形式は `source1:dest1,source2:dest2`）。次に例を示します。

```
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>
```

バックアップから元の名前空間に復元する

バックアップはいつでも元の名前空間に復元できます。In Place リストアを実行すると、Trident Protect は、無効なリカバリポイントを防止するために、保護スケジュールと進行中の操作を自動的に管理します。

- 復元が開始される前に、アプリケーションに対して有効になっているすべての保護スケジュールが無効になります。これにより、アプリケーションリソースの復元中に、スケジュールされたバックアップやスナップショットが実行されるのを防ぎます。
- 復元が正常に完了すると、復元前に有効になっていたスケジュールのみが再度有効になります。既に無効化されているスケジュールは、引き続き無効化されたままです。
- 復元が開始される前に、進行中のバックアップまたはスナップショット操作はすべてキャンセルされます。操作が5分以内にキャンセルされない場合、復元処理は続行され、復元CRステータスに警告が記録されます。

開始する前に

AWS セッショントークンの有効期限が、長時間実行される s3 復元処理に十分であることを確認します。復元操作中にトークンの有効期限が切れると、操作が失敗する可能性があります。

- 現在のセッショントークンの有効期限を確認する方法の詳細については、"[AWS API ドキュメント](#)"を参照してください。
- AWS リソースの認証情報の詳細については、"[AWS IAM ドキュメント](#)"を参照してください。



Kopia をデータムーバーとして使用してバックアップを復元する場合、オプションで CR に注釈を指定するか、CLI を使用して Kopia が使用する一時ストレージの動作を制御できます。設定できるオプションの詳細については、"[Kopiaのドキュメント](#)"を参照してください。Trident Protect CLI で注釈を指定する方法の詳細については、`tridentctl-protect create --help` コマンドを使用してください。

CRを使用する

手順

1. カスタムリソース（CR）ファイルを作成し、`trident-protect-backup-ipr-cr.yaml` という名前を付けます。

2. 作成したファイルで、次の属性を設定します：

- **metadata.name**：（必須）このカスタム リソースの名前。環境に合わせて一意かつ適切な名前を選択してください。
- **spec.appArchivePath**：AppVault内でバックアップコンテンツが保存されるパス。このパスを見つけるには、次のコマンドを使用できます：

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

- **spec.appVaultRef**：（必須）バックアップコンテンツが保存されるAppVaultの名前。

次に例を示します。

```
---  
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. （オプション）アプリケーションの特定のリソースのみを選択して復元する必要がある場合は、特定のラベルが付いたリソースを含めるか除外するかを指定するフィルタリングを追加します。



Trident Protect は、選択したリソースとの関係に基づいて、一部のリソースを自動的に選択します。例えば、永続ボリューム要求リソースを選択し、それに関連付けられたポッドがある場合、Trident Protect は、関連付けられたポッドも復元します。

- **resourceFilter.resourceSelectionCriteria**：（フィルタリングに必須）`Include`または`Exclude`を使用して、resourceMatchersで定義されたリソースを含めるか除外します。以下のresourceMatchersパラメータを追加して、含めるまたは除外するリソースを定義します：
 - **resourceFilter.resourceMatchers**：resourceMatcherオブジェクトの配列。この配列に複数の要素を定義した場合、それらはOR演算で一致し、各要素内のフィールド（group、kind、version）はAND演算で一致します。
 - **resourceMatchers[].group**：（オプション）フィルタリングするリソースのグループ。
 - **resourceMatchers[].kind**：（オプション）フィルタリングするリソースの種類。
 - **resourceMatchers[].version**：（オプション）フィルタリングするリソースのバージョン。
 - **resourceMatchers[].names**：（オプション）フィルタリングするリソースのKubernetes metadata.name フィールド内の名前。
 - **resourceMatchers[].namespaces**：（オプション）フィルタリングするリソースの

Kubernetes metadata.name フィールド内の名前空間。

- **resourceMatchers[].labelSelectors** : (オプション) "[Kubernetesドキュメント](#)"で定義されているリソースのKubernetesメタデータ.nameフィールドのラベルセクタ文字列。
例: "trident.netapp.io/os=linux"

次に例を示します。

```
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. 'trident-protect-backup-ipr-cr.yaml' ファイルに正しい値を入力したら、CRを適用します。

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

CLI を使用する

手順

1. 括弧内の値を環境の情報に置き換えて、バックアップを元の名前空間に復元します。`backup` 引数は、名前空間とバックアップ名を `/` の形式で使います。次に例を示します。

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

バックアップから別のクラスタにリストアする

元のクラスタに問題がある場合は、バックアップを別のクラスタにリストアできます。



- Kopia をデータムーバーとして使用してバックアップを復元する場合、オプションで CR に注釈を指定するか、CLI を使用して Kopia が使用する一時ストレージの動作を制御できます。設定できるオプションの詳細については、"[Kopiaのドキュメント](#)"を参照してください。Trident Protect CLI で注釈を指定する方法の詳細については、`tridentctl-protect create --help` コマンドを使用してください。
- CR を使用して新しい名前空間に復元する場合は、CR を適用する前に、宛先名前空間を手動で作成する必要があります。Trident Protect は、CLI を使用する場合にのみネームスペースを自動的に作成します。

開始する前に

次の前提条件が満たされていることを確認してください：

- デスティネーション クラスタに Trident Protect がインストールされている。
- デスティネーション クラスタは、バックアップが保存されているソース クラスタと同じAppVaultのバケット パスにアクセスできます。
- ローカル環境が、`tridentctl-protect get appvaultcontent` コマンドの実行時にAppVault CRで定義されたオブジェクトストレージバケットに接続できることを確認してください。ネットワーク制限によりアクセスできない場合は、代わりにデスティネーション クラスタ上のポッド内からTrident Protect CLIを実行してください。
- AWS セッション トークンの有効期限が、長時間実行される復元操作に十分であることを確認します。復元操作中にトークンの有効期限が切れると、操作が失敗する可能性があります。
 - 現在のセッショントークンの有効期限を確認する方法の詳細については、"[AWS API ドキュメント](#)"を参照してください。
 - AWS リソースの認証情報の詳細については、"[AWSのドキュメント](#)"を参照してください。

手順

1. Trident Protect CLIプラグインを使用して、デスティネーション クラスタにAppVault CRが存在することを確認します：

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



AppVault CR がデスティネーション クラスタに存在しない場合は、"[Trident Protect AppVaultオブジェクトを使用してバケットを管理する](#)"の手順に従って作成します。

2. デスティネーション クラスタで利用可能なAppVaultのバックアップコンテンツを表示し、復元するバックアップの `appArchivePath` をメモします：

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

このコマンドを実行すると、AppVault内の利用可能なバックアップが表示されます。これには、元のクラスタ、対応するアプリケーション名、タイムスタンプ、アーカイブ パスが含まれます。

出力例：

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|  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. AppVault名前とアーカイブパスを使用して、アプリケーションをデスティネーション クラスタに復元します：



CRを使用する場合は、アプリケーションのリストアに使用する名前空間がデスティネーション クラスタに存在することを確認してください。

CRを使用する

1. カスタムリソース（CR）ファイルを作成し、`trident-protect-backup-restore-cr.yaml`という名前を付けます。
2. 作成したファイルで、次の属性を設定します：
 - **metadata.name**：（必須）このカスタム リソースの名前。環境に合わせて一意かつ適切な名前を選択してください。
 - **spec.appVaultRef**：（必須）バックアップコンテンツが保存されるAppVaultの名前。
 - **spec.appArchivePath**：（必須）AppVault内のバックアップコンテンツが保存されるパス。ステップ2のコマンドを使用してバックアップの内容を表示し、復元するバックアップの`appArchivePath`を確認します。
 - **spec.destinationApplicationName**：（オプション）復元されたアプリケーションの名前。指定した場合、復元されたアプリケーションはこの名前を使用します。指定しない場合、復元されたアプリケーションは元のアプリケーション名を使用します。
 - **spec.namespaceMapping**： リストア処理のソースネームスペースからデスティネーションネームスペースへのマッピング。`my-source-namespace`と`my-destination-namespace`を環境の情報に置き換えます。

次に例を示します。

```
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. `trident-protect-backup-restore-cr.yaml` ファイルに正しい値を入力したら、CRを適用します。

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

CLI を使用する

1. 次のコマンドを使用してアプリケーションを復元し、括弧内の値を環境の情報に置き換えます。namespace-mapping 引数は、コロンで区切られた名前空間を使用して、source1:dest1、source2:dest2 の形式でソース名前空間を正しい宛先名前空間にマッピングします。次に例を示します。

```

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>

```

スナップショットから別の namespace にリストアする

カスタム リソース (CR) ファイルを使用して、スナップショットからデータを別の名前空間または元のソース名前空間に復元できます。SnapshotRestore CR を使用してスナップショットを別の名前空間に復元すると、Trident Protect は新しい名前空間にアプリケーションを復元し、復元されたアプリケーションのアプリケーション CR を作成します。復元されたアプリケーションを保護するには、オンデマンド バックアップまたはスナップショットを作成するか、保護スケジュールを設定します。



- SnapshotRestoreは `spec.storageClassMapping` 属性をサポートしていますが、ソース ストレージ クラスとデスティネーション ストレージ クラスが同じストレージ バックエンドを使用する場合のみです。異なるストレージ バックエンドを使用する `StorageClass` に復元しようとする、復元処理は失敗します。
- CR を使用して新しい名前空間に復元する場合は、CR を適用する前に、宛先名前空間を手動で作成する必要があります。Trident Protect は、CLI を使用する場合にのみネームスペースを自動的に作成します。

開始する前に

AWS セッショントークンの有効期限が、長時間実行される s3 復元処理に十分であることを確認します。復元操作中にトークンの有効期限が切れると、操作が失敗する可能性があります。

- 現在のセッショントークンの有効期限を確認する方法の詳細については、"[AWS API ドキュメント](#)"を参照してください。
- AWS リソースの認証情報の詳細については、"[AWS IAM ドキュメント](#)"を参照してください。

CRを使用する

手順

1. カスタムリソース（CR）ファイルを作成し、`trident-protect-snapshot-restore-cr.yaml` という名前を付けます。
2. 作成したファイルで、次の属性を設定します：
 - **metadata.name**：（必須）このカスタム リソースの名前。環境に合わせて一意かつ適切な名前を選択してください。
 - **spec.appVaultRef**：（必須）スナップショットの内容が保存されるAppVaultの名前。
 - **spec.appArchivePath**：AppVault内のパスで、スナップショットの内容が保存される場所。このパスを見つけるには、次のコマンドを使用できます：

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

- **spec.destinationApplicationName**：（オプション）復元されたアプリケーションの名前。指定した場合、復元されたアプリケーションはこの名前を使用します。指定しない場合、復元されたアプリケーションは元のアプリケーション名を使用します。
- **spec.namespaceMapping**：リストア処理のソースネームスペースからデスティネーションネームスペースへのマッピング。`my-source-namespace`と`my-destination-namespace`を環境の情報に置き換えます。

```
---  
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. （オプション）アプリケーションの特定のリソースのみを選択して復元する必要がある場合は、特定のラベルが付いたリソースを含めるか除外するかを指定するフィルタリングを追加します。



Trident Protect は、選択したリソースとの関係に基づいて、一部のリソースを自動的に選択します。例えば、永続ボリューム要求リソースを選択し、それに関連付けられたポッドがある場合、Trident Protect は、関連付けられたポッドも復元します。

- **resourceFilter.resourceSelectionCriteria** : (フィルタリングに必須) `Include` または `Exclude` を使用して、resourceMatchers で定義されたリソースを含めるか除外します。以下の resourceMatchers パラメータを追加して、含めるまたは除外するリソースを定義します：
 - **resourceFilter.resourceMatchers** : resourceMatcher オブジェクトの配列。この配列に複数の要素を定義した場合、それらは OR 演算で一致し、各要素内のフィールド (group、kind、version) は AND 演算で一致します。
 - **resourceMatchers[].group** : (オプション) フィルタリングするリソースのグループ。
 - **resourceMatchers[].kind** : (オプション) フィルタリングするリソースの種類。
 - **resourceMatchers[].version** : (オプション) フィルタリングするリソースのバージョン。
 - **resourceMatchers[].names** : (オプション) フィルタリングするリソースの Kubernetes metadata.name フィールド内の名前。
 - **resourceMatchers[].namespaces** : (オプション) フィルタリングするリソースの Kubernetes metadata.name フィールド内の名前空間。
 - **resourceMatchers[].labelSelectors** : (オプション) ["Kubernetes ドキュメント"](#) で定義されているリソースの Kubernetes メタデータ.name フィールドのラベルセクタ文字列。
例: "trident.netapp.io/os=linux"

次に例を示します。

```
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. `trident-protect-snapshot-restore-cr.yaml` ファイルに正しい値を入力したら、CRを適用します。

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

CLI を使用する

手順

1. 括弧内の値を環境の情報に置き換えて、スナップショットを別のネームスペースにリストアします。

- その snapshot`引数は、名前空間とスナップショット名を次の形式で使します
`<namespace>/<name>。
- namespace-mapping`引数はコロンで区切られた名前空間を使用して、ソース名前空間を正しいデスティネーション名前空間にマッピングします（形式は
`source1:dest1,source2:dest2）。

次に例を示します。

```
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>
```

スナップショットから元の名前空間に復元する

スナップショットはいつでも元の名前空間に復元できます。In Placeリストアを実行すると、Trident Protect は、無効なリカバリポイントを防止するために、保護スケジュールと進行中の操作を自動的に管理します。

- 復元が開始される前に、アプリケーションに対して有効になっているすべての保護スケジュールが無効になります。これにより、アプリケーションリソースの復元中に、スケジュールされたバックアップやスナップショットが実行されるのを防ぎます。
- 復元が正常に完了すると、復元前に有効になっていたスケジュールのみが再度有効になります。既に無効化されているスケジュールは、引き続き無効化されたままです。
- 復元が開始される前に、進行中のバックアップまたはスナップショット操作はすべてキャンセルされます。操作が5分以内にキャンセルされない場合、復元処理は続行され、復元CRステータスに警告が記録されます。

開始する前に

AWS セッショントークンの有効期限が、長時間実行される s3 復元処理に十分であることを確認します。復元操作中にトークンの有効期限が切れると、操作が失敗する可能性があります。

- 現在のセッショントークンの有効期限を確認する方法の詳細については、"[AWS API ドキュメント](#)"を参照してください。
- AWS リソースの認証情報の詳細については、"[AWS IAM ドキュメント](#)"を参照してください。

CRを使用する

手順

1. カスタムリソース（CR）ファイルを作成し、`trident-protect-snapshot-ipr-cr.yaml`という名前を付けます。
2. 作成したファイルで、次の属性を設定します：
 - **metadata.name**：（必須）このカスタム リソースの名前。環境に合わせて一意かつ適切な名前を選択してください。
 - **spec.appVaultRef**：（必須）スナップショットの内容が保存されるAppVaultの名前。
 - **spec.appArchivePath**：AppVault内のパスで、スナップショットの内容が保存される場所。このパスを見つけるには、次のコマンドを使用できます：

```
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
```

+ .（オプション）アプリケーションの特定のリソースのみを選択して復元する必要がある場合は、特定のラベルが付いたリソースを含めるか除外するかを指定するフィルタリングを追加します。

+ 注記：Trident Protect は、ユーザーが選択したリソースとの関連性に基づいて、一部のリソースを自動的に選択します。例えば、永続ボリューム要求リソースを選択し、それに関連付けられたポッドがある場合、Trident Protect は、関連付けられたポッドも復元します。

+ * **resourceFilter.resourceSelectionCriteria**：（フィルタリングに必須） `Include`または`Exclude``を使用して、`resourceMatchers`で定義されたリソースを含めるか除外します。含めるリソースまたは除外するリソースを定義するには、以下の`resourceMatchers`パラメータを追加します：
resourceFilter.resourceMatchers：`resourceMatcher`オブジェクトの配列。この配列に複数の要素を定義した場合、それらはOR演算で一致し、各要素内のフィールド（`group`、`kind`、`version`）はAND演算で一致します。
* **resourceMatchers[].group**：（オプション）フィルタリングするリソースのグループ。
* **resourceMatchers[].kind**：（オプション）フィルタリングするリソースの種類。
* **resourceMatchers[].version**：（オプション）フィルタリングするリソースのバージョン。
* **resourceMatchers[].names**：（オプション）フィルタリングするリソースのKubernetes `metadata.name`フィールドの名前。
* **resourceMatchers[].namespaces**：（オプション）フィルタリングするリソースのKubernetes `metadata.name`フィールドにある名前空間。
* **resourceMatchers[].labelSelectors**：（オプション） ["Kubernetesドキュメント"](#)で定義されたリソースのKubernetes `metadata.name`フィールド内のラベルセレクタ文字列。例：
``trident.netapp.io/os=linux``

+ 例：

+

```
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"]
```

+ . `trident-protect-snapshot-ipr-cr.yaml` ファイルに正しい値を入力したら、CRを適用します。

+

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

CLI を使用する

手順

1. 括弧内の値を環境の情報に置き換えて、スナップショットを元の名前空間に復元します。次に例を示します。

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

リストア処理のステータスを確認する

コマンド ラインを使用して、進行中、完了、または失敗した復元処理のステータスを確認できます。

手順

1. 復元操作のステータスを取得するには、次のコマンドを使用します。括弧内の値は、ご使用の環境の情報に置き換えてください：

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

高度な Trident Protect リストア設定を使用する

注釈、名前空間設定、ストレージ オプションなどの詳細設定を使用して、特定の要件を満たすように復元操作をカスタマイズできます。

リストアおよびフェイルオーバー処理中のネームスペースのアノテーションとラベル

復元およびフェイルオーバー処理中に、デスティネーションネームスペースのラベルとアノテーションは、ソースネームスペースのラベルとアノテーションと一致するように作成されます。デスティネーションネームスペースに存在しないソースネームスペースのラベルまたはアノテーションが追加され、既存のラベルまたはアノテーションはソースネームスペースの値と一致するように上書きされます。デスティネーションネームスペースにのみ存在するラベルまたはアノテーションは変更されません。



Red Hat OpenShiftを使用する場合は、OpenShift環境における名前空間アノテーションの重要な役割に注意することが重要です。名前空間アノテーションにより、リストアされたポッドがOpenShiftセキュリティコンテキスト制約（SCC）で定義された適切な権限とセキュリティ設定に準拠し、権限の問題なくボリュームにアクセスできるようになります。詳細については、"[OpenShiftセキュリティコンテキスト制約ドキュメント](#)"を参照してください。

リストアまたはフェイルオーバー処理を実行する前にKubernetes環境変数

`RESTORE_SKIP_NAMESPACE_ANNOTATIONS`を設定することで、デスティネーションネームスペース内の特定のアノテーションが上書きされるのを防ぐことができます。次に例を示します。

```
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
```



リストアまたはフェイルオーバー操作を実行すると、`restoreSkipNamespaceAnnotations`および`restoreSkipNamespaceLabels`で指定された名前空間アノテーションとラベルは、リストアまたはフェイルオーバー操作から除外されます。これらの設定は、Helmの初回インストール時に必ず構成してください。詳細については、"[追加のTrident Protectヘルムチャート設定を構成する](#)"を参照してください。

Helm を使用して `--create-namespace` フラグを指定してソースアプリケーションをインストールした場合、`name` ラベルキーには特別な処理が適用されます。リストアまたはフェイルオーバープロセス中に、Trident Protect はこのラベルをデスティネーションネームスペースにコピーしますが、ソースからの値がソースネームスペースと一致する場合は、値をデスティネーションネームスペースの値に更新します。この値がソースネームスペースと一致しない場合は、変更されずにデスティネーションネームスペースにコピーされます。

例

次の例は、それぞれ異なる注釈とラベルを持つソース名前空間と宛先名前空間を示しています。操作の前後の宛先名前空間の状態や、宛先名前空間で注釈とラベルがどのように結合または上書きされるかを確認できます。

リストアまたはフェイルオーバー処理の前に

次の表は、リストアまたはフェイルオーバー処理前のサンプルのソースネームスペースとデスティネーションネームスペースの状態を示しています：

ネームスペース	アノテーション	ラベル
ネームスペースns-1 (ソース)	<ul style="list-style-type: none">• annotation.one/key : "updatedvalue"• annotation.two/key : "true"	<ul style="list-style-type: none">• environment=production• コンプライアンス=hipaa• name=ns-1
ネームスペースns-2 (宛先)	<ul style="list-style-type: none">• annotation.one/key : "true"• annotation.three/key : "false"	<ul style="list-style-type: none">• ロール=database

リストア処理後

次の表は、復元またはフェイルオーバー操作後の宛先名前空間の例の状態を示しています。いくつかのキーが追加され、いくつかは上書きされ、`name`ラベルは宛先名前空間と一致するように更新されました：

ネームスペース	アノテーション	ラベル
ネームスペースns-2 (宛先)	<ul style="list-style-type: none">• annotation.one/key : "updatedvalue"• annotation.two/key : "true"• annotation.three/key : "false"	<ul style="list-style-type: none">• name=ns-2• コンプライアンス=hipaa• environment=production• ロール=database

サポートされているフィールド

このセクションでは、復元操作に使用できる追加のフィールドについて説明します。

ストレージクラスのマッピング

`spec.storageClassMapping`属性は、ソース アプリケーションに存在するストレージクラスからターゲット クラスタ上の新しいストレージクラスへのマッピングを定義します。異なるストレージクラスを持つクラスタ間でアプリケーションを移行する場合や、BackupRestore操作のストレージバックエンドを変更する場合にこれを使用できます。

例：

```
storageClassMapping:
```

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

サポートされているアノテーション

このセクションでは、システムにおける様々な動作を設定するためにサポートされている注釈を一覧表示します。ユーザーがアノテーションを明示的に設定しない場合、システムはデフォルト値を使用します。

注釈	タイプ	概要	デフォルト値
protect.trident.netapp.io/data-mover-timeout-sec	string	データムーバー処理が停止する最大時間（秒単位）。	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	string	Kopia コンテンツ キャッシュの最大サイズ制限（メガバイト単位）。	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	string	新しく作成されたPersistentVolumeClaims（PVC）が`Bound`フェーズに到達するまで待機する最大時間（秒単位）。この時間を超えると処理は失敗します。環境すべての復元CRタイプ（BackupRestore、BackupInplaceRestore、SnapshotRestore、SnapshotInplaceRestore）に適用されます。ストレージバックエンドまたはクラスターの処理に時間がかかることが多い場合は、より大きな値を使用してください。	「1200」（20分）

カスタムリソースを使用して復元する際にリソースを変更します

リソース変換機能を使用すると、リソースの復元中にリソースを変更できます。これは、復元されたバージョンがオリジナルと異なる必要がある場合に役立ちます。たとえば、アプリケーションを別のクラスターに復元する際にConfigMapのホスト名を更新する場合などです。

リソース変更の仕組み

`SnapshotRestore`、`BackupRestore`、`AppMirrorRelationship`、およびその他のリストア リソースの`transformations`フィールドを使用すると、リストア プロセス中に Kubernetes リソースを変更できます。これは、リストアされたアプリケーションをオリジナルとは異なる方法で構成する必要がある場合（ホスト名、レジストリ URL、リソース制限、環境変数の変更など）に便利です。

リソース変換では、"RFC 6902 JSON Patch"操作と "RFC 6901 JSONポインタ"パスを使用して、Kubernetesリソース内の特定のフィールドをターゲットにして変更します。

以下に、リソース変換を含む復元オブジェクトの基本的な構造を示します。

```
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)
```

サポートされているリソース

以下の条件に一致するリソースに対して、リソース変換を使用できます：

- **kind**（必須）：Kubernetes リソースの種類（例：ConfigMap、Deployment、Pod）
- **グループ**（オプション）：APIグループ（例：apps、route.openshift.io） — コアリソースの場合は省略
- **version**（オプション）：APIバージョン（例：v1、v1beta1）
- **名前**（オプション）：名前で指定した特定のリソースにのみ適用します



PersistentVolumeClaimsと名前空間の変更は現在サポートされていません。

サポートされている操作

リソースを変更するには、以下の操作を使用できます：add、copy、move、remove、replace、およびtest。

リソースに値を追加する

`add`操作を使用して、指定されたパスに新しいフィールドまたは値を追加します。オブジェクトや配列にデータを追加できます。

CRを使用する

次の例では、ノードセクタを `Deployment` リソースに追加します：

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

CLIを使用する

コマンドラインを使用してこの変換を実行するには、次のコマンドを使用します：

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

リソース内の値をコピーする

`copy`操作を使用して、同じリソース内のあるパスから別のパスに値をコピーします。ソースは変更されません。

CRを使用する

次の例では、`ConfigMap` オブジェクトのデータキーを複製します：

```
transformations:
- resource:
  kind: ConfigMap
```

```
operations:
  - op: copy
    from: "/data/source-key"
    path: "/data/backup-key"
```

CLI を使用する

コマンドラインを使用してこの変換を実行するには、次のコマンドを使用します：

```
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"}'
```

リソース内で値を移動する

`move` 操作を使用して、同じリソース内であるパスから別のパスに値を移動します。ソースが削除され、値が宛先に配置されます。

CR を使用する

次の例では、`ConfigMap` オブジェクトのデータキーの名前を変更します：

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

CLI を使用する

コマンドラインを使用してこの変換を実行するには、次のコマンドを使用します：

```
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"}'
```

リソースから値を削除する

`remove` 操作を使用して、指定されたパスにあるフィールドまたは値を削除します。

CRを使用する

次の例では、`ConfigMap` リソースから注釈を削除します：

```
transformations:
- resource:
  kind: ConfigMap
  operations:
  - op: remove
    path: "/metadata/annotations/kubect1.kubernetes.io~1last-
applied-configuration"
```



上記の例のパスでは、`~1` は `\'` のJSONポインターのエスケープシーケンスです。

CLIを使用する

コマンドラインを使用してこの変換を実行するには、次のコマンドを使用します：

```
tridentctl-protect create snapshotrestore <restore_name> \
--snapshot <namespace/snapshot-name> \
--namespace-mapping <source-ns>:<dest-ns> \
--transformation
',v1,ConfigMap:remove{"path":"/metadata/annotations/kubect1.kubernetes.
io~1last-applied-configuration"}'
```

リソース内の値を置き換える

`replace` 操作を使用して、指定されたパスにあるリソース内の既存の値を置き換えます。JSONパスは既に存在している必要があります。

CRを使用する

次の例では、`Route` オブジェクトのホスト名を変更します：

```
transformations:
- resource:
  kind: Route
  group: route.openshift.io
```

```
operations:
  - op: replace
    path: "/spec/host"
    value: "prod.example.com"
```

CLI を使用する

コマンドラインを使用してこの変換を実行するには、次のコマンドを使用します：

```
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"}'
```

リソースの変更をテストする

`test` 操作を使用して、他の操作が適用される前に、パス上の値が期待値と一致することを確認します。テストが失敗した場合、変更全体がロールバックされます。

CR を使用する

次の例では、`environment` が `staging` に設定されている場合にのみ `database-host` を更新します。

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

CLI を使用する

コマンドラインを使用してこの変換を実行するには、次のコマンドを使用します：

```
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"} '
```

著作権に関する情報

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S. このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータ ソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。