アプリケーションをリストアする
Trident

NetApp
July 01, 2026

目次

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。

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）復元するアプリケーションの特定のリソースのみを選択する必要がある場合は、特定のラベルでマークされたリソースを含めるか除外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。たとえば、永続ボリュームクレームリソースを選択し、それに関連付けられたポッドがある場合、Trident Protect は関連付けられている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すると、それらはOR演算として一致し、各要素内のフィールド（グループ、種類、バージョン）はAND演算として一致します。
  - **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レクタ文字列。例えば：`"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`）。例：

```
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）復元するアプリケーションの特定のリソースのみを選択する必要がある場合は、特定のラベルでマークされたリソースを含めるか除外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。たとえば、永続ボリュームクレームリソースを選択し、それに関連付けられたポッドがある場合、Trident Protect は関連付けられている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すると、それらはOR演算として一致し、各要素内のフィールド（グループ、種類、バージョン）はAND演算として一致します。
       - **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レクタ文字列。例えば：`"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 `<namespace>/<name>`の形式で使用します。例：

```
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。例：

```
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と、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 <SNAPHOT_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-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. （オプション）復元するアプリケーションの特定のリソースのみを選択する必要がある場合は、特定のラベルでマークされたリソースを含めるか除外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。たとえば、永続ボリュームクレームリソースを選択し、それに関連付けられたポッドがある場合、Trident Protect は関連付けられている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すると、それらはOR演算として一致し、各要素内のフィールド（グループ、種類、バージョン）はAND演算として一致します。
  - **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レクタ文字列。例えば：`"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用します `<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
```

3. （オプション）復元するアプリケーションの特定のリソースのみを選択する必要がある場合は、特定のラベルでマークされたリソースを含めるか除外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。たとえば、永続ボリュームクレームリソースを選択し、それに関連付けられたポッドがある場合、Trident Protect は関連付けられている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すると、それらはOR演算として一致し、各要素内のフィールド（グループ、種類、バージョン）はAND演算として一致します。
       - **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レクタ文字列。例えば：`"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-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。例：

```
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。例：

```
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）	• annotation.one/key："updatedvalue" • annotation.two/key："true"	• 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）	• annotation.one/key："true" • annotation.three/key："false"	• 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）	• annotation.one/key："updatedvalue" • annotation.two/key："true" • annotation.three/key："false"	• 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、システム内のさまざまな動作を構成するためにサポートされているアノテーションを一覧表示します。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：すべてのリストアCRタイプ（BackupRestore、BackupInplaceRestore、SnapshotRestore、SnapshotInplaceRestore）。ストレージバックエンドまたはクラスターで処理に時間がかかることが多い場合は、より大きい値を使用してください。	「1200」（20分）