还原应用程序

Trident

NetApp
July 01, 2026

目录

还原应用程序

使用 Trident Protect 还原应用程序

您可以使用 Trident Protect 从快照或备份恢复您的应用程序。将应用程序还原到同一集群时，从现有快照还原将更快。

- 还原应用程序时，为应用程序配置的所有执行挂钩都会随应用程序一起还原。如果存在还原后执行挂钩，它将作为还原操作的一部分自动运行。
- qtree 卷支持从备份还原到其他命名空间或原始命名空间。但是，qtree 卷不支持从快照还原到其他命名空间或原始命名空间。
- 您可以使用高级设置自定义还原操作。要了解更多信息，请参阅 "使用高级 Trident Protect 还原设置"。

从备份还原到其他命名空间

使用 BackupRestore CR 将备份还原到其他命名空间时，Trident Protect 会在新命名空间中还原应用程序，并为还原的应用程序创建应用程序 CR。要保护还原的应用程序，请创建按需备份或快照，或建立保护计划。

- 使用现有资源将备份还原到其他命名空间不会更改与备份中的名称共享的任何资源。要还原备份中的所有资源，请删除并重新创建目标命名空间，或将备份还原到新命名空间。
- 使用 CR 还原到新命名空间时，您必须在应用 CR 之前手动创建目标命名空间。Trident Protect 仅在使用 CLI 时自动创建命名空间。

开始之前

确保 AWS 会话令牌过期时间足以进行任何长期运行的 s3 还原操作。如果令牌在还原操作期间过期，则操作可能会失败。

- 有关检查当前会话令牌过期的详细信息，请参见 "AWS API 文档"。
- 有关 AWS 资源凭据的详细信息，请参见 "AWS IAM 文档"。

使用 Kopia 作为数据移动器还原备份时，可以选择在 CR 中指定注释或使用 CLI 控制 Kopia 使用的临时存储的行为。有关可以配置的选项的详细信息，请参见 "Kopia 文档"。有关使用 Trident Protect CLI 指定注释的详细信息，请使用 `tridentctl-protect create --help`命令。

使用 **CR**

步骤

1. 创建自定义资源 (CR) 文件并将其命名为 `trident-protect-backup-restore-cr.yaml`。
2. 在创建的文件中，配置以下属性：
   - **metadata.name**：（*Required*）此自定义资源的名称；为您的环境选择一个唯一且合理的名称。
   - **spec.appArchivePath**：AppVault 中存储备份内容的路径。您可以使用以下命令查找此路径：

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

   - **spec.appVaultRef**：（必需）存储备份内容的 AppVault 的名称。
   - **spec.destinationApplicationName**：（可选）已还原应用程序的名称。如果提供，还原的应用程序将使用此名称。如果未提供，还原的应用程序将使用源应用程序名称。
   - **spec.namespaceMapping**：还原操作的源命名空间到目标命名空间的映射。使用环境中的信息替换 `my-source-namespace`和 `my-destination-namespace`。

     ```
     ---
     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. （可选）如果需要仅选择要还原的应用程序的某些资源，请添加包含或排除标有特定标签的资源的筛选：

   > Trident Protect 会自动选择一些资源，因为它们与您选择的资源之间存在关系。例如，如果您选择了永久卷声明资源，并且它具有关联的 pod，则 Trident Protect 也将还原关联的 pod。

   - **resourceFilter.resourceSelectionCriteria**：（筛选时需要）使用 `Include`或 `Exclude`来包含或排除在 resourceMatchers 中定义的资源。添加以下 resourceMatchers 参数以定义要包括或排除的资源：
     - **resourceFilter.resourceMatchers**：resourceMatcher 对象数组。如果在此数组中定义多个元素，则它们将作为 OR 操作进行匹配，并且每个元素（组、种类、版本）内的字段将作为 AND 操作进行匹配。

- **resourceMatchers[].group**：（*Optional*）要筛选的资源的组。
- **resourceMatchers[].kind**：（*Optional*）要筛选的资源的类型。
- **resourceMatchers[].version**：（*Optional*）要筛选的资源的版本。
- **resourceMatchers[].names**：（可选）要过滤的资源的 Kubernetes metadata.name 字段中的名称。
- **resourceMatchers[].namespaces**: (*Optional*) 要过滤的资源的 Kubernetes metadata.name 字段中的命名空间。
- **resourceMatchers[].labelSelectors**: (*Optional*) 资源的 Kubernetes metadata.name 字段中的标签选择器字符串，如 "Kubernetes 文档" 中所定义。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. 使用正确的值填充 `trident-protect-backup-restore-cr.yaml`文件后，应用 CR：

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

使用 **CLI**

步骤

1. 将备份还原到其他命名空间，用环境中的信息替换括号中的值。 `namespace-mapping` 参数使用冒号分隔的命名空间将源命名空间映射到格式为 `source1:dest1,source2:dest2` 的正确目标命名空间。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>
```

从备份还原到原始命名空间

您可以随时将备份还原到原始命名空间。当您执行就地恢复时，Trident Protect 会自动管理保护计划和正在进行的操作，以防止无效的恢复点：

- 在开始还原之前，将禁用应用程序的所有已启用的保护计划。这会阻止在还原应用程序资源时运行定时备份或快照。
- 成功完成还原后，仅重新启用还原之前启用的计划。已禁用的计划将保持禁用状态。
- 在恢复开始之前，任何正在进行的备份或快照操作都将被取消。如果操作未在 5 分钟内取消，还原将继续，并在还原 CR 状态下记录警告。

开始之前

确保 AWS 会话令牌过期时间足以进行任何长期运行的 s3 还原操作。如果令牌在还原操作期间过期，则操作可能会失败。

- 有关检查当前会话令牌过期的详细信息，请参见 "AWS API 文档"。
- 有关 AWS 资源凭据的详细信息，请参见 "AWS IAM 文档"。

使用 Kopia 作为数据移动器还原备份时，可以选择在 CR 中指定注释或使用 CLI 控制 Kopia 使用的临时存储的行为。有关可以配置的选项的详细信息，请参见 "Kopia 文档"。有关使用 Trident Protect CLI 指定注释的详细信息，请使用 `tridentctl-protect create --help`命令。

使用 **CR**

步骤

1. 创建自定义资源 (CR) 文件并将其命名为 `trident-protect-backup-ipr-cr.yaml`。
2. 在创建的文件中，配置以下属性：
   - **metadata.name**：（*Required*）此自定义资源的名称；为您的环境选择一个唯一且合理的名称。
   - **spec.appArchivePath**：AppVault 中存储备份内容的路径。您可以使用以下命令查找此路径：

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

   - **spec.appVaultRef**：（必需）存储备份内容的 AppVault 的名称。

     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. （可选）如果需要仅选择要还原的应用程序的某些资源，请添加包含或排除标有特定标签的资源的筛选：

   Trident Protect 会自动选择一些资源，因为它们与您选择的资源之间存在关系。例如，如果您选择了永久卷声明资源，并且它具有关联的 pod，则 Trident Protect 也将还原关联的 pod。

   - **resourceFilter.resourceSelectionCriteria**：（筛选时需要）使用 `Include`或 `Exclude`来包含或排除在 resourceMatchers 中定义的资源。添加以下 resourceMatchers 参数以定义要包括或排除的资源：
     - **resourceFilter.resourceMatchers**：resourceMatcher 对象数组。如果在此数组中定义多个元素，则它们将作为 OR 操作进行匹配，并且每个元素（组、种类、版本）内的字段将作为 AND 操作进行匹配。
       - **resourceMatchers[].group**：（*Optional*）要筛选的资源的组。
       - **resourceMatchers[].kind**：（*Optional*）要筛选的资源的类型。
       - **resourceMatchers[].version**：（*Optional*）要筛选的资源的版本。
       - **resourceMatchers[].names**：（可选）要过滤的资源的 Kubernetes metadata.name 字段

中的名称。

- **resourceMatchers[].namespaces**: (*Optional*) 要过滤的资源的 Kubernetes metadata.name 字段中的命名空间。
- **resourceMatchers[].labelSelectors**: (*Optional*) 资源的 Kubernetes metadata.name 字段中的标签选择器字符串，如 "Kubernetes 文档" 中所定义。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. 使用正确的值填充 `trident-protect-backup-ipr-cr.yaml`文件后，应用 CR：

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

使用 **CLI**

步骤

1. 将备份还原到原始命名空间，用环境中的信息替换括号中的值。 `backup` 参数使用格式为 `<namespace>/<name>` 的命名空间和备份名称。例如：

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

从备份还原到其他集群

如果原始群集出现问题，可以将备份还原到其他群集。

- 使用 Kopia 作为数据移动器还原备份时，可以选择在 CR 中指定注释或使用 CLI 控制 Kopia 使用的临时存储的行为。有关可以配置的选项的详细信息，请参见 "Kopia 文档"。有关使用 Trident Protect CLI 指定注释的详细信息，请使用 `tridentctl-protect create --help`命令。
- 使用 CR 还原到新命名空间时，您必须在应用 CR 之前手动创建目标命名空间。Trident Protect 仅在使用 CLI 时自动创建命名空间。

开始之前

确保满足以下先决条件：

- 目标集群已安装 Trident Protect。
- 目标集群可以访问与源集群相同的 AppVault 存储桶路径，备份存储在该路径中。
- 运行 `tridentctl-protect get appvaultcontent` 命令时，确保您的本地环境可以连接到 AppVault CR 中定义的对象存储桶。如果网络限制阻止访问，请在目标集群的 pod 内运行 Trident Protect CLI。
- 确保 AWS 会话令牌过期时间足以进行任何长期运行的还原操作。如果令牌在还原操作期间过期，则操作可能会失败。
  - 有关检查当前会话令牌过期的详细信息，请参见 "AWS API 文档"。
  - 有关 AWS 资源凭据的详细信息，请参见 "AWS 文档"。

步骤

1. 使用 Trident Protect CLI 插件验证目标集群上是否存在 AppVault CR：

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

   如果目标集群上不存在 AppVault CR，请按照 "使用 Trident Protect AppVault 对象管理存储桶" 中的步骤创建它。

2. 查看目标集群上可用的 AppVault 的备份内容，并记录 `appArchivePath`要还原的备份：

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

   运行此命令会显示 AppVault 中的可用备份，包括其原始群集、相应的应用程序名称、时间戳和存档路径。

   输出示例：

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   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 名称和存档路径将应用程序还原到目标集群：

> 使用 CR 时，请确保目标集群上存在用于应用程序还原的命名空间。

使用 **CR**

1. 创建自定义资源 (CR) 文件并将其命名为 `trident-protect-backup-restore-cr.yaml`。
2. 在创建的文件中，配置以下属性：
   - **metadata.name**：（*Required*）此自定义资源的名称；为您的环境选择一个唯一且合理的名称。
   - **spec.appVaultRef**：（必需）存储备份内容的 AppVault 的名称。
   - **spec.appArchivePath**: (*Required*) AppVault 内存储备份内容的路径。使用步骤 2 中的命令查看备份内容并找到 `appArchivePath`要还原的备份。
   - **spec.destinationApplicationName**：（可选）已还原应用程序的名称。如果提供，还原的应用程序将使用此名称。如果未提供，还原的应用程序将使用源应用程序名称。
   - **spec.namespaceMapping**：还原操作的源命名空间到目标命名空间的映射。使用环境中的信息替换 `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
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. 使用正确的值填充 `trident-protect-backup-restore-cr.yaml`文件后，应用 CR：

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

使用 **CLI**

1. 使用以下命令还原应用程序，用环境中的信息替换括号中的值。namespace-mapping 参数使用冒号分隔的命名空间将源命名空间映射到格式为 source1:dest1,source2:dest2 的正确目标命名空间。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>
```

从快照还原到其他命名空间

您可以使用自定义资源 (CR) 文件将数据从快照还原到其他命名空间或原始源命名空间。使用 SnapshotRestore CR 将快照还原到其他命名空间时，Trident Protect 会在新命名空间中还原应用程序，并为还原的应用程序创建应用程序 CR。要保护还原的应用程序，请创建按需备份或快照，或建立保护计划。

- SnapshotRestore 支持 `spec.storageClassMapping`属性，但仅当源和目标存储类使用相同的存储后端时。如果尝试还原到使用不同存储后端的 `StorageClass`，还原操作将失败。
- 使用 CR 还原到新命名空间时，您必须在应用 CR 之前手动创建目标命名空间。Trident Protect 仅在使用 CLI 时自动创建命名空间。

开始之前

确保 AWS 会话令牌过期时间足以进行任何长期运行的 s3 还原操作。如果令牌在还原操作期间过期，则操作可能会失败。

- 有关检查当前会话令牌过期的详细信息，请参见 "AWS API 文档"。
- 有关 AWS 资源凭据的详细信息，请参见 "AWS IAM 文档"。

使用 **CR**

步骤

1. 创建自定义资源 (CR) 文件并将其命名为 `trident-protect-snapshot-restore-cr.yaml`。
2. 在创建的文件中，配置以下属性：
   - **metadata.name**：（*Required*）此自定义资源的名称；为您的环境选择一个唯一且合理的名称。
   - **spec.appVaultRef**：（必需）存储快照内容的 AppVault 的名称。
   - **spec.appArchivePath**：AppVault 中存储快照内容的路径。您可以使用以下命令查找此路径：

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

   - **spec.destinationApplicationName**：（可选）已还原应用程序的名称。如果提供，还原的应用程序将使用此名称。如果未提供，还原的应用程序将使用源应用程序名称。
   - **spec.namespaceMapping**：还原操作的源命名空间到目标命名空间的映射。使用环境中的信息替换 `my-source-namespace`和 `my-destination-namespace`。

     ```
     ---
     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. （可选）如果需要仅选择要还原的应用程序的某些资源，请添加包含或排除标有特定标签的资源的筛选：

   > Trident Protect 会自动选择一些资源，因为它们与您选择的资源之间存在关系。例如，如果您选择了永久卷声明资源，并且它具有关联的 pod，则 Trident Protect 也将还原关联的 pod。

   - **resourceFilter.resourceSelectionCriteria**：（筛选时需要）使用 `Include`或 `Exclude`来包含或排除在 resourceMatchers 中定义的资源。添加以下 resourceMatchers 参数以定义要包括或排除的资源：
     - **resourceFilter.resourceMatchers**：resourceMatcher 对象数组。如果在此数组中定义多个元素，则它们将作为 OR 操作进行匹配，并且每个元素（组、种类、版本）内的字段将作为 AND 操作进行匹配。
       - **resourceMatchers[].group**：（*Optional*）要筛选的资源的组。

- **resourceMatchers[].kind**：（*Optional*）要筛选的资源的类型。
- **resourceMatchers[].version**：（*Optional*）要筛选的资源的版本。
- **resourceMatchers[].names**：（可选）要过滤的资源的 Kubernetes metadata.name 字段中的名称。
- **resourceMatchers[].namespaces**: (*Optional*) 要过滤的资源的 Kubernetes metadata.name 字段中的命名空间。
- **resourceMatchers[].labelSelectors**: (*Optional*) 资源的 Kubernetes metadata.name 字段中的标签选择器字符串，如 "Kubernetes 文档" 中所定义。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. 使用正确的值填充 `trident-protect-snapshot-restore-cr.yaml`文件后，应用 CR：

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

使用 CLI

步骤

1. 将快照还原到其他命名空间，用环境中的信息替换括号中的值。
   - `snapshot` 参数使用格式为 `<namespace>/<name>` 的命名空间和快照名称。
   - `namespace-mapping` 参数使用冒号分隔的命名空间将源命名空间映射到格式为 `source1:dest1,source2:dest2` 的正确目标命名空间。

     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>
```

从快照还原到原始命名空间

您可以随时将快照还原到原始命名空间。当您执行就地恢复时，Trident Protect 会自动管理保护计划和正在进行的操作，以防止无效的恢复点：

- 在开始还原之前，将禁用应用程序的所有已启用的保护计划。这会阻止在还原应用程序资源时运行定时备份或快照。
- 成功完成还原后，仅重新启用还原之前启用的计划。已禁用的计划将保持禁用状态。
- 在恢复开始之前，任何正在进行的备份或快照操作都将被取消。如果操作未在 5 分钟内取消，还原将继续，并在还原 CR 状态下记录警告。

开始之前

确保 AWS 会话令牌过期时间足以进行任何长期运行的 s3 还原操作。如果令牌在还原操作期间过期，则操作可能会失败。

- 有关检查当前会话令牌过期的详细信息，请参见 "AWS API 文档"。
- 有关 AWS 资源凭据的详细信息，请参见 "AWS IAM 文档"。

使用 **CR**

步骤

1. 创建自定义资源 (CR) 文件并将其命名为 `trident-protect-snapshot-ipr-cr.yaml`。
2. 在创建的文件中，配置以下属性：
   - **metadata.name**：（*Required*）此自定义资源的名称；为您的环境选择一个唯一且合理的名称。
   - **spec.appVaultRef**：（必需）存储快照内容的 AppVault 的名称。
   - **spec.appArchivePath**：AppVault 中存储快照内容的路径。您可以使用以下命令查找此路径：

```
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. （可选）如果需要仅选择要还原的应用程序的某些资源，请添加包含或排除标有特定标签的资源的筛选：

> Trident Protect 会自动选择一些资源，因为它们与您选择的资源之间存在关系。例如，如果您选择了永久卷声明资源，并且它具有关联的 pod，则 Trident Protect 也将还原关联的 pod。

   - **resourceFilter.resourceSelectionCriteria**：（筛选时需要）使用 `Include`或 `Exclude`来包含或排除在 resourceMatchers 中定义的资源。添加以下 resourceMatchers 参数以定义要包括或排除的资源：
     - **resourceFilter.resourceMatchers**：resourceMatcher 对象数组。如果在此数组中定义多个元素，则它们将作为 OR 操作进行匹配，并且每个元素（组、种类、版本）内的字段将作为 AND 操作进行匹配。
       - **resourceMatchers[].group**：（*Optional*）要筛选的资源的组。
       - **resourceMatchers[].kind**：（*Optional*）要筛选的资源的类型。
       - **resourceMatchers[].version**：（*Optional*）要筛选的资源的版本。
       - **resourceMatchers[].names**：（可选）要过滤的资源的 Kubernetes metadata.name 字段中的名称。
       - **resourceMatchers[].namespaces**: (*Optional*) 要过滤的资源的 Kubernetes metadata.name 字段中的命名空间。

- **resourceMatchers[].labelSelectors**: (*Optional*) 资源的 Kubernetes metadata.name 字段中的标签选择器字符串，如 "Kubernetes 文档" 中所定义。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. 使用正确的值填充 `trident-protect-snapshot-ipr-cr.yaml`文件后，应用 CR：

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

使用 CLI

步骤

1. 将快照还原到原始命名空间，用环境中的信息替换括号中的值。例如：

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

检查还原操作的状态

您可以使用命令行检查正在进行、已完成或已失败的还原操作的状态。

步骤

1. 使用以下命令可检索还原操作的状态，用环境中的信息替换括号中的值：

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

使用高级 Trident Protect 还原设置

您可以使用高级设置（如注释、命名空间设置和存储选项）自定义还原操作，以满足您的特定要求。

还原和故障转移操作期间的命名空间注释和标签

在恢复和故障转移操作期间，目标命名空间中的标签和注释将与源命名空间中的标签和注释匹配。将添加目标命名空间中不存在的源命名空间中的标签或注释，并覆盖已存在的任何标签或注释，以匹配源命名空间中的值。仅存在于目标命名空间上的标签或注释保持不变。

如果使用 Red Hat OpenShift，请务必注意命名空间注释在 OpenShift 环境中的关键作用。命名空间注释可确保还原的 Pod 遵守 OpenShift 安全上下文约束 (SCC) 定义的适当权限和安全配置，并且可以访问卷而不会出现权限问题。有关详细信息，请参见 "OpenShift 安全上下文约束文档"。

在执行还原或故障转移操作之前，可以通过设置 Kubernetes 环境变量 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`来防止目标命名空间中的特定注释被覆盖。例如：

```
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
```

执行还原或故障切换操作时，在 `restoreSkipNamespaceAnnotations`和 `restoreSkipNamespaceLabels`中指定的任何命名空间批注和标签都将从还原或故障切换操作中排除。确保在初始 Helm 安装过程中配置了这些设置。要了解更多信息，请参阅 "配置其他 Trident Protect helm 图表设置"。

如果使用带有 `--create-namespace` 标志的 Helm 安装源应用程序，则会对 `name` 标签键进行特殊处理。在还原或故障转移过程中，Trident Protect 会将此标签复制到目标命名空间，但如果来自源的值与源命名空间匹配，则将值更新为目标命名空间值。如果此值与源命名空间不匹配，则将其复制到目标命名空间，不进行任何更改。

示例

以下示例显示了源和目标命名空间，每个命名空间都有不同的注释和标签。您可以查看操作之前和之后的目标命名空间的状态，以及如何在目标命名空间中组合或覆盖注释和标签。

还原或故障转移操作之前

下表显示了还原或故障转移操作之前的示例源和目标命名空间的状态：

命名空间	标注	标签
命名空间 ns-1（源）	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • 合规性=hipaa • name=ns-1
命名空间 ns-2（目标）	• annotation.one/key: "true" • annotation.three/key: "false"	• 角色=数据库

还原操作后

下表显示了还原或故障转移操作后示例目标命名空间的状态。已添加一些键，一些键已被覆盖，并且已更新 `name` 标签以匹配目标命名空间：

命名空间	标注	标签
命名空间 ns-2（目标）	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • 合规性=hipaa • environment=production • 角色=数据库

支持的字段

本节描述可用于还原操作的其他字段。

存储类映射

该 `spec.storageClassMapping` 属性定义从源应用程序中存在的存储类到目标集群上的新存储类的映射。您可以在具有不同存储类的集群之间迁移应用程序或更改 BackupRestore 操作的存储后端时使用此功能。

示例：

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

支持的注释

本节列出了用于在系统中配置各种行为的支持注释。如果用户未明确设置注释，系统将使用默认值。

标注	类型	说明	默认值
protect.trident.netapp.io/data-mover-timeout-sec	string	允许数据移动器操作停止的最长时间（以秒为单位）。	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	string	Kopia 内容缓存的最大大小限制（以兆字节为单位）。	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	string	在操作失败之前，等待任何新创建的 PersistentVolumeClaims (PVC) 到达 `Bound`阶段的最长时间（以秒为单位）。适用于所有还原 CR 类型（BackupRestore、BackupInplaceRestore、SnapshotRestore、SnapshotInplaceRestore）。如果您的存储后端或集群通常需要更多时间，请使用更高的值。	"1200"（20 分钟）