



还原应用程序 Trident

NetApp
October 29, 2026

目录

- 还原应用程序 1
 - 使用 Trident Protect 还原应用程序 1
 - 从备份还原到其他命名空间 1
 - 从备份还原到原始命名空间 4
 - 从备份还原到其他集群 6
 - 从快照还原到其他命名空间 8
 - 从快照还原到原始命名空间 11
 - 检查还原操作的状态 13
 - 使用高级 Trident Protect 还原设置 13
 - 还原和故障转移操作期间的命名空间注释和标签 13
 - 支持的字段 15
 - 支持的注释 15
 - 使用自定义资源还原资源时修改资源 16
 - 资源修改的工作原理 16
 - 支持的资源 16
 - 支持的操作 17

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"](#)。
- 要从存档还原特定 KubeVirt 虚拟机 (VM) 而不还原所有命名空间资源，请使用还原 CR 上的 ``includedVirtualMachines`` 字段。有关支持的 CR、字段引用和 YAML 示例，请参阅 ["使用 Trident Protect 保护 KubeVirt 虚拟机"](#)。

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. (Optional) 如果仅需要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, 如果您选择了持久卷声明资源, 且该资源具有关联的 pod, 则 Trident Protect 也将还原该关联的 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 操作匹配, 并且每个元素 (`group`、`kind`、`version`) 内的字段将作为 AND 操作匹配。
 - **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. (Optional) 如果仅需要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，如果您选择了持久卷声明资源，且该资源具有关联的 pod，则 Trident Protect 也将还原该关联的 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 操作匹配，并且每个元素（group、kind、version）内的字段将作为 AND 操作匹配。

- **resourceMatchers[].group**：(Optional) 要筛选的资源的组。
- **resourceMatchers[].kind**：(Optional) 要筛选的资源类型。
- **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

如果原始集群出现问题，可以将备份还原到其他集群。



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

3. (Optional) 如果仅需要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，如果您选择了持久卷声明资源，且该资源具有关联的 pod，则 Trident Protect 也将还原该关联的 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 操作匹配，并且每个元素 (group、kind、version) 内的字段将作为 AND 操作匹配。
 - **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
```

+ *(Optional)* 如果仅需要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：

+ 注意：Trident Protect 会自动选择某些资源，因为这些资源与您所选资源之间存在关联关系。例如，如果您选择了持久卷声明资源，且该资源具有关联的 pod，则 Trident Protect 也将还原该关联的 pod。

+ * **resourceFilter.resourceSelectionCriteria**: (筛选必需) 使用 `Include` 或 `Exclude` 以包含或排除在 `resourceMatchers` 中定义的资源。添加以下 `resourceMatchers` 参数以定义要包含或排除的资源： **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` 操作匹配，并且每个元素 (`group`、`kind`、`version`) 内的字段将作为 `AND` 操作匹配。* **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**: (*Optional*) 要筛选的资源的 `Kubernetes metadata.name` 字段中的名称。* **resourceMatchers[].namespaces**: (*Optional*) 要筛选的资源的 `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"]
```

+ .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_key_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）	<ul style="list-style-type: none">• annotation.one/key: "updatedvalue"• annotation.two/key: "true"	<ul style="list-style-type: none">• environment=production• 合规性=hipaa• name=ns-1
命名空间 ns-2（目标）	<ul style="list-style-type: none">• annotation.one/key: "true"• annotation.three/key: "false"	<ul style="list-style-type: none">• 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)	<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 • 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）

使用自定义资源还原资源时修改资源

资源转换使您能够在还原资源时对其进行修改。这在还原版本需要与原始版本不同时非常有用，例如，在将应用程序还原到其他集群时更新 ConfigMap 中的主机名。

资源修改的工作原理

``SnapshotRestore``、``BackupRestore``、``AppMirrorRelationship`` 以及其他恢复资源中的 ``transformations`` 字段允许您在恢复过程中修改 Kubernetes 资源。当需要对恢复后的应用程序进行与原始应用程序不同的配置时（例如，更改主机名、注册表 URL、资源限制或环境变量），此功能非常有用。

资源转换使用 "RFC 6902 JSON Patch" 操作和 "RFC 6901 JSON 指针" 路径来定位和修改 Kubernetes 资源中的特定字段。

下面显示了包含资源转换的还原对象的基本结构：

```
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)
```

支持的资源

您可以将资源转换与符合以下标准的资源一起使用：

- **kind** (必填) : Kubernetes 资源类型 (例如, ConfigMap、Deployment、Pod)
- **group** (可选) : API 组 (例如, apps、route.openshift.io) ——省略核心资源
- **version** (可选) : API 版本 (例如, v1、v1beta1)
- 名称 (可选) : 仅按名称应用于特定资源



当前不支持修改 PersistentVolumeClaims 和命名空间。

支持的操作

您可以使用以下操作修改资源: add、copy、move、remove、replace`和`test。

向资源添加值

使用 add 操作在指定路径上添加新字段或值。您可以将数据添加到对象或数组。

使用 CR

以下示例将节点选择器添加到`Deployment`资源:

```
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

使用以下命令通过命令行运行此转换:

```
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"}}'
```

复制资源中的值

使用 copy 操作将同一资源中的值从一个路径复制到另一个路径。源未更改。

使用 CR

以下示例复制 ConfigMap 对象的数据键：

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

使用 CLI

使用以下命令通过命令行运行此转换：

```
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"}'
```

在资源内移动值

使用 move 操作将同一资源内的值从一个路径移动到另一个路径。源被删除，值被放置在目的地。

使用 CR

以下示例重命名 ConfigMap 对象的数据键：

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

使用 CLI

使用以下命令通过命令行运行此转换：

```
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"}'
```

从资源中删除值

使用 `remove` 操作可删除指定路径中的字段或值。

使用 CR

以下示例从 `ConfigMap` 资源中删除批注：

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



在以上示例的路径中，`~1` 是 `/` 的 JSON 指针转义序列。

使用 CLI

使用以下命令通过命令行运行此转换：

```
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"}'
```

替换资源中的值

使用该 `replace` 操作替换指定路径下资源中的现有值。JSON 路径必须已存在。

使用 CR

以下示例更改 `Route` 对象的主机名：

```
transformations:
- resource:
```

```

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

```

使用 CLI

使用以下命令通过命令行运行此转换：

```

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"}'

```

测试资源修改

使用 `test` 操作验证路径上的值是否与预期值匹配，然后再应用其他操作。如果测试失败，则回滚整个修改。

使用 CR

仅当 environment 设置为 staging 时，以下示例才会更新 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

使用以下命令通过命令行运行此转换：

```

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"} '
```

版权信息

版权所有 © 2026 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本文档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 <http://www.netapp.com/TM> 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。