



애플리케이션을 관리하고 보호합니다

Trident

NetApp
October 07, 2025

목차

애플리케이션을 관리하고 보호합니다	1
Trident Protect AppVault 객체를 사용하여 버킷을 관리합니다	1
AppVault 인증 및 암호를 구성합니다	1
AppVault 생성 예	5
AppVault 정보를 봅니다	12
AppVault를 제거합니다	13
Trident Protect로 관리할 애플리케이션을 정의합니다	14
AppVault CR을 작성합니다	14
응용 프로그램을 정의합니다	14
Trident Protect를 사용하여 애플리케이션을 보호합니다	18
필요 시 스냅샷을 생성합니다	18
필요 시 백업을 생성합니다	21
데이터 보호 스케줄을 생성합니다	23
스냅샷을 삭제합니다	26
백업을 삭제합니다	26
백업 작업의 상태를 확인합니다	27
Azure-NetApp-files(ANF) 작업을 위한 백업 및 복원이 가능합니다	27
응용 프로그램을 복원합니다	28
Trident Protect를 사용하여 애플리케이션을 복원합니다	28
고급 Trident 보호 복원 설정 사용	44
NetApp SnapMirror 및 Trident Protect를 사용하여 애플리케이션을 복제합니다	46
복원 및 페일오버 작업 중 네임스페이스 주석 및 레이블	46
장애 조치 및 역방향 작업 중 실행 후크	47
복제 관계를 설정합니다	48
애플리케이션 복제 방향을 반대로 전환합니다	57
Trident Protect를 사용하여 애플리케이션을 마이그레이션합니다	60
백업 및 복원 작업	60
한 스토리지 클래스에서 다른 스토리지 클래스로 애플리케이션을 마이그레이션합니다	61
Trident Protect 실행 후크를 관리합니다	64
실행 후크 유형	64
사용자 정의 실행 후크에 대한 중요 참고 사항	65
실행 후크 필터	65
실행 후크 예	66
실행 후크를 만듭니다	66
실행 후크를 수동으로 실행합니다	69

애플리케이션을 관리하고 보호합니다

Trident Protect AppVault 객체를 사용하여 버킷을 관리합니다

Trident Protect의 버킷 사용자 지정 리소스(CR)를 AppVault라고 합니다. AppVault 개체는 스토리지 버킷의 선언적 Kubernetes 워크플로우 표현입니다. AppVault CR에는 백업, 스냅샷, 복원 작업 및 SnapMirror 복제와 같은 보호 작업에 사용되는 버킷에 필요한 구성이 포함되어 있습니다. 관리자만 AppVaults를 만들 수 있습니다.

애플리케이션에서 데이터 보호 작업을 수행할 때는 AppVault CR을 수동으로 또는 명령줄에서 생성해야 합니다. AppVault CR은 사용자 환경에 따라 다르므로, 이 페이지의 예시를 참고하여 AppVault CR을 생성할 수 있습니다.



Trident Protect가 설치된 클러스터에 AppVault CR이 있는지 확인하세요. AppVault CR이 없거나 액세스할 수 없는 경우 명령줄에 오류가 표시됩니다.

AppVault 인증 및 암호를 구성합니다

AppVault CR을 생성하기 전에 AppVault와 선택한 데이터 이동자가 공급자 및 관련 리소스에 대해 인증할 수 있는지 확인하세요.

Data Mover 저장소 암호입니다

CR 또는 Trident Protect CLI 플러그인을 사용하여 AppVault 객체를 생성할 때 Restic 및 Kopia 암호화를 위한 사용자 지정 비밀번호와 함께 Kubernetes 비밀번호를 지정할 수 있습니다. 비밀번호를 지정하지 않으면 Trident Protect는 기본 비밀번호를 사용합니다.

- AppVault CR을 수동으로 생성할 때 **spec.dataMoverPasswordSecretRef** 필드를 사용하여 비밀번호를 지정합니다.
- Trident Protect CLI를 사용하여 AppVault 객체를 생성할 때 다음을 사용하십시오. `--data-mover-password -secret-ref` 비밀을 지정하는 인수입니다.

Data Mover 리포지토리 암호 암호를 생성합니다

다음 예제를 사용하여 암호 암호를 만듭니다. AppVault 개체를 생성할 때 Trident Protect에서 이 암호를 사용하여 Data Mover 리포지토리를 인증하도록 지시할 수 있습니다.



- 사용 중인 Data Mover에 따라 해당 Data Mover에 대한 암호만 포함하면 됩니다. 예를 들어, Restic을 사용하고 있고 나중에 Kopia를 사용할 계획이 없는 경우 암호를 만들 때 Restic 암호만 포함할 수 있습니다.
- 비밀번호를 안전한 곳에 보관하세요. 같은 클러스터나 다른 클러스터에서 데이터를 복원할 때 필요합니다. 클러스터 또는 `trident-protect` 네임스페이스가 삭제되면 비밀번호 없이는 백업이나 스냅샷을 복원할 수 없습니다.

CR을 사용합니다

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

CLI를 사용합니다

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

S3 호환 스토리지 IAM 권한

Amazon S3, Generic S3 등 S3 호환 스토리지에 접속할 경우 "[StorageGRID S3](#)", 또는 "[ONTAP S3](#) 를 참조하십시오" Trident Protect를 사용하는 경우, 제공하는 사용자 자격 증명에 버킷 액세스에 필요한 권한이 있는지 확인해야 합니다. 다음은 Trident Protect를 사용하여 액세스하는 데 필요한 최소 권한을 부여하는 정책의 예입니다. 이 정책은 S3 호환 버킷 정책을 관리하는 사용자에게 적용할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

Amazon S3 정책에 대한 자세한 내용은 다음 예제를 참조하세요. "[Amazon S3 문서](#)".

Amazon S3(AWS) 인증을 위한 EKS Pod ID

Trident Protect는 Kopia 데이터 이동 작업을 위한 EKS Pod Identity를 지원합니다. 이 기능을 사용하면 Kubernetes 비밀에 AWS 자격 증명을 저장하지 않고도 S3 버킷에 안전하게 액세스할 수 있습니다.

- Trident 프로텍트가 포함된 EKS 포드 아이덴티티에 대한 요구 사항*

Trident Protect와 함께 EKS Pod Identity를 사용하기 전에 다음 사항을 확인하세요.

- EKS 클러스터에 Pod Identity가 활성화되어 있습니다.
- 필요한 S3 버킷 권한이 있는 IAM 역할을 생성했습니다. 자세한 내용은 다음을 참조하세요. "[S3 호환 스토리지 IAM 권한](#)".
- IAM 역할은 다음 Trident Protect 서비스 계정과 연결됩니다.
 - <trident-protect>-controller-manager
 - <trident-protect>-resource-backup
 - <trident-protect>-resource-restore
 - <trident-protect>-resource-delete

Pod Identity를 활성화하고 IAM 역할을 서비스 계정과 연결하는 방법에 대한 자세한 지침은 다음을 참조하세요. "[AWS EKS Pod Identity 문서](#)".

AppVault 구성 EKS Pod Identity를 사용하는 경우 AppVault CR을 다음과 같이 구성합니다. `useIAM: true` 명시적 자격 증명 대신 플래그:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

클라우드 공급자를 위한 AppVault 키 생성 예

AppVault CR을 정의할 때 IAM 인증을 사용하지 않는 한 공급자가 호스팅하는 리소스에 액세스하기 위한 자격 증명을 포함해야 합니다. 자격 증명에 대한 키를 생성하는 방법은 공급자에 따라 다릅니다. 다음은 여러 공급자에 대한 명령줄 키 생성 예입니다. 다음 예를 사용하여 각 클라우드 공급자의 자격 증명에 대한 키를 생성할 수 있습니다.

Google 클라우드

```
kubectl create secret generic <secret-name> \  
--from-file=credentials=<mycreds-file.json> \  
-n trident-protect
```

Amazon S3(AWS)

```
kubectl create secret generic <secret-name> \  
--from-literal=accessKeyID=<objectstorage-accesskey> \  
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket  
-secret> \  
-n trident-protect
```

Microsoft Azure를 참조하십시오

```
kubectl create secret generic <secret-name> \  
--from-literal=accountKey=<secret-name> \  
-n trident-protect
```

일반 S3

```
kubectl create secret generic <secret-name> \  
--from-literal=accessKeyID=<objectstorage-accesskey> \  
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket  
-secret> \  
-n trident-protect
```

ONTAP S3 를 참조하십시오

```
kubectl create secret generic <secret-name> \  
--from-literal=accessKeyID=<objectstorage-accesskey> \  
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket  
-secret> \  
-n trident-protect
```

StorageGRID S3

```
kubectl create secret generic <secret-name> \  
--from-literal=accessKeyID=<objectstorage-accesskey> \  
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src  
-bucket-secret> \  
-n trident-protect
```

AppVault 생성 예

다음은 각 공급자에 대한 AppVault 정의의 예입니다.

AppVault CR 예

다음 CR 예제를 사용하여 각 클라우드 공급자에 대한 AppVault 개체를 만들 수 있습니다.



- 선택적으로 Restic 및 Kopia 저장소 암호화에 대한 사용자 지정 암호가 포함된 Kubernetes 암호를 지정할 수 있습니다. 자세한 내용은 [Data Mover 저장소 암호입니다](#) 참조하십시오.
- Amazon S3(AWS) AppVault 객체의 경우 선택적으로 sessionToken을 지정할 수 있습니다. 이는 인증에 SSO(Single Sign-On)를 사용하는 경우에 유용합니다. 이 토큰은 예제 공급자에 대한 키를 생성할 때 [클라우드 공급자를 위한 AppVault 키 생성 예](#) 생성됩니다.
- S3 AppVault 객체의 경우 선택적으로 키를 사용하여 아웃바운드 S3 트래픽에 대한 송신 프록시 URL을 지정할 수 `spec.providerConfig.S3.proxyURL` 있습니다.

Google 클라우드

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3(AWS)


```

---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret

```



Kopia 데이터 무버와 함께 Pod Identity를 사용하는 EKS 환경의 경우 다음을 제거할 수 있습니다. providerCredentials 섹션을 추가하고 추가하세요 useIAM: true 아래에 s3 대신 구성을 선택하세요.

Microsoft Azure를 참조하십시오

```

apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret

```

일반 S3

```

apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret

```

ONTAP S3 를 참조하십시오

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

StorageGRID S3

```

apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret

```

Trident Protect CLI를 사용한 AppVault 생성 예

다음 CLI 명령 예제를 사용하여 각 공급자에 대해 AppVault CRS를 만들 수 있습니다.



- 선택적으로 Restic 및 Kopia 저장소 암호화에 대한 사용자 지정 암호가 포함된 Kubernetes 암호를 지정할 수 있습니다. 자세한 내용은 [Data Mover 저장소 암호입니다](#) 참조하십시오.
- S3 AppVault 개체의 경우 선택적으로 인수를 사용하여 아웃바운드 S3 트래픽에 대한 송신 프록시 URL을 지정할 수 --proxy-url <ip_address:port> 있습니다.

Google 클라우드

```
tridentctl-protect create vault GCP <vault-name> \  
--bucket <mybucket> \  
--project <my-gcp-project> \  
--secret <secret-name>/credentials \  
--data-mover-password-secret-ref <my-optional-data-mover-secret> \  
-n trident-protect
```

Amazon S3(AWS)

```
tridentctl-protect create vault AWS <vault-name> \  
--bucket <bucket-name> \  
--secret <secret-name> \  
--endpoint <s3-endpoint> \  
--data-mover-password-secret-ref <my-optional-data-mover-secret> \  
-n trident-protect
```

Microsoft Azure를 참조하십시오

```
tridentctl-protect create vault Azure <vault-name> \  
--account <account-name> \  
--bucket <bucket-name> \  
--secret <secret-name> \  
--data-mover-password-secret-ref <my-optional-data-mover-secret> \  
-n trident-protect
```

일반 S3

```
tridentctl-protect create vault GenericS3 <vault-name> \  
--bucket <bucket-name> \  
--secret <secret-name> \  
--endpoint <s3-endpoint> \  
--data-mover-password-secret-ref <my-optional-data-mover-secret> \  
-n trident-protect
```

ONTAP S3 를 참조하십시오

```
tridentctl-protect create vault OntapS3 <vault-name> \  
--bucket <bucket-name> \  
--secret <secret-name> \  
--endpoint <s3-endpoint> \  
--data-mover-password-secret-ref <my-optional-data-mover-secret> \  
-n trident-protect
```

StorageGRID S3

```
tridentctl-protect create vault StorageGridS3 <vault-name> \  
--bucket <bucket-name> \  
--secret <secret-name> \  
--endpoint <s3-endpoint> \  
--data-mover-password-secret-ref <my-optional-data-mover-secret> \  
-n trident-protect
```

AppVault 정보를 봅니다

Trident Protect CLI 플러그인을 사용하여 클러스터에서 생성한 AppVault 개체에 대한 정보를 볼 수 있습니다.

단계

1. AppVault 개체의 내용을 봅니다.

```
tridentctl-protect get appvaultcontent gcp-vault \  
--show-resources all \  
-n trident-protect
```

◦ 출력 예 *:

```

+-----+-----+-----+-----+
+-----+
| CLUSTER | APP | TYPE | NAME |
|-----|
|-----+-----+-----+-----+
+-----+
|          | mysql | snapshot | mysnap | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup | mybackup5 | 2024-
08-09 22:25:13 (UTC) |
|          | mysql | backup | mybackup | 2024-
08-09 21:02:52 (UTC) |
+-----+-----+-----+-----+
+-----+

```

2. 선택적으로, 각 리소스의 AppVaultPath를 보려면 플래그를 `--show-paths` 사용합니다.

테이블의 첫 번째 열에 있는 클러스터 이름은 Trident Protect Helm 설치에 클러스터 이름이 지정된 경우에만 사용할 수 있습니다. 예를 들면 다음과 `--set clusterName=production1` 같습니다.

AppVault를 제거합니다

언제든지 AppVault 개체를 제거할 수 있습니다.



AppVault 개체를 삭제하기 전에 AppVault CR에서 키를 제거하지 `finalizers` 마십시오. 이렇게 하면 AppVault 버킷의 잔여 데이터와 클러스터의 분리된 리소스가 생성될 수 있습니다.

시작하기 전에

삭제하려는 AppVault에서 사용 중인 모든 스냅샷 및 백업 CRS를 삭제했는지 확인합니다.

Kubernetes CLI를 사용하여 AppVault를 제거합니다

1. AppVault 개체를 제거하고 `appvault-name` 제거할 AppVault 개체의 이름으로 바꿉니다.

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Trident Protect CLI를 사용하여 AppVault를 제거합니다

1. AppVault 개체를 제거하고 `appvault-name` 제거할 AppVault 개체의 이름으로 바꿉니다.

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Trident Protect로 관리할 애플리케이션을 정의합니다

애플리케이션 CR 및 연결된 AppVault CR을 생성하여 Trident Protect로 관리할 애플리케이션을 정의할 수 있습니다.

AppVault CR을 작성합니다

응용 프로그램에서 데이터 보호 작업을 수행할 때 사용할 AppVault CR을 만들어야 하며 Trident Protect가 설치된 클러스터에 AppVault CR이 있어야 합니다. AppVault CR은 사용자 환경에 따라 다릅니다. AppVault CRS의 예는 을 참조하십시오 "[AppVault 사용자 지정 리소스](#)."

응용 프로그램을 정의합니다

Trident Protect로 관리하려는 각 애플리케이션을 정의해야 합니다. 응용 프로그램 CR을 수동으로 생성하거나 Trident Protect CLI를 사용하여 관리할 응용 프로그램을 정의할 수 있습니다.

CR을 사용하여 응용 프로그램을 추가합니다

단계

1. 대상 응용 프로그램 CR 파일을 만듭니다.

a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `maria-app.yaml`).

b. 다음 특성을 구성합니다.

- **metadata.name:** *(required)* 응용 프로그램 사용자 정의 리소스의 이름입니다. 보호 작업에 필요한 다른 CR 파일이 이 값을 참조하기 때문에 선택하는 이름을 확인합니다.
- **spec.includedNamespaces:** *(required)* 네임스페이스 및 레이블 선택기를 사용하여 응용 프로그램에서 사용하는 네임스페이스와 리소스를 지정합니다. 응용 프로그램 네임스페이스는 이 목록의 일부여야 합니다. 레이블 선택기는 선택 사항이며 지정된 각 네임스페이스 내에서 리소스를 필터링하는 데 사용할 수 있습니다.
- **spec.includedClusterScopedResources:** *(Optional)* 이 속성을 사용하여 응용 프로그램 정의에 포함될 클러스터 범위 리소스를 지정합니다. 이 속성을 사용하면 해당 그룹, 버전, 종류 및 레이블을 기반으로 이러한 리소스를 선택할 수 있습니다.
 - **groupVersionKind:** *(required)* 클러스터 범위 리소스의 API 그룹, 버전 및 종류를 지정합니다.
 - **labelSelector:** *(Optional)* 레이블을 기반으로 클러스터 범위 리소스를 필터링합니다.
- **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze:** *(Optional)* 이 주석은 KubeVirt 환경과 같이 스냅샷 전에 파일 시스템이 정지되는 가상 머신에서 정의된 애플리케이션에만 적용됩니다. 이 애플리케이션이 스냅샷 중에 파일 시스템에 쓸 수 있는지 여부를 지정합니다. `true`로 설정하면 응용 프로그램은 전역 설정을 무시하고 스냅샷 중에 파일 시스템에 쓸 수 있습니다. `false`로 설정하면 응용 프로그램에서 전역 설정을 무시하고 스냅샷 중에 파일 시스템이 고정됩니다. 지정되었지만 응용 프로그램에 가상 머신이 애플리케이션 정의에 없는 경우 주석은 무시됩니다. 지정하지 않으면 응용 프로그램이 ["Global Trident Protect Freeze\(전체 고정 보호\) 설정입니다"](#) 따릅니다.

응용 프로그램이 이미 생성된 후에 이 주석을 적용해야 하는 경우 다음 명령을 사용할 수 있습니다.

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
YAML 예:

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
          app: another-example-app
  includedClusterScopedResources:
    - groupVersionKind:
        group: rbac.authorization.k8s.io
        kind: ClusterRole
        version: v1
      labelSelector:
        matchLabels:
          mylabel: test
```

1. (선택 사항) 특정 레이블이 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

- **resourceFilter.resourceSelectionCriteria**:(필터링에 필요) Include resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 또는 을 Exclude 사용합니다. 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.

- **resourceFilter.resourceMatchers**: resourceMatcher 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.

- **resourceMatchers[].group**:(Optional) 필터링할 리소스의 그룹입니다.

- * resourceMatchers [].kind *: (Optional) 필터링할 리소스의 종류입니다.

- **resourceMatchers [].version**: (Optional) 필터링할 리소스의 버전입니다.

- **resourceMatchers[].names**:(Optional) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.

- `*resourceMatchers [].namespaces *`: (Optional) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- `resourceMatchers [ ].labelSelectors`: (Optional) 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "[Kubernetes 문서](#)" 예를 들면 다음과 `"trident.netapp.io/os=linux"` 같습니다.



둘 다 resourceFilter 그리고 labelSelector 사용됩니다, resourceFilter 먼저 실행한 다음 labelSelector 결과 리소스에 적용됩니다.

예를 들면 다음과 같습니다.

```
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"]
```

2. 환경에 맞게 응용 프로그램 CR을 만든 후 CR을 적용합니다. 예를 들면 다음과 같습니다.

```
kubectl apply -f maria-app.yaml
```

단계

1. 다음 예제 중 하나를 사용하여 응용 프로그램 정의를 만들고 적용합니다. 대괄호 안의 값을 사용자 환경의 정보로 바꿉니다. 예제에 표시된 인수를 사용하여 심표로 구분된 목록을 사용하여 응용 프로그램 정의에 네임스페이스와 리소스를 포함할 수 있습니다.

스냅샷을 생성하는 동안 애플리케이션이 파일 시스템에 쓸 수 있는지 여부를 지정하기 위해 앱을 생성할 때 주석을 선택적으로 사용할 수 있습니다. 이는 스냅샷이 발생하기 전에 파일 시스템이 정지되는 KubeVirt 환경과 같이 가상 시스템에서 정의된 애플리케이션에만 적용됩니다. 주석을 로 `'true'` 설정하면 응용 프로그램은 전역 설정을 무시하고 스냅샷 중에 파일 시스템에 쓸 수 있습니다. 로 `'false'` 설정하면 애플리케이션에서 전역 설정을 무시하고 스냅샷 중에 파일 시스템이 고정됩니다. 주석을 사용하지만 응용 프로그램에 가상 머신이 애플리케이션 정의에 없는 경우 주석은 무시됩니다. 주석을 사용하지 않으면 응용 프로그램이 을 "[Global Trident Protect Freeze\(전체 고정 보호\) 설정입니다](#)"따릅니다.

CLI를 사용하여 애플리케이션을 생성할 때 주석을 지정하려면 `--annotation` 플래그를 사용합니다.

- 응용 프로그램을 만들고 파일 시스템 고정 동작에 대한 전역 설정을 사용합니다.

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-namespace>
```

- 응용 프로그램을 생성하고 파일 시스템 고정 동작에 대한 로컬 응용 프로그램 설정을 구성합니다.

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-namespace> --annotation protect.trident.netapp.io/skip-vm-freeze
=<"true"|"false">
```

사용할 수 있습니다 `--resource-filter-include` 그리고 `--resource-filter-exclude` 리소스를 포함하거나 제외하기 위한 플래그 `resourceSelectionCriteria` 다음 예에서 볼 수 있듯이 그룹, 종류, 버전, 레이블, 이름 및 네임스페이스와 같은 것들이 있습니다.

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-namespace>
--resource-filter-include
' [{"Group": "apps", "Kind": "Deployment", "Version": "v1", "Names": ["my-deployment"], "Namespaces": ["my-namespace"], "LabelSelectors": ["app=my-app"]} ] '
```

Trident Protect를 사용하여 애플리케이션을 보호합니다

Trident Protect에서 관리하는 모든 앱은 자동화된 보호 정책을 사용하거나 임시적으로 스냅샷과 백업을 생성하여 보호할 수 있습니다.



데이터 보호 작업 중에 파일 시스템을 고정 및 고정 해제하도록 Trident Protect를 구성할 수 있습니다. ["Trident Protect를 사용하여 파일 시스템 중지를 구성하는 방법에 대해 자세히 알아보십시오"](#)..

필요 시 스냅샷을 생성합니다

언제든지 주문형 스냅샷을 생성할 수 있습니다.



클러스터 범위 리소스는 애플리케이션 정의에 명시적으로 참조되거나 애플리케이션 네임스페이스에 대한 참조가 있는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

CR을 사용하여 스냅샷을 생성합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-snapshot-cr.yaml` 지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
 - **metadata.name:** (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.applicationRef *:** 스냅샷을 생성할 애플리케이션의 Kubernetes 이름입니다.
 - *** spec.appVaultRef *:** (*required*) 스냅샷 내용(메타데이터)을 저장할 AppVault의 이름입니다.
 - **spec.reclaimPolicy:** (*Optional*) 스냅샷 CR이 삭제될 때 스냅샷의 AppArchive에 발생하는 동작을 정의합니다. 즉, 로 설정된 경우에도 Retain 스냅샷이 삭제됩니다. 유효한 옵션:
 - Retain (기본값)
 - Delete

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Delete
```

3. 파일을 올바른 값으로 채운 후 trident-protect-snapshot-cr.yaml CR:

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

CLI를 사용하여 스냅샷을 생성합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 스냅샷을 생성합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault <my_appvault_name> --app <name_of_app_to_snapshot> -n <application_namespace>
```

필요 시 백업을 생성합니다

언제든지 앱을 백업할 수 있습니다.



클러스터 범위 리소스는 애플리케이션 정의에 명시적으로 참조되거나 애플리케이션 네임스페이스에 대한 참조가 있는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

시작하기 전에

AWS 세션 토큰의 만료가 장시간 실행되는 S3 백업 작업에 충분한지 확인합니다. 백업 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 ["AWS API 설명서"](#) 참조하십시오.
- AWS 리소스의 자격 증명에 대한 자세한 내용은 ["AWS IAM 설명서"](#) 참조하십시오.

CR을 사용하여 백업을 생성합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-backup-cr.yaml` 지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
 - **metadata.name:** *(required)* 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.applicationRef *:** *(required)* 백업할 응용 프로그램의 Kubernetes 이름입니다.
 - *** spec.appVaultRef *:** *(required)* 백업 내용을 저장할 AppVault의 이름입니다.
 - *** spec.datamover *:** *(Optional)* 백업 작업에 사용할 백업 도구를 나타내는 문자열입니다. 가능한 값 (대소문자 구분):
 - Restic
 - Kopia (기본값)
 - **spec.reclaimPolicy:** *(Optional)* 클레임에서 해제될 때 백업에 어떤 일이 발생하는지를 정의합니다. 가능한 값:
 - Delete
 - Retain (기본값)
 - **spec.snapshotRef:** (선택 사항): 백업 소스로 사용할 스냅샷의 이름입니다. 제공하지 않으면 임시 스냅샷이 생성되고 백업됩니다.
 - **metadata.annotations.protect.trident.netapp.io/full-backup :** (선택 사항) 이 주석은 백업이 증분 방식이 아니어야 하는지 여부를 지정하는 데 사용됩니다. 기본적으로 모든 백업은 증분 백업입니다. 그러나 이 주석을 `true` 설정하면 백업이 비증분 백업이 됩니다. 지정하지 않으면 백업이 기본 증분 백업 설정을 따릅니다. 전체 백업을 주기적으로 수행한 다음 전체 백업 사이에 증분 백업을 수행하여 복원과 관련된 위험을 최소화하는 것이 좋습니다.

YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
  annotations:
    protect.trident.netapp.io/full-backup: "true"
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. 파일을 올바른 값으로 채운 후 `trident-protect-backup-cr.yaml` CR:


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

CLI를 사용하여 백업을 생성합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 백업을 만듭니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create backup <my_backup_name> --appvault <my-vault-name> --app <name_of_app_to_back_up> --data-mover  
<Kopia_or_Restic> -n <application_namespace>
```

필요에 따라 플래그를 사용하여 백업이 비증분 백업이어야 하는지 여부를 지정할 수 --full-backup 있습니다. 기본적으로 모든 백업은 증분 백업입니다. 이 플래그를 사용하면 백업이 비증분 백업이 됩니다. 전체 백업을 주기적으로 수행한 다음 전체 백업 사이에 증분 백업을 수행하여 복원과 관련된 위험을 최소화하는 것이 좋습니다.

데이터 보호 스케줄을 생성합니다

보호 정책은 정의된 일정에 따라 스냅샷, 백업 또는 둘 다를 생성하여 앱을 보호합니다. 매시간, 매일, 매주, 매월 스냅샷과 백업을 만들도록 선택할 수 있으며, 보관할 복사본 수를 지정할 수 있습니다. full-backup-rule 주석을 사용하여 비증분 전체 백업을 예약할 수 있습니다. 기본적으로 모든 백업은 증분 백업입니다. 주기적으로 전체 백업을 수행하고 그 사이에 증분 백업을 수행하면 복원과 관련된 위험을 줄이는 데 도움이 됩니다.



- 스냅샷에 대한 일정은 다음을 설정하여 생성할 수 있습니다. backupRetention 0으로 그리고 snapshotRetention 0보다 큰 값으로. 환경 snapshotRetention 0으로 설정하면 예약된 백업은 여전히 스냅샷을 생성하지만, 이는 임시적이며 백업이 완료되면 즉시 삭제됩니다.
- 클러스터 범위 리소스는 애플리케이션 정의에 명시적으로 참조되거나 애플리케이션 네임스페이스에 대한 참조가 있는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

CR을 사용하여 일정을 생성합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-schedule-cr.yaml` 지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
 - **metadata.name:** (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.datamover *:** (*Optional*) 백업 작업에 사용할 백업 도구를 나타내는 문자열입니다. 가능한 값 (대소문자 구분):
 - Restic
 - Kopia (기본값)
 - *** spec.applicationRef *:** 백업할 응용 프로그램의 Kubernetes 이름입니다.
 - *** spec.appVaultRef *:** (*required*) 백업 내용을 저장할 AppVault의 이름입니다.
 - **spec.backupRetention:** 보관할 백업 수. 0은 백업을 생성하지 않음을 나타냅니다(스냅샷만 생성).
 - *** spec.snapshotRetention *:** 보존할 스냅샷 수입니다. 0은 스냅샷을 생성하지 않아야 함을 나타냅니다.
 - **spec.granularity:** 일정이 실행되는 빈도. 가능한 값과 필수 관련 필드:
 - Hourly(지정해야 함) `spec.minute`)
 - Daily(지정해야 함) `spec.minute` 그리고 `spec.hour`)
 - Weekly(지정해야 함) `spec.minute` , `spec.hour` , 그리고 `spec.dayOfWeek`)
 - Monthly(지정해야 함) `spec.minute` , `spec.hour` , 그리고 `spec.dayOfMonth`)
 - Custom
 - **spec.dayOfMonth:** (선택 사항) 일정을 실행해야 하는 날짜(1~31)입니다. 세분성이 설정된 경우 이 필드가 필요합니다. Monthly . 값은 문자열로 제공되어야 합니다.
 - **spec.dayOfWeek:** (선택 사항) 일정을 실행해야 하는 요일(0~7). 0 또는 7의 값은 일요일을 나타냅니다. 세분성이 설정된 경우 이 필드가 필요합니다. Weekly . 값은 문자열로 제공되어야 합니다.
 - **spec.hour:** (선택 사항) 일정을 실행해야 하는 시간(0~23)입니다. 세분성이 설정된 경우 이 필드가 필요합니다. Daily , Weekly , 또는 Monthly . 값은 문자열로 제공되어야 합니다.
 - **spec.minute:** (선택 사항) 일정을 실행해야 하는 분(0~59)입니다. 세분성이 설정된 경우 이 필드가 필요합니다. Hourly , Daily , Weekly , 또는 Monthly . 값은 문자열로 제공되어야 합니다.
 - **metadata.annotations.protect.trident.netapp.io/full-backup-rule:** (*Optional*) 이 주석은 전체 백업 예약 규칙을 지정하는 데 사용됩니다. 지속적인 전체 백업을 위해 을 설정하거나 요구 사항에 따라 사용자 지정할 수 `always` 있습니다. 예를 들어, 매일 세분화를 선택하는 경우 전체 백업이 수행되는 요일을 지정할 수 있습니다.

백업 및 스냅샷 일정에 대한 YAML 예:

```

---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
  annotations:
    protect.trident.netapp.io/full-backup-rule: "Monday,Thursday"
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"

```

스냅샷 전용 일정에 대한 YAML 예:

```

---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"

```

3. 파일을 올바른 값으로 채운 후 trident-protect-schedule-cr.yaml CR:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

CLI를 사용하여 일정을 생성합니다

단계

1. 괄호 안의 값을 사용자 환경의 정보로 대체하여 보호 스케줄을 생성합니다. 예를 들면 다음과 같습니다.



을 사용하여 `tridentctl-protect create schedule --help` 이 명령에 대한 자세한 도움말 정보를 볼 수 있습니다.

```
tridentctl-protect create schedule <my_schedule_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> --backup
--retention <how_many_backups_to_retain> --data-mover
<Kopia_or_Restic> --day-of-month <day_of_month_to_run_schedule>
--day-of-week <day_of_month_to_run_schedule> --granularity
<frequency_to_run> --hour <hour_of_day_to_run> --minute
<minute_of_hour_to_run> --recurrence-rule <recurrence> --snapshot
--retention <how_many_snapshots_to_retain> -n <application_namespace>
--full-backup-rule <string>
```

고정 전체 백업을 위해 플래그를 `always` 설정하거나 요구 사항에 따라 사용자 지정할 수 `--full-backup-rule` 있습니다. 예를 들어, 매일 세분화를 선택하는 경우 전체 백업이 수행되는 요일을 지정할 수 있습니다. 예를 들어 월요일과 목요일에 전체 백업을 예약하려면 `--full-backup-rule "Monday,Thursday"` 사용합니다.

스냅샷 전용 일정의 경우 다음을 설정합니다. `--backup-retention` 0 0보다 큰 값을 지정합니다. `--snapshot-retention`.

스냅샷을 삭제합니다

더 이상 필요하지 않은 예약된 스냅샷 또는 주문형 스냅샷을 삭제합니다.

단계

1. 스냅샷과 연결된 스냅샷 CR을 제거합니다.

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

백업을 삭제합니다

더 이상 필요하지 않은 예약된 백업 또는 필요 시 백업을 삭제합니다.



회수 정책이 설정되어 있는지 확인하십시오. Delete 개체 저장소에서 모든 백업 데이터를 제거합니다. 정책의 기본 설정은 다음과 같습니다. Retain 실수로 인한 데이터 손실을 방지하기 위해. 정책이 변경되지 않으면 Delete 백업 데이터는 개체 스토리지에 남아 있으며 수동으로 삭제해야 합니다.

단계

1. 백업과 연결된 백업 CR을 제거합니다.

```
kubectl delete backup <backup_name> -n my-app-namespace
```

백업 작업의 상태를 확인합니다

명령줄을 사용하여 진행 중이거나, 완료되었거나, 실패한 백업 작업의 상태를 확인할 수 있습니다.

단계

1. 다음 명령을 사용하여 백업 작업의 상태를 검색하여 대괄호의 값을 사용자 환경의 정보로 바꿉니다.

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

Azure-NetApp-files(ANF) 작업을 위한 백업 및 복원이 가능합니다

Trident Protect를 설치한 경우 Azure-NetApp-files 스토리지 클래스를 사용하고 Trident 24.06 이전에 생성된 스토리지 백엔드에 공간 효율적인 백업 및 복원 기능을 사용할 수 있습니다. 이 기능은 NFSv4 볼륨에 적용되며 용량 풀에서 추가 공간을 사용하지 않습니다.

시작하기 전에

다음을 확인합니다.

- Trident Protect가 설치되어 있습니다.
- Trident Protect에서 애플리케이션을 정의했습니다. 이 응용 프로그램은 이 절차를 완료할 때까지 제한된 보호 기능을 제공합니다.
- `azure-netapp-files` 스토리지 백엔드의 기본 스토리지 클래스로 선택했습니다.

구성 단계를 위해 확장합니다

1. Trident 24.10으로 업그레이드하기 전에 ANF 볼륨을 생성한 경우 Trident에서 다음을 수행하십시오.

a. Azure-NetApp-files 기반이며 애플리케이션과 연결된 각 PV에 대해 스냅샷 디렉토리를 활성화합니다.

```
tridentctl update volume <pv name> --snapshot-dir=true -n trident
```

b. 연결된 각 PV에 대해 스냅샷 디렉토리가 활성화되었는지 확인합니다.

```
tridentctl get volume <pv name> -n trident -o yaml | grep  
snapshotDir
```

응답:

```
snapshotDirectory: "true"
```

+

스냅샷 디렉토리가 활성화되지 않은 경우 Trident Protect는 일반 백업 기능을 선택하여 백업 프로세스 중에 용량 풀의 공간을 일시적으로 사용합니다. 이 경우 용량 풀에서 백업 중인 볼륨 크기의 임시 볼륨을 생성할 수 있는 충분한 공간이 있는지 확인합니다.

결과

Trident Protect를 통해 애플리케이션이 백업 및 복원 준비가 되어 있습니다. 각 PVC는 백업 및 복원을 위해 다른 응용 프로그램에서 사용할 수도 있습니다.

응용 프로그램을 복원합니다

Trident Protect를 사용하여 애플리케이션을 복원합니다

Trident Protect를 사용하여 스냅샷 또는 백업에서 애플리케이션을 복원할 수 있습니다.

애플리케이션을 동일한 클러스터로 복구할 경우 기존 스냅샷에서 복구하는 속도가 빨라집니다.



- Application Restore 시 해당 Application에 설정된 모든 execution hook이 App으로 복구된다. 복원 후 실행 후크가 있는 경우 복구 작업의 일부로 자동으로 실행됩니다.
- Qtree 볼륨의 경우 백업에서 다른 네임스페이스 또는 원래 네임스페이스로 복원하는 기능이 지원됩니다. 그러나 Qtree 볼륨의 경우 스냅샷에서 다른 네임스페이스 또는 원래 네임스페이스로 복원하는 기능은 지원되지 않습니다.
- 고급 설정을 사용하여 복원 작업을 사용자 정의할 수 있습니다. 자세한 내용은 다음을 참조하세요. "[고급 Trident 보호 복원 설정 사용](#)".

백업에서 다른 네임스페이스로 복원합니다

BackupRestore CR을 사용하여 다른 네임스페이스로 백업을 복원하면 Trident Protect는 새 네임스페이스에서 응용 프로그램을 복원하고 복원된 응용 프로그램에 대한 응용 프로그램 CR을 만듭니다. 복원된 애플리케이션을 보호하려면 필요 시 백업 또는 스냅샷을 생성하거나 보호 일정을 설정합니다.



기존 리소스가 있는 다른 네임스페이스로 백업을 복원해도 백업의 리소스와 이름을 공유하는 리소스는 변경되지 않습니다. 백업의 모든 리소스를 복구하려면 타겟 네임스페이스를 삭제하고 다시 생성하거나 백업을 새 네임스페이스로 복원하십시오.

시작하기 전에

AWS 세션 토큰의 만료가 장시간 실행되는 S3 복원 작업에 충분한지 확인합니다. 복구 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 ["AWS API 설명서"](#) 참조하십시오.
- AWS 리소스의 자격 증명에 대한 자세한 내용은 ["AWS IAM 설명서"](#) 참조하십시오.



Kopia를 데이터 이동자로 사용하여 백업을 복원하는 경우 CR이나 CLI에서 주석을 지정하여 Kopia에서 사용하는 임시 저장소의 동작을 제어할 수 있습니다. ["코피아 문서"](#)를 참조하세요. 구성할 수 있는 옵션에 대한 자세한 내용은 다음을 참조하세요. 사용하세요 `tridentctl-protect create --help` Trident Protect CLI로 주석을 지정하는 방법에 대한 자세한 내용은 명령을 참조하세요.

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-backup-restore-cr.yaml` 지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
 - **metadata.name:** *(required)* 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.appArchivePath *:** 백업 콘텐츠가 저장되는 AppVault 내부의 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

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

- *** spec.appVaultRef *:** *(required)* 백업 내용이 저장된 AppVault의 이름입니다.
- **spec.namespaceMapping:** 복구 작업의 소스 네임스페이스를 대상 네임스페이스에 매핑하는 것입니다. `my-source-namespace` 및 `my-destination-namespace` 사용자 환경의 정보로 바꿉니다.

```
---  
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  
  namespaceMapping: [{"source": "my-source-namespace",  
"destination": "my-destination-namespace"}]
```

3. (선택 사항) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.



Trident Protect는 선택한 리소스와의 관계에 따라 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택했고 이 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

- **resourceFilter.resourceSelectionCriteria:** (필터링에 필요) Include resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 또는 Exclude 사용합니다. 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
 - **resourceFilter.resourceMatchers:** resourceMatcher 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.

- **resourceMatchers[].group:**(*Optional*) 필터링할 리소스의 그룹입니다.
- * resourceMatchers [].kind *: (*Optional*) 필터링할 리소스의 종류입니다.
- **resourceMatchers [].version:** (*Optional*) 필터링할 리소스의 버전입니다.
- **resourceMatchers[].names:**(*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
- *resourceMatchers [].namespaces *: (*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- **resourceMatchers [].labelSelectors:** (*Optional*) 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "[Kubernetes 문서](#)" 예를 들면 다음과 "trident.netapp.io/os=linux" 같습니다.

예를 들면 다음과 같습니다.

```
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. 파일을 올바른 값으로 채운 후 trident-protect-backup-restore-cr.yaml CR:

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

CLI를 사용합니다

단계

1. 대괄호 안의 값을 환경의 정보로 대체하여 백업을 다른 네임스페이스로 복원합니다. 이 namespace-mapping 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 올바른 대상 네임스페이스에 형식 `source1:dest1,source2:dest2`으로 매핑합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create backuprestore <my_restore_name> \  
--backup <backup_namespace>/<backup_to_restore> \  
--namespace-mapping <source_to_destination_namespace_mapping> \  
-n <application_namespace>
```

백업에서 원래 네임스페이스로 복구합니다

언제든지 원래 네임스페이스로 백업을 복원할 수 있습니다.

시작하기 전에

AWS 세션 토큰의 만료가 장시간 실행되는 S3 복원 작업에 충분한지 확인합니다. 복구 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 ["AWS API 설명서"](#) 참조하십시오.
- AWS 리소스의 자격 증명에 대한 자세한 내용은 ["AWS IAM 설명서"](#) 참조하십시오.



Kopia를 데이터 이동자로 사용하여 백업을 복원하는 경우 CR이나 CLI에서 주석을 지정하여 Kopia에서 사용하는 임시 저장소의 동작을 제어할 수 있습니다. ["코피아 문서"](#)를 참조하세요. 구성할 수 있는 옵션에 대한 자세한 내용은 다음을 참조하세요. 사용하세요 `tridentctl-protect create --help` Trident Protect CLI로 주석을 지정하는 방법에 대한 자세한 내용은 명령을 참조하세요.

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-backup-ipr-cr.yaml` 지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
 - **metadata.name:** *(required)* 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.appArchivePath *:** 백업 콘텐츠가 저장되는 AppVault 내부의 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

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

- *** spec.appVaultRef *:** *(required)* 백업 내용이 저장된 AppVault의 이름입니다.

예를 들면 다음과 같습니다.

```
---
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. (선택 사항) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.



Trident Protect는 선택한 리소스와의 관계에 따라 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택했고 이 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

- **resourceFilter.resourceSelectionCriteria:** (필터링에 필요) Include resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 또는 을 Exclude 사용합니다. 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
 - **resourceFilter.resourceMatchers:** resourceMatcher 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.
 - **resourceMatchers[].group:** *(Optional)* 필터링할 리소스의 그룹입니다.
 - *** resourceMatchers [].kind *:** *(Optional)* 필터링할 리소스의 종류입니다.
 - **resourceMatchers [].version:** *(Optional)* 필터링할 리소스의 버전입니다.

- **resourceMatchers[].names:** *(Optional)* 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
- ***resourceMatchers [].namespaces *:** *(Optional)* 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- **resourceMatchers [].labelSelectors:** *(Optional)* 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "[Kubernetes 문서](#)" 예를 들면 다음과 `"trident.netapp.io/os=linux"` 같습니다.

예를 들면 다음과 같습니다.

```
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. 파일을 올바른 값으로 채운 후 trident-protect-backup-ipr-cr.yaml CR:

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

CLI를 사용합니다

단계

1. 대괄호 안의 값을 환경의 정보로 대체하여 백업을 원래 네임스페이스로 복원합니다. backup`인수에 네임스페이스 및 백업 이름이 형식으로 `<namespace>/<name>` 사용됩니다. 예를 들면 다음과 같습니다.

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

백업에서 다른 클러스터로 복원합니다

원래 클러스터에 문제가 있는 경우 백업을 다른 클러스터로 복구할 수 있습니다.



Kopia를 데이터 이동자로 사용하여 백업을 복원하는 경우 CR이나 CLI에서 주석을 지정하여 Kopia에서 사용하는 임시 저장소의 동작을 제어할 수 있습니다. 를 참조하세요 ["코피아 문서"](#) 구성할 수 있는 옵션에 대한 자세한 내용은 다음을 참조하세요. 사용하세요 `tridentctl-protect create --help` Trident Protect CLI로 주석을 지정하는 방법에 대한 자세한 내용은 명령을 참조하세요.

시작하기 전에

다음 필수 구성 요소가 충족되는지 확인합니다.

- 대상 클러스터에 Trident Protect가 설치되어 있습니다.
- 대상 클러스터에는 백업이 저장되는 소스 클러스터와 동일한 AppVault의 버킷 경로에 대한 액세스 권한이 있습니다.
- AppVault CR에 정의된 개체 저장소 버킷에 로컬 환경이 연결할 수 있는지 확인하십시오. `tridentctl-protect get appvaultcontent` 명령. 네트워크 제한으로 인해 액세스할 수 없는 경우 대상 클러스터의 포드 내에서 Trident Protect CLI를 실행하세요.
- AWS 세션 토큰 만료가 장기 실행 중인 복구 작업에 충분한지 확인합니다. 복구 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.
 - 현재 세션 토큰 만료 확인에 대한 자세한 내용은 를 ["AWS API 설명서"](#) 참조하십시오.
 - AWS 리소스의 자격 증명에 대한 자세한 내용은 를 ["AWS 설명서"](#) 참조하십시오.

단계

1. Trident Protect CLI 플러그인을 사용하여 대상 클러스터에서 AppVault CR의 가용성을 확인합니다.

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



응용 프로그램 복구용 네임스페이스가 대상 클러스터에 있는지 확인합니다.

2. 대상 클러스터에서 사용 가능한 AppVault의 백업 콘텐츠를 봅니다.

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

이 명령을 실행하면 원래 클러스터, 해당 응용 프로그램 이름, 타임스탬프 및 아카이브 경로를 비롯하여 AppVault에 사용 가능한 백업이 표시됩니다.

- 출력 예: *

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
|  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 이름 및 보관 경로를 사용하여 응용 프로그램을 대상 클러스터로 복원합니다.

CR을 사용합니다

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-backup-restore-cr.yaml` 지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.

- **metadata.name:** (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
- *** spec.appVaultRef *:** (*required*) 백업 내용이 저장된 AppVault의 이름입니다.
- *** spec.appArchivePath *:** 백업 콘텐츠가 저장되는 AppVault 내부의 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

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



BackupRestore CR을 사용할 수 없는 경우 2단계에서 언급한 명령을 사용하여 백업 내용을 볼 수 있습니다.

- **spec.namespaceMapping:** 복구 작업의 소스 네임스페이스를 대상 네임스페이스에 매핑하는 것입니다. `my-source-namespace` 및 `my-destination-namespace` 사용자 환경의 정보로 바꿉니다.

예를 들면 다음과 같습니다.

```
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
  namespaceMapping: [{"source": "my-source-namespace", "destination": "my-destination-namespace"}]
```

3. 파일을 올바른 값으로 채운 후 `trident-protect-backup-restore-cr.yaml` CR:

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

CLI를 사용합니다

1. 다음 명령을 사용하여 응용 프로그램을 복원하고 대괄호 안의 값을 사용자 환경의 정보로 바꿉니다. 네임스페이스 매핑 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 `Source1:dest1`, `source2:dest2` 형식으로 올바른 대상 네임스페이스에 매핑합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--context <destination_cluster_name> \
-n <application_namespace>
```

스냅샷에서 다른 네임스페이스로 복구합니다

사용자 지정 리소스(CR) 파일을 사용하여 스냅샷에서 데이터를 다른 네임스페이스 또는 원래 소스 네임스페이스로 복원할 수 있습니다. SnapshotRestore CR을 사용하여 스냅샷을 다른 네임스페이스로 복구할 경우 Trident Protect는 새 네임스페이스에서 애플리케이션을 복원하고 복원된 애플리케이션에 대한 애플리케이션 CR을 생성합니다. 복원된 애플리케이션을 보호하려면 필요 시 백업 또는 스냅샷을 생성하거나 보호 일정을 설정합니다.



SnapshotRestore는 다음을 지원합니다. spec.storageClassMapping 속성이지만 소스 및 대상 저장소 클래스가 동일한 저장소 백엔드를 사용하는 경우에만 해당됩니다. 복원을 시도하는 경우 StorageClass 다른 스토리지 백엔드를 사용하는 경우 복원 작업이 실패합니다.

시작하기 전에

AWS 세션 토큰의 만료가 장시간 실행되는 S3 복원 작업에 충분한지 확인합니다. 복구 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 ["AWS API 설명서"](#) 참조하십시오.
- AWS 리소스의 자격 증명에 대한 자세한 내용은 ["AWS IAM 설명서"](#) 참조하십시오.

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-snapshot-restore-cr.yaml` 지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
 - **metadata.name:** *(required)* 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.appVaultRef *:** *(required)* 스냅샷 내용이 저장된 AppVault의 이름입니다.
 - *** spec.appArchivePath *:** 스냅샷 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

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

- **spec.namespaceMapping:** 복구 작업의 소스 네임스페이스를 대상 네임스페이스에 매핑하는 것입니다. `my-source-namespace` 및 `my-destination-namespace` 사용자 환경의 정보로 바꿉니다.

```
---
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. (선택 사항) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.



Trident Protect는 선택한 리소스와의 관계에 따라 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택했고 이 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

- **resourceFilter.resourceSelectionCriteria:** (필터링에 필요) Include resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 또는 을 Exclude 사용합니다. 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
 - **resourceFilter.resourceMatchers:** resourceMatcher 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.

- **resourceMatchers[].group:**(*Optional*) 필터링할 리소스의 그룹입니다.
- * resourceMatchers [].kind *: (*Optional*) 필터링할 리소스의 종류입니다.
- **resourceMatchers [].version:** (*Optional*) 필터링할 리소스의 버전입니다.
- **resourceMatchers[].names:**(*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
- *resourceMatchers [].namespaces *: (*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- **resourceMatchers [].labelSelectors:** (*Optional*) 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "[Kubernetes 문서](#)" 예를 들면 다음과 "trident.netapp.io/os=linux" 같습니다.

예를 들면 다음과 같습니다.

```
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. 파일을 올바른 값으로 채운 후 trident-protect-snapshot-restore-cr.yaml CR:

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

CLI를 사용합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 스냅샷을 다른 네임스페이스로 복원합니다.

- snapshot`인수에 네임스페이스 및 스냅샷 이름이 형식으로 `- 이 namespace-mapping 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 올바른 대상 네임스페이스에 형식 `source1:dest1,source2:dest2`으로 매핑합니다.

예를 들면 다음과 같습니다.

```
tridentctl-protect create snapshotrestore <my_restore_name> \  
--snapshot <namespace/snapshot_to_restore> \  
--namespace-mapping <source_to_destination_namespace_mapping> \  
-n <application_namespace>
```

스냅샷에서 원래 네임스페이스로 복구합니다

언제든지 스냅샷을 원래 네임스페이스로 복구할 수 있습니다.

시작하기 전에

AWS 세션 토큰의 만료가 장시간 실행되는 S3 복원 작업에 충분한지 확인합니다. 복구 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 ["AWS API 설명서"](#) 참조하십시오.
- AWS 리소스의 자격 증명에 대한 자세한 내용은 ["AWS IAM 설명서"](#) 참조하십시오.

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-snapshot-ipr-cr.yaml` 지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
 - **metadata.name:** *(required)* 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.appVaultRef *:** *(required)* 스냅샷 내용이 저장된 AppVault의 이름입니다.
 - *** spec.appArchivePath *:** 스냅샷 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

```
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. (선택 사항) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.



Trident Protect는 선택한 리소스와의 관계에 따라 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택했고 이 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

- **resourceFilter.resourceSelectionCriteria:**(필터링에 필요) Include resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 또는 을 Exclude 사용합니다. 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
 - **resourceFilter.resourceMatchers:** resourceMatcher 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.
 - **resourceMatchers[].group:** *(Optional)* 필터링할 리소스의 그룹입니다.
 - *** resourceMatchers [].kind *:** *(Optional)* 필터링할 리소스의 종류입니다.
 - **resourceMatchers [].version:** *(Optional)* 필터링할 리소스의 버전입니다.
 - **resourceMatchers[].names:** *(Optional)* 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.

- `*resourceMatchers [].namespaces *`: (*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- `resourceMatchers [ ].labelSelectors`: (*Optional*) 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "[Kubernetes 문서](#)" 예를 들면 다음과 `"trident.netapp.io/os=linux"` 같습니다.

예를 들면 다음과 같습니다.

```
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. 파일을 올바른 값으로 채운 후 `trident-protect-snapshot-ipr-cr.yaml` CR:

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

CLI를 사용합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 스냅샷을 원래 네임스페이스로 복원합니다. 예를 들면 다음과 같습니다.

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

복구 작업의 상태를 확인합니다

명령줄을 사용하여 진행 중이거나, 완료되었거나, 실패한 복구 작업의 상태를 확인할 수 있습니다.

단계

1. 다음 명령을 사용하여 복원 작업의 상태를 검색하여 대괄호의 값을 사용자 환경의 정보로 바꿉니다.

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

고급 Trident 보호 복원 설정 사용

주석, 네임스페이스 설정, 스토리지 옵션 등의 고급 설정을 사용하여 복원 작업을 사용자 정의하여 특정 요구 사항을 충족할 수 있습니다.

복원 및 페일오버 작업 중 네임스페이스 주석 및 레이블

복원 및 페일오버 작업 중에 대상 네임스페이스의 레이블과 주석이 소스 네임스페이스의 레이블 및 주석과 일치하도록 만듭니다. 대상 네임스페이스에 없는 소스 네임스페이스의 레이블 또는 주석이 추가되고 이미 존재하는 모든 레이블 또는 주석이 소스 네임스페이스의 값과 일치하도록 덮어쓰여집니다. 대상 네임스페이스에만 있는 레이블이나 주석은 변경되지 않습니다.



Red Hat OpenShift를 사용하는 경우 OpenShift 환경에서 네임스페이스 주석의 중요한 역할을 알아두는 것이 중요합니다. 네임스페이스 주석은 복원된 포드가 OpenShift 보안 컨텍스트 제약(SCC)에 의해 정의된 적절한 권한과 보안 구성을 준수하고 권한 문제 없이 볼륨에 액세스할 수 있도록 보장합니다. 자세한 내용은 다음을 참조하세요. ["OpenShift 보안 컨텍스트 제약 조건 문서"](#).

복원 또는 페일오버 작업을 수행하기 전에 Kubernetes 환경 변수를 설정하여 타겟 네임스페이스의 특정 주석을 덮어쓰지 않도록 할 수 있습니다 RESTORE_SKIP_NAMESPACE_ANNOTATIONS. 예를 들면 다음과 같습니다.

```
helm upgrade trident-protect --set  
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key  
_to_skip_2> --reuse-values
```



복원 또는 장애 조치 작업을 수행할 때 지정된 네임스페이스 주석 및 레이블 restoreSkipNamespaceAnnotations 그리고 restoreSkipNamespaceLabels 복구 또는 장애 조치 작업에서 제외됩니다. 이러한 설정은 Helm을 처음 설치할 때 구성되어야 합니다. 자세한 내용은 다음을 참조하세요. ["AutoSupport 및 네임스페이스 필터링 옵션 구성"](#).

플래그가 있는 Helm을 사용하여 소스 애플리케이션을 설치한 경우 --create-namespace 레이블 키에 특수 치료가 name 제공됩니다. 복구 또는 페일오버 프로세스 중에 Trident Protect는 이 레이블을 대상 네임스페이스에 복제하지만 소스의 값이 소스 네임스페이스와 일치하면 대상 네임스페이스 값으로 업데이트합니다. 이 값이 소스 네임스페이스와 일치하지 않으면 변경 없이 대상 네임스페이스로 복사됩니다.

예

다음 예제에서는 각각 다른 주석과 레이블이 있는 소스 및 대상 네임스페이스를 보여 줍니다. 작업 전후에 대상 네임스페이스의 상태와 대상 네임스페이스에서 주석과 레이블이 결합되거나 덮어쓰지는 방법을 확인할 수 있습니다.

복구 또는 페일오버 작업 전

다음 표에서는 복구 또는 페일오버 작업 이전의 예제 소스 및 대상 네임스페이스의 상태를 보여 줍니다.

네임스페이스	주석	라벨
네임스페이스 ns-1(소스)	• <code>annotation.one/key:"updatedvalue"</code>(주석 1개/키) • <code>Annotation.two/key(주석.two/키):</code>	• 환경 = 운영 • 규정 준수 = HIPAA • 이름 = ns-1
네임스페이스 ns-2(대상)	• <code>Annotation.one/key(주석. 하나/키):"true"</code> • <code>Annotation.Three/key:"false"</code>	• 역할 = 데이터베이스

복구 작업 후

다음 표에서는 복구 또는 페일오버 작업 후의 예제 대상 네임스페이스의 상태를 보여 줍니다. 일부 키가 추가되고, 일부 키가 덮어쓰여졌으며, name 대상 네임스페이스와 일치하도록 레이블이 업데이트되었습니다.

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• <code>annotation.one/key:"updatedvalue"</code>(주석 1개/키) • <code>Annotation.two/key(주석.two/키):</code> • <code>Annotation.Three/key:"false"</code>	• 이름 = ns-2 • 규정 준수 = HIPAA • 환경 = 운영 • 역할 = 데이터베이스

지원되는 필드

이 섹션에서는 복원 작업에 사용할 수 있는 추가 필드에 대해 설명합니다.

스토리지 클래스 매핑

그만큼 `spec.storageClassMapping` 속성은 소스 애플리케이션에 있는 스토리지 클래스에서 대상 클러스터의 새 스토리지 클래스로의 매핑을 정의합니다. 서로 다른 스토리지 클래스를 사용하는 클러스터 간에 애플리케이션을 마이그레이션하거나 BackupRestore 작업에 대한 스토리지 백엔드를 변경할 때 이 기능을 사용할 수 있습니다.

- 예: *

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

지원되는 주석

이 섹션에서는 시스템의 다양한 동작을 구성하는 데 지원되는 애노테이션을 나열합니다. 사용자가 애노테이션을 명시적으로 설정하지 않으면 시스템은 기본값을 사용합니다.

주석	유형	설명	기본값
보호.트라이던트.넷앱.io/데이터-무버-타임아웃-초	문자열	데이터 이동 작업이 중단되는 데 허용되는 최대 시간(초)입니다.	"300"
보호.트라이던트.넷앱.io/코피아-콘텐츠-캐시-크기-제한-mb	문자열	Kopia 콘텐츠 캐시의 최대 크기 제한(메가바이트)입니다.	"1000"

NetApp SnapMirror 및 Trident Protect를 사용하여 애플리케이션을 복제합니다

Trident Protect를 사용하면 NetApp SnapMirror 기술의 비동기식 복제 기능을 사용하여 동일한 클러스터 또는 서로 다른 클러스터 간에 데이터 및 애플리케이션 변경 사항을 스토리지 백엔드에서 다른 스토리지 백엔드로 복제할 수 있습니다.

복원 및 페일오버 작업 중 네임스페이스 주석 및 레이블

복원 및 페일오버 작업 중에 대상 네임스페이스의 레이블과 주석이 소스 네임스페이스의 레이블 및 주석과 일치하도록 만듭니다. 대상 네임스페이스에 없는 소스 네임스페이스의 레이블 또는 주석이 추가되고 이미 존재하는 모든 레이블 또는 주석이 소스 네임스페이스의 값과 일치하도록 덮어쓰여집니다. 대상 네임스페이스에만 있는 레이블이나 주석은 변경되지 않습니다.



Red Hat OpenShift를 사용하는 경우 OpenShift 환경에서 네임스페이스 주석의 중요한 역할을 알아두는 것이 중요합니다. 네임스페이스 주석은 복원된 포드가 OpenShift 보안 컨텍스트 제약(SCC)에 의해 정의된 적절한 권한과 보안 구성을 준수하고 권한 문제 없이 볼륨에 액세스할 수 있도록 보장합니다. 자세한 내용은 다음을 참조하세요. ["OpenShift 보안 컨텍스트 제약 조건 문서"](#).

복원 또는 페일오버 작업을 수행하기 전에 Kubernetes 환경 변수를 설정하여 타겟 네임스페이스의 특정 주석을 덮어쓰지 않도록 할 수 있습니다 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`. 예를 들면 다음과 같습니다.

```
helm upgrade trident-protect --set  
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key  
_to_skip_2> --reuse-values
```



복원 또는 장애 조치 작업을 수행할 때 지정된 네임스페이스 주석 및 레이블 `restoreSkipNamespaceAnnotations` 그리고 `restoreSkipNamespaceLabels` 복구 또는 장애 조치 작업에서 제외됩니다. 이러한 설정은 Helm을 처음 설치할 때 구성되어야 합니다. 자세한 내용은 다음을 참조하세요. ["AutoSupport 및 네임스페이스 필터링 옵션 구성"](#).

플래그가 있는 Helm을 사용하여 소스 애플리케이션을 설치한 경우 `--create-namespace` 레이블 키에 특수 치료가

name 제공됩니다. 복구 또는 페일오버 프로세스 중에 Trident Protect는 이 레이블을 대상 네임스페이스에 복제하지만 소스의 값이 소스 네임스페이스와 일치하면 대상 네임스페이스 값으로 업데이트합니다. 이 값이 소스 네임스페이스와 일치하지 않으면 변경 없이 대상 네임스페이스로 복사됩니다.

예

다음 예제에서는 각각 다른 주석과 레이블이 있는 소스 및 대상 네임스페이스를 보여 줍니다. 작업 전후에 대상 네임스페이스의 상태와 대상 네임스페이스에서 주석과 레이블이 결합되거나 덮어쓰지는 방법을 확인할 수 있습니다.

복구 또는 페일오버 작업 전

다음 표에서는 복구 또는 페일오버 작업 이전의 예제 소스 및 대상 네임스페이스의 상태를 보여 줍니다.

네임스페이스	주석	라벨
네임스페이스 ns-1(소스)	• annotation.one/key:"updatedvalue"(주석 1개/키) • Annotation.two/key(주석.two/키):	• 환경 = 운영 • 규정 준수 = HIPAA • 이름 = ns-1
네임스페이스 ns-2(대상)	• Annotation.one/key(주석. 하나/키):"true" • Annotation.Three/key:"false"	• 역할 = 데이터베이스

복구 작업 후

다음 표에서는 복구 또는 페일오버 작업 후의 예제 대상 네임스페이스의 상태를 보여 줍니다. 일부 키가 추가되고, 일부 키가 덮어쓰여졌으며, name 대상 네임스페이스와 일치하도록 레이블이 업데이트되었습니다.

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• annotation.one/key:"updatedvalue"(주석 1개/키) • Annotation.two/key(주석.two/키): • Annotation.Three/key:"false"	• 이름 = ns-2 • 규정 준수 = HIPAA • 환경 = 운영 • 역할 = 데이터베이스



데이터 보호 작업 중에 파일 시스템을 고정 및 고정 해제하도록 Trident Protect를 구성할 수 있습니다. ["Trident Protect를 사용하여 파일 시스템 중지를 구성하는 방법에 대해 자세히 알아보십시오"](#)..

장애 조치 및 역방향 작업 중 실행 후크

AppMirror 관계를 사용하여 애플리케이션을 보호하는 경우 장애 조치 및 역방향 작업 중에 알고 있어야 할 실행 후크와 관련된 특정 동작이 있습니다.

- 장애 조치(failover) 중에 실행 후크는 소스 클러스터에서 대상 클러스터로 자동으로 복사됩니다. 따라서 수동으로 다시 만들 필요가 없습니다. 장애 조치(failover) 후에는 실행 후크가 애플리케이션에 존재하며 관련 작업 중에 실행됩니다.

- 역방향 또는 역방향 재동기화 중에는 애플리케이션에 있는 기존 실행 후크가 모두 제거됩니다. 소스 애플리케이션이 대상 애플리케이션으로 변경되면 이러한 실행 후크는 유효하지 않으므로 삭제되어 실행되지 않습니다.

실행 후크에 대해 자세히 알아보려면 다음을 참조하세요. "[Trident Protect 실행 후크를 관리합니다](#)".

복제 관계를 설정합니다

복제 관계를 설정하려면 다음을 수행해야 합니다.

- Trident Protect에서 앱 스냅샷 생성 빈도 선택(앱의 Kubernetes 리소스 및 각 앱 볼륨에 대한 볼륨 스냅샷 포함)
- 복제 일정 선택(Kubernetes 리소스 및 영구 볼륨 데이터 포함)
- 스냅샷을 생성할 시간을 설정합니다

단계

1. 소스 클러스터에서 소스 응용 프로그램에 대한 AppVault를 생성합니다. 스토리지 공급자에 따라 예시 "[AppVault 사용자 지정 리소스](#)" 환경에 맞게 수정합니다.

CR을 사용하여 AppVault를 작성합니다

- a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: trident-protect-appvault-primary-source.yaml).
- b. 다음 특성을 구성합니다.
 - **metadata.name:** (required) AppVault 사용자 정의 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일은 이 값을 참조하므로 선택한 이름을 기록해 둡니다.
 - **spec.providerConfig:** (required) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다. bucketName과 공급자에게 필요한 기타 세부 정보를 선택합니다. 복제 관계에 필요한 다른 CR 파일은 이러한 값을 참조하므로 선택한 값을 기록해 둡니다. 다른 공급자의 AppVault CRS에 대한 예는 ["AppVault 사용자 지정 리소스"](#) 참조하십시오.
 - **spec.providerCredentials:** (required) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
 - **spec.providerCredentials.valueFromSecret:** (required) 자격 증명 값이 비밀에서 와야 함을 나타냅니다.
 - * key *: (required) 선택할 비밀의 유효한 키입니다.
 - * name *: (required) 이 필드의 값을 포함하는 비밀의 이름입니다. 같은 네임스페이스에 있어야 합니다.
 - **spec.providerCredentials.secretAccessKey:** (required) 제공자에 액세스하는 데 사용되는 액세스 키입니다. name * 은 * spec.providerCredentials.valueFromSecret.name* 과 일치해야 합니다.
 - **spec.providerType:** (required) 백업을 제공하는 항목을 결정합니다(예: NetApp ONTAP S3, 일반 S3, Google Cloud 또는 Microsoft Azure). 가능한 값:
 - 설치하고
 - Azure를 지원합니다
 - GCP
 - 일반 - S3
 - ONTAP-S3
 - StorageGRID-S3

- c. 파일을 올바른 값으로 채운 후 trident-protect-appvault-primary-source.yaml CR:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n trident-protect
```

CLI를 사용하여 AppVault를 작성합니다

- a. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 AppVault를 작성합니다.

```
tridentctl-protect create vault Azure <vault-name> --account <account-name> --bucket <bucket-name> --secret <secret-name>
```

2. 소스 클러스터에서 소스 애플리케이션 CR:

CR을 사용하여 소스 응용 프로그램을 만듭니다

- 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: trident-protect-app-source.yaml).
- 다음 특성을 구성합니다.
 - metadata.name:** (required) 응용 프로그램 사용자 정의 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일은 이 값을 참조하므로 선택한 이름을 기록해 둡니다.
 - spec.includedNamespaces:** (required) 네임스페이스 및 관련 레이블의 배열입니다. 네임스페이스 이름을 사용하고 선택적으로 레이블을 사용하여 네임스페이스 범위를 좁혀 여기에 나열된 네임스페이스에 있는 리소스를 지정합니다. 응용 프로그램 네임스페이스는 이 배열의 일부여야 합니다.
 - YAML 예 *:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      labelSelector: {}
```

- 파일을 올바른 값으로 채운 후 trident-protect-app-source.yaml CR:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

CLI를 사용하여 소스 애플리케이션을 생성합니다

- 소스 응용 프로그램을 만듭니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create app <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

- 선택적으로 소스 클러스터에서 소스 애플리케이션의 스냅샷을 만듭니다. 이 스냅샷은 대상 클러스터에서 애플리케이션의 기반으로 사용됩니다. 이 단계를 건너뛸 경우 다음 예약된 스냅샷이 실행될 때까지 기다려야 최신 스냅샷이 생성됩니다.

아래 제공된 일정 외에도, 피어링된 ONTAP 클러스터 간에 공통 스냅샷을 유지하기 위해 7일의 보존 기간을 갖는 별도의 일일 스냅샷 일정을 생성하는 것이 좋습니다. 이를 통해 스냅샷을 최대 7일 동안 사용할 수 있지만, 보존 기간은 사용자 요구 사항에 따라 맞춤 설정할 수 있습니다.



장애 조치(failover)가 발생하면 시스템은 최대 7일 동안 이러한 스냅샷을 역방향 작업에 사용할 수 있습니다. 이 방식은 마지막 스냅샷 이후 변경된 내용만 전송하고 모든 데이터는 전송하지 않으므로 역방향 프로세스가 더 빠르고 효율적입니다.

해당 애플리케이션의 기존 일정이 이미 원하는 보존 요구 사항을 충족하는 경우 추가 일정은 필요하지 않습니다.

CR을 사용하여 스냅샷을 촬영합니다

a. 소스 애플리케이션에 대한 복제 스케줄을 생성합니다.

i. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-schedule.yaml`).

ii. 다음 특성을 구성합니다.

- **metadata.name:** *(required)* 일정 사용자 정의 리소스의 이름입니다.
- *** spec.AppVaultRef *:** *(required)* 이 값은 원본 응용 프로그램에 대한 AppVault의 `metadata.name` 필드와 일치해야 합니다.
- **spec.ApplicationRef:** *(required)* 이 값은 소스 응용 프로그램 CR의 `metadata.name` 필드와 일치해야 합니다.
- **spec.backupRetention:** *(required)* 이 필드는 필수 필드이며 값을 0으로 설정해야 합니다.
- *** spec.enabled *:** 반드시 `true`로 설정해야 합니다.
- **spec.granularity:** 으로 설정해야 `Custom` 합니다.
- **spec.recurrenceRule:** UTC 시간과 반복 간격으로 시작 날짜를 정의합니다.
- **spec.snapshotRetention:** 2로 설정해야 합니다.

YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule-0e1f88ab-f013-4bce-8ae9-6afed9df59a1
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. 파일을 올바른 값으로 채운 후 `trident-protect-schedule.yaml` CR:

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-namespace
```

CLI를 사용하여 스냅샷을 생성합니다

- a. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 스냅샷을 생성합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault  
<my_appvault_name> --app <name_of_app_to_snapshot> -n  
<application_namespace>
```

4. 대상 클러스터에서 소스 클러스터에 적용한 AppVault CR과 동일한 소스 응용 프로그램 AppVault CR을 생성하고 이름을 지정합니다(예: trident-protect-appvault-primary-destination.yaml).

5. CR 적용:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n  
my-app-namespace
```

6. 대상 클러스터에서 대상 응용 프로그램에 대한 대상 AppVault CR을 생성합니다. 스토리지 공급자에 따라 의 예를 "AppVault 사용자 지정 리소스" 환경에 맞게 수정합니다.

- a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: trident-protect-appvault-secondary-destination.yaml).

- b. 다음 특성을 구성합니다.

- **metadata.name:** (required) AppVault 사용자 정의 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일은 이 값을 참조하므로 선택한 이름을 기록해 둡니다.
- **spec.providerConfig:** (required) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다. 'bucketName'와 공급자에게 필요한 기타 세부 정보를 선택합니다. 복제 관계에 필요한 다른 CR 파일은 이러한 값을 참조하므로 선택한 값을 기록해 둡니다. 다른 공급자의 AppVault CRS에 대한 예는 "AppVault 사용자 지정 리소스" 참조하십시오.
- **spec.providerCredentials:** (required) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
 - **spec.providerCredentials.valueFromSecret:** (required) 자격 증명 값이 비밀에서 와야 함을 나타냅니다.
 - * key *: (required) 선택할 비밀의 유효한 키입니다.
 - * name *: (required) 이 필드의 값을 포함하는 비밀의 이름입니다. 같은 네임스페이스에 있어야 합니다.
 - **spec.providerCredentials.secretAccessKey:** (required) 제공자에 액세스하는 데 사용되는 액세스 키입니다. name * 은 * spec.providerCredentials.valueFromSecret.name* 과 일치해야 합니다.
- **spec.providerType:** (required) 백업을 제공하는 항목을 결정합니다(예: NetApp ONTAP S3, 일반 S3, Google Cloud 또는 Microsoft Azure). 가능한 값:
 - 설치하고
 - Azure를 지원합니다
 - GCP

- 일반 - S3
- ONTAP-S3
- StorageGRID-S3

c. 파일을 올바른 값으로 채운 후 `trident-protect-appvault-secondary-destination.yaml` CR:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml  
-n my-app-namespace
```

7. 대상 클러스터에서 AppMirrorRelationship CR 파일을 생성합니다.

CR을 사용하여 **AppMirrorRelationship**을 생성합니다

- a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: trident-protect-relationship.yaml).
- b. 다음 특성을 구성합니다.

- **metadata.name:** (필수) AppMirrorRelationship 사용자 정의 리소스의 이름입니다.
- **spec.destinationAppVaultRef:** (*required*) 이 값은 대상 클러스터의 대상 응용 프로그램에 대한 AppVault 이름과 일치해야 합니다.
- **spec.namespaceMapping:** (*required*) 대상 및 소스 네임스페이스는 해당 응용 프로그램 CR에 정의된 응용 프로그램 네임스페이스와 일치해야 합니다.
- *** spec.sourceAppVaultRef *:** (*required*) 이 값은 소스 응용 프로그램의 AppVault 이름과 일치해야 합니다.
- *** spec.sourceApplicationName *:** (*required*) 이 값은 소스 응용 프로그램 CR에서 정의한 소스 응용 프로그램의 이름과 일치해야 합니다.
- *** spec.storageClassName *:** (*required*) 클러스터에서 유효한 스토리지 클래스의 이름을 선택하십시오. 스토리지 클래스는 소스 환경을 피어링하는 ONTAP 스토리지 VM에 연결되어야 한다.
- **spec.recurrenceRule:** UTC 시간과 반복 간격으로 시작 날짜를 정의합니다.

YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsrm-2
```

- c. 파일을 올바른 값으로 채운 후 trident-protect-relationship.yaml CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

CLI를 사용하여 **AppMirrorRelationship**을 생성한다

- a. AppMirrorRelationship 개체를 만들고 적용하여 대괄호 안의 값을 사용자 환경의 정보로 바꿉니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create appmirrorrelationship  
<name_of_appmirrorrelationship> --destination-app-vault  
<my_vault_name> --recurrence-rule <rule> --source-app  
<my_source_app> --source-app-vault <my_source_app_vault> -n  
<application_namespace>
```

8. (선택 사항) 대상 클러스터에서 복제 관계의 상태 및 상태를 확인합니다.

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath  
='{.status}' | jq
```

대상 클러스터로 페일오버합니다

Trident Protect를 사용하면 복제된 애플리케이션을 대상 클러스터로 페일오버할 수 있습니다. 이 절차는 복제 관계를 중지하고 대상 클러스터에서 앱을 온라인으로 전환합니다. Trident Protect는 소스 클러스터의 앱이 작동 중이었다면 중지하지 않습니다.

단계

1. 대상 클러스터에서 AppMirrorRelationship CR 파일(예 trident-protect-relationship.yaml:)을 편집하고 *spec.desiredState* 값을 로 변경합니다. Promoted
2. CR 파일을 저장합니다.
3. CR 적용:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (선택 사항) 페일오버된 애플리케이션에 필요한 보호 스케줄을 생성합니다.
5. (선택 사항) 복제 관계의 상태 및 상태를 확인합니다.

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath  
='{.status}' | jq
```

페일오버된 복제 관계를 다시 동기화합니다

재동기화 작업은 복제 관계를 다시 설정합니다. 재동기화 작업을 수행하면 원래 소스 애플리케이션이 실행 중인 애플리케이션이 되고 대상 클러스터에서 실행 중인 애플리케이션에 대한 변경 내용은 모두 삭제됩니다.

이 프로세스는 복제를 다시 설정하기 전에 대상 클러스터에서 앱을 중지합니다.



페일오버 중에 대상 애플리케이션에 기록된 모든 데이터가 손실됩니다.

단계

1. 선택 사항: 소스 클러스터에서 소스 애플리케이션의 스냅샷을 생성합니다. 이렇게 하면 소스 클러스터의 최신 변경 사항이 캡처됩니다.
2. 대상 클러스터에서 AppMirrorRelationship CR 파일(예 `trident-protect-relationship.yaml`)을 편집하고 `spec.desiredState` 값을 `Established` 로 변경합니다.
3. CR 파일을 저장합니다.
4. CR 적용:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. 대상 클러스터에서 페일오버된 애플리케이션을 보호하기 위해 보호 스케줄을 생성한 경우 제거하십시오. 남아 있는 스케줄은 볼륨 스냅샷에 장애를 일으킵니다.

페일오버된 복제 관계를 역방향으로 재동기화합니다

페일오버된 복제 관계를 역동기화하는 경우 대상 애플리케이션은 소스 애플리케이션이 되고 소스는 대상이 됩니다. 페일오버 중에 대상 애플리케이션에 대한 변경 사항은 유지됩니다.

단계

1. 원래 대상 클러스터에서 AppMirrorRelationship CR을 삭제합니다. 그러면 대상이 원본이 됩니다. 새 대상 클러스터에 남아 있는 보호 스케줄이 있는 경우 제거합니다.
2. 관계를 설정할 때 원래 사용했던 CR 파일을 반대 클러스터에 적용하여 복제 관계를 설정합니다.
3. 새 대상(원래 소스 클러스터)이 두 AppVault CRS로 구성되어 있는지 확인합니다.
4. 반대 클러스터에서 복제 관계를 설정하여 반대 방향에 대한 값을 구성합니다.

애플리케이션 복제 방향을 반대로 전환합니다

복제 방향을 반대로 바꾸면 Trident Protect는 애플리케이션을 대상 스토리지 백엔드로 이동하고 계속해서 원래 소스 스토리지 백엔드로 복제합니다. Trident Protect는 소스 애플리케이션을 중지하고 타겟 앱으로 페일오버하기 전에 데이터를 대상에 복제합니다.

이 경우 소스와 대상을 스와핑합니다.

단계

1. 소스 클러스터에서 종료 스냅샷을 생성합니다.

CR을 사용하여 종료 스냅샷을 생성합니다

a. 소스 애플리케이션에 대한 보호 정책 일정을 해제합니다.

b. ShutdownSnapshot CR 파일 생성:

i. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: trident-protect-shutdownsnapshot.yaml).

ii. 다음 특성을 구성합니다.

- **metadata.name:** (*required*) 사용자 정의 리소스의 이름입니다.
- *** spec.AppVaultRef *:** (*required*) 이 값은 원본 응용 프로그램에 대한 AppVault의 metadata.name 필드와 일치해야 합니다.
- **spec.ApplicationRef:** (*required*) 이 값은 소스 응용 프로그램 CR 파일의 metadata.name 필드와 일치해야 합니다.

YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-
c08a5dbe844e
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
```

c. 파일을 올바른 값으로 채운 후 trident-protect-shutdownsnapshot.yaml CR:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

CLI를 사용하여 종료 스냅샷을 생성합니다

a. 괄호 안의 값을 사용자 환경의 정보로 대체하여 종료 스냅샷을 만듭니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot> -n
<application_namespace>
```

2. 소스 클러스터에서 종료 스냅샷이 완료된 후 종료 스냅샷의 상태를 가져옵니다.

```
kubectl get shutdownsnapshot -n my-app-namespace  
<shutdown_snapshot_name> -o yaml
```

3. 소스 클러스터에서 다음 명령을 사용하여 * shutdownsnapshot.status.appArchivePath * 의 값을 찾고 파일 경로의 마지막 부분(basename라고도 함. 이것은 마지막 슬래시 다음에 모두 있음)을 기록합니다.

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o  
jsonpath='{.status.appArchivePath}'
```

4. 다음 변경 사항을 적용하여 새 대상 클러스터에서 새 소스 클러스터로 페일오버를 수행합니다.



페일오버 절차의 2단계에서 AppMirrorRelationship CR 파일에 필드를 포함하고 spec.promotedSnapshot 위의 3단계에서 기록한 기본 이름으로 값을 설정합니다.

5. 의 역동기화 단계를 **페일오버된 복제 관계를 역방향으로 재동기화합니다**수행합니다.

6. 새 소스 클러스터에서 보호 스케줄을 설정합니다.

결과

역방향 복제 때문에 다음 작업이 발생합니다.

- 원본 소스 앱의 Kubernetes 리소스에 대한 스냅샷이 생성됩니다.
- 앱의 Kubernetes 리소스를 삭제하여 원본 소스 앱의 Pod를 정상적으로 중지할 수 있습니다(PVC 및 PVS를 그대로 둡니다).
- 포드가 종료된 후 앱 볼륨의 스냅샷이 촬영되고 복제됩니다.
- SnapMirror 관계가 끊어져 타겟 볼륨이 읽기/쓰기 준비가 되었습니다.
- 앱의 Kubernetes 리소스는 원래 소스 애플리케이션이 종료된 후 복제된 볼륨 데이터를 사용하여 사전 종료 스냅샷에서 복구됩니다.
- 복제는 반대 방향으로 다시 설정됩니다.

애플리케이션을 원래 소스 클러스터로 페일백합니다

Trident Protect를 사용하면 다음 작업 순서를 사용하여 페일오버 작업 후에 "페일백"을 수행할 수 있습니다. 이 워크플로우에서 원래 복제 방향을 복구하면 Trident Protect는 복제 방향을 바꾸기 전에 애플리케이션 변경 내용을 원래 소스 애플리케이션으로 다시 복제(재동기화)합니다.

이 프로세스는 대상에 대한 페일오버를 완료한 관계로부터 시작되며 다음 단계를 포함합니다.

- 페일오버된 상태로 시작합니다.
- 복제 관계를 역방향으로 다시 동기화합니다.



일반 재동기화 작업을 수행하지 마십시오. 그러면 페일오버 절차 중에 대상 클러스터에 기록된 데이터가 삭제됩니다.

- 복제 방향을 반대로 바꿉니다.

단계

1. **페일오버된 복제 관계를 역방향으로 재동기화합니다** 다음 단계를 수행합니다.
2. **애플리케이션 복제 방향을 반대로 전환합니다** 다음 단계를 수행합니다.

복제 관계를 삭제합니다

언제든지 복제 관계를 삭제할 수 있습니다. 애플리케이션 복제 관계를 삭제하면 서로 관계가 없는 두 개의 개별 애플리케이션이 생성됩니다.

단계

1. 현재 대상 클러스터에서 AppMirrorRelationship CR을 삭제합니다.

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Trident Protect를 사용하여 애플리케이션을 마이그레이션합니다

백업 데이터를 복원하여 클러스터 간이나 다른 스토리지 클래스로 애플리케이션을 마이그레이션할 수 있습니다.



애플리케이션을 마이그레이션하면 애플리케이션에 대해 구성된 모든 실행 후크가 앱과 함께 마이그레이션됩니다. 복원 후 실행 후크가 있는 경우 복구 작업의 일부로 자동으로 실행됩니다.

백업 및 복원 작업

다음 시나리오에 대한 백업 및 복원 작업을 수행하려면 특정 백업 및 복원 작업을 자동화할 수 있습니다.

같은 클러스터에 클론 복제

애플리케이션을 동일한 클러스터에 복제하려면 스냅샷 또는 백업을 생성하고 데이터를 동일한 클러스터에 복구합니다.

단계

1. 다음 중 하나를 수행합니다.
 - a. **"스냅샷을 생성합니다"**..
 - b. **"백업을 생성합니다"**..
2. 동일한 클러스터에서 스냅샷 또는 백업을 생성했는지에 따라 다음 중 하나를 수행합니다.
 - a. **"스냅샷에서 데이터를 복원합니다"**..
 - b. **"백업에서 데이터를 복원합니다"**..

다른 클러스터에 클론 복제

애플리케이션을 다른 클러스터에 클론 생성(클러스터 간 클론 수행)하려면 소스 클러스터에 백업을 생성한 다음 다른 클러스터에 백업을 복원합니다. 대상 클러스터에 Trident Protect가 설치되어 있는지 확인하십시오.



을 사용하여 서로 다른 클러스터 간에 애플리케이션을 복제할 수 **"SnapMirror 복제"** 있습니다.

단계

1. **"백업을 생성합니다"**..
2. 백업이 포함된 객체 스토리지 버킷의 AppVault CR이 대상 클러스터에 구성되어 있는지 확인합니다.
3. 대상 클러스터에서 **"백업에서 데이터를 복원합니다"**.

한 스토리지 클래스에서 다른 스토리지 클래스로 애플리케이션을 마이그레이션합니다

대상 스토리지 클래스에 백업을 복원하여 애플리케이션을 한 스토리지 클래스에서 다른 스토리지 클래스로 마이그레이션할 수 있습니다.

예를 들어 복원 CR에서 비밀 제외):

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

CR을 사용하여 스냅샷을 복원합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-snapshot-restore-cr.yaml` 지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
 - **metadata.name:** *(required)* 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.appArchivePath *:** 스냅샷 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

```
kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
```

- *** spec.appVaultRef *:** *(required)* 스냅샷 내용이 저장된 AppVault의 이름입니다.
- **spec.namespaceMapping:** 복구 작업의 소스 네임스페이스를 대상 네임스페이스에 매핑하는 것입니다. `my-source-namespace` 및 `my-destination-namespace` 사용자 환경의 정보로 바꿉니다.

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

3. 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.
 - **resourceFilter.resourceSelectionCriteria:** (필터링에 필요) `include` or `exclude` `resourceMatchers`에 정의된 리소스를 포함하거나 제외하는 데 사용합니다. 다음 `resourceMatchers` 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
 - **resourceFilter.resourceMatchers:** `resourceMatcher` 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.
 - **resourceMatchers[].group:** *(Optional)* 필터링할 리소스의 그룹입니다.
 - *** resourceMatchers [].kind *:** *(Optional)* 필터링할 리소스의 종류입니다.
 - **resourceMatchers [].version:** *(Optional)* 필터링할 리소스의 버전입니다.

- **resourceMatchers[].names:** *(Optional)* 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
- ***resourceMatchers [].namespaces *:** *(Optional)* 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- **resourceMatchers [].labelSelectors:** *(Optional)* 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "[Kubernetes 문서](#)" 예를 들면 다음과 `"trident.netapp.io/os=linux"` 같습니다.

예를 들면 다음과 같습니다.

```
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. 파일을 올바른 값으로 채운 후 trident-protect-snapshot-restore-cr.yaml CR:

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

CLI를 사용하여 스냅샷을 복원합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 스냅샷을 다른 네임스페이스로 복원합니다.

- snapshot`인수에 네임스페이스 및 스냅샷 이름이 형식으로 `<namespace>/<name>` 사용됩니다.
- 이 namespace-mapping 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 올바른 대상 네임스페이스에 형식 `source1:dest1,source2:dest2`으로 매핑합니다.

예를 들면 다음과 같습니다.

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Trident Protect 실행 후크를 관리합니다

실행 후크는 관리되는 앱의 데이터 보호 작업과 함께 실행되도록 구성할 수 있는 사용자 지정 작업입니다. 예를 들어 데이터베이스 앱이 있는 경우 실행 후크를 사용하여 스냅샷 전에 모든 데이터베이스 트랜잭션을 일시 중지하고 스냅샷이 완료된 후 트랜잭션을 다시 시작할 수 있습니다. 따라서 애플리케이션 정합성이 보장되는 스냅샷이 보장됩니다.

실행 후크 유형

Trident Protect는 실행 가능한 시기에 따라 다음과 같은 유형의 실행 후크를 지원합니다.

- 사전 스냅샷
- 사후 스냅샷
- 사전 백업
- 백업 후
- 사후 복원
- 장애 조치 후

실행 순서

데이터 보호 작업이 실행되면 실행 후크 이벤트가 다음 순서로 발생합니다.

1. 해당되는 모든 사용자 정의 사전 작업 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사전 작업 후크를 만들고 실행할 수 있지만, 이 후크의 실행 순서는 보장되거나 구성할 수 없습니다.
2. 해당하는 경우 파일 시스템 작동이 중지됩니다. ["Trident Protect를 사용하여 파일 시스템 중지를 구성하는 방법에 대해 자세히 알아보십시오"](#)..
3. 데이터 보호 작업이 수행됩니다.
4. 해당되는 경우 동결된 파일 시스템은 동결 해제됩니다.
5. 해당되는 모든 사용자 지정 작업 후 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사후 작업 후크를 만들고 실행할 수 있지만 작업 후 후크의 실행 순서는 보장되거나 구성할 수 없습니다.

같은 유형의 실행 후크를 여러 개 생성하는 경우(예: 사전 스냅샷) 해당 후크의 실행 순서는 보장되지 않습니다. 그러나 다른 유형의 후크를 실행하는 순서는 보장됩니다. 예를 들어, 다음과 같은 순서로 서로 다른 유형의 후크가 모두 있는 구성을 실행할 수 있습니다.

1. 사전 스냅샷 후크가 실행되었습니다
2. 사후 스냅샷 후크가 실행되었습니다

3. 예비 후크가 실행되었습니다

4. 백업 후 후크가 실행되었습니다



위의 순서 예제는 기존 스냅샷을 사용하지 않는 백업을 실행하는 경우에만 적용됩니다.



운영 환경에서 실행 후크 스크립트를 사용하려면 항상 해당 스크립트를 테스트해야 합니다. 'kubbeck exec' 명령을 사용하여 스크립트를 편리하게 테스트할 수 있습니다. 운영 환경에서 실행 후크를 사용하도록 설정한 후 결과 스냅샷과 백업을 테스트하여 정합성이 보장되는지 확인합니다. 앱을 임시 네임스페이스에 클론 복제하고, 스냅샷 또는 백업을 복원한 다음 앱을 테스트하여 이 작업을 수행할 수 있습니다.



스냅샷 사전 실행 후크가 Kubernetes 리소스를 추가, 변경 또는 제거하는 경우 스냅샷 또는 백업과 이후의 복원 작업에 변경 사항이 포함됩니다.

사용자 정의 실행 후크에 대한 중요 참고 사항

앱에 대한 실행 후크를 계획할 때 다음 사항을 고려하십시오.

- 실행 후크는 스크립트를 사용하여 작업을 수행해야 합니다. 많은 실행 후크가 동일한 스크립트를 참조할 수 있습니다.
- Trident Protect는 실행 후크를 사용하는 스크립트를 실행 가능한 셸 스크립트 형식으로 작성해야 합니다.
- 스크립트 크기는 96KB로 제한됩니다.
- Trident Protect는 실행 후크 설정과 일치하는 기준을 사용하여 스냅샷, 백업 또는 복구 작업에 적용할 수 있는 후크를 결정합니다.



실행 후크는 실행 중인 응용 프로그램의 기능을 줄이거나 완전히 비활성화하기 때문에 사용자 지정 실행 후크가 실행되는 시간을 최소화해야 합니다. 연결된 실행 후크와 함께 백업 또는 스냅샷 작업을 시작한 다음 취소하면 백업 또는 스냅샷 작업이 이미 시작된 경우에도 후크를 실행할 수 있습니다. 즉, 백업 후 실행 후크에 사용되는 논리는 백업이 완료된 것으로 가정할 수 없습니다.

실행 후크 필터

응용 프로그램의 실행 후크를 추가하거나 편집할 때 실행 후크에 필터를 추가하여 후크가 일치할 컨테이너를 관리할 수 있습니다. 필터는 모든 컨테이너에서 동일한 컨테이너 이미지를 사용하는 응용 프로그램에 유용하지만 각 이미지를 다른 용도(예: Elasticsearch)로 사용할 수 있습니다. 필터를 사용하면 실행 후크가 실행되는 시나리오를 만들 수 있습니다. 단, 모든 동일한 컨테이너를 실행하는 것은 아닙니다. 단일 실행 후크에 대해 여러 개의 필터를 만들면 논리적 필터 및 연산자와 결합됩니다. 실행 후크당 최대 10개의 활성 필터를 사용할 수 있습니다.

실행 후크에 추가하는 각 필터는 클러스터의 컨테이너와 일치시키기 위해 정규식을 사용합니다. 후크가 컨테이너와 일치하면 후크는 해당 컨테이너에서 연결된 스크립트를 실행합니다. 필터에 대한 정규식은 정규식 2(RE2) 구문을 사용합니다. 이 구문은 일치 목록에서 컨테이너를 제외하는 필터를 만드는 것을 지원하지 않습니다. Trident Protect가 실행 후크 필터에서 정규식을 지원하는 구문에 대한 자세한 내용은 ["정규식 2\(RE2\) 구문 지원"](#)을 참조하십시오.



복원 또는 클론 작업 후에 실행되는 실행 후크에 네임스페이스 필터를 추가하고 복원 또는 클론 소스와 대상이 서로 다른 네임스페이스에 있는 경우 네임스페이스 필터는 대상 네임스페이스에만 적용됩니다.

실행 후크 예

를 방문하여 "[NetApp Verda GitHub 프로젝트](#)" Apache Cassandra 및 Elasticsearch와 같은 인기 앱의 실제 실행 후크를 다운로드하십시오. 예제를 보고 사용자 지정 실행 후크를 구조화하는 아이디어를 얻을 수도 있습니다.

실행 후크를 만듭니다

Trident Protect를 사용하여 앱에 대한 사용자 지정 실행 후크를 생성할 수 있습니다. 실행 후크를 만들려면 소유자, 관리자 또는 구성원 권한이 있어야 합니다.

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-hook.yaml` 지정합니다.
2. 다음 특성을 Trident Protect 환경과 클러스터 구성과 일치하도록 구성합니다.
 - **metadata.name:** *(required)* 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.applicationRef *:** *(required)* 실행 후크를 실행할 응용 프로그램의 Kubernetes 이름입니다.
 - *** spec.stage *:** *(required)* 실행 후크가 실행되어야 하는 작업 중 단계를 나타내는 문자열입니다. 가능한 값:
 - 사전
 - 게시
 - *** spec.action *:** *(required)* 지정된 실행 후크 필터가 일치한다고 가정할 때 실행 후크가 수행할 작업을 나타내는 문자열입니다. 가능한 값:
 - 스냅샷
 - 백업
 - 복원
 - 파일오버
 - **spec.enabled:** *(Optional)* 이 실행 후크의 활성화 여부를 나타냅니다. 지정하지 않으면 기본값은 true 입니다.
 - **spec.hookSource:** *(required)* base64로 인코딩된 후크 스크립트를 포함하는 문자열입니다.
 - **spec.timeout:** *(Optional)* 실행 후크가 실행될 수 있는 시간을 분 단위로 정의하는 숫자입니다. 최소값은 1분이고, 지정하지 않은 경우 기본값은 25분입니다.
 - **spec.arguments:** *(Optional)* 실행 후크에 지정할 수 있는 인수의 YAML 목록입니다.
 - *** spec.matchingCriteria *:** *(Optional)* 각 쌍이 실행 후크 필터를 구성하는 조건 키 값 쌍의 선택적 목록입니다. 실행 후크당 최대 10개의 필터를 추가할 수 있습니다.
 - *** spec.matchingCriteria.type *:** *(Optional)* 실행 후크 필터 유형을 식별하는 문자열입니다. 가능한 값:
 - 컨테이너이미지
 - 컨테이너명
 - PodName을 선택합니다
 - PodLabel을 선택합니다
 - 이름 이름 이름
 - *** spec.matchingCriteria.value *:** *(Optional)* 실행 후크 필터 값을 식별하는 문자열 또는 정규식입니다.

YAML 예:

```

apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNoYAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production

```

3. CR 파일을 올바른 값으로 채운 후 CR:

```
kubectl apply -f trident-protect-hook.yaml
```

CLI를 사용합니다

단계

1. 괄호 안의 값을 사용자 환경의 정보로 대체하여 실행 후크를 만듭니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create exehook <my_exec_hook_name> --action  
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>  
--source-file <script-file> -n <application_namespace>
```

실행 후크를 수동으로 실행합니다

테스트 목적으로 실행 후크를 수동으로 실행하거나 실패 후 수동으로 후크를 다시 실행해야 하는 경우 실행할 수 있습니다. 실행 후크를 수동으로 실행하려면 소유자, 관리자 또는 구성원 권한이 있어야 합니다.

실행 후크를 수동으로 실행하는 작업은 다음 두 가지 기본 단계로 구성됩니다.

1. 리소스 백업을 생성하여 리소스를 수집하고 해당 백업을 생성하여 후크가 실행될 위치를 결정합니다
2. 백업에 대해 실행 후크를 실행합니다

1단계: 리소스 백업을 생성합니다



CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-resource-backup.yaml` 지정합니다.
2. 다음 특성을 Trident Protect 환경과 클러스터 구성과 일치하도록 구성합니다.
 - **metadata.name:** *(required)* 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.applicationRef *:** *(required)* 리소스 백업을 생성할 애플리케이션의 Kubernetes 이름입니다.
 - *** spec.appVaultRef *:** *(required)* 백업 내용이 저장된 AppVault의 이름입니다.
 - *** spec.appArchivePath *:** 백업 콘텐츠가 저장되는 AppVault 내부의 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

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

YAML 예:

```
---  
apiVersion: protect.trident.netapp.io/v1  
kind: ResourceBackup  
metadata:  
  name: example-resource-backup  
spec:  
  applicationRef: my-app-name  
  appVaultRef: my-appvault-name  
  appArchivePath: example-resource-backup
```

3. CR 파일을 올바른 값으로 채운 후 CR:

```
kubectl apply -f trident-protect-resource-backup.yaml
```

CLI를 사용합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 백업을 만듭니다. 예를 들면 다음과 같습니다.

```
tridentctl protect create resourcebackup <my_backup_name> --app  
<my_app_name> --appvault <my_appvault_name> -n  
<my_app_namespace> --app-archive-path <app_archive_path>
```

2. 백업 상태를 봅니다. 작업이 완료될 때까지 이 예제 명령을 반복해서 사용할 수 있습니다.

```
tridentctl protect get resourcebackup -n <my_app_namespace>  
<my_backup_name>
```

3. 백업이 성공했는지 확인합니다.

```
kubectl describe resourcebackup <my_backup_name>
```

단계 2: 실행 후크를 실행합니다

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-hook-run.yaml` 지정합니다.
2. 다음 특성을 Trident Protect 환경과 클러스터 구성과 일치하도록 구성합니다.
 - **metadata.name:** *(required)* 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
 - *** spec.applicationRef *:** *(required)* 이 값이 1단계에서 만든 ResourceBackup CR의 응용 프로그램 이름과 일치하는지 확인합니다.
 - *** spec.appVaultRef *:** *(required)* 이 값이 1단계에서 작성한 ResourceBackup CR의 appVaultRef와 일치하는지 확인합니다.
 - **spec.appArchivePath:** 이 값이 1단계에서 만든 ResourceBackup CR의 appArchivePath 와 일치하는지 확인하십시오.

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

- *** spec.action *:** *(required)* 지정된 실행 후크 필터가 일치한다고 가정할 때 실행 후크가 수행할 작업을 나타내는 문자열입니다. 가능한 값:
 - 스냅샷
 - 백업
 - 복원
 - 페일오버
- *** spec.stage *:** *(required)* 실행 후크가 실행되어야 하는 작업 중 단계를 나타내는 문자열입니다. 이 후크 실행은 다른 단계에서 후크를 실행하지 않습니다. 가능한 값:
 - 사전
 - 게시

YAML 예:

```

---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover

```

3. CR 파일을 올바른 값으로 채운 후 CR:

```
kubectl apply -f trident-protect-hook-run.yaml
```

CLI를 사용합니다

단계

1. 수동 실행 후크 실행 요청을 생성합니다.

```

tridentctl protect create exehookrun <my_exec_hook_run_name>
-n <my_app_namespace> --action snapshot --stage <pre_or_post>
--app <my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>

```

2. execution hook run의 상태를 확인한다. 작업이 완료될 때까지 이 명령을 반복적으로 실행할 수 있습니다.

```

tridentctl protect get exehookrun -n <my_app_namespace>
<my_exec_hook_run_name>

```

3. exehookrun 객체를 설명하여 최종 세부 정보 및 상태를 확인합니다.

```

kubectl -n <my_app_namespace> describe exehookrun
<my_exec_hook_run_name>

```

저작권 정보

Copyright © 2025 NetApp, Inc. All Rights Reserved. 미국에서 인쇄된 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이선스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이선스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이선스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이선스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 <http://www.netapp.com/TM>에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.