애플리케이션 관리 및 보호

Trident

NetApp
July 01, 2026

목차

애플리케이션 관리 및 보호

Trident Protect AppVault 객체를 사용하여 버킷을 관리하세요.

Trident Protect의 버킷 사용자 정의 리소스(CR)는 AppVault로 알려져 있습니다. AppVault 오브젝트는 스토리지 버킷을 Kubernetes 워크플로에서 선언적으로 표현한 것입니다. AppVault CR에는 백업, 스냅샷, 복원 작업 및 SnapMirror 복제와 같은 보호 작업에 버킷을 사용하는 데 필요한 구성이 포함되어 있습니다. 관리자만 AppVaults를 생성할 수 있습니다.

애플리케이션에 대한 데이터 보호 작업을 수행할 때는 AppVault CR을 수동으로 또는 명령줄에서 생성해야 합니다. AppVault CR은 사용 환경에 따라 다르며, 이 페이지의 예제를 참고하여 AppVault CR을 생성할 수 있습니다.

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

AppVault 인증 및 암호를 구성합니다

AppVault CR을 생성하기 전에 AppVault와 선택한 데이터 이동 도구가 공급자 및 관련 리소스에 인증할 수 있는지 확인하십시오.

Data Mover 리포지토리 암호

CR 또는 Trident Protect CLI 플러그인을 사용하여 AppVault 객체를 생성할 때 Restic 및 Kopia 암호화에 대한 사용자 지정 암호가 포함된 Kubernetes 시크릿을 지정할 수 있습니다. 시크릿을 지정하지 않으면 Trident Protect는 기본 암호를 사용합니다.

- AppVault CR을 수동으로 생성할 때는 **spec.dataMoverPasswordSecretRef** 필드를 사용하여 비밀 키를 지정하십시오.
- Trident Protect CLI를 사용하여 AppVault 객체를 생성할 때 `--data-mover-password-secret-ref` 인수를 사용하여 비밀 키를 지정하십시오.

데이터 이동 저장소 암호 보안 암호를 생성합니다

다음 예제를 사용하여 암호 비밀 키를 생성하세요. AppVault 객체를 생성할 때 Trident Protect가 이 비밀 키를 사용하여 데이터 이동 저장소와 인증하도록 설정할 수 있습니다.

- 사용하는 데이터 이동 도구에 따라 해당 데이터 이동 도구에 대한 해당 암호만 포함하면 됩니다. 예를 들어 Restic을 사용하고 향후 Kopia를 사용할 계획이 없는 경우 시크릿을 생성할 때 Restic 암호만 포함하면 됩니다.
- 암호를 안전한 곳에 보관하십시오. 동일한 클러스터 또는 다른 클러스터에서 데이터를 복원할 때 암호가 필요합니다. 클러스터 또는 `trident-protect` 네임스페이스가 삭제되면 암호 없이는 백업이나 스냅샷을 복원할 수 없습니다.

CR 사용

```
---
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
```

CLI 사용

```
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

Trident Protect를 사용하여 Amazon S3, Generic S3, "StorageGrid S3" 또는 "ONTAP S3"와 같은 S3 호환 스토리지에 액세스할 때는 제공하는 사용자 자격 증명에 버킷에 액세스하는 데 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 "Amazon S3 문서"의 예제를 참조하십시오.

Amazon S3(AWS) 인증을 위한 EKS Pod Identity

Trident Protect는 Kopia 데이터 이동 작업에서 EKS Pod Identity를 지원합니다. 이 기능을 통해 AWS 자격 증명을 Kubernetes 시크릿에 저장하지 않고도 S3 버킷에 안전하게 액세스할 수 있습니다.

Trident Protect를 사용한 EKS Pod Identity 요구 사항

Trident Protect에서 EKS Pod Identity를 사용하기 전에 다음 사항을 확인하십시오.

- EKS 클러스터에서 Pod Identity가 활성화되었습니다.
- 필요한 S3 버킷 권한이 있는 IAM 역할을 생성했습니다. 자세한 내용은 "S3 호환 스토리지 IAM 권한"를 참조하십시오.
- IAM 역할은 다음 Trident Protect 서비스 계정과 연결되어 있습니다.
  - `<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 "AWS EKS Pod Identity 문서"을 참조하십시오.

AppVault 구성 EKS Pod Identity를 사용하는 경우 `useIAM: true` 플래그를 사용하여 명시적인 자격 증명 대신 AppVault CR을 구성하십시오.

```
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 예시입니다. 다음 예시를 사용하여 각 클라우드 공급자의 자격 증명에 대한 키를 생성할 수 있습니다.

Google Cloud

```
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 Azure

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

Generic 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

```
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 예시

다음은 각 공급자에 대한 AppVault 정의 예시입니다.

AppVault CR 예시

다음 CR 예제를 사용하여 각 클라우드 공급자에 대한 AppVault 객체를 생성할 수 있습니다.

- 선택적으로 Restic 및 Kopia 리포지토리 암호화에 대한 사용자 지정 암호가 포함된 Kubernetes 시크릿을 지정할 수 있습니다. 자세한 내용은 Data Mover 리포지토리 암호을 참조하십시오.
- Amazon S3(AWS) AppVault 객체의 경우 선택적으로 sessionToken을 지정할 수 있으며, 이는 Single Sign-On(SSO)을 인증에 사용하는 경우 유용합니다. 이 토큰은 클라우드 공급자를 위한 AppVault 키 생성 예에서 공급자용 키를 생성할 때 생성됩니다.
- S3 AppVault 객체의 경우 `spec.providerConfig.S3.proxyURL` 키를 사용하여 아웃바운드 S3 트래픽에 대한 이그레스 프록시 URL을 선택적으로 지정할 수 있습니다.

Google Cloud

```
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, `providerCredentials` 섹션을 제거하고 `useIAM: true`를 `s3` 구성 아래에 추가할 수 있습니다.

Microsoft Azure

```
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
```

Generic 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

```
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 예제

다음 CLI 명령 예제를 사용하여 각 공급자에 대한 AppVault CR을 생성할 수 있습니다.

- 선택적으로 Restic 및 Kopia 리포지토리 암호화에 대한 사용자 지정 암호가 포함된 Kubernetes 시크릿을 지정할 수 있습니다. 자세한 내용은 Data Mover 리포지토리 암호을 참조하십시오.
- S3 AppVault 객체의 경우 `--proxy-url <ip_address:port>` 인수를 사용하여 아웃바운드 S3 트래픽에 대한 이그레스 프록시 URL을 선택적으로 지정할 수 있습니다.

Google Cloud

```
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 Azure

```
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
```

Generic 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

```
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
```

지원되는 `providerConfig.s3` 구성 옵션

S3 공급자 구성 옵션은 다음 표를 참조하십시오.

매개변수	설명	기본값	예
`providerConfig.s3.skipCertValidation`	SSL/TLS 인증서 확인을 비활성화합니다.	거짓	"true", "false"
`providerConfig.s3.secure`	S3 엔드포인트와의 안전한 HTTPS 통신을 활성화합니다.	true	"true", "false"
`providerConfig.s3.proxyURL`	S3에 연결하는 데 사용되는 프록시 서버의 URL을 지정하십시오.	None	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	SSL/TLS 검증을 위해 사용자 지정 루트 CA 인증서를 제공하십시오.	None	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	S3 버킷 액세스를 위한 IAM 인증을 활성화합니다. EKS Pod Identity에 적용됩니다.	거짓	참, 거짓

AppVault 정보를 확인하세요

Trident Protect CLI 플러그인을 사용하여 클러스터에 생성한 AppVault 객체에 대한 정보를 볼 수 있습니다.

단계

1. AppVault 객체의 내용을 보려면 다음을 수행하십시오.

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

예제 출력:

```
+-------------+-------+----------+-----------------------------
+---------------------------+
|   CLUSTER   |  APP  |   TYPE   |            NAME             |
TIMESTAMP         |
+-------------+-------+----------+-----------------------------
+---------------------------+
|             | 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 플래그 `--show-paths`를 사용하십시오.

표의 첫 번째 열에 있는 클러스터 이름은 Trident Protect helm 설치 시 클러스터 이름이 지정된 경우에만 사용할 수 있습니다. 예를 들면 다음과 같습니다: `--set clusterName=production1`.

AppVault 제거

AppVault 개체는 언제든지 삭제할 수 있습니다.

AppVault CR에서 `finalizers` 키를 AppVault 객체를 삭제하기 전에 제거하지 마십시오. 키를 제거하면 AppVault 버킷에 잔여 데이터가 남고 클러스터에 고아 리소스가 생성될 수 있습니다.

시작하기 전에

삭제하려는 AppVault에서 사용 중인 모든 스냅샷 및 백업 CR을 삭제했는지 확인하십시오.

Kubernetes CLI를 사용하여 **AppVault** 제거

1. AppVault 객체를 제거합니다. 제거할 AppVault 객체의 이름으로 `appvault-name`을(를) 바꿉니다.

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

Trident Protect CLI를 사용하여 **AppVault** 제거

1. AppVault 객체를 제거합니다. 제거할 AppVault 객체의 이름으로 `appvault-name`을(를) 바꿉니다.

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

Trident Protect를 사용하여 관리용 애플리케이션 정의

Trident Protect로 관리할 애플리케이션을 정의하려면 애플리케이션 CR과 연결된 AppVault CR을 생성하면 됩니다.

AppVault CR 생성

애플리케이션에 대한 데이터 보호 작업을 수행할 때 사용할 AppVault CR을 생성해야 하며, AppVault CR은 Trident Protect가 설치된 클러스터에 있어야 합니다. AppVault CR은 사용 환경에 따라 다릅니다. AppVault CR의 예는 "AppVault 사용자 지정 리소스."을 참조하십시오.

애플리케이션 정의

Trident Protect로 관리할 각 애플리케이션을 정의해야 합니다. 애플리케이션 CR을 수동으로 생성하거나 Trident Protect CLI를 사용하여 관리할 애플리케이션을 정의할 수 있습니다.

CR을 사용하여 애플리케이션 추가

단계

1. 타겟 애플리케이션 CR 파일을 생성합니다.
   a. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `maria-app.yaml`).
   b. 다음 속성을 구성하십시오.
      - **metadata.name**: (필수) 애플리케이션 사용자 지정 리소스의 이름입니다. 보호 작업에 필요한 다른 CR 파일에서 이 값을 참조하므로 선택한 이름을 기억해 두십시오.
      - **spec.includedNamespaces**: (필수) 네임스페이스 및 레이블 선택기를 사용하여 애플리케이션에서 사용하는 네임스페이스와 리소스를 지정합니다. 애플리케이션 네임스페이스는 이 목록에 반드시 포함되어야 합니다. 레이블 선택기는 선택 사항이며, 지정된 각 네임스페이스 내의 리소스를 필터링하는 데 사용할 수 있습니다.
      - **spec.includedClusterScopedResources**: (선택 사항) 이 속성을 사용하여 애플리케이션 정의에 포함할 클러스터 범위 리소스를 지정합니다. 이 속성을 사용하면 그룹, 버전, 종류 및 레이블을 기준으로 이러한 리소스를 선택할 수 있습니다.
        - **groupVersionKind**: (필수) 클러스터 범위 리소스의 API 그룹, 버전 및 종류를 지정합니다.
        - **labelSelector**: (선택 사항) 레이블을 기준으로 클러스터 범위 리소스를 필터링합니다.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (선택 사항) 이 어노테이션은 KubeVirt 환경과 같이 스냅샷 전에 파일 시스템이 동결되는 가상 머신에서 정의된 애플리케이션에만 적용됩니다. 이 애플리케이션이 스냅샷 중에 파일 시스템에 쓸 수 있는지 여부를 지정합니다. true로 설정하면 애플리케이션은 전역 설정을 무시하고 스냅샷 중에 파일 시스템에 쓸 수 있습니다. false로 설정하면 애플리케이션은 전역 설정을 무시하고 스냅샷 중에 파일 시스템이 동결됩니다. 지정되었지만 애플리케이션 정의에 가상 머신이 없는 경우 어노테이션은 무시됩니다. 지정하지 않으면 애플리케이션은 "글로벌 Trident Protect 동결 설정"를 따릅니다.

애플리케이션이 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) 필요한 경우 동일한 CR에 리소스 필터링을 추가하여 특정 리소스를 포함하거나 제외할 수 있습니다.

   ◦ 일반 필터 예:

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `Include` 또는 `Exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.

     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
     - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 그룹입니다.
     - **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.

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

> ⓘ `resourceFilter`와 `labelSelector`가 모두 사용될 경우, `resourceFilter`가 먼저 실행된 다음 `labelSelector`가 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"]
```

◦ **PVC 전용 필터 예:**

PVC 전용 애플리케이션을 정의하려면 리소스 필터에 `PersistentVolume`와 `VolumeSnapshotClass`도 포함해야 합니다. 스냅샷 및 백업 작업은 `PersistentVolume`(각 PVC에 바인딩된 클러스터 범위 볼륨)과 `VolumeSnapshotClass`(스냅샷 드라이버)에 따라 달라지며, 이들이 없으면 실패합니다. 예를 들면 다음과 같습니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-pvc-app
  namespace: my-app-namespace
spec:
  includedNamespaces:
  - namespace: my-app-namespace
  resourceFilter:
    resourceMatchers:
    - kind: PersistentVolumeClaim
      version: v1
    - kind: PersistentVolume
      version: v1
    - kind: VolumeSnapshotClass
      version: v1
    resourceSelectionCriteria: Include
```

2. 사용 환경에 맞는 애플리케이션 CR을 생성한 후 CR을 적용합니다. 예를 들면 다음과 같습니다.

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

단계

1. 다음 예제 중 하나를 사용하여 애플리케이션 정의를 생성하고 적용하세요. 괄호 안의 값은 사용자 환경에 맞는 정보로 바꿔야 합니다. 예제에 표시된 인수를 쉼표로 구분한 목록을 사용하여 네임스페이스와 리소스를 애플리케이션 정의에 포함할 수 있습니다.

   앱을 생성할 때 스냅샷 중에 애플리케이션이 파일 시스템에 쓸 수 있는지 여부를 지정하는 어노테이션을 선택적으로 사용할 수 있습니다. 이는 KubeVirt 환경과 같이 스냅샷 전에 파일 시스템이 고정되는 가상 머신에서 정의된 애플리케이션에만 적용됩니다. 어노테이션을 `true`로 설정하면 애플리케이션은 전역 설정을 무시하고 스냅샷 중에 파일 시스템에 쓸 수 있습니다. 어노테이션을 `false`로 설정하면 애플리케이션은 전역 설정을 무시하고 스냅샷 중에 파일 시스템이 고정됩니다. 어노테이션을 사용했지만 애플리케이션 정의에 가상 머신이 없는 경우 어노테이션은 무시됩니다. 어노테이션을 사용하지 않으면 애플리케이션은 "글로벌 Trident Protect 동결 설정"를 따릅니다.

   CLI를 사용하여 애플리케이션을 생성할 때 주석을 지정하려면 `--annotation` 플래그를 사용할 수 있습니다.

   - 애플리케이션을 생성하고 파일 시스템 동결 동작에 대한 글로벌 설정을 사용하십시오.

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

- 애플리케이션을 생성하고 파일 시스템 고정 동작에 대한 로컬 애플리케이션 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">
```

- `--resource-filter-include` 및 `--resource-filter-exclude` 플래그를 사용하여 다음 예와 같이 그룹, 종류, 버전, 레이블, 이름 및 네임스페이스와 같은 `resourceSelectionCriteria`을(를) 기반으로 리소스를 포함하거나 제외할 수 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"]}]'
```

- PVC 전용 애플리케이션을 정의하려면 리소스 필터에 `PersistentVolume`와 `VolumeSnapshotClass``도 포함해야 합니다. 스냅샷 및 백업 작업은 `PersistentVolume`(각 PVC에 바인딩된 클러스터 범위 볼륨)과 `VolumeSnapshotClass`(스냅샷 드라이버)에 따라 달라지며, 이들이 없으면 실패합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create app my-pvc-app --namespaces <my-app-
namespace> --resource-filter-include
'[{"Kind":"PersistentVolumeClaim","Version":"v1"},{"Kind":"Persis
tentVolume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Versio
n":"v1"}]' -n <my-app-namespace>
```

Trident Protect를 사용하여 애플리케이션을 보호하세요.

Trident Protect에서 관리하는 모든 앱은 자동 보호 정책을 사용하거나 임시로 스냅샷 및 백업을 생성하여 보호할 수 있습니다.

Trident Protect를 구성하면 데이터 보호 작업 중에 파일 시스템을 동결 및 해제할 수 있습니다. "Trident Protect를 사용한 파일 시스템 동결 구성에 대해 자세히 알아보십시오".

온디맨드 스냅샷 생성

언제든지 필요에 따라 스냅샷을 생성할 수 있습니다.

클러스터 범위 리소스는 애플리케이션 정의에서 명시적으로 참조되거나 애플리케이션 네임스페이스를 참조하는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

CR을 사용하여 스냅샷 생성

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-snapshot-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: 스냅샷을 생성할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 스냅샷 콘텐츠(메타데이터)를 저장할 AppVault의 이름입니다.
   - **spec.reclaimPolicy**: (선택 사항) 스냅샷 CR이 삭제될 때 스냅샷의 AppArchive에 어떤 일이 발생하는지 정의합니다. 즉, `Retain`로 설정하더라도 스냅샷은 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. `trident-protect-snapshot-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI를 사용하여 스냅샷 생성

단계

1. 스냅샷을 생성할 때 괄호 안의 값을 사용자 환경 정보로 바꾸세요. 예를 들면 다음과 같습니다.

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

온디맨드 백업 생성

언제든지 앱을 백업할 수 있습니다.

> 클러스터 범위 리소스는 애플리케이션 정의에서 명시적으로 참조되거나 애플리케이션 네임스페이스를 참조하는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

시작하기 전에

장시간 실행되는 s3 백업 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 백업 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
- AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS IAM 문서"를 참조하십시오.

CR을 사용하여 백업 생성

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-backup-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 백업할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 백업 콘텐츠를 저장할 AppVault의 이름입니다.
   - **spec.dataMover**: (선택 사항) 백업 작업에 사용할 백업 도구를 나타내는 문자열입니다. 가능한 값(대소문자 구분):
     - `Restic`
     - `Kopia` (기본값)
   - **spec.reclaimPolicy**: (선택 사항) 백업이 클레임에서 해제될 때 발생하는 상황을 정의합니다. 가능한 값:
     - `Delete`
     - `Retain` (기본값)
   - **spec.snapshotRef**: (선택 사항): 백업 소스로 사용할 스냅샷의 이름입니다. 지정하지 않으면 임시 스냅샷이 생성되어 백업됩니다.

     YAML 예:

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

3. `trident-protect-backup-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI를 사용하여 백업 생성

단계

1. 백업을 생성할 때 괄호 안의 값을 사용자 환경 정보로 바꿔주세요. 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>
```

선택적으로 `--full-backup` 플래그를 사용하여 백업을 비증분 방식으로 수행할지 여부를 지정할 수 있습니다. 기본적으로 모든 백업은 증분 백업입니다. 이 플래그를 사용하면 백업이 비증분 방식으로 수행됩니다. 복원과 관련된 위험을 최소화하려면 정기적으로 전체 백업을 수행하고 전체 백업 사이에 증분 백업을 수행하는 것이 좋습니다.

지원되는 백업 주석

다음 표에서는 백업 CR을 생성할 때 사용할 수 있는 주석에 대해 설명합니다.

주석	유형	설명	기본값
protect.trident.netapp.io/full-backup	문자열	백업을 비증분 백업으로 수행할지 여부를 지정합니다. `true`로 설정하여 비증분 백업을 생성합니다. 복원과 관련된 위험을 최소화하려면 정기적으로 전체 백업을 수행하고 전체 백업 사이에 증분 백업을 수행하는 것이 좋습니다.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	문자열	전체 스냅샷 작업이 완료되는 데 허용되는 최대 시간입니다.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	문자열	볼륨 스냅샷이 사용 가능한 상태에 도달하는 데 허용되는 최대 시간입니다.	"30분"
protect.trident.netapp.io/volume-snapshots-created-timeout	문자열	볼륨 스냅샷을 생성하는 데 허용되는 최대 시간입니다.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	문자열	새로 생성된 PersistentVolumeClaims(PVC)가 `Bound` 단계에 도달하기까지 기다리는 최대 시간(초)입니다. 이 시간이 지나면 작업이 실패합니다.	"1200" (20분)

데이터 보호 일정 생성

보호 정책은 정의된 일정에 따라 스냅샷, 백업 또는 둘 다를 생성하여 앱을 보호합니다. 스냅샷과 백업 생성 주기를 시간별, 일별, 주별, 월별로 설정할 수 있으며, 보존할 백업 복사본의 개수를 지정할 수도 있습니다. full-backup-rule 어노테이션을 사용하면 증분 백업이 아닌 전체 백업을 예약할 수 있습니다. 기본적으로 모든 백업은 증분 백업입니다. 전체 백업을 주기적으로 수행하고 그 사이에 증분 백업을 수행하면 복원과 관련된 위험을 줄이는 데 도움이 됩니다.

- `backupRetention`을 0으로 설정하고 `snapshotRetention`을 0보다 큰 값으로 설정하여 스냅샷 전용 스케줄을 생성할 수 있습니다. `snapshotRetention`을 0으로 설정하면 예약된 백업에서 여전히 스냅샷이 생성되지만 이는 임시적이며 백업이 완료된 후 즉시 삭제됩니다.
- 클러스터 범위 리소스는 애플리케이션 정의에서 명시적으로 참조되거나 애플리케이션 네임스페이스를 참조하는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

CR을 사용하여 일정 생성

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-schedule-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.dataMover**: (선택 사항) 백업 작업에 사용할 백업 도구를 나타내는 문자열입니다. 가능한 값(대소문자 구분):
     - `Restic`
     - `Kopia` (기본값)
   - **spec.applicationRef**: 백업할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 백업 콘텐츠를 저장할 AppVault의 이름입니다.
   - **spec.backupRetention**: (필수) 보존할 백업 개수입니다. 0으로 설정하면 백업을 생성하지 않고 스냅샷만 생성합니다.
   - **spec.backupReclaimPolicy**: (선택 사항) 보존 기간 동안 백업 CR이 삭제될 경우 백업 데이터에 발생하는 작업을 결정합니다. 보존 기간 이후에는 백업이 항상 삭제됩니다. 가능한 값(대소문자 구분):
     - `Retain` (기본값)
     - `Delete`
   - **spec.snapshotRetention**: (필수) 보존할 스냅샷 개수입니다. 0으로 설정하면 스냅샷을 생성하지 않습니다.
   - **spec.snapshotReclaimPolicy**: (선택 사항) 보존 기간 동안 스냅샷 CR이 삭제될 경우 스냅샷에 발생하는 상황을 결정합니다. 보존 기간이 지나면 스냅샷은 항상 삭제됩니다. 가능한 값(대소문자 구분):
     - `Retain`
     - `Delete` (기본값)
   - **spec.granularity**: 스케줄이 실행될 빈도입니다. 가능한 값과 필수 관련 필드는 다음과 같습니다.
     - `Hourly` (지정해야 합니다 `spec.minute`)
     - `Daily` (`spec.minute` 및 `spec.hour`을 지정해야 함)
     - `Weekly`(, `spec.minute, spec.hour` 및 `spec.dayOfWeek`를 지정해야 합니다)
     - `Monthly`(, `spec.minute, spec.hour` 및 `spec.dayOfMonth`를 지정해야 합니다)
     - `Custom`
   - **spec.dayOfMonth**: (선택 사항) 스케줄이 실행될 날짜(1~31일)입니다. 세분성이 `Monthly`로 설정된 경우 이 필드는 필수입니다. 값은 문자열로 제공해야 합니다.
   - **spec.dayOfWeek**: (선택 사항) 스케줄을 실행할 요일(0 - 7)입니다. 0 또는 7 값은 일요일을 나타냅니다. 세분성이 `Weekly`로 설정된 경우 이 필드는 필수입니다. 값은 문자열로 제공해야 합니다.
   - **spec.hour**: (선택 사항) 스케줄이 실행될 시간(0 - 23)입니다. 이 필드는 세분성이 `Daily`, `Weekly` 또는 `Monthly`로 설정된 경우 필수입니다. 값은 문자열로 제공해야 합니다.
   - **spec.minute**: (선택 사항) 스케줄이 실행되어야 하는 시간의 분(0 - 59)입니다. 세분성이 `Hourly`,

`Daily`, `Weekly` 또는 `Monthly`로 설정된 경우 이 필드는 필수입니다. 값은 문자열로 제공해야 합니다.

- **spec.runImmediately**: (선택 사항) `true`로 설정하면 스케줄 생성 시 일회성으로 즉시 기준선 실행(보존 설정에 따른 백업 및/또는 스냅샷)을 트리거합니다. 기본값은 `false`입니다. 이 설정은 이후 반복 실행에는 영향을 미치지 않습니다.

백업 및 스냅샷 스케줄에 대한 YAML 예:

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

스냅샷 전용 스케줄의 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"
```

즉시 실행이 포함된 스케줄에 대한 YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-daily-schedule-run-immediately
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "7"
  snapshotRetention: "7"
  granularity: Daily
  hour: "3"
  minute: "0"
  runImmediately: true
```

3. `trident-protect-schedule-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI를 사용하여 스케줄 생성

단계

1. 보호 일정을 생성하고 괄호 안의 값을 사용자 환경 정보로 대체하십시오. 예를 들면 다음과 같습니다.

> ⓘ `tridentctl-protect create schedule --help`을 사용하여 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> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_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> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

다음 플래그를 사용하면 일정을 더욱 세밀하게 제어할 수 있습니다.

- 전체 백업 예약: `--full-backup-rule` 플래그를 사용하여 증분 방식이 아닌 전체 백업을 예약합니다. 이 플래그는 `--granularity Daily`에서만 작동합니다. 가능한 값:
  - `Always`: 매일 전체 백업을 생성하세요.
  - 특정 요일: 쉼표로 구분하여 하나 이상의 요일을 지정합니다(예: `"Monday,Thursday"`). 유효한 값: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

  > `--full-backup-rule` 플래그는 시간별, 주별 또는 월별 세분화에서는 작동하지 않습니다.

- 즉시 기준선 보호: `--run-immediately true`을 사용하여 첫 번째 예약 실행 시간을 기다리지 않고 예약이 생성될 때 즉시 초기 백업 또는 스냅샷을 생성합니다. 기본값은 `false`입니다.
- 스냅샷 전용 일정: `--backup-retention 0`을(를) 설정하고 `--snapshot-retention`에 대해 0보다 큰 값을 지정하십시오.

지원되는 스케줄 주석

다음 표에서는 스케줄 CR을 생성할 때 사용할 수 있는 주석에 대해 설명합니다.

주석	유형	설명	기본값
protect.trident.netapp.io/full-backup-rule	문자열	전체 백업 예약 규칙을 지정합니다. `Always`로 설정하여 정기적인 전체 백업을 수행하거나 필요에 따라 사용자 지정할 수 있습니다. 예를 들어, 일 단위로 예약하는 경우 전체 백업이 수행될 요일을 지정할 수 있습니다(예: `"Monday,Thursday"`). 유효한 요일 값은 Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday입니다. 이 어노테이션은 `granularity`가 `Daily`로 설정된 예약에만 사용할 수 있습니다.	설정되지 않음(모든 백업은 증분)
protect.trident.netapp.io/snapshot-completion-timeout	문자열	전체 스냅샷 작업이 완료되는 데 허용되는 최대 시간입니다.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	문자열	볼륨 스냅샷이 사용 가능한 상태에 도달하는 데 허용되는 최대 시간입니다.	"30분"
protect.trident.netapp.io/volume-snapshots-created-timeout	문자열	볼륨 스냅샷을 생성하는 데 허용되는 최대 시간입니다.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	문자열	새로 생성된 PersistentVolumeClaims(PVC)가 `Bound` 단계에 도달하기까지 기다리는 최대 시간(초)입니다. 이 시간이 지나면 작업이 실패합니다.	"1200" (20분)

스냅샷 삭제

더 이상 필요하지 않은 예약된 스냅샷 또는 필요 시 스냅샷을 삭제합니다.

단계

1. 스냅샷과 연결된 스냅샷 CR을 제거합니다.

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

백업 삭제

더 이상 필요하지 않은 예약된 백업 또는 온디맨드 백업을 삭제합니다.

객체 스토리지에서 모든 백업 데이터를 제거하려면 reclaim 정책이 `Delete`로 설정되어 있는지 확인하십시오. 정책의 기본 설정은 의도하지 않은 데이터 손실을 방지하기 위해 `Retain`입니다. 정책을 `Delete`로 변경하지 않으면 백업 데이터가 객체 스토리지에 남아 있으므로 수동으로 삭제해야 합니다.

단계

1. 백업과 연결된 백업 CR을 제거합니다.

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

백업 작업 상태 확인

명령줄을 사용하여 진행 중인 백업 작업, 완료된 백업 작업 또는 실패한 백업 작업의 상태를 확인할 수 있습니다.

단계

1. 다음 명령을 사용하여 백업 작업의 상태를 검색하고 대괄호 안의 값을 사용자 환경의 정보로 바꾸십시오.

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

azure-netapp-files(ANF) 작업에 대한 백업 및 복원 활성화

Trident Protect를 설치한 경우, azure-netapp-files 스토리지 클래스를 사용하고 Trident 24.06 이전에 생성된 스토리지 백엔드에 대해 공간 효율적인 백업 및 복원 기능을 활성화할 수 있습니다. 이 기능은 NFSv4 볼륨에서 작동하며 용량 풀에서 추가 공간을 차지하지 않습니다.

시작하기 전에

다음 사항을 확인하십시오.

- Trident Protect를 설치했습니다.
- Trident Protect에서 애플리케이션을 정의했습니다. 이 절차를 완료할 때까지 이 애플리케이션의 보호 기능이 제한됩니다.
- `azure-netapp-files`을(를) 스토리지 백엔드의 기본 스토리지 클래스로 선택했습니다.

구성 단계를 보려면 펼치세요

1. ANF 볼륨이 Trident 24.10으로 업그레이드하기 전에 생성된 경우 Trident에서 다음을 수행하십시오.

   a. 애플리케이션과 연결된 azure-netapp-files 기반의 각 PV에 대해 스냅샷 디렉터리를 활성화합니다.

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

   b. 각 연결된 PV에 대해 스냅샷 디렉토리가 활성화되었는지 확인합니다.

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

   응답:

   ```
   snapshotDirectory: "true"
   ```

   +
   스냅샷 디렉터리가 활성화되지 않은 경우, Trident Protect는 일반 백업 기능을 선택하며, 이 기능은 백업 프로세스 중에 용량 풀의 공간을 일시적으로 사용합니다. 이 경우 백업 대상 볼륨 크기의 임시 볼륨을 생성할 수 있도록 용량 풀에 충분한 공간이 확보되어 있는지 확인하십시오.

결과

이 애플리케이션은 Trident Protect를 사용하여 백업 및 복원할 준비가 되어 있습니다. 각 PVC는 다른 애플리케이션에서도 백업 및 복원에 사용할 수 있습니다.

애플리케이션 복원

Trident Protect를 사용하여 애플리케이션을 복원하세요

Trident Protect를 사용하면 스냅샷 또는 백업에서 애플리케이션을 복원할 수 있습니다. 기존 스냅샷에서 복원하면 동일한 클러스터에 애플리케이션을 복원할 때 더 빠릅니다.

- 애플리케이션을 복원하면 해당 애플리케이션에 대해 구성된 모든 실행 후크가 애플리케이션과 함께 복원됩니다. 복원 후 실행 후크가 있는 경우 복원 작업의 일부로 자동으로 실행됩니다.
- qtree 볼륨의 경우 백업에서 다른 네임스페이스 또는 원래 네임스페이스로 복원하는 것이 지원됩니다. 그러나 스냅샷에서 다른 네임스페이스 또는 원래 네임스페이스로 복원하는 것은 qtree 볼륨에서 지원되지 않습니다.
- 고급 설정을 사용하여 복원 작업을 사용자 지정할 수 있습니다. 자세한 내용은 "고급 Trident Protect 복원 설정을 사용하십시오"를 참조하십시오.

백업에서 다른 네임스페이스로 복원

BackupRestore CR을 사용하여 다른 네임스페이스로 백업을 복원하면 Trident Protect는 새 네임스페이스에 애플리케이션을 복원하고 복원된 애플리케이션에 대한 애플리케이션 CR을 생성합니다. 복원된 애플리케이션을 보호하려면 온디맨드 백업 또는 스냅샷을 생성하거나 보호 일정을 설정하십시오.

- 기존 리소스가 있는 다른 네임스페이스로 백업을 복원해도 백업에 포함된 리소스와 이름이 같은 리소스는 변경되지 않습니다. 백업의 모든 리소스를 복원하려면 대상 네임스페이스를 삭제하고 다시 생성하거나 새 네임스페이스로 백업을 복원해야 합니다.
- CR을 사용하여 새 네임스페이스로 복원할 경우, CR을 적용하기 전에 타겟 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

시작하기 전에

장시간 소요되는 s3 복원 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
- AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS IAM 문서"를 참조하십시오.

Kopia를 데이터 이동 도구로 사용하여 백업을 복원할 때 CR 또는 CLI를 사용하여 Kopia에서 사용하는 임시 스토리지의 동작을 제어하는 주석을 선택적으로 지정할 수 있습니다. 구성할 수 있는 옵션에 대한 자세한 내용은 "Kopia 문서"을 참조하십시오. Trident Protect CLI를 사용하여 주석을 지정하는 방법에 대한 자세한 내용은 `tridentctl-protect create --help` 명령을 사용하십시오.

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-backup-restore-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appArchivePath**: 백업 콘텐츠가 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

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

   - **spec.appVaultRef**: (필수) 백업 콘텐츠가 저장되는 AppVault의 이름입니다.
   - **spec.destinationApplicationName**: (선택 사항) 복원된 애플리케이션의 이름입니다. 제공된 경우 복원된 애플리케이션은 이 이름을 사용합니다. 제공되지 않은 경우 복원된 애플리케이션은 원본 애플리케이션 이름을 사용합니다.
   - **spec.namespaceMapping**: 복원 작업의 소스 네임스페이스를 타겟 네임스페이스로 매핑합니다. `my-source-namespace` 및 `my-destination-namespace`를 사용자 환경의 정보로 교체하십시오.

     ```
     ---
     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. (선택 사항) 애플리케이션의 특정 리소스만 복원해야 하는 경우, 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가하십시오.

   > Trident Protect는 사용자가 선택한 리소스와의 연관성 때문에 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 해당 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `Include` 또는 `Exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.

- **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
  - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 그룹입니다.
  - **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.
  - **resourceMatchers[].version**: (선택 사항) 필터링할 리소스의 버전입니다.
  - **resourceMatchers[].names**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
  - **resourceMatchers[].namespaces**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
  - **resourceMatchers[].labelSelectors**: (선택 사항) "Kubernetes 문서"에 정의된 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. 예를 들면 다음과 같습니다: `"trident.netapp.io/os=linux"`.

    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. `trident-protect-backup-restore-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI 사용

단계

1. 백업을 다른 네임스페이스로 복원하고, 괄호 안의 값을 사용자 환경 정보로 바꿉니다. `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 `source1:dest1,source2:dest2` 형식으로 올바른 타겟 네임스페이스에 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> \
--destination-app-name<custom_app_name>\
-n <application_namespace>
```

원래 네임스페이스로 백업에서 복원합니다

언제든지 백업을 원래 네임스페이스로 복원할 수 있습니다. 제자리 복원을 수행하면 Trident Protect는 잘못된 복구 지점이 생성되는 것을 방지하기 위해 보호 일정과 진행 중인 작업을 자동으로 관리합니다.

- 복원 작업이 시작되기 전에 애플리케이션에 대해 활성화된 모든 보호 일정이 비활성화됩니다. 이렇게 하면 애플리케이션 리소스가 복원되는 동안 예약된 백업이나 스냅샷이 실행되는 것을 방지할 수 있습니다.
- 복원 작업이 성공적으로 완료되면 복원 전에 활성화되어 있던 일정만 다시 활성화됩니다. 이미 비활성화된 일정은 계속 비활성화된 상태로 유지됩니다.
- 복원이 시작되기 전에 진행 중인 모든 백업 또는 스냅샷 작업이 취소됩니다. 5분 이내에 작업이 취소되지 않으면 복원이 진행되고 복원 CR 상태에 경고가 기록됩니다.

시작하기 전에

장시간 소요되는 s3 복원 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
- AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS IAM 문서"를 참조하십시오.

Kopia를 데이터 이동 도구로 사용하여 백업을 복원할 때 CR 또는 CLI를 사용하여 Kopia에서 사용하는 임시 스토리지의 동작을 제어하는 주석을 선택적으로 지정할 수 있습니다. 구성할 수 있는 옵션에 대한 자세한 내용은 "Kopia 문서"을 참조하십시오. Trident Protect CLI를 사용하여 주석을 지정하는 방법에 대한 자세한 내용은 `tridentctl-protect create --help` 명령을 사용하십시오.

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-backup-ipr-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appArchivePath**: 백업 콘텐츠가 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

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

   - **spec.appVaultRef**: (필수) 백업 콘텐츠가 저장되는 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) 애플리케이션의 특정 리소스만 복원해야 하는 경우, 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가하십시오.

   Trident Protect는 사용자가 선택한 리소스와의 연관성 때문에 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 해당 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `Include` 또는 `Exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
       - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 그룹입니다.
       - **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.

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

  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. `trident-protect-backup-ipr-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI 사용

단계

1. 원래 네임스페이스에 백업을 복원하고, 괄호 안의 값을 사용자 환경 정보로 바꿉니다. `backup` 인수는 `<namespace>/<name>` 형식으로 네임스페이스와 백업 이름을 사용합니다. 예를 들면 다음과 같습니다.

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

다른 클러스터로 백업에서 복원

원래 클러스터에 문제가 발생한 경우 백업을 다른 클러스터로 복원할 수 있습니다.

- Kopia를 데이터 이동 도구로 사용하여 백업을 복원할 때 CR 또는 CLI를 사용하여 Kopia에서 사용하는 임시 스토리지의 동작을 제어하는 주석을 선택적으로 지정할 수 있습니다. 구성할 수 있는 옵션에 대한 자세한 내용은 "Kopia 문서"을 참조하십시오. Trident Protect CLI를 사용하여 주석을 지정하는 방법에 대한 자세한 내용은 `tridentctl-protect create --help` 명령을 사용하십시오.
- CR을 사용하여 새 네임스페이스로 복원할 경우, CR을 적용하기 전에 타겟 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

시작하기 전에

다음 사전 요구 사항이 충족되는지 확인하십시오.

- 타겟 클러스터에 Trident Protect가 설치되어 있습니다.
- 타겟 클러스터는 백업이 저장된 소스 클러스터와 동일한 AppVault의 버킷 경로에 액세스할 수 있습니다.
- `tridentctl-protect get appvaultcontent` 명령을 실행할 때 로컬 환경에서 AppVault CR에 정의된 오브젝트 스토리지 버킷에 연결할 수 있는지 확인하십시오. 네트워크 제한으로 인해 액세스할 수 없는 경우 타겟 클러스터의 Pod 내에서 Trident Protect CLI를 실행하십시오.
- 장시간 소요되는 복원 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.
  - 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
  - AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS 문서"를 참조하십시오.

단계

1. Trident Protect CLI 플러그인을 사용하여 타겟 클러스터에 AppVault CR이 있는지 확인합니다.

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

타겟 클러스터에 AppVault CR이 없는 경우 "Trident Protect AppVault 객체를 사용하여 버킷을 관리하세요."의 단계에 따라 생성하십시오.

2. 타겟 클러스터에서 사용 가능한 AppVault의 백업 콘텐츠를 확인하고 복원할 백업의 `appArchivePath`을 기록해 두십시오.

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

이 명령을 실행하면 AppVault에서 사용 가능한 백업이 표시되며, 여기에는 백업이 생성된 클러스터, 해당 애플리케이션 이름, 타임스탬프 및 아카이브 경로가 포함됩니다.

예시 출력:

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   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 이름과 아카이브 경로를 사용하여 타겟 클러스터에 애플리케이션을 복원합니다.

CR을 사용할 때는 애플리케이션 복원에 사용할 네임스페이스가 타겟 클러스터에 있는지 확인하십시오.

CR 사용

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-backup-restore-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appVaultRef**: (필수) 백업 콘텐츠가 저장되는 AppVault의 이름입니다.
   - **spec.appArchivePath**: (필수) AppVault 내에서 백업 콘텐츠가 저장되는 경로입니다. 2단계의 명령을 사용하여 백업 콘텐츠를 확인하고 `appArchivePath`복원하려는 백업을 찾으십시오.
   - **spec.destinationApplicationName**: (선택 사항) 복원된 애플리케이션의 이름입니다. 제공된 경우 복원된 애플리케이션은 이 이름을 사용합니다. 제공되지 않은 경우 복원된 애플리케이션은 원본 애플리케이션 이름을 사용합니다.
   - **spec.namespaceMapping**: 복원 작업의 소스 네임스페이스를 타겟 네임스페이스로 매핑합니다. `my-source-namespace` 및 `my-destination-namespace`를 사용자 환경의 정보로 교체하십시오.

     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
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. `trident-protect-backup-restore-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI 사용

1. 다음 명령을 사용하여 애플리케이션을 복원하고 대괄호 안의 값을 사용자 환경의 정보로 바꿉니다. namespace-mapping 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 source1:dest1,source2:dest2 형식으로 올바른 타겟 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> \
--destination-app-name <custom_app_name> \
--context <destination_cluster_name> \
-n <application_namespace>
```

스냅샷에서 다른 네임스페이스로 복원

스냅샷에서 사용자 지정 리소스(CR) 파일을 사용하여 다른 네임스페이스 또는 원래 소스 네임스페이스로 데이터를 복원할 수 있습니다. SnapshotRestore CR을 사용하여 스냅샷을 다른 네임스페이스로 복원하면 Trident Protect는 새 네임스페이스에 애플리케이션을 복원하고 복원된 애플리케이션에 대한 애플리케이션 CR을 생성합니다. 복원된 애플리케이션을 보호하려면 온디맨드 백업 또는 스냅샷을 생성하거나 보호 일정을 설정하십시오.

- SnapshotRestore는 `spec.storageClassMapping` 속성을 지원하지만, 소스 및 타겟 스토리지 클래스가 동일한 스토리지 백엔드를 사용하는 경우에만 가능합니다. 다른 스토리지 백엔드를 사용하는 `StorageClass`로 복원을 시도하면 복원 작업이 실패합니다.
- CR을 사용하여 새 네임스페이스로 복원할 경우, CR을 적용하기 전에 타겟 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

시작하기 전에

장시간 소요되는 s3 복원 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
- AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS IAM 문서"를 참조하십시오.

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-snapshot-restore-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appVaultRef**: (필수) 스냅샷 콘텐츠가 저장된 AppVault의 이름입니다.
   - **spec.appArchivePath**: 스냅샷 콘텐츠가 저장되는 AppVault 내부의 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

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

   - **spec.destinationApplicationName**: (선택 사항) 복원된 애플리케이션의 이름입니다. 제공된 경우 복원된 애플리케이션은 이 이름을 사용합니다. 제공되지 않은 경우 복원된 애플리케이션은 원본 애플리케이션 이름을 사용합니다.
   - **spec.namespaceMapping**: 복원 작업의 소스 네임스페이스를 타겟 네임스페이스로 매핑합니다. `my-source-namespace` 및 `my-destination-namespace`를 사용자 환경의 정보로 교체하십시오.

     ```
     ---
     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) 애플리케이션의 특정 리소스만 복원해야 하는 경우, 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가하십시오.

   Trident Protect는 사용자가 선택한 리소스와의 연관성 때문에 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 해당 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `Include` 또는 `Exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.

- **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
  - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 그룹입니다.
  - **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.
  - **resourceMatchers[].version**: (선택 사항) 필터링할 리소스의 버전입니다.
  - **resourceMatchers[].names**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
  - **resourceMatchers[].namespaces**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
  - **resourceMatchers[].labelSelectors**: (선택 사항) "Kubernetes 문서"에 정의된 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. 예를 들면 다음과 같습니다: `"trident.netapp.io/os=linux"`.

    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. `trident-protect-snapshot-restore-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI 사용

단계

1. 스냅샷을 다른 네임스페이스로 복원하고 괄호 안의 값을 사용자 환경의 정보로 바꾸십시오.

   ◦ `snapshot` 인수는 `<namespace>/<name>` 형식의 네임스페이스와 스냅샷 이름을 사용합니다.

◦ `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 `source1:dest1,source2:dest2` 형식으로 올바른 타겟 네임스페이스에 매핑합니다.

예를 들면 다음과 같습니다.

```
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>
```

스냅샷에서 원래 네임스페이스로 복원합니다

언제든지 스냅샷을 원래 네임스페이스로 복원할 수 있습니다. 제자리 복원을 수행하면 Trident Protect는 잘못된 복구 지점이 생성되는 것을 방지하기 위해 보호 일정과 진행 중인 작업을 자동으로 관리합니다.

- 복원 작업이 시작되기 전에 애플리케이션에 대해 활성화된 모든 보호 일정이 비활성화됩니다. 이렇게 하면 애플리케이션 리소스가 복원되는 동안 예약된 백업이나 스냅샷이 실행되는 것을 방지할 수 있습니다.
- 복원 작업이 성공적으로 완료되면 복원 전에 활성화되어 있던 일정만 다시 활성화됩니다. 이미 비활성화된 일정은 계속 비활성화된 상태로 유지됩니다.
- 복원이 시작되기 전에 진행 중인 모든 백업 또는 스냅샷 작업이 취소됩니다. 5분 이내에 작업이 취소되지 않으면 복원이 진행되고 복원 CR 상태에 경고가 기록됩니다.

시작하기 전에

장시간 소요되는 s3 복원 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
- AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS IAM 문서"를 참조하십시오.

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-snapshot-ipr-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appVaultRef**: (필수) 스냅샷 콘텐츠가 저장된 AppVault의 이름입니다.
   - **spec.appArchivePath**: 스냅샷 콘텐츠가 저장되는 AppVault 내부의 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) 애플리케이션의 특정 리소스만 복원해야 하는 경우, 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가하십시오.

> Trident Protect는 사용자가 선택한 리소스와의 연관성 때문에 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 해당 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `Include` 또는 `Exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
       - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 그룹입니다.
       - **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.
       - **resourceMatchers[].version**: (선택 사항) 필터링할 리소스의 버전입니다.
       - **resourceMatchers[].names**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name

필드에 있는 이름입니다.

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

  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. `trident-protect-snapshot-ipr-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI 사용

단계

1. 스냅샷을 원래 네임스페이스로 복원하고 괄호 안의 값을 사용자 환경 정보로 바꾸십시오. 예를 들면 다음과 같습니다.

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

복원 작업의 상태를 확인합니다

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

단계

1. 다음 명령을 사용하여 복원 작업의 상태를 검색하고 대괄호 안의 값을 사용자 환경의 정보로 바꾸십시오.

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

고급 **Trident Protect** 복원 설정을 사용하십시오

주석, 네임스페이스 설정, 스토리지 옵션과 같은 고급 설정을 사용하여 특정 요구 사항에 맞게 복원 작업을 사용자 지정할 수 있습니다.

복원 및 페일오버 작업 중 네임스페이스 주석 및 레이블

복원 및 페일오버 작업 중에 대상 네임스페이스의 레이블과 주석은 소스 네임스페이스의 레이블과 주석과 일치하도록 조정됩니다. 대상 네임스페이스에 존재하지 않는 소스 네임스페이스의 레이블 또는 주석은 추가되며, 이미 존재하는 레이블 또는 주석은 소스 네임스페이스의 값과 일치하도록 덮어쓰여집니다. 대상 네임스페이스에만 존재하는 레이블 또는 주석은 변경되지 않습니다.

Red Hat OpenShift를 사용하는 경우 OpenShift 환경에서 네임스페이스 어노테이션의 중요한 역할을 알아두는 것이 중요합니다. 네임스페이스 어노테이션은 복원된 Pod가 OpenShift 보안 컨텍스트 제약 조건(SCC)에 정의된 적절한 권한 및 보안 구성을 준수하고 권한 문제 없이 볼륨에 액세스할 수 있도록 보장합니다. 자세한 내용은 "OpenShift 보안 컨텍스트 제약 조건 문서"를 참조하십시오.

복원 또는 장애 조치 작업을 수행하기 전에 Kubernetes 환경 변수 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`를 설정하여 대상 네임스페이스의 특정 어노테이션이 덮어쓰여지는 것을 방지할 수 있습니다. 예를 들면 다음과 같습니다.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

복원 또는 페일오버 작업을 수행할 때 `restoreSkipNamespaceAnnotations` 및 `restoreSkipNamespaceLabels`에 지정된 네임스페이스 주석 및 레이블은 복원 또는 페일오버 작업에서 제외됩니다. 초기 Helm 설치 시 이러한 설정이 구성되어 있는지 확인하십시오. 자세한 내용은 "Trident Protect 헬름 차트의 추가 설정을 구성합니다"를 참조하십시오.

`--create-namespace` 플래그와 함께 `Helm`을 사용하여 소스 애플리케이션을 설치한 경우 `name` 레이블 키에 특별한 처리가 적용됩니다. 복원 또는 페일오버 프로세스 중에 `Trident Protect`는 이 레이블을 대상 네임스페이스로 복사하지만 소스의 값이 소스 네임스페이스와 일치하는 경우 값을 대상 네임스페이스 값으로 업데이트합니다. 이 값이 소스 네임스페이스와 일치하지 않으면 변경 없이 대상 네임스페이스로 복사됩니다.

예

다음 예제는 서로 다른 어노테이션과 레이블을 가진 소스 및 대상 네임스페이스를 보여줍니다. 작업 전후의 대상 네임스페이스 상태와 어노테이션 및 레이블이 대상 네임스페이스에서 어떻게 결합되거나 덮어쓰여지는지 확인할 수 있습니다.

복원 또는 페일오버 작업 전에

다음 표는 복원 또는 페일오버 작업 전 예시 소스 및 대상 네임스페이스의 상태를 보여줍니다.

네임스페이스	주석	라벨
네임스페이스 ns-1(소스)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • 컴플라이언스=hipaa • name=ns-1
네임스페이스 ns-2(대상)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

복원 작업 후

다음 표는 복원 또는 장애 조치 작업 후 예시 대상 네임스페이스의 상태를 보여줍니다. 일부 키가 추가되었고, 일부는 덮어쓰여졌으며, `name` 레이블은 대상 네임스페이스와 일치하도록 업데이트되었습니다.

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • 컴플라이언스=hipaa • environment=production • role=database

지원되는 필드

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

스토리지 클래스 매핑

`spec.storageClassMapping` 속성은 소스 애플리케이션에 있는 스토리지 클래스를 대상 클러스터의 새 스토리지 클래스로 매핑하는 방법을 정의합니다. 이 속성은 스토리지 클래스가 다른 클러스터 간에 애플리케이션을 마이그레이션하거나 `BackupRestore` 작업의 스토리지 백엔드를 변경할 때 사용할 수 있습니다.

예:

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

지원되는 주석

이 섹션에서는 시스템의 다양한 동작을 구성하는 데 지원되는 어노테이션 목록을 제공합니다. 사용자가 어노테이션을 명시적으로 설정하지 않으면 시스템은 기본값을 사용합니다.

주석	유형	설명	기본값
protect.trident.netapp.io/data-mover-timeout-sec	문자열	데이터 이동기 작동이 정지될 수 있는 최대 허용 시간(초)입니다.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	문자열	Kopia 콘텐츠 캐시의 최대 크기 제한(메가바이트)입니다.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	문자열	새로 생성된 PersistentVolumeClaims(PVC)가 `Bound` 단계에 도달하기까지 기다리는 최대 시간(초)입니다. 이 시간이 지나면 작업이 실패합니다. 모든 복원 CR 유형(BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore)에 적용됩니다. 스토리지 백엔드 또는 클러스터에서 더 많은 시간이 필요한 경우 더 높은 값을 사용하십시오.	"1200" (20분)

NetApp SnapMirror 및 Trident Protect를 사용하여 애플리케이션을 복제합니다

Trident Protect를 사용하면 NetApp SnapMirror 기술의 비동기 복제 기능을 활용하여 동일한 클러스터 내에서 또는 서로 다른 클러스터 간에 데이터 및 애플리케이션 변경 사항을 한 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 주석은 소스 네임스페이스의 레이블과 주석과 일치하도록 조정됩니다. 대상 네임스페이스에 존재하지 않는 소스 네임스페이스의 레이블 또는 주석은 추가되며, 이미 존재하는 레이블 또는 주석은 소스 네임스페이스의 값과 일치하도록 덮어쓰여집니다. 대상 네임스페이스에만 존재하는 레이블 또는 주석은 변경되지 않습니다.

Red Hat OpenShift를 사용하는 경우 OpenShift 환경에서 네임스페이스 어노테이션의 중요한 역할을 알아두는 것이 중요합니다. 네임스페이스 어노테이션은 복원된 Pod가 OpenShift 보안 컨텍스트 제약 조건(SCC)에 정의된 적절한 권한 및 보안 구성을 준수하고 권한 문제 없이 볼륨에 액세스할 수 있도록 보장합니다. 자세한 내용은 "OpenShift 보안 컨텍스트 제약 조건 문서"를 참조하십시오.

복원 또는 장애 조치 작업을 수행하기 전에 Kubernetes 환경 변수 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`를 설정하여 대상 네임스페이스의 특정 어노테이션이 덮어쓰여지는 것을 방지할 수 있습니다. 예를 들면 다음과 같습니다.

```
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
```

복원 또는 페일오버 작업을 수행할 때 `restoreSkipNamespaceAnnotations` 및 `restoreSkipNamespaceLabels`에 지정된 네임스페이스 주석 및 레이블은 복원 또는 페일오버 작업에서 제외됩니다. 초기 Helm 설치 시 이러한 설정이 구성되어 있는지 확인하십시오. 자세한 내용은 "Trident Protect 헬름 차트의 추가 설정을 구성합니다"를 참조하십시오.

`--create-namespace` 플래그와 함께 `Helm`을 사용하여 소스 애플리케이션을 설치한 경우 `name` 레이블 키에 특별한 처리가 적용됩니다. 복원 또는 페일오버 프로세스 중에 `Trident Protect`는 이 레이블을 대상 네임스페이스로 복사하지만 소스의 값이 소스 네임스페이스와 일치하는 경우 값을 대상 네임스페이스 값으로 업데이트합니다. 이 값이 소스 네임스페이스와 일치하지 않으면 변경 없이 대상 네임스페이스로 복사됩니다.

예

다음 예제는 서로 다른 어노테이션과 레이블을 가진 소스 및 대상 네임스페이스를 보여줍니다. 작업 전후의 대상 네임스페이스 상태와 어노테이션 및 레이블이 대상 네임스페이스에서 어떻게 결합되거나 덮어쓰여지는지 확인할 수 있습니다.

복원 또는 페일오버 작업 전에

다음 표는 복원 또는 페일오버 작업 전 예시 소스 및 대상 네임스페이스의 상태를 보여줍니다.

네임스페이스	주석	라벨
네임스페이스 ns-1(소스)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • 컴플라이언스=hipaa • name=ns-1
네임스페이스 ns-2(대상)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

복원 작업 후

다음 표는 복원 또는 장애 조치 작업 후 예시 대상 네임스페이스의 상태를 보여줍니다. 일부 키가 추가되었고, 일부는 덮어쓰여졌으며, `name` 레이블은 대상 네임스페이스와 일치하도록 업데이트되었습니다.

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • 컴플라이언스=hipaa • environment=production • role=database

Trident Protect를 구성하면 데이터 보호 작업 중에 파일 시스템을 동결 및 해제할 수 있습니다. "Trident Protect를 사용한 파일 시스템 동결 구성에 대해 자세히 알아보십시오".

페일오버 및 역방향 작업 중 실행 후크

AppMirror 관계를 사용하여 애플리케이션을 보호할 때 페일오버 및 역방향 작업 중에 알아야 할 실행 후크와 관련된 특정 동작이 있습니다.

- 장애 조치 시 실행 후크는 소스 클러스터에서 타겟 클러스터로 자동으로 복사됩니다. 수동으로 다시 생성할 필요가 없습니다. 장애 조치가 완료되면 애플리케이션에 실행 후크가 존재하며 관련 작업이 수행될 때 실행됩니다.
- 역방향 또는 역방향 재동기화 중에 애플리케이션의 기존 실행 후크가 제거됩니다. 소스 애플리케이션이 타겟 애플리케이션이 되면 이러한 실행 후크는 유효하지 않으며 실행을 방지하기 위해 삭제됩니다.

실행 후크에 대한 자세한 내용은 "Trident Protect 실행 후크 관리"을(를) 참조하십시오.

복제 관계를 설정합니다

복제 관계를 설정하는 과정은 다음과 같습니다.

- Trident Protect에서 앱 스냅샷(앱의 Kubernetes 리소스와 앱의 각 볼륨에 대한 볼륨 스냅샷 포함)을 생성할 빈도를 선택합니다
- 복제 일정 선택(Kubernetes 리소스 및 영구 볼륨 데이터 포함)
- 스냅샷을 생성할 시간 설정

단계

1. 소스 클러스터에서 소스 애플리케이션용 AppVault를 생성합니다. 스토리지 공급자에 따라 "AppVault 사용자 지정 리소스"의 예제를 환경에 맞게 수정하십시오.

CR을 사용하여 **AppVault**를 생성합니다

a. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-appvault-primary-source.yaml`).

b. 다음 속성을 구성하십시오.

- **metadata.name**: (필수) AppVault 사용자 지정 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일에서 이 값을 참조하므로 선택한 이름을 기록해 두십시오.
- **spec.providerConfig**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다. bucketName 및 공급자에 필요한 기타 세부 정보를 선택하십시오. 복제 관계에 필요한 다른 CR 파일에서 이러한 값을 참조하므로 선택한 값을 기록해 두십시오. 다른 공급자를 사용하는 AppVault CR의 예는 "AppVault 사용자 지정 리소스"를 참조하십시오.
- **spec.providerCredentials**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
  - **spec.providerCredentials.valueFromSecret**: (필수) 자격 증명 값이 비밀 키에서 가져와야 함을 나타냅니다.
    - **key**: (필수) 선택할 시크릿의 유효한 키입니다.
    - **name**: (필수) 이 필드의 값을 포함하는 시크릿의 이름입니다. 동일한 네임스페이스에 있어야 합니다.
  - **spec.providerCredentials.secretAccessKey**: (필수) 공급자에 액세스하는 데 사용되는 액세스 키입니다. *name*은 *spec.providerCredentials.valueFromSecret.name*과 일치해야 합니다.
- **spec.providerType**: (필수) 백업을 제공하는 서비스(예: NetApp ONTAP S3, 일반 S3, Google Cloud 또는 Microsoft Azure)를 지정합니다. 가능한 값:
  - aws
  - Azure
  - gcp
  - generic-s3
  - ONTAP S3
  - storagegrid-s3

c. `trident-protect-appvault-primary-source.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI를 사용하여 **AppVault**를 생성합니다

a. AppVault를 생성하고 괄호 안의 값을 사용자 환경 정보로 바꾸세요.

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

2. 소스 클러스터에서 소스 애플리케이션 CR을 생성합니다.

CR을 사용하여 소스 애플리케이션을 생성합니다

a. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-app-source.yaml`).

b. 다음 속성을 구성하십시오.

- **metadata.name**: (필수) 애플리케이션 사용자 지정 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일에서 이 값을 참조하므로 선택한 이름을 기록해 두십시오.
- **spec.includedNamespaces**: (필수) 네임스페이스 및 관련 레이블 배열입니다. 네임스페이스 이름을 사용하고 선택적으로 레이블을 사용하여 네임스페이스의 범위를 좁혀 여기에 나열된 네임스페이스에 존재하는 리소스를 지정합니다. 애플리케이션 네임스페이스는 이 배열에 반드시 포함되어야 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: {}
```

c. `trident-protect-app-source.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI를 사용하여 소스 애플리케이션 생성

a. 소스 애플리케이션을 생성합니다. 예를 들면 다음과 같습니다.

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

3. 선택적으로 소스 클러스터에서 소스 애플리케이션의 스냅샷을 생성할 수 있습니다. 이 스냅샷은 타겟 클러스터의 애플리케이션에 대한 기반으로 사용됩니다. 이 단계를 건너뛰면 최신 스냅샷을 얻기 위해 다음 예약된 스냅샷이 실행될 때까지 기다려야 합니다. 온디맨드 스냅샷을 생성하려면 "온디맨드 스냅샷 생성"을 참조하십시오.
4. 소스 클러스터에서 복제 일정 CR을 생성합니다.

아래 제공된 일정 외에도, 피어링된 ONTAP 클러스터 간의 공통 스냅샷을 유지하기 위해 7일의 보존 기간을 가진 별도의 일일 스냅샷 일정을 생성하는 것이 좋습니다. 이렇게 하면 스냅샷을 최대 7일 동안 사용할 수 있지만, 보존 기간은 사용자 요구 사항에 따라 사용자 지정할 수 있습니다.

장애 조치가 발생할 경우, 시스템은 최대 7일 동안 이러한 스냅샷을 사용하여 역방향 작업을 수행할 수 있습니다. 이 방식을 통해 역방향 프로세스는 모든 데이터가 아닌 마지막 스냅샷 이후 변경된 내용만 전송되므로 더 빠르고 효율적으로 진행됩니다.

애플리케이션에 대한 기존 스케줄이 원하는 보존 요구 사항을 이미 충족하는 경우 추가 스케줄이 필요하지 않습니다.

CR을 사용하여 복제 일정을 생성합니다

a. 소스 애플리케이션에 대한 복제 일정을 생성합니다.

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

ii. 다음 속성을 구성하십시오.

- **metadata.name**: (필수) 스케줄 사용자 지정 리소스의 이름입니다.
- **spec.appVaultRef**: (필수) 이 값은 소스 애플리케이션의 AppVault에 있는 metadata.name 필드와 일치해야 합니다.
- **spec.applicationRef**: (필수) 이 값은 소스 애플리케이션 CR의 metadata.name 필드와 일치해야 합니다.
- **spec.backupRetention**: (필수) 이 필드는 필수이며 값을 0으로 설정해야 합니다.
- **spec.enabled**: true로 설정해야 합니다.
- **spec.granularity**: `Custom`로 설정해야 합니다.
- **spec.recurrenceRule**: UTC 시간으로 시작 날짜와 반복 간격을 정의합니다.
- **spec.snapshotRetention**: 2로 설정해야 합니다.

YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. `trident-protect-schedule.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI를 사용하여 복제 스케줄 생성

a. 복제 일정을 생성하고 괄호 안의 값을 사용자 환경의 정보로 바꾸십시오.

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

예:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. 타겟 클러스터에서 소스 클러스터에 적용한 AppVault CR과 동일한 소스 애플리케이션 AppVault CR을 생성하고 이름을 지정합니다(예: `trident-protect-appvault-primary-destination.yaml`).
6. CR 적용:

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

7. 타겟 클러스터의 타겟 애플리케이션에 대한 타겟 AppVault CR을 생성합니다. 스토리지 공급자에 따라 "AppVault 사용자 지정 리소스"의 예제를 환경에 맞게 수정하십시오.
   a. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-appvault-secondary-destination.yaml`).
   b. 다음 속성을 구성하십시오.
      - **metadata.name**: (필수) AppVault 사용자 지정 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일에서 이 값을 참조하므로 선택한 이름을 기록해 두십시오.
      - **spec.providerConfig**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다. `bucketName`및 공급자에 필요한 기타 세부 정보를 선택하십시오. 복제 관계에 필요한 다른 CR 파일에서 이러한 값을 참조하므로 선택한 값을 기록해 두십시오. 다른 공급자를 사용하는 AppVault CR의 예는 "AppVault 사용자 지정 리소스"를 참조하십시오.
      - **spec.providerCredentials**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
        - **spec.providerCredentials.valueFromSecret**: (필수) 자격 증명 값이 비밀 키에서 가져와야 함을 나타냅니다.
          - **key**: (필수) 선택할 시크릿의 유효한 키입니다.
          - **name**: (필수) 이 필드의 값을 포함하는 시크릿의 이름입니다. 동일한 네임스페이스에 있어야

합니다.

- **spec.providerCredentials.secretAccessKey**: (필수) 공급자에 액세스하는 데 사용되는 액세스 키입니다. *name*은 *spec.providerCredentials.valueFromSecret.name*과 일치해야 합니다.

- **spec.providerType**: (필수) 백업을 제공하는 서비스(예: NetApp ONTAP S3, 일반 S3, Google Cloud 또는 Microsoft Azure)를 지정합니다. 가능한 값:
  - aws
  - Azure
  - gcp
  - generic-s3
  - ONTAP S3
  - storagegrid-s3

c. `trident-protect-appvault-secondary-destination.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

8. 타겟 클러스터에서 AppMirrorRelationship CR 파일을 생성합니다.

> CR을 사용할 때는 CR을 적용하기 전에 타겟 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

CR을 사용하여 **AppMirrorRelationship**을 생성합니다

a. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-relationship.yaml`).

b. 다음 속성을 구성하십시오.

- **metadata.name:** (필수) AppMirrorRelationship 사용자 지정 리소스의 이름입니다.
- **spec.destinationAppVaultRef**: (필수) 이 값은 타겟 클러스터의 타겟 애플리케이션에 대한 AppVault 이름과 일치해야 합니다.
- **spec.namespaceMapping**: (필수) 타겟 및 소스 네임스페이스는 각 애플리케이션 CR에 정의된 애플리케이션 네임스페이스와 일치해야 합니다.
- **spec.sourceAppVaultRef**: (필수) 이 값은 소스 애플리케이션의 AppVault 이름과 일치해야 합니다.
- **spec.sourceApplicationName**: (필수) 이 값은 소스 애플리케이션 CR에 정의한 소스 애플리케이션의 이름과 일치해야 합니다.
- **spec.sourceApplicationUID**: (필수) 이 값은 소스 애플리케이션 CR에 정의한 소스 애플리케이션의 UID와 일치해야 합니다.
- **spec.storageClassName**: (선택 사항) 클러스터에서 유효한 스토리지 클래스의 이름을 선택하십시오. 스토리지 클래스는 소스 환경과 피어링된 ONTAP 스토리지 VM에 연결되어 있어야 합니다. 스토리지 클래스를 제공하지 않으면 클러스터의 기본 스토리지 클래스가 기본적으로 사용됩니다.
- **spec.recurrenceRule**: UTC 시간으로 시작 날짜와 반복 간격을 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-vsim-2
```

c. `trident-protect-relationship.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI를 사용하여 **AppMirrorRelationship** 생성

a. AppMirrorRelationship 객체를 생성하고 적용합니다. 괄호 안의 값은 사용자 환경의 정보로 바꿔야 합니다.

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> -n <application_namespace>
```

예:

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. (선택 사항) 타겟 클러스터에서 복제 관계의 상태를 확인합니다.

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

타겟 클러스터로 페일오버

Trident Protect를 사용하면 복제된 애플리케이션을 타겟 클러스터로 페일오버할 수 있습니다. 이 절차는 복제 관계를 중지하고 타겟 클러스터에서 애플리케이션을 온라인 상태로 전환합니다. Trident Protect는 소스 클러스터에서 애플리케이션이 작동 중인 경우 해당 애플리케이션을 중지하지 않습니다.

단계

1. 타겟 클러스터에서 AppMirrorRelationship CR 파일(예: `trident-protect-relationship.yaml`)을 편집하고 **spec.desiredState** 값을 `Promoted`로 변경합니다.
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 상태를 확인합니다:

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

페일오버된 복제 관계 재동기화

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

이 프로세스는 복제를 다시 설정하기 전에 타겟 클러스터에서 앱을 중지합니다.

페일오버 중에 타겟 애플리케이션에 기록된 모든 데이터는 손실됩니다.

단계

1. 선택 사항: 소스 클러스터에서 소스 애플리케이션의 스냅샷을 생성합니다. 이렇게 하면 소스 클러스터의 최신 변경 사항이 캡처됩니다.
2. 타겟 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. 장애 조치된 애플리케이션을 보호하기 위해 타겟 클러스터에 보호 일정을 생성한 경우 해당 일정을 제거합니다. 남아 있는 모든 일정은 볼륨 스냅샷 실패의 원인이 됩니다.

페일오버된 복제 관계 역 재동기화

장애 조치 복제 관계를 역동기화하면 타겟 애플리케이션이 소스 애플리케이션이 되고 소스가 타겟이 됩니다. 장애 조치 중에 타겟 애플리케이션에 변경된 내용은 유지됩니다.

단계

1. 원래 타겟 클러스터에서 AppMirrorRelationship CR을 삭제합니다. 이렇게 하면 타겟이 소스가 됩니다. 새 타겟 클러스터에 보호 스케줄이 남아 있는 경우 이를 제거합니다.
2. 원래 관계를 설정하는 데 사용한 CR 파일을 반대 클러스터에 적용하여 복제 관계를 설정합니다.
3. 새 타겟(원본 소스 클러스터)에 AppVault CR이 모두 구성되어 있는지 확인합니다.
4. 반대쪽 클러스터에 복제 관계를 설정하여 반대 방향에 대한 값을 구성합니다.

애플리케이션 복제 방향 반전

복제 방향을 반대로 하면 Trident Protect는 원래 소스 스토리지 백엔드로 계속 복제하면서 애플리케이션을 타겟 스토리지 백엔드로 이동합니다. Trident Protect는 소스 애플리케이션을 중지하고 타겟 앱으로 페일오버하기 전에 데이터를 타겟으로 복제합니다.

이 상황에서는 소스와 타겟을 교체하는 것입니다.

단계

1. 소스 클러스터에서 종료 스냅샷을 생성합니다.

CR을 사용하여 종료 스냅샷 생성

a. 소스 애플리케이션에 대한 보호 정책 스케줄을 비활성화합니다.

b. ShutdownSnapshot CR 파일을 생성합니다.

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

ii. 다음 속성을 구성하십시오.

- **metadata.name**: (필수) 사용자 지정 리소스의 이름입니다.
- **spec.AppVaultRef**: (필수) 이 값은 소스 애플리케이션에 대한 AppVault의 metadata.name 필드와 일치해야 합니다.
- **spec.ApplicationRef**: (필수) 이 값은 소스 애플리케이션 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. `trident-protect-shutdownsnapshot.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI를 사용하여 종료 스냅샷 만들기

a. 종료 스냅샷을 만들고 괄호 안의 값을 사용자 환경의 정보로 바꿉니다. 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 확인합니다.

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

3. 소스 클러스터에서 다음 명령을 사용하여 **shutdownsnapshot.status.appArchivePath** 값을 찾고 파일 경로의 마지막 부분(베이스네임이라고도 함, 마지막 슬래시 뒤의 모든 부분)을 기록합니다.

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

4. 다음과 같은 변경 사항을 적용하여 새 타겟 클러스터에서 새 소스 클러스터로 페일오버를 수행하십시오.

> 장애 조치 절차의 2단계에서 AppMirrorRelationship CR 파일에 `spec.promotedSnapshot` 필드를 포함하고, 그 값을 위 3단계에서 기록한 기본 이름으로 설정합니다.

5. 페일오버된 복제 관계 역 재동기화에서 역방향 재동기화 단계를 수행하십시오.
6. 새 소스 클러스터에서 보호 스케줄을 활성화합니다.

결과

역복제로 인해 다음 작업이 수행됩니다.

- 원본 소스 애플리케이션의 Kubernetes 리소스에 대한 스냅샷이 생성됩니다.
- 원래 소스 앱의 Pod는 앱의 Kubernetes 리소스를 삭제하여 정상적으로 중지됩니다(PVC 및 PV는 그대로 유지됨).
- Pod가 종료된 후 앱 볼륨의 스냅샷이 생성되어 복제됩니다.
- SnapMirror 관계가 끊어져 타겟 볼륨을 읽기/쓰기에 사용할 수 있습니다.
- 앱의 Kubernetes 리소스는 원래 소스 앱이 종료된 후 복제된 볼륨 데이터를 사용하여 종료 전 스냅샷에서 복원됩니다.
- 복제는 역방향으로 다시 설정됩니다.

애플리케이션을 원래 소스 클러스터로 페일백

Trident Protect를 사용하면 다음 일련의 작업을 통해 페일오버 작업 후 "페일 백"을 구현할 수 있습니다. 원래 복제 방향을 복원하는 이 워크플로에서 Trident Protect는 복제 방향을 반전하기 전에 모든 애플리케이션 변경 사항을 원래 소스 애플리케이션으로 복제(재동기화)합니다.

이 프로세스는 타겟으로 페일오버가 완료된 관계에서 시작되며 다음 단계를 포함합니다.

- 장애 조치 상태로 시작합니다.
- 복제 관계를 역방향으로 다시 동기화합니다.

> 일반적인 재동기화 작업을 수행하지 마십시오. 이 작업을 수행하면 장애 조치 절차 중에 타겟 클러스터에 기록된 데이터가 손실됩니다.

- 복제 방향을 반대로 합니다.

단계

1. 페일오버된 복제 관계 역 재동기화 단계를 수행하십시오.
2. 애플리케이션 복제 방향 반전 단계를 수행하십시오.

복제 관계 삭제

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

단계

1. 현재 타겟 클러스터에서 AppMirrorRelationship CR을 삭제합니다.

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

Trident Protect를 사용하여 애플리케이션을 마이그레이션하세요.

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

애플리케이션을 마이그레이션할 때 애플리케이션에 대해 구성된 모든 실행 후크가 애플리케이션과 함께 마이그레이션됩니다. 복원 후 실행 후크가 있는 경우 복원 작업의 일부로 자동으로 실행됩니다.

백업 및 복원 작업

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

동일한 클러스터에 클론 생성

애플리케이션을 동일한 클러스터에 복제하려면 스냅샷 또는 백업을 생성하고 데이터를 동일한 클러스터로 복원합니다.

단계

1. 다음 중 하나를 수행하십시오.
   a. "스냅샷 생성".
   b. "백업 생성".
2. 동일한 클러스터에서 스냅샷을 생성했는지 백업을 생성했는지에 따라 다음 중 하나를 수행하십시오.
   a. "스냅샷에서 데이터 복원".
   b. "백업에서 데이터 복원".

다른 클러스터로 클론

애플리케이션을 다른 클러스터로 복제(클러스터 간 복제)하려면 소스 클러스터에서 백업을 생성한 다음 해당 백업을 다른 클러스터로 복원하십시오. 타겟 클러스터에 Trident Protect가 설치되어 있는지 확인하십시오.

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

단계

1. "백업 생성".
2. 백업이 포함된 오브젝트 스토리지 버킷에 대한 AppVault CR이 타겟 클러스터에 구성되어 있는지 확인합니다.
3. 타겟 클러스터에서 "백업에서 데이터를 복원합니다".

한 스토리지 클래스에서 다른 스토리지 클래스로 애플리케이션 마이그레이션

백업을 타겟 스토리지 클래스로 복원하면 애플리케이션을 한 스토리지 클래스에서 다른 스토리지 클래스로 마이그레이션할 수 있습니다.

예를 들면(복원 CR의 시크릿 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. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-snapshot-restore-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appArchivePath**: 스냅샷 콘텐츠가 저장되는 AppVault 내부의 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

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

   - **spec.appVaultRef**: (필수) 스냅샷 콘텐츠가 저장된 AppVault의 이름입니다.
   - **spec.namespaceMapping**: 복원 작업의 소스 네임스페이스를 타겟 네임스페이스로 매핑합니다. `my-source-namespace` 및 `my-destination-namespace`를 사용자 환경의 정보로 교체하십시오.

     ```
     ---
     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. 선택적으로, 애플리케이션의 특정 리소스만 복원해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가할 수 있습니다.
   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `include or exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
       - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 그룹입니다.
       - **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.
       - **resourceMatchers[].version**: (선택 사항) 필터링할 리소스의 버전입니다.

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

  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. `trident-protect-snapshot-restore-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

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

CLI를 사용하여 스냅샷 복원

단계

1. 스냅샷을 다른 네임스페이스로 복원하고 괄호 안의 값을 사용자 환경의 정보로 바꾸십시오.

   - `snapshot` 인수는 `<namespace>/<name>` 형식의 네임스페이스와 스냅샷 이름을 사용합니다.
   - `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 `source1:dest1,source2:dest2` 형식으로 올바른 타겟 네임스페이스에 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 후크 관리

실행 후크는 관리형 앱의 데이터 보호 작업과 함께 실행되도록 구성할 수 있는 사용자 지정 작업입니다. 예를 들어 데이터베이스 앱이 있는 경우 실행 후크를 사용하여 스냅샷 생성 전에 모든 데이터베이스 트랜잭션을 일시 중지하고 스냅샷 생성이 완료된 후 트랜잭션을 다시 시작할 수 있습니다. 이렇게 하면 애플리케이션 정합성 보장 스냅샷을 생성할 수 있습니다.

실행 후크의 유형

Trident Protect는 실행 가능 시점에 따라 다음과 같은 유형의 실행 후크를 지원합니다.

- 사전 스냅샷
- 스냅샷 이후
- 사전 백업
- 백업 후
- 복원 후
- 페일오버 후

실행 순서

데이터 보호 작업이 실행될 때 실행 후크 이벤트는 다음 순서로 발생합니다.

1. 적용 가능한 사용자 지정 사전 작업 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사전 작업 후크를 생성하고 실행할 수 있지만, 작업 전에 이러한 후크가 실행되는 순서는 보장되지 않으며 구성할 수도 없습니다.
2. 해당되는 경우 파일 시스템이 일시 중지됩니다. "Trident Protect를 사용한 파일 시스템 동결 구성에 대해 자세히 알아보십시오".
3. 데이터 보호 작업이 수행됩니다.
4. 동결된 파일 시스템은 해당되는 경우 동결이 해제됩니다.
5. 적용 가능한 사용자 지정 사후 작업 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사후 작업 후크를 생성하고 실행할 수 있지만, 작업 후 이러한 후크의 실행 순서는 보장되지 않으며 구성할 수도 없습니다.

동일한 유형(예: pre-snapshot)의 실행 후크를 여러 개 생성하는 경우 해당 후크의 실행 순서는 보장되지 않습니다. 하지만 유형이 다른 후크의 실행 순서는 보장됩니다. 예를 들어, 다음은 모든 유형의 후크가 포함된 구성의 실행 순서입니다.

1. 스냅샷 이전 후크가 실행되었습니다
2. 스냅샷 후 후크가 실행되었습니다

3. 사전 백업 후크 실행됨
4. 백업 후 후크 실행됨

앞의 순서 예는 기존 스냅샷을 사용하지 않는 백업을 실행할 때만 적용됩니다.

운영 환경에서 실행 후크 스크립트를 활성화하기 전에 항상 테스트해야 합니다. 'kubectl exec' 명령을 사용하여 스크립트를 편리하게 테스트할 수 있습니다. 운영 환경에서 실행 후크를 활성화한 후에는 생성된 스냅샷과 백업이 일관성을 유지하는지 테스트해야 합니다. 이를 위해 앱을 임시 네임스페이스에 복제하고 스냅샷 또는 백업을 복원한 다음 앱을 테스트하면 됩니다.

스냅샷 실행 전 후크가 Kubernetes 리소스를 추가, 변경 또는 제거하는 경우 이러한 변경 사항은 스냅샷 또는 백업 및 이후의 모든 복원 작업에 포함됩니다.

사용자 지정 실행 후크에 대한 중요 참고 사항

앱 실행 후크를 계획할 때 다음 사항을 고려하십시오.

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

실행 후크는 종종 실행되는 애플리케이션의 기능을 제한하거나 완전히 비활성화하기 때문에 사용자 지정 실행 후크의 실행 시간을 최소화해야 합니다. 실행 후크가 연결된 백업 또는 스냅샷 작업을 시작한 후 취소하더라도 백업 또는 스냅샷 작업이 이미 시작된 경우 후크는 계속 실행될 수 있습니다. 즉, 백업 후 실행 후크에서 사용되는 로직은 백업이 완료되었다고 가정할 수 없습니다.

실행 후크 필터

애플리케이션에 실행 후크를 추가하거나 편집할 때, 후크가 일치할 컨테이너를 관리하기 위해 필터를 추가할 수 있습니다. 필터는 모든 컨테이너에서 동일한 컨테이너 이미지를 사용하지만 각 이미지를 다른 용도로 사용하는 애플리케이션(예: Elasticsearch)에 유용합니다. 필터를 사용하면 실행 후크가 모든 동일한 컨테이너가 아닌 일부 컨테이너에서만 실행되는 시나리오를 만들 수 있습니다. 하나의 실행 후크에 여러 필터를 생성하는 경우 논리 AND 연산자로 결합됩니다. 실행 후크당 최대 10개의 활성 필터를 사용할 수 있습니다.

실행 후크에 추가하는 각 필터는 정규 표현식을 사용하여 클러스터의 컨테이너를 일치시킵니다. 후크가 컨테이너와 일치하면 해당 컨테이너에서 연결된 스크립트를 실행합니다. 필터에 사용되는 정규 표현식은 RE2(Regular Expression 2) 구문을 사용하며, 이 구문은 일치 목록에서 컨테이너를 제외하는 필터를 생성하는 것을 지원하지 않습니다. Trident Protect에서 실행 후크 필터의 정규 표현식에 대해 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 타겟이 서로 다른 네임스페이스에 있는 경우 네임스페이스 필터는 타겟 네임스페이스에만 적용됩니다.

실행 후크 예

https://github.com/NetApp/Verda["NetApp Verda GitHub 프로젝트"]를 방문하여 Apache Cassandra 및 Elasticsearch와 같은 인기 애플리케이션의 실제 실행 후크를 다운로드하세요. 또한 예제를 살펴보고 자신만의 사용자 지정 실행 후크를 구성하는 데 필요한 아이디어를 얻을 수 있습니다.

실행 후크를 생성합니다

Trident Protect를 사용하여 앱에 대한 사용자 지정 실행 후크를 생성할 수 있습니다. 실행 후크를 생성하려면 소유자, 관리자 또는 구성원 권한이 필요합니다.

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-hook.yaml`.
2. 다음 속성을 Trident Protect 환경 및 클러스터 구성에 맞게 구성합니다.

    - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
    - **spec.applicationRef**: (필수) 실행 후크를 실행할 애플리케이션의 Kubernetes 이름입니다.
    - **spec.stage**: (필수) 실행 후크가 실행되어야 하는 작업 중 단계를 나타내는 문자열입니다. 가능한 값:
        - 사전
        - 게시
    - **spec.action**: (필수) 지정된 실행 후크 필터가 모두 일치하는 경우 실행 후크가 수행할 작업을 나타내는 문자열입니다. 가능한 값:
        - 스냅샷
        - 백업
        - 복원
        - 페일오버
    - **spec.enabled**: (선택 사항) 이 실행 후크가 활성화되었는지 비활성화되었는지 나타냅니다. 지정하지 않으면 기본값은 true입니다.
    - **spec.hookSource**: (필수) base64로 인코딩된 훅 스크립트를 포함하는 문자열입니다.
    - **spec.timeout**: (선택 사항) 실행 후크가 실행될 수 있는 시간을 분 단위로 정의하는 숫자입니다. 최소값은 1분이며, 지정하지 않으면 기본값은 25분입니다.
    - **spec.arguments**: (선택 사항) 실행 후크에 지정할 수 있는 인수의 YAML 목록입니다.
    - **spec.matchingCriteria**: (선택 사항) 실행 후크 필터를 구성하는 각 쌍의 기준 키-값 쌍의 선택적 목록입니다. 실행 후크당 최대 10개의 필터를 추가할 수 있습니다.
    - **spec.matchingCriteria.type**: (선택 사항) 실행 후크 필터 유형을 식별하는 문자열입니다. 가능한 값:
        - ContainerImage
        - ContainerName
        - PodName
        - PodLabel
        - NamespaceName
    - **spec.matchingCriteria.value**: (선택 사항) 실행 후크 필터 값을 식별하는 문자열 또는 정규 표현식입니다.

      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: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  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 파일에 올바른 값을 입력한 후 CR을 적용합니다.

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

CLI 사용

단계

1. 실행 후크를 생성할 때 괄호 안의 값을 환경 정보로 바꾸세요. 예를 들면 다음과 같습니다.

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

수동으로 실행 후크 실행

테스트 목적으로 또는 오류 발생 후 수동으로 실행 후크를 다시 실행해야 하는 경우 수동으로 실행 후크를 실행할 수 있습니다. 수동으로 실행 후크를 실행하려면 소유자, 관리자 또는 멤버 권한이 필요합니다.

실행 후크를 수동으로 실행하는 것은 기본적으로 두 단계로 구성됩니다.

1. 리소스 백업을 생성합니다. 이 백업은 리소스를 수집하고 백업을 생성하며 후크가 실행될 위치를 결정합니다
2. 백업에 대해 실행 후크를 실행합니다

1단계: 리소스 백업 생성

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-resource-backup.yaml`.
2. 다음 속성을 Trident Protect 환경 및 클러스터 구성에 맞게 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 리소스 백업을 생성할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 백업 콘텐츠가 저장되는 AppVault의 이름입니다.
   - **spec.appArchivePath**: 백업 콘텐츠가 저장되는 AppVault 내부 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 파일에 올바른 값을 입력한 후 CR을 적용합니다.

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

CLI 사용

단계

1. 백업을 생성할 때 괄호 안의 값을 사용자 환경 정보로 바꿔주세요. 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 확인하세요. 이 예시 명령어를 작업이 완료될 때까지 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>
```

2단계: 실행 후크 실행

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-hook-run.yaml`.
2. 다음 속성을 Trident Protect 환경 및 클러스터 구성에 맞게 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 이 값이 1단계에서 생성한 ResourceBackup CR의 애플리케이션 이름과 일치하는지 확인하십시오.
   - **spec.appVaultRef**: (필수) 이 값이 1단계에서 생성한 ResourceBackup CR의 appVaultRef와 일치하는지 확인하십시오.
   - **spec.appArchivePath**: 이 값이 1단계에서 생성한 ResourceBackup CR의 appArchivePath와 일치하는지 확인하세요.

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

   - **spec.action**: (필수) 지정된 실행 후크 필터가 모두 일치하는 경우 실행 후크가 수행할 작업을 나타내는 문자열입니다. 가능한 값:
     - 스냅샷
     - 백업
     - 복원
     - 페일오버
   - **spec.stage**: (필수) 실행 후크가 실행되어야 하는 작업 중 단계를 나타내는 문자열입니다. 이 후크 실행은 다른 단계의 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 파일에 올바른 값을 입력한 후 CR을 적용합니다.

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

CLI 사용

단계

1. 수동 실행 후크 실행 요청을 생성합니다.

```
tridentctl protect create exechooksrun <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. 실행 후크 실행 상태를 확인합니다. 작업이 완료될 때까지 이 명령을 반복해서 실행할 수 있습니다.

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

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

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```