Astra Control Provisioner를 사용합니다

Astra Control Center

NetApp
August 11, 2025

목차

Astra Control Provisioner를 사용합니다

스토리지 백엔드 암호화를 구성합니다

Astra Control Provisioner를 사용하면 관리형 클러스터와 스토리지 백엔드 간의 트래픽 암호화를 활성화하여 데이터 액세스 보안을 개선할 수 있습니다.

Astra Control Provisioner는 두 가지 유형의 스토리지 백엔드에 대해 Kerberos 암호화를 지원합니다.

- * 온프레미스 ONTAP * - Astra Control Provisioner는 Red Hat OpenShift 및 업스트림 Kubernetes 클러스터에서 온프레미스 ONTAP 볼륨까지 NFSv3 및 NFSv4 연결을 통해 Kerberos 암호화를 지원합니다.
- * Azure NetApp Files * - Astra Control Provisioner는 업스트림 Kubernetes 클러스터에서 Azure NetApp Files 볼륨으로의 NFSv4.1 연결을 통한 Kerberos 암호화를 지원합니다.

따라서 스냅샷을 생성하고, 삭제하고, 크기 조정하고, 스냅샷, 클론 읽기 전용 클론 생성 및 NFS 암호화를 사용하는 볼륨 가져오기

사내 **ONTAP** 볼륨과 전송 중인 **Kerberos** 암호화를 구성합니다

관리 클러스터와 온프레미스 ONTAP 스토리지 백엔드 사이의 스토리지 트래픽에 Kerberos 암호화를 활성화할 수 있습니다.

> 온프레미스 ONTAP 스토리지 백엔드를 통한 NFS 트래픽에 대한 Kerberos 암호화는 를 사용해야만 지원됩니다 `ontap-nas` 스토리지 드라이버.

시작하기 전에

- 가 있는지 확인합니다 "Astra Control Provisioner를 활성화했습니다" 관리 대상 클러스터에서.
- 에 액세스할 수 있는지 확인합니다 `tridentctl` 유틸리티.
- ONTAP 스토리지 백엔드에 대한 관리자 액세스 권한이 있어야 합니다.
- ONTAP 스토리지 백엔드에서 공유할 볼륨의 이름을 알고 있어야 합니다.
- NFS 볼륨에 대한 Kerberos 암호화를 지원하는 ONTAP 스토리지 VM을 준비했는지 확인합니다. 을 참조하십시오 "데이터 LIF에서 Kerberos를 사용하도록 설정합니다" 를 참조하십시오.
- Kerberos 암호화로 사용하는 NFSv4 볼륨이 올바르게 구성되어 있는지 확인합니다. 의 NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하십시오 "NetApp NFSv4의 향상된 기능 및 모범 사례 가이드 를 참조하십시오".

ONTAP 엑스포트 정책을 추가 또는 수정합니다

기존 ONTAP 엑스포트 정책에 규칙을 추가하거나, ONTAP 스토리지 VM 루트 볼륨뿐 아니라 업스트림 Kubernetes 클러스터와 공유된 ONTAP 볼륨에 대해 Kerberos 암호화를 지원하는 새 엑스포트 정책을 생성해야 합니다. 추가한 내보내기 정책 규칙 또는 새로 만든 내보내기 정책은 다음 액세스 프로토콜 및 액세스 권한을 지원해야 합니다.

액세스 프로토콜

NFS, NFSv3 및 NFSv4 액세스 프로토콜을 사용하여 엑스포트 정책을 구성합니다.

액세스 세부 정보

볼륨에 대한 필요에 따라 세 가지 Kerberos 암호화 버전 중 하나를 구성할 수 있습니다.

- * Kerberos 5 * - (인증 및 암호화)
- * Kerberos 5i * - (인증 및 ID 보호 암호화)
- * Kerberos 5p * - (인증 및 암호화, ID 및 개인 정보 보호)

적절한 액세스 권한을 사용하여 ONTAP 엑스포트 정책 규칙을 구성합니다. 예를 들어, 클러스터가 Kerberos 5i 및 Kerberos 5p 암호화가 혼합된 NFS 볼륨을 마운트하는 경우 다음 액세스 설정을 사용합니다.

유형	읽기 전용 액세스	읽기/쓰기 권한	고급 사용자 액세스
Unix	활성화됨	활성화됨	활성화됨
Kerberos 5i	활성화됨	활성화됨	활성화됨
Kerberos 5p	활성화됨	활성화됨	활성화됨

ONTAP 엑스포트 정책과 엑스포트 정책 규칙을 생성하는 방법은 다음 문서를 참조하십시오.

- "엑스포트 정책을 생성합니다"
- "엑스포트 정책에 규칙 추가"

스토리지 백엔드를 생성합니다

Kerberos 암호화 기능을 포함하는 Astra Control Provisioner 스토리지 백엔드 구성을 생성할 수 있습니다.

이 작업에 대해

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 만들 때 를 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다 `spec.nfsMountOptions` 매개 변수:

- `spec.nfsMountOptions: sec=krb5` (인증 및 암호화)
- `spec.nfsMountOptions: sec=krb5i` (인증 및 암호화, ID 보호)
- `spec.nfsMountOptions: sec=krb5p` (인증 및 암호화, ID 및 개인 정보 보호)

Kerberos 수준을 하나만 지정하십시오. 매개 변수 목록에서 Kerberos 암호화 수준을 두 개 이상 지정하면 첫 번째 옵션만 사용됩니다.

단계

1. 관리되는 클러스터에서 다음 예제를 사용하여 스토리지 백엔드 구성 파일을 생성합니다. 괄호 안의 값을 사용자 환경의 정보로 대체:

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. 이전 단계에서 생성한 구성 파일을 사용하여 백엔드를 생성합니다.

```
tridentctl create backend -f <backend-configuration-file>
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 보고 원인을 확인할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 확인하고 수정한 후 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스를 생성합니다

스토리지 클래스를 만들어 Kerberos 암호화를 사용하여 볼륨을 프로비저닝할 수 있습니다.

이 작업에 대해

저장소 클래스 개체를 만들 때 를 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다 `mountOptions` 매개 변수:

- `mountOptions: sec=krb5` (인증 및 암호화)
- `mountOptions: sec=krb5i` (인증 및 암호화, ID 보호)
- `mountOptions: sec=krb5p` (인증 및 암호화, ID 및 개인 정보 보호)

Kerberos 수준을 하나만 지정하십시오. 매개 변수 목록에서 Kerberos 암호화 수준을 두 개 이상 지정하면 첫 번째 옵션만 사용됩니다. 스토리지 백엔드 구성에서 지정한 암호화 수준이 스토리지 클래스 객체에 지정한 레벨과 다른 경우 스토리지 클래스 객체가 우선합니다.

단계

1. 다음 예제를 사용하여 StorageClass Kubernetes 개체를 생성합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
parameters:
  backendType: "ontap-nas"
  storagePools: "ontapnas_pool"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: True
```

2. 스토리지 클래스를 생성합니다.

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. 스토리지 클래스가 생성되었는지 확인합니다.

```
kubectl get sc ontap-nas-sc
```

다음과 유사한 출력이 표시됩니다.

```
NAME            PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

볼륨 프로비저닝

스토리지 백엔드와 스토리지 클래스를 생성한 후 이제 볼륨을 프로비저닝할 수 있습니다. 에 대해서는 이 지침을

참조하십시오 "볼륨 프로비저닝".

Azure NetApp Files 볼륨과 함께 전송 중인 Kerberos 암호화를 구성합니다

관리 클러스터와 단일 Azure NetApp Files 스토리지 백엔드 또는 Azure NetApp Files 스토리지 백엔드의 가상 풀 사이의 스토리지 트래픽에 Kerberos 암호화를 활성화할 수 있습니다.

시작하기 전에

- 관리형 Red Hat OpenShift 클러스터에서 Astra Control Provisioner를 활성화했는지 확인합니다. 을 참조하십시오 "Astra Control Provisioner를 활성화합니다" 를 참조하십시오.
- 에 액세스할 수 있는지 확인합니다 `tridentctl` 유틸리티.
- 요구 사항을 확인하고 의 지침에 따라 Kerberos 암호화용 Azure NetApp Files 스토리지 백엔드를 준비했는지 확인합니다 "Azure NetApp Files 설명서".
- Kerberos 암호화로 사용하는 NFSv4 볼륨이 올바르게 구성되어 있는지 확인합니다. 의 NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하십시오 "NetApp NFSv4의 향상된 기능 및 모범 사례 가이드 를 참조하십시오".

스토리지 백엔드를 생성합니다

Kerberos 암호화 기능을 포함하는 Azure NetApp Files 스토리지 백엔드 구성을 만들 수 있습니다.

이 작업에 대해

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 만들 때 다음 두 가지 가능한 수준 중 하나에 적용되도록 정의할 수 있습니다.

- 를 사용하는 * 스토리지 백엔드 레벨 * `spec.kerberos` 필드에 입력합니다
- 를 사용하는 * 가상 풀 레벨 * `spec.storage.kerberos` 필드에 입력합니다

가상 풀 레벨에서 구성을 정의하면 스토리지 클래스의 레이블을 사용하여 풀이 선택됩니다.

두 레벨에서 Kerberos 암호화의 세 가지 버전 중 하나를 지정할 수 있습니다.

- `kerberos: sec=krb5` (인증 및 암호화)
- `kerberos: sec=krb5i` (인증 및 암호화, ID 보호)
- `kerberos: sec=krb5p` (인증 및 암호화, ID 및 개인 정보 보호)

단계

1. 관리되는 클러스터에서 스토리지 백엔드(스토리지 백엔드 레벨 또는 가상 풀 레벨)를 정의해야 하는 위치에 따라 다음 예제 중 하나를 사용하여 스토리지 백엔드 구성 파일을 생성합니다. 괄호 안의 값을 사용자 환경의 정보로 대체:

스토리지 백엔드 레벨의 예

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-anf-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-anf-secret
```

가상 풀 레벨 예

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-anf-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-anf-secret
```

2. 이전 단계에서 생성한 구성 파일을 사용하여 백엔드를 생성합니다.

```
tridentctl create backend -f <backend-configuration-file>
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 보고 원인을 확인할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 확인하고 수정한 후 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스를 생성합니다

스토리지 클래스를 만들어 Kerberos 암호화를 사용하여 볼륨을 프로비저닝할 수 있습니다.

단계

1. 다음 예제를 사용하여 StorageClass Kubernetes 개체를 생성합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "nfs"
  selector: "type=encryption"
```

2. 스토리지 클래스를 생성합니다.

```
kubectl create -f sample-input/storage-class-anf-sc-nfs.yaml
```

3. 스토리지 클래스가 생성되었는지 확인합니다.

```
kubectl get sc anf-sc-nfs
```

다음과 유사한 출력이 표시됩니다.

```
NAME          PROVISIONER             AGE
anf-sc-nfs    csi.trident.netapp.io   15h
```

볼륨 프로비저닝

스토리지 백엔드와 스토리지 클래스를 생성한 후 이제 볼륨을 프로비저닝할 수 있습니다. 에 대해서는 이 지침을 참조하십시오 "볼륨 프로비저닝".

스냅샷을 사용하여 볼륨 데이터를 복구합니다

Astra Control Provisioner는 를 사용하여 스냅샷에서 데이터를 데이터 이동 없이 빠르게 복원할 수 있도록 합니다 `TridentActionSnapshotRestore` (TASR) CR 이 CR은 필수 Kubernetes 조치로 작동하며 작업이 완료된 후에도 유지되지 않습니다.

Astra Control Provisioner는 에서 스냅샷 복원을 지원합니다 `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `gcp-cvs`, 및 `solidfire-san` 드라이버.

시작하기 전에

바인딩된 PVC 및 사용 가능한 볼륨 스냅샷이 있어야 합니다.

- PVC 상태가 Bound인지 확인한다.

```
kubectl get pvc
```

- 볼륨 스냅샷을 사용할 준비가 되었는지 확인합니다.

```
kubectl get vs
```

단계

1. TASR CR을 생성합니다. 이 예에서는 PVC용 CR을 생성합니다 `pvc1` 및 볼륨 스냅샷 `pvc1-snapshot`.

```
cat tasr-pvc1-snapshot.yaml

apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: this-doesnt-matter
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. CR을 적용하여 스냅샷에서 복원합니다. 이 예는 스냅샷에서 복구합니다 `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml

tridentactionsnapshotrestore.trident.netapp.io/this-doesnt-matter
created
```

결과

Astra Control Provisioner는 스냅샷에서 데이터를 복원합니다. 스냅샷 복구 상태를 확인할 수 있습니다.

```
kubectl get tasr -o yaml

apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: this-doesnt-matter
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- 대부분의 경우 Astra Control Provisioner는 장애 발생 시 작업을 자동으로 재시도하지 않습니다. 작업을 다시 수행해야 합니다.
- 관리자 권한이 없는 Kubernetes 사용자는 애플리케이션 네임스페이스에서 TASR CR을 생성할 수 있는 관리자의 권한을 받아야 할 수 있습니다.

SnapMirror를 사용하여 볼륨을 복제합니다

Astra Control Provisioner를 사용하면 재해 복구를 위한 데이터 복제를 위해 한 클러스터의 소스 볼륨과 피어링된 클러스터의 타겟 볼륨 간에 미러 관계를 생성할 수 있습니다. 이름이 지정된 CRD(사용자 지정 리소스 정의)를 사용하여 다음 작업을 수행할 수 있습니다.

- 볼륨 간 미러 관계 생성(PVC)
- 볼륨 간 미러 관계를 제거합니다
- 미러 관계를 해제합니다
- 재해 상태(페일오버) 중에 2차 볼륨 승격
- 계획된 페일오버 또는 마이그레이션 중에 클러스터에서 클러스터로 애플리케이션의 무손실 전환 수행

복제 사전 요구 사항

시작하기 전에 다음과 같은 사전 요구 사항이 충족되는지 확인하십시오.

ONTAP 클러스터

- * Astra Control Provisioner *: Astra Control Provisioner 버전 23.10 이상 또는 A "Astra Trident를 지원했습니다" ONTAP를 백엔드로 활용하는 소스 및 타겟 Kubernetes 클러스터 모두에 있어야 합니다.
- * 라이센스 *: 소스 및 대상 ONTAP 클러스터 모두에서 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이센스를 활성화해야 합니다. 을 참조하십시오 "ONTAP의 SnapMirror 라이센스 개요" 를 참조하십시오.

피어링

- * 클러스터 및 SVM *: ONTAP 스토리지 백엔드를 피어링해야 합니다. 을 참조하십시오 "클러스터 및 SVM 피어링 개요" 를 참조하십시오.

  두 ONTAP 클러스터 간의 복제 관계에 사용되는 SVM 이름이 고유한지 확인합니다.

- * Astra Control Provisioner 및 SVM *: 피어링된 원격 SVM을 대상 클러스터의 Astra Control Provisioner에서 사용할 수 있어야 합니다.

지원되는 드라이버

- 볼륨 복제는 ONTAP-NAS 및 ONTAP-SAN 드라이버에서 지원됩니다.

대칭 복사된 **PVC**를 작성합니다

다음 단계를 수행하고 CRD 예제를 사용하여 운영 볼륨과 보조 볼륨 간의 미러 관계를 생성합니다.

단계

1. 운영 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. 를 사용하여 StorageClass 객체를 생성합니다 `trident.netapp.io/replication: true` 매개 변수.

      예

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. 이전에 생성된 StorageClass를 사용하여 PVC를 생성합니다.

예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. 로컬 정보로 MirrorRelationship CR을 생성합니다.

예

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Astra Control Provisioner는 볼륨과 볼륨의 현재 DP(Data Protection) 상태에 대한 내부 정보를 가져온 다음 MirrorRelationship의 상태 필드를 채웁니다.

d. TridentMirrorRelationship CR을 가져와 PVC의 내부 이름과 SVM을 얻습니다.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. 보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. trident.netapp.io/replication: true 매개 변수를 사용하여 StorageClass 를 만듭니다.

      예

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        trident.netapp.io/replication: true
      ```

   b. 대상 및 소스 정보를 사용하여 MirrorRelationship CR을 생성합니다.

      예

      ```
      kind: TridentMirrorRelationship
      apiVersion: trident.netapp.io/v1
      metadata:
        name: csi-nas
      spec:
        state: established
        volumeMappings:
        - localPVCName: csi-nas
          remoteVolumeHandle:
      "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
      ```

Astra Control Provisioner는 구성된 관계 정책 이름(또는 ONTAP의 기본값)을 사용하여 SnapMirror 관계를 생성하고 초기화합니다.

c. 이전에 생성한 StorageClass를 사용하여 PVC를 생성하여 보조(SnapMirror 대상) 역할을 합니다.

예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Astra Control Provisioner가 TridentMirrorRelationship CRD를 확인하고 관계가 없는 경우 볼륨을 생성하지 못합니다. 이 관계가 있으면 Astra Control Provisioner는 새로운 FlexVol 볼륨을 MirrorRelationship에 정의된 원격 SVM과 함께 피어링된 SVM에 배치합니다.

볼륨 복제 상태입니다

Trident Mirror Relationship(TMR)은 PVC 간 복제 관계의 한쪽 끝을 나타내는 CRD입니다. 대상 TMR에는 Astra Control Provisioner에게 원하는 상태를 알려주는 상태가 있습니다. 대상 TMR의 상태는 다음과 같습니다.

- * 설립 * : 로컬 PVC는 미러 관계의 대상 볼륨이며, 이것은 새로운 관계입니다.
- * 승진된 * : 로컬 PVC는 현재 유효한 미러 관계가 없는 ReadWrite 및 마운트 가능합니다.
- * 재설립 * : 로컬 PVC는 미러 관계의 대상 볼륨이며 이전에 해당 미러 관계에 있었습니다.
  - 대상 볼륨이 대상 볼륨 내용을 덮어쓰므로 대상 볼륨이 소스 볼륨과 관계가 있는 경우 다시 설정된 상태를 사용해야 합니다.
  - 볼륨이 소스와 이전에 관계가 없는 경우 재설정된 상태가 실패합니다.

비계획 페일오버 중에 보조 **PVC**를 승격합니다

보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.

- TridentMirrorRelationship의 _spec.state_field를 로 업데이트합니다 `promoted`.

계획된 페일오버 중에 보조 **PVC**를 승격합니다

계획된 장애 조치(마이그레이션) 중에 다음 단계를 수행하여 보조 PVC를 승격합니다.

단계

1. 운영 Kubernetes 클러스터에서 PVC의 스냅샷을 생성하고 스냅샷이 생성될 때까지 기다립니다.
2. 운영 Kubernetes 클러스터에서 SnapshotInfo CR을 생성하여 내부 세부 정보를 가져옵니다.

   예

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. 보조 Kubernetes 클러스터에서 *TridentMirrorRelationship_CR* 의 *_spec.state_field*를 *_promitted* 및 *spec.promotedSnapshotHandle* 으로 업데이트하여 스냅샷의 내부 이름으로 업데이트합니다.
4. 보조 Kubernetes 클러스터에서 승격될 TridentMirrorRelationship의 상태(status.state 필드)를 확인합니다.

페일오버 후 미러 관계를 복구합니다

미러 관계를 복구하기 전에 새 1차 사이트로 만들 측면을 선택합니다.

단계

1. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 _spec.remoteVolumeHandle_field 값이 업데이트되었는지 확인합니다.
2. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 _spec.mirror_field를 로 업데이트합니다 `reestablished`.

추가 작업

Astra Control Provisioner는 운영 볼륨과 2차 볼륨에서 다음 작업을 지원합니다.

1차 **PVC**를 새로운 **2**차 **PVC**로 복제합니다

이미 1차 PVC와 2차 PVC가 있는지 확인하십시오.

단계

1. 설정된 보조(대상) 클러스터에서 PersistentVolumeClaim 및 TridentMirrorRelationship CRD를 삭제합니다.
2. 운영(소스) 클러스터에서 TridentMirrorRelationship CRD를 삭제합니다.
3. 설정하려는 새 2차(대상) PVC에 대해 1차(소스) 클러스터에 새 TridentMirrorRelationship CRD를 생성합니다.

대칭 복사**, 1**차 또는 **2**차 **PVC**의 크기를 조정합니다

PVC는 평소대로 크기를 조정할 수 있으며, 데이터 양이 현재 크기를 초과할 경우 ONTAP는 자동으로 대상 flevxols를 확장합니다.

PVC에서 복제를 제거합니다

복제를 제거하려면 현재 보조 볼륨에 대해 다음 작업 중 하나를 수행합니다.

- 2차 PVC에서 MirrorRelationship을 삭제합니다. 이렇게 하면 복제 관계가 끊어집니다.
- 또는 spec.state 필드를 _promessed_로 업데이트합니다.

PVC 삭제**(**이전에 미러링됨**)**

Astra Control Provisioner는 복제된 PVC를 확인하고 볼륨을 삭제하기 전에 복제 관계를 해제합니다.

TMR을 삭제합니다

미러링된 관계의 한 쪽에서 TMR을 삭제하면 Astra Control Provisioner가 삭제를 완료하기 전에 나머지 TMR이 _promessed_state로 전환됩니다. 삭제하도록 선택한 TMR이 이미 _PROJED_STATE에 있는 경우 기존 미러 관계가 없으며 TMR이 제거되고 Astra Control Provisioner가 로컬 PVC를 _ReadWrite_로 승격합니다. 이렇게 삭제하면 ONTAP의 로컬 볼륨에 대한 SnapMirror 메타데이터가 해제됩니다. 이 볼륨이 향후 미러 관계에 사용될 경우 새 미러 관계를 생성할 때 _establed_volume 복제 상태의 새 TMR을 사용해야 합니다.

ONTAP가 온라인 상태일 때 미러 관계를 업데이트합니다

미러 관계는 설정된 후 언제든지 업데이트할 수 있습니다. 를 사용할 수 있습니다 `state: promoted` 또는 `state: reestablished` 관계를 업데이트하는 필드
대상 볼륨을 일반 ReadWrite 볼륨으로 승격할 때 _promotedSnapshotHandle_을 사용하여 현재 볼륨을 복구할 특정 스냅샷을 지정할 수 있습니다.

ONTAP이 오프라인일 때 미러 관계를 업데이트합니다

Astra Control이 ONTAP 클러스터에 직접 연결되지 않은 상태에서 CRD를 사용하여 SnapMirror 업데이트를 수행할 수 있습니다. 다음 TridentActionMirrorUpdate 예제 형식을 참조하십시오.

예

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` TridentActionMirrorUpdate CRD의 상태를 반영합니다. 이 값은 *SUCCEEDED*,*In Progress* 또는_Failed_에서 가져올 수 있습니다.