Astra Trident를 사용해 보십시오

Astra Trident

NetApp
April 16, 2024

This PDF was generated from https://docs.netapp.com/ko-kr/trident-2201/trident-use/anf.html on April 16, 2024. Always check docs.netapp.com for the latest.

목차

Astra Trident를 사용해 보십시오 . . . 1
백엔드 구성 . . . 1
kubectl로 백엔드를 만듭니다 . . . 67
kubeck을 사용하여 백엔드 관리 수행 . . . 73
tridentctl을 사용하여 백엔드 관리를 수행합니다 . . . 75
백엔드 관리 옵션 간 이동 . . . 76
스토리지 클래스를 관리합니다 . . . 83
볼륨 작업을 수행합니다 . . . 85
작업자 노드를 준비합니다 . . . 110
작업자 노드 자동 준비 . . . 113
Astra Trident를 모니터링합니다 . . . 113

Astra Trident를 사용해 보십시오

백엔드 구성

백엔드는 Astra Trident와 스토리지 시스템 간의 관계를 정의합니다. Astra Trident가 스토리지 시스템과 통신하는 방법과 Astra Trident가 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다. Astra Trident는 스토리지 클래스에 정의된 요구 사항과 일치하는 백엔드에서 스토리지 풀을 자동으로 제공합니다. 사용 중인 스토리지 시스템 유형에 따라 백엔드를 구성하는 방법에 대해 자세히 알아보십시오.

- "Azure NetApp Files 백엔드를 구성합니다"
- "Google Cloud Platform 백엔드에 Cloud Volumes Service를 구성합니다"
- "NetApp HCI 또는 SolidFire 백엔드를 구성합니다"
- "ONTAP 또는 Cloud Volumes ONTAP NAS 드라이버를 사용하여 백엔드를 구성합니다"
- "ONTAP 또는 Cloud Volumes ONTAP SAN 드라이버를 사용하여 백엔드를 구성합니다"
- "NetApp ONTAP용 Amazon FSx와 함께 Astra Trident를 사용하십시오"

Azure NetApp Files 백엔드를 구성합니다

제공된 샘플 구성을 사용하여 Astra Trident 설치의 백엔드로 Azure NetApp Files(ANF)를 구성하는 방법에 대해 알아보십시오.

> Azure NetApp Files 서비스는 100GB 미만의 볼륨을 지원하지 않습니다. Astra Trident는 더 작은 볼륨을 요청하는 경우 100GB 볼륨을 자동으로 생성합니다.

필요한 것

를 구성하고 사용합니다 "Azure NetApp Files" 백엔드, 다음이 필요합니다.

- Azure NetApp Files가 활성화된 Azure 구독의 'SubscriptionID'입니다.
- 테난티디, 클라이언트ID, 그리고 고객비밀 "앱 등록" Azure NetApp Files 서비스에 대한 충분한 권한이 있는 Azure Active Directory에서 앱 등록은 Azure에서 미리 정의한 소유자 또는 기고자 역할을 사용해야 합니다.

  > Azure 기본 제공 역할에 대한 자세한 내용은 를 참조하십시오 "Azure 문서".

- 하나 이상의 Azure 위치가 있다 "위임된 서브넷". Trident 22.01부터 "location" 매개 변수는 백엔드 구성 파일의 최상위 수준에 있는 필수 필드입니다. 가상 풀에 지정된 위치 값은 무시됩니다.
- Azure NetApp Files를 처음 사용하거나 새 위치에서 사용하는 경우 일부 초기 구성이 필요합니다. 를 참조하십시오 "빠른 시작 가이드".

이 작업에 대해

Trident는 백엔드 구성(서브넷, 가상 네트워크, 서비스 수준 및 위치)을 기반으로 요청된 위치에서 사용 가능한 용량 풀에 ANF 볼륨을 생성하고 요청된 서비스 수준 및 서브넷과 일치시킵니다.

> 참고: Astra Trident는 수동 QoS 용량 풀을 지원하지 않습니다.

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개 변수	설명	기본값
'내전'		항상 1
'torageDriverName'입니다	스토리지 드라이버의 이름입니다	"Azure-NetApp-파일"
백엔드이름	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + 임의 문자
'스크립트 ID'입니다	Azure 구독의 구독 ID입니다	
텐antID	앱 등록에서 테넌트 ID입니다	
'클라이언트 ID'	앱 등록의 클라이언트 ID입니다	
'clientSecret	앱 등록에서 클라이언트 암호	
'저급'	'초표준', '프리미엄', '울트라' 중 하나	""(임의)
위치	새 볼륨을 생성할 Azure 위치의 이름입니다	
'저급'	'초표준', '프리미엄', '울트라' 중 하나	""(임의)
[재치 단체	검색된 자원을 필터링하기 위한 자원 그룹 목록입니다	"[]"(필터 없음)
'netap계정'	검색된 리소스를 필터링하기 위한 NetApp 계정의 목록입니다	"[]"(필터 없음)
용량풀	검색된 리소스를 필터링하기 위한 용량 풀 목록입니다	"[]"(필터 없음, 임의)
가상네트워크	위임된 서브넷이 있는 가상 네트워크의 이름입니다	""
'우방'	Microsoft.Netapp/volumes` 에 위임된 서브넷의 이름입니다	""
nfsMountOptions를 선택합니다	NFS 마운트 옵션에 대한 세밀한 제어	"nfsvers=3"
LimitVolumeSize	요청된 볼륨 크기가 이 값보다 큰 경우 용량 할당에 실패합니다	""(기본적으로 적용되지 않음)
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: "\{"api":false, "method":true, "discovery":true}". 문제 해결 중이 아니며 자세한 로그 덤프가 필요한 경우가 아니면 이 방법을 사용하지 마십시오.	null입니다

PVC를 생성하려고 할 때 "용량 풀을 찾을 수 없음" 오류가 발생하면 앱 등록에 필요한 권한과 리소스(서브넷, 가상 네트워크, 용량 풀)가 연결되지 않은 것일 수 있습니다. Astra Trident는 디버깅을 설정할 때 백엔드가 생성될 때 검색한 Azure 리소스를 기록합니다. 적절한 역할이 사용되고 있는지 확인하십시오.

> NFS 버전 4.1을 사용하여 볼륨을 마운트하려는 경우 쉼표로 구분된 마운트 옵션 목록에 "nfsvers=4'를 포함하여 NFS v4.1을 선택할 수 있습니다. 스토리지 클래스에 설정된 마운트 옵션은 백엔드 구성 파일에 설정된 마운트 옵션을 재정의합니다.

"소스 그룹", "넷계정", "용량 풀", "virtualNetwork" 및 "ubnet"의 값은 짧은 이름 또는 정규화된 이름을 사용하여 지정할 수 있습니다. 짧은 이름은 이름이 같은 여러 리소스와 일치할 수 있으므로 대부분의 경우 정규화된 이름을 사용하는 것이 좋습니다. "소스 그룹", "netap계정" 및 "용량 풀" 값은 검색된 리소스 세트를 이 스토리지 백엔드에서 사용할 수 있는 리소스로 제한하며 어떤 조합으로도 지정할 수 있는 필터입니다. 정규화된 이름은 다음 형식입니다.

유형	형식
리소스 그룹	리소스 그룹>
NetApp 계정	리소스 그룹>/<NetApp 계정>
용량 풀	리소스 그룹>/<NetApp 계정>/<용량 풀>
가상 네트워크	리소스 그룹>/<가상 네트워크>
서브넷	리소스 그룹>/<가상 네트워크>/<서브넷>

구성 파일의 특수 섹션에서 다음 옵션을 지정하여 각 볼륨의 프로비저닝 방식을 기본적으로 제어할 수 있습니다. 아래의 구성 예를 참조하십시오.

매개 변수	설명	기본값
엑포트 규칙	새 볼륨의 내보내기 규칙	"0.0.0.0/0"
나프산디렉토리	스냅샷 디렉터리의 표시 여부를 제어합니다	"거짓"
'크기'입니다	새 볼륨의 기본 크기입니다	"100G"
유니크권한	새 볼륨의 UNIX 사용 권한(8진수 4자리)	""(미리보기 기능, 가입 시 화이트리스트 필요)

exportRule 값은 IPv4 주소 또는 IPv4 서브넷을 CIDR 표기로 조합한 쉼표로 구분된 목록이어야 합니다.

> ANF 백엔드에서 생성된 모든 볼륨의 경우, Astra Trident는 스토리지 풀에 있는 모든 레이블을 프로비저닝할 때 스토리지 볼륨에 복사합니다. 스토리지 관리자는 스토리지 풀별로 레이블을 정의하고 스토리지 풀에서 생성된 모든 볼륨을 그룹화할 수 있습니다. 이를 통해 백엔드 구성에서 제공되는 사용자 지정 가능한 레이블 세트를 기반으로 볼륨을 쉽게 구별할 수 있습니다.

예 **1:** 최소 구성

이는 절대적인 최소 백엔드 구성입니다. 이 구성을 통해 Astra Trident는 구성된 위치에서 ANF에 위임된 모든 NetApp 계정, 용량 풀 및 서브넷을 검색하고 해당 풀 및 서브넷 중 하나에 무작위로 새 볼륨을 배치합니다.

이 구성은 ANF를 사용하여 지금 막 시작하는 데 이상적이지만, 실제로 용량 할당을 수행하는 볼륨에 대한 추가적인 범위 지정을 제공하려는 경우에 적합합니다.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "location": "eastus"
}
```

예 **2:** 용량 풀 필터를 사용하는 특정 서비스 수준 구성

이 백엔드 구성은 Azure의 "eastus" 위치에 볼륨을 "Ultra" 용량 풀에 배치합니다. Astra Trident는 해당 위치의 ANF에 위임된 모든 서브넷을 자동으로 검색하여 그 중 하나에 무작위로 새 볼륨을 배치합니다.

```
    {
        "version": 1,
        "storageDriverName": "azure-netapp-files",
        "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
        "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
        "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
        "clientSecret": "SECRET",
        "location": "eastus",
        "serviceLevel": "Ultra",
        "capacityPools": [
            "application-group-1/account-1/ultra-1",
            "application-group-1/account-1/ultra-2"
],
    }
```

예 **3:** 고급 구성

이 백엔드 구성은 단일 서브넷에 대한 볼륨 배치 범위를 더욱 줄여주고 일부 볼륨 프로비저닝 기본값도 수정합니다.

```
    {
        "version": 1,
        "storageDriverName": "azure-netapp-files",
        "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
        "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
        "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
        "clientSecret": "SECRET",
        "location": "eastus",
        "serviceLevel": "Ultra",
        "capacityPools": [
            "application-group-1/account-1/ultra-1",
            "application-group-1/account-1/ultra-2"
],
        "virtualNetwork": "my-virtual-network",
        "subnet": "my-subnet",
        "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
        "limitVolumeSize": "500Gi",
        "defaults": {
            "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
            "snapshotDir": "true",
            "size": "200Gi",
            "unixPermissions": "0777"
=======
        }
    }
```

예 **4:** 가상 스토리지 풀 구성

이 백엔드 구성은 단일 파일에 여러 스토리지 풀을 정의합니다. 다양한 서비스 수준을 지원하는 여러 용량 풀이 있고 이를 나타내는 Kubernetes의 스토리지 클래스를 생성하려는 경우에 유용합니다.

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "9f87c765-4774-fake-ae98-a721add45451",
    "tenantID": "68e4f836-edc1-fake-bff9-b2d865ee56cf",
    "clientID": "dd043f63-bf8e-fake-8076-8de91e5713aa",
    "clientSecret": "SECRET",
    "location": "eastus",
    "resourceGroups": ["application-group-1"],
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "labels": {
        "cloud": "azure"
    },
    "location": "eastus",

    "storage": [
        {
            "labels": {
                "performance": "gold"
            },
            "serviceLevel": "Ultra",
            "capacityPools": ["ultra-1", "ultra-2"]
        },
        {
            "labels": {
                "performance": "silver"
            },
            "serviceLevel": "Premium",
            "capacityPools": ["premium-1"]
        },
        {
            "labels": {
                "performance": "bronze"
            },
            "serviceLevel": "Standard",
            "capacityPools": ["standard-1", "standard-2"]
        }
    ]
}
```

다음 'torageClass' 정의는 위의 스토리지 풀을 참조합니다. parameters.selector` 필드를 사용하면 볼륨을 호스팅하는 데 사용되는 각 'torageClass'에 대해 지정할 수 있습니다. 볼륨은 선택한 풀에 정의된 측면을 갖습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze"
allowVolumeExpansion: true
```

다음 단계

백엔드 구성 파일을 생성한 후 다음 명령을 실행합니다.

```
tridentctl create backend -f <backend-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.

CVS for GCP 백엔드를 구성합니다

제공된 샘플 구성을 사용하여 Astra Trident 설치를 위한 백엔드로 NetApp CVS(Cloud Volumes Service) for Google Cloud Platform(GCP)을 구성하는 방법에 대해 알아보십시오.

> NetApp Cloud Volumes Service for Google Cloud는 CVS를 지원하지 않습니다. 크기 100GiB 미만의 성능 볼륨 또는 크기 300GiB 미만의 CVS 볼륨을 지원합니다. Astra Trident는 요청된 볼륨이 최소 크기보다 작은 경우 최소 크기의 볼륨을 자동으로 생성합니다.

필요한 것

를 구성하고 사용합니다 "Google Cloud용 Cloud Volumes Service" 백엔드, 다음이 필요합니다.

- NetApp CVS로 구성된 Google Cloud 계정
- Google Cloud 계정의 프로젝트 번호입니다
- Google Cloud 서비스 계정에는 netcloudvolumes.admin 역할이 포함되어 있습니다
- CVS 서비스 계정에 대한 API 키 파일입니다

이제 Astra Trident에 더 작은 볼륨에 대한 지원이 기본 제공됩니다 "GCP의 CVS 서비스 유형입니다". 로 만든 백엔드의 경우 `storageClass=software`이제 볼륨에 최소 프로비저닝 크기가 300GiB가 됩니다. CVS는 현재 제어된 가용성 하에서 이 기능을 제공하며 기술 지원을 제공하지 않습니다. 사용자는 1TiB 미만의 볼륨에 액세스하려면 등록해야 합니다 "여기". NetApp은 고객이 비운영 워크로드에 1TiB 미만의 볼륨을 사용할 것을 권장합니다.

> 기본 CVS 서비스 유형('storageClass=software')을 사용하여 백엔드를 배포할 때 사용자는 해당 프로젝트 번호 및 프로젝트 ID에 대해 GCP에서 1TiB 미만의 볼륨 기능에 대한 액세스 권한을 얻어야 합니다. 이는 Astra Trident에서 1TiB 미만의 볼륨을 프로비저닝하는 데 필요합니다. 그렇지 않은 경우 600GiB 미만의 PVC에 대해 체적 생성에 실패합니다. 를 사용하여 1TiB 미만의 볼륨에 대한 액세스 권한을 얻습니다 "이 양식입니다".

기본 CVS 서비스 레벨을 위해 Astra Trident에서 생성한 볼륨은 다음과 같이 프로비저닝됩니다.

- 300GiB보다 작은 PVC로 인해 Astra Trident가 300GiB CVS 볼륨을 생성합니다.
- 300GiB ~ 600GiB의 PVC로 인해 Astra Trident가 요청된 크기의 CVS 볼륨을 생성합니다.
- 600GiB와 1TiB 사이의 PVC로 인해 Astra Trident가 1TiB CVS 볼륨을 생성합니다.
- 1TiB보다 큰 PVC로 인해 Astra Trident가 요청된 크기의 CVS 볼륨을 생성합니다.

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개 변수	설명	기본값
'내전'		항상 1
'torageDriverName'입니다	스토리지 드라이버의 이름입니다	"GCP-CV"
백엔드이름	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + API 키의 일부
'원어클래스'	스토리지 유형입니다. 하드웨어(성능 최적화) 또는 소프트웨어(CVS 서비스 유형)를 선택합니다.	

매개 변수	설명	기본값
'프로젝트 번호'	Google Cloud 계정 프로젝트 번호입니다. 이 값은 Google Cloud 포털의 홈 페이지에서 확인할 수 있습니다.	
아피지역	CVS 계정 지역. 백엔드에서 볼륨을 프로비저닝할 영역입니다.	
아피키	Google Cloud 서비스 계정의 API 키로, 'netappcloudvolumes.admin' 역할을 수행합니다. 여기에는 Google Cloud 서비스 계정의 개인 키 파일(백엔드 구성 파일에 verbatim 복사)의 JSON 형식 콘텐츠가 포함됩니다.	
proxyURL	프록시 서버가 CVS 계정에 연결해야 하는 경우 프록시 URL입니다. 프록시 서버는 HTTP 프록시 또는 HTTPS 프록시일 수 있습니다. HTTPS 프록시의 경우 프록시 서버에서 자체 서명된 인증서를 사용할 수 있도록 인증서 유효성 검사를 건너뜁니다. 인증이 활성화된 프록시 서버는 지원되지 않습니다.	
nfsMountOptions를 선택합니다	NFS 마운트 옵션에 대한 세밀한 제어	"nfsvers=3"
LimitVolumeSize	요청된 볼륨 크기가 이 값보다 큰 경우 용량 할당에 실패합니다	""(기본적으로 적용되지 않음)
'저급'	새 볼륨에 대한 CVS 서비스 수준입니다. 값은 "표준", "프리미엄" 및 "익스트림"입니다.	"표준"
네트워크	CVS 볼륨에 사용되는 GCP 네트워크	“기본값”
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: "\{"api":false,"method":true}". 문제 해결 중이 아니며 자세한 로그 덤프가 필요한 경우가 아니면 이 방법을 사용하지 마십시오.	null입니다

공유 VPC 네트워크를 사용하는 경우 projectNumber와 hostProjectNumber를 모두 지정해야 합니다. 이 경우 projectNumber는 서비스 프로젝트이고 hostProjectNumber는 호스트 프로젝트입니다.

아피지역(apiRegion)은 Astra Trident가 CVS 볼륨을 생성하는 GCP 지역을 나타냅니다. 지역 간 Kubernetes 클러스터를 생성할 때 "apiRegion"에서 생성된 CVS 볼륨을 여러 GCP 지역 노드에 예약된 워크로드에 사용할 수 있습니다. 지역 간 트래픽에는 추가 비용이 발생합니다.

- 교차 지역 액세스를 활성화하려면 "allowedTopologies"에 대한 StorageClass 정의에 모든 영역이 포함되어야 합니다. 예를 들면 다음과 같습니다.

```
- key: topology.kubernetes.io/region
  values:
  - us-east1
  - europe-west1
```

- 'torageClass'는 원하는 항목을 선택하는 데 사용할 수 있는 선택적 매개 변수입니다 "CVS 서비스 유형입니다". 기본 CVS 서비스 유형('S torageClass=software') 또는 Trident에서 기본적으로 사용하는 CVS 성능 서비스 유형('torageClass=hardware')을 선택할 수 있습니다. 백엔드 정의에 각 CVS의 torageClass를 제공하는 apiRegion을 지정해야 합니다.

Astra Trident가 Google Cloud에 기본 CVS 서비스 유형과 통합된 베타 기능은, 운영 워크로드용으로 제공되지 않습니다. **Trident**는 CVS에서 완벽하게 지원됨- 성능 서비스 유형으로 기본적으로 사용됩니다.

각 백엔드는 단일 Google Cloud 지역에 볼륨을 프로비저닝합니다. 다른 영역에 볼륨을 생성하려면 추가 백엔드를 정의할 수 있습니다.

구성 파일의 특수 섹션에서 다음 옵션을 지정하여 각 볼륨의 프로비저닝 방식을 기본적으로 제어할 수 있습니다. 아래의 구성 예를 참조하십시오.

매개 변수	설명	기본값
엑포트 규칙	새 볼륨의 내보내기 규칙	"0.0.0.0/0"
나프산디렉토리	'.snapshot' 디렉토리에 액세스합니다	"거짓"
안산예비역	스냅숏용으로 예약된 볼륨의 백분율입니다	""(CVS 기본값 0 허용)
'크기'입니다	새 볼륨의 크기입니다	"100Gi"

exportRule 값은 IPv4 주소 또는 IPv4 서브넷을 CIDR 표기로 조합한 쉼표로 구분된 목록이어야 합니다.

CVS Google Cloud 백엔드에서 생성된 모든 볼륨에 대해 Trident는 스토리지 풀에 있는 모든 레이블을 프로비저닝할 때 스토리지 볼륨에 복사합니다. 스토리지 관리자는 스토리지 풀별로 레이블을 정의하고 스토리지 풀에서 생성된 모든 볼륨을 그룹화할 수 있습니다. 이를 통해 백엔드 구성에서 제공되는 사용자 지정 가능한 레이블 세트를 기반으로 볼륨을 쉽게 구별할 수 있습니다.

예 **1:** 최소 구성

이는 절대적인 최소 백엔드 구성입니다.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
```

```
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    }
}
```


예 **2:** 기본 **CVS** 서비스 유형 구성

이 예에서는 기본 CVS 서비스 유형을 사용하는 백엔드 정의를 보여 줍니다. 이는 범용 워크로드에 사용되며, 높은 조널

가용성과 함께 가벼운/중간 수준의 성능을 제공합니다.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "storageClass": "software",
    "apiRegion": "us-east4",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
```

```
sa%40my-gcp-project.iam.gserviceaccount.com"
    }
}
```

예 **3:** 단일 서비스 수준 구성

이 예에서는 Google Cloud Us-west2 지역에서 Astra Trident가 생성한 모든 스토리지에 동일한 측면을 적용하는 백엔드 파일을 보여 줍니다. 이 예제에서는 백엔드 구성 파일에서 proxyURL을 사용하는 방법도 보여 줍니다.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
```

```
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    },
    "proxyURL": "http://proxy-server-hostname/",
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
    "limitVolumeSize": "10Ti",
    "serviceLevel": "premium",
    "defaults": {
        "snapshotDir": "true",
        "snapshotReserve": "5",
        "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
        "size": "5Ti"
    }
}
```

예 **4:** 가상 스토리지 풀 구성

이 예에서는 가상 스토리지 풀을 사용하여 구성된 백엔드 정의 파일과 이를 다시 참조하는 'torageClasses'를 보여줍니다.

아래 표시된 샘플 백엔드 정의 파일에서 모든 스토리지 풀에 대한 특정 기본값이 설정되어 있는데, 이 경우 '스냅샷 보존'이 5%로 설정되고 'exportRule'이 0.0.0.0/0으로 설정됩니다. 가상 스토리지 풀은 '스토리지' 섹션에 정의되어 있습니다. 이 예에서는 각 개별 스토리지 풀이 자체 'erviceLevel'을 설정하고 일부 풀이 기본값을 덮어씁니다.

```
{
    "version": 1,
    "storageDriverName": "gcp-cvs",
    "projectNumber": "012345678901",
    "apiRegion": "us-west2",
    "apiKey": {
        "type": "service_account",
        "project_id": "my-gcp-project",
        "private_key_id": "1234567890123456789012345678901234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisI
sAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSa
PIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZN
chRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlz
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl
```

/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHi
sIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOgu
SaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyA
ZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz
lzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrt
HisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbO
guSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKe
yAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRA
GzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
        "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
        "client_id": "123456789012345678901",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
        "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
    },
    "nfsMountOptions": "vers=3,proto=tcp,timeo=600",

    "defaults": {
        "snapshotReserve": "5",
        "exportRule": "0.0.0.0/0"
    },

    "labels": {
        "cloud": "gcp"
    },
    "region": "us-west2",

    "storage": [
        {
            "labels": {
                "performance": "extreme",
                "protection": "extra"

```
            },
            "serviceLevel": "extreme",
            "defaults": {
                "snapshotDir": "true",
                "snapshotReserve": "10",
                "exportRule": "10.0.0.0/24"
            }
        },
        {
            "labels": {
                "performance": "extreme",
                "protection": "standard"
            },
            "serviceLevel": "extreme"
        },
        {
            "labels": {
                "performance": "premium",
                "protection": "extra"
            },
            "serviceLevel": "premium",
            "defaults": {
                "snapshotDir": "true",
                "snapshotReserve": "10"
            }
        },

        {
            "labels": {
                "performance": "premium",
                "protection": "standard"
            },
            "serviceLevel": "premium"
        },

        {
            "labels": {
                "performance": "standard"
            },
            "serviceLevel": "standard"
        }
    ]
}
```

다음 StorageClass 정의는 위의 스토리지 풀을 참조합니다. "parameters.selector` 필드를 사용하여 각 StorageClass에 볼륨을 호스팅하는 데 사용되는 가상 풀을 지정할 수 있습니다. 볼륨은 선택한 풀에 정의된 측면을

갖습니다.

첫 번째 StorageClass('cvs-extreme-extra-protection')는 첫 번째 가상 스토리지 풀에 매핑됩니다. 이 풀은 스냅샷 예약 공간이 10%인 최고 성능을 제공하는 유일한 풀입니다. 마지막 StorageClass('cvs-extra-protection')는 10%의 스냅샷 예비 공간을 제공하는 스토리지 풀을 호출합니다. Astra Trident는 선택된 가상 스토리지 풀을 결정하고 스냅샷 예약 요구 사항이 충족되는지 확인합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
```

```
provisioner: netapp.io/trident
parameters:
  selector: "performance=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

다음 단계

백엔드 구성 파일을 생성한 후 다음 명령을 실행합니다.

```
tridentctl create backend -f <backend-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.

NetApp HCI 또는 SolidFire 백엔드를 구성합니다

Astra Trident 설치와 함께 Element 백엔드를 생성하고 사용하는 방법에 대해 알아보십시오.

필요한 것

- Element 소프트웨어를 실행하는 지원되는 스토리지 시스템
- 볼륨을 관리할 수 있는 NetApp HCI/SolidFire 클러스터 관리자 또는 테넌트 사용자에 대한 자격 증명
- 모든 Kubernetes 작업자 노드에 적절한 iSCSI 툴이 설치되어 있어야 합니다. 을 참조하십시오 "작업자 노드 준비 정보".

알아야 할 내용

'솔드파이어-SAN' 스토리지 드라이버는 파일 및 블록의 볼륨 모드를 모두 지원합니다. 파일 시스템 볼륨 코드의 경우 Astra Trident가 볼륨을 생성하고 파일 시스템을 생성합니다. 파일 시스템 유형은 StorageClass에 의해 지정됩니다.

드라이버	프로토콜	볼륨 모드	액세스 모드가 지원됩니다	지원되는 파일 시스템
'솔더불-산'	iSCSI	블록	RWO, ROX, rwx	파일 시스템이 없습니다. 원시 블록 장치.
'솔더불-산'	iSCSI	블록	RWO, ROX, rwx	파일 시스템이 없습니다. 원시 블록 장치.
'솔더불-산'	iSCSI	파일 시스템	RWO, ROX	xfs, ext3, ext4
'솔더불-산'	iSCSI	파일 시스템	RWO, ROX	xfs, ext3, ext4

Astra Trident는 향상된 CSI Provisioner로 작동할 때 CHAP를 사용합니다. CHAP(CSI의 기본값)를 사용하는 경우 추가 준비가 필요하지 않습니다. 비 CSI Trident와 CHAP를 사용하려면 명시적으로 UseCHAP 옵션을 설정하는 것이 좋습니다. 그렇지 않으면 를 참조하십시오 "여기".

볼륨 액세스 그룹은 Astra Trident의 기존 비 CSI 프레임워크에서만 지원됩니다. CSI 모드에서 작동하도록 구성된 경우 Astra Trident는 CHAP를 사용합니다.

AccessGroups나 UseCHAP가 설정되지 않은 경우 다음 규칙 중 하나가 적용됩니다.

- 기본 '트리덴트' 액세스 그룹이 감지되면 액세스 그룹이 사용됩니다.
- 액세스 그룹이 감지되지 않고 Kubernetes 버전이 1.7 이상인 경우 CHAP가 사용됩니다.

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개 변수	설명	기본값
'내전'		항상 1
'torageDriverName'입니다	스토리지 드라이버의 이름입니다	항상 "solidfire-san"
백엔드이름	사용자 지정 이름 또는 스토리지 백엔드	“SolidFire_” + 스토리지(iSCSI) IP 주소입니다
끝점	테넌트 자격 증명이 있는 SolidFire 클러스터의 MVIP입니다	
'VIP'	스토리지(iSCSI) IP 주소 및 포트	
'라벨'	볼륨에 적용할 임의의 JSON 형식 레이블 세트입니다.	“”
테난트이름	사용할 테넌트 이름(찾을 수 없는 경우 생성됨)	

매개 변수	설명	기본값
이니토IFace	iSCSI 트래픽을 특정 호스트 인터페이스로 제한합니다	“기본값”
'UseCHAP’입니다	CHAP를 사용하여 iSCSI를 인증합니다	참
"액세스 그룹"	사용할 액세스 그룹 ID 목록입니다	"트리덴트"라는 액세스 그룹의 ID를 찾습니다.
'유형'	QoS 사양	
LimitVolumeSize	요청된 볼륨 크기가 이 값보다 큰 경우 용량 할당에 실패합니다	“”(기본적으로 적용되지 않음)
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: {“api”:false, “method”:true}	null입니다

문제 해결 및 자세한 로그 덤프가 필요한 경우가 아니면 debugTraceFlags를 사용하지 마십시오.

생성된 모든 볼륨에 대해 Astra Trident는 스토리지 풀에 있는 모든 레이블을 프로비저닝할 때 해당 스토리지 LUN에 복사합니다. 스토리지 관리자는 스토리지 풀별로 레이블을 정의하고 스토리지 풀에서 생성된 모든 볼륨을 그룹화할 수 있습니다. 이를 통해 백엔드 구성에서 제공되는 사용자 지정 가능한 레이블 세트를 기반으로 볼륨을 쉽게 구별할 수 있습니다.

예 **1:** 에 대한 백엔드 구성 `solidfire-san` 세 가지 볼륨 유형을 가진 드라이버

이 예에서는 CHAP 인증을 사용하는 백엔드 파일을 보여 주고 특정 QoS 보장을 포함하는 세 가지 볼륨 유형을 모델링합니다. 그런 다음 "IOPS" 스토리지 클래스 매개 변수를 사용하여 각 스토리지 클래스를 사용할 스토리지 클래스를 정의할 가능성이 높습니다.

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://<user>:<password>@<mvip>/json-rpc/8.0",
    "SVIP": "<svip>:3260",
    "TenantName": "<tenant>",
    "labels": {"k8scluster": "dev1", "backend": "dev1-element-cluster"},
    "UseCHAP": true,
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS": 2000,
"burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS": 6000,
"burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS": 8000,
"burstIOPS": 10000}}]
}
```

예 **2:** 에 대한 백엔드 및 스토리지 클래스 구성 `solidfire-san` 가상 스토리지 풀이 있는 드라이버

이 예에서는 가상 스토리지 풀과 이를 다시 참조하는 StorageClasses와 함께 구성된 백엔드 정의 파일을 보여 줍니다.

아래 표시된 샘플 백엔드 정의 파일에서 모든 스토리지 풀에 대해 특정 기본값이 설정되어 있으며, 이 경우 Silver에서 'type'을 설정합니다. 가상 스토리지 풀은 '스토리지' 섹션에 정의되어 있습니다. 이 예에서는 일부 스토리지 풀이 자체 유형을 설정하고 일부 풀은 위에 설정된 기본값을 덮어씁니다.

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://<user>:<password>@<mvip>/json-rpc/8.0",
    "SVIP": "<svip>:3260",
    "TenantName": "<tenant>",
    "UseCHAP": true,
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS": 2000,
"burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS": 6000,
"burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS": 8000,
"burstIOPS": 10000}}],

    "type": "Silver",
    "labels":{"store":"solidfire", "k8scluster": "dev-1-cluster"},
    "region": "us-east-1",

    "storage": [
        {
            "labels":{"performance":"gold", "cost":"4"},
            "zone":"us-east-1a",
            "type":"Gold"
        },
        {
            "labels":{"performance":"silver", "cost":"3"},
            "zone":"us-east-1b",
            "type":"Silver"
        },
        {
            "labels":{"performance":"bronze", "cost":"2"},
            "zone":"us-east-1c",
            "type":"Bronze"
        },
        {
            "labels":{"performance":"silver", "cost":"1"},
            "zone":"us-east-1d"
        }
    ]
}
```

다음 StorageClass 정의는 위의 가상 스토리지 풀을 참조합니다. parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다. 선택한 가상 풀에 볼륨이 정의되어 있습니다.

첫 번째 StorageClass(스몰드파이어-골드-포)는 첫 번째 가상 스토리지 풀에 매핑됩니다. 골드의 Volume Type QoS로

골드 성능을 제공하는 유일한 풀입니다. 마지막 스토리지클래스(스몰드파이어-실버)는 실버 성능을 제공하는 스토리지 풀을 호출합니다. Astra Trident가 선택한 가상 스토리지 풀을 결정하고 스토리지 요구 사항을 충족해 줍니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold; cost=4"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=3"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze; cost=2"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=1"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
  fsType: "ext4"
```

자세한 내용을 확인하십시오

- "볼륨 액세스 그룹"

ONTAP 또는 Cloud Volumes ONTAP SAN 드라이버를 사용하여 백엔드를 구성합니다

ONTAP 및 Cloud Volumes ONTAP SAN 드라이버를 사용하여 ONTAP 백엔드를 구성하는 방법에 대해 알아보십시오.

- "준비"
- "구성 및 예"

사용자 권한

Astra Trident는 일반적으로 "admin" 클러스터 사용자 또는 "vsadmin" SVM 사용자를 사용하거나 동일한 역할을 가진 다른 이름을 가진 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. NetApp ONTAP 구축을 위한 Amazon FSx의 경우 Astra Trident는 클러스터 'fsxadmin' 사용자 또는 'vsadmin' SVM 사용자 또는 동일한 역할을 가진 다른 이름의 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. 'fsxadmin' 사용자는 클러스터 관리자 사용자에게 제한된 교체품입니다.

> 'limitAggregateUsage' 매개 변수를 사용하는 경우 클러스터 관리자 권한이 필요합니다. Astra Trident와 함께 NetApp ONTAP용 Amazon FSx를 사용하는 경우 "limitAggregateUsage" 매개 변수는 "vsadmin" 및 "fsxadmin" 사용자 계정과 작동하지 않습니다. 이 매개 변수를 지정하면 구성 작업이 실패합니다.

ONTAP 내에서 Trident 드라이버가 사용할 수 있는 보다 제한적인 역할을 만들 수 있지만 권장하지 않습니다. Trident의 대부분의 새로운 릴리즈에서는 추가 API를 호출하므로 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

준비

ONTAP SAN 드라이버를 사용하여 ONTAP 백엔드를 구성하는 방법에 대해 알아보십시오. 모든 ONTAP 백엔드의 경우, Astra Trident는 SVM에 하나 이상의 Aggregate가 할당되어 있어야 합니다.

또한 둘 이상의 드라이버를 실행하고 둘 중 하나를 가리키는 스토리지 클래스를 생성할 수도 있습니다. 예를 들어, ONTAP-SAN 드라이버와 ONTAP-SAN-이코노미 클래스를 사용하는 '기본 클래스'를 사용하는 'san-dev' 클래스를 구성할 수 있습니다.

모든 Kubernetes 작업자 노드에 적절한 iSCSI 툴이 설치되어 있어야 합니다. 을 참조하십시오 "여기" 를 참조하십시오.

인증

Astra Trident는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 필요한 권한이 있는 ONTAP 사용자의 사용자 이름 및 암호입니다. ONTAP 버전과의 호환성을 최대한 보장하기 위해 admin 또는 vsadmin과 같은 미리 정의된 보안 로그인 역할을 사용하는 것이 좋습니다.
- 인증서 기반: Astra Trident는 백엔드에 설치된 인증서를 사용하여 ONTAP 클러스터와 통신할 수도 있습니다. 이 경우 백엔드 정의에는 클라이언트 인증서, 키 및 사용할 경우 신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값이 있어야 합니다(권장).

사용자는 기존 백엔드를 업데이트할 수도 있으며 자격 증명 기반 에서 인증서 기반 으로, 그 반대로 전환할 수도 있습니다. 자격 증명과 인증서가 둘 다 제공된 경우 *, Astra Trident는 기본적으로 인증서를 사용하는 동시에 백엔드

정의에서 자격 증명을 제거하는 경고를 표시합니다.

자격 증명 기반 인증을 사용합니다

Astra Trident는 SVM 범위/클러스터 범위 관리자에게 ONTAP 백엔드와 통신하기 위한 자격 증명을 요구합니다. admin 또는 vsadmin과 같이 미리 정의된 표준 역할을 사용하는 것이 좋습니다. 이를 통해 향후 Astra Trident 릴리스에서 사용할 기능 API를 노출할 수 있는 향후 ONTAP 릴리스와 향후 호환성이 보장됩니다. 사용자 지정 보안 로그인 역할은 Astra Trident와 함께 생성 및 사용할 수 있지만 권장되지 않습니다.

백엔드 정의의 예는 다음과 같습니다.

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret",
}
```

백엔드 정의는 자격 증명이 일반 텍스트로 저장되는 유일한 위치라는 점에 유의하십시오. 백엔드가 생성된 후 사용자 이름/암호는 Base64로 인코딩되어 Kubernetes 암호로 저장됩니다. 백엔드의 생성/업딩은 자격 증명에 대한 지식이 필요한 유일한 단계입니다. 따라서 Kubernetes/스토리지 관리자가 수행할 수 있는 관리 전용 작업입니다.

인증서 기반 인증을 사용합니다

신규 및 기존 백엔드는 인증서를 사용하여 ONTAP 백엔드와 통신할 수 있습니다. 백엔드 정의에는 세 가지 매개 변수가 필요합니다.

- clientCertificate: Base64로 인코딩된 클라이언트 인증서 값입니다.
- clientPrivateKey: Base64 - 연결된 개인 키의 인코딩된 값입니다.
- TrustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개 변수를 제공해야 합니다. 신뢰할 수 있는 CA가 사용되지 않으면 이 작업을 무시할 수 있습니다.

일반적인 워크플로에는 다음 단계가 포함됩니다.

단계

1. 클라이언트 인증서 및 키를 생성합니다. 생성 시 CN(일반 이름)을 ONTAP 사용자로 설정하여 인증하십시오.

   ```
   openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
   -out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
   ```

2. 신뢰할 수 있는 CA 인증서를 ONTAP 클러스터에 추가합니다. 이는 스토리지 관리자가 이미 처리한 것일 수 있습니다. 트러스트된 CA가 사용되지 않으면 무시합니다.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. ONTAP 클러스터에 클라이언트 인증서 및 키(1단계)를 설치합니다.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. ONTAP 보안 로그인 역할이 인증서 인증 방법을 지원하는지 확인합니다.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. 생성된 인증서를 사용하여 인증을 테스트합니다. ONTAP 관리 LIF> 및 <SVM 이름>을 관리 LIF IP 및 SVM 이름으로 바꿉니다.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Base64로 인증서, 키 및 신뢰할 수 있는 CA 인증서를 인코딩합니다.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 이전 단계에서 얻은 값을 사용하여 백엔드를 생성합니다.

```
$ cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaallllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

$ tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                  UUID                  |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

인증 방법을 업데이트하거나 자격 증명을 회전합니다

기존 백엔드를 업데이트하여 다른 인증 방법을 사용하거나 해당 자격 증명을 회전할 수 있습니다. 이렇게 하면 사용자 이름/암호를 사용하는 백엔드를 인증서를 사용하도록 업데이트할 수 있고 인증서를 사용하는 백엔드는 사용자 이름/암호 기반으로 업데이트할 수 있습니다. 이를 위해서는 tridentctl backend update를 실행하는 데 필요한 parameter가 포함된 update 된 backend.json 파일을 사용한다.

```
$ cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "secret",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
$ tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                  UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

> 암호를 회전할 때 스토리지 관리자는 먼저 ONTAP에서 사용자의 암호를 업데이트해야 합니다. 그 다음에는 백엔드 업데이트가 있습니다. 인증서를 회전할 때 여러 인증서를 사용자에게 추가할 수 있습니다. 그런 다음 백엔드가 업데이트되어 새 인증서를 사용합니다. 그러면 ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스가 중단되거나 이후에 생성된 볼륨 연결에 영향을 미치지 않습니다. 백엔드 업데이트가 성공적이면 Astra Trident가 ONTAP 백엔드와 통신하고 향후 볼륨 작업을 처리할 수 있음을 나타냅니다.

Igroup을 지정합니다

Astra Trident에서 igroup을 사용하여 프로비저닝하는 볼륨(LUN)에 대한 액세스를 제어합니다. 관리자는 백엔드에 대한 igroup을 지정할 때 다음 두 가지 옵션을 사용할 수 있습니다.

- Astra Trident는 백엔드에 따라 igroup을 자동으로 생성하고 관리할 수 있습니다. 만약 'triviName'이 백엔드 정의에 포함되지 않으면, Astra Trident는 SVM에 'trident-<backend-UUID>'라는 igroup을 생성합니다. 그러면 각 백엔드에 전용 igroup이 있고 Kubernetes 노드 IQN의 자동 추가/삭제를 처리합니다.
- 또는 미리 생성된 igroup을 백엔드 정의로 제공할 수도 있습니다. 이 작업은 'show'(이름) config 매개 변수를 사용하여 수행할 수 있습니다. Astra Trident가 기존 igroup에 Kubernetes 노드 IQN을 추가/삭제합니다.

'인명이름'이 정의된 백엔드의 경우, '인명이름'을 '트리멘틀백엔드 업데이트'를 사용하여 삭제하고 Astra Trident에서 Igroup을 자동 처리할 수 있습니다. 이 경우 워크로드에 이미 연결된 볼륨에 대한 액세스가 중단되지 않습니다. 생성된 igroup Astra Trident를 사용하여 향후 연결을 처리할 것입니다.

> Astra Trident의 각 고유 인스턴스에 대해 igroup을 할당하는 것은 Kubernetes 관리자 및 스토리지 관리자에게 유용한 모범 사례입니다. CSI Trident는 클러스터 노드 IQN을 igroup에 추가 및 제거하여 관리를 크게 단순화합니다. 전용 igroup을 사용하여 Kubernetes 환경(및 Astra Trident 설치)에서 동일한 SVM을 사용할 경우 한 Kubernetes 클러스터의 변경 사항이 다른 Kubernetes 클러스터와 관련된 igroup에 영향을 미치지 않도록 합니다. 또한 Kubernetes 클러스터의 각 노드에 고유한 IQN이 있는지 확인하는 것도 중요합니다. 위에 언급한 바와 같이, Astra Trident는 IQN의 추가 및 제거를 자동으로 처리합니다. 호스트 간에 IQN을 재사용하면 호스트가 서로 잘못 인식되어 LUN에 대한 액세스가 거부되는 바람직하지 않은 시나리오가 발생할 수 있습니다.

Astra Trident가 CSI Provisioner로 작동하도록 구성된 경우 Kubernetes 노드 IQN이 igroup에 자동으로 추가/제거됩니다. Kubernetes 클러스터에 노드를 추가하면, "trident-CSI" DemonSet는 새로 추가된 노드에 pod("trident-CSI-xxxxx")를 배포하고 볼륨을 연결할 수 있는 새 노드를 등록합니다. 노드 IQN도 백엔드의 igroup에 추가됩니다. 이와 유사한 일련의 단계에서는 Kubernetes에서 노드에 코드로닝, 드레이닝 및 삭제가 발생하는 경우 IQN 제거를 처리합니다.

Astra Trident가 CSI Provisioner로 실행되지 않을 경우, Kubernetes 클러스터의 모든 작업자 노드에서 iSCSI IQN을 포함하도록 igroup을 수동으로 업데이트해야 합니다. Kubernetes 클러스터에 참여하는 노드의 IQN을 igroup에 추가해야 합니다. 마찬가지로, Kubernetes 클러스터에서 제거된 노드의 IQN을 igroup에서 제거해야 합니다.

양방향 **CHAP**를 사용하여 연결을 인증합니다

Astra Trident는 ONTAP-SAN과 ONTAP-SAN 절약 드라이버에 양방향 CHAP를 사용하여 iSCSI 세션을 인증할 수 있습니다. 이를 위해서는 백엔드 정의에서 'useCHAP' 옵션을 사용해야 합니다. "true"로 설정하면 Astra Trident가 SVM의 기본 이니시에이터 보안을 양방향 CHAP로 구성하고 백엔드 파일의 사용자 이름과 암호를 설정합니다. 양방향 CHAP를 사용하여 연결을 인증하는 것이 좋습니다. 다음 샘플 구성을 참조하십시오.

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "FaKePaSsWoRd",
    "igroupName": "trident",
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}
```

> useCHAP는 한 번만 설정할 수 있는 Boolean 옵션이다. 기본적으로 false로 설정되어 있습니다. true로 설정한 후에는 false 로 설정할 수 없습니다.

useCHAP=true 외에, chapInitatorSecret, chapTargetInitialatorSecret, chapchTargetUsername, chapUsername 필드가 백엔드 정의에 포함되어야 합니다. tridentctl update를 실행하여 백엔드를 생성한 후 비밀을 변경할 수 있다.

작동 방식

스토리지 관리자는 useCHAP를 true로 설정하여 스토리지 백엔드에서 CHAP를 구성하도록 Astra Trident에 지시합니다. 여기에는 다음이 포함됩니다.

- SVM에서 CHAP 설정:
  - SVM의 기본 이니시에이터 보안 유형이 없음(기본값으로 설정) * 이고 * 볼륨에 이미 기존 LUN이 없는 경우 Astra Trident는 기본 보안 유형을 "CHAP"로 설정하고 CHAP 이니시에이터 및 대상 사용자 이름 및 암호 구성을 진행합니다.
  - SVM에 LUN이 포함된 경우 Astra Trident는 SVM에서 CHAP를 활성화하지 않습니다. 따라서 SVM에 이미 있는 LUN에 대한 액세스가 제한되지 않습니다.
- CHAP 이니시에이터 및 타겟 사용자 이름과 암호를 구성합니다. 이러한 옵션은 백엔드 구성에 지정해야 합니다(위 참조).
- 백엔드에 있는 'si. 이름'에 이니셜레이터 추가 관리 지정되지 않은 경우 기본적으로 트리덴트가 사용됩니다.

백엔드가 생성된 후 Astra Trident는 해당 'tridentbackend' CRD를 생성하고 CHAP 비밀과 사용자 이름을 Kubernetes 비밀로 저장합니다. 이 백엔드에서 Astra Trident에 의해 생성된 모든 PVS는 CHAP를 통해 마운트되고 연결됩니다.

자격 증명을 회전하고 백엔드를 업데이트합니다

backend.json 파일에서 CHAP 파라미터를 업데이트하여 CHAP 자격 증명을 업데이트할 수 있다. 이렇게 하려면 CHAP 암호를 업데이트하고 "tridentctl update" 명령을 사용하여 이러한 변경 사항을 반영해야 합니다.

> 백엔드의 CHAP 암호를 업데이트할 때 "tridentctl"을 사용하여 백엔드를 업데이트해야 합니다. Astra Trident에서 변경 사항을 선택할 수 없으므로 CLI/ONTAP UI를 통해 스토리지 클러스터의 자격 증명을 업데이트하지 마십시오.

```
$ cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "FaKePaSsWoRd",
    "igroupName": "trident",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

$ ./tridentctl update backend ontap_san_chap -f backend-san.json -n
trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

기존 연결은 영향을 받지 않습니다. SVM에서 Astra Trident가 자격 증명을 업데이트하면 활성 상태로 유지됩니다. 새 연결은 업데이트된 자격 증명을 사용하며 기존 연결은 계속 활성 상태로 유지됩니다. 기존 PVS를 연결 해제하고 다시 연결하면 업데이트된 자격 증명을 사용하게 됩니다.

구성 옵션 및 예

Astra Trident 설치를 통해 ONTAP SAN 드라이버를 생성하고 사용하는 방법에 대해 알아보십시오. 이 섹션에서는 백엔드 구성 예제 및 백엔드를 StorageClasses에 매핑하는 방법에 대한 세부 정보를 제공합니다.

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개 변수	설명	기본값
'내전'		항상 1

매개 변수	설명	기본값
'torageDriverName'입니다	스토리지 드라이버의 이름입니다	"ONTAP-NAS", "ONTAP-NAS-이코노미", "ONTAP-NAS-Flexgroup", "ONTAP-SAN", "ONTAP-SAN-이코노미"
백엔드이름	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
마나멘타LIF	클러스터 또는 SVM 관리 LIF의 IP 주소입니다	"10.0.0.1", "[2001:1234:ABCD::fee]"
다타LIF	프로토콜 LIF의 IP 주소입니다. IPv6의 경우 대괄호를 사용합니다. 설정한 후에는 업데이트할 수 없습니다	지정되지 않은 경우 SVM에서 파생됩니다
'useCHAP'입니다	CHAP를 사용하여 ONTAP SAN 드라이버에 대한 iSCSI 인증 [부울]	거짓
챕터시크릿	CHAP 이니시에이터 암호입니다. useCHAP=true인 경우 필수입니다	""
'라벨'	볼륨에 적용할 임의의 JSON 형식 레이블 세트입니다	""
챕터타겟이니터시크릿	CHAP 타겟 이니시에이터 암호입니다. useCHAP=true인 경우 필수입니다	""
'chapUsername'입니다	인바운드 사용자 이름입니다. useCHAP=true인 경우 필수입니다	""
'chapTargetUsername'입니다	대상 사용자 이름입니다. useCHAP=true인 경우 필수입니다	""
'고객증명서'	Base64 - 클라이언트 인증서의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다	""
'clientPrivateKey'입니다	Base64 - 클라이언트 개인 키의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다	""
신탁인증서다	Base64 - 신뢰할 수 있는 CA 인증서의 인코딩된 값입니다. 선택 사항. 인증서 기반 인증에 사용됩니다	""
'사용자 이름'	클러스터/SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다	""
"암호"	클러스터/SVM에 연결하는 암호 자격 증명 기반 인증에 사용됩니다	""
'VM'입니다	사용할 스토리지 가상 머신입니다	SVM 'managementLIF'가 지정된 경우에 파생됩니다
"인명이름"입니다	사용할 SAN 볼륨에 대한 igroup의 이름입니다	"삼중 - <backend-UUID>"

매개 변수	설명	기본값
'토르agePrefix'	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사 설정한 후에는 업데이트할 수 없습니다	"삼중류"
제한선태사용법	사용량이 이 비율을 초과하면 프로비저닝이 실패합니다. ONTAP * 용 아마존 FSx에는 * 가 적용되지 않습니다	“”(기본적으로 적용되지 않음)
LimitVolumeSize	요청된 볼륨 크기가 이 값보다 큰 경우 용량 할당에 실패합니다.	“”(기본적으로 적용되지 않음)
'오만유연한'	FlexVol당 최대 LUN 수는 범위[50, 200]에 있어야 합니다.	“100”
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: {“api”:false, “method”:true}	null입니다
'useREST'	ONTAP REST API를 사용하는 부울 매개 변수입니다. * 기술 미리 보기 *	거짓

> "useREST"는 프로덕션 워크로드가 아닌 테스트 환경에 권장되는** 기술 미리 보기로 제공됩니다. "true"로 설정하면 Astra Trident는 ONTAP REST API를 사용하여 백엔드와 통신합니다. 이 기능을 사용하려면 ONTAP 9.9 이상이 필요합니다. 또한 사용되는 ONTAP 로그인 역할은 ONTAP 애플리케이션에 대한 액세스 권한이 있어야 합니다. 이는 미리 정의된 vsadmin과 cluster-admin의 역할에서 충족됩니다.

ONTAP 클러스터와 통신하려면 인증 매개 변수를 제공해야 합니다. 보안 로그인 또는 설치된 인증서의 사용자 이름/암호일 수 있습니다.

> NetApp ONTAP 백엔드에 Amazon FSx를 사용하는 경우 'limitAggregateUsage' 매개 변수를 지정하지 마십시오. NetApp ONTAP용 Amazon FSx에서 제공하는 "fsxadmin" 및 "vsadmin" 역할에는 애그리게이트 사용을 검색하고 Astra Trident를 통해 제한하는 데 필요한 액세스 권한이 없습니다.

> 문제 해결 및 자세한 로그 덤프가 필요한 경우가 아니면 debugTraceFlags를 사용하지 마십시오.

'ONTAP-SAN' 드라이버의 경우, 기본값은 SVM의 모든 데이터 LIF IP를 사용하고 iSCSI 다중 경로를 사용하는 것입니다. 'ONTAP-SAN' 드라이버에 데이터 LIF의 IP 주소를 지정하면 다중 경로를 비활성화하고 지정된 주소만 사용하게 됩니다.

> 백엔드를 생성할 때 생성 후에는 dataLIF와 toragePrefix를 수정할 수 없습니다. 이러한 매개 변수를 업데이트하려면 새 백엔드를 생성해야 합니다.

'ONTAP 이름’을 클러스터에 이미 생성된 igroup으로 설정할 수 있습니다. 지정하지 않으면 Astra Trident가 트리덴트-<backend-UUID>라는 igroup을 자동으로 생성합니다. 미리 정의된 횟수 이름을 제공하는 경우, 환경 간에 SVM을 공유하려면 Kubernetes 클러스터 당 igroup을 사용하는 것이 좋습니다. 이는 Astra Trident가 IQN 추가/삭제를 자동으로 유지 관리하는 데 필요합니다.

백엔드는 또한 생성 후 igroup을 업데이트할 수 있습니다.

- ACA Trident 외부의 SVM에서 생성 및 관리되는 새로운 igroup을 가리키도록 특정 igroup 이름을 업데이트할 수 있습니다.
- 고객 이름을 생략할 수 있습니다. 이 경우 Astra Trident가 트리덴트 - <backend-UUID> igroup을 자동으로 생성하고 관리합니다.

두 경우 모두 볼륨 첨부 파일에 계속 액세스할 수 있습니다. 향후 볼륨 첨부 파일은 업데이트된 igroup을 사용합니다. 이 업데이트는 백엔드에 있는 볼륨에 대한 액세스를 방해하지 않습니다.

'managementLIF' 옵션에 대해 FQDN(정규화된 도메인 이름)을 지정할 수 있습니다.

모든 ONTAP 드라이버에 대한 manementLIF도 IPv6 주소로 설정할 수 있습니다. '--use-ipv6' 플래그를 사용하여 Trident를 설치하십시오. 대괄호 안에 'managementLIF' IPv6 주소를 정의할 때는 주의해야 합니다.

> IPv6 주소를 사용할 때는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같은 대괄호 안에 'managementLIF' 및 'dataLIF'(백엔드 정의에 포함된 경우)가 정의되어 있는지 확인하십시오. 다타 LIF가 제공되지 않으면 Astra Trident가 SVM에서 IPv6 데이터 LIF를 가져옵니다.

ONTAP-SAN 드라이버가 CHAP를 사용하도록 설정하려면 백엔드 정의에서 useCHAP 매개 변수를 true로 설정합니다. 그러면 Astra Trident가 백엔드에 제공된 SVM에 대한 기본 인증으로 양방향 CHAP를 구성하고 사용합니다. 을 참조하십시오 "여기" 작동 방법에 대해 알아보십시오.

ONTAP-SAN-이코노미 드라이버의 경우 LimitVolumeSize 옵션도 qtree 및 LUN에 대해 관리하는 볼륨의 최대 크기를 제한합니다.

> Astra Trident는 "ONTAP-SAN" 드라이버를 사용하여 생성된 모든 볼륨의 "Comments" 필드에 제공 레이블을 설정합니다. 생성된 각 볼륨에 대해 FlexVol의 "Comments" 필드는 스토리지 풀에 있는 모든 레이블로 채워집니다. 스토리지 관리자는 스토리지 풀별로 레이블을 정의하고 스토리지 풀에서 생성된 모든 볼륨을 그룹화할 수 있습니다. 이를 통해 백엔드 구성에서 제공되는 사용자 지정 가능한 레이블 세트를 기반으로 볼륨을 쉽게 구별할 수 있습니다.

볼륨 프로비저닝을 위한 백엔드 구성 옵션

구성의 특수 섹션에서 이러한 옵션을 사용하여 각 볼륨이 기본적으로 프로비저닝되는 방식을 제어할 수 있습니다. 예를 들어, 아래 구성 예제를 참조하십시오.

매개 변수	설명	기본값
'팩시배부	LUN에 대한 공간 할당	"참"
'예비공간'	공간 예약 모드, "없음"(씬) 또는 " 볼륨"(일반)	"없음"
냅샷정책	사용할 스냅샷 정책입니다	"없음"
"qosPolicy"	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드에서 qosPolicy 또는 adapativeQosPolicy 중 하나를 선택합니다	“”

매개 변수	설명	기본값
적응성 QosPolicy	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀 /백엔드에서 qosPolicy 또는 adapativeQosPolicy 중 하나를 선택합니다	“”
안산예비역	스냅샷 "0"에 예약된 볼륨의 백분율	"스냅샷 정책"이 "없음"이면 "없음"
'plitOnClone’을 선택합니다	생성 시 상위 클론에서 클론을 분할합니다	"거짓"
'plitOnClone’을 선택합니다	생성 시 상위 클론에서 클론을 분할합니다	"거짓"
암호화	NetApp 볼륨 암호화를 활성화합니다	"거짓"
'생태성 스타일’을 참조하십시오	새로운 볼륨에 대한 보안 스타일	“UNIX”
'계층화 정책'	"없음"을 사용하는 계층화 정책	ONTAP 9.5 이전 SVM-DR 구성의 경우 "스냅샷 전용

Astra Trident와 함께 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 비공유 QoS 정책 그룹을 사용하고 정책 그룹이 각 구성요소별로 적용되었는지 확인하는 것이 좋습니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 대해 상한을 적용합니다.

다음은 기본값이 정의된 예입니다.

```
{
 "version": 1,
 "storageDriverName": "ontap-san",
 "managementLIF": "10.0.0.1",
 "dataLIF": "10.0.0.2",
 "svm": "trident_svm",
 "username": "admin",
 "password": "password",
 "labels": {"k8scluster": "dev2", "backend": "dev2-sanbackend"},
 "storagePrefix": "alternate-trident",
 "igroupName": "custom",
 "debugTraceFlags": {"api":false, "method":true},
 "defaults": {
     "spaceReserve": "volume",
     "qosPolicy": "standard",
     "spaceAllocation": "false",
     "snapshotPolicy": "default",
     "snapshotReserve": "10"
 }
}
```

> 'ONTAP-SAN' 드라이버를 사용하여 생성된 모든 볼륨의 경우, Astra Trident가 FlexVol에 10%의 용량을 추가하여 LUN 메타데이터를 수용합니다. LUN은 사용자가 PVC에서 요청하는 정확한 크기로 프로비저닝됩니다. Astra Trident가 FlexVol에 10%를 더합니다(ONTAP에서 사용 가능한 크기로 표시). 이제 사용자가 요청한 가용 용량을 얻을 수 있습니다. 또한 이 변경으로 인해 사용 가능한 공간이 완전히 활용되지 않는 한 LUN이 읽기 전용이 되는 것을 방지할 수 있습니다. ONTAP-SAN-경제에는 적용되지 않습니다.

'스냅샷 보존'을 정의하는 백엔드의 경우 Astra Trident는 다음과 같이 볼륨의 크기를 계산합니다.

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

1.1은 LUN 메타데이터를 수용하도록 FlexVol에 추가된 10%의 Astra Trident입니다. 나프산예비공간 = 5%, PVC 요청 = 5GiB의 경우 총 용적 크기는 5.79GiB이고 사용 가능한 크기는 5.5GiB입니다. 'volume show' 명령은 다음 예와 유사한 결과를 표시합니다.

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
          _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                                  online     RW         10GB     5.00GB    0%
          _pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d
                                  online     RW       5.79GB     5.50GB    0%
          _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                                  online     RW          1GB    511.8MB    0%
3 entries were displayed.
```

현재 기존 볼륨에 대해 새 계산을 사용하는 유일한 방법은 크기 조정입니다.

최소 구성의 예

다음 예에서는 대부분의 매개 변수를 기본값으로 두는 기본 구성을 보여 줍니다. 이는 백엔드를 정의하는 가장 쉬운 방법입니다.

> Astra Trident가 있는 NetApp ONTAP에서 Amazon FSx를 사용하는 경우 IP 주소 대신 LIF에 대한 DNS 이름을 지정하는 것이 좋습니다.

`ontap-san` 인증서 기반 인증을 사용하는 드라이버

이는 최소 백엔드 구성의 예입니다. clientCertificate, clientPrivateKey, trustedCACertificate(신뢰할 수 있는 CA를 사용하는 경우 선택 사항)가 backend.json에 채워지고 클라이언트 인증서, 개인 키, 트러스트된 CA 인증서의 base64로 인코딩된 값을 각각 가져갑니다.

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "DefaultSANBackend",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
    "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
    "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz"
}
```

`ontap-san` 양방향 **CHAP**가 있는 드라이버

이는 최소 백엔드 구성의 예입니다. 이 기본 구성은 useCHAP가 true로 설정된 ONTAP-SAN 백엔드를 생성합니다.

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "labels": {"k8scluster": "test-cluster-1", "backend": "testcluster1-
sanbackend"},
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret"
}
```

`ontap-san-economy` 드라이버

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "svm": "svm_iscsi_eco",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret"
}
```

가상 스토리지 풀의 백엔드 예

아래 표시된 백엔드 정의 샘플 파일에서 'paceReserve'는 none, 'pacteAllocation'은 false, 암호화 같은 모든 스토리지 풀에 대해 특정 기본값이 설정됩니다. 가상 스토리지 풀은 스토리지 섹션에 정의됩니다.

이 예에서는 일부 스토리지 풀이 자체 'spaceReserve', 'spaceAllocation' 및 'encryption' 값을 설정하고 일부 풀은 위에 설정된 기본값을 덮어씁니다.

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceAllocation": "false",
          "encryption": "false",
          "qosPolicy": "standard"
    },
    "labels":{"store": "san_store", "kubernetes-cluster": "prod-cluster-
1"},
```

```
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"protection":"gold", "creditpoints":"40000"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "true",
                "adaptiveQosPolicy": "adaptive-extreme"
            }
        },
        {
            "labels":{"protection":"silver", "creditpoints":"20000"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceAllocation": "false",
                "encryption": "true",
                "qosPolicy": "premium"
            }
        },
        {
            "labels":{"protection":"bronze", "creditpoints":"5000"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "false"
            }
        }
    ]
}
```

다음은 iSCSI의 예로, ONTAP-SAN-이코노미 드라이버를 들 수 있습니다.

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "svm": "svm_iscsi_eco",
    "useCHAP": true,
    "chapInitiatorSecret": "cl9qxIm36DKyawxy",
    "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
    "igroupName": "trident",
    "username": "vsadmin",
```

```
    "password": "secret",

    "defaults": {
          "spaceAllocation": "false",
          "encryption": "false"
    },
    "labels":{"store":"san_economy_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"oracledb", "cost":"30"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "true"
            }
        },
        {
            "labels":{"app":"postgresdb", "cost":"20"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceAllocation": "false",
                "encryption": "true"
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"10"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceAllocation": "true",
                "encryption": "false"
            }
        }
    ]
}
```

백엔드를 **StorageClasses**에 매핑합니다

다음 StorageClass 정의는 위의 가상 스토리지 풀을 참조합니다. parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다. 선택한 가상 풀에 볼륨이 정의되어 있습니다.

- 첫 번째 StorageClass('protection-gold')는 ONTAP-NAS-flexgroup 백엔드의 첫 번째 가상 스토리지 풀과 ONTAP-SAN 백엔드의 첫 번째 가상 스토리지 풀에 매핑됩니다. 골드 레벨 보호 기능을 제공하는 유일한 풀입니다.
- 두 번째 StorageClass('금전 보호')는 ONTAP-NAS-Flexgroup 백엔드의 세 번째 가상 스토리지 풀과 ONTAP-SAN 백엔드의 세 번째 가상 스토리지 풀에 매핑됩니다. 금 이외의 보호 수준을 제공하는 유일한 풀입니다.

- 세 번째 StorageClass('app-mysqldb')는 ONTAP-NAS 백엔드의 네 번째 가상 스토리지 풀과 ONTAP-SAN-이코노미 백엔드의 세 번째 가상 스토리지 풀에 매핑됩니다. mysqldb 유형 앱에 대한 스토리지 풀 구성을 제공하는 유일한 풀입니다.
- 네 번째 StorageClass('protection-silver-creditpoints-20k')는 ONTAP-NAS-flexgroup 백엔드의 세 번째 가상 스토리지 풀과 ONTAP-SAN 백엔드의 두 번째 가상 스토리지 풀에 매핑됩니다. 20000 크레딧 포인트에서 골드 레벨 보호 기능을 제공하는 유일한 풀입니다.
- 다섯 번째 StorageClass('크레딧점-5k')는 ONTAP-NAS-이코노미 백엔드의 두 번째 가상 스토리지 풀과 ONTAP-SAN 백엔드의 세 번째 가상 스토리지 풀에 매핑됩니다. 5000 크레딧 포인트에 있는 유일한 풀 서비스입니다.

Astra Trident가 선택한 가상 스토리지 풀을 결정하고 스토리지 요구 사항을 충족해 줍니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

ONTAP NAS 드라이버를 사용하여 백엔드를 구성합니다

ONTAP 및 Cloud Volumes ONTAP NAS 드라이버를 사용하여 ONTAP 백엔드를 구성하는 방법에 대해 알아보십시오.

- "준비"
- "구성 및 예"

사용자 권한

Astra Trident는 일반적으로 "admin" 클러스터 사용자 또는 "vsadmin" SVM 사용자를 사용하거나 동일한 역할을 가진 다른 이름을 가진 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. NetApp ONTAP 구축을 위한 Amazon FSx의 경우 Astra Trident는 클러스터 'fsxadmin' 사용자 또는 'vsadmin' SVM 사용자 또는 동일한 역할을 가진 다른 이름의 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. 'fsxadmin' 사용자는 클러스터 관리자 사용자에게 제한된 교체품입니다.

> 'limitAggregateUsage' 매개 변수를 사용하는 경우 클러스터 관리자 권한이 필요합니다. Astra Trident와 함께 NetApp ONTAP용 Amazon FSx를 사용하는 경우 "limitAggregateUsage" 매개 변수는 "vsadmin" 및 "fsxadmin" 사용자 계정과 작동하지 않습니다. 이 매개 변수를 지정하면 구성 작업이 실패합니다.

ONTAP 내에서 Trident 드라이버가 사용할 수 있는 보다 제한적인 역할을 만들 수 있지만 권장하지 않습니다. Trident의 대부분의 새로운 릴리즈에서는 추가 API를 호출하므로 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

준비

ONTAP NAS 드라이버를 사용하여 ONTAP 백엔드를 구성하는 방법에 대해 알아보십시오. 모든 ONTAP 백엔드의 경우, Astra Trident는 SVM에 하나 이상의 Aggregate가 할당되어 있어야 합니다.

모든 ONTAP 백엔드의 경우, Astra Trident는 SVM에 하나 이상의 Aggregate가 할당되어 있어야 합니다.

또한 둘 이상의 드라이버를 실행하고 둘 중 하나를 가리키는 스토리지 클래스를 생성할 수도 있습니다. 예를 들어, 'ONTAP-NAS' 드라이버와 'ONTAP-NAS-이코노미'을 사용하는 Bronze 클래스를 사용하는 Gold 클래스를 구성할 수 있습니다.

모든 Kubernetes 작업자 노드에 적절한 NFS 툴이 설치되어 있어야 합니다. 을 참조하십시오 "여기" 를 참조하십시오.

인증

Astra Trident는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 필요한 권한이 있는 ONTAP 사용자의 사용자 이름 및 암호입니다. ONTAP 버전과의 호환성을 최대한 보장하기 위해 admin 또는 vsadmin과 같은 미리 정의된 보안 로그인 역할을 사용하는 것이 좋습니다.
- 인증서 기반: Astra Trident는 백엔드에 설치된 인증서를 사용하여 ONTAP 클러스터와 통신할 수도 있습니다. 이 경우 백엔드 정의에는 클라이언트 인증서, 키 및 사용할 경우 신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값이 있어야 합니다(권장).

사용자는 기존 백엔드를 업데이트할 수도 있으며 자격 증명 기반 에서 인증서 기반 으로, 그 반대로 전환할 수도 있습니다. 자격 증명과 인증서가 둘 다 제공된 경우 *, Astra Trident는 기본적으로 인증서를 사용하는 동시에 백엔드 정의에서 자격 증명을 제거하는 경고를 표시합니다.

자격 증명 기반 인증을 사용합니다

Astra Trident는 SVM 범위/클러스터 범위 관리자에게 ONTAP 백엔드와 통신하기 위한 자격 증명을 요구합니다. admin 또는 vsadmin과 같이 미리 정의된 표준 역할을 사용하는 것이 좋습니다. 이를 통해 향후 Astra Trident 릴리스에서 사용할 기능 API를 노출할 수 있는 향후 ONTAP 릴리스와 향후 호환성이 보장됩니다. 사용자 지정 보안 로그인 역할은 Astra Trident와 함께 생성 및 사용할 수 있지만 권장되지 않습니다.

백엔드 정의의 예는 다음과 같습니다.

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret"
}
```

백엔드 정의는 자격 증명이 일반 텍스트로 저장되는 유일한 위치라는 점에 유의하십시오. 백엔드가 생성된 후 사용자 이름/암호는 Base64로 인코딩되어 Kubernetes 암호로 저장됩니다. 백엔드의 생성/업딩은 자격 증명에 대한 지식이 필요한 유일한 단계입니다. 따라서 Kubernetes/스토리지 관리자가 수행할 수 있는 관리 전용 작업입니다.

인증서 기반 인증을 사용합니다

신규 및 기존 백엔드는 인증서를 사용하여 ONTAP 백엔드와 통신할 수 있습니다. 백엔드 정의에는 세 가지 매개 변수가 필요합니다.

- clientCertificate: Base64로 인코딩된 클라이언트 인증서 값입니다.
- clientPrivateKey: Base64 - 연결된 개인 키의 인코딩된 값입니다.
- TrustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개 변수를 제공해야 합니다. 신뢰할 수 있는 CA가 사용되지 않으면 이 작업을 무시할 수 있습니다.

일반적인 워크플로에는 다음 단계가 포함됩니다.

단계

1. 클라이언트 인증서 및 키를 생성합니다. 생성 시 CN(일반 이름)을 ONTAP 사용자로 설정하여 인증하십시오.

   ```
   openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
   -out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
   ```

2. 신뢰할 수 있는 CA 인증서를 ONTAP 클러스터에 추가합니다. 이는 스토리지 관리자가 이미 처리한 것일 수 있습니다. 트러스트된 CA가 사용되지 않으면 무시합니다.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. ONTAP 클러스터에 클라이언트 인증서 및 키(1단계)를 설치합니다.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. ONTAP 보안 로그인 역할이 인증서 인증 방법을 지원하는지 확인합니다.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. 생성된 인증서를 사용하여 인증을 테스트합니다. ONTAP 관리 LIF> 및 <SVM 이름>을 관리 LIF IP 및 SVM 이름으로 바꿉니다. LIF의 서비스 정책이 'default-data-management'로 설정되어 있는지 확인해야 합니다.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Base64로 인증서, 키 및 신뢰할 수 있는 CA 인증서를 인코딩합니다.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 이전 단계에서 얻은 값을 사용하여 백엔드를 생성합니다.

```
$ cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaallllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
$ tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

인증 방법을 업데이트하거나 자격 증명을 회전합니다

기존 백엔드를 업데이트하여 다른 인증 방법을 사용하거나 해당 자격 증명을 회전할 수 있습니다. 이렇게 하면 사용자 이름/암호를 사용하는 백엔드를 인증서를 사용하도록 업데이트할 수 있고 인증서를 사용하는 백엔드는 사용자 이름/암호 기반으로 업데이트할 수 있습니다. 이를 위해서는 tridentctl backend update를 실행하는 데 필요한 parameter가 포함된 update 된 backend.json 파일을 사용한다.

```
$ cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "secret",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
$ tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                  UUID                |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

> 암호를 회전할 때 스토리지 관리자는 먼저 ONTAP에서 사용자의 암호를 업데이트해야 합니다. 그 다음에는 백엔드 업데이트가 있습니다. 인증서를 회전할 때 여러 인증서를 사용자에게 추가할 수 있습니다. 그런 다음 백엔드가 업데이트되어 새 인증서를 사용합니다. 그러면 ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스가 중단되거나 이후에 생성된 볼륨 연결에 영향을 미치지 않습니다. 백엔드 업데이트가 성공적이면 Astra Trident가 ONTAP 백엔드와 통신하고 향후 볼륨 작업을 처리할 수 있음을 나타냅니다.

NFS 엑스포트 정책을 관리합니다

Astra Trident는 NFS 엑스포트 정책을 사용하여 프로비저닝하는 볼륨에 대한 액세스를 제어합니다.

Astra Trident는 엑스포트 정책을 사용할 때 다음 두 가지 옵션을 제공합니다.

- Astra Trident는 엑스포트 정책 자체를 동적으로 관리할 수 있습니다. 이 운영 모드에서 스토리지 관리자는 허용할 수 있는 IP 주소를 나타내는 CIDR 블록 목록을 지정합니다. Astra Trident는 이러한 범위에 속하는 노드 IP를 엑스포트 정책에 자동으로 추가합니다. 또는 CIDR을 지정하지 않으면 노드에서 발견된 글로벌 범위의 유니캐스트 IP가 내보내기 정책에 추가됩니다.

- 스토리지 관리자는 엑스포트 정책을 생성하고 규칙을 수동으로 추가할 수 있습니다. Astra Trident는 구성에 다른 엑스포트 정책 이름을 지정하지 않는 한 기본 엑스포트 정책을 사용합니다.

엑스포트 정책을 동적으로 관리

CSI Trident의 20.04 릴리스는 ONTAP 백엔드에 대한 엑스포트 정책을 동적으로 관리할 수 있는 기능을 제공합니다. 따라서 스토리지 관리자는 명시적 규칙을 수동으로 정의하는 대신 작업자 노드 IP에 허용되는 주소 공간을 지정할 수 있습니다. 엑스포트 정책 관리를 크게 간소화하므로, 엑스포트 정책을 수정하면 더 이상 스토리지 클러스터에 대한 수동 작업이 필요하지 않습니다. 또한 스토리지 클러스터에 대한 액세스를 지정된 범위의 IP가 있는 작업자 노드에만 제한함으로써 세분화된 자동 관리를 지원합니다.

> CSI Trident에만 내보내기 정책의 동적 관리를 사용할 수 있습니다. 작업자 노드가 NATED가 아닌지 확인하는 것이 중요합니다.

예

두 가지 구성 옵션을 사용해야 합니다. 다음은 백엔드 정의의 예입니다.

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap_nas_auto_export,
    "managementLIF": "192.168.0.135",
    "svm": "svm1",
    "username": "vsadmin",
    "password": "FaKePaSsWoRd",
    "autoExportCIDRs": ["192.168.0.0/24"],
    "autoExportPolicy": true
}
```

> 이 기능을 사용할 때는 SVM의 루트 교차점에 노드 CIDR 블록(예: 기본 엑스포트 정책)을 허용하는 엑스포트 규칙과 함께 사전 화된 엑스포트 정책이 있는지 확인해야 합니다. Astra Trident를 위한 SVM 전용 NetApp의 권장 모범 사례를 항상 따르십시오.

다음은 위의 예를 사용하여 이 기능이 작동하는 방식에 대한 설명입니다.

- 자동내보내기정책은 참으로 설정된다. 이는 Astra Trident가 'vm1' SVM에 대한 수출 정책을 만들고 'autoExportCIDR' 주소 블록을 사용하여 규칙 추가 및 삭제를 처리한다는 것을 의미합니다. 예를 들어 UUID 403b5326-8482-40dB-96d0-d83fb3f4daec 및 "true"로 설정된 autoExportPolicy가 있는 백엔드는 SVM에 trident-403b5326-8482-40db-96d0-d83fb3f4daec라는 이름의 엑스포트 정책을 생성합니다.
- autoExportCIDR에는 주소 블록 목록이 포함되어 있습니다. 이 필드는 선택 사항이며 기본적으로 ["0.0.0.0/0", ":/0"]입니다. 정의되지 않은 경우 Astra Trident는 작업자 노드에 있는 모든 전역 범위의 유니캐스트 주소를 추가합니다.

이 예에서는 192.168.0.0/24 주소 공간을 제공한다. 이 주소 범위에 속하는 Kubernetes 노드 IP가 Astra Trident가 생성하는 엑스포트 정책에 추가됨을 나타냅니다. Astra Trident가 실행 중인 노드를 등록하면 해당 노드의 IP 주소를 조회하여 autoExportCIDR에서 제공하는 주소 블록과 대조합니다. IP를 필터링한 후 Astra Trident는 검색된 클라이언트 IP에 대한 엑스포트 정책 규칙을 생성하며, 식별하는 각 노드에 대해 하나의 규칙을 사용합니다.

백엔드를 생성한 후 백엔드에 대한 자동 내보내기 정책 및 자동 내보내기 CIDR을 업데이트할 수 있습니다. 기존 CIDR을 자동으로 관리하거나 삭제하는 백엔드에 새 CIDR을 추가할 수 있습니다. CIDR을 삭제할 때는 기존 연결이 끊어지지 않도록 주의해야 합니다. 백엔드에 대해 'autoExportPolicy'를 사용하지 않도록 설정하고 수동으로 생성된 내보내기 정책으로 돌아갈 수도 있습니다. 이렇게 하려면 백엔드 구성에서 'exportPolicy' 매개 변수를 설정해야 합니다.

Astra Trident가 백엔드를 생성하거나 업데이트한 후 'tridentctl' 또는 해당 'tridentbackend' CRD:

```
$ ./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

노드가 Kubernetes 클러스터에 추가되고 Astra Trident 컨트롤러에 등록되면 기존 백엔드의 내보내기 정책이 업데이트됩니다(백엔드의 "autoExportCIDR"에 지정된 주소 범위에 속하는 경우).

노드가 제거되면 Astra Trident는 온라인 상태인 모든 백엔드를 검사하여 노드에 대한 액세스 규칙을 제거합니다. Astra Trident는 관리되는 백엔드의 내보내기 정책에서 이 노드 IP를 제거하여 불량 마운트를 방지합니다. 단, 클러스터의 새 노드에서 이 IP를 다시 사용하지 않는 한 마찬가지입니다.

기존 백엔드의 경우 백엔드를 "tridentctl update backend"로 업데이트하면 Astra Trident가 자동으로 내보내기 정책을 관리합니다. 그러면 백엔드의 UUID 뒤에 이름이 지정된 새 내보내기 정책이 생성되고 백엔드에 있는 볼륨은 새로 생성된 내보내기 정책을 다시 마운트할 때 사용합니다.

> 자동 관리되는 내보내기 정책이 있는 백엔드를 삭제하면 동적으로 생성된 내보내기 정책이 삭제됩니다. 백엔드가 다시 생성되면 백엔드가 새 백엔드로 처리되어 새 엑스포트 정책이 생성됩니다.

라이브 노드의 IP 주소가 업데이트되면 노드에서 Astra Trident POD를 다시 시작해야 합니다. 그런 다음 Astra Trident가 이 IP 변경 사항을 반영하도록 관리하는 백엔드에 대한 엑스포트 정책을 업데이트합니다.

구성 옵션 및 예

Astra Trident 설치를 통해 ONTAP NAS 드라이버를 생성하고 사용하는 방법에 대해 알아보십시오. 이 섹션에서는

백엔드 구성 예제 및 백엔드를 StorageClasses에 매핑하는 방법에 대한 세부 정보를 제공합니다.

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개 변수	설명	기본값
'내전'		항상 1
'torageDriverName'입니다	스토리지 드라이버의 이름입니다	"ONTAP-NAS", "ONTAP-NAS-이코노미", "ONTAP-NAS-Flexgroup", "ONTAP-SAN", "ONTAP-SAN-이코노미"
백엔드이름	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
마나멘타LIF	클러스터 또는 SVM 관리 LIF의 IP 주소입니다	"10.0.0.1", "[2001:1234:ABCD::fee]"
다타LIF	프로토콜 LIF의 IP 주소입니다. IPv6의 경우 대괄호를 사용합니다. 설정한 후에는 업데이트할 수 없습니다	지정되지 않은 경우 SVM에서 파생됩니다
자동 내보내기 정책	자동 엑스포트 정책 생성 및 업데이트 활성화 [부울]	거짓
자동 내보내기	"autoExportPolicy"가 활성화된 경우 Kubernetes 노드 IP를 필터링하기 위한 CIDR 목록입니다	["0.0.0.0/0", "::/0"]"
'라벨'	볼륨에 적용할 임의의 JSON 형식 레이블 세트입니다	""
'고객증명서'	Base64 - 클라이언트 인증서의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다	""
'clientPrivateKey'입니다	Base64 - 클라이언트 개인 키의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다	""
신탁인증서다	Base64 - 신뢰할 수 있는 CA 인증서의 인코딩된 값입니다. 선택 사항. 인증서 기반 인증에 사용됩니다	""
'사용자 이름'	클러스터/SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다	
"암호"	클러스터/SVM에 연결하는 암호 자격 증명 기반 인증에 사용됩니다	
'VM'입니다	사용할 스토리지 가상 머신입니다	SVM 'managementLIF'가 지정된 경우에 파생됩니다
"인명이름"입니다	사용할 SAN 볼륨에 대한 igroup의 이름입니다	"삼중 - <backend-UUID>"

매개 변수	설명	기본값
'토르agePrefix'	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사 설정한 후에는 업데이트할 수 없습니다	"삼중류"
제한선태사용법	사용량이 이 비율을 초과하면 프로비저닝이 실패합니다. ONTAP * 용 아마존 FSx에는 * 가 적용되지 않습니다	“”(기본적으로 적용되지 않음)
LimitVolumeSize	요청된 볼륨 크기가 이 값보다 큰 경우 용량 할당에 실패합니다.	“”(기본적으로 적용되지 않음)
'오만유연한'	FlexVol당 최대 LUN 수는 범위[50, 200]에 있어야 합니다.	“100”
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: {“api”:false, “method”:true}	null입니다
nfsMountOptions를 선택합니다	쉼표로 구분된 NFS 마운트 옵션 목록입니다	“”
"케트리스퍼플랙스볼륨"	FlexVol당 최대 qtree, 범위 [50, 300]에 있어야 함	“200”
'useREST'	ONTAP REST API를 사용하는 부울 매개 변수입니다. * 기술 미리 보기 *	거짓

> "useREST"는 프로덕션 워크로드가 아닌 테스트 환경에 권장되는** 기술 미리 보기로 제공됩니다. "true"로 설정하면 Astra Trident는 ONTAP REST API를 사용하여 백엔드와 통신합니다. 이 기능을 사용하려면 ONTAP 9.9 이상이 필요합니다. 또한 사용되는 ONTAP 로그인 역할은 ONTAP 애플리케이션에 대한 액세스 권한이 있어야 합니다. 이는 미리 정의된 vsadmin과 cluster-admin의 역할에서 충족됩니다.

ONTAP 클러스터와 통신하려면 인증 매개 변수를 제공해야 합니다. 보안 로그인 또는 설치된 인증서의 사용자 이름/암호일 수 있습니다.

> NetApp ONTAP 백엔드에 Amazon FSx를 사용하는 경우 'limitAggregateUsage' 매개 변수를 지정하지 마십시오. NetApp ONTAP용 Amazon FSx에서 제공하는 "fsxadmin" 및 "vsadmin" 역할에는 애그리게이트 사용을 검색하고 Astra Trident를 통해 제한하는 데 필요한 액세스 권한이 없습니다.

> 문제 해결 및 자세한 로그 덤프가 필요한 경우가 아니면 debugTraceFlags를 사용하지 마십시오.

> 백엔드를 생성할 때 생성 후에는 "다LIF"와 "toragePrefix"를 수정할 수 없습니다. 이러한 매개 변수를 업데이트하려면 새 백엔드를 생성해야 합니다.

'managementLIF' 옵션에 대해 FQDN(정규화된 도메인 이름)을 지정할 수 있습니다. 또한 dataLIF 옵션에 FQDN을 지정할 수 있으며, 이 경우 NFS 마운트 작업에 FQDN이 사용됩니다. 이렇게 하면 라운드 로빈 DNS를 생성하여 여러 데이터 LIF 간에 로드 밸런싱을 수행할 수 있습니다.

모든 ONTAP 드라이버에 대한 manementLIF도 IPv6 주소로 설정할 수 있습니다. '--use-ipv6' 플래그를 사용하여

Astra Trident를 설치하십시오. 대괄호 안에 있는 'managementLIF' IPv6 주소를 정의할 때는 주의해야 합니다.

> IPv6 주소를 사용할 때는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같은 대괄호 안에 'managementLIF' 및 'dataLIF'(백엔드 정의에 포함된 경우)가 정의되어 있는지 확인하십시오. 다타 LIF가 제공되지 않으면 Astra Trident가 SVM에서 IPv6 데이터 LIF를 가져옵니다.

CSI Trident는 autoExportPolicy와 autoExportCIDR 옵션을 사용하여 내보내기 정책을 자동으로 관리할 수 있습니다. 이 기능은 모든 ONTAP-NAS- * 드라이버에 대해 지원됩니다.

ONTAP-NAS-이코노미 드라이버의 경우 LimitVolumeSize 옵션도 qtree 및 LUN에 대해 관리하는 볼륨의 최대 크기를 제한하고, qtreesPerFlexvol 옵션을 사용하면 FlexVol당 최대 qtree 수를 사용자 지정할 수 있습니다.

nfsMountOptions 매개 변수를 사용하여 마운트 옵션을 지정할 수 있습니다. Kubernetes 영구 볼륨의 마운트 옵션은 일반적으로 스토리지 클래스에서 지정되지만 스토리지 클래스에 마운트 옵션이 지정되지 않은 경우 Astra Trident는 스토리지 백엔드의 구성 파일에 지정된 마운트 옵션을 사용하여 로 돌아갑니다. 스토리지 클래스 또는 구성 파일에 마운트 옵션을 지정하지 않으면 Astra Trident가 연결된 영구 볼륨에 마운트 옵션을 설정하지 않습니다.

> Astra Trident는 ONTAP-NAS와 ONTAP-NAS-Flexgroup을 사용하여 생성된 모든 볼륨의 “Comments” 필드에 제공 레이블을 설정합니다. 사용된 드라이버에 따라 FlexVol('ONTAP-NAS') 또는 FlexGroup('ONTAP-NAS-Flexgroup')에 주석이 설정됩니다. Astra Trident는 스토리지 풀에 있는 모든 레이블을 프로비저닝할 때 스토리지 볼륨에 복사합니다. 스토리지 관리자는 스토리지 풀별로 레이블을 정의하고 스토리지 풀에서 생성된 모든 볼륨을 그룹화할 수 있습니다. 이를 통해 백엔드 구성에서 제공되는 사용자 지정 가능한 레이블 세트를 기반으로 볼륨을 쉽게 구별할 수 있습니다.

볼륨 프로비저닝을 위한 백엔드 구성 옵션

구성의 특수 섹션에서 이러한 옵션을 사용하여 각 볼륨이 기본적으로 프로비저닝되는 방식을 제어할 수 있습니다. 예를 들어, 아래 구성 예제를 참조하십시오.

매개 변수	설명	기본값
'팩시배부	LUN에 대한 공간 할당	"참"
'예비공간'	공간 예약 모드, "없음"(씬) 또는 " 볼륨"(일반)	"없음"
냅샷정책	사용할 스냅샷 정책입니다	"없음"
"qosPolicy"	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드에서 qosPolicy 또는 adapativeQosPolicy 중 하나를 선택합니다	“”
적응성 QosPolicy	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀 /백엔드에서 qosPolicy 또는 adapativeQosPolicy 중 하나를 선택합니다. ONTAP에서 지원되지 않음 - NAS - 이코노미	“”
안산예비역	스냅샷 "0"에 예약된 볼륨의 백분율	"스냅샷 정책"이 "없음"이면 "없음"
'plitOnClone’을 선택합니다	생성 시 상위 클론에서 클론을 분할합니다	"거짓"
암호화	NetApp 볼륨 암호화를 활성화합니다	"거짓"

매개 변수	설명	기본값
'생태성 스타일'을 참조하십시오	새로운 볼륨에 대한 보안 스타일	“UNIX”
'계층화 정책'	"없음"을 사용하는 계층화 정책	ONTAP 9.5 이전 SVM-DR 구성의 경우 "스냅샷 전용
unixPermissions	모드를 선택합니다	“777”
스냅샷 디렉토리	'.snapshot' 디렉토리의 가시성을 제어합니다	"거짓"
내보내기 정책	사용할 엑스포트 정책	“기본값”
보안 스타일	새로운 볼륨에 대한 보안 스타일	“UNIX”

> Astra Trident와 함께 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 비공유 QoS 정책 그룹을 사용하고 정책 그룹이 각 구성요소별로 적용되었는지 확인하는 것이 좋습니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 대해 상한을 적용합니다.

다음은 기본값이 정의된 예입니다.

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "customBackendName",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "labels": {"k8scluster": "dev1", "backend": "dev1-nasbackend"},
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "password",
  "limitAggregateUsage": "80%",
  "limitVolumeSize": "50Gi",
  "nfsMountOptions": "nfsvers=4",
  "debugTraceFlags": {"api":false, "method":true},
  "defaults": {
    "spaceReserve": "volume",
    "qosPolicy": "premium",
    "exportPolicy": "myk8scluster",
    "snapshotPolicy": "default",
    "snapshotReserve": "10"
  }
}
```

ONTAP-NAS와 ONTAP-NAS-Flexgroups의 경우, Astra Trident는 이제 새로운 계산을 통해 스냅샷 예비 공간 비율 및 PVC로 FlexVol의 크기를 올바르게 조정했습니다. 사용자가 PVC를 요청하면 Astra Trident는 새 계산을 사용하여 더 많은 공간을 가진 원본 FlexVol를 생성합니다. 이 계산을 통해 사용자는 PVC에서 요청한 쓰기 가능 공간을 확보할 수 있으며 요청된 공간보다 적은 공간을 확보할 수 있습니다. v21.07 이전에는 사용자가 스냅샷 보존 공간을 50%로 하여 PVC(예: 5GiB)를 요청할 때 쓰기 가능한 공간은 2.5GiB에 불과합니다. 이는 사용자가 요청한 전체 볼륨과 스냅샷 보존

비율이 다르기 때문입니다. Trident 21.07을 사용하면 쓰기 가능한 공간이 요청되고 Astra Trident는 '스냅샷 보존' 번호를 전체 볼륨의 백분율로 정의합니다. 이는 ONTAP-NAS-이코노미(ONTAP-NAS-이코노미)에는 적용되지 않습니다. 이 작동 방식을 보려면 다음 예를 참조하십시오.

계산은 다음과 같습니다.

```
Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)
```

snapshotReserve = 50%, PVC request = 5GiB의 경우, 총 볼륨 크기는 2/.5 = 10GiB이고 사용 가능한 크기는 5GiB입니다. 이는 사용자가 PVC 요청에서 요청한 것입니다. 'volume show' 명령은 다음 예와 유사한 결과를 표시합니다.

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
                 _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                                      online     RW         10GB     5.00GB    0%
                 _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                                      online     RW          1GB    511.8MB    0%
2 entries were displayed.
```

이전 설치에서 기존 백엔드는 Astra Trident를 업그레이드할 때 위에서 설명한 대로 볼륨을 프로비저닝합니다. 업그레이드하기 전에 생성한 볼륨의 경우 변경 사항을 관찰하기 위해 볼륨의 크기를 조정해야 합니다. 예를 들어, '스냅샷 보존 = 50'의 2GiB PVC는 쓰기 가능한 공간 1GiB를 제공하는 볼륨을 낳았습니다. 예를 들어, 볼륨을 3GiB로 조정하면 애플리케이션에 6GiB 볼륨의 쓰기 가능 공간이 3GiB로 표시됩니다.

최소 구성의 예

다음 예에서는 대부분의 매개 변수를 기본값으로 두는 기본 구성을 보여 줍니다. 이는 백엔드를 정의하는 가장 쉬운 방법입니다.

> Trident가 있는 NetApp ONTAP에서 Amazon FSx를 사용하는 경우 IP 주소 대신 LIF에 대한 DNS 이름을 지정하는 것이 좋습니다.

`ontap-nas` 인증서 기반 인증을 사용하는 드라이버

이는 최소 백엔드 구성의 예입니다. clientCertificate, clientPrivateKey, trustedCACertificate(신뢰할 수 있는 CA를 사용하는 경우 선택 사항)가 backend.json에 채워지고 클라이언트 인증서, 개인 키, 트러스트된 CA 인증서의 base64로 인코딩된 값을 각각 가져갑니다.

```
{
  "version": 1,
  "backendName": "DefaultNASBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.15",
  "svm": "nfs_svm",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz",
  "storagePrefix": "myPrefix_"
}
```

`ontap-nas` 자동 내보내기 정책이 있는 드라이버

이 예에서는 Astra Trident가 동적 엑스포트 정책을 사용하여 엑스포트 정책을 자동으로 생성하고 관리하도록 지시하는 방법을 보여 줍니다. 이는 ONTAP-NAS-이코노미 및 ONTAP-NAS-Flexgroup 드라이버에도 동일하게 적용됩니다.

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "labels": {"k8scluster": "test-cluster-east-1a", "backend": "test1-
nasbackend"},
    "autoExportPolicy": true,
    "autoExportCIDRs": ["10.0.0.0/24"],
    "username": "admin",
    "password": "secret",
    "nfsMountOptions": "nfsvers=4",
}
```

`ontap-nas-flexgroup` 드라이버

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "labels": {"k8scluster": "test-cluster-east-1b", "backend": "test1-
ontap-cluster"},
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",
}
```

`ontap-nas` **IPv6**를 사용하는 드라이버

```
{
 "version": 1,
 "storageDriverName": "ontap-nas",
 "backendName": "nas_ipv6_backend",
 "managementLIF": "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]",
 "labels": {"k8scluster": "test-cluster-east-1a", "backend": "test1-ontap-
ipv6"},
 "svm": "nas_ipv6_svm",
 "username": "vsadmin",
 "password": "netapp123"
}
```

`ontap-nas-economy` 드라이버

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret"
}
```

가상 스토리지 풀의 백엔드 예

아래 표시된 백엔드 정의 샘플 파일에서 'paceReserve'는 none, 'pacteAllocation'은 false, 암호화 같은 모든 스토리지 풀에 대해 특정 기본값이 설정됩니다. 가상 스토리지 풀은 스토리지 섹션에 정의됩니다.

이 예에서는 일부 스토리지 풀이 자체 'spaceReserve', 'spaceAllocation' 및 'encryption' 값을 설정하고 일부 풀은 위에 설정된 기본값을 덮어씁니다.

`ontap-nas` 드라이버

```
{
    {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "admin",
    "password": "secret",
    "nfsMountOptions": "nfsvers=4",

    "defaults": {
          "spaceReserve": "none",
          "encryption": "false",
          "qosPolicy": "standard"
    },
    "labels":{"store":"nas_store", "k8scluster": "prod-cluster-1"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"msoffice", "cost":"100"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755",
                "adaptiveQosPolicy": "adaptive-premium"
            }
        },
        {
            "labels":{"app":"slack", "cost":"75"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"app":"wordpress", "cost":"50"},
            "zone":"us_east_1c",
```

```
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"25"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

`ontap-nas-flexgroup` 드라이버

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceReserve": "none",
          "encryption": "false"
    },
    "labels":{"store":"flexgroup_store", "k8scluster": "prod-cluster-1"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"protection":"gold", "creditpoints":"50000"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
```

```
        {
            "labels":{"protection":"gold", "creditpoints":"30000"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"protection":"silver", "creditpoints":"20000"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"protection":"bronze", "creditpoints":"10000"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

`ontap-nas-economy` 드라이버

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "secret",

    "defaults": {
          "spaceReserve": "none",
          "encryption": "false"
    },
```

```
    "labels":{"store":"nas_economy_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"department":"finance", "creditpoints":"6000"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"department":"legal", "creditpoints":"5000"},
            "zone":"us_east_1b",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"department":"engineering", "creditpoints":"3000"},
            "zone":"us_east_1c",
            "defaults": {
                "spaceReserve": "none",
                "encryption": "true",
                "unixPermissions": "0775"
            }
        },
        {
            "labels":{"department":"humanresource",
"creditpoints":"2000"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

백엔드를 **StorageClasses**에 매핑합니다

다음 StorageClass 정의는 위의 가상 스토리지 풀을 참조합니다. parameters.selector` 필드를 사용하여 각

StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다. 선택한 가상 풀에 볼륨이 정의되어 있습니다.

- 첫 번째 StorageClass('protection-gold')는 ONTAP-NAS-flexgroup 백엔드의 첫 번째 가상 스토리지 풀과 ONTAP-SAN 백엔드의 첫 번째 가상 스토리지 풀에 매핑됩니다. 골드 레벨 보호 기능을 제공하는 유일한 풀입니다.
- 두 번째 StorageClass('금전 보호')는 ONTAP-NAS-Flexgroup 백엔드의 세 번째 가상 스토리지 풀과 ONTAP-SAN 백엔드의 세 번째 가상 스토리지 풀에 매핑됩니다. 금 이외의 보호 수준을 제공하는 유일한 풀입니다.
- 세 번째 StorageClass('app-mysqldb')는 ONTAP-NAS 백엔드의 네 번째 가상 스토리지 풀과 ONTAP-SAN-이코노미 백엔드의 세 번째 가상 스토리지 풀에 매핑됩니다. mysqldb 유형 앱에 대한 스토리지 풀 구성을 제공하는 유일한 풀입니다.
- 네 번째 StorageClass('protection-silver-creditpoints-20k')는 ONTAP-NAS-flexgroup 백엔드의 세 번째 가상 스토리지 풀과 ONTAP-SAN 백엔드의 두 번째 가상 스토리지 풀에 매핑됩니다. 20000 크레딧 포인트에서 골드 레벨 보호 기능을 제공하는 유일한 풀입니다.
- 다섯 번째 StorageClass('크레딧점-5k')는 ONTAP-NAS-이코노미 백엔드의 두 번째 가상 스토리지 풀과 ONTAP-SAN 백엔드의 세 번째 가상 스토리지 풀에 매핑됩니다. 5000 크레딧 포인트에 있는 유일한 풀 서비스입니다.

Astra Trident가 선택한 가상 스토리지 풀을 결정하고 스토리지 요구 사항을 충족해 줍니다.

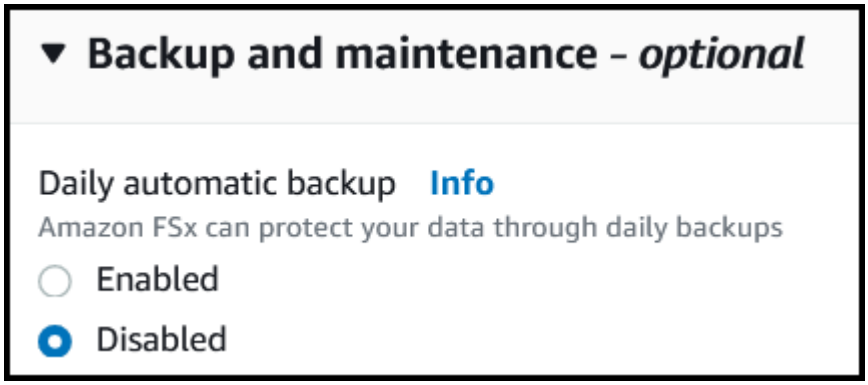
```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

NetApp ONTAP용 Amazon FSx와 함께 Astra Trident를 사용하십시오

"NetApp ONTAP용 Amazon FSx"는 NetApp의 ONTAP 스토리지 운영 체제가 제공하는 파일 시스템을 실행하고 실행할 수 있도록 완벽하게 관리되는 AWS 서비스입니다. NetApp ONTAP용 Amazon FSx를 사용하면 익숙한 NetApp 기능, 성능 및 관리 기능을 활용하는 동시에, AWS에 데이터를 저장하는 간편성, 민첩성, 보안, 확장성을 활용할 수 있습니다. FSX는 ONTAP의 다양한 파일 시스템 기능과 관리 API를 지원합니다.

파일 시스템은 Amazon FSx의 주요 리소스이며, 이는 사내 ONTAP 클러스터와 유사합니다. 각 SVM 내에서 파일 시스템에 파일과 폴더를 저장하는 데이터 컨테이너인 하나 이상의 볼륨을 생성할 수 있습니다. NetApp ONTAP용 Amazon FSx를 사용하면 클라우드에서 Data ONTAP가 관리형 파일 시스템으로 제공됩니다. 새로운 파일 시스템 유형을 * NetApp ONTAP * 라고 합니다.

NetApp ONTAP용 Amazon FSx와 Astra Trident를 사용하면 Amazon EKS(Elastic Kubernetes Service)에서 실행되는 Kubernetes 클러스터가 ONTAP에서 지원하는 블록 및 파일 영구 볼륨을 프로비저닝할 수 있습니다.

> **ONTAP 파일 시스템용 Amazon FSx 생성**
>
> 자동 백업이 설정된 Amazon FSx 파일 시스템에 생성된 볼륨은 Trident에서 삭제할 수 없습니다. PVC를 삭제하려면 ONTAP 체적에 대한 PV 및 FSx를 수동으로 삭제해야 합니다.
>
> 이 문제를 방지하려면:
>
> - **Quick create**를 사용하여 ONTAP 파일 시스템용 FSx를 생성하지 마십시오. 빠른 생성 워크플로에서는 자동 백업을 사용할 수 있으며 수신 거부 옵션은 제공하지 않습니다.
> - 표준 생성 을 사용하는 경우 자동 백업을 비활성화합니다. 자동 백업을 비활성화하면 Trident는 추가적인 수동 개입 없이 볼륨을 성공적으로 삭제할 수 있습니다.
>
> 
>

Astra Trident에 대해 자세히 알아보십시오

Astra Trident를 처음 사용하는 경우 아래에 제공된 링크를 사용하여 숙지하십시오.

- "FAQ 를 참조하십시오"
- "Astra Trident를 사용하기 위한 요구사항"
- "Astra Trident 구축"
- "NetApp ONTAP용 ONTAP, Cloud Volumes ONTAP 및 Amazon FSx를 구성하는 모범 사례"
- "Astra Trident 통합"
- "ONTAP SAN 백엔드 구성"

- "ONTAP NAS 백엔드 구성"

드라이버 기능에 대해 자세히 알아보십시오 "여기".

NetApp ONTAP용 Amazon FSx에서 사용됩니다 "FabricPool" 스토리지 계층을 관리합니다. 이 기능을 사용하면 데이터의 액세스 빈도 여부에 따라 데이터를 계층에 저장할 수 있습니다.

Astra Trident는 "vsadmin" SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자로 실행될 것으로 예상합니다. NetApp ONTAP용 Amazon FSx에는 ONTAP 'admin' 클러스터 사용자를 제한적으로 대체하는 'fsxadmin' 사용자가 있습니다. vsadmin SVM 사용자는 더 많은 Astra Trident 기능을 사용할 수 있으므로 Trident와 함께 fsxadmin 사용자를 사용하지 않는 것이 좋습니다.

드라이버

다음 드라이버를 사용하여 Astra Trident를 NetApp ONTAP용 Amazon FSx와 통합할 수 있습니다.

- 'ONTAP-SAN': 프로비저닝되는 각 PV는 NetApp ONTAP 볼륨용 자체 Amazon FSx 내에 있는 LUN입니다.
- 'ONTAP-SAN-이코노미': 프로비저닝되는 각 PV는 NetApp ONTAP 볼륨에 대해 Amazon FSx당 구성 가능한 LUN 수를 가진 LUN입니다.
- ONTAP-NAS: 프로비저닝되는 각 PV는 NetApp ONTAP 볼륨용 전체 Amazon FSx입니다.
- 'ONTAP-NAS-E경제적인': 각 PV 프로비저닝은 qtree이며, NetApp ONTAP 볼륨에 대해 Amazon FSx당 qtree를 구성할 수 있습니다.
- 'ONTAP-NAS-flexgroup': 프로비저닝되는 각 PV는 NetApp ONTAP FlexGroup 볼륨에 대한 전체 Amazon FSx입니다.

인증

Astra Trident는 두 가지 인증 모드를 제공합니다.

- 인증서 기반: Astra Trident는 SVM에 설치된 인증서를 사용하여 FSx 파일 시스템의 SVM과 통신합니다.
- 자격 증명 기반: 파일 시스템에 대해 'fsxadmin' 사용자를 사용하거나 SVM용으로 구성된 'vsadmin' 사용자를 사용할 수 있습니다.

> 백엔드를 구성하려면 'fsxadmin' 대신 'vsadmin' 사용자를 사용하는 것이 좋습니다. Astra Trident는 이 사용자 이름과 암호를 사용하여 FSx 파일 시스템과 통신합니다.

인증에 대한 자세한 내용은 다음 링크를 참조하십시오.

- "ONTAP NAS를 지원합니다"
- "ONTAP SAN"

NetApp ONTAP용 Amazon FSx를 사용하여 **EKS**에서 **Astra Trident**를 구축하고 구성합니다

필요한 것

- kubbtl이 설치된 기존 Amazon EKS 클러스터 또는 자체 관리 Kubernetes 클러스터
- 클러스터의 작업자 노드에서 연결할 수 있는 NetApp ONTAP 파일 시스템용 기존 Amazon FSx 및 SVM(스토리지 가상 시스템)입니다.

- 에 대해 준비된 작업자 노드입니다 "NFS 및/또는 iSCSI".

> Amazon Linux 및 Ubuntu에 필요한 노드 준비 단계를 따라야 합니다 "Amazon Machine Images(아마존 머신 이미지)" (AMI) EKS AMI 유형에 따라 다릅니다.

다른 Astra Trident 요구 사항은 를 참조하십시오 "여기".

단계

1. ../triment-get-started/Kubernetes-deploy.html[배포 방법^] 중 하나를 사용하여 Astra Trident를 구축하십시오.
2. 다음과 같이 Astra Trident를 구성합니다.
   a. SVM의 관리 LIF DNS 이름을 수집합니다. 예를 들어, AWS CLI를 사용하여 다음 명령을 실행한 후 "Endpoints" → "anagement"에서 DNSName 항목을 찾습니다.

```
aws fsx describe-storage-virtual-machines --region <file system
region>
```

3. 인증을 위한 인증서를 만들고 설치합니다. 'ONTAP-SAN' 백엔드를 사용하는 경우 을 참조하십시오 "여기". 'ONTAP-NAS' 백엔드를 사용하는 경우 를 참조하십시오 "여기".

> 파일 시스템에 연결할 수 있는 모든 위치에서 SSH를 사용하여 파일 시스템(예: 인증서 설치)에 로그인할 수 있습니다. 파일 시스템 생성 시 구성한 fsxadmin 사용자, AWS FSX 기술파일 시스템 "의 관리 DNS 이름을 사용합니다.

4. 아래 예에 표시된 대로 인증서와 관리 LIF의 DNS 이름을 사용하여 백엔드 파일을 생성합니다.

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "customBackendName",
  "managementLIF": "svm-XXXXXXXXXXXXXXXXX.fs-XXXXXXXXXXXXXXXXX.fsx.us-
east-2.aws.internal",
  "svm": "svm01",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz",
 }
```

백엔드 만들기에 대한 자세한 내용은 다음 링크를 참조하십시오.

- "ONTAP NAS 드라이버를 사용하여 백엔드를 구성합니다"
- "ONTAP SAN 드라이버를 사용하여 백엔드를 구성합니다"

Astra Trident에서 다중 경로를 사용할 수 있도록 ONTAP-SAN 및 ONTAP-SAN-이코노미 드라이버에 대해 "LIF"를 지정하지 마십시오.

limitAggregateUsage 매개변수는 vsadmin과 fsxadmin 사용자 계정에서는 작동하지 않습니다. 이 매개 변수를 지정하면 구성 작업이 실패합니다.

배포 후 단계를 수행하여 을 생성합니다 "스토리지 클래스, 볼륨 프로비저닝 및 POD에 볼륨 마운트".

자세한 내용을 확인하십시오

- "NetApp ONTAP용 Amazon FSx 문서"
- "NetApp ONTAP용 Amazon FSx 블로그 게시물"

kubectl로 백엔드를 만듭니다

백엔드는 Astra Trident와 스토리지 시스템 간의 관계를 정의합니다. Astra Trident가 스토리지 시스템과 통신하는 방법과 Astra Trident가 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다. Astra Trident를 설치한 후 다음 단계는 백엔드를 생성하는 것입니다. '트리펜엔드구성' 사용자 정의 리소스 정의(CRD)를 사용하면 Kubernetes 인터페이스를 통해 Trident 백엔드를 직접 생성하고 관리할 수 있습니다. 쿠베틀이나 이와 동등한 CLI 툴을 사용하여 Kubernetes를 배포할 수 있습니다.

`TridentBackendConfig`

트리젠벤디Config('tbc', 'tbconfig', 'tbackendconfig')는 쿠벡틀로 아스트라 트리덴트 백엔드를 관리할 수 있는 프론트이자 이름있는 CRD입니다. 이제 Kubernetes 및 스토리지 관리자는 전용 명령줄 유틸리티('tridentctl')를 사용하지 않고 Kubernetes CLI를 통해 직접 백엔드를 생성 및 관리할 수 있습니다.

'트리펜엔드구성' 객체가 생성되면 다음과 같은 현상이 발생합니다.

- 백엔드는 사용자가 제공하는 구성에 따라 Astra Trident에서 자동으로 생성합니다. 이는 내부적으로 트리덴백엔드(트리덴백엔드) CR로 표현된다.
- 트리젠트백엔드구성은 아스트라 트리덴트(Astra Trident)가 만든 트리젠백엔드(트리젠백엔드)에 고유하게 바인딩됩니다.

각각의 트리젠백엔드구성은 트리젠백엔드(트리젠백엔드)를 통해 일대일 매핑을 유지합니다. 전자는 백엔드를 설계 및 구성하기 위해 사용자에게 제공되는 인터페이스이며, 후자는 Trident가 실제 백엔드 객체를 나타내는 방법입니다.

트리젠백엔드 CRS는 아스트라 트리덴트(Astra Trident)에 의해 자동으로 생성됩니다. 수정할 수 없습니다. 백엔드에 대한 업데이트를 하려면 '트리엔백구성' 개체를 수정하여 이 작업을 수행하십시오.

'트리엔백엔드구성' CR의 형식은 다음 예를 참조하십시오.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

의 예를 살펴볼 수도 있습니다 "Trident - 장착 공구" 원하는 스토리지 플랫폼/서비스의 샘플 구성을 위한 디렉토리입니다.

이 'pec'은 백엔드 구성 매개 변수를 사용합니다. 이 예에서는 백엔드에서 'ONTAP-SAN' 스토리지 드라이버를 사용하고 여기에 표로 제공된 구성 매개 변수를 사용합니다. 원하는 스토리지 드라이버에 대한 구성 옵션 목록은 를 참조하십시오 "스토리지 드라이버에 대한 백엔드 구성 정보입니다".

이번 시기에는 트리젠백엔드Config CR에 새로 도입된 자격 증명과 delitionPolicy 필드가 포함되어 있습니다.

- "credentials": 이 매개변수는 필수 필드이며 스토리지 시스템/서비스를 인증하는 데 사용되는 자격 증명을 포함합니다. 사용자 생성 Kubernetes Secret으로 설정됩니다. 자격 증명을 일반 텍스트로 전달할 수 없으며 오류가 발생합니다.
- "설명 정책": 이 필드는 트리젠BackendConfig가 삭제될 때 발생하는 동작을 정의합니다. 다음 두 가지 값 중 하나를 사용할 수 있습니다.
  - 삭제(Delete): 이 경우 트리엔백엔드Config CR과 관련 백엔드가 모두 삭제됩니다. 이 값이 기본값입니다.
  - [Tain]: 트리덴트Config CR이 삭제되면 백엔드 정의가 계속 존재하고 tridentctl로 관리될 수 있다. 삭제 정책을 "보존"으로 설정하면 사용자는 이전 릴리스(21.04 이전)로 다운그레이드하고 생성된 백엔드를 유지할 수 있습니다. 이 필드의 값은 '트리엔백엔드구성'이 생성된 후에 업데이트할 수 있습니다.

> 백엔드 이름은 'pec.backendName'을 사용하여 설정됩니다. 지정하지 않으면 백엔드 이름이 '트리엔백엔드구성' 객체(metadata.name 지정합니다. 'pec.backendName'을 사용하여 백엔드 이름을 명시적으로 설정하는 것이 좋습니다.

> tridentctl로 만든 백엔드에는 관련된 트리젠BackendConfig 객체가 없습니다. 트리젠백엔드Config CR을 만들어 kubctl로 백엔드를 관리할 수 있습니다. 동일한 구성 매개 변수(예: 'pec.backendName', 'pec.storagePrefix', 'pec.storageDriverName' 등)를 지정할 때는 주의해야 합니다. Astra Trident는 새로 생성된 '트리젠백엔드구성'을 기존 백엔드와 자동으로 바인딩합니다.

단계 개요

kubbeck을 사용하여 새 백엔드를 생성하려면 다음을 수행해야 합니다.

1. 을 생성합니다 "쿠버네티스 비밀". 비밀에는 Astra Trident가 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. '트리멘백엔드구성' 객체를 만듭니다. 스토리지 클러스터/서비스에 대한 자세한 내용과 이전 단계에서 생성한 암호를 참조하십시오.

백엔드를 생성한 후 "kubbtl get tbc<tbc-name>-n<trident-namespace>"를 사용하여 해당 상태를 관찰하고 추가 세부 정보를 수집할 수 있습니다.

1단계: Kubernetes Secret 생성

백엔드에 대한 액세스 자격 증명이 포함된 암호를 생성합니다. 이는 각 스토리지 서비스/플랫폼마다 다릅니다. 예를 들면 다음과 같습니다.

```
$ kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: t@Ax@7q(>
```

이 표에는 각 스토리지 플랫폼의 비밀에 포함되어야 하는 필드가 요약되어 있습니다.

스토리지 플랫폼 암호 필드 설명입니다	비밀	필드 설명입니다
Azure NetApp Files	클라이언트 ID입니다	앱 등록에서 클라이언트 ID
GCP용 Cloud Volumes Service	private_key_id	개인 키의 ID입니다. CVS 관리자 역할을 가진 GCP 서비스 계정에 대한 API 키의 일부
GCP용 Cloud Volumes Service	개인 키	개인 키. CVS 관리자 역할을 가진 GCP 서비스 계정에 대한 API 키의 일부
요소(NetApp HCI/SolidFire)	엔드포인트	테넌트 자격 증명이 있는 SolidFire 클러스터의 MVIP입니다
ONTAP	사용자 이름	클러스터/SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다
ONTAP	암호	클러스터/SVM에 연결하는 암호 자격 증명 기반 인증에 사용됩니다

스토리지 플랫폼 암호 필드 설명입니다	비밀	필드 설명입니다
ONTAP	clientPrivateKey를 선택합니다	Base64 - 클라이언트 개인 키의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다
ONTAP	챕터 사용자 이름	인바운드 사용자 이름입니다. useCHAP = TRUE인 경우 필수입니다. ONTAP-SAN과 ONTAP-SAN 경제입니다
ONTAP	챕터시토시크릿	CHAP 이니시에이터 암호입니다. useCHAP = TRUE인 경우 필수입니다. ONTAP-SAN과 ONTAP-SAN 경제입니다
ONTAP	chapTargetUsername 을 선택합니다	대상 사용자 이름입니다. useCHAP = TRUE인 경우 필수입니다. ONTAP-SAN과 ONTAP-SAN 경제입니다
ONTAP	챕터타겟이니터시크릿	CHAP 타겟 이니시에이터 암호입니다. useCHAP = TRUE인 경우 필수입니다. ONTAP-SAN과 ONTAP-SAN 경제입니다

이 단계에서 만든 암호는 다음 단계에서 만든 트리젠백엔드Config 개체의 '증명서' 필드에 참조됩니다.

2단계: 을 작성합니다 `TridentBackendConfig` 있습니다

이제 '트리엔백구성' CR을 만들 준비가 되었습니다. 이 예에서 'ONTAP-SAN' 드라이버를 사용하는 백엔드는 아래에 나와 있는 ' TridentBackendConfig ' 객체를 사용하여 생성합니다.

```
$ kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

3단계: 의 상태를 확인합니다 `TridentBackendConfig` 있습니다

이제 '트리펜엔드구성' CR을 생성했으므로 상태를 확인할 수 있습니다. 다음 예를 참조하십시오.

```
$ kubectl -n trident get tbc backend-tbc-ontap-san
NAME                     BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

백엔드가 성공적으로 생성되어 '트리엔백엔드구성' CR에 바인딩되었습니다.

위상은 다음 값 중 하나를 사용할 수 있습니다.

- 바운드: 트리젠백엔드Config CR은 백엔드와 연결되며 백엔드에는 트리젠백엔드Config CR의 uid로 설정된 configRef가 포함되어 있습니다.
- 'Unbound': "로 표현됨. 트리젠백엔드Config 객체가 백엔드에 바인딩되지 않습니다. 새로 만든 트리젠백엔드Config CRS는 기본적으로 이 단계에 있습니다. 단계가 변경된 후에는 다시 바인딩되지 않은 상태로 되돌릴 수 없습니다.
- "트리엔테구성" CR의 "설명 정책"이 삭제되도록 설정되었습니다. 트리젠백엔드Config CR이 삭제되면 삭제 상태로 전환됩니다.
  - 백엔드에 영구 볼륨 클레임(PVCs)이 없는 경우, 트리엔백엔드구성을 삭제하면 Astra Trident가 백엔드를 삭제하고 '트리엔백구성' CR을 삭제합니다.
  - 백엔드에 PVC가 하나 이상 있는 경우 삭제 상태로 전환됩니다. 이후 트리젠백엔드Config CR도 삭제 단계로 진입한다. 모든 PVC가 삭제된 후에만 백엔드 및 트리젠백엔드구성이 삭제됩니다.
- 손실: 트리젠백엔드Config CR과 관련된 백엔드가 실수로 또는 고의적으로 삭제되었고, 트리젠백엔드Config CR에는 삭제된 백엔드에 대한 참조가 여전히 있습니다. 이 경우에도 '항목 정책' 값에 관계없이 '트리멘백엔드구성' CR은 삭제할 수 있습니다.
- 알 수 없음: Astra Trident가 ' Trident' CR과 연관된 백엔드의 상태 또는 존재를 확인할 수 없습니다. 예를 들어, API 서버가 응답하지 않거나 'tridentbackends.trident.netapp.io` CRD가 없는 경우 이 경우 사용자의 개입이 필요할 수 있습니다.

이 단계에서는 백엔드가 성공적으로 생성됩니다! 다음과 같은 몇 가지 작업을 추가로 처리할 수 있습니다 "백엔드 업데이트 및 백엔드 삭제".

(선택 사항) 4단계: 자세한 내용을 확인하십시오

다음 명령을 실행하여 백엔드에 대한 자세한 정보를 얻을 수 있습니다.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                      BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

또한 '트리엔백구성'의 YAML/JSON 덤프를 얻을 수도 있습니다.

```
$ kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: "2021-04-21T20:45:11Z"
  finalizers:
  - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

'backendInfo'에는 '트리젠BackendConfig' CR에 대응하여 만든 백엔드의 'backendName'과 'backendUUID'가 포함되어 있습니다. 'lastOperationStatus' 필드는 사용자 트리거(예: 사용자가 'spec'에서 무언가를 변경한 경우) 또는 Astra Trident(예: Astra Trident 재시작 시)에 의해 트리거될 수 있는 '트리엔백엔드 Config' CR의 마지막 작업 상태를 나타냅니다. 성공 또는 실패일 수 있습니다. 단계 는 트리젠백엔드Config CR과 백엔드 간의 관계를 나타냅니다. 위의 예에서 'phase'는 값이 바인딩되어 있어 '트리젠백엔드구성' CR이 백엔드와 연결되어 있음을 의미합니다.

"kubbctl -n trident tbc <tbc-cr-name>" 명령을 실행하여 이벤트 로그의 세부 정보를 확인할 수 있습니다.

> tridentctl을 사용하여 연결된 'TrientBackendConfig' 객체가 포함된 백엔드는 업데이트하거나 삭제할 수 없습니다. tridentctl과 트리멘BackendConfig의 전환 단계를 이해하려면 "여기 를 참조하십시오".

kubeck을 사용하여 백엔드 관리 수행

kubbtl을 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보십시오.

백엔드를 삭제합니다

'트리멘BackendConfig'를 삭제하면 Astra Trident가 백엔드 삭제/보존('정책' 기준)을 수행하도록 지시합니다. 백엔드를 삭제하려면 deletionPolicy가 delete로 설정되어 있는지 확인합니다. 트리젠백엔드구성만 삭제하려면, '정책'이 '유지'로 설정되어 있는지 확인하십시오. 이렇게 하면 백엔드가 계속 존재하고 'tridentctl'을 사용하여 관리할 수 있습니다.

다음 명령을 실행합니다.

```
$ kubectl delete tbc <tbc-name> -n trident
```

Astra Trident는 트리엔엔드구성에서 사용 중인 Kubernetes 비밀을 삭제하지 않습니다. Kubernetes 사용자는 기밀을 정해야 합니다. 비밀 정보를 삭제할 때는 주의해야 합니다. 암호는 백엔드에서 사용하지 않는 경우에만 삭제해야 합니다.

기존 백엔드를 봅니다

다음 명령을 실행합니다.

```
$ kubectl get tbc -n trident
```

또한 'tridentctl get backend-n trident' 또는 'tridentctl get backend-o YAML-n trident'를 실행하여 존재하는 모든 백엔드 목록을 확인할 수 있습니다. 이 목록에는 tridentctl로 만든 백엔드 또한 포함됩니다.

백엔드를 업데이트합니다

백엔드를 업데이트해야 하는 이유는 여러 가지가 있을 수 있습니다.

- 스토리지 시스템에 대한 자격 증명이 변경되었습니다. 자격 증명을 업데이트하려면 '트리펜엔드구성' 객체에 사용되는 Kubernetes Secret를 업데이트해야 합니다. Astra Trident가 자동으로 백엔드를 제공된 최신 자격 증명으로 업데이트합니다. 다음 명령을 실행하여 Kubernetes Secret를 업데이트하십시오.

```
$ kubectl apply -f <updated-secret-file.yaml> -n trident
```

- 매개 변수(예: 사용 중인 ONTAP SVM의 이름)를 업데이트해야 합니다. 이 경우 Kubernetes를 통해 '트리펜엔드구성' 객체를 직접 업데이트할 수 있습니다.

```
$ kubectl apply -f <updated-backend-file.yaml>
```

또는 다음 명령을 실행하여 기존 '트리엔백엔드구성' CR을 변경합니다.

```
$ kubectl edit tbc <tbc-name> -n trident
```

백엔드 업데이트에 실패하면 백엔드는 마지막으로 알려진 구성으로 계속 유지됩니다. 'kubtl get tbc<tbc-name>-o YAML-n trident' 또는 'kubtl t설명해 tbc<tbc-name>-n trident'를 실행하여 로그를 보고 원인을 확인할 수 있습니다.

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

tridentctl을 사용하여 백엔드 관리를 수행합니다

tridentctl을 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보십시오.

백엔드를 생성합니다

을 만든 후 "백엔드 구성 파일"에서 다음 명령을 실행합니다.

```
$ tridentctl create backend -f <backend-file> -n trident
```

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

```
$ tridentctl logs -n trident
```

구성 파일의 문제를 확인하고 수정한 후에는 간단히 'create' 명령을 다시 실행할 수 있습니다.

백엔드를 삭제합니다

Astra Trident에서 백엔드를 삭제하려면 다음을 수행합니다.

1. 백엔드 이름 검색:

   ```
   $ tridentctl get backend -n trident
   ```

2. 백엔드를 삭제합니다.

   ```
   $ tridentctl delete backend <backend-name> -n trident
   ```

> Astra Trident가 백엔드에서 여전히 존재하는 볼륨 및 스냅샷을 프로비저닝한 경우 백엔드를 삭제하면 새 볼륨이 백엔드에서 프로비저닝되지 않습니다. 백엔드는 계속해서 "삭제" 상태에 있으며, Trident는 삭제될 때까지 해당 볼륨 및 스냅샷을 계속 관리합니다.

기존 백엔드를 봅니다

Trident가 알고 있는 백엔드를 보려면 다음을 실행합니다.

- 요약을 보려면 다음 명령을 실행합니다.

```
$ tridentctl get backend -n trident
```

- 모든 세부 정보를 보려면 다음 명령을 실행합니다.

```
$ tridentctl get backend -o json -n trident
```

백엔드를 업데이트합니다

새 백엔드 구성 파일을 생성한 후 다음 명령을 실행합니다.

```
$ tridentctl update backend <backend-name> -f <backend-file> -n trident
```

백엔드 업데이트에 실패하면 백엔드 구성에 문제가 있거나 잘못된 업데이트를 시도했습니다. 다음 명령을 실행하여 로그를 보고 원인을 확인할 수 있습니다.

```
$ tridentctl logs -n trident
```

구성 파일의 문제를 확인하고 수정한 후에는 간단히 'update' 명령을 다시 실행할 수 있습니다.

백엔드를 사용하는 스토리지 클래스를 식별합니다

이것은 JSON으로 백엔드 객체에 대해 tridentctl이 출력하는 질문의 예입니다. 이 유틸리티는 설치해야 하는 JQ 유틸리티를 사용합니다.

```
$ tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

이는 ' 재젠백엔드구성'을 사용하여 만든 백엔드에도 적용됩니다.

백엔드 관리 옵션 간 이동

Astra Trident에서 백엔드를 관리하는 다양한 방법에 대해 알아보십시오. 트리젠백엔드구성이 도입됨에 따라 관리자는 이제 두 가지 방법으로 백엔드를 관리할 수 있게 되었습니다. 이 질문은 다음과 같습니다.

- tridentctl을 사용하여 만든 백엔드는 트리엔백엔드구성을 사용하여 관리할 수 있습니까?
- 트리덴ctl을 사용하여 트리엔디Config를 사용하여 만든 백엔드를 관리할 수 있습니까?

관리 `tridentctl` 을 사용하여 백엔드를 만듭니다 `TridentBackendConfig`

이 섹션에서는 'Tridentctl' 객체를 만들어 Kubernetes 인터페이스를 통해 직접 생성된 백엔드를 관리하는 데 필요한 단계에 대해 설명합니다.

이 내용은 다음 시나리오에 적용됩니다.

- tridentctl로 만든 트리젠백엔드구성이 없는 기존 백엔드입니다.
- tridentctl로 만든 새 백엔드와 다른 트리젠BackendConfig 개체가 있습니다.

두 시나리오 모두 Astra Trident가 볼륨을 예약하고 운영하면서 백엔드가 계속 존재할 것입니다. 관리자는 다음 두 가지 옵션 중 하나를 선택할 수 있습니다.

- tridentctl을 사용하여 만든 백엔드를 관리하려면 계속 사용합니다.
- tridentctl을 사용하여 만든 백엔드를 새 트리젠BackendConfig 개체에 바인딩합니다. 이를 통해 뒷골은 트리덴틀이 아니라 쿠벤틀로 관리된다는 뜻이다.

kubbeck을 사용하여 기존 백엔드를 관리하려면 기존 백엔드에 바인딩되는 '트리젠백엔드구성'을 생성해야 합니다. 작동 방식에 대한 개요는 다음과 같습니다.

1. Kubernetes 암호를 생성하십시오. 비밀에는 Astra Trident가 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. '트리멘백엔드구성' 객체를 만듭니다. 스토리지 클러스터/서비스에 대한 자세한 내용과 이전 단계에서 생성한 암호를 참조하십시오. 동일한 구성 매개 변수(예: 'pec.backendName', 'pec.storagePrefix', 'pec.storageDriverName' 등)를 지정할 때는 주의해야 합니다. '현재 백엔드 이름'을 설정해야 합니다.

단계 **0:** 백엔드를 식별합니다

기존 백엔드에 바인딩하는 '트리젠백엔드구성'을 생성하려면 백엔드의 구성을 확보해야 합니다. 이 예에서는 다음과 같은 JSON 정의를 사용하여 백엔드를 생성했다고 가정합니다.

```
$ tridentctl get backend ontap-nas-backend -n trident
+----------------------+----------------
+--------------------------------------+--------+---------+
|          NAME        | STORAGE DRIVER |                  UUID
| STATE   | VOLUMES |
+----------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend    | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+----------------------+----------------
+--------------------------------------+--------+---------+

$ cat ontap-nas-backend.json

{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.10.10.1",
    "dataLIF": "10.10.10.2",
    "backendName": "ontap-nas-backend",
    "svm": "trident_svm",
```

```
    "username": "cluster-admin",
    "password": "admin-password",

    "defaults": {
        "spaceReserve": "none",
        "encryption": "false"
    },
    "labels":{"store":"nas_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"msoffice", "cost":"100"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"25"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

1단계**: Kubernetes Secret** 생성

이 예에 표시된 것처럼 백엔드에 대한 자격 증명이 포함된 암호를 생성합니다.

```
$ cat tbc-ontap-nas-backend-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  passWord: admin-password

$ kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

2단계: A를 작성합니다 `TridentBackendConfig` 있습니다

다음 단계는 기존 ONTAP-NAS-백엔드에 자동으로 바인딩되는 '트리젠백엔드구성' CR을 생성하는 것입니다(예:). 다음 요구 사항이 충족되는지 확인합니다.

- 같은 백엔드 이름은 'sepec.backendName'에 정의되어 있습니다.
- 구성 매개 변수는 원래 백엔드와 동일합니다.
- 가상 스토리지 풀(있는 경우)은 원래 백엔드와 동일한 순서를 유지해야 합니다.
- 자격 증명은 일반 텍스트가 아닌 Kubernetes Secret을 통해 제공됩니다.

이 경우 트리젠백엔드구성은 다음과 같습니다.

```
$ cat backend-tbc-ontap-nas.yaml
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'

$ kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

3단계: 의 상태를 확인합니다 `TridentBackendConfig` 있습니다

트리젠백엔드구성이 만들어지면 그 단계는 반드시 '바운드'되어야 한다. 또한 기존 백엔드의 백엔드 이름과 UUID도 동일하게 반영되어야 합니다.

```
$ kubectl -n trident get tbc tbc-ontap-nas-backend -n trident
NAME                    BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
$ tridentctl get backend -n trident
+----------------------+----------------
+--------------------------------------+--------+---------+
|          NAME        | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+----------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+----------------------+----------------
+--------------------------------------+--------+---------+
```

이제 백엔드는 'tbc-ONTAP-nas-backend' 트리펜엔드구성 객체를 사용하여 완벽하게 관리됩니다.

관리 `TridentBackendConfig` 을 사용하여 백엔드를 만듭니다 `tridentctl`

트리덴ctl은 트리엔백구성(TrientBackendConfig)을 사용하여 만든 백엔드를 나열하는 데 사용할 수 있습니다. 또한 관리자는 트리텐틀Config를 삭제하고 pec.deletionPolicy` 가 "Stain"으로 설정되어 있는지 확인하여 tridentctl을 통해 이러한 백엔드를 완벽하게 관리할 수도 있습니다.

단계 **0:** 백엔드를 식별합니다

예를 들어, 다음 백엔드가 ``트리엔백구성”을 사용하여 생성되었다고 가정해 보겠습니다.

```
$ kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

$ tridentctl get backend ontap-san-backend -n trident
+--------------------+----------------
+--------------------------------------+--------+---------+
|       NAME         | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+--------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+--------------------+----------------
+--------------------------------------+--------+---------+
```

출력으로부터 '(트리젠백엔드구성) '이(가) 성공적으로 생성되었으며 백엔드에 바인딩되어 있습니다 [백엔드의 UUID 확인].

1단계: 확인 `deletionPolicy` 가 로 설정되어 있습니다 `retain`

정책이라는 가치를 한번 살펴보자. 이를 보철로 설정해야 합니다. 이렇게 하면 ' TrientBackendConfig ' CR을 삭제할 때 백엔드 정의가 계속 존재하고 'tridentctl'을 사용하여 관리할 수 있습니다.

```
$ kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
$ kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
$ kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

'정책'이 '유지'로 설정되어 있지 않으면 다음 단계로 진행하지 마십시오.

2단계: 를 삭제합니다 `TridentBackendConfig` 있습니다

마지막 단계는 트리엔디Config CR을 삭제하는 것이다. '정책'이 '유지'로 설정되어 있는지 확인한 후 삭제를 계속 수행할 수 있습니다.

```
$ kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

$ tridentctl get backend ontap-san-backend -n trident
+--------------------+----------------
+--------------------------------------+--------+---------+
|       NAME         | STORAGE DRIVER |                  UUID
| STATE   | VOLUMES |
+--------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+--------------------+----------------
+--------------------------------------+--------+---------+
```

트리젠벤트Config 객체를 삭제하면 Astra Trident는 백엔드 자체를 삭제하지 않고 간단히 해당 객체를 제거합니다.

스토리지 클래스를 관리합니다

스토리지 클래스 생성, 스토리지 클래스 삭제 및 기존 스토리지 클래스 보기에 대한 정보를 찾을 수 있습니다.

스토리지 클래스를 설계합니다

을 참조하십시오 "여기" 스토리지 클래스의 정의 및 구성 방법에 대한 자세한 내용은 를 참조하십시오.

스토리지 클래스를 생성합니다

스토리지 클래스 파일이 있으면 다음 명령을 실행합니다.

```
kubectl create -f <storage-class-file>
```

'<storage-class-file>'을(를) 스토리지 클래스 파일 이름으로 바꿔야 합니다.

스토리지 클래스를 삭제합니다

Kubernetes에서 스토리지 클래스를 삭제하려면 다음 명령을 실행합니다.

```
kubectl delete storageclass <storage-class>
```

'<storage-class>'은(는) 스토리지 클래스로 교체해야 합니다.

이 스토리지 클래스를 통해 생성된 영구 볼륨은 변경되지 않으며 Astra Trident는 계속 관리합니다.

> Astra Trident는 생성한 볼륨에 대해 빈 "fsType"을 적용합니다. iSCSI 백엔드의 경우 StorageClass 에 parameters.fsType 을 적용하는 것이 좋습니다. esixing StorageClasses를 삭제하고 parameters.fsType을 지정하여 다시 생성해야 합니다.

기존 스토리지 클래스를 봅니다

- 기존 Kubernetes 스토리지 클래스를 보려면 다음 명령을 실행합니다.

```
kubectl get storageclass
```

- Kubernetes 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행합니다.

```
kubectl get storageclass <storage-class> -o json
```

- Astra Trident의 동기화된 스토리지 클래스를 보려면 다음 명령을 실행합니다.

```
tridentctl get storageclass
```

- Astra Trident의 동기화된 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행합니다.

```
tridentctl get storageclass <storage-class> -o json
```

기본 스토리지 클래스를 설정합니다

Kubernetes 1.6에는 기본 스토리지 클래스를 설정하는 기능이 추가되었습니다. 사용자가 영구 볼륨 클레임(PVC)에 영구 볼륨을 지정하지 않는 경우 영구 볼륨을 프로비저닝하는 데 사용되는 스토리지 클래스입니다.

- 스토리지 클래스 정의에서 주석 'torageclass.Kubernetes.io/is-default-class'를 true로 설정하여 기본 스토리지 클래스를 정의합니다. 사양에 따라 다른 값이나 주석 부재는 FALSE로 해석됩니다.
- 다음 명령을 사용하여 기존 스토리지 클래스를 기본 스토리지 클래스로 구성할 수 있습니다.

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 마찬가지로 다음 명령을 사용하여 기본 스토리지 클래스 주석을 제거할 수 있습니다.

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

또한 Trident 설치 프로그램 번들에는 이 주석을 포함하는 예제도 있습니다.

> 언제든지 클러스터에는 기본 스토리지 클래스가 하나만 있어야 합니다. Kubernetes에서 둘 이상의 작업을 수행하는 것을 기술적으로 금지하지는 않지만 기본 스토리지 클래스가 없는 것처럼 동작합니다.

스토리지 클래스에 대한 백엔드를 식별합니다

이 예는 Astra Trident 백엔드 객체에 대해 tridentctl이 출력하는 JSON으로 답할 수 있는 질문의 예입니다. 먼저 설치해야 할 JQ 유틸리티가 사용됩니다.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

볼륨 작업을 수행합니다

볼륨 관리를 위한 Astra Trident의 기능에 대해 알아보십시오.

- "CSI 토폴로지를 사용합니다"
- "스냅샷 작업"
- "볼륨 확장"
- "볼륨 가져오기"

CSI 토폴로지를 사용합니다

Astra Trident는 을 사용하여 Kubernetes 클러스터에 있는 노드를 선택적으로 생성하여 연결할 수 있습니다 "CSI 토폴로지 기능". CSI 토폴로지 기능을 사용하면 지역 및 가용성 영역에 따라 볼륨에 대한 액세스가 노드의 하위 집합으로 제한될 수 있습니다. 오늘날의 클라우드 공급자는 Kubernetes 관리자가 영역 기반의 노드를 생성할 수 있습니다. 노드는 지역 내 또는 여러 지역의 여러 가용성 영역에 위치할 수 있습니다. Astra Trident는 다중 영역 아키텍처에서 워크로드용 볼륨 프로비저닝을 지원하기 위해 CSI 토폴로지를 사용합니다.

> CSI 토폴로지 기능에 대해 자세히 알아보십시오 "여기".

Kubernetes는 두 가지 고유한 볼륨 바인딩 모드를 제공합니다.

- 'VolumeBindingMode'를 'Immediate'로 설정하면 Astra Trident는 토폴로지 인식 없이 볼륨을 생성합니다. 볼륨 바인딩 및 동적 프로비저닝은 PVC가 생성될 때 처리됩니다. 이는 기본 'VolumeBindingMode'이며 토폴로지 제약 조건을 적용하지 않는 클러스터에 적합합니다. 영구 볼륨은 요청 포드의 예약 요구 사항에 의존하지 않고 생성됩니다.
- VolumeBindingMode를 WaitForFirstConsumer로 설정하면 PVC를 사용하는 POD가 예약 및 생성될 때까지

PVC에 대한 영구 볼륨의 생성 및 바인딩이 지연됩니다. 이렇게 하면 토폴로지 요구 사항에 따라 적용되는 일정 제한을 충족하기 위해 볼륨이 생성됩니다.

> WaitForFirstConsumer의 바인딩 모드에는 토폴로지 레이블이 필요하지 않습니다. 이 기능은 CSI 토폴로지 기능과 독립적으로 사용할 수 있습니다.

필요한 것

CSI 토폴로지를 사용하려면 다음이 필요합니다.

- 1.17 이상을 실행하는 Kubernetes 클러스터

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- 클러스터의 노드에는 토폴로지 인식(topology.Kubernetes.io/region' 및 topology.Kubernetes.io/zone')을 소개하는 레이블이 있어야 합니다. Astra Trident가 토폴로지 인식을 위해 설치되기 전에 클러스터의 노드에 이러한 레이블 * 이 있어야 합니다.

```
$ kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

1단계**:** 토폴로지 인식 백엔드 생성

Astra Trident 스토리지 백엔드는 가용성 영역에 따라 볼륨을 선택적으로 프로비저닝하도록 설계할 수 있습니다. 각 백엔드는 지원해야 하는 영역과 영역의 목록을 나타내는 선택적 'SupportedTopologies' 블록을 전달할 수 있습니다. 이러한 백엔드를 사용하는 StorageClasses의 경우 지원되는 영역/영역에서 예약된 애플리케이션에서 요청하는 경우에만 볼륨이 생성됩니다.

다음은 백엔드 정의의 예입니다.

```
{
 "version": 1,
 "storageDriverName": "ontap-san",
 "backendName": "san-backend-us-east1",
 "managementLIF": "192.168.27.5",
 "svm": "iscsi_svm",
 "username": "admin",
 "password": "xxxxxxxxxxxx",
 "supportedTopologies": [
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-a"},
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-b"}
]
}
```

'SupportedTopologies'는 백엔드 당 지역 및 영역 목록을 제공하는 데 사용됩니다. 이러한 영역 및 영역은 StorageClass 에서 제공할 수 있는 허용 가능한 값의 목록을 나타냅니다. 백엔드에서 제공되는 지역 및 영역의 하위 집합이 포함된 StorageClasses의 경우 Astra Trident는 백엔드에 볼륨을 생성합니다.

스토리지 풀별로 'SupportedTopologies'를 정의할 수도 있습니다. 다음 예를 참조하십시오.

```
{"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "nas-backend-us-central1",
"managementLIF": "172.16.238.5",
"svm": "nfs_svm",
"username": "admin",
"password": "Netapp123",
"supportedTopologies": [
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"},
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
    ]
"storage": [
   {
       "labels": {"workload":"production"},
        "region": "Iowa-DC",
        "zone": "Iowa-DC-A",
        "supportedTopologies": [
            {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"}
        ]
    },
    {
        "labels": {"workload":"dev"},
         "region": "Iowa-DC",
         "zone": "Iowa-DC-B",
         "supportedTopologies": [
             {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
         ]
     }
]
}
```

이 예에서는 "reGion" 및 "zone" 레이블이 스토리지 풀의 위치를 나타냅니다. "topology.Kubernetes.io/region" 및 "topology.Kubernetes.io/zone"은 스토리지 풀을 사용할 수 있는 위치를 지정합니다.

2단계**:** 토폴로지를 인식하는 **StorageClasses**를 정의합니다

클러스터의 노드에 제공되는 토폴로지 레이블을 기반으로 StorageClasses를 정의하여 토폴로지 정보를 포함할 수 있습니다. 이렇게 하면 PVC 요청에 대한 후보 역할을 하는 스토리지 풀과 Trident에서 제공하는 볼륨을 사용할 수 있는 노드의 하위 세트가 결정됩니다.

다음 예를 참조하십시오.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"
```

위에서 제공한 StorageClass 정의에서 volumeBindingMode는 WaitForFirstConsumer로 설정됩니다. 이 StorageClass에 요청된 PVC는 POD에서 참조될 때까지 작동하지 않습니다. 그리고, “allowedTopologies”는 사용할 지역과 지역을 제공한다. NetApp-SAN-us-east1 StorageClass는 위에 정의된 'AN-BACKEND-Us-east1' 백엔드에서 PVC를 생성합니다.

3단계**: PVC** 생성 및 사용

StorageClass가 생성되어 백엔드에 매핑되면 PVC를 생성할 수 있습니다.

아래의 '샘플’을 참조하십시오.

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

이 매니페스트를 사용하여 PVC를 만들면 다음과 같은 결과가 발생합니다.

```
$ kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
$ kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
$ kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Trident에서 볼륨을 생성하여 PVC에 바인딩하려면 POD에서 PVC를 사용합니다. 다음 예를 참조하십시오.

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

이 podSpec은 us-east1 지역에 존재하는 노드에서 pPod를 예약하고 us-east1-a 또는 us-east1-b 영역에 있는 노드 중에서 선택하도록 지시합니다.

다음 출력을 참조하십시오.

```
$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
$ kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

포함할 백엔드를 업데이트합니다 `supportedTopologies`

기존 백엔드는 'tridentctl backend update'를 사용하여 'upportedTopologies' 목록을 포함하도록 업데이트할 수 있습니다. 이는 이미 프로비저닝된 체적에 영향을 주지 않으며 후속 PVC에만 사용됩니다.

자세한 내용을 확인하십시오

- "컨테이너에 대한 리소스를 관리합니다"
- "노드 선택기"
- "친화성 및 반친화성"
- "오염과 내약입니다"

스냅샷 작업

Astra Trident의 20.01 릴리즈부터 Kubernetes 계층에서 PVS의 스냅샷을 생성할 수 있습니다. 이러한 스냅샷을 사용하여 Astra Trident에서 생성한 볼륨의 시점 복사본을 유지하고 추가 볼륨(클론) 생성을 예약할 수 있습니다. 볼륨 스냅샷은 'ONTAP-NAS', 'ONTAP-SAN', 'ONTAP-SAN-이코노미', '솔리드산', 'GCP-CV'에서 지원됩니다. Azure-NetApp-files 드라이버를 비롯한 모든 드라이버를 지원합니다.

> 이 기능은 Kubernetes 1.17(베타)에서 사용할 수 있으며 1.20에서 GA됩니다. 베타 버전에서 GA로 전환하는 데 따른 변경 사항을 이해하려면 를 참조하십시오 "릴리스 블로그". GA로 졸업하면 v1 API 버전이 도입되고 v1beta1 스냅샷과 역호환됩니다.

필요한 것

- 볼륨 스냅숏을 생성하려면 외부 스냅샷 컨트롤러와 일부 사용자 지정 리소스 정의(CRD)를 생성해야 합니다. 사용 중인 Kubernetes Orchestrator는 이 책임이 있습니다(예: Kubeadm, GKE, OpenShift).

다음과 같이 외부 스냅샷 컨트롤러 및 스냅샷 CRD를 생성할 수 있습니다.

1. 볼륨 스냅샷 CRD 생성:

```
$ cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. 원하는 네임스페이스에 스냅샷 컨트롤러를 생성합니다. 아래 YAML 매니페스트를 편집하여 네임스페이스를 수정하십시오.

> GKE 환경에서 필요 시 볼륨 스냅샷을 설정하는 경우 스냅샷-컨트롤러를 생성하지 마십시오. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

> CSI Snapshotter는 를 제공합니다 "Webhook 확인 중" 사용자가 기존 v1beta1 스냅샷의 유효성을 검사하고 유효한 리소스 객체인지 확인할 수 있도록 합니다. Validating webhook는 잘못된 스냅샷 개체에 자동으로 레이블을 지정하며 나중에 유효하지 않은 객체가 생성되지 않도록 합니다. Validating webhook은 Kubernetes Orchestrator에 의해 구축됩니다. 유효성을 검사하는 Webhook을 수동으로 배포하는 방법은 지침을 참조하십시오 "여기". 유효하지 않은 스냅샷 매니페스트의 예를 찾습니다 "여기".

아래 예에서는 스냅샷 작업에 필요한 구조를 설명하고 스냅샷을 생성 및 사용하는 방법을 보여 줍니다.

1단계: 을 설정합니다 `VolumeSnapshotClass`

볼륨 스냅샷을 생성하기 전에 를 설정합니다 "d7ca7162c394dee752c35d07a92823da".

```
$ cat snap-sc.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.17 -
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

이 강물은 아스트라 트라이던트 CSI 드라이버를 가리킵니다. 정책이란 '삭제'나 '유지'가 될 수 있습니다. "유지"로 설정하면 "VolumeSnapshot" 객체가 삭제되더라도 스토리지 클러스터의 기본 물리적 스냅샷이 보존됩니다.

2단계: 기존 **PVC**의 스냅샷을 생성합니다

```
$ cat snap.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.17 -
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

pvc1이라는 PVC에 대해 스냅샷이 생성되고 스냅샷 이름이 pvc1-snap으로 설정됩니다.

```
$ kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

$ kubectl get volumesnapshots
NAME                   AGE
pvc1-snap              50s
```

이로 인해 'VolumeSnapshot' 객체가 생성되었습니다. VolumeSnapshot은 PVC와 유사하며 실제 스냅샷을 나타내는 'VolumeSnapshotContent' 객체와 연결됩니다.

pvc1-snap VolumeSnapshot에 대한 VolumeSnapshotContent 객체를 설명하여 식별할 수 있다.

```
$ kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
.
.
```

'스냅샷 컨텐츠 이름'은 이 스냅샷을 제공하는 VolumeSnapshotContent 객체를 식별합니다. Ready to Use 매개변수는 스냅샷이 새 PVC를 생성하는 데 사용될 수 있음을 나타냅니다.

3단계: VolumeSnapshots에서 PVC를 생성합니다

스냅샷을 사용하여 PVC를 생성하는 예는 다음 예를 참조하십시오.

```
$ cat pvc-from-snap.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

"다소스"는 데이터 소스로 "pvc1-snap"이라는 VolumeSnapshot을 사용하여 PVC를 생성해야 함을 나타냅니다. 이렇게 하면 Astra Trident가 스냅샷에서 PVC를 생성하도록 지시합니다. PVC가 생성된 후 POD에 부착하여 다른 PVC와 마찬가지로 사용할 수 있습니다.

> 연결된 스냅샷이 있는 영구 볼륨을 삭제하면 해당 Trident 볼륨이 "삭제 상태"로 업데이트됩니다. Astra Trident 볼륨을 삭제하려면 볼륨의 스냅샷을 제거해야 합니다.

자세한 내용을 확인하십시오

- "볼륨 스냅숏"
- "d7ca7162c394dee752c35d07a92823da"

볼륨 확장

Astra Trident를 사용하면 Kubernetes 사용자가 볼륨을 생성한 후 확장할 수 있습니다. iSCSI 및 NFS 볼륨을 확장하는 데 필요한 구성에 대한 정보를 찾습니다.

iSCSI 볼륨을 확장합니다

CSI 프로비저닝을 사용하여 iSCSI PV(Persistent Volume)를 확장할 수 있습니다.

> iSCSI 볼륨 확장은 ONTAP-SAN, ONTAP-SAN-이코노미, Solidfire-SAN 드라이버로 지원되며 Kubernetes 1.16 이상이 필요합니다.

개요

iSCSI PV를 확장하면 다음 단계가 포함됩니다.

- StorageClass 정의를 편집하여 allowVolumeExpansion 필드를 true로 설정합니다.
- PVC 정의 편집 및 'pec.resources.requests.storage'를 업데이트하여 원래 크기보다 큰 새로 원하는 크기를 반영한다.
- 크기를 조절하려면 PV를 포드에 부착해야 합니다. iSCSI PV의 크기를 조정할 때 두 가지 시나리오가 있습니다.
  - PV가 포드에 연결된 경우 Astra Trident는 스토리지 백엔드의 볼륨을 확장하고 디바이스를 다시 검사하며 파일 시스템의 크기를 조정합니다.
  - 연결되지 않은 PV의 크기를 조정하려고 하면 Astra Trident가 스토리지 백엔드의 볼륨을 확장합니다. PVC가 POD에 바인딩되면 Trident가 디바이스를 다시 검사해 파일 시스템의 크기를 조정합니다. 그런 다음 확장 작업이 성공적으로 완료된 후 Kubernetes에서 PVC 크기를 업데이트합니다.

아래 예에서는 iSCSI PVS의 확장 작동 방식을 보여 줍니다.

1단계**:** 볼륨 확장을 지원하도록 **StorageClass**를 구성합니다

```
$ cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

이미 존재하는 StorageClass의 경우 allowVolumeExpansion 매개변수를 포함하도록 편집합니다.

2단계: 생성한 StorageClass를 사용하여 PVC를 생성합니다

```
$ cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Astra Trident가 PV(Persistent Volume)를 생성하여 이 PVC(Persistent Volume Claim)와 연결합니다.

```
$ kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               10s
```

3단계: PVC를 부착하는 POD를 정의합니다

이 예제에서는 'AN-PVC'를 사용하는 POD가 생성됩니다.

```
$  kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
centos-pod   1/1     Running   0          65s

$  kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    centos-pod
```

4단계: PV를 확장합니다

1Gi 에서 2Gi 로 생성된 PV 의 크기를 조정하려면 PVC 정의를 편집하여 'pec.resources.requests.storage'를 2Gi 로 업데이트합니다.

```
$ kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 ...
```

5단계: 확장 확인

PVC, PV, Astra Trident 볼륨의 크기를 확인하여 확장이 제대로 작동하는지 확인할 수 있습니다.

```
$ kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
$ tridentctl get volumes -n trident
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

NFS 볼륨을 확장합니다

Astra Trident는 ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup, GCP-CV, Azure-NetApp-files 백엔드에 프로비저닝된 NFS PVS의 볼륨 확장을 지원합니다.

1단계: 볼륨 확장을 지원하도록 **StorageClass**를 구성합니다

NFS PV의 크기를 조정하려면 먼저 관리자가 "allowVolumeExpansion" 필드를 "true"로 설정하여 볼륨 확장을 허용하도록 스토리지 클래스를 구성해야 합니다.

```
$ cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

이 옵션 없이 스토리지 클래스를 이미 생성한 경우 'kubeck edit storageclass'를 사용하여 기존 스토리지 클래스를 편집하여 볼륨을 확장할 수 있습니다.

2단계: 생성한 StorageClass를 사용하여 PVC를 생성합니다

```
$ cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Astra Trident는 이 PVC에 대해 20MiB NFS PV를 생성해야 합니다.

```
$ kubectl get pvc
NAME           STATUS   VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

$ kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

3단계: PV를 확장합니다

새로 만든 20MiB PV의 크기를 1GiB로 조정하려면 PVC를 편집하여 pec.resources.requests.storage를 1GB로 설정합니다.

```
$ kubectl edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
...
```

4단계: 확장을 확인합니다

PVC, PV, Astra Trident 볼륨의 크기를 확인하여 크기가 올바르게 작동하는지 확인할 수 있습니다.

```
$ kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

$ kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

$ tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n
trident
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

볼륨 가져오기

tridentctl import를 사용하여 기존 스토리지 볼륨을 Kubernetes PV로 가져올 수 있습니다.

볼륨 가져오기를 지원하는 드라이버입니다

이 표에는 볼륨 가져오기를 지원하는 드라이버와 볼륨 가져오기가 도입된 릴리스가 나와 있습니다.

드라이버	놓습니다
'ONTAP-NAS'	19.04
'ONTAP-NAS-Flexgroup'	19.04
'솔더불-산'	19.04
'Azure-NetApp-파일'	19.04

드라이버	놓습니다
GCP-CV	19.04
'ONTAP-SAN'	19.04

볼륨을 가져와야 하는 이유는 무엇입니까?

Trident로 볼륨을 가져오는 데는 다음과 같은 몇 가지 사용 사례가 있습니다.

- 응용 프로그램을 Containerizing 하고 기존 데이터 집합을 다시 사용합니다
- 수명이 짧은 애플리케이션에 대한 데이터 세트 클론 사용
- 오류가 발생한 Kubernetes 클러스터를 리빌드합니다
- 재해 복구 중에 애플리케이션 데이터 마이그레이션

가져오기는 어떻게 작동합니까?

영구 볼륨 클레임(PVC) 파일은 볼륨 가져오기 프로세스에서 PVC를 생성하는 데 사용됩니다. 최소한 PVC 파일에는 다음 예제와 같이 이름, 네임스페이스, accessModes 및 storageClassName 필드가 포함되어야 합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

tridentctl 클라이언트는 기존 스토리지 볼륨을 가져오는 데 사용됩니다. Trident는 볼륨 메타데이터를 유지하고 PVC 및 PV를 생성하여 볼륨을 가져옵니다.

```
$ tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-
file>
```

스토리지 볼륨을 가져오려면 볼륨을 포함하는 Astra Trident 백엔드의 이름과 스토리지에서 볼륨을 고유하게 식별하는 이름을 지정합니다(예: ONTAP FlexVol, Element Volume, CVS 볼륨 경로). 스토리지 볼륨은 읽기/쓰기 액세스를 허용해야 하며 지정된 Astra Trident 백엔드에서 액세스할 수 있어야 합니다. '-f' 문자열 인수가 필요하며 YAML 또는 JSON PVC 파일의 경로를 지정합니다.

Astra Trident가 볼륨 가져오기 요청을 받으면 기존 볼륨 크기를 결정하고 PVC에 설정합니다. 스토리지 드라이버에서 볼륨을 가져온 후 PV는 PVC에 대한 ClaimRef를 사용하여 생성됩니다. 부가세 반환 청구액 정책은 당초 PV에서 '유지'로 설정되어 있습니다. Kubernetes에서 PVC 및 PV를 성공적으로 바인딩하면 스토리지 클래스의 부가세 반환 청구액 정책에 맞게 부가세 반환 청구액 정책이 업데이트됩니다. Storage Class의 Reclaim 정책이 'Delete'인 경우, PV 삭제

시 스토리지 볼륨이 삭제된다.

'--no-manage' 인수를 사용하여 볼륨을 가져올 때 Trident는 개체의 수명 주기 동안 PVC 또는 PV에 대한 추가 작업을 수행하지 않습니다. Trident는 `--no-manage' 객체의 PV 및 PVC 이벤트를 무시하므로 PV가 삭제되어도 스토리지 볼륨은 삭제되지 않습니다. 볼륨 클론 및 볼륨 크기 조정과 같은 다른 작업도 무시됩니다. 이 옵션은 컨테이너화된 워크로드에 Kubernetes를 사용하고, 그렇지 않고 Kubernetes 외부 스토리지 볼륨의 라이프사이클을 관리하려는 경우에 유용합니다.

PVC 및 PV에 주석이 추가되어 용적을 가져온 후 PVC와 PV가 관리되었는지 여부를 나타내는 두 가지 목적으로 사용됩니다. 이 주석은 수정하거나 제거할 수 없습니다.

Trident 19.07 이상 버전에서는 PVS 연결을 처리하고 볼륨을 가져오는 과정에서 볼륨을 마운트합니다. 이전 버전의 Astra Trident를 사용하여 가져오는 경우 데이터 경로에 작업이 없으며 볼륨 가져오기에서 볼륨을 마운트할 수 있는지 여부를 확인하지 않습니다. 볼륨 가져오기에서 오류가 발생한 경우(예: StorageClass가 올바르지 않은 경우) PV에 대한 부가세 반환 청구액 정책을 '유지'로 변경하고 PVC 및 PV를 삭제한 후 볼륨 가져오기 명령을 다시 시도하면 복구할 수 있습니다.

`ontap-nas` 및 `ontap-nas-flexgroup` 가져오기

ONTAP-NAS 드라이버로 생성된 각 볼륨은 ONTAP 클러스터의 FlexVol입니다. ONTAP-NAS 드라이버를 사용하여 FlexVol을 가져오는 작업은 동일합니다. ONTAP 클러스터에 이미 존재하는 FlexVol은 ONTAP-NAS PVC로 수입할 수 있다. 마찬가지로 FlexGroup vols는 ONTAP-NAS-Flexgroup PVC로 가져올 수 있습니다.

ONTAP 볼륨을 Trident에서 가져오려면 RW 유형이어야 합니다. 볼륨이 DP 유형인 경우 SnapMirror 대상 볼륨이므로 볼륨을 Trident로 가져오기 전에 미러 관계를 끊어야 합니다.

ONTAP-NAS 드라이버는 qtree를 가져오고 관리할 수 없습니다. ONTAP-NAS와 ONTAP-NAS-Flexgroup 드라이버는 중복 볼륨 이름을 허용하지 않습니다.

예를 들어, 이름이 "ONTAP_NAS"인 백엔드에서 이름이 managed_volume"인 볼륨을 가져오려면 다음 명령을 사용합니다.

```
$ tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Trident에서 관리하지 않을 'unmanaged_volume'('ONTAP_NAS 백엔드')이라는 볼륨을 가져오려면 다음 명령을 사용합니다.

```
$ tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file>
--no-manage

+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Trident는 '--no-manage' 인수를 사용할 때 볼륨의 이름을 바꾸거나 볼륨이 마운트되어 있는지 확인하지 않습니다. 볼륨이 수동으로 마운트되지 않은 경우 볼륨 가져오기 작업이 실패합니다.

> 사용자 지정 UnixPermissions를 사용하여 볼륨을 가져오는 기존 버그가 수정되었습니다. PVC 정의 또는 백엔드 구성에서 unixPermissions를 지정하고 Astra Trident에 볼륨을 가져오도록 지시할 수 있습니다.

`ontap-san` 가져오기

Astra Trident는 단일 LUN이 포함된 ONTAP SAN FlexVol도 가져올 수 있습니다. 이는 FlexVol 내 각 PVC와 LUN에 대해 FlexVol를 생성하는 ONTAP-SAN 드라이버와 일치합니다. 다른 경우와 마찬가지로 'tridentctl import' 명령을 사용할 수 있습니다.

- ONTAP-SAN 백엔드 이름을 포함합니다.
- 가져올 FlexVol의 이름을 입력합니다. 이 FlexVol에는 가져와야 하는 LUN이 하나만 포함되어 있습니다.
- '-f' 플래그와 함께 사용해야 하는 PVC 정의 경로를 제공합니다.
- PVC 관리 또는 비관리형 중에서 선택합니다. 기본적으로 Trident는 PVC를 관리하고 백엔드에서 FlexVol 및 LUN의 이름을 바꿉니다. 관리되지 않는 볼륨으로 가져오려면 '--no-manage' 플래그를 전달합니다.

> 관리되지 않는 'ONTAP-SAN' 볼륨을 가져올 때는 FlexVol의 LUN이 'LUN0'으로 명명되고 원하는 이니시에이터가 있는 igroup에 매핑되어 있는지 확인해야 합니다. Astra Trident에서 관리되는 가져오기를 위해 이 작업을 자동으로 처리합니다.

그러면 Astra Trident가 FlexVol를 가져와 PVC 정의와 연결합니다. Astra Trident는 FlexVol의 이름을 PVC-<uuid> 형식으로 바꾸고 FlexVol 내의 LUN은 LUN0으로 바꿉니다.

> 기존 활성 연결이 없는 볼륨을 가져오는 것이 좋습니다. 사용 중인 볼륨을 가져오려는 경우 먼저 볼륨을 클론한 다음 가져오기를 수행합니다.

예

ONTAP_SAN_DEFAULT 백엔드에 존재하는 ONTAP-SAN-MANDATED FlexVol를 가져오려면 tridentctl import

명령을 다음과 같이 실행합니다.

```
$ tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

> Astra Trident에서 가져오려면 ONTAP 볼륨이 RW 유형이어야 합니다. 볼륨이 DP 유형인 경우 SnapMirror 대상 볼륨이므로 볼륨을 Astra Trident로 가져오기 전에 미러 관계를 끊어야 합니다.

`element` 가져오기

Trident를 사용하여 NetApp Element 소프트웨어/NetApp HCI 볼륨을 Kubernetes 클러스터로 가져올 수 있습니다. "tridentctl import" 명령의 인수로 Astra Trident 백엔드의 이름과 볼륨의 고유 이름 및 PVC 파일이 필요합니다.

```
$ tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

> Element 드라이버는 중복 볼륨 이름을 지원합니다. 중복된 볼륨 이름이 있는 경우 Trident의 볼륨 가져오기 프로세스에서 오류가 반환됩니다. 이 문제를 해결하려면 볼륨을 복제하여 고유한 볼륨 이름을 제공합니다. 그런 다음 복제된 볼륨을 가져옵니다.

`gcp-cvs` 가져오기

> GCP에서 NetApp Cloud Volumes Service가 지원하는 볼륨을 가져오려면 해당 이름 대신 볼륨 경로를 기준으로 볼륨을 식별합니다.

adroit-jolly-swift 볼륨 경로를 사용하여 백엔드에서 gcpcvs_YEppr이라는 GCP-CV 볼륨을 가져오려면 다음 명령을 사용합니다.

```
$ tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-
pvc-file> -n trident

+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

> 볼륨 경로는 / 이후의 볼륨 내보내기 경로 부분입니다. 예를 들어, 내보내기 경로가 10.0.0.1:/adroit-jolly-swift인 경우 볼륨 경로는 매우 교묘한 -jolly-swift입니다.

`azure-netapp-files` 가져오기

볼륨 경로 importvol1을 사용하여 백엔드에서 azurenetappfiles_40517이라는 Azure-NetApp-files 볼륨을 가져오려면 다음 명령을 실행합니다.

```
$ tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

ANF 볼륨의 볼륨 경로는 다음:/ 이후의 마운트 경로에 있습니다. 예를 들어, 마운트 경로가 10.0.0.2:/importvol1 인 경우 볼륨 경로는 importvol1 입니다.

작업자 노드를 준비합니다

Kubernetes 클러스터의 모든 작업자 노드는 Pod용으로 프로비저닝된 볼륨을 마운트할 수 있어야 합니다. 백엔드 중 하나에 ONTAP-NAS, ONTAP-NAS-이코노미 또는 ONTAP-NAS-Flexgroup 드라이버를 사용하는 경우 작업자 노드에 NFS 툴이 필요합니다. 그렇지 않으면 iSCSI 도구가 필요합니다.

최신 버전의 RedHat CoreOS에는 기본적으로 NFS와 iSCSI가 모두 설치되어 있습니다.

NFS 또는 iSCSI 툴을 설치한 후에는 항상 작업자 노드를 재부팅해야 하며, 그렇지 않으면 컨테이너에 볼륨을 연결할 수 없습니다.

NFS 볼륨

프로토콜	운영 체제	명령
NFS 를 참조하십시오	RHEL/CentOS	'SUDO yum install -y nfs-utils'
NFS 를 참조하십시오	Ubuntu/Debian	'UDO apt-get install-y nfs-common'을 선택합니다

부팅 중에 NFS 서비스가 시작되었는지 확인해야 합니다.

iSCSI 볼륨

iSCSI 볼륨을 사용할 때는 다음 사항을 고려하십시오.

- Kubernetes 클러스터의 각 노드에는 고유한 IQN이 있어야 합니다. * 이것은 필수 전제 조건입니다 *.
- RHCOS 버전 4.5 이상을 사용하거나 RHEL 또는 CentOS 버전 8.2 이상을 'olidfire-SAN' 드라이버로 사용하는 경우, CHAP 인증 알고리즘이 '/etc/iscsi/iscsid.conf'에서 MD5로 설정되어 있는지 확인하십시오.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- iSCSI PVS에서 RHEL/RedHat CoreOS를 실행하는 작업자 노드를 사용하는 경우 StorageClass에서 discard 마운트 옵션을 지정하여 인라인 공간 재확보를 수행하십시오. 을 참조하십시오 "RedHat 설명서".

프로토콜	운영 체제	명령
iSCSI	RHEL/CentOS	1. 다음 시스템 패키지를 설치합니다. 'SUDO yum install -y lsscsi iscsi -initiator-utils sg3_utils device-mapper-multipath' 2. iscsi-initiator-utils 버전이 6.2.0.874-2.el7 이상인지 확인합니다. rpm -q iscsi-initiator-utils 3. 스캔을 수동으로 설정합니다. 'SUDO SED- I's/^\(node.session.scan\).*/\1 = manual/'/etc/iscsi/iscsid.conf' 4. 다중 경로 설정: 'SUDO mpathconf - -enable - -with_multipathd y - -find_multiprohs n' etc/multipath.conf에 debrofs 아래에 find_multiprohs no가 포함되어 있는지 확인합니다. 5. iscsid와 multipathd가 실행 중인지 확인합니다. 'SUDO systemctl enable — now iscsid multipathd' 6. "iSCSI" 활성화 및 시작: 'SUDO systemctl enable — now iscsi'

프로토콜	운영 체제	명령
iSCSI	Ubuntu/Debian	1. 다음 시스템 패키지를 설치합니다. 'SUDO apt-get install-y open-iscsi lsscsg3-utils multipath-tools scitools' 2. open-iscsi 버전이 2.0.874-5ubuntu2.10 이상(bionic) 또는 2.0.874-7.1uubuttu6.1 이상(focal)인지 확인합니다. d패키지-l open-iscsi 3. 스캔을 수동으로 설정합니다. 'SUDO SED-I's/^\(node.session.scan\).*/\1 = manual/'/etc/iscsi/iscsid.conf' 4. 다중 경로 설정: 'sudo tee /etc/multipath.conf←'EOF' 기본값 {user_friendly_names yes find_multipath no} EOF sudo systemctl enable—now multipath-tools.service sudo service multipath-tools restart'입니다 etc/multipath.conf에 debrofs 아래에 find_multiprohs no가 포함되어 있는지 확인합니다. 5. 'open-iscsi'와 'multirpath-tools'가 활성화되어 실행되고 있는지 확인합니다. 'SUDO systemctl status multipath -tools"SUDO systemctl enable — now open-iscsi.service'SUDO systemctl status open-iscsi'

Ubuntu 18.04의 경우 iSCSI 데몬이 시작되도록 "open-iscsi"를 시작하기 전에 iscsiadm"이 있는 타겟 포트를 검색해야 합니다. 또는 iSCSI 서비스를 수정하여 iscsid를 자동으로 시작할 수 있습니다.

베타 기능인 작업자 노드 자동 준비에 대한 자세한 내용은 을 참조하십시오 "여기".

작업자 노드 자동 준비

Astra Trident는 Kubernetes 클러스터에 있는 노드에 필요한 NFS 및 iSCSI 툴을 자동으로 설치할 수 있습니다. 이것은 * 베타 기능 * 이며 * 프로덕션 클러스터용으로 * 제공되지 않습니다. 현재 이 기능은 * CentOS, RHEL 및 Ubuntu * 를 실행하는 노드에 사용할 수 있습니다.

이 기능을 위해 Astra Trident는 tridentctl을 사용하여 배포된 설치를 위한 '--enable-node-prep'라는 새로운 설치 플래그를 포함합니다. Trident 연산자가 있는 배포의 경우 부울 옵션 "enableNodePrep"을 사용합니다.

'--enable-node-prep' 설치 옵션은 Astra Trident가 볼륨을 작업자 노드에 마운트할 때 NFS 및 iSCSI 패키지 및/또는 서비스가 실행 중인지 설치 및 확인하도록 지시합니다. 이 기능은 개발/테스트 환경에서 사용할 수 있도록 고안된 * 베타 기능 * 으로, 운영용으로 * 검증되지 않았습니다.

tridentctl로 배포된 Astra Trident 설치에 '--enable-node-prep' 플래그가 포함된 경우 다음 작업이 수행됩니다.

1. 설치의 일부로, Astra Trident가 실행되는 노드를 등록합니다.
2. 영구 볼륨 클레임(PVC) 요청이 이루어지면 Astra Trident는 관리하는 백엔드 중 하나에서 PV를 생성합니다.
3. POD에서 PVC를 사용하려면 POD가 실행되는 노드에 볼륨을 마운트하기 위해 Astra Trident가 필요합니다. Astra Trident는 필요한 NFS/iSCSI 클라이언트 유틸리티를 설치하고 필요한 서비스가 활성 상태인지 확인합니다. 볼륨이 마운트되기 전에 이 작업이 수행됩니다.

작업자 노드의 준비는 볼륨을 마운트하려는 첫 번째 시도의 일부로 한 번만 수행됩니다. Astra Trident 외부의 변경 사항이 NFS와 iSCSI 유틸리티에 닿지 않는 한 이후의 모든 볼륨 마운트는 성공해야 합니다.

이러한 방법으로 Astra Trident는 Kubernetes 클러스터의 모든 노드에 볼륨을 마운트 및 연결하는 데 필요한 유틸리티가 있는지 확인할 수 있습니다. NFS 볼륨의 경우 내보내기 정책에서도 볼륨을 마운트할 수 있어야 합니다. Trident는 백엔드별로 엑스포트 정책을 자동으로 관리할 수도 있고, 사용자는 대역 외 엑스포트 정책을 관리할 수도 있습니다.

Astra Trident를 모니터링합니다

Astra Trident는 Astra Trident의 성능을 모니터링하는 데 사용할 수 있는 Prometheus 메트릭 엔드포인트 세트를 제공합니다.

Astra Trident에서 제공하는 메트릭을 통해 다음을 수행할 수 있습니다.

- Astra Trident의 상태 및 구성을 계속 확인하십시오. 성공적인 작업이 어떻게 이루어지는지, 예상대로 백엔드와 통신할 수 있는지 확인할 수 있습니다.
- 백엔드 사용 정보를 검토하고 백엔드에서 프로비저닝되는 볼륨 수와 사용된 공간 등을 파악합니다.
- 사용 가능한 백엔드에 프로비저닝된 볼륨 양의 매핑을 유지합니다.
- 성과 추적. Astra Trident가 백엔드 및 작업을 수행하는 데 걸리는 시간을 확인할 수 있습니다.

기본적으로 Trident의 메트릭은 '/metrics' 끝점의 타겟 포트 8001에 표시됩니다. Trident가 설치된 경우 이러한 메트릭은 기본적으로 * 활성화됩니다.

필요한 것

- Astra Trident가 설치된 Kubernetes 클러스터
- 프로메테우스(Prometheus) 인스턴스. 이것은 일 수 있습니다 "컨테이너형 Prometheus 구축" 또는 Prometheus를로 실행하도록 선택할 수 있습니다 "네이티브 애플리케이션".

1단계: Prometheus 목표를 정의합니다

메트릭을 수집하고 백엔드 Astra Trident가 관리하는, 생성하는 볼륨 등에 대한 정보를 얻으려면 Prometheus 타겟을 정의해야 합니다. 여기 "블로그" Prometheus 및 Grafana를 Astra Trident와 함께 사용하여 메트릭을 검색하는 방법에 대해 설명합니다. 이 블로그에서는 Kubernetes 클러스터에서 운영자로 Prometheus를 실행하고 Astra Trident의 메트릭을 얻기 위해 ServiceMonitor를 생성하는 방법을 설명합니다.

2단계: Prometheus ServiceMonitor를 만듭니다

Trident 메트릭을 소비하려면 트리덴트 CSI 서비스를 감시하고 메티우스 포트에서 수신 대기하는 프로메테우스 서비스 모니터를 만들어야 합니다. 샘플 ServiceMonitor의 모양은 다음과 같습니다.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
      release: prom-operator
  spec:
    jobLabel: trident
    selector:
      matchLabels:
        app: controller.csi.trident.netapp.io
    namespaceSelector:
      matchNames:
      - trident
    endpoints:
    - port: metrics
      interval: 15s
```

이 ServiceMonitor 정의는 '삼중상 CSI' 서비스에서 반환된 지표를 검색하고 특히 서비스의 산부인과를 찾습니다. 따라서 이제 Prometheus가 Astra Trident의 메트릭을 이해하도록 구성되었습니다.

kubelet은 Astra Trident에서 직접 제공하는 측정 지표 외에도 자사의 측정 지표를 통해 많은 "kubelet_volume_ *" 측정 지표를 노출합니다. Kubelet는 연결된 볼륨, Pod 및 처리하는 기타 내부 작업에 대한 정보를 제공할 수 있습니다. 을 참조하십시오 "여기".

3단계: PromQL을 사용하여 Trident 메트릭 쿼리

PromQL은 시계열 또는 표 형식 데이터를 반환하는 식을 만드는 데 적합합니다.

다음은 사용할 수 있는 몇 가지 PromQL 쿼리입니다.

Trident 상태 정보를 가져옵니다

- Astra Trident** 의 HTTP 2XX 응답률

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- Astra Trident에서 상태 코드를 통해 얻은 REST 응답의 비율**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Astra Trident** 에 의해 수행된 작업의 평균 지속 시간(ms)

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Astra Trident 사용 정보를 확인하십시오

- 평균 볼륨 크기

```
trident_volume_allocated_bytes/trident_volume_count
```

- 각 백엔드에서 프로비저닝된 총 볼륨 공간

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

개별 볼륨 사용량을 가져옵니다

> 이 기능은 kubelet 메트릭도 수집한 경우에만 사용할 수 있습니다.

- 각 볼륨에 사용된 공간의 비율

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Astra Trident AutoSupport 텔레메트리 에 대해 자세히 알아보십시오

기본적으로 Astra Trident는 Prometheus 메트릭 및 기본 백엔드 정보를 매일 NetApp에 보냅니다.

- Astra Trident가 Prometheus 메트릭 및 기본 백엔드 정보를 NetApp에 전송하지 않도록 하려면 Astra Trident 설치 중에 '--st침묵-autosupport' 플래그를 전달하십시오.
- Astra Trident는 또한 "tridentctl Send AutoSupport"를 통해 요청 시 NetApp 지원으로 컨테이너 로그를 보낼 수 있습니다. 로그를 업로드하려면 Astra Trident를 트리거해야 합니다. 로그를 제출하기 전에 NetApp의 내용에 동의해야 합니다https://www.netapp.com/company/legal/privacy-policy/["개인 정보 보호 정책"^].
- 지정되지 않은 경우 Astra Trident는 지난 24시간 동안 로그를 가져옵니다.
- '--since' 플래그를 사용하여 로그 보존 기간을 지정할 수 있습니다. 예: 'tridentctl send AutoSupport --since=1h' 이 정보는 Astra Trident와 함께 설치되는 트리덴트 자동 지원 컨테이너를 통해 수집 및 전송됩니다. 에서 컨테이너 이미지를 얻을 수 있습니다 "Trident AutoSupport를 누릅니다".
- Trident AutoSupport는 개인 식별 정보(PII) 또는 개인 정보를 수집하거나 전송하지 않습니다. 이 제품은 와 함께 제공됩니다 "EULA" Trident 컨테이너 이미지 자체에는 적용되지 않습니다. 데이터 보안 및 신뢰에 대한 NetApp의 노력에 대해 자세히 알아볼 수 있습니다 "여기".

Astra Trident에서 보낸 페이로드의 예는 다음과 같습니다.

```
{
  "items": [
    {
      "backendUUID": "ff3852e1-18a5-4df4-b2d3-f59f829627ed",
      "protocol": "file",
      "config": {
        "version": 1,
        "storageDriverName": "ontap-nas",
        "debug": false,
        "debugTraceFlags": null,
        "disableDelete": false,
        "serialNumbers": [
          "nwkvzfanek_SN"
        ],
        "limitVolumeSize": ""
      },
      "state": "online",
      "online": true
    }
  ]
}
```

- AutoSupport 메시지는 NetApp의 AutoSupport 엔드포인트로 전송됩니다. 개인 레지스트리를 사용하여 컨테이너 이미지를 저장하는 경우 '--image-registry' 플래그를 사용할 수 있습니다.
- 또한 설치 YAML 파일을 생성하여 프록시 URL을 구성할 수도 있습니다. 이는 트라이덴트ctl install --generate -custom-YAML을 이용해 YAML 파일을 생성하고 트리덴트 배포(trident-deployment)의 트리덴트 자동 지원 컨테이너에 대한 '--proxy-url' 주장을 추가하는 방식으로 가능하다.

Astra Trident 메트릭을 비활성화합니다

- 메트릭을 보고하지 않으려면 ('--generate-custom-YAML' 플래그를 사용하여) 사용자 지정 YAML을 생성하고 이를 편집하여 삼중류-main' 컨테이너에 대해 호출되는 '--metrics' 플래그를 제거해야 합니다.

저작권 정보

Copyright © 2024 NetApp, Inc. All Rights Reserved. 미국에서 인쇄된 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

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

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

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

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

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

상표 정보

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