Trident 사용

Trident

NetApp
July 01, 2026

목차

Trident 사용

작업자 노드를 준비합니다

Kubernetes 클러스터의 모든 워커 노드는 Pod에 프로비저닝된 볼륨을 마운트할 수 있어야 합니다. 워커 노드를 준비하려면 선택한 드라이버에 따라 NFS, iSCSI, NVMe/TCP 또는 FC 도구를 설치해야 합니다.

적절한 도구 선택

여러 드라이버를 조합해서 사용하는 경우, 해당 드라이버에 필요한 모든 도구를 설치해야 합니다. 최신 버전의 Red Hat Enterprise Linux CoreOS(RHCOS)에는 이러한 도구가 기본적으로 설치되어 있습니다.

NFS 도구

"NFS 툴을 설치합니다"다음을 사용하는 경우: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` 또는 `azure-netapp-files`.

iSCSI 도구

"iSCSI 도구를 설치합니다"다음을 사용하는 경우: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

NVMe 도구

"NVMe 툴을 설치합니다"TCP(NVMe/TCP) 프로토콜을 통한 NVMe(Nonvolatile Memory Express)에 `ontap-san`를 사용하는 경우.

> NetApp에서는 NVMe/TCP에 ONTAP 9.12 이상을 권장합니다.

FC를 통한 **SCSI** 툴

FC 및 FC-NVMe SAN 호스트 구성에 대한 자세한 내용은 "FC 및 FC-NVMe SAN 호스트를 구성하는 방법"를 참조하십시오.

"FC 툴을 설치합니다" `ontap-san`sanType `fcp`(FC를 통한 SCSI)을 사용하는 경우.

고려 사항: * OpenShift 및 KubeVirt 환경에서는 FC를 통한 SCSI가 지원됩니다. * Docker에서는 FC를 통한 SCSI가 지원되지 않습니다. * iSCSI 자체 복구 기능은 FC를 통한 SCSI에는 적용되지 않습니다.

SMB 도구

"SMB 볼륨 프로비저닝 준비" 다음을 사용하는 경우: `ontap-nas` SMB 볼륨을 프로비저닝합니다.

노드 서비스 검색

Trident는 노드가 iSCSI 또는 NFS 서비스를 실행할 수 있는지 자동으로 감지하려고 시도합니다.

> 노드 서비스 검색은 발견된 서비스를 식별하지만 서비스가 올바르게 구성되었음을 보장하지는 않습니다. 반대로, 검색된 서비스가 없다고 해서 볼륨 마운트가 반드시 실패하는 것은 아닙니다.

이벤트 검토

Trident는 검색된 서비스를 식별하기 위해 노드에 대한 이벤트를 생성합니다. 이러한 이벤트를 검토하려면 다음을 실행합니다.

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

검색된 서비스 검토

Trident는 Trident 노드 CR에서 각 노드에 대해 활성화된 서비스를 식별합니다. 검색된 서비스를 보려면 다음을 실행합니다.

```
tridentctl get node -o wide -n <Trident namespace>
```

NFS 볼륨

운영 체제에 맞는 명령을 사용하여 NFS 툴을 설치합니다. 부팅 시 NFS 서비스가 시작되는지 확인합니다.

RHEL 8+

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

NFS 도구를 설치한 후 워커 노드를 재부팅하여 컨테이너에 볼륨을 연결할 때 발생하는 오류를 방지하십시오.

iSCSI 볼륨

Trident는 iSCSI 세션을 자동으로 설정하고, LUN을 스캔하고, 다중 경로 디바이스를 검색하고, 포맷하고, 포드에 마운트할 수 있습니다.

iSCSI 자가 복구 기능

ONTAP 시스템의 경우 Trident는 다음을 위해 5분마다 iSCSI 자체 복구를 실행합니다.

1. 원하는 iSCSI 세션 상태와 현재 iSCSI 세션 상태를 *식별*하십시오.
2. 원하는 상태와 현재 상태를 비교하여 필요한 수리를 파악합니다. Trident는 수리 우선순위와 수리를 선제적으로 진행해야 할 시점을 결정합니다.
3. 현재 iSCSI 세션 상태를 원하는 iSCSI 세션 상태로 되돌리기 위해 필요한 *복구*를 수행합니다.

자가 복구 활동에 대한 로그는 해당 Daemonset Pod의 `trident-main` 컨테이너에 있습니다. 로그를 보려면 Trident 설치 시 `debug`를 "true"로 설정해야 합니다.

Trident iSCSI 자가 복구 기능은 다음을 방지하는 데 도움이 될 수 있습니다.

- 네트워크 연결 문제 발생 후 발생할 수 있는 오래되었거나 불안정한 iSCSI 세션. 오래된 세션의 경우 Trident는 로그아웃하기 전에 7분 동안 기다린 후 포털과의 연결을 다시 설정합니다.

  예를 들어 스토리지 컨트롤러에서 CHAP 암호가 교체되고 네트워크 연결이 끊어지면 이전(만료된) CHAP 암호가 계속 남아 있을 수 있습니다. 자가 복구 기능은 이러한 상황을 감지하고 자동으로 세션을 재설정하여 업데이트된 CHAP 암호를 적용할 수 있습니다.

- iSCSI 세션 누락
- 누락된 LUN

Trident 업그레이드 전 고려 사항

- 노드별 igroup(23.04 이상 버전에서 도입)만 사용하는 경우 iSCSI 자체 복구 기능은 SCSI 버스의 모든 디바이스에 대해 SCSI 재검색을 시작합니다.
- 백엔드 범위 igroup(23.04부터 더 이상 사용되지 않음)만 사용 중인 경우 iSCSI 자가 복구는 SCSI 버스의 정확한 LUN ID에 대해 SCSI 재검색을 시작합니다.
- 노드별 igroup과 백엔드 범위 igroup이 혼합되어 사용되는 경우 iSCSI 자체 복구는 SCSI 버스에서 정확한 LUN ID에 대한 SCSI 재검색을 시작합니다.

iSCSI 도구를 설치합니다

운영 체제에 맞는 명령을 사용하여 iSCSI 툴을 설치합니다.

시작하기 전에

- Kubernetes 클러스터의 각 노드는 고유한 IQN을 가져야 합니다. 이는 필수 조건입니다.
- RHCOS 버전 4.5 이상 또는 기타 RHEL 호환 Linux 배포판을 `solidfire-san` 드라이버 및 Element OS 12.5 이하와 함께 사용하는 경우 `/etc/iscsi/iscsid.conf`에서 CHAP 인증 알고리즘이 MD5로 설정되어 있는지 확인하십시오. 안전한 FIPS 규격 준수 CHAP 알고리즘인 SHA1, SHA-256 및 SHA3-256은 Element 12.7에서 사용할 수 있습니다.

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

- RHEL/Red Hat Enterprise Linux CoreOS (RHCOS)를 실행하는 워커 노드에서 iSCSI PV를 사용할 때, 인라인 공간 재확보를 수행하려면 `discard` mountOption을 StorageClass에 지정하십시오. 을 참조하십시오. "Red Hat 문서"
- `multipath-tools`의 최신 버전으로 업그레이드했는지 확인하십시오.

RHEL 8+

1. 다음 시스템 패키지를 설치하십시오.

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-
multipath
```

2. iscsi-initiator-utils 버전이 6.2.0.874-2.el7 이상인지 확인하십시오.

```
rpm -q iscsi-initiator-utils
```

3. 스캔을 수동으로 설정:

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

4. 다중 경로 지정 활성화:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> `/etc/multipath.conf`에 `find_multipaths no`이(가) `defaults` 아래에 포함되어 있는지 확인하십시오.

5. `iscsid` 및 `multipathd`가 실행 중인지 확인하십시오.

```
sudo systemctl enable --now iscsid multipathd
```

6. 활성화 및 시작 `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. 다음 시스템 패키지를 설치하십시오.

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. open-iscsi 버전이 2.0.874-5ubuntu2.10 이상(bionic의 경우) 또는 2.0.874-7.1ubuntu6.1 이상(focal의 경우)인지 확인하십시오.

```
dpkg -l open-iscsi
```

3. 스캔을 수동으로 설정:

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

4. 다중 경로 지정 활성화:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> ⓘ `/etc/multipath.conf`에 `find_multipaths no`이(가) `defaults` 아래에 포함되어 있는지 확인하십시오.

5. `open-iscsi` 및 `multipath-tools`가 활성화되어 실행 중인지 확인하십시오.

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

> ⓘ Ubuntu 18.04의 경우, iSCSI 데몬이 시작되기 전에 `iscsiadm`를 사용하여 대상 포트를 검색해야 합니다 `open-iscsi`. 또는 `iscsi` 서비스를 수정하여 `iscsid`가 자동으로 시작되도록 할 수 있습니다.

iSCSI 자가 복구 구성 또는 비활성화

다음 Trident iSCSI 자체 복구 설정을 구성하여 오래된 세션을 수정할 수 있습니다.

- **iSCSI** 자가 복구 간격: iSCSI 자가 복구가 실행되는 빈도를 결정합니다(기본값: 5분). 이 값을 작게 설정하면 더 자주 실행되도록 구성할 수 있고, 크게 설정하면 덜 자주 실행되도록 구성할 수 있습니다.

> ⓘ iSCSI 자체 복구 간격을 0으로 설정하면 iSCSI 자체 복구가 완전히 중지됩니다. iSCSI 자체 복구를 비활성화하는 것은 권장하지 않으며, iSCSI 자체 복구가 의도한 대로 작동하지 않거나 디버깅 목적으로 필요한 특정 시나리오에서만 비활성화해야 합니다.

- **iSCSI** 자가 복구 대기 시간: iSCSI 자가 복구 기능이 비정상 세션에서 로그아웃한 후 다시 로그인을 시도하기 전에 대기하는 시간을 결정합니다(기본값: 7분). 비정상으로 식별된 세션이 로그아웃된 후 다시 로그인을 시도하기 전에 더 오래 대기하도록 더 큰 값으로 구성하거나, 더 일찍 로그아웃하고 로그인하도록 더 작은 값으로 구성할 수 있습니다.

Helm

iSCSI 자체 복구 설정을 구성하거나 변경하려면 helm 설치 또는 helm 업데이트 중에 `iscsiSelfHealingInterval` 및 `iscsiSelfHealingWaitTime` 매개변수를 전달하십시오.

다음 예는 iSCSI 자체 복구 간격을 3분으로, 자체 복구 대기 시간을 6분으로 설정합니다.

```
helm install trident trident-operator-100.2602.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

iSCSI 자체 복구 설정을 구성하거나 변경하려면 tridentctl 설치 또는 업데이트 중에 `iscsi-self-healing-interval` 및 `iscsi-self-healing-wait-time` 매개변수를 전달하십시오.

다음 예는 iSCSI 자체 복구 간격을 3분으로, 자체 복구 대기 시간을 6분으로 설정합니다.

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

NVMe/TCP 볼륨

사용 중인 운영 체제에 맞는 명령을 사용하여 NVMe 툴을 설치하십시오.

- NVMe에는 RHEL 9 이상이 필요합니다.
- Kubernetes 노드의 커널 버전이 너무 오래되었거나 해당 커널 버전에 맞는 NVMe 패키지를 사용할 수 없는 경우 노드의 커널 버전을 NVMe 패키지가 포함된 버전으로 업데이트해야 할 수 있습니다.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

설치 확인

설치 후 다음 명령을 사용하여 Kubernetes 클러스터의 각 노드에 고유한 NQN이 있는지 확인하십시오.

```
cat /etc/nvme/hostnqn
```

Trident는 `ctrl_device_tmo` 값을 수정하여 경로가 다운되더라도 NVMe가 경로를 포기하지 않도록 합니다. 이 설정을 변경하지 마십시오.

FC를 통한 SCSI 볼륨

이제 Trident와 함께 파이버 채널(FC) 프로토콜을 사용하여 ONTAP 시스템의 스토리지 리소스를 프로비저닝하고 관리할 수 있습니다.

필수 구성 요소

FC에 필요한 네트워크 및 노드 설정을 구성하십시오.

네트워크 설정

1. 대상 인터페이스의 WWPN을 가져오십시오. 자세한 내용은 "네트워크 인터페이스 show"을 참조하십시오.
2. 이니시에이터(호스트)의 인터페이스에 대한 WWPN을 가져옵니다.

   해당 호스트 운영 체제 유틸리티를 참조하십시오.
3. 호스트 및 타겟의 WWPN을 사용하여 FC 스위치에서 조닝을 구성합니다.

   자세한 내용은 해당 스위치 공급업체 설명서를 참조하십시오.

   자세한 내용은 다음 ONTAP 문서를 참조하십시오.

   - "Fibre Channel 및 FCoE 조닝 개요"

- "FC 및 FC-NVMe SAN 호스트를 구성하는 방법"

FC 툴을 설치합니다

운영 체제에 맞는 명령을 사용하여 FC 툴을 설치합니다.

- FC PV를 사용하는 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS) 워커 노드를 사용할 때, 인라인 공간 재확보를 수행하려면 `discard` StorageClass에서 mountOption을 지정하십시오. 을 참조하십시오. "Red Hat 문서"

RHEL 8+

1. 다음 시스템 패키지를 설치하십시오.

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. 다중 경로 지정 활성화:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> `/etc/multipath.conf`에 `find_multipaths no`이(가) `defaults` 아래에 포함되어 있는지 확인하십시오.

3. `multipathd`이(가) 실행 중인지 확인하십시오.

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. 다음 시스템 패키지를 설치하십시오.

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. 다중 경로 지정 활성화:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> `/etc/multipath.conf`에 `find_multipaths no`이(가) `defaults` 아래에 포함되어 있는지 확인하십시오.

3. `multipath-tools`이(가) 활성화되어 실행 중인지 확인하십시오.

```
sudo systemctl status multipath-tools
```

SMB 볼륨 프로비저닝 준비

`ontap-nas` 드라이버를 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다.

ONTAP 온프레미스 클러스터용 ontap-nas-economy SMB 볼륨을 생성하려면 SVM에서 NFS 및 SMB/CIFS 프로토콜을 모두 구성해야 합니다. 이러한 프로토콜 중 하나라도 구성하지 않으면 SMB 볼륨 생성이 실패합니다.

`autoExportPolicy`는 SMB 볼륨에서 지원되지 않습니다.

시작하기 전에

SMB 볼륨을 프로비저닝하기 전에 다음 사항을 충족해야 합니다.

- Linux 컨트롤러 노드와 Windows Server 2022를 실행하는 하나 이상의 Windows 워커 노드로 구성된 Kubernetes 클러스터입니다. Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Active Directory 자격 증명이 포함된 Trident 비밀 키가 하나 이상 필요합니다. 비밀 키를 생성하려면 smbcreds:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Windows 서비스로 구성된 CSI 프록시입니다. `csi-proxy`를 구성하려면 Windows에서 실행되는 Kubernetes 노드에 대한 "GitHub: CSI 프록시" 또는 "GitHub: Windows용 CSI 프록시"를 참조하십시오.

단계

1. 온프레미스 ONTAP의 경우 선택적으로 SMB 공유를 생성하거나 Trident가 대신 생성해 줄 수 있습니다.

   Amazon FSx for ONTAP에는 SMB 공유가 필요합니다.

   SMB 관리자 공유는 "Microsoft Management Console" 공유 폴더 스냅인을 사용하거나 ONTAP CLI를 사용하는 두 가지 방법으로 생성할 수 있습니다. ONTAP CLI를 사용하여 SMB 공유를 생성하려면 다음을 수행합니다.

   a. 필요한 경우 공유에 대한 디렉터리 경로 구조를 생성하십시오.

      `vserver cifs share create` 명령은 공유 생성 시 -path 옵션에 지정된 경로를 확인합니다. 지정된 경로가 존재하지 않으면 명령이 실패합니다.

   b. 지정된 SVM과 연결된 SMB 공유를 생성합니다.

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

c. 공유가 생성되었는지 확인합니다.

```
vserver cifs share show -share-name share_name
```

자세한 내용은 "SMB 공유 생성"를 참조하십시오.

2. 백엔드를 생성할 때 SMB 볼륨을 지정하려면 다음을 구성해야 합니다. 모든 FSx for ONTAP 백엔드 구성 옵션에 대한 자세한 내용은 "FSx for ONTAP 구성 옵션 및 예"를 참조하십시오.

매개변수	설명	예
`smbShare`	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console 또는 ONTAP CLI를 사용하여 생성한 SMB 공유의 이름, Trident가 SMB 공유를 생성할 수 있도록 허용하는 이름, 또는 볼륨에 대한 공통 공유 액세스를 방지하려면 매개 변수를 비워 둘 수 있습니다. 이 매개 변수는 온프레미스 ONTAP의 경우 선택 사항입니다. 이 매개 변수는 Amazon FSx for ONTAP 백엔드에 필요하며 비워 둘 수 없습니다.	`smb-share`
`nasType`	* `smb`로 설정해야 합니다.* null인 경우 기본값은 `nfs`입니다.	`smb`
`securityStyle`	새 볼륨에 대한 보안 스타일입니다. **SMB** 볼륨의 경우 **`ntfs`** 또는 **`mixed`**로 설정해야 합니다.	`ntfs` 또는 `mixed` SMB 볼륨의 경우
`unixPermissions`	새 볼륨 모드입니다. **SMB** 볼륨의 경우 비워 두어야 합니다.	""

백엔드 구성 및 관리

백엔드 구성

백엔드는 Trident와 스토리지 시스템 간의 관계를 정의합니다. 백엔드는 Trident가 해당 스토리지 시스템과 통신하는 방법과 Trident가 해당 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다.

Trident는 스토리지 클래스에 정의된 요구 사항과 일치하는 백엔드에서 스토리지 풀을 자동으로 제공합니다. 스토리지 시스템의 백엔드를 구성하는 방법을 알아보십시오.

- "Azure NetApp Files 백엔드 구성"
- "Google Cloud NetApp Volumes 백엔드 구성"
- "NetApp HCI 또는 SolidFire 백엔드 구성"
- "ONTAP 또는 Cloud Volumes ONTAP NAS 드라이버로 백엔드 구성"
- "ONTAP 또는 Cloud Volumes ONTAP SAN 드라이버로 백엔드 구성"
- "Amazon FSx for NetApp ONTAP와 함께 Trident 사용"

Azure NetApp Files

Azure NetApp Files 백엔드 구성

Azure NetApp Files를 Trident의 백엔드로 사용합니다. 이 백엔드는 NFS 및 SMB 볼륨을 지원합니다. Trident는 Azure Kubernetes Service(AKS) 클러스터에 대한 관리 ID 및 워크로드 ID를 지원합니다.

지원되는 Azure 클라우드 환경

Trident는 다양한 Azure 클라우드 환경에서 Azure NetApp Files 백엔드를 지원합니다.

지원되는 Azure 클라우드는 다음과 같습니다.

- Azure Commercial
- Azure Government(Azure Government/MAG)

Trident를 배포하거나 Azure NetApp Files 백엔드를 구성할 때 Azure Resource Manager 및 인증 엔드포인트가 Azure 클라우드 환경과 일치하는지 확인하십시오.

Azure NetApp Files 드라이버 지원 검토

Trident는 다음과 같은 Azure NetApp Files 스토리지 드라이버를 제공합니다.

지원되는 액세스 모드에는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(RWX) 및 *ReadWriteOncePod*(RWOP)가 포함됩니다.

드라이버	프로토콜	**volumeMode**	지원되는 액세스 모드	지원되는 파일 시스템
`azure-netapp-files`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	`nfs`, `smb`

검토 시 고려 사항

- Azure NetApp Files는 50GiB보다 작은 볼륨을 지원하지 않습니다. 더 작은 볼륨을 요청하면 Trident는 50GiB 볼륨을 생성합니다.
- Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- 비상업용 Azure 클라우드에 Azure NetApp Files를 배포하려면 클라우드별 Azure Resource Manager 및 인증 엔드포인트가 필요합니다. Trident와 모든 백엔드 구성이 Azure 클라우드 환경에 적합한 엔드포인트를 사용하는지 확인하십시오.

AKS에 관리형 ID 사용

Trident는 AKS 클러스터에 대한 "관리 ID"를 지원합니다.

`tridentctl`를 사용하여 `Azure NetApp Files` 백엔드를 생성하거나 관리하는 경우 올바른 `Azure` 클라우드 환경에 대해 구성되어 있는지 확인하십시오.

관리형 ID를 사용하려면 다음이 필요합니다.

- AKS를 사용하여 배포된 Kubernetes 클러스터
- AKS Kubernetes 클러스터에 구성된 관리 ID
- `cloudProvider`이(가) `"Azure"`(으)로 설정된 상태로 Trident가 설치되었습니다

Trident 운영자

`tridentorchestrator_cr.yaml`을 편집하고 `cloudProvider`를 `"Azure"`로 설정하세요.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

Helm

다음 예시는 Trident를 설치하고 환경 변수 `cloudProvider`를 사용하여 `$CP`를 설정합니다:

```
helm install trident trident-operator-100.2602.0.tgz --create-namespace
--namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

다음 예제는 Trident를 설치하고 `cloud-provider` 플래그를 `Azure`로 설정합니다:

```
tridentctl install --cloud-provider="Azure" -n trident
```

AKS에 워크로드 ID 사용

워크로드 ID를 사용하면 Kubernetes Pod가 워크로드 ID로 인증하여 Azure 리소스에 액세스할 수 있습니다.

`tridentctl`를 사용하여 `Azure NetApp Files` 백엔드를 생성하거나 관리하는 경우 올바른 `Azure` 클라우드 환경에 대해 구성되어 있는지 확인하십시오.

워크로드 ID를 사용하려면 다음이 필요합니다.

- AKS를 사용하여 배포된 Kubernetes 클러스터
- AKS Kubernetes 클러스터에 구성된 워크로드 ID 및 oidc-issuer
- Trident `cloudProvider "Azure" `cloudIdentity`워크로드 ID 값으로 설정하여 설치되었습니다.

Trident 운영자

`tridentorchestrator_cr.yaml`을 편집하고 `cloudProvider`를 `"Azure"`로 설정하세요. `cloudIdentity`를 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`로 설정하세요.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

Helm

다음 환경 변수를 사용하여 **cloud-provider(CP)** 및 **cloud-identity(CI)** 플래그 값을 설정하십시오.

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx'"
```

다음 예제는 Trident를 설치하고 `cloudProvider`를 사용하여 `$CP`를 설정하고 `cloudIdentity`를 사용하여 `$CI`를 설정합니다:

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>tridentctl</code>

다음 환경 변수를 사용하여 **cloud provider** 및 **cloud identity** 플래그 값을 설정하십시오.

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

다음 예제는 Trident를 설치하고 `cloud-provider`을 `$CP`로, `cloud-identity`을 `$CI`로 설정합니다:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Azure NetApp Files 백엔드 구성을 준비하세요

Azure NetApp Files 백엔드를 구성하기 전에 다음 요구 사항을 충족해야 합니다.

지원되는 **Azure** 클라우드 환경

Trident는 다양한 Azure 클라우드 환경에서 Azure NetApp Files 백엔드를 지원합니다.

지원되는 Azure 클라우드는 다음과 같습니다.

- Azure Commercial
- Azure Government(Azure Government/MAG)

환경을 준비할 때 Azure 구독, ID 구성 및 Azure NetApp Files 리소스가 적절한 Azure 클라우드 환경에 생성되었는지 확인하십시오.

NFS 및 **SMB** 볼륨의 사전 요구 사항

Azure NetApp Files를 처음 사용하거나 새 위치에서 사용하는 경우 Azure NetApp Files를 설정하고 NFS 볼륨을 생성하기 위해 몇 가지 초기 구성이 필요합니다. 을 참조하십시오. "Azure: Azure NetApp Files 설정 및 NFS 볼륨 생성"

```
https://azure.microsoft.com/en-us/products/netapp/["Azure NetApp Files"^]
백엔드를 구성하고 사용하려면 다음이 필요합니다.
```

- `subscriptionID`, `tenantID`, `clientID`, `location` 및 `clientSecret`는 AKS 클러스터에서 관리 ID를 사용할 때 선택 사항입니다.
- `tenantID`, `clientID` 및 `clientSecret`는 AKS 클러스터에서 클라우드 ID를 사용할 때 선택 사항입니다.
- 비상업용 Azure 클라우드에 Azure NetApp Files를 배포하려면 클라우드별 Azure Resource Manager 및 인증 엔드포인트가 필요합니다. Trident와 모든 백엔드 구성이 Azure 클라우드 환경에 적합한 엔드포인트를 사용하는지 확인하십시오.

- 용량 풀. 다음을 참조하십시오. "Microsoft: Azure NetApp Files용 용량 풀 생성".
- Azure NetApp Files에 위임된 서브넷입니다. 다음을 참조하십시오. "Microsoft: Azure NetApp Files에 서브넷 위임".
- `subscriptionID`Azure NetApp Files가 활성화된 Azure 구독에서 가져온 것입니다.
- `tenantID`, `clientID` 및 `clientSecret`는 Azure NetApp Files 서비스에 대한 충분한 권한이 있는 Azure Active Directory의 "앱 등록"에서 가져와야 합니다. 앱 등록은 다음 중 하나를 사용해야 합니다.
  - 소유자 또는 기여자 역할"Azure에서 미리 정의됨".
  - 구독 수준에서 "사용자 지정 Contributor 역할"(`assignableScopes`Trident에 필요한 권한으로만 제한된 다음 권한을 가진 사용자 지정 역할입니다. 사용자 지정 역할을 생성한 후 "Azure 포털을 사용하여 역할을 할당합니다".

사용자 지정 기여자 역할

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
```

```
ions/delete",
          "Microsoft.Features/features/read",
          "Microsoft.Features/operations/read",
          "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
        ],
        "notActions": [],
        "dataActions": [],
        "notDataActions": []
      }
    ]
  }
}
```

- `location`하나 이상의 "위임된 서브넷"를 포함하는 `Azure`입니다. `Trident 22.01`부터 `location` 매개 변수는 백엔드 구성 파일의 최상위 수준에서 필수 필드입니다. 가상 풀에 지정된 위치 값은 무시됩니다.
- `Cloud Identity`를 사용하려면, `client ID`를 "사용자 할당 관리 ID"에서 가져와서 해당 ID를 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`에 지정하십시오.

SMB 볼륨에 대한 추가 요구 사항

SMB 볼륨을 생성하려면 다음이 필요합니다.

- Active Directory가 구성되어 Azure NetApp Files에 연결되었습니다. 다음을 참조하십시오. "Microsoft: Azure NetApp Files용 Active Directory 연결 생성 및 관리".
- Linux 컨트롤러 노드와 Windows Server 2022를 실행하는 하나 이상의 Windows 워커 노드로 구성된 Kubernetes 클러스터입니다. Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Azure NetApp Files가 Active Directory에 인증할 수 있도록 Active Directory 자격 증명이 포함된 Trident 암호를 하나 이상 생성해야 합니다. 비밀 키를 생성하려면 `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Windows 서비스로 구성된 CSI 프록시입니다. `csi-proxy`를 구성하려면 Windows에서 실행되는 Kubernetes 노드에 대한 "GitHub: CSI 프록시" 또는 "GitHub: Windows용 CSI 프록시"를 참조하십시오.

Azure NetApp Files 백엔드 구성 옵션 및 예

Azure NetApp Files의 NFS 및 SMB 백엔드 구성 옵션에 대해 알아보고 구성 예제를 살펴보세요.

백엔드 **configuration** 옵션

Trident는 백엔드 구성(서브넷, 가상 네트워크, 서비스 수준, 위치)을 사용하여 요청된 위치에서 사용 가능하고 요청된 서비스 수준 및 서브넷과 일치하는 용량 풀에 Azure NetApp Files 볼륨을 생성합니다.

Azure NetApp Files 백엔드는 이러한 구성 옵션을 제공합니다.

매개변수	설명	기본값
`version`	백엔드 구성 버전.	항상 1
`storageDriverName`	스토리지 드라이버의 이름	"azure-netapp-files"
`backendName`	스토리지 백엔드의 사용자 지정 이름	드라이버 이름 + "_" + 임의의 문자
`subscriptionID`	Azure 구독의 구독 ID AKS 클러스터에서 관리 ID가 활성화된 경우 선택 사항입니다.	
`tenantID`	AKS 클러스터에서 관리형 ID 또는 클라우드 ID를 사용하는 경우 앱 등록의 테넌트 ID는 선택 사항입니다.	
`clientID`	AKS 클러스터에서 관리형 ID 또는 클라우드 ID를 사용하는 경우 앱 등록의 클라이언트 ID는 선택 사항입니다.	
`clientSecret`	AKS 클러스터에서 관리형 ID 또는 클라우드 ID를 사용하는 경우 앱 등록의 클라이언트 암호는 선택 사항입니다.	
`serviceLevel`	`Standard`, `Premium` 또는 `Ultra` 중 하나	"" (무작위)
`location`	새 볼륨이 생성될 Azure 위치의 이름입니다. AKS 클러스터에서 관리 ID가 활성화된 경우 선택 사항입니다.	
`resourceGroups`	검색된 리소스를 필터링하기 위한 리소스 그룹 목록	"[]" (필터 없음)
`netappAccounts`	검색된 리소스를 필터링하기 위한 NetApp 계정 목록	"[]" (필터 없음)
`capacityPools`	검색된 리소스를 필터링하기 위한 용량 풀 목록	"[]" (필터 없음, 무작위)
`virtualNetwork`	위임된 서브넷이 있는 가상 네트워크의 이름	""
`subnet`	에 위임된 서브넷의 이름 `Microsoft.Netapp/volumes`	""

매개변수	설명	기본값
`networkFeatures`	볼륨에 대한 VNet 기능 집합은 `Basic` 또는 `Standard`일 수 있습니다. 네트워크 기능은 모든 지역에서 사용할 수 없으며 구독에서 활성화해야 할 수도 있습니다. 기능이 활성화되지 않은 상태에서 `networkFeatures`을(를) 지정하면 볼륨 프로비저닝이 실패합니다.	""
`nfsMountOptions`	NFS 마운트 옵션을 세밀하게 제어할 수 있습니다. SMB 볼륨에는 적용되지 않습니다. NFS 버전 4.1을 사용하여 볼륨을 마운트하려면 쉼표로 구분된 마운트 옵션 목록에 `nfsvers=4`를 포함하여 NFS v4.1을 선택합니다. 스토리지 클래스 정의에 설정된 마운트 옵션은 백엔드 구성에 설정된 마운트 옵션을 재정의합니다.	"nfsvers=3"
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다.	"" (기본적으로 적용되지 않음)
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예 `\{"api": false, "method": true, "discovery": true}`. 문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마십시오.	null
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 `nfs`, `smb` 또는 null입니다. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	`nfs`
`supportedTopologies`	이 백엔드에서 지원하는 지역 및 영역 목록을 나타냅니다. 자세한 내용은 "CSI 토폴로지 사용"을 참조하십시오.	
`qosType`	QoS 유형(자동 또는 수동)을 나타냅니다.	자동
`maxThroughput`	최대 허용 처리량을 MiB/sec 단위로 설정합니다. 수동 QoS 용량 풀에서만 지원됩니다.	`4 MiB/sec`

네트워크 기능에 대한 자세한 내용은 "Azure NetApp Files 볼륨에 대한 네트워크 기능 구성"을(를) 참조하십시오.

Azure 클라우드 환경 고려**(26.02)**

26.02 릴리스부터 Trident는 여러 Azure 클라우드 환경에서 Azure NetApp Files 백엔드를 생성하고 관리하는 것을 지원합니다.

지원되는 Azure 클라우드는 다음과 같습니다.

- Azure Commercial
- Azure Government(Azure Government/MAG)

Trident를 배포하거나 Azure NetApp Files 백엔드를 생성할 때 Azure Resource Manager 및 인증 엔드포인트가 Azure 클라우드 환경과 일치하는지 확인하십시오. 엔드포인트가 일치하지 않으면 `tridentctl` 인증할 수 없으며 백엔드 생성이 실패합니다.

필요한 권한 및 리소스

PVC를 생성할 때 "No capacity pools found" 오류가 발생하는 경우, 앱 등록에 필요한 권한 및 리소스(서브넷, 가상 네트워크, 용량 풀)가 연결되어 있지 않을 가능성이 높습니다. 디버그 모드가 활성화된 경우, Trident는 백엔드 생성 시 검색된 Azure 리소스를 로그에 기록합니다. 적절한 역할이 사용되고 있는지 확인하십시오.

`resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` 및 `subnet`의 값은 단축 이름 또는 정규화된 이름을 사용하여 지정할 수 있습니다. 단축 이름은 동일한 이름을 가진 여러 리소스와 일치할 수 있으므로 대부분의 경우 정규화된 이름을 사용하는 것이 좋습니다.

vNet이 Azure NetApp Files(ANF) 스토리지 계정과 다른 리소스 그룹에 있는 경우 백엔드에 대한 resourceGroups 목록을 구성할 때 가상 네트워크의 리소스 그룹을 지정합니다.

`resourceGroups`, `netappAccounts` 및 `capacityPools` 값은 검색된 리소스 집합을 이 스토리지 백엔드에서 사용 가능한 리소스로 제한하는 필터이며, 어떤 조합으로든 지정할 수 있습니다. 정규화된 이름은 다음 형식을 따릅니다.

유형	형식
리소스 그룹	<resource group>
NetApp 계정	<resource group>/<netapp account>
용량 풀	<resource group>/<netapp account>/<capacity pool>
가상 네트워크	<resource group>/<virtual network>
서브넷	<resource group>/<virtual network>/<subnet>

볼륨 프로비저닝

구성 파일의 특수 섹션에서 다음 옵션을 지정하여 기본 볼륨 프로비저닝을 제어할 수 있습니다. 자세한 내용은 예시 구성를 참조하십시오.

매개변수	설명	기본값
`exportRule`	새 볼륨에 대한 내보내기 규칙입니다. `exportRule`는 CIDR 표기법으로 IPv4 주소 또는 IPv4 서브넷을 쉼표로 구분한 목록이어야 합니다. SMB 볼륨에는 적용되지 않습니다.	"0.0.0.0/0"

매개변수	설명	기본값
`snapshotDir`	`.snapshot` 디렉토리에 대한 액세스	`true`, `false` (명시적으로 설정).
`size`	새 볼륨의 기본 크기	"100G"
`unixPermissions`	새 볼륨의 Unix 권한(8진수 4자리). SMB 볼륨에는 적용되지 않습니다.	"" (미리보기 기능이며, 구독 시 화이트리스트 등록이 필요합니다.)

예시 구성

다음 예시들은 대부분의 매개변수를 기본값으로 유지하는 기본 구성을 보여줍니다. 이것이 백엔드를 정의하는 가장 쉬운 방법입니다.

최소 구성

이는 최소한의 백엔드 구성입니다. 이 구성을 사용하면 Trident는 구성된 위치에서 Azure NetApp Files에 위임된 모든 NetApp 계정, 용량 풀 및 서브넷을 검색하고 해당 풀과 서브넷 중 하나에 새 볼륨을 무작위로 배치합니다. `nasType`이 생략되었으므로 `nfs` 기본값이 적용되어 백엔드에서 NFS 볼륨을 프로비저닝합니다.

이 구성은 Azure NetApp Files를 처음 시작하고 기능을 테스트할 때 이상적이지만, 실제로는 프로비저닝하는 볼륨에 대한 추가 범위 지정이 필요할 것입니다.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  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
```

AKS용 관리 ID

이 백엔드 구성에서는 관리 ID를 사용할 때 선택 사항인 `subscriptionID`, `tenantID`, `clientID` 및 `clientSecret`가 생략되었습니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/eastus-anf-subnet
```

AKS용 클라우드 ID

이 백엔드 구성은 클라우드 ID를 사용할 때 선택 사항인 `tenantID`, `clientID` 및 `clientSecret`를 생략합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

용량 풀 필터를 사용한 특정 서비스 수준 구성

이 백엔드 구성은 Azure의 `eastus` 위치에 있는 `Ultra` 용량 풀에 볼륨을 배치합니다. Trident는 해당 위치에서 Azure NetApp Files에 위임된 모든 서브넷을 자동으로 검색하고 그중 하나에 새 볼륨을 임의로 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
```

수동 **QoS** 용량 풀을 사용한 백엔드 예

이 백엔드 구성은 수동 QoS 용량 풀을 사용하여 Azure의 `eastus` 위치에 볼륨을 배치합니다.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

고급 구성

이 백엔드 구성은 볼륨 배치 범위를 단일 서브넷으로 더욱 축소하고 일부 볼륨 프로비저닝 기본값을 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: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/my-subnet
networkFeatures: Standard
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"
```

가상 풀 구성

이 백엔드 구성은 단일 파일에 여러 스토리지 풀을 정의합니다. 이는 서로 다른 서비스 수준을 지원하는 여러 용량 풀이 있고 이를 나타내는 스토리지 클래스를 Kubernetes에 생성하려는 경우에 유용합니다. 가상 풀 레이블은 `performance`을 기반으로 풀을 구분하는 데 사용되었습니다.

```
---
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
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
  - labels:
      performance: gold
    serviceLevel: Ultra
    capacityPools:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

지원되는 토폴로지 구성

Trident는 리전 및 가용 영역을 기반으로 워크로드용 볼륨 프로비저닝을 지원합니다. 이 백엔드 구성의 `supportedTopologies` 블록은 백엔드별 리전 및 영역 목록을 제공하는 데 사용됩니다. 여기에 지정된 리전 및 영역 값은 각 Kubernetes 클러스터 노드의 레이블에 있는 리전 및 영역 값과 일치해야 합니다. 이러한 리전 및 영역은 스토리지 클래스에 제공될 수 있는 허용 값 목록을 나타냅니다. 백엔드에 제공된 리전 및 영역의 하위 집합을 포함하는 스토리지 클래스의 경우 Trident는 언급된 리전 및 영역에 볼륨을 생성합니다. 자세한 내용은 "CSI 토폴로지 사용"을 참조하십시오.

```
---
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
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

스토리지 클래스 정의

다음 `StorageClass` 정의는 위의 스토리지 풀을 참조합니다.

`parameter.selector` 필드를 사용한 예시 정의

`parameter.selector`을 사용하면 각 `StorageClass`에 대해 볼륨을 호스팅하는 데 사용되는 가상 풀을 지정할 수 있습니다. 볼륨은 선택한 풀에 정의된 속성을 갖게 됩니다.

```
---
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
```

SMB 볼륨에 대한 정의 예

`nasType`, `node-stage-secret-name` 및 `node-stage-secret-namespace`를 사용하면 SMB 볼륨을 지정하고 필요한 Active Directory 자격 증명을 제공할 수 있습니다.

기본 네임스페이스의 기본 구성

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

네임스페이스별로 서로 다른 시크릿 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

볼륨마다 다른 시크릿 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb` SMB 볼륨을 지원하는 풀에 대한 필터입니다.
`nasType: nfs` 또는 `nasType: null` NFS 풀에 대한 필터입니다.

백엔드 생성

백엔드 구성 파일을 생성한 후 다음 명령을 실행하십시오.

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

상용이 아닌 Azure 클라우드를 사용하는 경우 `tridentctl`이(가) Azure 클라우드 환경에 대한 Azure Resource Manager 및 인증 엔드포인트를 사용하도록 구성되어 있는지 확인하십시오. 백엔드 생성이 실패하면 백엔드 구성을 확인하고 로그를 검토하여 원인을 파악하십시오.

```
tridentctl logs
```

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

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes 구성

Google Cloud NetApp Volumes를 Trident의 백엔드로 구성하여 Kubernetes 워크로드용 스토리지를 프로비저닝할 수 있습니다.

개요

Trident는 NAS(NFS 및 SMB)와 블록(iSCSI) 워크로드 모두에 대해 Google Cloud NetApp Volumes를 지원합니다.

- NAS 워크로드는 `google-cloud-netapp-volumes` 백엔드를 사용합니다
- 블록(iSCSI) 워크로드는 `google-cloud-netapp-volumes-san` 백엔드를 사용합니다.

NAS 볼륨은 파일 기반 스토리지를 제공하며 NFS 또는 SMB 프로토콜을 사용하여 액세스합니다. 이러한 볼륨은 여러 파드 또는 노드에서 공유 액세스를 지원합니다.

블록 볼륨은 원시 블록 스토리지를 제공하며 Kubernetes 노드에 연결된 iSCSI 장치로 액세스됩니다. 이러한 볼륨은 애플리케이션에 블록 수준 액세스가 필요한 경우에 사용됩니다.

이는 다음 환경에 적용됩니다.

- Trident 26.02 이상
- Google Kubernetes Engine(GKE) 또는 Red Hat OpenShift
- Google Cloud NetApp Volumes 스토리지 풀

블록(iSCSI) 스토리지를 구성하려면 "블록 스토리지(iSCSI) 구성"을(를) 참조하십시오.

Cloud ID를 사용하면 Kubernetes 워크로드가 정적 자격 증명을 사용하는 대신 워크로드 ID로 인증하여 Google Cloud 리소스에 액세스할 수 있습니다.

Google Cloud NetApp Volumes에서 클라우드 ID를 사용하려면 다음이 필요합니다.

- Google Kubernetes Engine(GKE)을 사용하여 배포된 Kubernetes 클러스터
- GKE 클러스터에서 워크로드 ID가 활성화되고 노드 풀에서 메타데이터 서버가 활성화되었습니다
- Google Cloud NetApp Volumes 관리자 역할(`roles/netapp.admin`이 있는 Google Cloud 서비스 계정 또는 이에 상응하는 사용자 지정 역할
- Trident가 클라우드 공급자를 `GCP`로 설정하고 클라우드 ID 주석을 구성한 상태로 설치됨

Trident 운영자

Trident 오퍼레이터를 사용하여 Trident를 설치하려면 다음을 편집하십시오 `tridentorchestrator_cr.yaml`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  cloudProvider: "GCP"
  cloudIdentity: "iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

Helm

Helm을 사용하여 Trident를 설치할 때 클라우드 공급자와 클라우드 ID를 설정하십시오.

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

tridentctl

클라우드 공급자와 클라우드 ID를 지정하여 Trident를 설치하십시오.

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
  -n trident
```

NAS 스토리지 구성

Google Cloud NetApp Volumes UNIFIED 스토리지 풀의 경우, Trident는 볼륨 작업 중에 UNIFIED 전용 명명 및 검증 규칙을 적용합니다.

볼륨을 찾을 때 Trident는 여러 호환 가능한 볼륨 이름 변형(예: 하이픈 및 밑줄 형식)을 평가하여 가져오기 및 검색 신뢰성을 향상시킬 수 있습니다.

드라이버 세부 정보

Trident는 `google-cloud-netapp-volumes` Google Cloud NetApp Volumes에서 NAS 스토리지를

프로비저닝하는 드라이버를 제공합니다.

드라이버는 다음과 같은 액세스 모드를 지원합니다.

- ReadWriteOnce(RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)
- ReadWriteOncePod(RWOP)

드라이버	프로토콜	**volumeMode**	지원되는 액세스 모드	지원되는 파일 시스템
`google-cloud-netapp-volumes`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Trident NAS 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

NAS 볼륨 프로비저닝

NAS 볼륨은 `google-cloud-netapp-volumes` 백엔드를 사용하여 프로비저닝되며 NFS 및 SMB 프로토콜을 지원합니다.

NFS 볼륨용 **StorageClass**

NFS 볼륨을 프로비저닝하려면 `nasType`을(를) `nfs`(으)로 설정하십시오.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: true
```

SMB 볼륨용 StorageClass

SMB 볼륨을 프로비저닝하려면 `nasType`을(를) `smb`(으)로 설정하고 자격 증명을 제공하십시오.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
allowVolumeExpansion: true
```

PersistentVolumeClaim 예시(RWX)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

PersistentVolumeClaim 예시(RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

NAS 볼륨은 `volumeMode: Filesystem`을 사용합니다.

SAN 워크로드를 위해 Google Cloud NetApp Volumes 구성

Google Cloud NetApp Volumes에서 iSCSI 프로토콜을 사용하여 블록 스토리지 볼륨을 프로비저닝하도록 Trident를 구성할 수 있습니다. SAN 볼륨은 `google-cloud-netapp-volumes-san` 스토리지 드라이버를 사용하여 Flex Unified 스토리지 풀에서 프로비저닝됩니다.

이 드라이버는 블록 워크로드 전용이며 NAS 프로토콜을 지원하지 않습니다.

`google-cloud-netapp-volumes-san` 백엔드는 iSCSI 블록 볼륨을 프로비저닝하는 데 필요합니다. `google-cloud-netapp-volumes` 백엔드는 NAS 프로토콜만 지원하며 SAN 워크로드에는 사용할 수 없습니다.

개요

Trident는 `google-cloud-netapp-volumes-san` 드라이버를 사용하여 Google Cloud NetApp Volumes SAN(iSCSI) 워크로드를 지원합니다.

SAN 볼륨은 Flex Unified 스토리지 풀에서 프로비저닝되어 iSCSI 블록 디바이스로 Kubernetes 노드에 제공됩니다.

이는 다음 환경에 적용됩니다.

- Trident 26.02 이상
- Google Kubernetes Engine(GKE) 또는 Red Hat OpenShift
- Google Cloud NetApp Volumes Flex 통합 스토리지 풀
- iSCSI 기반 워크로드

Flex Unified 스토리지 풀

Flex Unified 스토리지 풀은 iSCSI 프로토콜을 사용하여 블록 스토리지를 제공하며 SAN 프로비저닝에 필요합니다.

- Flex Unified REGIONAL 풀이 지원됩니다.
- Flex Unified ZONAL 풀은 Trident 26.02.1부터 지원됩니다.
- SAN 워크로드의 경우 **Flex** 서비스 레벨만 지원됩니다.

Trident SAN 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

StorageClass 생성

SAN 백엔드를 구성한 후 `google-cloud-netapp-volumes-san` 드라이버를 참조하는 StorageClass를 생성합니다.

파일 시스템 유형은 백엔드가 아닌 StorageClass에서 정의됩니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

지원되는 파일 시스템 유형:

- `ext4` (기본값)

- `ext3`
- `xfs`

> SAN 드라이버는 Flex 서비스 레벨만 지원하며 `exportRule`, `unixPermissions`, `nasType`, `snapshotDir`, `nfsMountOptions`와 같은 NAS 관련 백엔드 매개변수나 티어링 관련 설정은 사용하지 않습니다.

블록 볼륨 프로비저닝

ReadWriteOnce(RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadWriteOncePod(RWOP)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadOnlyMany (ROX)

ROX에서 흔히 사용되는 패턴은 기존 ReadWriteOnce 볼륨을 복제하고 복제본을 읽기 전용으로 마운트하는 것입니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

ReadWriteMany (RWX) — 원시 블록 전용

ReadWriteMany는 `volumeMode: Block`인 경우에만 지원됩니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

블록 볼륨 동작

블록 볼륨은 iSCSI LUN으로 프로비저닝되고 Kubernetes 노드에 블록 디바이스로 제공됩니다.

블록 볼륨:

- iSCSI 프로토콜 사용
- 파일 시스템 및 원시 블록 프레젠테이션 지원
- Trident에 의해 연결되고 관리됩니다
- 여러 Kubernetes 액세스 모드 지원

액세스 모드

Trident에서 프로비저닝한 블록 볼륨은 다음과 같은 액세스 모드를 지원합니다.

- `ReadWriteOnce` (RWO)
- `ReadOnlyMany` (ROX)
- `ReadWriteOncePod` (RWOP)
- `ReadWriteMany` (RWX), 다음과 같은 경우에만 지원됩니다 `volumeMode: Block`

volumeMode 동작

`volumeMode` 필드는 블록 볼륨이 노출되는 방식을 제어합니다.

- `Filesystem` Trident가 볼륨을 포맷하고 마운트합니다.
- `Block` Trident는 장치를 연결하고 이를 원시 블록 장치로 노출합니다.

지원되는 작업

`google-cloud-netapp-volumes-san` 드라이버를 사용하여 프로비저닝된 블록 볼륨은 다음을 지원합니다:

- 생성
- 삭제
- 클론
- 스냅샷
- 크기 조정
- 가져오기

추가 **GiB** 오버프로비저닝 동작

Google Cloud NetApp Volumes 블록 볼륨에는 내부 메타데이터 오버헤드가 포함됩니다. 이 오버헤드로 인해 커널에서 인식되는 장치 크기가 프로비저닝된 용량보다 작아집니다.

테스트 결과:

- 초기 생성 시 약 300 KiB의 오버헤드가 발생합니다
- 크기 조정 후 최대 약 107MiB의 오버헤드

Google Cloud NetApp Volumes는 GiB 단위의 할당만 허용하므로 Trident는 다음과 같은 방법으로 사용 가능한 장치 크기가 항상 PVC 요청을 충족하거나 초과하도록 합니다.

- 요청된 크기를 다음 정수 GiB로 올림합니다
- 추가 1 GiB 버퍼 추가

예:

- PVC 요청: 100 GiB
- Google Cloud NetApp Volumes에 프로비저닝된 크기: 101 GiB
- 애플리케이션에 표시되는 사용 가능한 공간: 최소 100 GiB

Pod 예

파일 시스템 마운트 블록 볼륨(RWO)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

원시 블록 디바이스(RWX)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

연결 및 마운트 동작

Google Cloud NetApp Volumes에서 프로비저닝된 SAN 볼륨의 경우:

- Trident는 Flex Unified 스토리지 풀에 LUN(논리 유닛 번호)을 생성합니다.
- 게시 중에 Trident는 LUN을 노드별 호스트 그룹에 매핑합니다.
- 노드 스테이징 중에 Trident는 다음을 수행합니다.
  - iSCSI 타겟에 로그인합니다
  - LUN을 검색합니다
  - 다중 경로를 구성합니다
- `volumeMode: Filesystem`Trident는 필요한 경우 장치를 포맷하고 마운트합니다.
- `volumeMode: Block`인 경우 Trident는 포맷이나 마운트 없이 장치를 연결하고 Pod에 직접 노출합니다.

> SAN 블록 볼륨은 분산 잠금 또는 쓰기 조정을 제공하지 않습니다. 여러 노드에서 블록 볼륨에 액세스할 때(ReadWriteMany 사용 시 `volumeMode: Block`) 애플리케이션 또는 파일 시스템이 동시성을 관리해야 합니다.

Google Cloud NetApp Volumes 백엔드 구성을 준비하세요

Google Cloud NetApp Volumes 백엔드를 구성하기 전에 다음 요구 사항을 충족해야 합니다.

NFS 또는 **SMB** 볼륨의 사전 요구 사항

Google Cloud NetApp Volumes를 처음 사용하거나 새 위치에서 사용하는 경우 Google Cloud NetApp Volumes를 설정하고 NFS 또는 SMB 볼륨을 생성하려면 몇 가지 초기 구성이 필요합니다. 다음을 참조하십시오. "시작하기 전에".

Google Cloud NetApp Volumes 백엔드를 구성하기 전에 다음 사항을 확인하십시오.

- Google Cloud NetApp Volumes 서비스로 구성된 Google Cloud 계정. 다음을 참조하십시오. "Google Cloud NetApp Volumes".
- Google Cloud 계정의 프로젝트 번호입니다. 다음을 참조하십시오. "프로젝트 식별".
- NetApp Volumes Admin (`roles/netapp.admin` 역할을 가진 Google Cloud 서비스 계정 다음을 참조하십시오. "Identity and Access Management 역할 및 권한".
- GCNV 계정용 API 키 파일입니다. 다음을 참조하십시오. "서비스 계정 키 생성"
- 스토리지 풀. 다음을 참조하십시오. "스토리지 풀 개요".

Google Cloud NetApp Volumes에 대한 액세스 설정 방법에 대한 자세한 내용은 "Google Cloud NetApp Volumes에 대한 액세스 설정"을 참조하십시오.

Google Cloud NetApp Volumes 백엔드 구성 옵션 및 예시

Google Cloud NetApp Volumes의 백엔드 구성 옵션에 대해 알아보고 구성 예제를 검토하십시오.

백엔드 **configuration** 옵션

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

매개변수	설명	기본값
`version`		항상 1
`storageDriverName`	스토리지 드라이버의 이름	`storageDriverName`의 값은 "google-cloud-netapp-volumes"로 지정해야 합니다.
`backendName`	(선택 사항) 스토리지 백엔드의 사용자 지정 이름	드라이버 이름 + "_" + API key의 일부
`storagePools`	볼륨 생성을 위한 스토리지 풀을 지정하는 데 사용되는 선택적 매개 변수입니다.	
`projectNumber`	Google Cloud 계정 프로젝트 번호입니다. 이 값은 Google Cloud 포털 홈페이지에서 확인할 수 있습니다.	
`location`	Trident가 GCNV 볼륨을 생성하는 Google Cloud 위치입니다. 교차 리전 Kubernetes 클러스터를 생성할 때 `location`에서 생성된 볼륨은 여러 Google Cloud 리전의 노드에서 예약된 워크로드에 사용할 수 있습니다. 교차 리전 트래픽에는 추가 비용이 발생합니다.	

매개변수	설명	기본값
apiKey	netapp.admin 역할이 있는 Google Cloud 서비스 계정의 API 키입니다. 여기에는 Google Cloud 서비스 계정의 개인 키 파일의 JSON 형식 내용이 포함됩니다(백엔드 구성 파일에 그대로 복사됨). apiKey`에는 다음 키에 대한 키-값 쌍이 포함되어야 합니다: `type, project_id, client_email, client_id, auth_uri, token_uri, auth_provider_x509_cert_url 및 client_x509_cert_url.	
nfsMountOptions	NFS 마운트 옵션을 세밀하게 제어할 수 있습니다.	"nfsvers=3"
limitVolumeSize	요청된 볼륨 크기가 이 값보다 크면 프로비저닝이 실패합니다.	"" (기본적으로 적용되지 않음)
serviceLevel	스토리지 풀 및 해당 볼륨의 서비스 수준입니다. 값은 flex, standard, premium 또는 `extreme`입니다.	
labels	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
network	GCNV 볼륨에 사용되는 Google Cloud 네트워크입니다.	
debugTraceFlags	문제 해결 시 사용할 디버그 플래그입니다. 예 {"api":false, "method":true}. 문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마십시오.	null
nasType	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 nfs, smb 또는 null입니다. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	nfs
supportedTopologies	이 백엔드에서 지원하는 지역 및 영역 목록을 나타냅니다. 자세한 내용은 "CSI 토폴로지 사용"을 참조하십시오. 예를 들면 다음과 같습니다. supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a	

볼륨 프로비저닝 옵션

구성 파일의 defaults 섹션에서 기본 볼륨 프로비저닝을 제어할 수 있습니다.

매개변수	설명	기본값
exportRule	새 볼륨에 대한 내보내기 규칙입니다. IPv4 주소의 모든 조합을 쉼표로 구분한 목록이어야 합니다.	"0.0.0.0/0"
snapshotDir	.snapshot 디렉토리에 대한 액세스	true, false (기본 동작은 다를 수 있습니다. 명시적으로 설정) NFSv3의 경우 "false"
snapshotReserve	스냅샷용으로 예약된 볼륨의 비율	""(기본값 0 적용)

매개변수	설명	기본값
`unixPermissions`	새 볼륨의 Unix 권한(8진수 4자리).	""

예시 구성

다음 예시들은 대부분의 매개변수를 기본값으로 유지하는 기본 구성을 보여줍니다. 이것이 백엔드를 정의하는 가장 쉬운 방법입니다.

최소 구성

이는 최소한의 백엔드 구성입니다. 이 구성을 사용하면 Trident는 구성된 위치에서 Google Cloud NetApp Volumes에 위임된 모든 스토리지 풀을 검색하고 새 볼륨을 해당 풀 중 하나에 무작위로 배치합니다. `nasType`이 생략되었으므로 `nfs` 기본값이 적용되어 백엔드에서 NFS 볼륨을 프로비저닝합니다.

이 구성은 Google Cloud NetApp Volumes를 처음 사용하고 테스트할 때 이상적이지만, 실제로는 프로비저닝하는 볼륨에 대해 추가적인 범위 지정이 필요할 수 있습니다.

`<id_value>` 및 `<key_value>`를 서비스 계정 자격 증명으로 바꾸십시오.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

SMB 볼륨 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    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/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

StoragePools 필터를 사용한 구성

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

가상 풀 구성

이 백엔드 구성은 단일 파일에 여러 개의 가상 풀을 정의합니다. 가상 풀은 `storage` 섹션에 정의됩니다. 여러 개의 스토리지 풀이 서로 다른 서비스 수준을 지원하고, 이를 나타내는 스토리지 클래스를 Kubernetes에 생성하려는 경우에 유용합니다. 가상 풀 레이블은 풀을 구분하는 데 사용됩니다. 예를 들어, 아래 예시에서는 `performance` 레이블과 `serviceLevel` 유형을 사용하여 가상 풀을 구분합니다.

모든 가상 풀에 적용할 기본값을 설정하고 개별 가상 풀의 기본값을 재정의할 수도 있습니다. 다음 예에서는 `snapshotReserve`와 `exportRule`가 모든 가상 풀의 기본값으로 사용됩니다.

자세한 내용은 "가상 풀"을 참조하십시오.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
```

```
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

GKE용 클라우드 ID

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

지원되는 토폴로지 구성

Trident는 리전 및 가용 영역을 기반으로 워크로드용 볼륨 프로비저닝을 지원합니다. 이 백엔드 구성의 `supportedTopologies` 블록은 백엔드별 리전 및 영역 목록을 제공하는 데 사용됩니다. 여기에 지정된 리전 및 영역 값은 각 Kubernetes 클러스터 노드의 레이블에 있는 리전 및 영역 값과 일치해야 합니다. 이러한 리전 및 영역은 스토리지 클래스에 제공될 수 있는 허용 값 목록을 나타냅니다. 백엔드에 제공된 리전 및 영역의 하위 집합을 포함하는 스토리지 클래스의 경우 Trident는 언급된 리전 및 영역에 볼륨을 생성합니다. 자세한 내용은 "CSI 토폴로지 사용"을 참조하십시오.

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

다음 단계

백엔드 구성 파일을 생성한 후 다음 명령을 실행하십시오.

```
kubectl create -f <backend-file>
```

백엔드가 성공적으로 생성되었는지 확인하려면 다음 명령을 실행하십시오.

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. `kubectl get tridentbackendconfig <backend-name>` 명령을 사용하여 백엔드를 설명하거나 다음 명령을 실행하여 로그를 보고 원인을 확인할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 식별하고 수정한 후 백엔드를 삭제하고 생성 명령을 다시 실행할 수 있습니다.

스토리지 클래스 정의

다음은 위의 백엔드를 참조하는 기본 `StorageClass` 정의입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

- `parameter.selector` 필드를 사용한 예시 정의:*

  `parameter.selector`을 사용하면 각 `StorageClass`에 대해 볼륨을 호스팅하는 데 사용되는 xref:{relative_path}../trident-concepts/virtual-storage-pool.html["가상 풀"]을 지정할 수 있습니다. 볼륨은 선택한 풀에 정의된 속성을 갖게 됩니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

스토리지 클래스에 대한 자세한 내용은 "스토리지 클래스를 생성합니다"을(를) 참조하십시오.

SMB 볼륨에 대한 정의 예

`nasType`, `node-stage-secret-name` 및 `node-stage-secret-namespace`를 사용하여 SMB 볼륨을 지정하고 필요한 Active Directory 자격 증명을 제공할 수 있습니다. 노드 단계 비밀에는 권한이 있거나 없는 모든 Active Directory 사용자/암호를 사용할 수 있습니다.

기본 네임스페이스의 기본 구성

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

네임스페이스별로 서로 다른 시크릿 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

볼륨마다 다른 시크릿 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb` SMB 볼륨을 지원하는 풀에 대한 필터입니다. `nasType: nfs` 또는 `nasType: null` NFS 풀에 대한 필터입니다.

PVC 정의 예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

PVC가 바인딩되었는지 확인하려면 다음 명령을 실행합니다.

```
kubectl get pvc gcnv-nfs-pvc

NAME           STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc   Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213   100Gi
RWX            gcnv-nfs-sc  1m
```

Google Cloud NetApp Volumes에 대한 자동 계층화 구성

자동 계층화는 볼륨 프로비저닝 중에 Trident 백엔드 매개변수와 PersistentVolumeClaim 주석을 통해 구성됩니다. Trident를 사용하여 Google Cloud NetApp Volumes에 대한 자동 계층화를 구성할 수 있습니다.

개요

자동 계층화 기능을 통해 Trident는 비활성 데이터를 성능 계층에서 용량 계층으로 자동으로 이동시키는 볼륨을 프로비저닝할 수 있습니다. 이를 통해 스토리지 비용을 절감하는 동시에 자주 액세스하는 데이터의 성능을 유지할 수 있습니다.

Trident는 볼륨 생성 시에만 자동 계층화 설정을 적용합니다. Trident 26.02에서는 프로비저닝 후 변경이 지원되지 않습니다.

개념

자동 계층화

자동 계층화는 접근 패턴에 따라 접근 빈도가 낮은 데이터를 성능 계층에서 용량 계층으로 이동시킵니다. 데이터 이동은 비동기적으로 발생하며 즉시 이루어지지 않습니다.

계층화 정책

계층화 정책은 볼륨에 대해 자동 계층화를 활성화할지 여부를 결정합니다.

다음 정책이 지원됩니다. * `auto`: 액세스 패턴에 따른 자동 계층화 활성화 * `none`: 자동 계층화 비활성화

냉방 일수

냉각일은 데이터 블록이 계층화 대상이 되기 전에 비활성 상태로 유지되어야 하는 최소 일수를 지정합니다. 냉각일은 계층화 정책이 `auto`로 설정된 경우에만 적용됩니다.

구성 모델

구성 범위

자동 계층화는 여러 범위에서 구성할 수 있습니다.

- 스토리지 풀 범위 풀에서 프로비저닝된 모든 볼륨에 적용됩니다.
- 볼륨 범위 PersistentVolumeClaim 주석을 통해 단일 볼륨에 적용됩니다.

Trident는 각 설정이 정의된 위치를 기반으로 유효한 구성을 결정합니다.

구성 우선 순위

동일한 설정이 여러 범위에서 정의된 경우 Trident는 다음 우선순위를 적용합니다.

1. PersistentVolumeClaim 주석
2. Trident 백엔드 구성
3. 스토리지 풀 기본값

우선 순위가 높은 설정이 하위 수준 값을 재정의합니다.

Trident 26.02에서 지원되는 기능

Trident 26.02는 Google Cloud NetApp Volumes에 대해 다음과 같은 자동 계층화 기능을 지원합니다.

- 볼륨 프로비저닝 중 자동 계층화 설정 또는 해제
- Trident 백엔드 구성에서 계층화 정책 정의
- PVC 주석을 사용하여 볼륨별 계층화 정책 및 냉각 일수 재정의
- 자동 계층화가 활성화된 볼륨에 대한 냉각 일수 구성

Trident 26.02에서 지원되지 않는 기능

다음 작업은 지원되지 않습니다.

- 볼륨 생성 후 자동 계층화 설정 수정
- Kubernetes 업데이트를 사용하여 기존 볼륨의 계층화 정책 변경
- Trident에서 관리하는 프로비저닝 워크플로 외부에서 자동 계층화 설정 적용

백엔드 구성 매개 변수

다음 매개변수는 Trident 백엔드 구성에 정의될 때 자동 계층화 동작을 제어합니다.

매개변수	필수	설명
tieringPolicy	아니요	볼륨에 대한 계층화 정책 (`auto` 또는 `none`)
tieringMinimumCoolingDays	아니요	데이터가 계층화되기 전의 비활성 일수(범위: 2-183, 기본값: 31)

PersistentVolumeClaim 어노테이션을 사용한 볼륨 수준 재정의

지원되는 주석

PersistentVolumeClaim 어노테이션을 사용하면 볼륨별로 자동 계층화 설정을 재정의할 수 있습니다.

주석	설명
`trident.netapp.io/tieringPolicy`	볼륨에 대한 티어링 정책을 재정의합니다
`trident.netapp.io/tieringMinimumCoolingDays`	볼륨에 대한 쿨링 일수 값을 재정의합니다

예시: 자동 계층화 재정의가 적용된 **PersistentVolumeClaim**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

동작 및 제한 사항

프로비저닝 동작

- 자동 계층화 설정은 볼륨 생성 시에만 평가 및 적용됩니다.
- Trident는 프로비저닝 후 계층화 구성을 조정하지 않습니다.
- 계층화 정책이 `none`로 설정된 경우 쿨링 기간은 무시됩니다.

플랫폼 제한 사항

- 자동 계층화는 NAS 볼륨(NFS 및 SMB)에서만 지원됩니다.
- 블록 볼륨(iSCSI)은 자동 계층화를 지원하지 않습니다.
- Google Cloud NetApp Volumes 스토리지 풀은 Google Cloud에서 자동 계층화가 활성화되어 있어야 합니다.

지원되는 값

- `tieringMinimumCoolingDays`의 유효 범위: 2~183
- 기본값: 31

NetApp HCI 또는 SolidFire 백엔드 구성

Trident 설치 환경에서 Element 백엔드를 생성하고 사용하는 방법을 알아보십시오.

Element 드라이버 세부 정보

Trident는 `solidfire-san` 스토리지 드라이버를 제공하여 클러스터와 통신합니다. 지원되는 액세스 모드는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(RWX), *ReadWriteOncePod*(RWOP)입니다.

> `solidfire-san` 스토리지 드라이버는 _file_ 및 _block_ 볼륨 모드를 지원합니다. `Filesystem` volumeMode의 경우 Trident는 볼륨을 생성하고 파일 시스템을 생성합니다. 파일 시스템 유형은 StorageClass에 의해 지정됩니다.

드라이버	프로토콜	VolumeMode	지원되는 액세스 모드	지원되는 파일 시스템
`solidfire-san`	iSCSI	블록	RWO, ROX, RWX, RWOP	파일 시스템이 없습니다. 원시 블록 장치입니다.
`solidfire-san`	iSCSI	파일 시스템	RWO, RWOP	`xfs`, `ext3`, `ext4`

시작하기 전에

Element 백엔드를 생성하기 전에 다음 사항이 필요합니다.

- Element 소프트웨어를 실행하는 지원되는 스토리지 시스템.
- 볼륨을 관리할 수 있는 NetApp HCI/SolidFire 클러스터 관리자 또는 테넌트 사용자의 자격 증명입니다.

- 모든 Kubernetes 워커 노드에는 적절한 iSCSI 도구가 설치되어 있어야 합니다. 다음을 참조하십시오. "작업자 노드 준비 정보".

백엔드 **configuration** 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개변수	설명	기본값
`version`		항상 1
`storageDriverName`	스토리지 드라이버의 이름	항상 "solidfire-san"
`backendName`	사용자 지정 이름 또는 스토리지 백엔드	"solidfire_" + 스토리지(iSCSI) IP 주소
`Endpoint`	테넌트 자격 증명이 있는 SolidFire 클러스터의 MVIP	
`SVIP`	스토리지(iSCSI) IP 주소 및 포트	
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트입니다.	""
`TenantName`	사용할 테넌트 이름(찾을 수 없는 경우 생성됨)	
`InitiatorIFace`	iSCSI 트래픽을 특정 호스트 인터페이스로 제한합니다	"default"
`UseCHAP`	iSCSI 인증에는 CHAP를 사용합니다. Trident는 CHAP를 사용합니다.	true
`AccessGroups`	사용할 액세스 그룹 ID 목록	"Trident"라는 이름의 액세스 그룹 ID를 찾습니다.
`Types`	QoS 사양	
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다	"" (기본적으로 적용되지 않음)
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예를 들어, {"api":false, "method":true}	null

경고 `debugTraceFlags`문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마십시오.

예시 **1:** 세 가지 볼륨 유형을 사용하는 `solidfire-san` 드라이버의 백엔드 구성

이 예제는 CHAP 인증을 사용하는 백엔드 파일을 보여주며, 특정 QoS 보장을 적용한 세 가지 볼륨 유형을 모델링합니다. 일반적으로는 `IOPS` 스토리지 클래스 매개변수를 사용하여 이러한 각 볼륨 유형을 사용할 스토리지 클래스를 정의하게 됩니다.

```
---
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
```

예제 **2:** 가상 풀을 사용하는 `solidfire-san` 드라이버의 백엔드 및 스토리지 클래스 구성

이 예시는 가상 풀과 해당 가상 풀을 참조하는 StorageClasses로 구성된 백엔드 정의 파일을 보여줍니다.

Trident는 프로비저닝 시 스토리지 풀에 있는 레이블을 백엔드 스토리지 LUN으로 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

아래 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 특정 기본값이 설정되어 있으며, 이는 `type`을 `Silver`로 설정합니다. 가상 풀은 `storage` 섹션에서 정의됩니다. 이 예에서 일부 스토리지 풀은 자체 유형을 설정하고 일부 풀은 위에 설정된 기본값을 재정의합니다.

```
---
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 풀을 참조합니다. `parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 지정합니다. 볼륨은 선택된 가상 풀에 정의된 속성을 갖게 됩니다.

첫 번째 StorageClass(`solidfire-gold-four`는 첫 번째 가상 풀에 매핑됩니다. 이 풀은 Gold `Volume Type QoS`의 Gold 성능을 제공하는 유일한 풀입니다. 마지막 StorageClass(`solidfire-silver`는 Silver 성능을 제공하는 모든 스토리지 풀을 지정합니다. Trident는 어떤 가상 풀이 선택될지 결정하고 스토리지 요구사항이 충족되도록 보장합니다.

```
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 정보 찾기

- "볼륨 액세스 그룹"

ONTAP SAN 드라이버

ONTAP SAN 드라이버 개요

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

ONTAP SAN 드라이버 세부 정보

Trident는 ONTAP 클러스터와 통신하기 위해 다음과 같은 SAN 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(RWX), *ReadWriteOncePod*(RWOP)입니다.

드라이버	프로토콜	**volumeMode**	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-san`	FC를 통한 iSCSI SCSI	블록	RWO, ROX, RWX, RWOP	파일 시스템 없음, 원시 블록 디바이스
`ontap-san`	FC를 통한 iSCSI SCSI	파일 시스템	RWO, RWOP ROX 및 RWX는 Filesystem 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP 다음을 참조하십시오. NVMe/TCP에 대한 추가 고려 사항.	블록	RWO, ROX, RWX, RWOP	파일 시스템 없음, 원시 블록 디바이스

드라이버	프로토콜	**volumeMode**	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-san`	NVMe/TCP 다음을 참조하십시오. NVMe/TCP에 대한 추가 고려 사항.	파일 시스템	RWO, RWOP ROX 및 RWX는 Filesystem 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	iSCSI	블록	RWO, ROX, RWX, RWOP	파일 시스템 없음, 원시 블록 디바이스
`ontap-san-economy`	iSCSI	파일 시스템	RWO, RWOP ROX 및 RWX는 Filesystem 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3`, `ext4`

경고

- `ontap-san-economy`지속적 볼륨 사용 수가 "지원되는 ONTAP 볼륨 제한"보다 높을 것으로 예상되는 경우에만 사용하십시오.
- `ontap-nas-economy`영구 볼륨 사용 수가 "지원되는 ONTAP 볼륨 제한"보다 높을 것으로 예상되고 `ontap-san-economy` 드라이버를 사용할 수 없는 경우에만 사용하십시오.
- 데이터 보호, 재해 복구 또는 이동성이 필요할 것으로 예상되는 경우에는 `ontap-nas-economy`사용하지 마십시오.
- NetApp은 ontap-san을 제외한 모든 ONTAP 드라이버에서 Flexvol 자동 확장을 사용하지 않는 것을 권장합니다. 해결 방법으로 Trident는 스냅샷 예약 기능을 지원하며, 이에 따라 Flexvol 볼륨을 확장합니다.

사용자 권한

Trident는 일반적으로 `admin` 클러스터 사용자 또는 `vsadmin` SVM 사용자, 또는 동일한 역할을 가진 다른 이름의 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행되어야 합니다. Amazon FSx for NetApp ONTAP 배포의 경우 Trident는 클러스터 `fsxadmin` 사용자 또는 `vsadmin` SVM 사용자, 또는 동일한 역할을 가진 다른 이름의 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행되어야 합니다. `fsxadmin` 사용자는 클러스터 관리자 사용자를 제한적으로 대체합니다.

참고

`limitAggregateUsage` 매개변수를 사용하는 경우 클러스터 관리자 권한이 필요합니다. Trident와 함께 Amazon FSx for NetApp ONTAP를 사용하는 경우 `limitAggregateUsage` 매개변수는 `vsadmin` 및 `fsxadmin` 사용자 계정에서 작동하지 않습니다. 이 매개변수를 지정하면 구성 작업이 실패합니다.

ONTAP 내에서 Trident 드라이버가 사용할 수 있는 더욱 제한적인 역할을 생성하는 것도 가능하지만 권장하지 않습니다. 대부분의 새로운 Trident 릴리스에서는 추가 API를 호출하므로 이를 고려해야 하기 때문에 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

NVMe/TCP에 대한 추가 고려 사항

Trident는 `ontap-san` 드라이버를 사용하여 다음을 포함한 NVMe(Non-Volatile Memory Express) 프로토콜을 지원합니다.

- IPv6
- NVMe 볼륨의 스냅샷 및 클론
- NVMe 볼륨 크기 조정
- Trident 외부에서 생성된 NVMe 볼륨을 가져와서 Trident에서 수명 주기를 관리할 수 있도록 합니다
- NVMe 네이티브 다중 경로
- K8s 노드의 정상 종료 또는 비정상 종료(24.06)

Trident는 다음을 지원하지 않습니다.

- NVMe에서 기본적으로 지원하는 DH-HMAC-CHAP
- 장치 매퍼(DM) 다중 경로
- LUKS 암호화

참고	NVMe는 ONTAP REST API에서만 지원되며 ONTAPI(ZAPI)에서는 지원되지 않습니다.

ONTAP SAN 드라이버를 사용하여 백엔드를 구성할 준비를 하십시오

ONTAP SAN 드라이버를 사용하여 ONTAP 백엔드를 구성하기 위한 요구 사항 및 인증 옵션을 이해하십시오.

요구 사항

모든 ONTAP 백엔드의 경우 Trident를 사용하려면 SVM에 하나 이상의 애그리게이트를 할당해야 합니다.

참고	"ASA r2 시스템"는 스토리지 계층 구현 방식에서 다른 ONTAP 시스템(ASA, AFF, FAS)과 차이가 있습니다. ASA r2 시스템에서는 애그리게이트 대신 스토리지 가용성 영역을 사용합니다. ASA r2 시스템에서 SVM에 애그리게이트를 할당하는 방법에 대한 "이것" 기술 자료 문서를 참조하십시오.

여러 드라이버를 실행할 수도 있고, 특정 드라이버를 가리키는 스토리지 클래스를 생성할 수도 있다는 점을 기억하세요. 예를 들어, `san-dev` 드라이버를 사용하는 `ontap-san` 클래스와 `san-default` 드라이버를 사용하는 `ontap-san-economy` 클래스를 구성할 수 있습니다.

모든 Kubernetes 워커 노드에는 적절한 iSCSI 도구가 설치되어 있어야 합니다. 자세한 내용은 "작업자 노드를 준비합니다"를 참조하십시오.

ONTAP 백엔드를 인증합니다

Trident는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 필요한 권한을 가진 ONTAP 사용자의 사용자 이름과 암호입니다. ONTAP 버전과의 최대 호환성을 보장하기 위해 `admin` 또는 `vsadmin`와 같이 미리 정의된 보안 로그인 역할을 사용하는 것이 좋습니다.
- 인증서 기반: Trident는 백엔드에 설치된 인증서를 사용하여 ONTAP 클러스터와 통신할 수도 있습니다. 이 경우

백엔드 정의에는 클라이언트 인증서, 키, 그리고 사용하는 경우(권장) 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값이 포함되어야 합니다.

기존 백엔드를 업데이트하여 자격 증명 기반 방식과 인증서 기반 방식 간에 전환할 수 있습니다. 단, 한 번에 하나의 인증 방법만 지원됩니다. 다른 인증 방법으로 전환하려면 백엔드 구성에서 기존 방법을 제거해야 합니다.

경고	자격 증명과 인증서를 모두 제공하려고 하면 구성 파일에 두 개 이상의 인증 방법이 제공되었다는 오류와 함께 백엔드 생성이 실패합니다.

자격 증명 기반 인증 활성화

Trident는 ONTAP 백엔드와 통신하기 위해 SVM 범위/클러스터 범위 관리자의 자격 증명이 필요합니다. `admin` 또는 `vsadmin`와 같은 표준 사전 정의된 역할을 사용하는 것이 좋습니다. 이렇게 하면 향후 Trident 릴리스에서 사용할 수 있는 기능 API를 노출할 수 있는 향후 ONTAP 릴리스와의 상위 호환성이 보장됩니다. 사용자 지정 보안 로그인 역할을 생성하여 Trident와 함께 사용할 수 있지만 권장하지 않습니다.

백엔드 정의 샘플은 다음과 같습니다.

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

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

백엔드 정의는 자격 증명이 평문으로 저장되는 유일한 위치라는 점을 명심하십시오. 백엔드가 생성된 후 사용자 이름/암호는 Base64로 인코딩되어 Kubernetes 시크릿으로 저장됩니다. 백엔드 생성 또는 업데이트는 자격 증명을 알아야 하는 유일한 단계입니다. 따라서 이는 Kubernetes/스토리지 관리자가 수행하는 관리자 전용 작업입니다.

인증서 기반 인증을 활성화합니다

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

- clientCertificate: 클라이언트 인증서의 Base64 인코딩 값입니다.
- clientPrivateKey: 연결된 개인 키의 Base64 인코딩 값입니다.
- trustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개변수를 반드시 제공해야 합니다. 신뢰할 수 있는 CA를 사용하지 않는 경우에는 이 매개변수를 무시할 수 있습니다.

일반적인 워크플로에는 다음 단계가 포함됩니다.

단계

1. 클라이언트 인증서와 키를 생성합니다. 생성 시 CN(일반 이름)을 인증할 ONTAP 사용자로 설정합니다.

```
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. ONTAP 클러스터에 신뢰할 수 있는 CA 인증서를 추가합니다. 스토리지 관리자가 이미 처리했을 수 있습니다. 신뢰할 수 있는 CA를 사용하지 않는 경우 이 단계를 건너뛰십시오.

```
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
```

참고	이 명령을 실행하면 ONTAP에서 인증서 입력을 요청합니다. 1단계에서 생성된 `k8senv.pem` 파일의 내용을 붙여넣은 다음 `END`를 입력하여 설치를 완료하십시오.

4. ONTAP 보안 로그인 역할이 `cert` 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 테스트하십시오. <ONTAP Management LIF> 및 <vserver name>를 Management LIF IP 및 SVM 이름으로 바꾸십시오.

```
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. 인증서, 키 및 신뢰할 수 있는 CA 인증서를 Base64로 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",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"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 변경하세요

기존 백엔드를 업데이트하여 다른 인증 방법을 사용하거나 자격 증명을 교체할 수 있습니다. 이 기능은 양방향으로 작동합니다. 사용자 이름/암호를 사용하는 백엔드를 인증서를 사용하는 방식으로 업데이트할 수 있고, 인증서를 사용하는 백엔드를 사용자 이름/암호 기반으로 업데이트할 수도 있습니다. 이렇게 하려면 기존 인증 방법을 제거하고 새 인증 방법을 추가해야 합니다. 그런 다음 필요한 매개변수가 포함된 업데이트된 backend.json 파일을 사용하여 `tridentctl backend update`을(를) 실행합니다.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

참고	암호를 교체할 때는 스토리지 관리자가 먼저 ONTAP에서 사용자의 암호를 업데이트해야 합니다. 그 후 백엔드 업데이트를 진행합니다. 인증서를 교체할 때는 사용자에게 여러 개의 인증서를 추가할 수 있습니다. 백엔드를 업데이트하여 새 인증서를 사용하도록 설정한 후, ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스는 중단되지 않으며, 이후에 생성된 볼륨 연결에도 영향을 미치지 않습니다. 백엔드 업데이트가 성공적으로 완료되면 Trident가 ONTAP 백엔드와 통신하여 향후 볼륨 작업을 처리할 수 있음을 의미합니다.

Trident용 사용자 지정 **ONTAP** 역할 생성

Trident에서 작업을 수행하기 위해 ONTAP 관리자 역할을 사용할 필요가 없도록 최소 권한으로 ONTAP 클러스터 역할을 생성할 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident는 생성한 ONTAP 클러스터 역할을

사용하여 작업을 수행합니다.

Trident 사용자 지정 역할 생성에 대한 자세한 내용은 "Trident 사용자 지정 역할 생성기"을 참조하십시오.

ONTAP CLI 사용

1. 다음 명령을 사용하여 새 역할을 생성합니다.

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Trident 사용자의 사용자 이름을 생성합니다.

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. 사용자에게 역할 매핑:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

System Manager 사용

ONTAP System Manager에서 다음 단계를 수행하십시오.

1. 사용자 지정 역할 생성:
   a. 클러스터 수준에서 사용자 지정 역할을 생성하려면 * Cluster > Settings * 를 선택합니다.

      (또는) SVM 수준에서 사용자 지정 역할을 생성하려면 *스토리지 > 스토리지 VM > `required SVM`> 설정 > 사용자 및 역할*을 선택하십시오.

   b. 사용자 및 역할 옆에 있는 화살표 아이콘(→)을 선택합니다.
   c. 역할 에서 **+**추가 를 선택합니다.
   d. 역할에 대한 규칙을 정의하고 *저장*을 클릭합니다.
2. **Trident** 사용자에게 역할 매핑: + 사용자 및 역할 페이지에서 다음 단계를 수행합니다.
   a. 사용자 아래에서 추가 아이콘 *+*를 선택합니다.
   b. 필요한 사용자 이름을 선택하고 역할 드롭다운 메뉴에서 역할을 선택합니다.
   c. *저장*을 클릭합니다.

자세한 내용은 다음 페이지를 참조하십시오.

- "ONTAP 관리를 위한 사용자 지정 역할" 또는 "사용자 지정 역할 정의"
- "역할 및 사용자 작업"

Trident는 `ontap-san` 및 `ontap-san-economy` 드라이버에 대해 양방향 CHAP를 사용하여 iSCSI 세션을 인증할 수 있습니다. 이를 위해서는 백엔드 정의에서 `useCHAP` 옵션을 활성화해야 합니다. `true`로 설정하면 Trident는 SVM의 기본 이니시에이터 보안을 양방향 CHAP로 구성하고 백엔드 파일에서 사용자 이름과 암호를 설정합니다. NetApp은 연결 인증에 양방향 CHAP를 사용하는 것을 권장합니다. 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: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

경고	`useCHAP` 매개변수는 한 번만 구성할 수 있는 부울 옵션입니다. 기본적으로 false로 설정됩니다. true로 설정한 후에는 false로 설정할 수 없습니다.

`useCHAP=true` 외에도 `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername` 및 `chapUsername` 필드는 백엔드 정의에 반드시 포함되어야 합니다. 백엔드 생성 후에는 `tridentctl update`를 실행하여 비밀 키를 변경할 수 있습니다.

작동 방식

`useCHAP`을 `true`로 설정하면 스토리지 관리자가 `Trident`에 스토리지 백엔드에서 `CHAP`를 구성하도록 지시합니다. 여기에는 다음 사항이 포함됩니다.

- SVM에서 CHAP 설정:
  - SVM의 기본 이니시에이터 보안 유형이 없음(기본값으로 설정)이고 볼륨에 기존 LUN이 없는 경우, Trident는 기본 보안 유형을 `CHAP`로 설정하고 CHAP 이니시에이터 및 대상 사용자 이름과 암호 구성을 진행합니다.
  - SVM에 LUN이 포함되어 있는 경우 Trident는 SVM에서 CHAP를 활성화하지 않습니다. 이렇게 하면 SVM에 이미 있는 LUN에 대한 액세스가 제한되지 않습니다.
- CHAP 이니시에이터 및 타겟 사용자 이름과 암호를 구성합니다. 이러한 옵션은 백엔드 구성에 지정해야 합니다(위 참조).

백엔드가 생성되면 Trident는 해당 `tridentbackend` CRD를 생성하고 CHAP 시크릿과 사용자 이름을 Kubernetes

시크릿으로 저장합니다. 이 백엔드에서 Trident가 생성하는 모든 PV는 CHAP를 통해 마운트되고 연결됩니다.

자격 증명을 교체하고 백엔드를 업데이트합니다

`backend.json` 파일의 CHAP 매개변수를 업데이트하여 CHAP 자격 증명을 업데이트할 수 있습니다. 이를 위해서는 CHAP 암호를 업데이트하고 `tridentctl update` 명령을 사용하여 이러한 변경 사항을 반영해야 합니다.

경고 백엔드의 CHAP 암호를 업데이트할 때는 `tridentctl`을 사용하여 백엔드를 업데이트해야 합니다. Trident가 이러한 변경 사항을 인식할 수 없으므로 ONTAP CLI 또는 ONTAP System Manager를 사용하여 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": "password",
    "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 않으며, SVM에서 Trident가 자격 증명을 업데이트하면 계속 활성 상태로 유지됩니다. 새 연결은 업데이트된 자격 증명을 사용하고 기존 연결은 계속 활성 상태로 유지됩니다. 기존 PV를 연결 해제했다가 다시 연결하면 업데이트된 자격 증명이 사용됩니다.

ONTAP SAN 구성 옵션 및 예

Trident 설치 환경에서 ONTAP SAN 드라이버를 생성하고 사용하는 방법을 알아보세요. 이 섹션에서는 백엔드 구성 예제와 백엔드를 StorageClasses에 매핑하는 방법에 대한 세부 정보를 제공합니다. "ASA r2 시스템"는 스토리지 계층 구현 방식에서 다른 ONTAP 시스템(ASA, AFF, FAS)과 차이가 있습니다. 이러한 차이점은 표기된 특정 매개변수의 사용에 영향을 미칩니다. "ASA r2 시스템과 다른 ONTAP 시스템 간의 차이점에 대해 자세히 알아보십시오". Trident 백엔드 구성에서 시스템이 ASA r2임을 지정할 필요가 없습니다. `ontap-san`을 `storageDriverName`로 선택하면 Trident가 ASA r2 또는 기타 ONTAP 시스템을 자동으로 감지합니다. 아래 표에 나와 있는 것처럼 일부 백엔드 구성 매개 변수는 ASA r2 시스템에 적용되지 않습니다.

참고	`ontap-san` 드라이버(iSCSI, NVMe/TCP 및 FC 프로토콜 포함)만 ASA r2 시스템에서 지원됩니다.

백엔드 configuration 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개변수	설명	기본값
`version`		항상 1
`storageDriverName`	스토리지 드라이버의 이름	`ontap-san` 또는 `ontap-san-economy`
`backendName`	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소입니다. 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. Trident를 IPv6 플래그를 사용하여 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`와 같이 대괄호로 정의해야 합니다. 원활한 MetroCluster 전환을 위해서는 MetroCluster 예시을 참조하십시오. 참고: "vsadmin" 자격 증명을 사용하는 경우 `managementLIF`은 SVM의 자격 증명이어야 하며, "admin" 자격 증명을 사용하는 경우 `managementLIF`는 클러스터의 자격 증명이어야 합니다.	"10.0.0.1", "[2001:1234:abcd::fefe]"

매개변수	설명	기본값
`dataLIF`	프로토콜 LIF의 IP 주소입니다. Trident를 IPv6 플래그를 사용하여 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`와 같이 대괄호로 정의해야 합니다. **iSCSI**의 경우 지정하지 마십시오. Trident는 "ONTAP 선택적 LUN 매핑"를 사용하여 다중 경로 세션을 설정하는 데 필요한 iSCSI LIF를 검색합니다. `dataLIF`가 명시적으로 정의된 경우 경고가 생성됩니다. **MetroCluster**의 경우 생략합니다. MetroCluster 예시를 참조하십시오.	SVM에 의해 도출됨
`svm`	사용할 스토리지 가상 머신 **MetroCluster**의 경우 생략 MetroCluster 예시를 참조하십시오.	SVM `managementLIF`이 지정된 경우 파생됩니다
`useCHAP`	ONTAP SAN 드라이버용 iSCSI 인증에 CHAP를 사용합니다[부울 매개 변수]. 백엔드에 제공된 SVM에 대해 양방향 CHAP를 기본 인증으로 구성하고 사용하려면 Trident에 대해 `true`로 설정하십시오. 자세한 내용은 "ONTAP SAN 드라이버를 사용하여 백엔드를 구성할 준비를 하십시오"를 참조하십시오. **FCP** 또는 **NVMe/TCP**에서는 지원되지 않습니다.	`false`
`chapInitiatorSecret`	CHAP 이니시에이터 암호입니다. 다음의 경우 필수 사항입니다 `useCHAP=true`	""
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`chapTargetInitiatorSecret`	CHAP 대상 개시자 비밀 키. 다음의 경우 필수 사항입니다 `useCHAP=true`	""
`chapUsername`	인바운드 사용자 이름. 다음의 경우 필수 사항입니다 `useCHAP=true`	""
`chapTargetUsername`	대상 사용자 이름. 다음의 경우 필수 사항입니다 `useCHAP=true`	""
`clientCertificate`	클라이언트 인증서의 Base64 인코딩 값입니다. 인증서 기반 인증에 사용됩니다	""
`clientPrivateKey`	클라이언트 개인 키를 Base64로 인코딩한 값입니다. 인증서 기반 인증에 사용됩니다	""
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 선택 사항입니다. 인증서 기반 인증에 사용됩니다.	""
`username`	ONTAP 클러스터와 통신하는 데 필요한 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증은 "Active Directory 자격 증명을 사용하여 Trident를 백엔드 SVM에 인증합니다"을(를) 참조하십시오.	""
`password`	ONTAP 클러스터와 통신하는 데 필요한 암호입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증은 "Active Directory 자격 증명을 사용하여 Trident를 백엔드 SVM에 인증합니다"을(를) 참조하십시오.	""

매개변수	설명	기본값
`svm`	사용할 스토리지 가상 머신	SVM `managementLIF`이 지정된 경우 파생됩니다
`storagePrefix`	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 나중에 수정할 수 없습니다. 이 매개 변수를 업데이트하려면 새 백엔드를 생성해야 합니다.	`trident`
`aggregate`	프로비저닝용 애그리게이트(선택 사항, 설정된 경우 SVM에 할당해야 함). `ontap-nas-flexgroup` 드라이버의 경우 이 옵션은 무시됩니다. 할당하지 않으면 사용 가능한 애그리게이트 중 하나를 사용하여 FlexGroup 볼륨을 프로비저닝할 수 있습니다. 참고: SVM에서 애그리게이트가 업데이트되면 Trident 컨트롤러를 재시작하지 않고도 SVM을 폴링하여 Trident에 자동으로 업데이트됩니다. Trident에서 볼륨을 프로비저닝하도록 특정 애그리게이트를 구성한 경우, 해당 애그리게이트의 이름이 변경되거나 SVM에서 이동되면 SVM 애그리게이트를 폴링하는 동안 Trident에서 백엔드가 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 전환하려면 SVM에 있는 애그리게이트로 변경하거나 해당 애그리게이트를 완전히 제거해야 합니다. **ASA r2** 시스템의 경우 지정하지 마십시오.	""
`limitAggregateUsage`	사용량이 이 비율을 초과하면 프로비저닝이 실패합니다. Amazon FSx for NetApp ONTAP 백엔드를 사용하는 경우 `limitAggregateUsage`을(를) 지정하지 마십시오. 제공된 `fsxadmin` 및 `vsadmin`에는 Trident를 사용하여 애그리게이트 사용량을 검색하고 제한하는 데 필요한 권한이 포함되어 있지 않습니다. **ASA r2** 시스템의 경우 지정하지 마십시오.	"" (기본적으로 적용되지 않음)
`limitVolumeSize`	요청된 볼륨 크기가 이 값을 초과하면 프로비저닝이 실패합니다. 또한 LUN에 대해 관리하는 볼륨의 최대 크기를 제한합니다.	"" (기본적으로 적용되지 않음)
`lunsPerFlexvol`	FlexVol당 최대 LUN 수는 [50, 200] 범위 내에 있어야 합니다.	`100`
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예: {"api":false, "method":true} 문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마십시오.	`null`

매개변수	설명	기본값
useREST	ONTAP REST API를 사용하기 위한 부울 매개 변수입니다. `useREST`로 설정하면 `true` Trident는 ONTAP REST API를 사용하여 백엔드와 통신하고, `false`로 설정하면 Trident는 ONTAPI(ZAPI) 호출을 사용하여 백엔드와 통신합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한 사용되는 ONTAP 로그인 역할은 `ontapi` 애플리케이션에 대한 액세스 권한이 있어야 합니다. 이는 사전 정의된 `vsadmin` 및 `cluster-admin` 역할로 충족됩니다. Trident 24.06 릴리스 및 ONTAP 9.15.1 이상부터 `useREST`는 기본적으로 `true`로 설정되며, ONTAPI(ZAPI) 호출을 사용하려면 `useREST`를 `false`로 변경하십시오. `useREST`는 NVMe/TCP에 대한 모든 자격을 갖추고 있습니다. 참고: NVMe는 ONTAP REST API에서만 지원되며 ONTAPI(ZAPI)에서는 지원되지 않습니다. 지정된 경우 **ASA r2** 시스템의 경우 항상 **`true`**로 설정하십시오.	ONTAP 9.15.1 이상의 경우 true, 그렇지 않은 경우 false.
sanType	iSCSI의 경우 iscsi, NVMe/TCP의 경우 nvme 또는 Fibre Channel(FC)을 통한 SCSI의 경우 `fcp`를 선택하는 데 사용합니다.	iscsi 공백인 경우

매개변수	설명	기본값
formatOptions	`formatOptions`를 사용하여 `mkfs` 명령에 대한 명령줄 인수를 지정할 수 있으며, 이는 볼륨을 포맷할 때마다 적용됩니다. 이를 통해 원하는 대로 볼륨을 포맷할 수 있습니다. 장치 경로를 제외하고 mkfs 명령 옵션과 유사하게 formatOptions를 지정해야 합니다. 예: "-E nodiscard" • ontap-san 및 ontap-san-economy 드라이버에서 iSCSI 프로토콜과 함께 지원됩니다.* 또한 **iSCSI** 및 **NVMe/TCP** 프로토콜을 사용하는 **ASA r2** 시스템에서도 지원됩니다.	
limitVolumePoolSize	ontap-san-economy 백엔드에서 LUN을 사용할 때 요청할 수 있는 최대 FlexVol 크기입니다.	"" (기본적으로 적용되지 않음)
denyNewVolumePools	ontap-san-economy 백엔드에서 LUN을 포함할 새 FlexVol 볼륨 생성을 제한합니다. 기존 Flexvol만 새 PV 프로비저닝에 사용됩니다.	

formatOptions 사용 권장 사항

Trident는 포맷 프로세스를 신속하게 진행하기 위해 다음 옵션을 권장합니다.

- **-E nodiscard (ext3, ext4):** mkfs 실행 시 블록을 버리지 않도록 합니다(초기 블록 버리기는 솔리드 스테이트 장치 및 스파스/씬 프로비저닝 스토리지에서 유용합니다). 이 옵션은 더 이상 사용되지 않는 "-K" 옵션을 대체하며 ext3, ext4 파일 시스템에 적용됩니다.
- **-K (xfs):** mkfs 실행 시 블록을 버리려고 시도하지 않습니다. 이 옵션은 xfs 파일 시스템에 적용됩니다.

Active Directory 자격 증명을 사용하여 **Trident**를 백엔드 **SVM**에 인증합니다

Active Directory(AD) 자격 증명을 사용하여 백엔드 SVM에 인증하도록 Trident를 구성할 수 있습니다. AD 계정이 SVM에 액세스하려면 먼저 클러스터 또는 SVM에 대한 AD 도메인 컨트롤러 액세스를 구성해야 합니다. AD 계정을 사용하여 클러스터를 관리하려면 도메인 터널을 생성해야 합니다. 자세한 내용은 "ONTAP에서 Active Directory 도메인 컨트롤러 액세스 구성"를 참조하십시오.

단계

1. 백엔드 SVM에 대한 DNS(Domain Name System) 설정을 구성합니다.

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. Active Directory에서 SVM용 컴퓨터 계정을 생성하려면 다음 명령을 실행하십시오.

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. 이 명령을 사용하여 클러스터 또는 SVM을 관리할 AD 사용자 또는 그룹을 생성합니다

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. Trident 백엔드 구성 파일에서 `username` 및 `password` 매개 변수를 각각 AD 사용자 또는 그룹 이름과 암호로 설정하십시오.

볼륨 프로비저닝을 위한 백엔드 구성 옵션

구성의 `defaults` 섹션에서 이러한 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. 예를 들어, 아래 구성 예를 참조하십시오.

매개변수	설명	기본값
`spaceAllocation`	LUN의 공간 할당	"true" 지정된 경우 **ASA r2** 시스템의 경우 `**true**`로 설정하십시오.
`spaceReserve`	공간 예약 모드: "none"(씬) 또는 "volume"(씩). **ASA r2** 시스템의 경우 `none`로 설정합니다.	"없음"
`snapshotPolicy`	사용할 스냅샷 정책입니다. **ASA r2** 시스템의 경우 `**none**`로 설정하세요.	"없음"
`qosPolicy`	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하십시오. Trident에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유되지 않는 QoS 정책 그룹을 사용하고 정책 그룹이 각 구성 요소에 개별적으로 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 대한 상한선을 적용합니다.	""
`adaptiveQosPolicy`	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀/백엔드당 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하십시오	""
`snapshotReserve`	스냅샷에 할당할 볼륨 비율입니다. **ASA r2** 시스템의 경우 지정하지 마십시오.	`snapshotPolicy`이(가) "none"이면 "0", 그렇지 않으면 ""
`splitOnClone`	생성 시 상위 항목에서 클론 분할	"false"
`encryption`	새 볼륨에서 NetApp Volume Encryption(NVE)을 활성화합니다. 기본값은 `false`입니다. 이 옵션을 사용하려면 클러스터에서 NVE 라이센스가 있고 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하십시오. "Trident가 NVE 및 NAE와 작동하는 방식"	"false" 지정된 경우 **ASA r2** 시스템의 경우 `**true**`로 설정하십시오.
`luksEncryption`	LUKS 암호화를 활성화합니다. 다음을 참조하십시오. "Linux Unified Key Setup(LUKS) 사용".	"" **ASA r2** 시스템의 경우 `**false**`로 설정하세요.
`tieringPolicy`	계층화 정책에서 "none"을 사용합니다. **ASA r2** 시스템의 경우 지정하지 마십시오.	
`nameTemplate`	사용자 지정 볼륨 이름을 생성하기 위한 템플릿입니다.	""

볼륨 프로비저닝 예

다음은 기본값이 정의된 예입니다.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

참고

`ontap-san` 드라이버를 사용하여 생성된 모든 볼륨에 대해 Trident는 LUN 메타데이터를 수용하기 위해 FlexVol에 10%의 추가 용량을 더합니다. LUN은 사용자가 PVC에서 요청한 정확한 크기로 프로비저닝됩니다. Trident는 FlexVol에 10%를 추가합니다(ONTAP에서 사용 가능한 크기로 표시됨). 이제 사용자는 요청한 만큼의 사용 가능한 용량을 확보할 수 있습니다. 또한 이 변경 사항은 사용 가능한 공간이 완전히 활용되지 않는 한 LUN이 읽기 전용이 되는 것을 방지합니다. 이는 ontap-san-economy에는 적용되지 않습니다.

`snapshotReserve`을(를) 정의하는 백엔드의 경우 `Trident`는 다음과 같이 볼륨 크기를 계산합니다.

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

1.1은 Trident가 LUN 메타데이터를 수용하기 위해 FlexVol에 추가하는 10%입니다. `snapshotReserve` = 5%이고 PVC 요청 = 5GiB인 경우 전체 볼륨 크기는 5.79GiB이고 사용 가능한 크기는 5.5GiB입니다. `volume show` 명령은 다음 예와 유사한 결과를 표시해야 합니다.

현재로서는 크기 조정만이 기존 볼륨에 대한 새 계산을 사용하는 유일한 방법입니다.

최소 구성 예

다음 예시들은 대부분의 매개변수를 기본값으로 유지하는 기본 구성을 보여줍니다. 이것이 백엔드를 정의하는 가장 쉬운 방법입니다.

참고	Amazon FSx for NetApp ONTAP에서 Trident를 사용하는 경우 NetApp에서는 LIF에 IP 주소 대신 DNS 이름을 지정하는 것이 좋습니다.

ONTAP SAN 예

이는 `ontap-san` 드라이버를 사용한 기본 구성입니다.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

MetroCluster 예시

"SVM 복제 및 복구" 중에 스위치오버 및 스위치백 후 백엔드 정의를 수동으로 업데이트하지 않아도 되도록 백엔드를 구성할 수 있습니다.

원활한 전환 및 복귀를 위해 `managementLIF`을 사용하여 `SVM`을 지정하고 `svm` 매개변수를 생략하십시오. 예를 들면 다음과 같습니다.

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

ONTAP SAN 이코노미 예

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

인증서 기반 인증 예

이 기본 구성 예에서 `clientCertificate`, `clientPrivateKey` 및 `trustedCACertificate`(신뢰할 수 있는 CA를 사용하는 경우 선택 사항)는 `backend.json`에 채워지며 각각 클라이언트 인증서, 개인 키 및 신뢰할 수 있는 CA 인증서의 base64로 인코딩된 값을 사용합니다.

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

양방향 **CHAP** 예

이 예제들은 `useCHAP`로 설정된 `true` 백엔드를 생성합니다.

ONTAP SAN CHAP 예

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

ONTAP SAN 경제 **CHAP** 예

```
---
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
username: vsadmin
password: <password>
```

NVMe/TCP 예

ONTAP 백엔드에 NVMe가 구성된 SVM이 있어야 합니다. NVMe/TCP를 위한 기본 백엔드 구성입니다.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

FC(FCP)를 통한 SCSI 예

ONTAP 백엔드에 FC가 구성된 SVM이 있어야 합니다. 다음은 FC를 위한 기본 백엔드 구성입니다.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

nameTemplate을 사용한 백엔드 구성 예

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

formatOptions 예시(**ontap-san-economy** 드라이버용)

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

가상 풀이 있는 백엔드의 예

이 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 특정 기본값(예: `spaceReserve`없음, `spaceAllocation`false, `encryption`false)이 설정되어 있습니다. 가상 풀은 스토리지 섹션에서 정의됩니다.

Trident는 "설명" 필드에 프로비저닝 레이블을 설정합니다. 설명은 FlexVol 볼륨에 설정됩니다. Trident는 프로비저닝 시 가상 풀에 있는 모든 레이블을 스토리지 볼륨으로 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

이 예시에서 일부 스토리지 풀은 자체 `spaceReserve`, `spaceAllocation` 및 `encryption` 값을 설정하고, 일부 풀은 기본값을 재정의합니다.

ONTAP SAN 예

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
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"
```

ONTAP SAN 이코노미 예

```
---
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
username: vsadmin
password: <password>
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"
  - labels:
      department: legal
      creditpoints: "5000"
```

```
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

NVMe/TCP 예

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: "false"
  encryption: "true"
storage:
  - labels:
      app: testApp
      cost: "20"
    defaults:
      spaceAllocation: "false"
      encryption: "false"
```

백엔드를 **StorageClasses**에 매핑합니다

다음 StorageClass 정의는 가상 풀이 있는 백엔드의 예을 참조합니다. `parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 지정합니다. 볼륨은 선택된 가상 풀에 정의된 속성을 갖게 됩니다.

- `protection-gold` StorageClass는 `ontap-san` 백엔드의 첫 번째 가상 풀에 매핑됩니다. 이 풀만이 골드 레벨 보호를 제공합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- `protection-not-gold` StorageClass는 `ontap-san` 백엔드의 두 번째 및 세 번째 가상 풀에 매핑됩니다. 이 두 풀만이 골드 수준 외에 다른 보호 수준을 제공합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- `app-mysqldb` StorageClass는 `ontap-san-economy` 백엔드의 세 번째 가상 풀에 매핑됩니다. 이 풀은 mysqldb 유형 애플리케이션에 대한 스토리지 풀 구성을 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- `protection-silver-creditpoints-20k` StorageClass는 `ontap-san` 백엔드의 두 번째 가상 풀에 매핑됩니다. 이 풀은 실버 등급 보호와 20000 크레딧 포인트를 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- `creditpoints-5k` StorageClass는 `ontap-san` 백엔드의 세 번째 가상 풀과 `ontap-san-economy` 백엔드의 네 번째 가상 풀에 매핑됩니다. 5000 크레딧 포인트를 제공하는 풀은 이 두 가지뿐입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- `my-test-app-sc` StorageClass는 `testAPP` 드라이버의 `ontap-san` 가상 풀에 `sanType: nvme`로 매핑됩니다. 이것이 `testApp`을(를) 제공하는 유일한 풀입니다.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Trident는 어떤 가상 풀이 선택될지 결정하고 스토리지 요구사항이 충족되도록 보장합니다.

ONTAP NAS 드라이버

ONTAP NAS 드라이버 개요

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

ONTAP NAS 드라이버 세부 정보

Trident는 ONTAP 클러스터와 통신하기 위해 다음과 같은 NAS 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(RWX), *ReadWriteOncePod*(RWOP)입니다.

드라이버	프로토콜	**volumeMode**	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-nas`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-economy`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-flexgroup`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

경고

- `ontap-san-economy`지속적 볼륨 사용 수가 "지원되는 ONTAP 볼륨 제한"보다 높을 것으로 예상되는 경우에만 사용하십시오.
- `ontap-nas-economy`영구 볼륨 사용 수가 "지원되는 ONTAP 볼륨 제한"보다 높을 것으로 예상되고 `ontap-san-economy` 드라이버를 사용할 수 없는 경우에만 사용하십시오.
- 데이터 보호, 재해 복구 또는 이동성이 필요할 것으로 예상되는 경우에는 `ontap-nas-economy`사용하지 마십시오.
- NetApp은 ontap-san을 제외한 모든 ONTAP 드라이버에서 Flexvol 자동 확장을 사용하지 않는 것을 권장합니다. 해결 방법으로 Trident는 스냅샷 예약 기능을 지원하며, 이에 따라 Flexvol 볼륨을 확장합니다.

사용자 권한

Trident는 일반적으로 `admin` 클러스터 사용자 또는 `vsadmin` SVM 사용자, 또는 동일한 역할을 가진 다른 이름의 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행되어야 합니다.

Amazon FSx for NetApp ONTAP 배포의 경우 Trident는 클러스터 `fsxadmin` 사용자 또는 `vsadmin` SVM 사용자, 또는 동일한 역할을 가진 다른 이름의 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행되어야 합니다. `fsxadmin` 사용자는 클러스터 관리자 사용자를 제한적으로 대체합니다.

참고

`limitAggregateUsage` 매개변수를 사용하는 경우 클러스터 관리자 권한이 필요합니다. Trident와 함께 Amazon FSx for NetApp ONTAP를 사용하는 경우 `limitAggregateUsage` 매개변수는 `vsadmin` 및 `fsxadmin` 사용자 계정에서 작동하지 않습니다. 이 매개변수를 지정하면 구성 작업이 실패합니다.

ONTAP 내에서 Trident 드라이버가 사용할 수 있는 더욱 제한적인 역할을 생성하는 것도 가능하지만 권장하지 않습니다. 대부분의 새로운 Trident 릴리스에서는 추가 API를 호출하므로 이를 고려해야 하기 때문에 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

ONTAP NAS 드라이버를 사용하여 백엔드를 구성할 준비를 하십시오

ONTAP NAS 드라이버를 사용하여 ONTAP 백엔드를 구성하기 위한 요구 사항, 인증 옵션 및 엑스포트 정책을 이해하십시오. 25.10 릴리스부터 NetApp Trident는 "NetApp AFX 스토리지 시스템"을(를) 지원합니다. NetApp AFX 스토리지 시스템은 스토리지 계층 구현에서 다른

ONTAP 시스템(ASA, AFF, FAS)과 다릅니다. Trident 백엔드 구성에서 시스템이 AFX인지 여부를 지정할 필요가 없습니다. `ontap-nas`을(를) `storageDriverName`(으)로 선택하면 Trident가 AFX 시스템을 자동으로 감지합니다.

참고	AFX 시스템에서는 `ontap-nas` 드라이버(NFS 프로토콜 사용)만 지원되며 SMB 프로토콜은 지원되지 않습니다.

요구 사항

- 모든 ONTAP 백엔드의 경우 Trident를 사용하려면 SVM에 하나 이상의 애그리게이트를 할당해야 합니다.
- 여러 드라이버를 실행하고, 각 드라이버를 가리키는 스토리지 클래스를 생성할 수 있습니다. 예를 들어, `ontap-nas` 드라이버를 사용하는 Gold 클래스와 `ontap-nas-economy` 드라이버를 사용하는 Bronze 클래스를 구성할 수 있습니다.
- 모든 Kubernetes 워커 노드에는 적절한 NFS 도구가 설치되어 있어야 합니다. 자세한 내용은 "여기"을(를) 참조하십시오.
- Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다. 자세한 내용은 SMB 볼륨 프로비저닝 준비를 참조하십시오.

ONTAP 백엔드를 인증합니다

Trident는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 이 모드는 ONTAP 백엔드에 대한 충분한 권한이 필요합니다. ONTAP 버전과의 최대 호환성을 보장하기 위해 `admin` 또는 `vsadmin`와 같이 미리 정의된 보안 로그인 역할과 연결된 계정을 사용하는 것이 좋습니다.
- 인증서 기반: 이 모드에서는 Trident가 ONTAP 클러스터와 통신하기 위해 백엔드에 인증서가 설치되어 있어야 합니다. 이 경우 백엔드 정의에는 클라이언트 인증서, 키, 그리고 사용하는 경우(권장) 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값이 포함되어야 합니다.

기존 백엔드를 업데이트하여 자격 증명 기반 방식과 인증서 기반 방식 간에 전환할 수 있습니다. 단, 한 번에 하나의 인증 방법만 지원됩니다. 다른 인증 방법으로 전환하려면 백엔드 구성에서 기존 방법을 제거해야 합니다.

경고	자격 증명과 인증서를 모두 제공하려고 하면 구성 파일에 두 개 이상의 인증 방법이 제공되었다는 오류와 함께 백엔드 생성이 실패합니다.

자격 증명 기반 인증 활성화

Trident는 ONTAP 백엔드와 통신하기 위해 SVM 범위/클러스터 범위 관리자의 자격 증명이 필요합니다. `admin` 또는 `vsadmin`와 같은 표준 사전 정의된 역할을 사용하는 것이 좋습니다. 이렇게 하면 향후 Trident 릴리스에서 사용할 수 있는 기능 API를 노출할 수 있는 향후 ONTAP 릴리스와의 상위 호환성이 보장됩니다. 사용자 지정 보안 로그인 역할을 생성하여 Trident와 함께 사용할 수 있지만 권장하지 않습니다.

백엔드 정의 샘플은 다음과 같습니다.

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

백엔드 정의는 자격 증명이 평문으로 저장되는 유일한 위치라는 점을 명심하십시오. 백엔드가 생성된 후 사용자 이름/암호는 Base64로 인코딩되어 Kubernetes 시크릿으로 저장됩니다. 백엔드 생성/업데이트는 자격 증명을 알아야 하는 유일한 단계입니다. 따라서 이는 Kubernetes/스토리지 관리자가 수행하는 관리자 전용 작업입니다.

인증서 기반 인증 활성화

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

- clientCertificate: 클라이언트 인증서의 Base64 인코딩 값입니다.
- clientPrivateKey: 연결된 개인 키의 Base64 인코딩 값입니다.
- trustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개변수를 반드시 제공해야 합니다. 신뢰할 수 있는 CA를 사용하지 않는 경우에는 이 매개변수를 무시할 수 있습니다.

일반적인 워크플로에는 다음 단계가 포함됩니다.

단계

1. 클라이언트 인증서와 키를 생성합니다. 생성 시 CN(일반 이름)을 인증할 ONTAP 사용자로 설정합니다.

```
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. ONTAP 클러스터에 신뢰할 수 있는 CA 인증서를 추가합니다. 스토리지 관리자가 이미 처리했을 수 있습니다. 신뢰할 수 있는 CA를 사용하지 않는 경우 이 단계를 건너뛰십시오.

```
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 `cert` 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 테스트하십시오. <ONTAP Management LIF> 및 <vserver name>를 Management LIF IP 및 SVM 이름으로 바꾸십시오. 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. 인증서, 키 및 신뢰할 수 있는 CA 인증서를 Base64로 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...Vaaalllluuuueeee",
"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 변경하세요

기존 백엔드를 업데이트하여 다른 인증 방법을 사용하거나 자격 증명을 교체할 수 있습니다. 이 기능은 양방향으로 작동합니다. 사용자 이름/암호를 사용하는 백엔드를 인증서를 사용하는 방식으로 업데이트할 수 있고, 인증서를 사용하는 백엔드를 사용자 이름/암호 기반으로 업데이트할 수도 있습니다. 이렇게 하려면 기존 인증 방법을 제거하고 새 인증 방법을 추가해야 합니다. 그런 다음 필요한 매개변수가 포함된 업데이트된 backend.json 파일을 사용하여 `tridentctl update backend`을(를) 실행합니다.

```
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": "password",
"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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

참고

암호를 교체할 때는 스토리지 관리자가 먼저 ONTAP에서 사용자의 암호를 업데이트해야 합니다. 그 후 백엔드 업데이트를 진행합니다. 인증서를 교체할 때는 사용자에게 여러 개의 인증서를 추가할 수 있습니다. 백엔드를 업데이트하여 새 인증서를 사용하도록 설정한 후, ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스는 중단되지 않으며, 이후에 생성된 볼륨 연결에도 영향을 미치지 않습니다. 백엔드 업데이트가 성공적으로 완료되면 Trident가 ONTAP 백엔드와 통신하여 향후 볼륨 작업을 처리할 수 있음을 의미합니다.

Trident용 사용자 지정 ONTAP 역할 생성

Trident에서 작업을 수행하기 위해 ONTAP 관리자 역할을 사용할 필요가 없도록 최소 권한으로 ONTAP 클러스터 역할을 생성할 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident는 생성한 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

Trident 사용자 지정 역할 생성에 대한 자세한 내용은 "Trident 사용자 지정 역할 생성기"을 참조하십시오.

ONTAP CLI 사용

1. 다음 명령을 사용하여 새 역할을 생성합니다.

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Trident 사용자의 사용자 이름을 생성합니다.

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. 사용자에게 역할 매핑:

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

System Manager 사용

ONTAP System Manager에서 다음 단계를 수행하십시오.

1. 사용자 지정 역할 생성:
   a. 클러스터 수준에서 사용자 지정 역할을 생성하려면 * Cluster > Settings * 를 선택합니다.

      (또는) SVM 수준에서 사용자 지정 역할을 생성하려면 *스토리지 > 스토리지 VM > `required SVM`> 설정 > 사용자 및 역할*을 선택하십시오.

   b. 사용자 및 역할 옆에 있는 화살표 아이콘(→)을 선택합니다.
   c. 역할 에서 **+**추가 를 선택합니다.
   d. 역할에 대한 규칙을 정의하고 *저장*을 클릭합니다.
2. **Trident** 사용자에게 역할 매핑: + 사용자 및 역할 페이지에서 다음 단계를 수행합니다.
   a. 사용자 아래에서 추가 아이콘 *+*를 선택합니다.
   b. 필요한 사용자 이름을 선택하고 역할 드롭다운 메뉴에서 역할을 선택합니다.
   c. *저장*을 클릭합니다.

자세한 내용은 다음 페이지를 참조하십시오.

- "ONTAP 관리를 위한 사용자 지정 역할" 또는 "사용자 지정 역할 정의"
- "역할 및 사용자 작업"

NFS 엑스포트 정책 관리

Trident는 NFS 엑스포트 정책을 사용하여 프로비저닝하는 볼륨에 대한 액세스를 제어합니다.

Trident는 엑스포트 정책 작업 시 두 가지 옵션을 제공합니다.

- Trident는 엑스포트 정책을 동적으로 관리할 수 있습니다. 이 운영 모드에서 스토리지 관리자는 허용 가능한 IP 주소를 나타내는 CIDR 블록 목록을 지정합니다. Trident는 게시 시점에 이러한 범위에 속하는 해당 노드 IP를 엑스포트 정책에 자동으로 추가합니다. 또는 CIDR이 지정되지 않은 경우, 볼륨이 게시될 노드에서 발견된 모든 글로벌 범위 유니캐스트 IP가 엑스포트 정책에 추가됩니다.
- 스토리지 관리자는 엑스포트 정책을 생성하고 규칙을 수동으로 추가할 수 있습니다. Trident는 구성 파일에 다른 엑스포트 정책 이름이 지정되지 않은 경우 기본 엑스포트 정책을 사용합니다.

엑스포트 정책을 동적으로 관리합니다

Trident는 ONTAP 백엔드에 대한 엑스포트 규칙을 동적으로 관리할 수 있는 기능을 제공합니다. 이를 통해 스토리지 관리자는 명시적인 규칙을 수동으로 정의하는 대신 워커 노드 IP에 대해 허용 가능한 주소 공간을 지정할 수 있습니다. 이는 엑스포트 규칙 관리를 크게 간소화하며, 엑스포트 규칙을 수정할 때 더 이상 스토리지 클러스터에서 수동으로 개입할 필요가 없습니다. 또한, 이 기능을 통해 볼륨을 마운트하고 지정된 IP 범위 내에 있는 워커 노드만 스토리지 클러스터에 액세스할 수 있도록 제한하여 세밀하고 자동화된 관리를 지원합니다.

참고	동적 엑스포트 규칙을 사용할 때는 네트워크 주소 변환(NAT)을 사용하지 마십시오. NAT를 사용하면 스토리지 컨트롤러가 실제 IP 주소가 아닌 프런트엔드 NAT 주소를 인식하게 되므로 엑스포트 규칙에서 일치하는 항목이 없으면 액세스가 거부됩니다.

예

반드시 사용해야 하는 구성 옵션이 두 가지 있습니다. 다음은 백엔드 정의 예시입니다.

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
  - 192.168.0.0/24
autoExportPolicy: true
```

참고	이 기능을 사용할 때는 SVM의 루트 접합부에 노드 CIDR 블록을 허용하는 엑스포트 규칙(예: 기본 엑스포트 정책)이 포함된 엑스포트 정책이 미리 생성되어 있는지 확인해야 합니다. 항상 NetApp 권장 모범 사례에 따라 Trident 전용 SVM을 사용하십시오.

다음은 위의 예를 사용하여 이 기능이 작동하는 방식에 대한 설명입니다.

- `autoExportPolicy`이(가) `true`(으)로 설정되어 있습니다. 이는 Trident가 `svm1` SVM에 대해 이 백엔드로 프로비저닝된 각 볼륨에 대한 엑스포트 정책을 생성하고 `autoexportCIDRs` 주소 블록을 사용하여 규칙 추가 및 삭제를 처리함을 나타냅니다. 볼륨이 노드에 연결될 때까지 해당 볼륨은 원치 않는 액세스를 방지하기 위해 규칙이 없는 빈 엑스포트 정책을 사용합니다. 볼륨이 노드에 게시되면 Trident는 지정된 CIDR 블록 내의 노드 IP를 포함하는 기본 qtree와 동일한 이름의 엑스포트 정책을 생성합니다. 이러한 IP는 상위 FlexVol 볼륨에서 사용하는 엑스포트 정책에도 추가됩니다.

◦ 예를 들면 다음과 같습니다.

- 백엔드 UUID 403b5326-8482-40db-96d0-d83fb3f4daec
- `autoExportPolicy` 로 설정 `true`
- 스토리지 접두사 `trident`
- PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
- trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c라는 이름의 qtree는 FlexVol에 대한 엑스포트 규칙 `trident-403b5326-8482-40db96d0-d83fb3f4daec`, qtree에 대한 엑스포트 규칙 `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`, 그리고 SVM에 빈 엑스포트 규칙 `trident_empty`을 생성합니다. FlexVol 엑스포트 규칙의 규칙은 qtree 엑스포트 규칙에 포함된 모든 규칙의 상위 집합입니다. 빈 엑스포트 규칙은 연결되지 않은 볼륨에서 재사용됩니다.

- `autoExportCIDRs`에는 주소 블록 목록이 포함되어 있습니다. 이 필드는 선택 사항이며 기본값은 ["0.0.0.0/0", "::/0"]입니다. 정의되지 않은 경우 Trident는 게시와 함께 작업자 노드에서 발견된 모든 전역 범위 유니캐스트 주소를 추가합니다.

이 예에서는 `192.168.0.0/24` 주소 공간이 제공됩니다. 이는 게시와 함께 이 주소 범위 내에 속하는 Kubernetes 노드 IP가 Trident가 생성하는 엑스포트 정책에 추가됨을 나타냅니다. Trident가 실행 중인 노드를 등록할 때 해당 노드의 IP 주소를 검색하고 `autoExportCIDRs`에 제공된 주소 블록과 비교합니다. 게시 시점에 IP를 필터링한 후 Trident는 게시 대상 노드의 클라이언트 IP에 대한 엑스포트 정책 규칙을 생성합니다.

> `autoExportPolicy`및 `autoExportCIDRs`를 생성한 후 백엔드에 대해 업데이트할 수 있습니다. 자동으로 관리되는 백엔드에 대해 새 `CIDR`을 추가하거나 기존 `CIDR`을 삭제할 수 있습니다. 기존 연결이 끊어지지 않도록 `CIDR`을 삭제할 때 주의하십시오. 또한 백엔드에 대해 `autoExportPolicy`를 비활성화하고 수동으로 생성된 엑스포트 정책으로 대체하도록 선택할 수 있습니다. 이렇게 하려면 백엔드 구성에서 `exportPolicy` 매개 변수를 설정해야 합니다.

Trident가 백엔드를 생성하거나 업데이트한 후 `tridentctl` 또는 해당 `tridentbackend` CRD를 사용하여 백엔드를 확인할 수 있습니다.

```
./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 제거되면 Trident는 모든 엑스포트 정책을 확인하여 해당 노드에 해당하는 액세스 규칙을 제거합니다. 관리형 백엔드의 엑스포트 정책에서 이 노드 IP를 제거함으로써 Trident는 클러스터의 새 노드에서 이 IP가 재사용되지 않는 한 무단 마운트를 방지합니다.

기존 백엔드의 경우 `tridentctl update backend`로 백엔드를 업데이트하면 Trident에서 엑스포트 정책을 자동으로 관리합니다. 이렇게 하면 필요할 때 백엔드의 UUID와 qtree 이름을 따서 명명된 두 개의 새 엑스포트 정책이 생성됩니다. 백엔드에 있는 볼륨은 마운트 해제 후 다시 마운트되면 새로 생성된 엑스포트 정책을 사용합니다.

참고	자동 관리되는 엑스포트 규칙이 있는 백엔드를 삭제하면 동적으로 생성된 엑스포트 규칙도 삭제됩니다. 백엔드를 다시 생성하면 새 백엔드로 간주되어 새 엑스포트 규칙이 생성됩니다.

실행 중인 노드의 IP 주소가 업데이트되면 해당 노드에서 Trident Pod를 재시작해야 합니다. 그러면 Trident는 관리하는 백엔드에 대한 엑스포트 정책을 업데이트하여 이 IP 변경 사항을 반영합니다.

SMB 볼륨 프로비저닝 준비

약간의 추가 준비 작업을 거치면 `ontap-nas` 드라이버를 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다.

경고	ONTAP 온프레미스 클러스터용 `ontap-nas-economy` SMB 볼륨을 생성하려면 SVM에서 NFS 및 SMB/CIFS 프로토콜을 모두 구성해야 합니다. 이러한 프로토콜 중 하나라도 구성하지 않으면 SMB 볼륨 생성이 실패합니다.

참고	`autoExportPolicy`는 SMB 볼륨에서 지원되지 않습니다.

시작하기 전에

SMB 볼륨을 프로비저닝하기 전에 다음 사항을 충족해야 합니다.

- Linux 컨트롤러 노드와 Windows Server 2022를 실행하는 하나 이상의 Windows 워커 노드로 구성된 Kubernetes 클러스터입니다. Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Active Directory 자격 증명이 포함된 Trident 비밀 키가 하나 이상 필요합니다. 비밀 키를 생성하려면 `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Windows 서비스로 구성된 CSI 프록시입니다. `csi-proxy`를 구성하려면 Windows에서 실행되는 Kubernetes 노드에 대한 "GitHub: CSI 프록시" 또는 "GitHub: Windows용 CSI 프록시"를 참조하십시오.

단계

1. 온프레미스 ONTAP의 경우 선택적으로 SMB 공유를 생성하거나 Trident가 대신 생성해 줄 수 있습니다.

   참고 | Amazon FSx for ONTAP에는 SMB 공유가 필요합니다.

   SMB 관리자 공유는 "Microsoft Management Console" 공유 폴더 스냅인을 사용하거나 ONTAP CLI를 사용하는 두 가지 방법으로 생성할 수 있습니다. ONTAP CLI를 사용하여 SMB 공유를 생성하려면 다음을 수행합니다.

   a. 필요한 경우 공유에 대한 디렉터리 경로 구조를 생성하십시오.

      `vserver cifs share create` 명령은 공유 생성 시 -path 옵션에 지정된 경로를 확인합니다. 지정된 경로가 존재하지 않으면 명령이 실패합니다.

   b. 지정된 SVM과 연결된 SMB 공유를 생성합니다.

      ```
      vserver cifs share create -vserver vserver_name -share-name
      share_name -path path [-share-properties share_properties,...]
      [other_attributes] [-comment text]
      ```

   c. 공유가 생성되었는지 확인합니다.

      ```
      vserver cifs share show -share-name share_name
      ```

      참고 | 자세한 내용은 "SMB 공유 생성"를 참조하십시오.

2. 백엔드를 생성할 때 SMB 볼륨을 지정하려면 다음을 구성해야 합니다. 모든 FSx for ONTAP 백엔드 구성 옵션에 대한 자세한 내용은 "FSx for ONTAP 구성 옵션 및 예"를 참조하십시오.

매개변수	설명	예
smbShare	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console 또는 ONTAP CLI를 사용하여 생성한 SMB 공유의 이름, Trident가 SMB 공유를 생성할 수 있도록 허용하는 이름, 또는 볼륨에 대한 공통 공유 액세스를 방지하려면 매개 변수를 비워 둘 수 있습니다. 이 매개 변수는 온프레미스 ONTAP의 경우 선택 사항입니다. 이 매개 변수는 Amazon FSx for ONTAP 백엔드에 필요하며 비워 둘 수 없습니다.	smb-share
nasType	* `smb`로 설정해야 합니다.* null인 경우 기본값은 `nfs`입니다.	smb
securityStyle	새 볼륨에 대한 보안 스타일입니다. **SMB** 볼륨의 경우 **`ntfs`** 또는 **`mixed`**로 설정해야 합니다**.**	ntfs 또는 mixed SMB 볼륨의 경우
unixPermissions	새 볼륨 모드입니다. **SMB** 볼륨의 경우 비워 두어야 합니다**.**	""

보안 **SMB** 활성화

25.06 릴리스부터 NetApp Trident는 ontap-nas 및 ontap-nas-economy 백엔드를 사용하여 생성된 SMB 볼륨의 보안 프로비저닝을 지원합니다. 보안 SMB가 활성화되면 액세스 제어 목록(ACL)을 사용하여 Active Directory(AD) 사용자 및 사용자 그룹에 SMB 공유에 대한 제어된 액세스를 제공할 수 있습니다.

기억해야 할 사항

- ontap-nas-economy 볼륨 가져오기는 지원되지 않습니다.
- ontap-nas-economy 볼륨의 경우 읽기 전용 클론만 지원됩니다.
- Secure SMB가 활성화된 경우 Trident는 백엔드에 언급된 SMB 공유를 무시합니다.
- PVC 주석, 스토리지 클래스 주석 및 백엔드 필드를 업데이트해도 SMB 공유 ACL은 업데이트되지 않습니다.
- 클론 PVC의 주석에 지정된 SMB 공유 ACL은 소스 PVC의 ACL보다 우선합니다.
- 보안 SMB를 활성화할 때 유효한 AD 사용자를 제공해야 합니다. 유효하지 않은 사용자는 ACL에 추가되지 않습니다.
- 백엔드, 스토리지 클래스 및 PVC에 동일한 AD 사용자를 지정하고 각기 다른 권한을 부여하는 경우 권한 우선순위는 PVC, 스토리지 클래스, 백엔드 순이 됩니다.
- 보안 SMB는 ontap-nas 관리형 볼륨 가져오기에 대해 지원되며, 관리되지 않는 볼륨 가져오기에는 적용되지 않습니다.

단계

1. 다음 예시와 같이 TridentBackendConfig에서 adAdminUser를 지정하십시오:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. 스토리지 클래스에 주석을 추가합니다.

`trident.netapp.io/smbShareAdUser` 어노테이션을 스토리지 클래스에 추가하여 안전한 SMB를 확실하게 활성화하십시오. 어노테이션 `trident.netapp.io/smbShareAdUser`에 지정된 사용자 값은 `smbcreds` 시크릿에 지정된 사용자 이름과 동일해야 합니다. `smbShareAdUserPermission`에 대해 다음 중 하나를 선택할 수 있습니다: `full_control`, `change` 또는 `read`. 기본 권한은 `full_control`입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. PVC를 생성합니다.

다음 예제는 PVC를 생성합니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

ONTAP NAS 구성 옵션 및 예

Trident 설치 환경에서 ONTAP NAS 드라이버를 생성하고 사용하는 방법을 알아보세요. 이 섹션에서는 백엔드 구성 예제와 백엔드를 StorageClasses에 매핑하는 방법에 대한 세부 정보를 제공합니다. 25.10 릴리스부터 NetApp Trident는 "NetApp AFX 스토리지 시스템"을(를) 지원합니다. NetApp AFX 스토리지 시스템은 스토리지 계층 구현에서 다른 ONTAP 기반 시스템(ASA, AFF, FAS)과 다릅니다.

참고	`ontap-nas` 드라이버(NFS 프로토콜 사용)만 NetApp AFX 시스템에서 지원되며 SMB 프로토콜은 지원되지 않습니다.

백엔드 **configuration** 옵션

Trident 백엔드 구성에서 시스템이 NetApp AFX 스토리지 시스템임을 지정할 필요가 없습니다. `ontap-nas`을(를) `storageDriverName`(으)로 선택하면 Trident가 AFX 스토리지 시스템을 자동으로 감지합니다. 일부 백엔드 구성 매개변수는 AFX 스토리지 시스템에는 적용되지 않습니다.

다음 표는 백엔드 구성 옵션을 보여줍니다.

매개변수	설명	기본값
`version`		항상 1

매개변수	설명	기본값
`storageDriverName`	스토리지 드라이버의 이름 참고 NetApp AFX 시스템의 경우 `ontap-nas`만 지원됩니다.	`ontap-nas`, `ontap-nas-economy` 또는 `ontap-nas-flexgroup`
`backendName`	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소입니다. 정규화된 도메인 이름(FQDN)도 지정할 수 있습니다. Trident를 IPv6 플래그를 사용하여 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`와 같이 대괄호로 정의해야 합니다. 원활한 MetroCluster 전환을 위해서는 MetroCluster 예시을 참조하십시오.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	프로토콜 LIF의 IP 주소입니다. NetApp에서는 `dataLIF`을 지정하는 것을 권장합니다. 지정하지 않으면 Trident는 SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하여 여러 dataLIF에 걸쳐 로드 밸런싱을 수행하는 라운드 로빈 DNS를 구성할 수 있습니다. 초기 설정 후 변경할 수 있습니다. 다음을 참조하십시오. . Trident를 IPv6 플래그를 사용하여 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`와 같이 대괄호로 정의해야 합니다. **MetroCluster**의 경우 생략합니다. MetroCluster 예시를 참조하십시오.	지정된 주소 또는 지정되지 않은 경우 SVM에서 파생(권장하지 않음)
`svm`	사용할 스토리지 가상 머신 **MetroCluster**의 경우 생략 MetroCluster 예시를 참조하십시오.	SVM `managementLIF`이 지정된 경우 파생됩니다
`autoExportPolicy`	자동 엑스포트 정책 생성 및 업데이트를 활성화합니다[Boolean]. `autoExportPolicy` 및 `autoExportCIDRs` 옵션을 사용하면 Trident가 내보내기 정책을 자동으로 관리할 수 있습니다.	거짓
`autoExportCIDRs`	`autoExportPolicy`이 활성화된 경우 Kubernetes 노드 IP를 필터링하는 데 사용할 CIDR 목록입니다. `autoExportPolicy` 및 `autoExportCIDRs` 옵션을 사용하면 Trident가 내보내기 정책을 자동으로 관리할 수 있습니다.	["0.0.0.0/0", "::/0"]`
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`clientCertificate`	클라이언트 인증서의 Base64 인코딩 값입니다. 인증서 기반 인증에 사용됩니다	""
`clientPrivateKey`	클라이언트 개인 키를 Base64로 인코딩한 값입니다. 인증서 기반 인증에 사용됩니다	""
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 선택 사항입니다. 인증서 기반 인증에 사용됩니다	""

매개변수	설명	기본값
`username`	클러스터/SVM에 연결하는 데 사용할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대한 자세한 내용은 "Active Directory 자격 증명을 사용하여 Trident를 백엔드 SVM에 인증합니다"을(를) 참조하십시오.	
`password`	클러스터/SVM에 연결하는 데 사용되는 암호입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대한 자세한 내용은 "Active Directory 자격 증명을 사용하여 Trident를 백엔드 SVM에 인증합니다"을(를) 참조하십시오.	
`storagePrefix`	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 설정 후에는 업데이트할 수 없습니다 참고: ontap-nas-economy를 사용하고 storagePrefix가 24자 이상인 경우, qtree에는 스토리지 접두사가 포함되지 않지만 볼륨 이름에는 포함됩니다.	"Trident"
`aggregate`	프로비저닝용 애그리게이트(선택 사항, 설정된 경우 SVM에 할당해야 함). `ontap-nas-flexgroup` 드라이버의 경우 이 옵션은 무시됩니다. 할당하지 않으면 사용 가능한 애그리게이트 중 하나를 사용하여 FlexGroup 볼륨을 프로비저닝할 수 있습니다. 참고: SVM에서 애그리게이트가 업데이트되면 Trident 컨트롤러를 재시작하지 않고도 SVM을 폴링하여 Trident에 자동으로 업데이트됩니다. Trident에서 볼륨을 프로비저닝하도록 특정 애그리게이트를 구성한 경우, 해당 애그리게이트의 이름이 변경되거나 SVM에서 이동되면 SVM 애그리게이트를 폴링하는 동안 Trident에서 백엔드가 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 전환하려면 SVM에 있는 애그리게이트로 변경하거나 해당 애그리게이트를 완전히 제거해야 합니다. **AFX** 스토리지 시스템에는 지정하지 마십시오.	""
`limitAggregateUsage`	사용량이 이 비율을 초과하면 프로비저닝이 실패합니다. **Amazon FSx for ONTAP**에는 적용되지 않습니다. **AFX** 스토리지 시스템에는 지정하지 마십시오.	"" (기본적으로 적용되지 않음)

매개변수	설명	기본값
flexgroupAggregateList	프로비저닝을 위한 애그리게이트 목록(선택 사항, 설정된 경우 SVM에 할당되어야 함). SVM에 할당된 모든 애그리게이트는 FlexGroup 볼륨을 프로비저닝하는 데 사용됩니다. **ontap-nas-flexgroup** 스토리지 드라이버에서 지원됩니다. 참고 SVM에서 애그리게이트 목록이 업데이트되면 Trident Controller를 재시작하지 않고도 SVM을 폴링하여 Trident의 목록이 자동으로 업데이트됩니다. Trident에서 볼륨을 프로비저닝하도록 특정 애그리게이트 목록을 구성한 경우, 해당 애그리게이트 목록의 이름이 변경되거나 SVM에서 이동되면 SVM 애그리게이트를 폴링하는 동안 Trident의 백엔드 상태가 실패로 전환됩니다. 백엔드를 다시 온라인 상태로 전환하려면 애그리게이트 목록을 SVM에 있는 다른 목록으로 변경하거나 완전히 제거해야 합니다.	""
`limitVolumeSize`	요청된 볼륨 크기가 이 값을 초과하면 프로비저닝이 실패합니다.	"" (기본적으로 적용되지 않음)
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예, {"api":false, "method":true} 문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 `debugTraceFlags` 사용하지 마십시오.	null
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 `nfs`, `smb` 또는 null입니다. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다. **AFX** 스토리지 시스템의 경우 지정된 경우 항상 `**nfs**`로 설정하십시오.	`nfs`
`nfsMountOptions`	쉼표로 구분된 NFS 마운트 옵션 목록입니다. Kubernetes 영구 볼륨의 마운트 옵션은 일반적으로 스토리지 클래스에 지정되지만, 스토리지 클래스에 마운트 옵션이 지정되지 않은 경우 Trident는 스토리지 백엔드의 구성 파일에 지정된 마운트 옵션을 사용합니다. 스토리지 클래스 또는 구성 파일에 마운트 옵션이 지정되지 않은 경우 Trident는 연결된 영구 볼륨에 마운트 옵션을 설정하지 않습니다.	""
`qtreesPerFlexvol`	FlexVol당 최대 Qtree 수는 [50, 300] 범위 내에 있어야 합니다	"200"
`smbShare`	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console 또는 ONTAP CLI를 사용하여 생성한 SMB 공유의 이름, Trident가 SMB 공유를 생성할 수 있도록 허용하는 이름, 또는 볼륨에 대한 공통 공유 액세스를 방지하려면 매개 변수를 비워 둘 수 있습니다. 이 매개 변수는 온프레미스 ONTAP의 경우 선택 사항입니다. 이 매개 변수는 Amazon FSx for ONTAP 백엔드에 필요하며 비워 둘 수 없습니다.	`smb-share`

매개변수	설명	기본값
useREST	ONTAP REST API를 사용하기 위한 부울 매개 변수입니다. useREST`로 설정하면 `true Trident는 ONTAP REST API를 사용하여 백엔드와 통신하고, false`로 설정하면 Trident는 ONTAPI(ZAPI) 호출을 사용하여 백엔드와 통신합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한 사용되는 ONTAP 로그인 역할은 `ontapi 애플리케이션에 대한 액세스 권한이 있어야 합니다. 이는 사전 정의된 vsadmin 및 cluster-admin 역할로 충족됩니다. Trident 24.06 릴리스 및 ONTAP 9.15.1 이상부터 `useREST`는 기본적으로 `true`로 설정되며, ONTAPI(ZAPI) 호출을 사용하려면 `useREST`를 `false`로 변경하십시오. **AFX** 스토리지 시스템의 경우 지정된 경우 항상 **`true`**로 설정하십시오.	ONTAP 9.15.1 이상의 경우 true, 그렇지 않은 경우 false.
limitVolumePoolSize	ontap-nas-economy 백엔드에서 Qtree를 사용할 때 요청할 수 있는 최대 FlexVol 크기입니다.	"" (기본적으로 적용되지 않음)
denyNewVolumePools	`ontap-nas-economy`백엔드가 Qtree를 포함할 새 FlexVol 볼륨을 생성하지 못하도록 제한합니다. 기존 Flexvol만 새 PV 프로비저닝에 사용됩니다.	
adAdminUser	SMB 공유에 대한 전체 액세스 권한이 있는 Active Directory 관리자 사용자 또는 사용자 그룹입니다. 이 매개 변수를 사용하여 SMB 공유에 대한 전체 제어 권한이 있는 관리자 권한을 제공합니다.	

볼륨 프로비저닝을 위한 백엔드 구성 옵션

구성의 defaults 섹션에서 이러한 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. 예를 들어, 아래 구성 예를 참조하십시오.

매개변수	설명	기본값
spaceAllocation	Qtree에 대한 공간 할당	"true"
spaceReserve	공간 예약 모드, "none"(씬) 또는 "volume"(씩)	"없음"
snapshotPolicy	사용할 스냅샷 정책	"없음"
qosPolicy	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드당 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하십시오	""
adaptiveQosPolicy	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀/백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하십시오. ontap-nas-economy에서는 지원되지 않습니다.	""
snapshotReserve	스냅샷용으로 예약된 볼륨의 비율	`snapshotPolicy`이(가) "none"이면 "0", 그렇지 않으면 ""
splitOnClone	생성 시 상위 항목에서 클론 분할	"false"

매개변수	설명	기본값
`encryption`	새 볼륨에서 NetApp Volume Encryption(NVE)을 활성화합니다. 기본값은 `false`입니다. 이 옵션을 사용하려면 클러스터에서 NVE 라이센스가 있고 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하십시오. "Trident가 NVE 및 NAE와 작동하는 방식"	"false"
`tieringPolicy`	계층화 정책에서 "none"을 사용합니다.	
`unixPermissions`	새 볼륨용 모드	NFS 볼륨의 경우 "777", SMB 볼륨의 경우 비어 있음(해당 없음)
`snapshotDir`	`.snapshot` 디렉터리에 대한 액세스를 제어합니다	`true`, `false` (명시적으로 설정).
`exportPolicy`	사용할 엑스포트 정책	"default"
`securityStyle`	새 볼륨에 대한 보안 스타일입니다. NFS는 `mixed` 및 `unix` 보안 스타일을 지원합니다. SMB는 `mixed` 및 `ntfs` 보안 스타일을 지원합니다.	NFS 기본값은 `unix`입니다. SMB 기본값은 `ntfs`입니다.
`nameTemplate`	사용자 지정 볼륨 이름을 생성하기 위한 템플릿입니다.	""

참고 Trident에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유하지 않는 QoS 정책 그룹을 사용하고 각 구성 요소에 개별적으로 정책 그룹을 적용해야 합니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 대한 상한선을 적용합니다.

볼륨 프로비저닝 예

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-nas` 및 `ontap-nas-flexgroups`의 경우, Trident는 이제 FlexVol이 snapshotReserve 비율과 PVC에 맞게 올바르게 크기가 지정되도록 새로운 계산 방식을 사용합니다. 사용자가 PVC를 요청하면, Trident는 새로운 계산을 사용하여 더 많은 공간을 가진 원래 FlexVol을 생성합니다. 이 계산은 사용자가 PVC에서 요청한 만큼의 쓰기 가능한 공간을 받을 수 있도록 하며, 요청한 것보다 적은 공간을 제공하지 않습니다. v21.07 이전에는 사용자가 PVC(예: 5GiB)를 요청하고 snapshotReserve를 50퍼센트로 설정하면, 쓰기 가능한 공간은 2.5GiB만 제공되었습니다. 이는 사용자가 요청한 것이 전체 볼륨이고 `snapshotReserve`가 그 전체의 비율이기 때문입니다. Trident 21.07부터는 사용자가 요청하는 것이 쓰기 가능한 공간이며, Trident는 `snapshotReserve` 숫자를 전체 볼륨의 비율로 정의합니다. 이 내용은 `ontap-nas-economy`에는 적용되지 않습니다. 다음 예제를 통해 이 동작 방식을 확인할 수 있습니다:

계산 방법은 다음과 같습니다.

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

snapshotReserve = 50%이고 PVC 요청 = 5GiB인 경우, 전체 볼륨 크기는 5/.5 = 10GiB이고 사용 가능한 크기는

5GiB로, 사용자가 PVC 요청에서 요청한 크기입니다. `volume show` 명령은 다음 예와 유사한 결과를 표시해야 합니다.

이전 설치의 기존 백엔드는 Trident를 업그레이드할 때 위에서 설명한 대로 볼륨을 프로비저닝합니다. 업그레이드 전에 생성한 볼륨의 경우 변경 사항이 적용되도록 크기를 조정해야 합니다. 예를 들어, `snapshotReserve=50`이전에 2GiB PVC로 설정된 볼륨은 1GiB의 쓰기 공간을 제공합니다. 예를 들어 볼륨 크기를 3GiB로 조정하면 애플리케이션은 6GiB 볼륨에서 3GiB의 쓰기 공간을 사용할 수 있습니다.

최소 구성 예

다음 예시들은 대부분의 매개변수를 기본값으로 유지하는 기본 구성을 보여줍니다. 이것이 백엔드를 정의하는 가장 쉬운 방법입니다.

참고	Amazon FSx on NetApp ONTAP에서 Trident를 사용하는 경우 LIF에 대해 IP 주소 대신 DNS 이름을 지정하는 것이 좋습니다.

ONTAP NAS 이코노미 예

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

ONTAP NAS FlexGroup 예

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

MetroCluster 예시

"SVM 복제 및 복구" 중에 스위치오버 및 스위치백 후 백엔드 정의를 수동으로 업데이트하지 않아도 되도록 백엔드를 구성할 수 있습니다.

원활한 전환 및 복귀를 위해 `managementLIF`을 사용하여 SVM을 지정하고 `dataLIF` 및 `svm` 매개 변수를 생략하십시오. 예를 들면 다음과 같습니다.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

SMB 볼륨 예

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

인증서 기반 인증 예

이는 최소한의 백엔드 구성 예시입니다. `clientCertificate`, `clientPrivateKey` 및 `trustedCACertificate`(신뢰할 수 있는 CA를 사용하는 경우 선택 사항)는 `backend.json`에 입력되며 클라이언트 인증서, 개인 키 및 신뢰할 수 있는 CA 인증서의 base64 인코딩된 값을 각각 사용합니다.

```
---
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_
```

자동 엑스포트 정책 예

이 예제는 Trident가 동적 내보내기 정책을 사용하여 내보내기 정책을 자동으로 생성하고 관리하도록 지시하는 방법을 보여줍니다. 이는 `ontap-nas-economy` 및 `ontap-nas-flexgroup` 드라이버에 대해 동일하게 작동합니다.

```
---
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: password
nfsMountOptions: nfsvers=4
```

IPv6 주소 예

이 예시는 IPv6 주소를 사용하는 `managementLIF` 방법을 보여줍니다.

```
---
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: password
```

SMB 볼륨을 사용하는 **Amazon FSx for ONTAP** 예

`smbShare` 매개 변수는 SMB 볼륨을 사용하는 FSx for ONTAP에 필요합니다.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

nameTemplate을 사용한 백엔드 구성 예

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

가상 풀이 있는 백엔드의 예

아래에 표시된 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 `spaceReserve`없음, `spaceAllocation`false, `encryption`false와 같은 특정 기본값이 설정되어 있습니다. 가상 풀은 스토리지 섹션에서 정의됩니다.

Trident는 "설명" 필드에 프로비저닝 레이블을 설정합니다. 설명은 FlexVol의 경우 `ontap-nas` 또는 FlexGroup의 경우 `ontap-nas-flexgroup`에 설정됩니다. Trident는 프로비저닝 시 가상 풀에 있는 모든 레이블을 스토리지 볼륨으로 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

이 예시에서 일부 스토리지 풀은 자체 `spaceReserve`, `spaceAllocation` 및 `encryption` 값을 설정하고, 일부 풀은 기본값을 재정의합니다.

ONTAP NAS 예

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
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:
      department: legal
      creditpoints: "5000"
    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 예

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
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 이코노미 예

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
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:
      protection: bronze
      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 StorageClasses에 매핑합니다

다음 StorageClass 정의는 가상 풀이 있는 백엔드의 예를 참조합니다. `parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 지정합니다. 볼륨은 선택된 가상 풀에 정의된 속성을 갖게 됩니다.

- `protection-gold` StorageClass는 `ontap-nas-flexgroup` 백엔드의 첫 번째 및 두 번째 가상 풀에 매핑됩니다. 이 두 풀만 골드 레벨 보호를 제공합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- `protection-not-gold` StorageClass는 `ontap-nas-flexgroup` 백엔드의 세 번째 및 네 번째 가상 풀에 매핑됩니다. 이 두 풀만이 골드 등급 이외의 보호 수준을 제공합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- `app-mysqldb` StorageClass는 `ontap-nas` 백엔드의 네 번째 가상 풀에 매핑됩니다. 이 풀은 mysqldb 유형 애플리케이션에 대한 스토리지 풀 구성을 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- `protection-silver-creditpoints-20k` StorageClass는 `ontap-nas-flexgroup` 백엔드의 세 번째 가상 풀에 매핑됩니다. 이 풀은 실버 등급 보호와 20000 크레딧 포인트를 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- `creditpoints-5k` StorageClass는 `ontap-nas` 백엔드의 세 번째 가상 풀과 `ontap-nas-economy` 백엔드의 두 번째 가상 풀에 매핑됩니다. 5000 크레딧 포인트를 제공하는 풀은 이 두 가지뿐입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Trident는 어떤 가상 풀이 선택될지 결정하고 스토리지 요구사항이 충족되도록 보장합니다.

초기 구성 후 업데이트 `dataLIF`

초기 구성 후 다음 명령을 실행하여 업데이트된 dataLIF가 포함된 새 백엔드 JSON 파일을 제공함으로써 dataLIF를 변경할 수 있습니다.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

참고	PVC가 하나 이상의 Pod에 연결된 경우, 새 dataLIF가 적용되려면 해당 Pod를 모두 종료한 다음 다시 시작해야 합니다.

보안 **SMB** 예

ontap-nas 드라이버를 사용한 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

ontap-nas-economy 드라이버를 사용한 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

스토리지 풀을 사용한 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

ontap-nas 드라이버를 사용한 스토리지 클래스 예제

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

참고 | `annotations`을(를) 추가하여 보안 SMB를 활성화해야 합니다. 백엔드 또는 PVC에 설정된 구성과 관계없이 어노테이션이 없으면 보안 SMB가 작동하지 않습니다.

ontap-nas-economy 드라이버를 사용한 스토리지 클래스 예

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

단일 **AD** 사용자를 사용한 **PVC** 예

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

여러 **AD** 사용자를 포함하는 **PVC** 예

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP와 함께 Trident 사용

"Amazon FSx for NetApp ONTAP"는 NetApp ONTAP 스토리지 운영 체제를 기반으로 하는 파일 시스템을 실행하는 완전 관리형 AWS 서비스입니다. AWS의 확장성과 운영 간편성과 함께 ONTAP의 기능, 성능 및 관리를 제공합니다. 파일 시스템은 Amazon FSx의 기본 리소스이며 온프레미스 ONTAP 클러스터와 유사합니다. 각 파일 시스템에는 하나 이상의 스토리지 가상 머신(SVM)이 포함되고, 각 SVM에는 파일과 디렉터리를 저장하는 하나 이상의 볼륨이 포함됩니다. 이 통합을 통해 Amazon Elastic Kubernetes Service(EKS)에서 실행되는 Kubernetes 클러스터는 블록 및 파일 워크로드를 위해 ONTAP 기반 영구 볼륨을 프로비저닝할 수 있습니다.

요구 사항

"Trident 요구 사항" 외에도 FSx for ONTAP을 Trident와 통합하려면 다음이 필요합니다.

- `kubectl`가 설치된 기존 Amazon EKS 클러스터 또는 자체 관리형 Kubernetes 클러스터.
- 클러스터의 워커 노드에서 연결할 수 있는 기존 Amazon FSx for NetApp ONTAP 파일 시스템 및 스토리지 가상 머신(SVM).
- "NFS 또는 iSCSI"에 대해 준비된 작업자 노드.

> 참고 EKS AMI 유형에 따라 Amazon Linux 및 Ubuntu "Amazon Machine Images"(AMI)에 필요한 노드 준비 단계를 반드시 따르십시오.

고려 사항

- SMB 볼륨:
  - SMB 볼륨은 `ontap-nas` 드라이버를 통해서만 지원됩니다.
  - SMB 볼륨은 Trident EKS 애드온에서 지원되지 않습니다.
  - Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다. 자세한 내용은 "SMB 볼륨 프로비저닝 준비"를 참조하십시오.
- Trident 24.02 이전 버전에서는 자동 백업이 활성화된 Amazon FSx 파일 시스템에 생성된 볼륨을 Trident에서 삭제할 수 없었습니다. Trident 24.02 이상 버전에서 이 문제를 방지하려면 Amazon FSx for NetApp ONTAP의 백엔드 구성 파일에서 `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey`, AWS `secretKey`를 지정하십시오.

> 참고 Trident에 IAM 역할을 지정하는 경우 `apiRegion`, `apiKey` 및 `secretKey` 필드를 Trident에 명시적으로 지정하지 않아도 됩니다. 자세한 내용은 "FSx for ONTAP 구성 옵션 및 예"을 참조하십시오.

Trident SAN/iSCSI 및 **EBS-CSI** 드라이버의 동시 사용

AWS(EKS, ROSA, EC2 또는 기타 인스턴스)에서 ontap-san 드라이버(예: iSCSI)를 사용하려는 경우 노드에 필요한 다중 경로 구성이 Amazon Elastic Block Store(EBS) CSI 드라이버와 충돌할 수 있습니다. 동일한 노드의 EBS 디스크와 충돌 없이 다중 경로가 작동하도록 하려면 다중 경로 설정에서 EBS를 제외해야 합니다. 다음 예는 다중 경로에서 EBS 디스크를 제외하면서 필요한 Trident 설정을 포함하는 `multipath.conf` 파일입니다.

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

인증

Trident는 두 가지 인증 방식을 제공합니다.

- 자격 증명 기반(권장): 자격 증명을 AWS Secrets Manager에 안전하게 저장합니다. `fsxadmin` 사용자를 파일 시스템에 사용하거나 `vsadmin` 사용자를 SVM에 구성하여 사용할 수 있습니다.

> 경고 Trident는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름의 사용자로 실행되어야 합니다. Amazon FSx for NetApp ONTAP에는 ONTAP `admin` 클러스터 사용자의 제한된 대체 역할을 하는 `fsxadmin` 사용자가 있습니다. Trident와 함께 `vsadmin` 사용을 강력히 권장합니다.

- 인증서 기반: Trident는 SVM에 설치된 인증서를 사용하여 FSx 파일 시스템의 SVM과 통신합니다.

인증 활성화에 대한 자세한 내용은 드라이버 유형에 대한 인증을 참조하십시오.

- "ONTAP NAS 인증"
- "ONTAP SAN 인증"

테스트된 **Amazon Machine Images(AMI)**

EKS 클러스터는 다양한 운영 체제를 지원하지만, AWS는 컨테이너 및 EKS에 최적화된 특정 Amazon Machine Image(AMI)를 제공합니다. 다음 AMI는 NetApp Trident 25.02에서 테스트되었습니다.

AMI	NAS	NAS-이코노미	iSCSI	iSCSI-경제성
AL2023_x86_64_STANDARD	예	예	예	예
AL2_x86_64	예	예	예*	예*
BOTTLEROCKET_x86_64	예**	예	해당 없음	해당 없음
AL2023_ARM_64_STANDARD	예	예	예	예
AL2_ARM_64	예	예	예*	예*
BOTTLEROCKET_ARM_64	예**	예	해당 없음	해당 없음

- * 노드를 재시작하지 않고는 PV를 삭제할 수 없습니다
- ** Trident 버전 25.02에서는 NFSv3와 호환되지 않습니다.

> 참고 원하는 AMI가 이 목록에 없다고 해서 지원되지 않는다는 의미는 아닙니다. 단지 테스트가 완료되지 않았다는 뜻입니다. 이 목록은 정상 작동하는 것으로 확인된 AMI를 안내하기 위한 것입니다.

다음을 사용하여 수행된 테스트**:**

- EKS 버전: 1.32
- 설치 방법: Helm 25.06 및 AWS 추가 기능 25.06
- NAS의 경우 NFSv3과 NFSv4.1이 모두 테스트되었습니다.
- SAN의 경우 iSCSI만 테스트되었으며 NVMe-oF는 테스트되지 않았습니다.

수행된 테스트:

- 생성: Storage Class, pvc, pod
- 삭제: pod, pvc(일반, qtree/lun – economy, AWS 백업이 있는 NAS)

자세한 정보 찾기

- "Amazon FSx for NetApp ONTAP 설명서"
- "Amazon FSx for NetApp ONTAP에 대한 블로그 게시물"

IAM 역할 및 AWS Secret을 생성합니다

명시적인 AWS 자격 증명을 제공하는 대신 AWS IAM 역할로 인증하여 Kubernetes Pod가 AWS 리소스에 액세스하도록 구성할 수 있습니다.

참고	AWS IAM 역할을 사용하여 인증하려면 EKS를 사용하여 배포된 Kubernetes 클러스터가 있어야 합니다.

AWS Secrets Manager 시크릿 생성

Trident는 스토리지 관리를 위해 FSx vserver에 대한 API를 발행하므로 자격 증명이 필요합니다. 이러한 자격 증명을 안전하게 전달하는 방법은 AWS Secrets Manager 시크릿을 사용하는 것입니다. 따라서 아직 AWS Secrets Manager 시크릿이 없다면 vsadmin 계정의 자격 증명이 포함된 시크릿을 생성해야 합니다.

이 예제는 Trident CSI 자격 증명을 저장하기 위한 AWS Secrets Manager 시크릿을 생성합니다.

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

IAM 정책 생성

Trident가 제대로 실행되려면 AWS 권한도 필요합니다. 따라서 Trident에 필요한 권한을 부여하는 정책을 생성해야 합니다.

다음 예제는 AWS CLI를 사용하여 IAM 정책을 생성합니다.

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

정책 JSON 예:

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

서비스 계정 연결**(IRSA)**을 위한 **Pod Identity** 또는 **IAM** 역할을 생성합니다.

Kubernetes 서비스 계정이 EKS Pod Identity 또는 IRSA(서비스 계정 연결용 IAM 역할)를 사용하는 AWS Identity and Access Management(IAM) 역할을 수임하도록 구성할 수 있습니다. 해당 서비스 계정을 사용하도록 구성된 모든 Pod는 해당 역할에 액세스 권한이 있는 모든 AWS 서비스에 액세스할 수 있습니다.

Pod Identity

Amazon EKS Pod Identity 연결은 Amazon EC2 인스턴스 프로필이 Amazon EC2 인스턴스에 자격 증명을 제공하는 방식과 유사하게 애플리케이션의 자격 증명을 관리하는 기능을 제공합니다.

EKS 클러스터에 **Pod Identity** 설치:

AWS 콘솔을 사용하거나 다음 AWS CLI 명령을 사용하여 Pod ID를 생성할 수 있습니다.

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

자세한 내용은 "Amazon EKS Pod Identity Agent를 설정합니다"을(를) 참조하십시오.

trust-relationship.json 생성:

EKS 서비스 주체가 Pod Identity에 대한 이 역할을 수임할 수 있도록 trust-relationship.json을 생성합니다. 그런 다음 이 신뢰 정책을 사용하여 역할을 생성합니다.

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

trust-relationship.json 파일:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

IAM 역할에 역할 정책 연결:

이전 단계의 역할 정책을 생성된 IAM 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

포드 **ID** 연결 생성:

IAM 역할과 Trident 서비스 계정(trident-controller) 간에 Pod ID 연결을 생성합니다.

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

서비스 계정 연결을 위한 **IAM** 역할**(IRSA)**

AWS CLI 사용:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

trust-relationship.json 파일**:**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

`trust-relationship.json` 파일에서 다음 값을 업데이트합니다.

- **<account_id>** - AWS 계정 ID
- **<oidc_provider>** - EKS 클러스터의 OIDC입니다. 다음 명령을 실행하여 oidc_provider를 얻을 수 있습니다:

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

IAM 정책을 사용하여 **IAM** 역할을 연결합니다:

역할이 생성되면 이 명령을 사용하여 (위 단계에서 생성한) 정책을 역할에 연결합니다.

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

OICD 공급자가 연결되어 있는지 확인:

OIDC 공급자가 클러스터에 연결되어 있는지 확인하십시오. 다음 명령을 사용하여 확인할 수 있습니다.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

출력이 비어 있는 경우 다음 명령을 사용하여 IAM OIDC를 클러스터에 연결하십시오.

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

eksctl을 사용하는 경우 다음 예제를 사용하여 EKS에서 서비스 계정에 대한 IAM 역할을 생성하십시오.

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

Trident 설치

Trident는 Amazon FSx for NetApp ONTAP 스토리지 관리를 Kubernetes에서 간소화하여

개발자와 관리자가 애플리케이션 배포에 집중할 수 있도록 지원합니다. 다음 방법 중 하나를 사용하여 Trident를 설치할 수 있습니다.

- Helm
- EKS 추가 기능

스냅샷 기능을 사용하려면 CSI 스냅샷 컨트롤러 추가 기능을 설치하십시오. 자세한 내용은 "CSI 볼륨에 대한 스냅샷 기능 활성화"을 참조하십시오.

Helm을 통해 **Trident** 설치

Pod Identity

1. Trident Helm 리포지토리 추가:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 다음 예시를 사용하여 Trident를 설치합니다.

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

`helm list` 명령을 사용하여 이름, 네임스페이스, 차트, 상태, 앱 버전 및 개정 번호와 같은 설치 세부 정보를 검토할 수 있습니다.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                           APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

서비스 계정 연결**(IRSA)**

1. Trident Helm 리포지토리 추가:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 클라우드 공급자 및 클라우드 **ID** 값을 설정합니다.

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

`helm list` 명령을 사용하여 이름, 네임스페이스, 차트, 상태, 앱 버전 및 개정 번호와 같은 설치 세부 정보를 검토할 수 있습니다.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2510.0    25.10.0
```

참고

iSCSI를 사용하려는 경우 클라이언트 시스템에서 iSCSI가 활성화되어 있는지 확인하십시오. AL2023 Worker 노드 OS를 사용하는 경우 helm 설치에 node prep 매개 변수를 추가하여 iSCSI 클라이언트 설치를 자동화할 수 있습니다.

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

EKS 애드온을 통해 **Trident**를 설치하십시오

Trident EKS 애드온에는 최신 보안 패치와 버그 수정 사항이 포함되어 있으며, AWS에서 Amazon EKS와의 호환성을 검증받았습니다. EKS 애드온을 사용하면 Amazon EKS 클러스터의 보안과 안정성을 지속적으로 유지할 수 있으며, 애드온 설치, 구성 및 업데이트에 필요한 작업량을 줄일 수 있습니다.

필수 구성 요소

AWS EKS용 Trident 애드온을 구성하기 전에 다음 사항을 확인하십시오.

- 추가 기능 구독이 포함된 Amazon EKS 클러스터 계정

- AWS Marketplace에 대한 AWS 권한:
  ```
  "aws-marketplace:ViewSubscriptions",
  "aws-marketplace:Subscribe",
  "aws-marketplace:Unsubscribe
  ```
- AMI 유형: Amazon Linux 2(AL2_x86_64) 또는 Amazon Linux 2 Arm(AL2_ARM_64)
- 노드 유형: AMD 또는 ARM
- 기존 Amazon FSx for NetApp ONTAP 파일 시스템

AWS용 **Trident** 추가 기능을 활성화하세요

관리 콘솔

1. Amazon EKS 콘솔을 엽니다 https://console.aws.amazon.com/eks/home#/clusters.
2. 왼쪽 탐색 창에서 *클러스터*를 선택합니다.
3. NetApp Trident CSI 추가 기능을 구성할 클러스터의 이름을 선택하십시오.
4. * 추가 기능 * 을 선택한 다음 * 추가 기능 더 보기 * 를 선택합니다.
5. 추가 기능을 선택하려면 다음 단계를 따르십시오.
   a. **AWS Marketplace** 애드온 섹션으로 스크롤하여 검색 상자에 *"Trident"*를 입력하세요.
   b. Trident by NetApp 상자의 오른쪽 상단에 있는 확인란을 선택하십시오.
   c. * 다음 * 을 선택합니다.
6. 선택한 애드온 구성 설정 페이지에서 다음을 수행하십시오.

   참고 **Pod Identity** 연결을 사용하는 경우 이 단계를 건너뛰십시오.

   a. 사용할 *버전*을 선택합니다.
   b. IRSA 인증을 사용하는 경우 선택적 구성 설정에서 사용 가능한 구성 값을 설정했는지 확인하십시오.
      - 사용할 *버전*을 선택합니다.
      - **Add-on configuration schema***를 따르고 *구성 값 섹션의 **configurationValues** 매개 변수를 이전 단계에서 생성한 role-arn으로 설정합니다(값은 다음 형식이어야 합니다).

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

+
충돌 해결 방법으로 Override를 선택하면 기존 애드온 소프트웨어의 설정 중 하나 이상이 Amazon EKS 애드온 소프트웨어 설정으로 덮어쓰여질 수 있습니다. 이 옵션을 사용하지 않고 기존 설정과 충돌이 발생하면 작업이 실패합니다. 오류 메시지를 사용하여 충돌 문제를 해결할 수 있습니다. 이 옵션을 선택하기 전에 Amazon EKS 애드온 소프트웨어가 사용자가 직접 관리해야 하는 설정을 관리하지 않는지 확인하십시오.

7. * 다음 * 을 선택합니다.
8. 검토 및 추가 페이지에서 * 생성 * 을 선택합니다.

   애드온 소프트웨어 설치가 완료되면 설치된 애드온 소프트웨어를 확인할 수 있습니다.

AWS CLI

1. `add-on.json` 파일 생성:

Pod Identity의 경우 다음 형식을 사용하십시오:

참고 다음을 사용하세요

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

IRSA 인증의 경우 다음 형식을 사용하십시오:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

참고 `<role ARN>`을 이전 단계에서 생성한 역할의 ARN으로 바꿉니다.

2. Trident EKS 애드온을 설치합니다.

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

다음 예시 명령은 Trident EKS 애드온을 설치합니다.

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Trident EKS 애드온 업데이트

관리 콘솔

1. Amazon EKS 콘솔을 엽니다 https://console.aws.amazon.com/eks/home#/clusters.
2. 왼쪽 탐색 창에서 *클러스터*를 선택합니다.
3. NetApp Trident CSI 애드온을 업데이트할 클러스터의 이름을 선택합니다.
4. 애드온 탭을 선택합니다.
5. *Trident by NetApp*을 선택한 다음 *편집*을 선택합니다.
6. **NetApp**에서 **Trident** 구성 페이지에서 다음을 수행합니다.
   a. 사용할 *버전*을 선택합니다.
   b. *선택적 구성 설정*을 확장하고 필요에 따라 수정하십시오.
   c. *변경 사항 저장*을 선택합니다.

AWS CLI

다음 예에서는 EKS 애드온을 업데이트합니다.

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- FSxN Trident CSI 애드온의 현재 버전을 확인하십시오. `my-cluster`를 클러스터 이름으로 바꾸십시오.

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  예시 출력:

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- 이전 단계 출력의 UPDATE AVAILABLE에 반환된 버전으로 애드온 소프트웨어를 업데이트하십시오.

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

`--force` 옵션을 제거하고 Amazon EKS 애드온 설정이 기존 설정과 충돌하는 경우 Amazon EKS 애드온 업데이트가 실패하며 충돌 해결에 도움이 되는 오류 메시지가 표시됩니다. 이 옵션을 지정하기 전에 Amazon EKS 애드온이 관리해야 하는 설정을 관리하지 않는지 확인하십시오. 이 옵션을 사용하면 해당 설정이 덮어쓰여지기 때문입니다. 이 설정의 다른 옵션에 대한 자세한 내용은 link:https://eksctl.io/usage/addons/["애드온"]을 참조하십시오. Amazon EKS Kubernetes 필드 관리에 대한 자세한 내용은 link:https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-field-management.html["Kubernetes 필드 관리"]을 참조하십시오.

Trident EKS 애드온 제거/삭제

Amazon EKS 애드온을 제거하는 두 가지 옵션이 있습니다.

- 클러스터에 애드온 소프트웨어 유지 – 이 옵션을 선택하면 Amazon EKS에서 설정 관리가 제거됩니다. 또한 Amazon EKS에서 업데이트 알림을 보내거나 업데이트를 시작한 후 Amazon EKS 애드온을 자동으로 업데이트하는 기능도 제거됩니다. 하지만 클러스터에 애드온 소프트웨어는 유지됩니다. 이 옵션을 사용하면 애드온이 Amazon EKS 애드온이 아닌 자체 관리형 설치로 작동합니다. 이 옵션을 사용해도 애드온에 대한 다운타임은 발생하지 않습니다. 애드온을 유지하려면 명령에 `--preserve` 옵션을 포함하십시오.
- 클러스터에서 애드온 소프트웨어를 완전히 제거하기 – NetApp에서는 클러스터에 해당 애드온에 의존하는 리소스가 없는 경우에만 Amazon EKS 애드온을 클러스터에서 제거할 것을 권장합니다. 애드온을 제거하려면 `--preserve` 옵션을 `delete` 명령에서 제거하세요.

참고 애드온 소프트웨어에 IAM 계정이 연결되어 있는 경우 해당 IAM 계정은 제거되지 않습니다.

관리 콘솔

1. Amazon EKS 콘솔을 엽니다 https://console.aws.amazon.com/eks/home#/clusters.
2. 왼쪽 탐색 창에서 *클러스터*를 선택합니다.
3. NetApp Trident CSI 애드온을 제거할 클러스터의 이름을 선택하십시오.
4. **Add-ons** 탭을 선택한 다음 *Trident by NetApp*을 선택합니다.
5. *제거*를 선택합니다.
6. **netapp_trident-operator** 제거 확인 대화 상자에서 다음을 수행합니다.
   a. Amazon EKS에서 애드온 설정을 더 이상 관리하지 않도록 하려면 *클러스터에 유지*를 선택하십시오. 클러스터에 애드온 소프트웨어를 유지하여 애드온의 모든 설정을 직접 관리하려면 이 옵션을 선택하십시오.
   b. *netapp_trident-operator*를 입력합니다.
   c. *제거*를 선택합니다.

AWS CLI

`my-cluster`를 클러스터 이름으로 바꾼 다음 명령을 실행합니다.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

eksctl

다음 명령을 실행하면 Trident EKS 추가 기능이 제거됩니다.

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

스토리지 클래스 구성

이 "Kubernetes StorageClass 오브젝트" 설정은 프로비저너를 식별하고 프로비저너에게 볼륨을 프로비저닝하는 방법을 지시합니다. 이 섹션에서는 Trident를 프로비저너로 지정하는 Kubernetes StorageClass 객체를 구성하는 방법을 보여줍니다.

StorageClass 객체를 생성합니다

FSx for ONTAP용 StorageClass를 생성하면 Trident가 백엔드 구성을 자동으로 생성합니다.

참고	스토리지 백엔드를 수동으로 구성하려면 Trident 백엔드와 스토리지 클래스를 별도로 생성하는 방법에 대한 [create-a-kubernetes-storageclass-without-automatic-backend-configuration] 섹션을 참조하십시오.

필수 **StorageClass** 매개변수를 지정하세요

StorageClass를 생성할 때 다음 세 가지 매개 변수를 정의해야 합니다.

매개변수	필수	유형	설명
`fsxFilesystemID`	예	문자열	FSx for NetApp ONTAP 파일 시스템 ID
`storageDriverName`	예	문자열	Trident 스토리지 드라이버(예: `ontap-nas` 또는 `ontap-san`)
`credentialsName`	예	문자열	ONTAP 자격 증명용 FSx가 포함된 Kubernetes Secret의 이름

선택적 매개변수 지정

StorageClass를 통해 선택적 백엔드 매개 변수를 전달할 수 있습니다. StorageClass `parameters` 섹션에서 모든 선택적 값을 문자열로 정의하십시오. 백엔드 매개변수의 전체 목록은 다음을 참조하십시오: "FSx for NetApp ONTAP 백엔드 구성".

StorageClass 구성 파일의 예입니다**.**

다음 예는 자동 백엔드 구성을 트리거하는 StorageClass를 보여줍니다.

YAML

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-fsx-demo
  annotations:
    description: "Demo StorageClass for FSx for NetApp ONTAP"
provisioner: csi.trident.netapp.io
parameters:
  fsxFilesystemID: "fs-0abc123"
  storageDriverName: "ontap-nas"
  credentialsName: trident-fsx-credentials
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

JSON

```json
{
  "apiVersion": "storage.k8s.io/v1",
  "kind": "StorageClass",
  "metadata": {
    "name": "ontap-fsx-demo",
    "annotations": {
      "description": "Demo StorageClass for FSx for NetApp ONTAP"
    }
  },
  "provisioner": "csi.trident.netapp.io",
  "parameters": {
    "fsxFilesystemID": "fs-0abc123",
    "storageDriverName": "ontap-nas",
    "credentialsName": "trident-fsx-credentials"
  },
  "allowVolumeExpansion": true,
  "reclaimPolicy": "Delete",
  "volumeBindingMode": "Immediate"
}
```

StorageClass 생성

구성 파일을 생성한 후 다음 명령을 실행하여 스토리지 클래스를 생성하십시오.

```
kubectl create -f storage-class-ontapnas.yaml
```

이제 Kubernetes와 Trident 모두에서 **basic-csi** 스토리지 클래스가 표시되어야 하며, Trident가 백엔드에서 풀을 검색했을 것입니다.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

StorageClass를 적용하면 Trident가 백엔드를 자동으로 생성합니다. 그런 다음, 이 StorageClass를 참조하는 PersistentVolumeClaims를 생성할 수 있습니다.

백엔드 구성 상태를 확인합니다

Trident는 백엔드 생성 결과를 StorageClass 어노테이션에 기록합니다.

주석	설명
`trident.netapp.io/configuratorStatus`	구성 결과 (`Success` 또는 `Failure`)
`trident.netapp.io/configuratorMessage`	자세한 상태 또는 오류 메시지
`trident.netapp.io/configuratorName`	내부 구성자 리소스의 이름
`trident.netapp.io/managed`	StorageClass가 Trident에 의해 관리됨을 나타냅니다
`trident.netapp.io/additionalStoragePools`	이 백엔드용으로 생성된 스토리지 풀

상태를 확인하려면 다음을 실행합니다.

```
kubectl get storageclass ontap-fsx-demo -o yaml
```

`trident.netapp.io/configuratorStatus`이(가) `Success`(으)로 설정되어 있는지 확인하십시오. 값이 `Failure`인 경우 오류에 대해 `trident.netapp.io/configuratorMessage`을(를) 검토하십시오.

추가 **FSxN** 파일 시스템 추가

동일한 StorageClass를 계속 사용하면서 추가 스토리지 용량이 필요한 경우 추가 FSxN 파일 시스템 ID를 추가하십시오.

StorageClass를 편집하고 다음 주석을 추가하세요.

```
metadata:
  annotations:
    trident.netapp.io/additionalFsxNFileSystemID: '["fs-
xxxxxxxxxxxxxxxxxx"]'
```

변경 사항을 적용하면 Trident가 백엔드 구성을 업데이트하고 StorageClass 어노테이션을 업데이트합니다.

운영상의 고려사항 및 제한사항

- 자동 백엔드 구성이 있는 StorageClass를 삭제하면 일반적으로 연결된 Trident 백엔드도 함께 삭제됩니다. 이로 인해 스토리지 연결이 중단되고 실행 중인 워크로드가 중단될 수 있습니다. 관리형 StorageClass를 삭제하기 전에 영향을 검증하십시오.
- 자동 백엔드 구성은 AWS FSx for NetApp ONTAP에서만 지원됩니다.

자동 백엔드 구성 없이 **Kubernetes StorageClass** 생성

Trident 백엔드와 StorageClass를 별도로 생성하려면 다음 단계를 따르세요.

자동 백엔드 구성 작동 방식 이해

Trident는 StorageClass 정의에서 백엔드 구성을 가져옵니다. StorageClass를 적용하면 Trident는 필수 매개변수를 검증하고 백엔드를 생성한 다음 StorageClass에 상태 주석을 추가합니다.

Trident는 VolumeSnapshotClass를 한 번만 생성합니다. Trident는 이후의 StorageClasses에 대해 동일한 VolumeSnapshotClass를 재사용합니다.

Trident 백엔드를 생성합니다

Trident 백엔드를 생성하려면 JSON 또는 YAML 형식의 구성 파일을 만들어야 합니다. 이 파일에는 사용할 스토리지 유형(NAS 또는 SAN), 파일 시스템, 스토리지를 가져올 SVM, 그리고 인증 방법을 지정해야 합니다. 다음 예는 NAS 기반 스토리지를 정의하고 AWS 시크릿을 사용하여 사용할 SVM에 대한 자격 증명을 저장하는 방법을 보여줍니다.

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

FSx for ONTAP 드라이버 세부 정보

다음 드라이버를 사용하여 Trident를 Amazon FSx for NetApp ONTAP과 통합할 수 있습니다.

드라이버 이름	설명
`ontap-san`	프로비저닝된 각 PV는 자체 Amazon FSx for NetApp ONTAP 볼륨 내의 LUN입니다. 블록 스토리지에 권장됩니다.
`ontap-nas`	프로비저닝된 각 PV는 전체 Amazon FSx for NetApp ONTAP 볼륨입니다. NFS 및 SMB에 권장됩니다.
`ontap-san-economy`	프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP 볼륨당 구성 가능한 LUN 수를 가진 LUN입니다.
`ontap-nas-economy`	프로비저닝된 각 PV는 qtree이며, Amazon FSx for NetApp ONTAP 볼륨당 구성 가능한 수의 qtree가 할당됩니다.
`ontap-nas-flexgroup`	프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP FlexGroup 볼륨입니다.

드라이버 세부 정보는 "NAS 드라이버" 및 "SAN 드라이버"을 참조하십시오.

백엔드 생성

구성 파일을 생성한 후 다음 명령을 실행하여 Trident 백엔드 구성(TBC)을 생성하고 유효성을 검사하십시오.

- YAML 파일에서 Trident 백엔드 구성(TBC)을 생성하고 다음 명령을 실행하십시오.

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Trident 백엔드 구성(TBC)이 성공적으로 생성되었는지 확인합니다.

```
Kubectl get tbc -n trident
```

```
NAME                            BACKEND NAME          BACKEND UUID
PHASE   STATUS

backend-tbc-ontap-nas           tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

다른 구성 옵션에 대한 자세한 내용은 아래 [Backend-advanced-configuration-and-examples] 섹션을 참조하십시오.

자동 백엔드 구성 없이 스토리지 클래스 구성

다음은 Trident 및 FSx for ONTAP에서 사용할 스토리지 클래스 구성의 예입니다.

NFS용 스토리지 클래스

이 예제를 사용하여 NFS를 사용하는 볼륨에 대한 StorageClass를 설정할 수 있습니다(전체 속성 목록은 아래 Trident 속성 섹션을 참조하십시오).

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

iSCSI용 스토리지 클래스

iSCSI를 사용하여 볼륨에 대한 StorageClass를 설정하려면 다음 예를 사용하십시오.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

NFSv3 및 AWS Bottlerocket을 사용하는 스토리지 클래스

AWS Bottlerocket에서 NFSv3 볼륨을 프로비저닝하려면 스토리지 클래스에 필요한 `mountOptions`을(를) 추가하십시오.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

Trident StorageClass 속성

이러한 매개변수는 특정 유형의 볼륨을 프로비저닝하는 데 사용할 Trident 관리 스토리지 풀을 결정합니다.

속성	유형	값	제공	요청	지원 대상:
미디어[1]	문자열	HDD, 하이브리드, SSD	풀에는 이러한 유형의 미디어가 포함되어 있습니다. 하이브리드는 둘 다를 의미합니다	지정된 미디어 유형	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	문자열	씬, 씩	풀은 이 프로비저닝 방식을 지원합니다	프로비저닝 방법이 지정되었습니다	thick: 모든 ONTAP, thin: 모든 ONTAP 및 SolidFire-SAN
backendType	문자열	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	풀은 이러한 유형의 백엔드에 속합니다	지정된 백엔드	모든 드라이버
스냅샷	불	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다	스냅샷이 활성화된 볼륨	ontap-nas, ontap-san, solidfire-san
클론	불	참, 거짓	풀은 볼륨 복제를 지원합니다	클론이 활성화된 볼륨	ontap-nas, ontap-san, solidfire-san

속성	유형	값	제공	요청	지원 대상:
암호화	불	참, 거짓	스토리지 풀은 암호화된 볼륨을 지원합니다.	암호화가 설정된 볼륨	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	양의 정수	풀은 이 범위의 IOPS를 보장할 수 있습니다	볼륨에서 이러한 IOPS를 보장합니다	SolidFire-SAN

[1]: ONTAP Select 또는 FSx for ONTAP 시스템에서 지원되지 않습니다

스토리지 클래스가 "Kubernetes 및 Trident 객체"와 상호 작용하는 방법 및 Trident가 볼륨을 프로비저닝하는 방식을 제어하는 매개변수에 대한 자세한 내용은 `PersistentVolumeClaim`를 참조하십시오.

스토리지 클래스를 생성합니다

StorageClass를 구성한 후 Kubernetes에서 생성할 수 있습니다.

단계

1. 이것은 Kubernetes 객체이므로 `kubectl`을 사용하여 Kubernetes에서 생성합니다.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. 이제 Kubernetes와 Trident 모두에서 **basic-csi** 스토리지 클래스가 표시되어야 하며, Trident가 백엔드에서 풀을 검색했을 것입니다.

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

SMB 볼륨 프로비저닝

`ontap-nas` 드라이버를 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다. 하지만 그러려면 다음 단계를 완료해야 합니다: xref:{relative_path}worker-node-prep.html#prepare-to-provision-smb-volumes["SMB 볼륨 프로비저닝 준비"].

백엔드 고급 구성 및 예

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개변수	설명	예
`version`		항상 1
`storageDriverName`	스토리지 드라이버의 이름	`ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`
`backendName`	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소입니다. 정규화된 도메인 이름(FQDN)도 지정할 수 있습니다. Trident를 IPv6 플래그를 사용하여 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같이 대괄호 안에 정의해야 합니다. 만약 `fsxFilesystemID`를 `aws` 필드에 제공하면 `managementLIF`를 제공할 필요가 없습니다. `Trident`가 `AWS`에서 `SVM` `managementLIF` 정보를 가져오기 때문입니다. 따라서 SVM(예: vsadmin) 아래의 사용자에 대한 자격 증명을 제공해야 하며, 해당 사용자는 `vsadmin` 역할을 가지고 있어야 합니다.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	프로토콜 LIF의 IP 주소입니다. * ONTAP NAS 드라이버*: NetApp dataLIF를 지정하는 것을 권장합니다. 지정하지 않으면 Trident는 SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하여 여러 dataLIF에 걸쳐 로드 밸런싱을 수행하는 라운드 로빈 DNS를 구성할 수 있습니다. 초기 설정 후 변경할 수 있습니다. * ONTAP SAN 드라이버*: iSCSI의 경우 지정하지 마십시오. Trident는 ONTAP 선택적 LUN 맵을 사용하여 멀티 패스 세션을 설정하는 데 필요한 iSCSI LIF를 검색합니다. dataLIF가 명시적으로 정의된 경우 경고가 생성됩니다. Trident를 IPv6 플래그를 사용하여 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같이 대괄호 안에 정의해야 합니다.	

매개변수	설명	예
autoExportPolicy	자동 엑스포트 정책 생성 및 업데이트를 활성화합니다[Boolean]. autoExportPolicy 및 autoExportCIDRs 옵션을 사용하면 Trident가 내보내기 정책을 자동으로 관리할 수 있습니다.	false
autoExportCIDRs	autoExportPolicy`이 활성화된 경우 Kubernetes 노드 IP를 필터링하는 데 사용할 CIDR 목록입니다. `autoExportPolicy 및 autoExportCIDRs 옵션을 사용하면 Trident가 내보내기 정책을 자동으로 관리할 수 있습니다.	"["0.0.0.0/0", "::/0"]"
labels	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
clientCertificate	클라이언트 인증서의 Base64 인코딩 값입니다. 인증서 기반 인증에 사용됩니다	""
clientPrivateKey	클라이언트 개인 키를 Base64로 인코딩한 값입니다. 인증서 기반 인증에 사용됩니다	""
trustedCACertificate	신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 선택 사항입니다. 인증서 기반 인증에 사용됩니다.	""
username	클러스터 또는 SVM에 연결하는 데 사용할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. 예: vsadmin.	
password	클러스터 또는 SVM에 연결하는 데 사용되는 비밀번호입니다. 자격 증명 기반 인증에 사용됩니다.	
svm	사용할 스토리지 가상 머신	SVM 관리 LIF가 지정된 경우 파생됩니다.
storagePrefix	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 생성 후에는 수정할 수 없습니다. 이 매개 변수를 업데이트하려면 새 백엔드를 생성해야 합니다.	trident
limitAggregateUsage	**Amazon FSx for NetApp ONTAP**에는 지정하지 마십시오. 제공된 fsxadmin 및 `vsadmin`에는 Trident를 사용하여 애그리게이트 사용량을 검색하고 제한하는 데 필요한 권한이 포함되어 있지 않습니다.	사용하지 마십시오.

매개변수	설명	예
`limitVolumeSize`	요청된 볼륨 크기가 이 값을 초과하면 프로비저닝이 실패합니다. 또한 관리하는 qtree 및 LUN의 최대 크기를 제한하며, `qtreesPerFlexvol` 옵션을 통해 FlexVol 볼륨당 최대 qtree 수를 사용자 지정할 수 있습니다.	"" (기본적으로 적용되지 않음)
`lunsPerFlexvol`	FlexVol 볼륨당 최대 LUN 수는 [50, 200] 범위 내에 있어야 합니다. SAN 전용입니다.	“100”
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예, {"api":false, "method":true} 문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 `debugTraceFlags` 사용하지 마십시오.	null
`nfsMountOptions`	쉼표로 구분된 NFS 마운트 옵션 목록입니다. Kubernetes 영구 볼륨의 마운트 옵션은 일반적으로 스토리지 클래스에 지정되지만, 스토리지 클래스에 마운트 옵션이 지정되지 않은 경우 Trident는 스토리지 백엔드의 구성 파일에 지정된 마운트 옵션을 사용합니다. 스토리지 클래스 또는 구성 파일에 마운트 옵션이 지정되지 않은 경우 Trident는 연결된 영구 볼륨에 마운트 옵션을 설정하지 않습니다.	""
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 `nfs`, `smb` 또는 null입니다. **SMB** 볼륨의 경우 **`smb`**로 설정해야 합니다**.** null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	`nfs`
`qtreesPerFlexvol`	FlexVol 볼륨당 최대 Qtree 수는 [50, 300] 범위 내에 있어야 합니다.	`"200"`
`smbShare`	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console 또는 ONTAP CLI를 사용하여 생성한 SMB 공유의 이름 또는 Trident가 SMB 공유를 생성할 수 있도록 허용하는 이름입니다. 이 매개 변수는 Amazon FSx for NetApp ONTAP 백엔드에 필수입니다.	`smb-share`

매개변수	설명	예
useREST	ONTAP REST API를 사용하기 위한 부울 매개 변수입니다. `true`로 설정하면 Trident는 ONTAP REST API를 사용하여 백엔드와 통신합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한 사용되는 ONTAP 로그인 역할은 `ontap` 애플리케이션에 대한 액세스 권한이 있어야 합니다. 이는 사전 정의된 vsadmin 및 cluster-admin 역할로 충족됩니다.	false
aws	AWS FSx for ONTAP의 구성 파일에서 다음을 지정할 수 있습니다. - fsxFilesystemID: AWS FSx 파일 시스템의 ID를 지정합니다. - apiRegion: AWS API 리전 이름입니다. - apikey: AWS API 키입니다. - secretKey: AWS 비밀 키입니다.	"" "" ""
credentials	AWS Secrets Manager에 저장할 FSx SVM 자격 증명을 지정합니다. - name: SVM 자격 증명이 포함된 시크릿의 Amazon 리소스 이름(ARN)입니다. - type: `awsarn`로 설정합니다. 자세한 내용은 "AWS Secrets Manager 암호를 생성합니다"을 참조하십시오.	

볼륨 프로비저닝을 위한 백엔드 구성 옵션

구성의 defaults 섹션에서 이러한 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. 예를 들어, 아래 구성 예를 참조하십시오.

매개변수	설명	기본값
spaceAllocation	LUN의 공간 할당	true
spaceReserve	공간 예약 모드, "none"(씬) 또는 "volume"(씩)	none
snapshotPolicy	사용할 스냅샷 정책	none

매개변수	설명	기본값
qosPolicy	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀 또는 백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하십시오. Trident에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유되지 않는 QoS 정책 그룹을 사용하고 정책 그룹이 각 구성 요소에 개별적으로 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 대한 상한선을 적용합니다.	""
adaptiveQosPolicy	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀 또는 백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하십시오. ontap-nas-economy에서는 지원되지 않습니다.	""
snapshotReserve	스냅샷용으로 예약된 볼륨의 비율 "0"	snapshotPolicy`이 `none`인 경우, `else ""
splitOnClone	생성 시 상위 항목에서 클론 분할	false
encryption	새 볼륨에서 NetApp Volume Encryption(NVE)을 활성화합니다. 기본값은 `false`입니다. 이 옵션을 사용하려면 클러스터에서 NVE 라이센스가 있고 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하십시오. "Trident가 NVE 및 NAE와 작동하는 방식"	false
luksEncryption	LUKS 암호화를 활성화합니다. 다음을 참조하십시오. "Linux Unified Key Setup(LUKS) 사용". SAN 전용입니다.	""
tieringPolicy	사용할 계층화 정책 none	
unixPermissions	새 볼륨 모드입니다. **SMB** 볼륨의 경우 비워 두십시오**.**	""
securityStyle	새 볼륨에 대한 보안 스타일입니다. NFS는 mixed 및 unix 보안 스타일을 지원합니다. SMB는 mixed 및 ntfs 보안 스타일을 지원합니다.	NFS 기본값은 `unix`입니다. SMB 기본값은 `ntfs`입니다.

PVC 구성

이 섹션에는 구성된 Kubernetes StorageClass를 사용하여 PV를 요청하는

PersistentVolumeClaim(PVC)을 생성하는 방법에 대한 지침이 포함되어 있습니다. 성공하면 PV를 파드에 마운트할 수 있습니다.

PVC를 생성합니다

`https://kubernetes.io/docs/concepts/storage/persistent-volumes["_PersistentVolumeClaim_"^](PVC)`는 클러스터의 `PersistentVolume`에 대한 액세스 요청입니다. `PVC`는 특정 크기의 스토리지 또는 액세스 모드를 요청하도록 구성할 수 있습니다. 연결된 `StorageClass`를 사용하여 클러스터 관리자는 `PersistentVolume` 크기 및 액세스 모드 외에도 성능이나 서비스 수준과 같은 다양한 요소를 제어할 수 있습니다.

Trident 백엔드와 StorageClass를 생성한 후 PVC를 생성할 수 있습니다. PVC가 생성되면 해당 볼륨을 Pod에 마운트할 수 있습니다.

샘플 매니페스트

다음 예시는 기본적인 PVC 구성 옵션을 보여줍니다.

RWX 액세스가 있는 **PVC**

이 예제는 `basic-csi`라는 이름의 StorageClass와 연결된 RWX 액세스 권한이 있는 기본 PVC를 보여줍니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

iSCSI를 사용하는 **PVC** 예

이 예제는 RWO 액세스가 가능한 iSCSI용 기본 PVC를 보여주며, 이는 StorageClass라는 이름의 `protection-gold`와(과) 연결되어 있습니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

PVC 생성

단계

1. PVC를 생성합니다.

```
kubectl create -f pvc.yaml
```

2. PVC 상태를 확인합니다.

```
kubectl get pvc
```

```
NAME         STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi       RWO                          5m
```

스토리지 클래스가 "Kubernetes 및 Trident 객체"와 상호 작용하는 방법 및 Trident가 볼륨을 프로비저닝하는 방식을 제어하는 매개변수에 대한 자세한 내용은 `PersistentVolumeClaim`를 참조하십시오.

애플리케이션 배포

스토리지 클래스와 PVC가 생성되면 PV를 파드에 마운트할 수 있습니다. 이 섹션에서는 PV를 파드에 연결하는 예제 명령과 구성을 설명합니다.

샘플 애플리케이션 배포

단계

1. POD에 볼륨을 마운트합니다.

```
kubectl create -f pv-pod.yaml
```

다음 예시는 PVC를 포드에 연결하는 기본 구성을 보여줍니다. 기본 구성:

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

참고 `kubectl get pod --watch`를 사용하여 진행 상황을 모니터링할 수 있습니다.

2. 볼륨이 `/my/mount/path`에 마운트되었는지 확인하십시오.

```
kubectl exec -it pv-pod -- df -h /my/mount/path
```

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

이제 Pod를 삭제할 수 있습니다. Pod 애플리케이션은 더 이상 존재하지 않지만 볼륨은 그대로 유지됩니다.

```
kubectl delete pod pv-pod
```

EKS 클러스터에 **Trident EKS** 애드온을 구성합니다

NetApp Trident는 Kubernetes 환경에서 Amazon FSx for NetApp ONTAP 스토리지 관리를 간소화하여 개발자와 관리자가 애플리케이션 배포에 집중할 수 있도록 지원합니다. NetApp Trident EKS 애드온에는 최신 보안 패치와 버그 수정 사항이 포함되어 있으며, AWS에서

Amazon EKS와의 호환성을 검증받았습니다. EKS 애드온을 사용하면 Amazon EKS 클러스터의 보안과 안정성을 지속적으로 유지할 수 있으며, 애드온 설치, 구성 및 업데이트에 필요한 작업량을 줄일 수 있습니다.

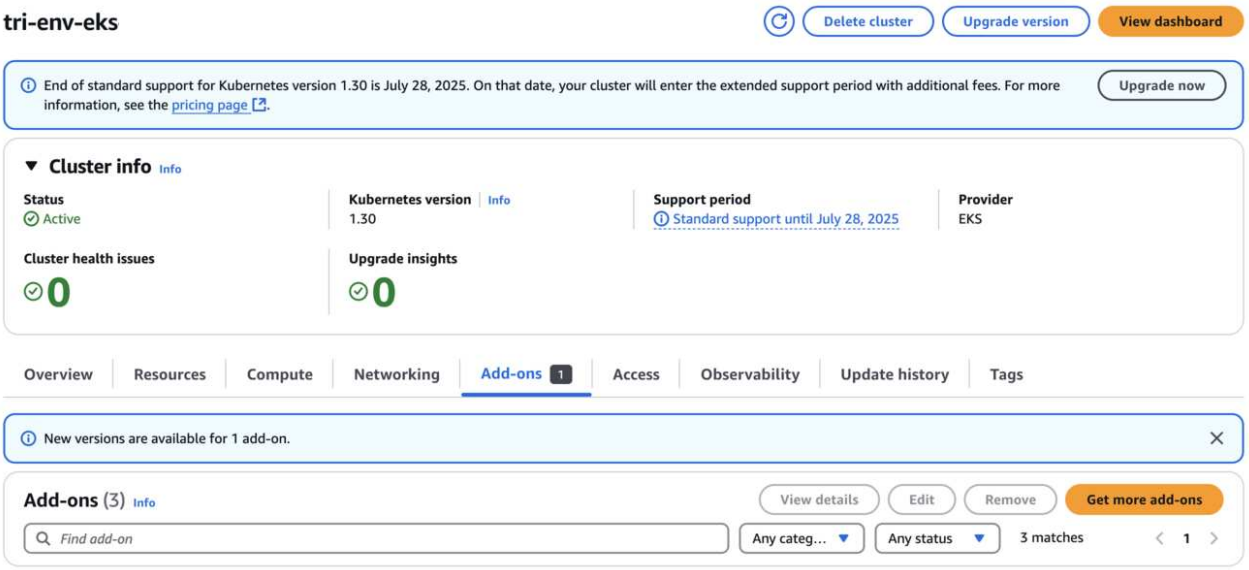
필수 구성 요소

AWS EKS용 Trident 애드온을 구성하기 전에 다음 사항을 확인하십시오.

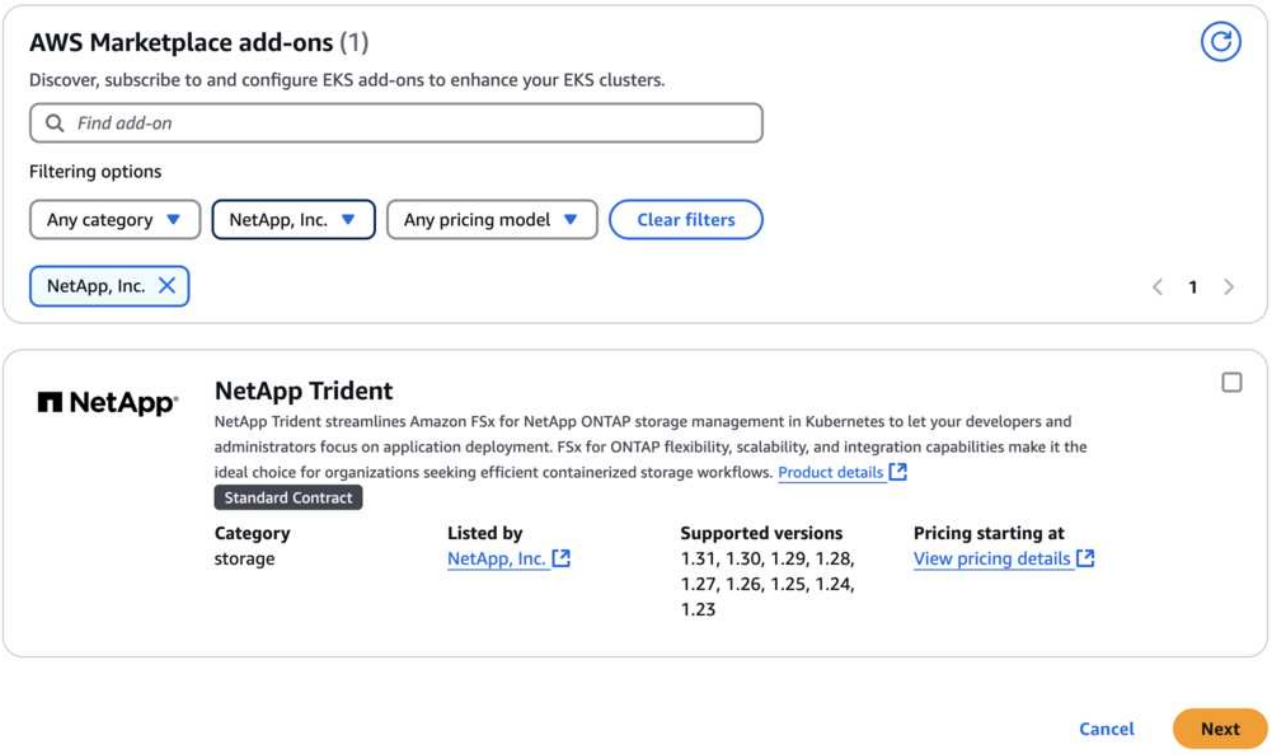
- 추가 기능을 사용할 수 있는 권한이 있는 Amazon EKS 클러스터 계정입니다. 다음을 참조하십시오. "Amazon EKS 추가 기능".
- AWS Marketplace에 대한 AWS 권한:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 유형: Amazon Linux 2(AL2_x86_64) 또는 Amazon Linux 2 Arm(AL2_ARM_64)
- 노드 유형: AMD 또는 ARM
- 기존 Amazon FSx for NetApp ONTAP 파일 시스템

단계

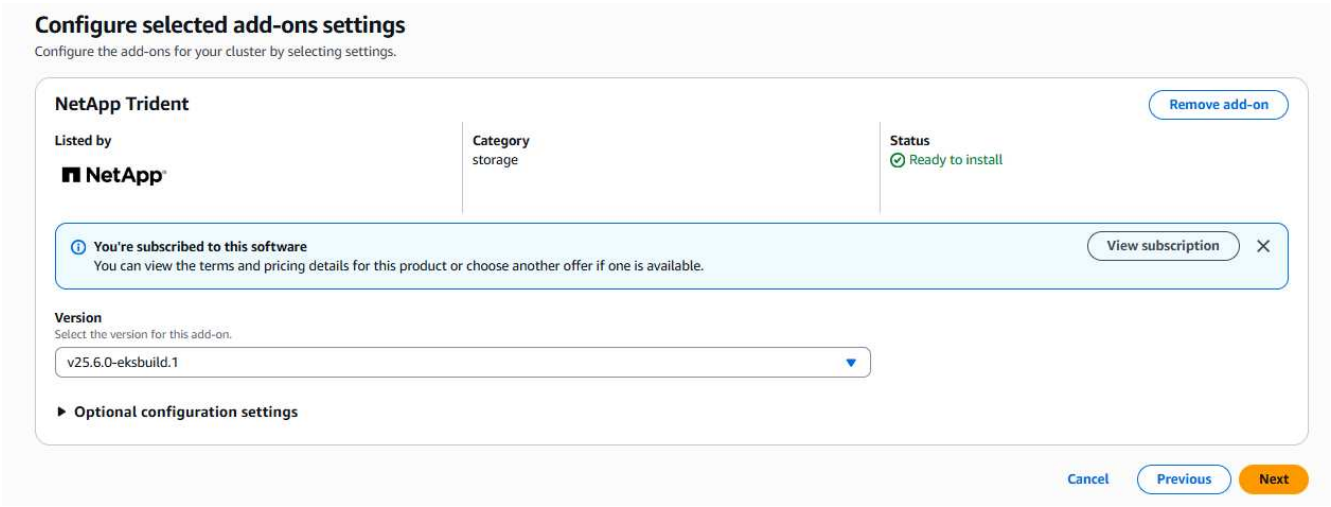
1. EKS Pod가 AWS 리소스에 액세스할 수 있도록 IAM 역할과 AWS 시크릿을 생성해야 합니다. 지침은 "IAM 역할 및 AWS Secret을 생성합니다"을 참조하십시오.
2. EKS Kubernetes 클러스터에서 **Add-ons** 탭으로 이동합니다.



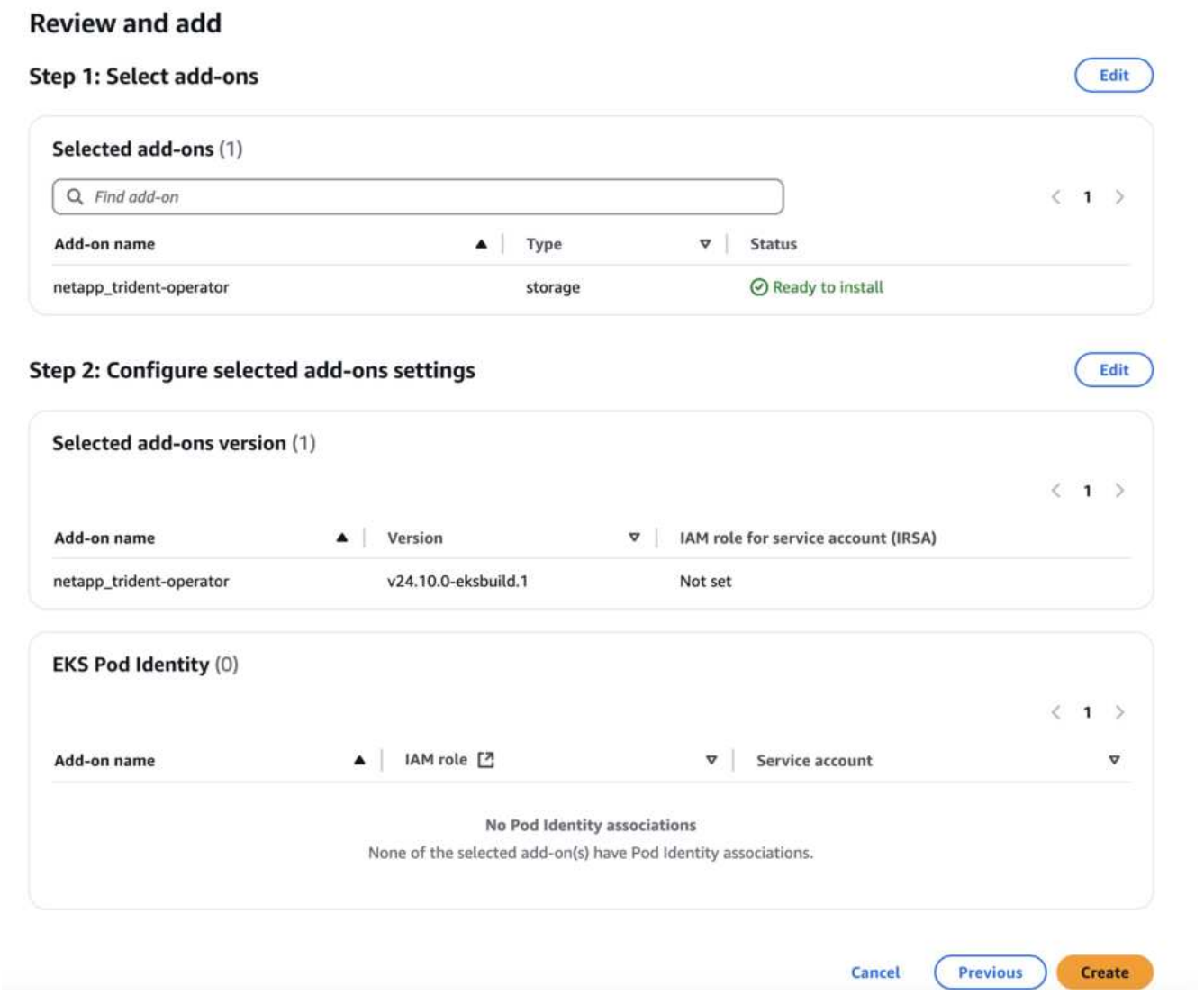
3. *AWS Marketplace add-ons*로 이동하여 *storage* 카테고리를 선택하세요.



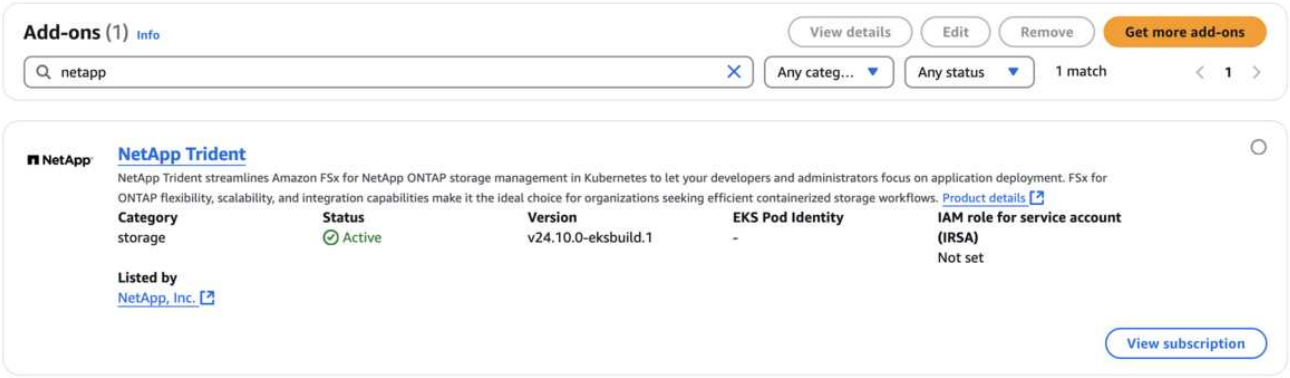
4. *NetApp Trident*를 찾아 Trident 추가 기능 확인란을 선택한 다음 *다음*을 클릭합니다.
5. 원하는 추가 기능 버전을 선택합니다.



6. 필요한 추가 기능 설정을 구성하십시오.



7. IRSA(서비스 계정용 IAM 역할)를 사용하는 경우 추가 구성 단계"여기"를 참조하십시오.
8. *생성*을 선택합니다.
9. 추가 기능의 상태가 _Active_인지 확인하십시오.



10. 다음 명령을 실행하여 Trident가 클러스터에 제대로 설치되었는지 확인하십시오.

```
kubectl get pods -n trident
```

11. 설정을 계속 진행하여 스토리지 백엔드를 구성하십시오. 자세한 내용은 "스토리지 백엔드 구성"를 참조하십시오.

CLI를 사용하여 Trident EKS 애드온 설치/제거

CLI를 사용하여 **NetApp Trident EKS** 추가 기능을 설치합니다**.**

다음 예시 명령어는 Trident EKS 애드온을 설치합니다.
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1`(전용 버전 포함)

다음 예시 명령은 Trident EKS 애드온 버전 25.6.1를 설치합니다:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1` (전용 버전 포함)

다음 예시 명령은 Trident EKS 애드온 버전 25.6.2를 설치합니다:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1` (전용 버전 포함)

CLI를 사용하여 **NetApp Trident EKS** 추가 기능을 제거합니다**.**

다음 명령을 실행하면 Trident EKS 추가 기능이 제거됩니다.

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

kubectl을 사용하여 백엔드 생성

백엔드는 Trident와 스토리지 시스템 간의 관계를 정의합니다. 백엔드는 Trident가 해당 스토리지 시스템과 통신하는 방법과 Trident가 해당 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다. Trident 설치 후 다음 단계는 백엔드를 생성하는 것입니다. `TridentBackendConfig` Custom Resource Definition(CRD)을 사용하면 Kubernetes 인터페이스를 통해 Trident 백엔드를 직접 생성하고 관리할 수 있습니다. `kubectl` 또는 Kubernetes 배포판에 맞는 CLI 도구를 사용하여 수행할 수 있습니다.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`)는 프런트엔드 네임스페이스 CRD로, `kubectl`를 사용하여 `Trident` 백엔드를 관리할 수 있도록 해줍니다. 이제 `Kubernetes` 및 스토리지 관리자는 별도의 명령줄 유틸리티(`tridentctl` 없이 Kubernetes CLI를 통해 백엔드를 직접 생성하고 관리할 수 있습니다.

```
`TridentBackendConfig` 객체가 생성될 때 다음과 같은 일이 발생합니다.
```

- 백엔드는 사용자가 제공하는 구성을 기반으로 Trident에 의해 자동으로 생성됩니다. 이는 내부적으로 `TridentBackend` (`tbe`, `tridentbackend`) CR로 표현됩니다.

- `TridentBackendConfig`는 Trident에 의해 생성된 `TridentBackend`에 고유하게 바인딩됩니다.

각 `TridentBackendConfig`은 `TridentBackend`와 일대일 매핑을 유지합니다. 전자는 백엔드를 설계하고 구성하기 위해 사용자에게 제공되는 인터페이스이며, 후자는 Trident가 실제 백엔드 객체를 표현하는 방식입니다.

경고	`TridentBackend` CR은 Trident에 의해 자동으로 생성됩니다. CR을 수정해서는 안 됩니다. 백엔드를 업데이트하려면 `TridentBackendConfig` 객체를 수정하십시오.

`TridentBackendConfig` 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
```

원하는 스토리지 플랫폼/서비스에 대한 샘플 구성은 "trident-installer" 디렉터리의 예제를 참조하십시오.

`spec`는 백엔드별 구성 매개변수를 사용합니다. 이 예에서 백엔드는 `ontap-san` 스토리지 드라이버를 사용하며 여기에 표로 정리된 구성 매개변수를 사용합니다. 원하는 스토리지 드라이버에 대한 구성 옵션 목록은 xref:{relative_path}backends.html["스토리지 드라이버에 대한 백엔드 구성 정보"^]를 참조하십시오.

`spec` 섹션에는 또한 `credentials` 및 `deletionPolicy` 필드가 포함되어 있으며, 이는 `TridentBackendConfig` CR에서 새롭게 도입되었습니다.

- `credentials`: 이 매개변수는 필수 입력 항목이며 스토리지 시스템/서비스 인증에 사용되는 자격 증명을 포함합니다. 이 값은 사용자가 생성한 Kubernetes Secret으로 설정됩니다. 자격 증명을 일반 텍스트로 전달할 수 없으며, 그렇게 할 경우 오류가 발생합니다.
- `deletionPolicy`: 이 필드는 `TridentBackendConfig`가 삭제될 때 발생해야 하는 작업을 정의합니다. 다음 두 가지 값 중 하나를 사용할 수 있습니다.
  - `delete`: 이렇게 하면 `TridentBackendConfig` CR과 관련 백엔드가 모두 삭제됩니다. 이것이 기본값입니다.

- retain: TridentBackendConfig CR이 삭제되더라도 백엔드 정의는 계속 남아 있으며 tridentctl`를 사용하여 관리할 수 있습니다. 삭제 정책을 `retain`로 설정하면 사용자는 이전 릴리스(21.04 이전 버전)로 다운그레이드하고 생성된 백엔드를 유지할 수 있습니다. 이 필드의 값은 `TridentBackendConfig 생성 후 업데이트할 수 있습니다.

참고	백엔드의 이름은 spec.backendName`를 사용하여 설정합니다. 지정하지 않으면 백엔드 이름은 `TridentBackendConfig 객체의 이름(metadata.name)으로 설정됩니다. `spec.backendName`를 사용하여 백엔드 이름을 명시적으로 설정하는 것이 좋습니다.
팁	tridentctl`로 생성된 백엔드는 연결된 `TridentBackendConfig 오브젝트가 없습니다. 이러한 백엔드는 kubectl`를 사용하여 `TridentBackendConfig CR을 생성함으로써 관리할 수 있습니다. spec.backendName, spec.storagePrefix, spec.storageDriverName 등과 같은 동일한 구성 파라미터를 지정해야 하므로 주의가 필요합니다. Trident는 새로 생성된 `TridentBackendConfig`를 기존 백엔드와 자동으로 바인딩합니다.

단계 개요

`kubectl`를 사용하여 새 백엔드를 생성하려면 다음을 수행해야 합니다.

1. "Kubernetes Secret"을 생성합니다. 이 secret에는 Trident가 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. TridentBackendConfig 개체를 만듭니다. 여기에는 스토리지 클러스터/서비스에 대한 세부 정보가 포함되어 있으며 이전 단계에서 만든 시크릿을 참조합니다.

백엔드를 생성한 후 `kubectl get tbc <tbc-name> -n <trident-namespace>`를 사용하여 상태를 확인하고 추가 세부 정보를 수집할 수 있습니다.

1단계**: Kubernetes Secret** 생성

백엔드에 대한 액세스 자격 증명이 포함된 Secret을 생성합니다. 이는 각 스토리지 서비스/플랫폼마다 고유합니다. 다음은 예입니다.

```
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: password
```

이 표는 각 스토리지 플랫폼의 Secret에 포함되어야 하는 필드를 요약한 것입니다.

스토리지 플랫폼 **Secret Fields** 설명	비밀	필드 설명
Azure NetApp Files	clientID	앱 등록의 클라이언트 ID
요소(NetApp HCI/SolidFire)	엔드포인트	테넌트 자격 증명이 있는 SolidFire 클러스터의 MVIP
ONTAP	사용자 이름	클러스터/SVM에 연결하는 데 사용할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다.
ONTAP	비밀번호	클러스터/SVM에 연결하는 데 사용되는 암호입니다. 자격 증명 기반 인증에 사용됩니다.
ONTAP	clientPrivateKey	클라이언트 개인 키를 Base64로 인코딩한 값입니다. 인증서 기반 인증에 사용됩니다.
ONTAP	chapUsername	인바운드 사용자 이름. useCHAP=true인 경우 필수입니다. `ontap-san` 및 `ontap-san-economy`
ONTAP	chapInitiatorSecret	CHAP 이니시에이터 암호입니다. useCHAP=true인 경우 필수입니다. `ontap-san` 및 `ontap-san-economy`
ONTAP	chapTargetUsername	대상 사용자 이름. useCHAP=true인 경우 필수입니다. `ontap-san` 및 `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	CHAP 대상 개시자 비밀 키. useCHAP=true인 경우 필수입니다. `ontap-san` 및 `ontap-san-economy`

이 단계에서 생성된 Secret은 다음 단계에서 생성되는 `spec.credentials` 객체의 `TridentBackendConfig` 필드에서 참조됩니다.

2단계: `TridentBackendConfig` **CR** 생성

이제 `TridentBackendConfig` CR을 생성할 준비가 되었습니다. 이 예에서는 `ontap-san` 드라이버를 사용하는 백엔드가 아래 표시된 `TridentBackendConfig` 객체를 사용하여 생성됩니다.

```
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: `TridentBackendConfig` **CR의 상태를 확인합니다**

`TridentBackendConfig` CR을 생성했으므로 이제 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 `TridentBackendConfig` CR에 바인딩되었습니다.

Phase는 다음 값 중 하나를 사용할 수 있습니다.

- `Bound`: `TridentBackendConfig` CR은 백엔드와 연결되어 있으며, 해당 백엔드에는 `configRef`가 `TridentBackendConfig` CR의 uid로 설정되어 있습니다.
- `Unbound`: `""`로 표현됩니다. `TridentBackendConfig` 객체는 백엔드에 바인딩되지 않았습니다. 새로 생성되는 모든 `TridentBackendConfig` CR은 기본적으로 이 단계에 있습니다. 단계가 변경된 후에는 다시 Unbound 상태로 되돌릴 수 없습니다.
- `Deleting`: `TridentBackendConfig`CR의 `deletionPolicy` 삭제가 설정되었습니다. `TridentBackendConfig`CR이 삭제되면 삭제 중 상태로 전환됩니다.
  - 백엔드에 영구 볼륨 클레임(PVC)이 없는 경우 `TridentBackendConfig`를 삭제하면 `Trident`가 백엔드와 `TridentBackendConfig` CR을 삭제합니다.
  - 백엔드에 하나 이상의 PVC가 존재하면 삭제 상태로 전환됩니다. `TridentBackendConfig` CR도 이후 삭제

단계로 들어갑니다. 백엔드와 `TridentBackendConfig`는 모든 PVC가 삭제된 후에만 삭제됩니다.

- `Lost`: `TridentBackendConfig` CR과 연결된 백엔드가 실수로 또는 의도적으로 삭제되었지만 `TridentBackendConfig` CR에는 여전히 삭제된 백엔드에 대한 참조가 남아 있습니다. `TridentBackendConfig` CR은 `deletionPolicy` 값과 관계없이 삭제할 수 있습니다.
- `Unknown`: Trident는 `TridentBackendConfig` CR과 연결된 백엔드의 상태 또는 존재 여부를 확인할 수 없습니다. 예를 들어 API 서버가 응답하지 않거나 `tridentbackends.trident.netapp.io` CRD가 없는 경우입니다. 이 경우 개입이 필요할 수 있습니다.

이 단계에서 백엔드가 성공적으로 구축되었습니다! "백엔드 업데이트 및 백엔드 삭제"와 같이 추가로 처리할 수 있는 몇 가지 작업이 있습니다.

(선택 사항) 4단계: 자세한 정보 확인

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 `TridentBackendConfig`의 YAML/JSON 덤프도 얻을 수 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 `backendName` 및 `backendUUID`가 해당 `TridentBackendConfig` CR에 대한 응답으로 생성된 백엔드의 정보가 포함되어 있습니다. `lastOperationStatus` 필드는 `TridentBackendConfig` CR의 마지막 작업 상태를 나타내며, 이는 사용자가 트리거한 경우(예: 사용자가 `spec`에서 무언가를 변경한 경우) 또는 `Trident`에 의해 트리거된 경우(예: `Trident` 재시작 중)에 발생할 수 있습니다. 이 값은 `Success` 또는 `Failed` 중 하나일 수 있습니다. `phase`는 `TridentBackendConfig` CR과 백엔드 간의 관계 상태를 나타냅니다. 위 예시에서, `phase`의 값은 `Bound`이며, 이는 `TridentBackendConfig` CR이 백엔드와 연결되어 있음을 의미합니다.

`kubectl -n trident describe tbc <tbc-cr-name>` 명령을 실행하여 이벤트 로그의 세부 정보를 확인할 수 있습니다.

경고	연결된 `TridentBackendConfig` 객체가 포함된 백엔드는 `tridentctl`를 사용하여 업데이트하거나 삭제할 수 없습니다. `tridentctl`와 `TridentBackendConfig` 간 전환에 필요한 단계들을 이해하려면 "여기를 참조하십시오".

백엔드 관리

kubectl을 사용하여 백엔드 관리를 수행합니다

`kubectl`를 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보십시오.

백엔드를 삭제합니다

`TridentBackendConfig`를 삭제하면 `Trident`에 백엔드를 삭제/유지하도록(`deletionPolicy`에 따라) 지시하는 것입니다. 백엔드를 삭제하려면 `deletionPolicy`가 `delete`로 설정되어 있는지 확인하십시오. `TridentBackendConfig`만 삭제하려면 `deletionPolicy`가 `retain`으로 설정되어 있는지 확인하십시오. 이렇게 하면 백엔드가 계속 존재하며 `tridentctl`를 사용하여 관리할 수 있습니다.

다음 명령을 실행합니다.

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

Trident는 `TridentBackendConfig`에서 사용 중인 Kubernetes Secret을 삭제하지 않습니다. Kubernetes 사용자가 Secret 정리를 담당합니다. Secret을 삭제할 때는 주의해야 합니다. 백엔드에서 사용하지 않는 Secret만 삭제해야 합니다.

기존 백엔드 보기

다음 명령을 실행합니다.

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

`tridentctl get backend -n trident` 또는 `tridentctl get backend -o yaml -n trident`를 실행하여 존재하는 모든 백엔드 목록을 얻을 수도 있습니다. 이 목록에는 `tridentctl`로 생성된 백엔드도 포함됩니다.

백엔드 업데이트

백엔드를 업데이트해야 하는 이유는 여러 가지가 있을 수 있습니다.

- 스토리지 시스템에 대한 자격 증명이 변경되었습니다. 자격 증명을 업데이트하려면 `TridentBackendConfig` 객체에 사용되는 Kubernetes Secret을 업데이트해야 합니다. Trident는 제공된 최신 자격 증명으로 백엔드를 자동으로 업데이트합니다. 다음 명령을 실행하여 Kubernetes Secret을 업데이트하십시오.

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

- 매개변수(예: 사용 중인 ONTAP SVM의 이름)를 업데이트해야 합니다.
  - 다음 명령을 사용하여 Kubernetes를 통해 `TridentBackendConfig` 객체를 직접 업데이트할 수 있습니다.

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

  - 또는 다음 명령을 사용하여 기존 `TridentBackendConfig` CR을 변경할 수 있습니다.

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

참고

- 백엔드 업데이트가 실패하면 백엔드는 마지막으로 알려진 구성을 계속 유지합니다. 로그를 확인하여 원인을 파악하려면 `kubectl get tbc <tbc-name> -o yaml -n trident` 또는 `kubectl describe tbc <tbc-name> -n trident`을 실행하십시오.
- 구성 파일의 문제를 식별하고 수정한 후 업데이트 명령을 다시 실행할 수 있습니다.

tridentctl을 사용하여 백엔드 관리를 수행합니다

`tridentctl`를 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보십시오.

백엔드 생성

"백엔드 구성 파일"를 생성한 후 다음 명령을 실행하십시오.

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

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 확인하고 원인을 파악할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 식별하고 수정한 후 `create` 명령을 다시 실행하기만 하면 됩니다.

백엔드를 삭제합니다

Trident에서 백엔드를 삭제하려면 다음 단계를 따르세요.

1. 백엔드 이름 검색:

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

2. 백엔드 삭제:

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

참고 | Trident가 이 백엔드에서 프로비저닝한 볼륨과 스냅샷이 아직 남아 있는 경우, 백엔드를 삭제하면 해당 백엔드에서 새 볼륨을 프로비저닝할 수 없습니다. 백엔드는 "Deleting" 상태로 계속 유지됩니다.

기존 백엔드 보기

Trident가 알고 있는 백엔드를 보려면 다음을 수행합니다.

- 요약을 보려면 다음 명령을 실행하십시오.

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

- 모든 세부 정보를 확인하려면 다음 명령을 실행하십시오.

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

백엔드 업데이트

새 백엔드 구성 파일을 생성한 후 다음 명령을 실행하십시오.

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

백엔드 업데이트가 실패하면 백엔드 구성에 문제가 있거나 유효하지 않은 업데이트를 시도한 것입니다. 다음 명령을 실행하여 로그를 확인하고 원인을 파악할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 식별하고 수정한 후 `update` 명령을 다시 실행하기만 하면 됩니다.

백엔드를 사용하는 스토리지 클래스를 식별합니다

이는 백엔드 객체에 대해 `tridentctl`이(가) 출력하는 `JSON`을 사용하여 답변할 수 있는 질문 유형의 예입니다. 이는 설치해야 하는 `jq` 유틸리티를 사용합니다.

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

이는 `TridentBackendConfig`을 사용하여 생성된 백엔드에도 적용됩니다.

Trident에서 백엔드를 관리하는 다양한 방법에 대해 알아보십시오.

백엔드 관리 옵션

`TridentBackendConfig`의 도입으로 관리자는 이제 백엔드를 관리하는 두 가지 고유한 방법을 갖게 되었습니다. 이로 인해 다음과 같은 질문이 제기됩니다.

- `tridentctl`를 사용하여 생성된 백엔드를 `TridentBackendConfig`로 관리할 수 있습니까?
- `TridentBackendConfig`를 사용하여 생성된 백엔드를 `tridentctl`를 사용하여 관리할 수 있습니까?

`tridentctl`를 사용하여 `TridentBackendConfig` 백엔드 관리

이 섹션에서는 Kubernetes 인터페이스를 통해 직접 `tridentctl`를 사용하여 생성된 백엔드를 관리하기 위해 `TridentBackendConfig` 오브젝트를 생성하는 데 필요한 단계를 다룹니다.

다음 시나리오에 적용됩니다.

- 기존 백엔드의 경우, `TridentBackendConfig`가 없으며, 이는 `tridentctl`로 생성되었기 때문입니다.
- `tridentctl`로 생성된 새 백엔드, 다른 `TridentBackendConfig` 객체가 존재합니다.

두 시나리오 모두에서 백엔드는 계속 존재하며 Trident는 볼륨을 예약하고 운영합니다. 관리자는 여기에서 두 가지 중 하나를 선택할 수 있습니다.

- `tridentctl`을 사용하여 생성한 백엔드를 계속 관리합니다.
- 생성된 백엔드를 `tridentctl` 새 `TridentBackendConfig` 오브젝트에 바인딩합니다. 이렇게 하면 백엔드는 `kubectl`를 사용하여 관리되고 `tridentctl`로는 관리되지 않습니다.

  `kubectl`을 사용하여 기존 백엔드를 관리하려면 기존 백엔드에 바인딩하는 `TridentBackendConfig`를 만들어야 합니다. 다음은 그 작동 방식에 대한 개요입니다.

  1. Kubernetes Secret을 생성합니다. 이 시크릿에는 Trident가 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
  2. `TridentBackendConfig` 개체를 만듭니다. 여기에는 스토리지 클러스터/서비스에 대한 세부 정보가 포함되어 있으며 이전 단계에서 만든 시크릿을 참조합니다. 동일한 구성 매개변수(예: `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` 등)를 지정하도록 주의해야 합니다. `spec.backendName`는 기존 백엔드의 이름으로 설정해야 합니다.

0단계: 백엔드 식별

기존 백엔드에 바인딩하는 `TridentBackendConfig`를 만들려면 백엔드 구성을 가져와야 합니다. 이 예에서는 다음 JSON 정의를 사용하여 백엔드를 만들었다고 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 대로 백엔드에 대한 자격 증명이 포함된 Secret을 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: `TridentBackendConfig` **CR** 생성

다음 단계는 기존에 존재하는 `ontap-nas-backend`에 자동으로 바인딩되는 `TridentBackendConfig` CR을 생성하는 것입니다(이 예시와 같이). 다음 요구 사항이 충족되는지 확인하십시오:

- 동일한 백엔드 이름이 `spec.backendName`에 정의되어 있습니다.
- 구성 매개변수는 원래 백엔드와 동일합니다.
- 가상 풀(있는 경우)은 원래 백엔드에서와 동일한 순서를 유지해야 합니다.
- 자격 증명은 일반 텍스트가 아닌 Kubernetes Secret을 통해 제공됩니다.

이 경우 `TridentBackendConfig`는 다음과 같이 표시됩니다:

```
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: `TridentBackendConfig` **CR의 상태를 확인합니다**

`TridentBackendConfig`을 생성한 후 해당 단계는 `Bound`이어야 합니다. 또한 기존 백엔드의 이름 및 UUID와 동일한 백엔드 이름을 반영해야 합니다.

```
kubectl 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 TridentBackendConfig 객체를 사용하여 완전히 관리됩니다.

`TridentBackendConfig`를 사용하여 `tridentctl` 백엔드 관리

`tridentctl`는 `TridentBackendConfig`를 사용하여 생성된 백엔드를 나열하는 데 사용할 수 있습니다. 또한, 관리자는 `tridentctl`를 통해 이러한 백엔드를 완전히 관리하도록 선택할 수 있으며, `TridentBackendConfig`를 삭제하고 `spec.deletionPolicy`가 `retain`로 설정되어 있는지 확인할 수 있습니다.

0단계**:** 백엔드 식별

예를 들어 다음 백엔드가 `TridentBackendConfig`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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

출력에서 `TridentBackendConfig`이(가) 성공적으로 생성되었으며 백엔드에 바인딩된 것을 볼 수 있습니다[백엔드의 UUID 관찰].

1단계: **`deletionPolicy`**가 **`retain`**로 설정되어 있는지 확인합니다

`deletionPolicy`의 값을 살펴보겠습니다. 이것은 `retain`로 설정해야 합니다. 이렇게 하면 `TridentBackendConfig` CR이 삭제되더라도 백엔드 정의는 계속 존재하며 `tridentctl`로 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
```

참고 `deletionPolicy`이(가) `retain`(으)로 설정되지 않은 경우 다음 단계로 진행하지 마십시오.

2단계: `TridentBackendConfig` **CR** 삭제

마지막 단계는 `TridentBackendConfig` CR을 삭제하는 것입니다. `deletionPolicy`가 `retain`로 설정되어 있는지 확인한 후 삭제를 진행하면 됩니다:

```
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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

`TridentBackendConfig` 개체를 삭제하면 `Trident`는 백엔드 자체를 실제로 삭제하지 않고 단순히 제거합니다.

스토리지 클래스 생성 및 관리

스토리지 클래스를 생성합니다

Kubernetes StorageClass 오브젝트를 구성하고 스토리지 클래스를 생성하여 Trident에 볼륨을 프로비저닝하는 방법을 지시합니다.

Kubernetes StorageClass 오브젝트 구성

```
https://kubernetes.io/docs/concepts/storage/storage-classes/["Kubernetes StorageClass 오브젝트"^]은 Trident를 해당 클래스에 사용되는 프로비저너로 식별하고 Trident에 볼륨 프로비저닝 방법을 지시합니다. 예를 들면 다음과 같습니다.
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

스토리지 클래스가 "Kubernetes 및 Trident 객체"와 상호 작용하는 방법 및 Trident가 볼륨을 프로비저닝하는 방식을 제어하는 매개변수에 대한 자세한 내용은 `PersistentVolumeClaim`를 참조하십시오.

스토리지 클래스를 생성합니다

StorageClass 객체를 생성한 후 스토리지 클래스를 생성할 수 있습니다. 스토리지 클래스 샘플에서는 사용하거나 수정할 수 있는 몇 가지 기본 샘플을 제공합니다.

단계

1. 이것은 Kubernetes 객체이므로 `kubectl`을 사용하여 Kubernetes에서 생성합니다.

   ```
   kubectl create -f sample-input/storage-class-basic-csi.yaml
   ```

2. 이제 Kubernetes와 Trident 모두에서 **basic-csi** 스토리지 클래스가 표시되어야 하며, Trident가 백엔드에서 풀을 검색했을 것입니다.

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

스토리지 클래스 샘플

Trident는 "특정 백엔드를 위한 간단한 스토리지 클래스 정의"를 제공합니다.

또는 설치 프로그램과 함께 제공되는 `sample-input/storage-class-csi.yaml.templ` 파일을 편집하여 `*BACKEND_TYPE*`를 스토리지 드라이버 이름으로 교체할 수도 있습니다.

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

스토리지 클래스 관리

기존 스토리지 클래스를 보고, 기본 스토리지 클래스를 설정하고, 스토리지 클래스 백엔드를 식별하고, 스토리지 클래스를 삭제할 수 있습니다.

기존 스토리지 클래스를 확인합니다

- 기존 Kubernetes 스토리지 클래스를 보려면 다음 명령을 실행하십시오.

```
kubectl get storageclass
```

- Kubernetes 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행하십시오.

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

- Trident의 동기화된 스토리지 클래스를 보려면 다음 명령을 실행하십시오.

```
tridentctl get storageclass
```

- Trident의 동기화된 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행하십시오.

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

기본 스토리지 클래스를 설정합니다

Kubernetes 1.6에서는 기본 스토리지 클래스를 설정하는 기능이 추가되었습니다. 사용자가 영구 볼륨 클레임(PVC)에서 스토리지 클래스를 지정하지 않으면 이 기본 스토리지 클래스가 영구 볼륨을 프로비저닝하는 데 사용됩니다.

- 스토리지 클래스 정의에서 주석 `storageclass.kubernetes.io/is-default-class`을 true로 설정하여 기본 스토리지 클래스를 정의합니다. 사양에 따르면 다른 값이나 주석이 없으면 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"}}}'
```

Trident 설치 프로그램 번들에도 이 주석이 포함된 예제가 있습니다.

참고	클러스터에는 한 번에 하나의 기본 스토리지 클래스만 있어야 합니다. Kubernetes는 기술적으로 두 개 이상의 스토리지 클래스를 사용하는 것을 막지는 않지만 기본 스토리지 클래스가 전혀 없는 것처럼 작동합니다.

스토리지 클래스의 백엔드를 식별합니다

이는 `tridentctl` Trident 백엔드 객체에 대해 출력되는 JSON을 사용하여 답변할 수 있는 질문 유형의 예입니다. 이 예시에서는 `jq` 유틸리티를 사용하는데, 필요에 따라 먼저 설치해야 할 수도 있습니다.

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

스토리지 클래스를 삭제합니다

Kubernetes에서 스토리지 클래스를 삭제하려면 다음 명령을 실행하십시오.

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

`<storage-class>`는 스토리지 클래스로 교체해야 합니다.

이 스토리지 클래스를 통해 생성된 영구 볼륨은 변경되지 않고 그대로 유지되며, Trident가 계속해서 해당 볼륨을 관리합니다.

참고 | Trident는 생성하는 볼륨에 대해 빈 `fsType`을 적용합니다. iSCSI 백엔드의 경우 StorageClass에서 `parameters.fsType`을 적용하는 것이 좋습니다. 기존 StorageClasses를 삭제하고 `parameters.fsType`이 지정된 상태로 다시 생성해야 합니다.

볼륨 프로비저닝 및 관리

볼륨 프로비저닝

구성된 Kubernetes StorageClass를 사용하여 PV에 대한 액세스를 요청하는 PersistentVolumeClaim (PVC)를 생성합니다. 그런 다음 PV를 파드에 마운트할 수 있습니다.

개요

`https://kubernetes.io/docs/concepts/storage/persistent-volumes["_PersistentVolumeClaim_"^](PVC)`는 클러스터의 `PersistentVolume`에 대한 액세스 요청입니다.

PVC는 특정 크기의 스토리지 또는 액세스 모드를 요청하도록 구성할 수 있습니다. 연결된 StorageClass를 사용하여 클러스터 관리자는 PersistentVolume 크기 및 액세스 모드 외에도 성능이나 서비스 수준과 같은 다양한 요소를 제어할 수 있습니다.

PVC를 생성한 후에는 볼륨을 포드에 마운트할 수 있습니다.

PVC를 생성합니다

단계

1. PVC를 생성합니다.

```
kubectl create -f pvc.yaml
```

2. PVC 상태를 확인합니다.

```
kubectl get pvc
```

```
NAME         STATUS  VOLUME     CAPACITY   ACCESS MODES  STORAGECLASS AGE
pvc-storage Bound   pv-name    1Gi        RWO                  5m
```

1. POD에 볼륨을 마운트합니다.

```
kubectl create -f pv-pod.yaml
```

참고 `kubectl get pod --watch`를 사용하여 진행 상황을 모니터링할 수 있습니다.

2. 볼륨이 `/my/mount/path`에 마운트되었는지 확인하십시오.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

3. 이제 Pod를 삭제할 수 있습니다. Pod 애플리케이션은 더 이상 존재하지 않지만 볼륨은 그대로 유지됩니다.

```
kubectl delete pod pv-pod
```

샘플 매니페스트

PersistentVolumeClaim 샘플 매니페스트

이 예에서는 기본 PVC 구성 옵션을 보여 줍니다.

RWO 액세스가 있는 **PVC**

이 예제는 `basic-csi`라는 이름의 StorageClass와 연결된 RWO 액세스 권한이 있는 기본 PVC를 보여줍니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

NVMe/TCP를 사용한 **PVC**

이 예제는 RWO 액세스가 가능한 NVMe/TCP용 기본 PVC를 보여주며, 이는 StorageClass라는 이름 `protection-gold`과 연결되어 있습니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Pod 매니페스트 샘플

이 예에서는 PVC를 포드에 연결하는 기본 구성을 보여 줍니다.

기본 구성

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

기본 **NVMe/TCP** 구성

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

스토리지 클래스가 "Kubernetes 및 Trident 객체"와 상호 작용하는 방법 및 Trident가 볼륨을 프로비저닝하는 방식을 제어하는 매개변수에 대한 자세한 내용은 `PersistentVolumeClaim`를 참조하십시오.

볼륨 확장

Trident는 Kubernetes 사용자에게 볼륨 생성 후 확장 기능을 제공합니다. iSCSI, NFS, SMB, NVMe/TCP 및 FC 볼륨 확장에 필요한 구성에 대한 정보를 확인하십시오.

iSCSI 볼륨 확장

CSI 프로비저너를 사용하여 iSCSI 영구 볼륨(PV)을 확장할 수 있습니다.

참고	iSCSI 볼륨 확장은 `ontap-san`, `ontap-san-economy`, `solidfire-san` 드라이버에서 지원되며 Kubernetes 1.16 이상이 필요합니다.

1단계: 볼륨 확장을 지원하도록 StorageClass 구성

StorageClass 정의를 편집하여 `allowVolumeExpansion` 필드를 `true`로 설정합니다.

```
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` 매개 변수를 포함하도록 편집합니다.

2단계: 생성한 StorageClass로 PVC 만들기

PVC 정의를 편집하고 새로 원하는 크기를 반영하도록 `spec.resources.requests.storage`을 업데이트합니다(원래 크기보다 커야 함).

```
cat pvc-ontapsan.yaml
```

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

Trident는 영구 볼륨(PV)을 생성하고 이 영구 볼륨 클레임(PVC)과 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: **PVC**를 연결하는 **POD** 정의

크기를 조정할 PV를 포드에 연결합니다. iSCSI PV의 크기를 조정할 때는 두 가지 시나리오가 있습니다.

- PV가 포드에 연결된 경우 Trident는 스토리지 백엔드에서 볼륨을 확장하고 장치를 다시 스캔한 후 파일 시스템 크기를 조정합니다.
- 연결되지 않은 PV의 크기를 조정하려고 할 때 Trident는 스토리지 백엔드에서 볼륨을 확장합니다. PVC가 파드에 바인딩된 후 Trident는 디바이스를 다시 스캔하고 파일 시스템의 크기를 조정합니다. 그런 다음 Kubernetes는 확장 작업이 성공적으로 완료된 후 PVC 크기를 업데이트합니다.

이 예에서는 `san-pvc`를 사용하는 Pod가 생성됩니다.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-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:    ubuntu-pod
```

4단계: **PV** 확장

1Gi에서 2Gi로 생성된 PV의 크기를 조정하려면 PVC 정의를 편집하고 `spec.resources.requests.storage`을 2Gi로 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 검증

PVC, PV 및 Trident 볼륨의 크기를 확인하여 확장이 올바르게 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

FC 볼륨 확장

CSI 프로비저너를 사용하여 FC 영구 볼륨(PV)을 확장할 수 있습니다.

참고 FC 볼륨 확장은 `ontap-san` 드라이버에서 지원되며 Kubernetes 1.16 이상이 필요합니다.

1단계: 볼륨 확장을 지원하도록 StorageClass 구성

StorageClass 정의를 편집하여 `allowVolumeExpansion` 필드를 `true`로 설정합니다.

```
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` 매개 변수를 포함하도록 편집합니다.

2단계: 생성한 **StorageClass**로 **PVC** 만들기

PVC 정의를 편집하고 새로 원하는 크기를 반영하도록 `spec.resources.requests.storage`을 업데이트합니다(원래 크기보다 커야 함).

```
cat pvc-ontapsan.yaml
```

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

Trident는 영구 볼륨(PV)을 생성하고 이 영구 볼륨 클레임(PVC)과 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: **PVC**를 연결하는 **POD** 정의

크기를 조정할 PV를 포드에 연결합니다. FC PV의 크기를 조정할 때는 두 가지 시나리오가 있습니다.

- PV가 포드에 연결된 경우 Trident는 스토리지 백엔드에서 볼륨을 확장하고 장치를 다시 스캔한 후 파일 시스템 크기를 조정합니다.
- 연결되지 않은 PV의 크기를 조정하려고 할 때 Trident는 스토리지 백엔드에서 볼륨을 확장합니다. PVC가 파드에 바인딩된 후 Trident는 디바이스를 다시 스캔하고 파일 시스템의 크기를 조정합니다. 그런 다음 Kubernetes는 확장 작업이 성공적으로 완료된 후 PVC 크기를 업데이트합니다.

이 예에서는 `san-pvc`를 사용하는 Pod가 생성됩니다.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-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:    ubuntu-pod
```

4단계: **PV** 확장

1Gi에서 2Gi로 생성된 PV의 크기를 조정하려면 PVC 정의를 편집하고 `spec.resources.requests.storage`을 2Gi로 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 검증

PVC, PV 및 Trident 볼륨의 크기를 확인하여 확장이 올바르게 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 볼륨 확장

Trident는 `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` 및 `azure-netapp-files` 백엔드에서 프로비저닝된 NFS PV의 볼륨 확장을 지원합니다.

1단계: 볼륨 확장을 지원하도록 StorageClass 구성

NFS PV의 크기를 조정하려면 관리자가 먼저 `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 이미 스토리지 클래스를 생성한 경우 `kubectl edit storageclass`을 사용하여 기존 스토리지 클래스를

편집하여 볼륨 확장을 허용할 수 있습니다.

2단계: 생성한 **StorageClass**로 **PVC** 만들기

```
cat pvc-ontapnas.yaml
```

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

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: **PV** 확장

새로 생성한 20 MiB PV의 크기를 1 GiB로 조정하려면 PVC를 편집하고 `spec.resources.requests.storage`을 1 GiB로 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: 확장 검증

PVC, PV 및 Trident 볼륨의 크기를 확인하여 크기 조정이 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

RWX NVMe 하위 시스템 제한 사항 이해

ReadWriteMany(RWX) 볼륨은 NVMe 프로토콜을 사용하며 볼륨당 64개 노드의 확장성 제한이 있습니다. 다음은 제한 사항, 관련 NVMe 서브시스템 아키텍처 및 필요한 해결 단계를 설명합니다.

64개 노드 제한 이해

NVMe 프로토콜과 함께 ReadWriteMany(RWX) 볼륨을 사용하려는 경우, 단일 RWX NVMe 볼륨은 Kubernetes 클러스터에서 64개 이상의 노드에 마운트할 수 없습니다.

64개 이상의 노드에 걸쳐 동일한 RWX NVMe PersistentVolumeClaim을 마운트하는 워크로드를 예약하지 마십시오.

이 제한 사항은 NVMe 프로토콜을 사용하는 RWX 볼륨에만 적용됩니다.

NVMe 하위 시스템 모델 이해

볼륨별 하위 시스템 모델(Trident 26.02 이전 릴리스)

Trident 26.02 이전 릴리스에서는 RWX NVMe 볼륨이 볼륨별 서브시스템 모델을 사용하여 프로비저닝됩니다. 각 RWX

NVMe 볼륨은 ONTAP의 자체 전용 NVMe 서브시스템에 매핑됩니다.

이 모델은 단순하지만 확장성 한계가 낮습니다. 대규모 Kubernetes 클러스터에서는 각 RWX 볼륨이 전용 서브시스템을 사용하기 때문에 서브시스템 컨트롤러 한계에 빠르게 도달합니다.

슈퍼 서브시스템 모델**(Trident 26.02**에서 도입**)**

Trident 26.02부터 RWX NVMe 볼륨은 공유 슈퍼 서브시스템 모델을 사용합니다. 여러 RWX NVMe 볼륨이 동일한 NVMe 서브시스템을 공유합니다.

각 슈퍼 서브시스템은 최대 1024개의 네임스페이스(볼륨)를 지원합니다. 이 모델은 RWX 워크로드의 확장성을 크게 향상시키고 ONTAP 서브시스템 한계에 도달할 가능성을 줄입니다.

각 RWX NVMe 볼륨은 최대 64개의 노드를 지원합니다.

오류 증상 파악

대규모로 RWX NVMe 볼륨을 생성하거나 연결하는 경우 다음과 유사한 오류가 발생할 수 있습니다.

```
Maximum number of controllers reached. No more controllers can be created.
```

이 오류는 ONTAP NVMe 서브시스템 컨트롤러 제한에 도달했음을 나타냅니다.

하위 시스템 제한 오류 해결

볼륨별 하위 시스템 제한을 넘어 슈퍼 하위 시스템 모델을 활용하려면 Trident 26.02 이상으로 업그레이드하십시오.

Trident를 업그레이드하여 슈퍼 서브시스템 모델을 적용합니다

RWX NVMe 볼륨에 슈퍼 서브시스템 모델을 적용하려면 다음을 수행합니다.

1. Trident를 버전 26.02 이상으로 업그레이드하십시오.
2. RWX NVMe 볼륨을 사용하는 모든 파드를 0개의 복제본으로 축소합니다.
3. RWX NVMe 볼륨을 적극적으로 사용하는 워크로드가 없는지 확인하십시오.
4. 포드를 다시 확장합니다.

이 재시작 시퀀스는 RWX NVMe 볼륨이 super-subsystem 모델을 사용하여 연결되도록 보장합니다.

- 이 제한 사항은 NVMe 프로토콜을 사용하는 RWX 볼륨에만 적용됩니다.
- 64노드 제한은 RWX NVMe 볼륨당 적용됩니다.
- 다른 액세스 모드 및 다른 프로토콜은 영향을 받지 않습니다.

컨트롤러 확장성

Trident는 여러 스토리지 드라이버에 걸쳐 향상된 동시성을 통해 컨트롤러 확장성을 제공합니다. 고객은 일반 출시 시점에 컨트롤러 확장성을 지원하는 Trident 드라이버와 Trident 26.02에서 기술 미리 보기로 제공되는 드라이버를 확인할 수 있습니다. 이를 통해 확장 가능한 Kubernetes

환경을 위한 정보에 기반한 배포 결정을 내리고 적절한 위험 관리를 수행할 수 있습니다.

핵심 개념 및 정의

컨트롤러 확장성

컨트롤러 확장성은 Trident 컨트롤러가 단일 잠금 뒤에서 여러 스토리지 작업을 직렬화하는 대신 병렬로 처리할 수 있는 능력을 의미합니다. 이러한 작업에는 볼륨 생성, 삭제, 크기 조정, 스냅샷 생성 및 삭제, 볼륨 게시 및 게시 취소, 백엔드 관리가 포함됩니다.

컨트롤러 확장성이 활성화되면 서로 다른 볼륨 및 백엔드에 대한 작업이 동시에 진행됩니다. 이는 PersistentVolumeClaim 및 VolumeSnapshot 작업이 동시에 많이 발생하는 환경에서 처리량을 높이고 엔드 투 엔드 작업 시간을 단축합니다.

컨트롤러 확장성 지원

Trident는 스토리지 드라이버에 따라 다양한 성숙도 수준의 컨트롤러 확장성을 지원합니다.

일반 가용성

다음 드라이버는 Trident 26.02의 정식 출시 버전에서 컨트롤러 확장성을 지원합니다.

- `ontap-san`
- `ontap-nas`
- `google-cloud-netapp-volumes`

참고

`google-cloud-netapp-volumes`와 `google-cloud-netapp-volumes-san` 드라이버는 서로 다릅니다. `google-cloud-netapp-volumes`만 지원됩니다. 백엔드 구성이나 예제에서 `google-cloud-netapp-volumes-san`를 사용하지 마십시오.

컨트롤러 확장성 활성화

컨트롤러 확장성은 `enableConcurrency` 구성 옵션을 통해 제어됩니다. 이 옵션은 Trident 설치 중 또는 기존 배포 업데이트 시 명시적으로 활성화해야 합니다.

Trident 운영자 배포

Trident operator에서 컨트롤러 확장성을 활성화하려면, `enableConcurrency`를 `true`로 `TridentOrchestrator` 커스텀 리소스에서 설정하세요.

신규 설치

`TridentOrchestrator` CR을 생성하거나 편집하고 `enableConcurrency`을(를) `true`(으)로 설정합니다:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

CR 적용:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

기존 설치

기존 `TridentOrchestrator` CR에 패치를 적용하여 컨트롤러 확장성을 활성화합니다.

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

설정이 적용되었는지 확인합니다.

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Helm 배포

Helm을 사용하여 컨트롤러 확장성을 활성화하려면 `enableConcurrency` 값을 `true`로 설정하십시오.

신규 설치

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

기존 설치

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

또는, 사용자 지정 `values.yaml` 파일에서 `enableConcurrency`을(를) `true`(으)로 설정하십시오:

```
# values.yaml
enableConcurrency: true
```

그런 다음 값 파일을 사용하여 설치 또는 업그레이드하십시오.

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

tridentctl 배포

`tridentctl`을(를) 사용하여 컨트롤러 확장성을 활성화하려면 설치 중에 `--enable-concurrency` 플래그를 전달하십시오.

신규 설치

```
tridentctl install -n trident --enable-concurrency
```

기존 설치

기존 `tridentctl` 기반 배포에서 컨트롤러 확장성을 활성화하려면 다음 플래그를 사용하여 제거한 후 다시 설치하십시오.

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

컨트롤러 확장성이 활성화되어 있는지 확인하십시오

컨트롤러 확장성을 활성화한 후 컨트롤러 파드 로그를 확인하여 Trident 컨트롤러가 동시 실행 기능을 활성화한 상태로 실행 중인지 확인하십시오.

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

동시 실행이 활성화되었음을 나타내는 로그 항목이 표시됩니다.

기술 미리 보기

다음 드라이버는 Trident 26.02에서 기술 미리 보기로 컨트롤러 확장성을 지원합니다.

- `nas-eco`
- `san-eco`

다음 드라이버의 경우:

- 컨트롤러 동시성을 평가 및 테스트할 수 있습니다.
- 향후 릴리스에서 동작 방식이 변경될 수 있습니다.
- 운영 환경에서는 사용하지 않는 것이 좋습니다

동시성 동작

컨트롤러 확장성이 활성화된 경우:

- Trident는 단일 전역 잠금을 리소스별 세분화된 잠금으로 대체합니다.
- 동일한 리소스를 수정하는 작업은 데이터 일관성을 유지하기 위해 직렬화됩니다
- 리소스에서 읽기만 하는 작업은 해당 리소스에 대한 다른 읽기 작업과 동시에 진행될 수 있습니다.
- Trident는 백엔드 스토리지 시스템의 과부하를 방지하기 위해 관리 LIF당 동시 ONTAP API 요청을 20개로 제한합니다
- 여러 백엔드가 동일한 관리 LIF를 공유하는 경우 이 20개 요청 제한을 공유합니다

알려진 제한 사항 및 고려 사항

컨트롤러 확장성에는 다음 사항이 적용됩니다.

- 동시성은 Trident 컨트롤러에 의해 내부적으로 관리됩니다
- 이번 릴리스에는 사용자가 구성할 수 있는 동시성 제한이 없습니다
- 전체 처리량은 다음 요소에 따라 달라집니다.
  - 사용 중인 스토리지 드라이버
  - 백엔드 응답성
  - Kubernetes API 서버 성능
- 높은 동시성은 백엔드 스토리지 시스템의 부하를 증가시킬 수 있습니다

주의사항 및 제한사항

Trident 26.02에는 다음과 같은 제한 사항이 적용됩니다.

- 컨트롤러 확장성 동작은 모든 드라이버에서 동일하지 않습니다
- 기술 미리 보기 드라이버에서 다음과 같은 현상이 나타날 수 있습니다.
  - 높은 부하에서 일관되지 않은 성능
  - 릴리스 간 동작 변경 사항
- 병렬 실행으로 인해 동시 작업 디버깅이 더 복잡해질 수 있습니다.
- 지표 및 로그에는 인터리브된 작업 출력이 표시될 수 있습니다

권장 사항

- 높은 확장성이 필요한 운영 환경에는 일반 가용성(GA) 드라이버를 사용하십시오

- 비프로덕션 환경에서 기술 미리 보기 드라이버를 평가합니다
- 대규모로 운영할 때 백엔드 및 컨트롤러 성능 모니터링
- 자동화 스크립트에서 작업 순서를 가정하지 마십시오

자동 볼륨 확장

자동 볼륨 확장을 사용하면 Trident에서 프로비저닝한 영구 볼륨이 사용된 용량이 정의된 임계값에 도달할 때 자동으로 증가합니다. 이 기능은 운영 오버헤드를 줄이고 용량 고갈로 인한 애플리케이션 중단을 방지하는 데 도움이 됩니다. 자동 볼륨 확장은 Autogrow 정책을 사용하여 구현됩니다. Autogrow 정책은 다음을 정의합니다.

- 확장을 트리거하는 사용률 임계값
- 볼륨이 증가하는 양
- 볼륨이 도달할 수 있는 최대 크기

참고	정의된 사용률 임계값을 초과하면 볼륨 크기가 자동으로 증가합니다. 볼륨은 자동으로 축소되지 않습니다.

요구 사항

자동 볼륨 확장을 구성하기 전에 다음 요구 사항을 충족하는지 확인하십시오.

- Trident 26.02 이상
- 역할 기반 액세스 제어 권한으로 `TridentAutogrowPolicy` 사용자 지정 리소스 생성
- StorageClasses로 구성되었습니다 `allowVolumeExpansion: true`
- 지원되는 ONTAP 프로토콜:
  - 네트워크 파일 시스템(NFS)
  - Internet Small Computer Systems Interface(iSCSI)
  - 비휘발성 메모리 익스프레스(NVMe)
  - 파이버 채널 프로토콜(FCP)

제한 사항

- ONTAP 9.16.1 이전 버전의 ONTAP Non-Volatile Memory Express 원시 블록 볼륨은 자동 확장을 지원하지 않습니다.
- 스토리지 영역 네트워크 볼륨의 경우, `growthAmount`이(가) 50메비바이트 이하이면 Trident는 크기 조정 전에 자동으로 값을 51메비바이트로 늘립니다. 단, 결과 크기가 `maxSize`을(를) 초과하지 않는 경우에 한합니다.
- 브라운필드 환경에서는 볼륨 게시 마이그레이션 동작으로 인해 특정 기존 볼륨에 대해 자동 확장이 작동하지 않을 수 있습니다.
- 볼륨이 `maxSize`에 도달하면 더 이상 확장되지 않습니다.
- 자동 볼륨 확장을 위해 지원되는 프로토콜:
  - 네트워크 파일 시스템(NFS)

- Internet Small Computer Systems Interface(iSCSI)
- 비휘발성 메모리 익스프레스(NVMe)
- 파이버 채널 프로토콜(FCP)

자동 확장 정책이 적용된 볼륨 프로비저닝

자동 확장 정책은 두 가지 수준에서 구성할 수 있습니다.

- 스토리지 클래스 수준: 모든 볼륨의 기본값을 설정합니다(주석 사용)
- PVC 레벨: 스토리지 클래스 기본값을 재정의합니다(annotation 사용)

자동 증가 정책 생성

자동 확장 정책을 사용하면 볼륨이 정의된 용량 임계값에 도달할 때 볼륨이 자동으로 확장됩니다.

다음 사항을 확인하십시오.

- Trident 26.02 이상이 설치되어 있습니다
- 역할 기반 액세스 제어 권한을 통해 `TridentAutogrowPolicy` 리소스를 생성합니다
- 워크로드 증가 요구 사항에 대한 이해

자동 확장 정책은 볼륨이 정의된 용량 임계값에 도달했을 때 자동으로 확장되는 방식을 정의합니다.

워크플로의 어느 단계에서든 자동 확장 정책을 생성할 수 있습니다.

- StorageClasses와 볼륨이 생성되기 전에
- StorageClasses가 존재한 후
- 볼륨이 프로비저닝된 후

이러한 유연성을 통해 기존 리소스를 다시 생성하지 않고도 자동 확장을 도입할 수 있습니다.

자동 확장 정책 사양

자동 증가 정책은 다음과 같이 정의된 Kubernetes 사용자 지정 리소스입니다.

필드	설명	형식	필수	예	기본값
이름	고유 정책 식별자	문자열	예	production-db-policy	None
usedThreshold	확장을 트리거하는 용량 비율	백분율 문자열	예	"80%"	None
growthAmount	임계값에 도달했을 때 증가할 양	백분율 또는 크기	아니요	"10%" 또는 "5Gi"	"10%"
maxSize	최대 볼륨 크기 제한	Kubernetes 수량	아니요	"500Gi"	무제한

자동 증가 정책 생성

단계

1. Autogrow Policy를 정의하는 YAML 파일을 생성합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. 클러스터에 정책을 적용합니다.

```
kubectl apply -f autogrow-policy.yaml
```

3. 정책이 생성되었는지 확인하십시오.

```
kubectl get tridentautogrowpolicy standard-autogrow
```

예상 출력

```
NAME                USED THRESHOLD   GROWTH AMOUNT   STATE
standard-autogrow   80%              10%             Success
```

정책 상태

정책을 생성하면 Trident가 사양을 검증하고 다음 상태 중 하나를 할당합니다.

상태	설명	조치 필요
성공	정책이 검증되어 사용할 준비가 되었습니다.	없음.
실패	유효성 검사 오류가 감지되었습니다.	사양을 검토하고 수정하십시오.
삭제 중	삭제가 진행 중입니다.	완료될 때까지 기다립니다.

정책을 **StorageClass**에 연결합니다

`trident.netapp.io/autogrowPolicy` 어노테이션을 사용하여 StorageClass에 자동 확장 정책을 연결할 수 있습니다. 해당 StorageClass에서 프로비저닝된 모든 볼륨은 이 정책을 상속받습니다.

참고 StorageClass는 다음을 가져야 합니다 `allowVolumeExpansion: true`.

단계

1. Autogrow Policy 어노테이션을 사용하여 StorageClass를 생성하거나 수정합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. StorageClass를 적용합니다.

```
kubectl apply -f storageclass.yaml
```

3. 주석을 확인하세요:

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

예상 출력

```
production-db-policy
```

정책 우선 순위

StorageClass와 PVC 모두에 Autogrow Policy 어노테이션이 설정된 경우 Trident는 다음 우선순위 규칙을 적용합니다.

1. **PVC** 주석이 우선합니다. PVC에 `trident.netapp.io/autogrowPolicy`가 설정되면 해당 값이 항상 사용됩니다.
2. **StorageClass** 어노테이션은 **PVC**에 어노테이션이 없는 경우에만 적용됩니다.

3. 어노테이션이 둘 다 없는 경우 Autogrow Policy가 적용되지 않습니다.

StorageClass 어노테이션	**PVC** 주석	효과적인 행동
`trident.netapp.io/autogrowPolicy: standard-agp`	설정되지 않음	용도 `standard-agp`.
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: logs-policy`	`logs-policy` 사용(PVC가 StorageClass를 재정의함).
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: "none"`	자동 증가 정책 없음(PVC에서 자동 증가 기능을 비활성화함).
설정되지 않음	`trident.netapp.io/autogrowPolicy: dev-policy`	용도 `dev-policy`.
설정되지 않음	설정되지 않음	자동 증가 정책 없음.

구성 예

다음 예시는 다양한 사용 사례에 대한 일반적인 Autogrow Policy 구성을 보여줍니다.

운영 데이터베이스에 대한 보수적인 정책

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

고정 증가 단위로 로그 스토리지

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

기본값이 포함된 최소 정책

`growthAmount`및 `maxSize`를 생략하면 Trident는 기본값(`10%` 증가, 무제한 크기)을 사용합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

사용자 지정 **maxSize** 및 기본 **growthAmount**가 있는 정책

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

무제한 **maxSize**로 공격적인 성장

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

소수 백분율이 포함된 정책

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```

참고 | 소수점 이하 백분율도 지원됩니다. 소수점 이하 세 자리 이상을 지정하면 Trident가 값을 소수점 셋째 자리까지 반올림합니다.

자동 확장 기능이 있는 **NAS StorageClass**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

자동 확장 기능이 있는 **SAN StorageClass**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

자동 증가 정책 관리

Autogrow 정책을 생성한 후에는 필요에 따라 정책을 보고, 업데이트하고, 삭제할 수 있습니다. 또한 특정 정책을 사용 중인 볼륨을 모니터링할 수도 있습니다.

자동 증가 정책 보기

모든 정책 나열

`kubectl`를 사용하여 클러스터의 모든 `Autogrow` 정책을 나열합니다.

```
kubectl get tridentautogrowpolicy
```

또는 다음을 사용하십시오 `tridentctl`:

```
tridentctl get autogrowpolicy
```

정책 세부 정보 보기

정책의 전체 사양 및 상태를 보려면 다음을 수행합니다.

```
kubectl describe tridentautogrowpolicy production-db-policy
```

YAML 형식으로 정책과 해당 정책에 연결된 볼륨을 보려면 다음을 수행합니다.

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

자동 증가 정책 업데이트

기존 정책을 수정하여 임계값, 증가량 또는 최대 크기를 변경할 수 있습니다. 변경 사항은 해당 정책을 사용하는 모든 볼륨에 즉시 적용됩니다.

중요함	변경 사항은 현재 해당 정책을 사용 중인 모든 볼륨에 적용됩니다. 가능하면 먼저 비운영 환경에서 변경 사항을 테스트하십시오.

단계

1. 정책을 편집합니다.

   ```
   kubectl edit tridentautogrowpolicy production-db-policy
   ```

2. `spec` 필드를 필요에 따라 수정합니다.

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

3. 저장 후 종료합니다. 변경 사항은 즉시 적용됩니다.

업데이트 고려 사항

- 즉시 효력 발생: 해당 정책을 사용하는 모든 볼륨은 다음 증가 평가 시 새로운 매개변수를 적용받습니다.
- 볼륨 재시작이 필요 없습니다. 변경 사항은 다음 확장 작업에 적용됩니다.
- 먼저 테스트: 가능하면 비운영 환경에서 변경 사항을 검증하십시오.
- 변경 사항 전달: 공유 정책을 수정할 때 팀에 알립니다.

자동 증가 정책 삭제

자동 확장 정책은 볼륨이 활발하게 사용 중인 동안 실수로 삭제되는 것을 방지하기 위해 finalizer 보호 기능을 사용합니다.

단계

1. 정책을 삭제합니다.

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. 볼륨이 여전히 정책을 사용 중인 경우 삭제 작업이 `Deleting` 상태로 전환됩니다. 어떤 볼륨이 영향을 받는지 확인하십시오:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. 영향을 받는 각 볼륨에서 정책을 제거합니다. 다음 옵션 중 하나를 선택하십시오.
   - 옵션 **A:** 주석을 `**"none"**`로 설정하여 자동 크기 증가를 명시적으로 비활성화:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

   - 옵션 **B:** 주석을 완전히 제거합니다.

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

삭제 동작

시나리오	동작
해당 정책을 사용하는 볼륨은 없습니다	정책은 즉시 삭제됩니다.
볼륨이 정책을 사용하고 있습니다	정책이 `Deleting` 상태에 진입합니다. 최종화 도구는 모든 볼륨이 제거될 때까지 완료를 차단합니다.
모든 볼륨이 정책에서 제거됩니다	최종 결정자가 제거되고 정책이 삭제됩니다.

자동 증가 정책 사용량 모니터링

정책을 사용하여 볼륨 확인

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

볼륨이 사용하는 정책 찾기

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

정책 이벤트 모니터링

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

지원되는 프로토콜

Autogrow는 다음과 같은 스토리지 프로토콜을 지원합니다:

- NFS
- iSCSI
- FCP
- NVMe

참고	SAN 볼륨의 경우, 구성된 `growthAmount`이 50MiB 이하이면 Trident는 크기 조정 작업 시 증가량을 자동으로 51MB로 늘립니다. 단, 최종 크기가 `maxSize`를 초과하지 않는 경우에 한합니다.

알려진 제한 사항

- * ONTAP NVMe 원시 블록 볼륨:* ONTAP 9.16.1 이전 버전으로 생성된 볼륨은 자동 확장을 지원하지 않습니다.
- 기존 볼륨**(**브라운필드 배포**):** 유효한 Autogrow Policy가 적용되었더라도 기존 볼륨에서는 Autogrow 기능이 작동하지 않을 수 있습니다. 이는 볼륨 게시 마이그레이션이 진행 중이기 때문입니다. 마이그레이션이 완료되었는지 확인하려면 Trident 컨트롤러 로그에서 "`Migration completed`" 메시지를 확인하십시오.

자주 묻는 질문

Trident는 언제 임계값을 평가합니까?

Trident는 볼륨 사용량을 지속적으로 모니터링합니다. 사용 용량이 `usedThreshold`를 초과하면 Trident는 내부 크기 조정 요청을 생성하고 구성된 `growthAmount`만큼 볼륨을 확장합니다.

예를 들어, 이 정책은 용량의 80%에 도달하면 확장을 시작하고 매번 10%씩 증가시켜 최대 500 GiB까지 늘립니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

볼륨이 이미 프로비저닝된 후에도 정책을 적용할 수 있습니까?

예. 자동 증가 정책은 언제든지 생성할 수 있으며, `trident.netapp.io/autogrowPolicy` 어노테이션을 추가하거나 업데이트하여 기존 PVC에 적용할 수 있습니다. PVC나 StorageClass를 다시 생성할 필요가 없습니다.

기존 PVC에 정책을 적용하려면:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

기존 StorageClass에 정책을 적용하려면 다음을 수행합니다.

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

StorageClass와 PVC 모두에 자동 증가 정책을 설정하면 어떻게 되나요?

PVC 어노테이션이 항상 우선합니다. PVC에 `trident.netapp.io/autogrowPolicy` 어노테이션이 있는 경우 Trident는 StorageClass에서 지정한 값과 관계없이 해당 값을 사용합니다. 자세한 내용은 "정책 우선 순위"를 참조하십시오.

예를 들어, 다음과 같은 StorageClass가 있습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

그리고 StorageClass 정책을 재정의하는 이 PVC는 다음과 같습니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident는 `logs-policy`를 위해 `database-pvc`를 사용하며, `standard-agp`는 사용하지 않습니다.

특정 볼륨의 자동 확장을 비활성화하려면 어떻게 해야 하나요?

PVC 주석을 `"none"`로 설정합니다. 이렇게 하면 해당 볼륨에 대한 StorageClass 수준 정책이 재정의됩니다.

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

자동 크기 증가 기능이 비활성화되어 있는지 확인할 수 있습니다.

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

예상 출력

```
none
```

볼륨이 **maxSize**에 도달하면 어떻게 됩니까**?**

Trident가 볼륨 확장을 중지합니다. 사용량이 `usedThreshold`를 초과하여 계속 증가하더라도 해당 볼륨에 대한 추가 크기 조정 요청은 생성되지 않습니다.

예를 들어, 이 정책을 사용하면 Trident는 볼륨이 100GiB에 도달하면 볼륨 증가를 중지합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

무제한 성장을 허용하려면 `maxSize`을 생략하거나 `0`로 설정하십시오:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

볼륨을 재시작하지 않고 정책을 변경할 수 있습니까**?**

예. 정책을 업데이트하면 해당 정책을 사용하는 모든 볼륨이 다음 용량 증가 평가 시 새 매개변수를 적용합니다. 볼륨을 다시 시작할 필요는 없습니다.

정책을 업데이트하려면 다음을 수행합니다.

```
kubectl edit tridentautogrowpolicy production-db-policy
```

필요에 따라 필드를 수정합니다.

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

저장 후 종료합니다. 업데이트된 정책을 확인합니다.

```
kubectl get tridentautogrowpolicy production-db-policy
```

예상 출력

```
NAME                    USED THRESHOLD   GROWTH AMOUNT   STATE
production-db-policy    75%              20%             Success
```

내 정책이 실패 상태인 이유는 무엇입니까?

`Failed` 상태는 정책 사양에 유효성 검사 오류가 있음을 나타냅니다. 다음 명령을 실행하여 오류 세부 정보를 확인하십시오.

```
kubectl describe tridentautogrowpolicy <policy-name>
```

일반적인 원인으로는 유효하지 않은 `usedThreshold`(1~99%여야 함), `growthAmount`을 초과하는 `maxSize` 또는 유효하지 않은 Kubernetes 수량 형식이 있습니다. 사양을 수정하고 다시 적용하십시오:

```
kubectl apply -f autogrow-policy.yaml
```

정책을 삭제할 수 없는 이유는 무엇인가요?

정책은 finalizer 보호 기능을 사용합니다. 볼륨이 여전히 정책을 사용 중인 경우 삭제 작업은 `Deleting` 상태로 들어가며 모든 볼륨이 정책에서 제거될 때까지 대기합니다.

영향을 받는 볼륨을 식별하십시오.

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

그런 다음 각 PVC에서 주석을 제거합니다.

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

모든 볼륨이 제거되면 파이널라이저가 해제되고 정책이 삭제됩니다.

자동 크기 증가 기능은 모든 **ONTAP** 백엔드에서 작동하나요?

자동 확장 기능은 NFS, iSCSI, FCP 및 NVMe 프로토콜을 지원합니다. 단, NVMe 원시 블록 볼륨을 사용하려면 ONTAP 9.16.1 이상 버전이 필요합니다.

기존 배포 환경의 경우 자동 확장이 적용되기 전에 볼륨 게시 마이그레이션이 완료되어야 할 수 있습니다. Trident 컨트롤러 로그를 확인하여 마이그레이션 상태를 확인하십시오.

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

다음 StorageClass 예제는 NAS 및 SAN 백엔드에 대해 자동 확장이 구성된 방식을 보여줍니다.

NAS 백엔드

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN 백엔드

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN 볼륨의 최소 증가량은 얼마입니까?

SAN 볼륨의 경우, 유효 최소 증가량은 51MB입니다. `growthAmount`를 50MiB 이하로 설정한 경우, Trident는 크기 조정 작업 시 증가량을 자동으로 51MB로 늘립니다.

예를 들어, 이 정책은 `growthAmount`을 `"40Mi"`로 설정하지만, Trident는 이 정책을 사용하는 모든 SAN 볼륨에 대해 51MB의 증가분을 적용합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

이러한 자동 조정을 방지하려면 `growthAmount`을 50 MiB보다 큰 값으로 설정하십시오.

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

볼륨 가져오기

기존 스토리지 볼륨을 Kubernetes PV로 가져오려면 `tridentctl import`를 사용하거나 Trident 가져오기 어노테이션을 사용하여 영구 볼륨 클레임(PVC)을 생성할 수 있습니다.

개요 및 고려 사항

다음과 같은 경우 Trident로 볼륨을 가져올 수 있습니다.

- 애플리케이션을 컨테이너화하고 기존 데이터 세트를 재사용합니다
- 임시 애플리케이션에 데이터 세트의 클론 사용
- 실패한 Kubernetes 클러스터 재구축
- 재해 복구 중 애플리케이션 데이터 마이그레이션

고려 사항

볼륨을 가져오기 전에 다음 사항을 검토하십시오.

- Trident는 RW(읽기-쓰기) 유형의 ONTAP 볼륨만 가져올 수 있습니다. DP(데이터 보호) 유형 볼륨은 SnapMirror 타겟 볼륨입니다. 볼륨을 Trident로 가져오기 전에 미러 관계를 끊어야 합니다.
- 활성 연결이 없는 볼륨을 가져오는 것이 좋습니다. 활성 상태인 볼륨을 가져오려면 볼륨 클론을 생성한 다음 가져오기를 수행하십시오.

경고	이는 특히 블록 볼륨의 경우 중요한데, Kubernetes가 이전 연결을 인식하지 못하고 활성 볼륨을 Pod에 연결할 수 있기 때문입니다. 이로 인해 데이터 손상이 발생할 수 있습니다.

- `StorageClass`PVC에 지정되어야 하지만 Trident는 가져오기 중에 이 매개 변수를 사용하지 않습니다. 스토리지 클래스는 볼륨 생성 중에 스토리지 특성을 기반으로 사용 가능한 풀에서 선택하는 데 사용됩니다. 볼륨이 이미 존재하므로 가져오기 중에 풀 선택이 필요하지 않습니다. 따라서 볼륨이 PVC에 지정된 스토리지 클래스와 일치하지 않는 백엔드 또는 풀에 존재하더라도 가져오기가 실패하지 않습니다.
- 기존 볼륨 크기는 PVC에서 확인 및 설정됩니다. 스토리지 드라이버가 볼륨을 가져온 후 PVC에 대한 ClaimRef를 포함하는 PV가 생성됩니다.
  - 재확보 정책은 처음에 PV에서 `retain`로 설정됩니다. Kubernetes가 PVC와 PV를 성공적으로 바인딩한 후 재확보 정책이 스토리지 클래스의 재확보 정책과 일치하도록 업데이트됩니다.
  - 스토리지 클래스의 회수 정책이 `delete`인 경우, PV가 삭제될 때 스토리지 볼륨도 삭제됩니다.
- 기본적으로 Trident는 PVC를 관리하고 백엔드에서 FlexVol 볼륨과 LUN의 이름을 변경합니다. 관리되지 않는 볼륨을 가져오려면 `--no-manage` 플래그를, 볼륨 이름을 유지하려면 `--no-rename` 플래그를 전달할 수 있습니다.
  - `--no-manage`* - `--no-manage` 플래그를 사용하면 Trident는 객체의 수명 주기 동안 PVC 또는 PV에 대해 추가 작업을 수행하지 않습니다. PV가 삭제될 때 스토리지 볼륨은 삭제되지 않으며 볼륨 클론 및 볼륨 크기 조정과 같은 다른 작업도 무시됩니다.
  - `--no-rename`* - `--no-rename` 플래그를 사용하면 Trident는 볼륨을 가져오는 동안 기존 볼륨 이름을 유지하고 볼륨의 수명 주기를 관리합니다. 이 옵션은 `ontap-nas`, `ontap-san`(ASA r2 시스템 포함) 및 `ontap-san-economy` 드라이버에서만 지원됩니다.

팁 | 이러한 옵션은 컨테이너화된 워크로드에 Kubernetes를 사용하면서도 스토리지 볼륨의 수명 주기는 Kubernetes 외부에서 관리하려는 경우에 유용합니다.

- PVC 및 PV에는 볼륨을 가져왔는지 여부와 PVC 및 PV가 관리되는지 여부를 나타내는 이중 목적의 주석이 추가됩니다. 이 주석은 수정하거나 제거해서는 안 됩니다.

볼륨 가져오기

`tridentctl import`를 사용하거나 `Trident` 가져오기 주석이 포함된 `PVC`를 생성하여 볼륨을 가져올 수 있습니다.

참고 | PVC 주석을 사용하는 경우 볼륨을 가져오기 위해 `tridentctl`를 다운로드하거나 사용할 필요가 없습니다.

tridentctl 사용

단계

1. PVC를 생성하는 데 사용할 PVC 파일(예: `pvc.yaml`)을 생성합니다. PVC 파일에는 `name`, `namespace`, `accessModes` 및 `storageClassName`가 포함되어야 합니다. 선택적으로 PVC 정의에서 `unixPermissions`를 지정할 수 있습니다.

   다음은 최소 사양의 예입니다.

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

   참고 | 필수 매개변수만 포함하십시오. PV 이름이나 볼륨 크기와 같은 추가 매개변수는 가져오기 명령이 실패할 수 있습니다.

2. `tridentctl import` 명령을 사용하여 볼륨이 포함된 Trident 백엔드의 이름과 스토리지에서 볼륨을 고유하게 식별하는 이름(예: ONTAP FlexVol, Element Volume)을 지정합니다. `-f` 인수는 PVC 파일의 경로를 지정하는 데 필요합니다.

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

PVC 주석 사용

단계

1. 필요한 Trident 가져오기 주석이 포함된 PVC YAML 파일(예: `pvc.yaml`)을 생성합니다. PVC 파일에는 다음 내용이 포함되어야 합니다.

   - `name` 및 `namespace` 메타데이터
   - `accessModes`, `resources.requests.storage` 및 `storageClassName` 사양
   - 주석:
     - `trident.netapp.io/importOriginalName`: 백엔드의 볼륨 이름
     - `trident.netapp.io/importBackendUUID`: 볼륨이 존재하는 백엔드 UUID
     - `trident.netapp.io/notManaged` (선택 사항): 관리되지 않는 볼륨의 경우 `"true"`로 설정합니다. 기본값은 `"false"`입니다.

     다음은 관리형 볼륨을 가져오기 위한 예시 사양입니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. PVC YAML 파일을 Kubernetes 클러스터에 적용합니다.

```
kubectl apply -f <pvc-file>.yaml
```

Trident는 볼륨을 자동으로 가져와 PVC에 바인딩합니다.

예

지원되는 드라이버에 대한 다음 볼륨 가져오기 예를 검토하십시오.

ONTAP NAS 및 ONTAP NAS FlexGroup

Trident는 `ontap-nas` 및 `ontap-nas-flexgroup` 드라이버를 사용하여 볼륨 가져오기를 지원합니다.

참고

- Trident는 `ontap-nas-economy` 드라이버를 사용한 볼륨 가져오기를 지원하지 않습니다.
- `ontap-nas` 및 `ontap-nas-flexgroup` 드라이버는 중복된 볼륨 이름을 허용하지 않습니다.

`ontap-nas` 드라이버를 사용하여 생성된 각 볼륨은 `ONTAP` 클러스터의 `FlexVol` 볼륨입니다. `ontap-nas` 드라이버를 사용하여 `FlexVol` 볼륨을 가져오는 과정도 동일합니다. `ONTAP` 클러스터에 이미 존재하는 `FlexVol` 볼륨은 `ontap-nas` PVC로 가져올 수 있습니다. 마찬가지로 `FlexGroup` 볼륨도 `ontap-nas-flexgroup` PVC로 가져올 수 있습니다.

tridentctl을 사용한 **ONTAP NAS** 예제

다음 예제는 `tridentctl`을 사용하여 관리형 볼륨과 관리되지 않는 볼륨을 가져오는 방법을 보여줍니다.

관리형 볼륨

다음 예시는 `managed_volume`라는 이름의 볼륨을 `ontap_nas`라는 이름의 백엔드에서 가져오는 방법을 보여줍니다:

```
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    |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
```

관리되지 않는 볼륨

`--no-manage` 인수를 사용할 때 Trident는 볼륨 이름을 변경하지 않습니다.

다음 예제는 unmanaged_volume`를 `ontap_nas 백엔드에서 가져옵니다:

```
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   |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
```

PVC 주석을 사용한 **ONTAP NAS** 예제

다음 예제는 PVC 주석을 사용하여 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리형 볼륨

다음 예제는 PVC 주석을 사용하여 RWO 액세스 모드가 설정된 백엔드 `81abcb27-ea63-49bb-b606-0a5315ac5f21`에서 `ontap_volume1`라는 이름의 `1GiB` `ontap-nas` 볼륨을 가져옵니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

관리되지 않는 볼륨

다음 예제는 PVC 주석을 사용하여 RWO 액세스 모드가 설정된 백엔드 `34abcb27-ea63-49bb-b606-0a5315ac5f34`에서 이름이 `ontap-volume2`인 `1Gi` `ontap-nas` 볼륨을 가져옵니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

ONTAP SAN

Trident는 `ontap-san`(iSCSI, NVMe/TCP 및 FC) 및 `ontap-san-economy`드라이버를 사용한 볼륨 가져오기를 지원합니다.

Trident는 단일 LUN을 포함하는 ONTAP SAN FlexVol 볼륨을 가져올 수 있습니다. 이는 각 PVC에 대해 FlexVol 볼륨을 생성하고 FlexVol 볼륨 내에 LUN을 생성하는 `ontap-san` 드라이버와 일치합니다. Trident는 FlexVol 볼륨을 가져와 PVC 정의와 연결합니다. Trident는 여러 LUN을 포함하는 `ontap-san-economy` 볼륨을 가져올 수 있습니다.

다음 예는 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리형 볼륨

관리형 볼륨의 경우 Trident는 FlexVol 볼륨의 이름을 `pvc-<uuid>` 형식으로 변경하고 FlexVol 볼륨 내의 LUN 이름을 `lun0`로 변경합니다.

다음 예제는 `ontap-san-managed` FlexVol 볼륨을 가져옵니다 `ontap_san_default` 백엔드에 있는:

```
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    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

관리되지 않는 볼륨

다음 예제는 `unmanaged_example_volume`를 `ontap_san` 백엔드에서 가져옵니다:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

다음 예시와 같이 Kubernetes 노드 IQN과 IQN을 공유하는 igroup에 매핑된 LUN이 있는 경우 다음 오류가 표시됩니다: `LUN already mapped to initiator(s) in this group`. 볼륨을 가져오려면 이니시에이터를 제거하거나 LUN 매핑을 해제해야 합니다.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

ONTAP SAN-economy 예

다음 예제는 `ontap-san-economy` 백엔드에 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리형 볼륨

관리형 볼륨을 가져오면 Trident가 FlexVol의 소유권을 가져오고 이름을 변경합니다. 동일한 FlexVol에서 여러 LUN을 가져올 때 이 이름 변경을 고려해야 합니다.

다음 예제는 FlexVol `toimport`에서 `lun1`를 가져와 관리되는 볼륨 `vol-managed-saneco`로 가져옵니다:

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

가져오기 `lun1` 후 Trident는 FlexVol의 이름을 변경합니다(예: `trident_lun_pool_xyz`). 동일한 FlexVol에서 추가 LUN을 가져오려면 새 FlexVol 이름을 사용하십시오.

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```

참고	`ontap-san-economy` 백엔드는 한 번에 하나의 LUN을 가져옵니다. 스크립트를 사용하여 여러 가져오기를 자동화할 수 있습니다.

관리되지 않는 볼륨

관리되지 않는 볼륨을 가져올 때 Trident는 FlexVol의 소유권을 가져오지 않습니다. 그러나 FlexVol과 LUN은 Trident 명명 규칙을 따라야 합니다.

FlexVol 명명 형식

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX`는 백엔드 구성의 `storagePrefix` 값입니다. 기본값은 `trident`입니다.
- `RANDOMSTRING`은(는) 선택한 문자열입니다.

LUN 명명 요구 사항

LUN의 이름은 `lun0`이어야 합니다.

예

`storagePrefix`이 `xyz`인 경우 LUN의 전체 경로는 다음과 같습니다.

```
trident_lun_pool_xyz_randomstring/lun0
```

요소

Trident는 `solidfire-san` 드라이버를 사용하여 NetApp Element 소프트웨어 및 NetApp HCI 볼륨 가져오기를 지원합니다.

참고 | Element 드라이버는 중복된 볼륨 이름을 지원합니다. 하지만 Trident는 중복된 볼륨 이름이 있는 경우 오류를 반환합니다. 해결 방법으로 볼륨을 복제하고 고유한 볼륨 이름을 지정한 다음 복제된 볼륨을 가져오십시오.

다음 예제는 백엔드 `element_default`에서 `element-managed` 볼륨을 가져옵니다.

```
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    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Azure NetApp Files

Trident는 `azure-netapp-files` 드라이버를 사용하여 볼륨 가져오기를 지원합니다.

참고 | Azure NetApp Files 볼륨을 가져오려면 볼륨 경로를 사용하여 볼륨을 식별해야 합니다. 볼륨 경로는 볼륨 내보내기 경로에서 `:/` 뒤에 오는 부분입니다. 예를 들어 마운트 경로가 `10.0.0.2:/importvol1`인 경우 볼륨 경로는 `importvol1`입니다.

다음 예제는 백엔드에서 `azure-netapp-files` 볼륨을 가져오며, 볼륨 경로는 `azurenetappfiles_40517`입니다 `importvol1`.

```
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    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud NetApp Volumes

Trident는 `google-cloud-netapp-volumes` 드라이버를 사용하여 볼륨 가져오기를 지원합니다.

다음 예제는 backend `backend-tbc-gcnv1`에서 volume `testvoleasiaeast1`을 가져옵니다.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

다음 예제는 동일한 영역에 두 개의 볼륨이 있는 경우 `google-cloud-netapp-volumes` 볼륨을 가져옵니다.

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

볼륨 이름 및 레이블 사용자 지정

Trident를 사용하면 생성하는 볼륨에 의미 있는 이름과 레이블을 지정할 수 있습니다. 이를 통해 볼륨을 쉽게 식별하고 해당 Kubernetes 리소스(PVC)에 매핑할 수 있습니다. 또한 백엔드 수준에서 사용자 지정 볼륨 이름과 사용자 지정 레이블을 생성하기 위한 템플릿을 정의할 수 있으며, 생성, 가져오기 또는 복제하는 모든 볼륨은 이러한 템플릿을 따릅니다.

시작하기 전에

사용자 지정 가능한 볼륨 이름 및 레이블 지원:

- 볼륨 생성, 가져오기 및 클론 작업.
- `ontap-nas-economy` 드라이버의 경우 Qtree 볼륨의 이름만 이름 템플릿을 준수합니다.
- `ontap-san-economy` 드라이버의 경우 LUN 이름만 이름 템플릿을 준수합니다.

제한 사항

- 사용자 지정 볼륨 이름은 ONTAP 온프레미스 드라이버에서만 호환됩니다.
- 사용자 지정 레이블은 `ontap-san`, `ontap-nas` 및 `ontap-nas-flexgroup` 드라이버에서만 지원됩니다.
- 사용자 지정 볼륨 이름은 기존 볼륨에 적용되지 않습니다.

사용자 지정 가능한 볼륨 이름의 주요 동작

- 이름 템플릿의 구문 오류로 인해 오류가 발생하면 백엔드 생성이 실패합니다. 하지만 템플릿 적용이 실패할 경우 볼륨 이름은 기존 명명 규칙에 따라 지정됩니다.
- 백엔드 구성의 이름 템플릿을 사용하여 볼륨 이름을 지정할 때는 스토리지 접두사를 적용할 수 없습니다. 원하는 접두사 값을 템플릿에 직접 추가할 수 있습니다.

이름 템플릿 및 레이블이 포함된 백엔드 구성 예

사용자 지정 이름 템플릿은 루트 및/또는 풀 수준에서 정의할 수 있습니다.

루트 레벨 예

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

풀 수준 예

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

이름 템플릿 예

예 **1**:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

예 **2**:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

고려해야 할 사항

1. 볼륨 가져오기의 경우, 기존 볼륨에 특정 형식의 레이블이 있는 경우에만 레이블이 업데이트됩니다. 예를 들면 다음과 같습니다: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. 관리형 볼륨 가져오기의 경우 볼륨 이름은 백엔드 정의의 루트 수준에 정의된 이름 템플릿을 따릅니다.
3. Trident는 스토리지 접두사와 함께 슬라이스 연산자를 사용하는 것을 지원하지 않습니다.
4. 템플릿을 사용했을 때 고유한 볼륨 이름이 생성되지 않으면 Trident가 임의의 문자를 추가하여 고유한 볼륨 이름을 생성합니다.
5. NAS 이코노미 볼륨의 사용자 지정 이름이 64자를 초과하면 Trident는 기존 명명 규칙에 따라 볼륨 이름을 지정합니다. 다른 모든 ONTAP 드라이버의 경우 볼륨 이름이 이름 제한을 초과하면 볼륨 생성 프로세스가 실패합니다.

네임스페이스 간에 **NFS** 볼륨 공유

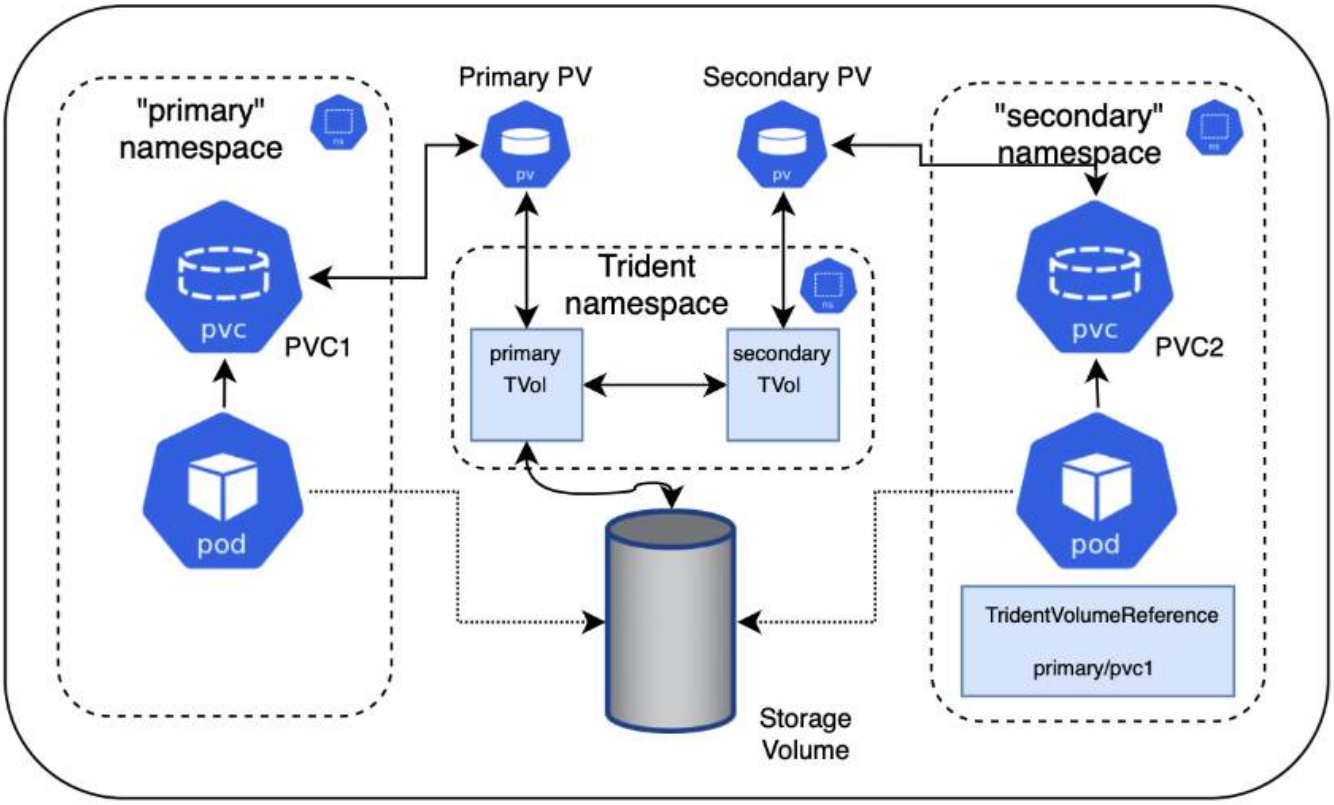
Trident를 사용하면 기본 네임스페이스에 볼륨을 생성하고 하나 이상의 보조 네임스페이스에서 공유할 수 있습니다.

기능

TridentVolumeReference CR을 사용하면 하나 이상의 Kubernetes 네임스페이스에서 ReadWriteMany(RWX) NFS 볼륨을 안전하게 공유할 수 있습니다. 이 Kubernetes 네이티브 솔루션은 다음과 같은 이점을 제공합니다.

- 보안을 보장하기 위한 여러 수준의 액세스 제어
- 모든 Trident NFS 볼륨 드라이버와 호환됩니다.
- tridentctl 또는 기타 비네이티브 Kubernetes 기능에 의존하지 않음

이 다이어그램은 두 Kubernetes 네임스페이스에서 NFS 볼륨 공유를 보여줍니다.



빠른 시작

몇 단계만 거치면 NFS 볼륨 공유를 설정할 수 있습니다.

1 소스 **PVC**를 구성하여 볼륨을 공유합니다

소스 네임스페이스 소유자는 소스 PVC의 데이터에 액세스할 수 있는 권한을 부여합니다.

2 대상 네임스페이스에 **CR**을 생성할 수 있는 권한을 부여하십시오

클러스터 관리자는 대상 네임스페이스 소유자에게 TridentVolumeReference CR을 생성할 수 있는 권한을 부여합니다.

3 대상 네임스페이스에 **TridentVolumeReference**를 생성합니다

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 TridentVolumeReference CR을 생성합니다.

4 대상 네임스페이스에 하위 **PVC**를 생성합니다

대상 네임스페이스의 소유자는 소스 PVC의 데이터 소스를 사용하기 위해 하위 PVC를 생성합니다.

소스 및 대상 네임스페이스를 구성합니다

보안을 보장하기 위해 네임스페이스 간 공유는 소스 네임스페이스 소유자, 클러스터 관리자 및 대상 네임스페이스 소유자의 협업과 조치가 필요합니다. 각 단계에서 사용자 역할이 지정됩니다.

단계

1. 소스 네임스페이스 소유자: 소스 네임스페이스에서 PVC (`pvc1`)를 생성하여 대상 네임스페이스와 공유할 수 있는 권한을 부여합니다 (`namespace2`). 이때 `shareToNamespace` 주석을 사용합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident는 PV와 해당 백엔드 NFS 스토리지 볼륨을 생성합니다.

참고

- 쉼표로 구분된 목록을 사용하여 여러 네임스페이스에서 PVC를 공유할 수 있습니다. 예를 들어 `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
- `*`를 사용하여 모든 네임스페이스에 공유할 수 있습니다. 예를 들어 `trident.netapp.io/shareToNamespace: *`
- 언제든지 PVC를 업데이트하여 `shareToNamespace` 주석을 포함시킬 수 있습니다.

2. 클러스터 관리자: 대상 네임스페이스 소유자가 대상 네임스페이스에 TridentVolumeReference CR을 생성할 수 있도록 적절한 RBAC가 설정되어 있는지 확인하십시오.
3. 대상 네임스페이스 소유자: 대상 네임스페이스에 소스 네임스페이스를 참조하는 TridentVolumeReference CR을 생성합니다 `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. 대상 네임스페이스 소유자: 대상 네임스페이스 (`namespace2`)에서 PVC (`pvc2`)를 생성하고, 소스 PVC를

지정하기 위해 `shareFromPVC` 주석을 사용합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

참고 | 대상 PVC의 크기는 소스 PVC보다 작거나 같아야 합니다.

결과

Trident는 대상 PVC의 `shareFromPVC` 어노테이션을 읽고 소스 PV를 가리키고 소스 PV 스토리지 리소스를 공유하는 자체 스토리지 리소스가 없는 하위 볼륨으로 대상 PV를 생성합니다. 대상 PVC와 PV는 정상적으로 바인딩된 것처럼 보입니다.

공유 볼륨 삭제

여러 네임스페이스에서 공유되는 볼륨을 삭제할 수 있습니다. Trident는 소스 네임스페이스에서 해당 볼륨에 대한 액세스를 제거하고 볼륨을 공유하는 다른 네임스페이스에 대한 액세스는 유지합니다. 볼륨을 참조하는 모든 네임스페이스가 제거되면 Trident는 해당 볼륨을 삭제합니다.

`tridentctl get`를 사용하여 하위 볼륨을 쿼리합니다

이 [`tridentctl` 유틸리티를 사용하면 `get` 명령을 실행하여 하위 볼륨을 가져올 수 있습니다. 자세한 내용은 `tridentctl` 명령 및 옵션을 참조하십시오.

```
Usage:
  tridentctl get [option]
```

플래그:

- `-h, --help`: 볼륨에 대한 도움말입니다.
- `--parentOfSubordinate string`: 쿼리를 하위 소스 볼륨으로 제한합니다.
- `--subordinateOf string`: 볼륨의 하위 항목으로 쿼리를 제한합니다.

제한 사항

- Trident는 대상 네임스페이스가 공유 볼륨에 쓰기 작업을 하는 것을 막을 수 없습니다. 공유 볼륨 데이터가 덮어쓰이는 것을 방지하려면 파일 잠금 또는 다른 프로세스를 사용해야 합니다.
- `shareToNamespace` 또는 `shareFromNamespace` 어노테이션을 제거하거나 `TridentVolumeReference` CR을 삭제해도 소스 PVC에 대한 액세스 권한을 취소할 수 없습니다. 액세스 권한을 취소하려면 하위 PVC를 삭제해야 합니다.
- 하위 볼륨에서는 스냅샷, 클론 및 미러링이 불가능합니다.

자세한 내용은

교차 네임스페이스 볼륨 액세스에 대한 자세한 내용은 다음을 참조하십시오.

- "NetAppTV"에서 데모를 시청하십시오.

네임스페이스 간 볼륨 복제

Trident를 사용하면 동일한 Kubernetes 클러스터 내의 다른 네임스페이스에 있는 기존 볼륨 또는 볼륨 스냅샷을 사용하여 새 볼륨을 생성할 수 있습니다.

필수 구성 요소

볼륨을 복제하기 전에 소스 및 대상 백엔드가 동일한 유형이고 동일한 스토리지 클래스를 가지고 있는지 확인하십시오.

참고	네임스페이스 간 복제는 `ontap-san` 및 `ontap-nas` 스토리지 드라이버에 한해서만 지원됩니다. 읽기 전용 복제는 지원되지 않습니다.

빠른 시작

몇 단계만 거치면 볼륨 클로닝을 설정할 수 있습니다.


소스 **PVC**를 구성하여 볼륨 복제

소스 네임스페이스 소유자는 소스 PVC의 데이터에 액세스할 수 있는 권한을 부여합니다.


대상 네임스페이스에 **CR**을 생성할 수 있는 권한을 부여하십시오

클러스터 관리자는 대상 네임스페이스 소유자에게 TridentVolumeReference CR을 생성할 수 있는 권한을 부여합니다.


대상 네임스페이스에 **TridentVolumeReference**를 생성합니다

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 TridentVolumeReference CR을 생성합니다.


대상 네임스페이스에 클론 **PVC**를 생성합니다

대상 네임스페이스의 소유자는 소스 네임스페이스의 PVC를 복제하기 위해 PVC를 생성합니다.

소스 및 대상 네임스페이스를 구성합니다

보안을 보장하기 위해 네임스페이스 간 볼륨 복제에는 소스 네임스페이스 소유자, 클러스터 관리자 및 대상 네임스페이스 소유자의 협업과 조치가 필요합니다. 각 단계에서 사용자 역할이 지정됩니다.

단계

1. 소스 네임스페이스 소유자: 소스 네임스페이스(`namespace1`)에서 PVC(`pvc1`)를 생성하고, `cloneToNamespace`주석을 사용하여 대상 네임스페이스(`namespace2`)와 공유할 수 있는 권한을 부여합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident가 PV와 해당 백엔드 스토리지 볼륨을 생성합니다.

참고

- 쉼표로 구분된 목록을 사용하여 여러 네임스페이스에서 PVC를 공유할 수 있습니다. 예를 들어 `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
- `*`를 사용하여 모든 네임스페이스에 공유할 수 있습니다. 예를 들어 `trident.netapp.io/cloneToNamespace: *`
- 언제든지 PVC를 업데이트하여 `cloneToNamespace` 주석을 포함시킬 수 있습니다.

2. 클러스터 관리자: 대상 네임스페이스 소유자가 대상 네임스페이스에 TridentVolumeReference CR을 생성할 수 있도록 적절한 RBAC가 설정되어 있는지 확인하십시오(`namespace2`.
3. 대상 네임스페이스 소유자: 대상 네임스페이스에 소스 네임스페이스를 참조하는 TridentVolumeReference CR을 생성합니다 `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. 대상 네임스페이스 소유자: 대상 네임스페이스 (`namespace2`)에서 PVC (`pvc2`)를 생성하고, `cloneFromPVC` 또는 `cloneFromSnapshot` 및 `cloneFromNamespace` 주석을 사용하여 소스 PVC를 지정합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

제한 사항

- ontap-nas-economy 드라이버를 사용하여 프로비저닝된 PVC의 경우 읽기 전용 클론은 지원되지 않습니다.

SnapMirror를 사용하여 볼륨 복제

Trident는 재해 복구를 위한 데이터 복제를 위해 한 클러스터의 소스 볼륨과 피어링된 클러스터의 타겟 볼륨 간의 미러 관계를 지원합니다. Trident Mirror Relationship(TMR)이라는 네임스페이스 Custom Resource Definition(CRD)을 사용하여 다음 작업을 수행할 수 있습니다.

- 볼륨(PVC) 간의 미러 관계 생성
- 볼륨 간의 미러 관계를 제거합니다
- 미러 관계를 중단합니다
- 재해 조건(페일오버) 중에 보조 볼륨을 승격합니다

- 클러스터 간 애플리케이션의 무손실 전환 수행(계획된 페일오버 또는 마이그레이션 중)

복제 사전 요구 사항

시작하기 전에 다음 사전 요구 사항이 충족되었는지 확인하십시오.

ONTAP 클러스터

- **Trident**: ONTAP를 백엔드로 사용하는 소스 및 타겟 Kubernetes 클러스터 모두에 Trident 버전 22.10 이상이 있어야 합니다.
- 라이선스: 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이선스는 소스 및 타겟 ONTAP 클러스터 모두에서 활성화되어야 합니다. 자세한 내용은 "SnapMirror ONTAP 라이센싱 개요"을 참조하십시오.

  ONTAP 9.10.1부터 모든 라이센스는 여러 기능을 활성화하는 단일 파일인 NetApp 라이센스 파일(NLF)로 제공됩니다. 자세한 내용은 "ONTAP One에 포함된 라이센스"을 참조하십시오.

참고	SnapMirror 비동기 보호 기능만 지원됩니다.

피어링

- 클러스터 및 **SVM**: ONTAP 스토리지 백엔드는 피어링되어야 합니다. 자세한 내용은 "클러스터 및 SVM 피어링 개요"을 참조하십시오.

중요함	두 ONTAP 클러스터 간의 복제 관계에 사용되는 SVM 이름이 고유한지 확인하십시오.

- **Trident** 및 **SVM**: 피어링된 원격 SVM은 타겟 클러스터의 Trident에서 사용할 수 있어야 합니다.

지원되는 드라이버

NetApp Trident는 다음 드라이버가 지원하는 스토리지 클래스를 사용하여 NetApp SnapMirror 기술로 볼륨 복제를 지원합니다. **`ontap-nas`: NFS** `ontap-san`: iSCSI **`ontap-san`: FC** `ontap-san`: NVMe/TCP(최소 ONTAP 버전 9.15.1 필요)

참고	SnapMirror를 사용한 볼륨 복제는 ASA r2 시스템에서 지원되지 않습니다. ASA r2 시스템에 대한 자세한 내용은 "ASA r2 스토리지 시스템에 대해 알아보세요"를 참조하십시오.

미러링된 **PVC** 생성

다음 단계를 따르고 CRD 예제를 사용하여 운영 볼륨과 2차 볼륨 간의 미러 관계를 생성하십시오.

단계

1. 기본 Kubernetes 클러스터에서 다음 단계를 수행하십시오.
   a. `trident.netapp.io/replication: true` 매개 변수를 사용하여 StorageClass 객체를 생성합니다.

예

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

b. 이전에 생성한 StorageClass로 PVC를 생성합니다.

예

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

c. 로컬 정보를 사용하여 MirrorRelationship CR을 생성합니다.

예

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

Trident는 볼륨에 대한 내부 정보와 볼륨의 현재 데이터 보호(DP) 상태를 가져온 다음 MirrorRelationship의 상태 필드를 채웁니다.

d. TridentMirrorRelationship CR을 가져와 PVC의 내부 이름과 SVM을 얻으십시오.

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

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

2. 보조 Kubernetes 클러스터에서 다음 단계를 수행하십시오.
   a. trident.netapp.io/replication: true 매개변수를 사용하여 StorageClass를 생성합니다.

      예

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

   b. 대상 및 소스 정보를 사용하여 MirrorRelationship CR을 생성합니다.

예

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

Trident는 구성된 관계 정책 이름(또는 ONTAP의 기본값)으로 SnapMirror 관계를 생성하고 초기화합니다.

c. 이전에 생성한 StorageClass를 사용하여 보조(SnapMirror 타겟) 역할을 하는 PVC를 생성합니다.

예

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

Trident는 TridentMirrorRelationship CRD를 확인하고, 관계가 존재하지 않으면 볼륨 생성을 실패합니다. 관계가 존재하는 경우, Trident는 새 FlexVol 볼륨이 MirrorRelationship에 정의된 원격 SVM과 피어링된 SVM에 배치되도록 합니다.

볼륨 복제 상태

Trident 미러 관계(TMR)는 PVC 간의 복제 관계의 한쪽 끝을 나타내는 CRD입니다. 대상 TMR에는 원하는 상태를 Trident에 알려주는 상태가 있습니다. 대상 TMR은 다음과 같은 상태를 가집니다.

- **Established**: 로컬 PVC는 미러 관계의 타겟 볼륨이며, 이는 새로운 관계입니다.
- 승격됨: 로컬 PVC는 ReadWrite이며 마운트 가능하고, 현재 미러 관계가 적용되지 않습니다.
- 재설정됨: 로컬 PVC는 미러 관계의 타겟 볼륨이며 이전에도 해당 미러 관계에 있었습니다.

- 타겟 볼륨이 소스 볼륨과 관계를 맺은 적이 있는 경우 타겟 볼륨 콘텐츠를 덮어쓰므로 재설정된 상태를 사용해야 합니다.
- 볼륨이 이전에 소스와 관계를 맺지 않았던 경우 재설정된 상태가 실패합니다.

예기치 않은 페일오버 시 보조 **PVC** 승격

보조 Kubernetes 클러스터에서 다음 단계를 수행하십시오.

- TridentMirrorRelationship의 *spec.state* 필드를 `promoted`으로 업데이트합니다.

계획된 페일오버 중에 보조 **PVC**를 승격합니다

계획된 페일오버(마이그레이션) 중에 다음 단계를 수행하여 보조 PVC를 승격시키십시오.

단계

1. 기본 Kubernetes 클러스터에서 PVC의 스냅샷을 생성하고 스냅샷이 생성될 때까지 기다립니다.
2. 기본 Kubernetes 클러스터에서 SnapshotInfo CR을 생성하여 내부 정보를 얻습니다.

   예

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

3. 보조 Kubernetes 클러스터에서 *TridentMirrorRelationship* CR의 *spec.state* 필드를 _promoted_로 업데이트하고 _spec.promotedSnapshotHandle_을 스냅샷의 internalName으로 업데이트합니다.
4. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 상태(status.state 필드)가 promoted로 승격되었는지 확인합니다.

페일오버 후 미러 관계 복원

미러 관계를 복원하기 전에 새 운영 환경으로 지정할 측을 선택합니다.

단계

1. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 *spec.remoteVolumeHandle* 필드 값이 업데이트되었는지 확인하십시오.
2. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 *spec.mirror* 필드를 `reestablished`로 업데이트합니다.

추가 작업

Trident는 운영 볼륨과 2차 볼륨에서 다음 작업을 지원합니다.

기본 **PVC**를 새 보조 **PVC**로 복제합니다

기본 PVC와 보조 PVC가 이미 있는지 확인하십시오.

단계

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

미러링된 운영 또는 보조 **PVC**의 크기 조정

PVC는 정상적으로 크기를 조정할 수 있으며, 데이터 양이 현재 크기를 초과하면 ONTAP은 대상 FlexVol을 자동으로 확장합니다.

PVC에서 복제를 제거합니다

복제를 제거하려면 현재 보조 볼륨에서 다음 작업 중 하나를 수행하십시오.

- 보조 PVC에서 MirrorRelationship을 삭제합니다. 이렇게 하면 복제 관계가 끊어집니다.
- 또는 spec.state 필드를 _promoted_로 업데이트하십시오.

PVC 삭제**(**이전에 미러링됨**)**

Trident는 복제된 PVC를 확인하고 볼륨 삭제를 시도하기 전에 복제 관계를 해제합니다.

TMR 삭제

미러링 관계의 한쪽에서 TMR을 삭제하면 Trident가 삭제를 완료하기 전에 나머지 TMR이 *promoted* 상태로 전환됩니다. 삭제 대상으로 선택된 TMR이 이미 *promoted* 상태인 경우 기존 미러 관계가 없으므로 해당 TMR이 제거되고 Trident는 로컬 PVC를 *ReadWrite_*로 승격시킵니다. 이 삭제는 *ONTAP*의 로컬 볼륨에 대한 *SnapMirror* 메타데이터를 해제합니다. 향후 이 볼륨을 미러 관계에 사용하려면 새 미러 관계를 생성할 때 *_established* 볼륨 복제 상태의 새 TMR을 사용해야 합니다.

ONTAP이 온라인 상태일 때 미러 관계를 업데이트합니다

미러 관계는 설정 후 언제든지 업데이트할 수 있습니다. `state: promoted` 또는 `state: reestablished` 필드를 사용하여 관계를 업데이트할 수 있습니다. 대상 볼륨을 일반 ReadWrite 볼륨으로 승격할 때 _promotedSnapshotHandle_을 사용하여 현재 볼륨을 복원할 특정 스냅샷을 지정할 수 있습니다.

ONTAP이 오프라인일 때 미러 관계를 업데이트합니다

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

예

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

`status.state` TridentActionMirrorUpdate CRD의 상태를 나타냅니다. *Succeeded*, *In Progress*, 또는 *Failed* 값을 가질 수 있습니다.

CSI 토폴로지 사용

Trident은 "CSI 토폴로지 기능"을 사용하여 Kubernetes 클러스터에 있는 노드에 선택적으로 볼륨을 생성하고 연결할 수 있습니다.

개요

CSI 토폴로지 기능을 사용하면 리전 및 가용 영역을 기반으로 볼륨 액세스를 노드 하위 집합으로 제한할 수 있습니다. 오늘날 클라우드 제공업체는 Kubernetes 관리자가 영역 기반 노드를 생성할 수 있도록 지원합니다. 노드는 리전 내의 서로 다른 가용 영역에 위치하거나 여러 리전에 걸쳐 위치할 수 있습니다. 다중 영역 아키텍처에서 워크로드에 대한 볼륨 프로비저닝을 용이하게 하기 위해 Trident는 CSI 토폴로지를 사용합니다.

> 팁 CSI 토폴로지 기능에 대해 자세히 알아보십시오 "여기".

Kubernetes는 두 가지 고유한 볼륨 바인딩 모드를 제공합니다.

- `VolumeBindingMode`을 `Immediate`로 설정하면 Trident는 토폴로지를 고려하지 않고 볼륨을 생성합니다. 볼륨 바인딩 및 동적 프로비저닝은 PVC 생성 시 처리됩니다. 이는 기본 `VolumeBindingMode`이며 토폴로지 제약 조건을 적용하지 않는 클러스터에 적합합니다. 영구 볼륨은 요청하는 Pod의 스케줄링 요구 사항에 관계없이 생성됩니다.
- `VolumeBindingMode`을 `WaitForFirstConsumer`로 설정하면 PVC에 대한 영구 볼륨의 생성 및 바인딩이 해당 PVC를 사용하는 파드가 스케줄링되고 생성될 때까지 지연됩니다. 이렇게 하면 토폴로지 요구 사항에 따라 적용되는 스케줄링 제약 조건을 충족하도록 볼륨이 생성됩니다.

> 참고 `WaitForFirstConsumer` 바인딩 모드는 토폴로지 레이블을 필요로 하지 않습니다. CSI 토폴로지 기능과 별개로 사용할 수 있습니다.

필요한 것

CSI 토폴로지를 사용하려면 다음이 필요합니다.

- "지원되는 Kubernetes 버전"를 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 인식을 도입하는 레이블이 있어야 합니다(`topology.kubernetes.io/region` 및 `topology.kubernetes.io/zone`). 이러한 레이블은 Trident가 토폴로지를 인식하도록 하려면 Trident를 설치하기 전에 클러스터의 노드에 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

Trident 스토리지 백엔드는 가용성 영역을 기반으로 볼륨을 선택적으로 프로비저닝하도록 설계할 수 있습니다. 각 백엔드는 지원되는 영역 및 리전 목록을 나타내는 선택적 `supportedTopologies` 블록을 포함할 수 있습니다. 이러한 백엔드를 사용하는 StorageClasses의 경우, 지원되는 리전/영역에 스케줄링된 애플리케이션에서 요청한 경우에만 볼륨이 생성됩니다.

다음은 백엔드 정의의 예입니다.

YAML

```yaml
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
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
```

JSON

```json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "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"
    }
  ]
}
```

참고

`supportedTopologies`은 백엔드별 리전 및 영역 목록을 제공하는 데 사용됩니다. 이러한 리전 및 영역은 StorageClass에 제공할 수 있는 허용 가능한 값 목록을 나타냅니다. 백엔드에 제공된 리전 및 영역의 하위 집합을 포함하는 StorageClasses의 경우 Trident는 해당 백엔드에 볼륨을 생성합니다.

`supportedTopologies`를 스토리지 풀별로 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: password
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
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

이 예에서 `region` 및 `zone` 레이블은 스토리지 풀의 위치를 나타냅니다. `topology.kubernetes.io/region` 및 `topology.kubernetes.io/zone`는 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. 이를 통해 PVC 요청에 대한 후보로 사용되는 스토리지 풀과 Trident에서 프로비저닝한 볼륨을 사용할 수 있는 노드의 하위 집합이 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
```

위에 제공된 StorageClass 정의에서 `volumeBindingMode`는 `WaitForFirstConsumer`로 설정되어 있습니다. 이 `StorageClass`로 요청된 `PVC`는 `Pod`에서 참조될 때까지 처리되지 않습니다. 그리고 `allowedTopologies`는 사용할 영역과 지역을 제공합니다. `netapp-san-us-east1` StorageClass는 위에 정의된 `san-backend-us-east1` 백엔드에 PVC를 생성합니다.

3단계: **PVC** 생성 및 사용

StorageClass를 생성하고 백엔드에 매핑했으면 이제 PVC를 생성할 수 있습니다.

아래 예시를 참조하세요 `spec`:

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

이 매니페스트를 사용하여 PVC를 생성하면 다음과 같은 결과가 나타납니다.

```
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
```

Trident가 볼륨을 생성하고 PVC에 바인딩하려면 포드에서 PVC를 사용하십시오. 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 Kubernetes에게 `us-east1` 지역에 있는 노드에 Pod를 예약하도록 지시하며, `us-east1-a` 또는 `us-east1-b` 영역에 있는 모든 노드 중에서 선택합니다.

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
```

백엔드를 업데이트하여 다음을 포함하도록 합니다 `supportedTopologies`

기존 백엔드를 업데이트하여 `supportedTopologies`목록을 포함할 수 있습니다 `tridentctl backend update`. 이는 이미 프로비저닝된 볼륨에는 영향을 미치지 않으며 이후 PVC에만 사용됩니다.

자세한 정보 찾기

- "컨테이너의 리소스 관리"
- "nodeSelector"
- "친화성 및 반친화성"
- "Taints 및 Tolerations"

스냅샷 작업

Kubernetes 볼륨 스냅샷의 영구 볼륨(PVs)은 볼륨의 시점 복사본을 활성화합니다. Trident를 사용하여 생성된 볼륨의 스냅샷을 생성하고, Trident 외부에서 생성된 스냅샷을 가져오고, 기존 스냅샷에서 새 볼륨을 생성하고, 스냅샷에서 볼륨 데이터를 복구할 수 있습니다.

개요

볼륨 스냅샷은 `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` 및 `google-cloud-netapp-volumes` 드라이버에서 지원됩니다.

시작하기 전에

스냅샷으로 작업하려면 외부 스냅샷 컨트롤러와 CRD(사용자 정의 리소스 정의)가 있어야 합니다. 이는 Kubernetes 오케스트레이터의 책임입니다(예: Kubeadm, GKE, OpenShift).

Kubernetes 배포판에 스냅샷 컨트롤러와 CRD가 포함되어 있지 않은 경우 볼륨 스냅샷 컨트롤러를 배포합니다을 참조하십시오.

참고	GKE 환경에서 온디맨드 볼륨 스냅샷을 생성하는 경우 스냅샷 컨트롤러를 생성하지 마십시오. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

볼륨 스냅샷을 생성합니다

단계

1. `VolumeSnapshotClass`을(를) 생성합니다. 자세한 내용은 "VolumeSnapshotClass"을(를) 참조하십시오.
   - `driver`는 Trident CSI 드라이버를 가리킵니다.
   - `deletionPolicy`은 `Delete` 또는 `Retain`일 수 있습니다. `Retain`로 설정하면 `VolumeSnapshot` 객체가 삭제되더라도 스토리지 클러스터의 기본 물리적 스냅샷이 유지됩니다.

   예

   ```
   cat snap-sc.yaml
   ```

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshotClass
   metadata:
     name: csi-snapclass
   driver: csi.trident.netapp.io
   deletionPolicy: Delete
   ```

2. 기존 PVC의 스냅샷을 생성합니다.

   예
   - 이 예제는 기존 PVC의 스냅샷을 생성합니다.

   ```
   cat snap.yaml
   ```

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshot
   metadata:
     name: pvc1-snap
   spec:
     volumeSnapshotClassName: csi-snapclass
     source:
       persistentVolumeClaimName: pvc1
   ```

   - 이 예제는 `pvc1`라는 이름의 PVC에 대한 볼륨 스냅샷 객체를 생성하고 스냅샷 이름을 `pvc1-snap`로 설정합니다. VolumeSnapshot은 PVC와 유사하며 실제 스냅샷을 나타내는 `VolumeSnapshotContent` 객체와 연결됩니다.

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

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

- `VolumeSnapshotContent` 객체를 `pvc1-snap` VolumeSnapshot에 대해 설명하여 식별할 수 있습니다. `Snapshot Content Name`는 이 스냅샷을 제공하는 `VolumeSnapshotContent` 객체를 식별합니다. `Ready To Use` 매개변수는 스냅샷을 사용하여 새 PVC를 생성할 수 있음을 나타냅니다.

```
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
...
```

볼륨 스냅샷에서 **PVC**를 생성합니다

`dataSource`을 사용하여 `VolumeSnapshot`이라는 이름의 `<pvc-name>`를 데이터 소스로 사용하는 `PVC`를 생성할 수 있습니다. `PVC`가 생성되면 `Pod`에 연결하여 다른 `PVC`와 마찬가지로 사용할 수 있습니다.

경고 PVC는 소스 볼륨과 동일한 백엔드에서 생성됩니다. 다음을 참조하십시오. "KB: Trident PVC 스냅샷에서 PVC를 생성하는 작업은 다른 백엔드에서 수행할 수 없습니다".

다음 예제는 `pvc1-snap`를 데이터 소스로 사용하여 PVC를 생성합니다.

```
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
```

볼륨 스냅샷 가져오기

Trident는 "Kubernetes 사전 프로비저닝된 스냅샷 프로세스"을 지원하여 클러스터 관리자가 `VolumeSnapshotContent` 객체를 생성하고 Trident 외부에서 생성된 스냅샷을 가져올 수 있도록 합니다.

시작하기 전에

Trident가 스냅샷의 상위 볼륨을 생성하거나 가져와야 합니다.

단계

1. 클러스터 관리자: 백엔드 스냅샷을 참조하는 `VolumeSnapshotContent` 객체를 생성합니다. 이렇게 하면 Trident에서 스냅샷 워크플로우가 시작됩니다.
   - 백엔드 스냅샷의 이름을 `annotations`에서 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`로 지정하세요.
   - `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>`를 `snapshotHandle`에 지정하십시오. 이것이 `ListSnapshots` 호출에서 외부 스냅샷터가 Trident에 제공하는 유일한 정보입니다.

     참고 | `<volumeSnapshotContentName>`는 CR 명명 제약 조건으로 인해 백엔드 스냅샷 이름과 항상 일치하는 것은 아닙니다.

     예

     다음 예제는 백엔드 스냅샷 `snap-01`을 참조하는 `VolumeSnapshotContent` 오브젝트를 생성합니다.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. 클러스터 관리자: `VolumeSnapshot` 객체를 참조하는 CR을 생성합니다. `VolumeSnapshotContent` 이것은 지정된 네임스페이스에서 `VolumeSnapshot` 사용 권한을 요청합니다.

   예

   다음 예제는 `VolumeSnapshot`라는 이름의 CR을 생성하며, `import-snap`를 참조하고, `VolumeSnapshotContent`라는 이름의 `import-snap-content`를 참조합니다.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content
```

3. 내부 처리(별도 조치 필요 없음): 외부 스냅샷 생성기가 새로 생성된 `VolumeSnapshotContent`을 인식하고 `ListSnapshots` 호출을 실행합니다. Trident가 `TridentSnapshot`을 생성합니다.
   - 외부 snapshotter는 `VolumeSnapshotContent`를 `readyToUse`로, `VolumeSnapshot`를 `true`로 설정합니다.
   - Trident가 반환됩니다 `readyToUse=true`.
4. 모든 사용자: 새 `PersistentVolumeClaim`를 참조하기 위해 `VolumeSnapshot`를 생성합니다. 여기서 `spec.dataSource` (또는 `spec.dataSourceRef`) 이름은 `VolumeSnapshot` 이름입니다.

   예

다음 예제는 `VolumeSnapshot`라는 이름의 `import-snap`을(를) 참조하는 PVC를 생성합니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

스냅샷을 사용하여 볼륨 데이터 복구

스냅샷 디렉터리는 기본적으로 숨겨져 있어 `ontap-nas` 및 `ontap-nas-economy` 드라이버를 사용하여 프로비저닝된 볼륨의 최대 호환성을 용이하게 합니다. 스냅샷에서 직접 데이터를 복구하려면 `.snapshot` 디렉터리를 활성화하십시오.

볼륨 스냅샷 복원 ONTAP CLI를 사용하여 볼륨을 이전 스냅샷에 기록된 상태로 복원합니다.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

참고 | 스냅샷 복사본을 복원하면 기존 볼륨 구성이 덮어쓰여집니다. 스냅샷 복사본이 생성된 후 볼륨 데이터에 대한 변경 사항은 손실됩니다.

스냅샷에서 볼륨 제자리 복원

Trident는 `TridentActionSnapshotRestore` (TASR) CR을 사용하여 스냅샷에서 신속하게 제자리 볼륨 복원을 제공합니다. 이 CR은 명령형 Kubernetes 작업으로 작동하며 작업 완료 후에는 지속되지 않습니다.

Trident는 `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` 및 `solidfire-san` 드라이버에서 스냅샷 복원을 지원합니다.

시작하기 전에

바인딩된 PVC와 사용 가능한 볼륨 스냅샷이 있어야 합니다.

- PVC 상태가 바인딩되었는지 확인합니다.

```
kubectl get pvc
```

- 볼륨 스냅샷을 사용할 준비가 되었는지 확인합니다.

```
kubectl get vs
```

단계

1. TASR CR을 생성합니다. 이 예제에서는 PVC `pvc1` 및 볼륨 스냅샷 `pvc1-snapshot`용 CR을 생성합니다.

   참고 | TASR CR은 PVC 및 VS가 존재하는 네임스페이스에 있어야 합니다.

```
cat tasr-pvc1-snapshot.yaml
```

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

2. 스냅샷에서 복원하려면 CR을 적용하세요. 이 예에서는 스냅샷에서 복원합니다 `pvc1`.

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

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

결과

Trident 스냅샷에서 데이터를 복원합니다. 스냅샷 복원 상태는 다음과 같이 확인할 수 있습니다.

```
kubectl get tasr -o yaml
```

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

참고

- 대부분의 경우 Trident는 실패 시 자동으로 작업을 재시도하지 않습니다. 작업을 다시 수행해야 합니다.
- 관리자 액세스 권한이 없는 Kubernetes 사용자는 관리자로부터 애플리케이션 네임스페이스에 TASR CR을 생성할 수 있는 권한을 부여받아야 할 수도 있습니다.

연결된 스냅샷이 있는 **PV** 삭제

스냅샷이 연결된 영구 볼륨을 삭제하면 해당 Trident 볼륨의 상태가 "삭제 중"으로 업데이트됩니다. Trident 볼륨을 삭제하려면 볼륨 스냅샷을 제거하십시오.

볼륨 스냅샷 컨트롤러를 배포합니다

Kubernetes 배포판에 스냅샷 컨트롤러와 CRD가 포함되어 있지 않은 경우 다음과 같이 배포할 수 있습니다.

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
cat snapshot-setup.sh
```

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

2. 스냅샷 컨트롤러를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

참고	필요한 경우 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`를 열고 `namespace`를 네임스페이스로 업데이트하십시오.

관련 링크

- "볼륨 스냅샷"
- "VolumeSnapshotClass"

볼륨 그룹 스냅샷 작업

퍼시스턴트 볼륨(PV)의 Kubernetes 볼륨 그룹 스냅샷 NetApp Trident는 여러 볼륨의 스냅샷(볼륨 스냅샷 그룹)을 생성하는 기능을 제공합니다. 이 볼륨 그룹 스냅샷은 동일한 시점에 생성된 여러 볼륨의 복사본을 나타냅니다.

참고	VolumeGroupSnapshot은 베타 API를 사용하는 Kubernetes의 베타 기능입니다. VolumeGroupSnapshot에 필요한 최소 버전은 Kubernetes 1.32입니다.

볼륨 그룹 스냅샷 만들기

볼륨 그룹 스냅샷은 다음 스토리지 드라이버에서 지원됩니다.

- `ontap-san` driver - iSCSI 및 FC 프로토콜에만 해당되며 NVMe/TCP 프로토콜에는 해당되지 않습니다.
- `ontap-san-economy` - iSCSI 프로토콜에만 해당합니다.
- `ontap-nas`

참고	볼륨 그룹 스냅샷은 NetApp ASA r2 또는 AFX 스토리지 시스템에서는 지원되지 않습니다.

시작하기 전에

- Kubernetes 버전이 K8s 1.32 이상인지 확인합니다.
- 스냅샷으로 작업하려면 외부 스냅샷 컨트롤러와 CRD(사용자 정의 리소스 정의)가 있어야 합니다. 이는 Kubernetes 오케스트레이터의 책임입니다(예: Kubeadm, GKE, OpenShift).

  Kubernetes 배포에 외부 스냅샷 컨트롤러와 CRD가 포함되어 있지 않은 경우 볼륨 스냅샷 컨트롤러를 배포합니다을 참조하십시오.

참고	GKE 환경에서 온디맨드 볼륨 그룹 스냅샷을 생성하는 경우 스냅샷 컨트롤러를 생성하지 마십시오. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

- 스냅샷 컨트롤러 YAML에서 `CSIVolumeGroupSnapshot` 기능 게이트를 'true'로 설정하여 볼륨 그룹 스냅샷이 활성화되도록 합니다.
- 볼륨 그룹 스냅샷을 생성하기 전에 필요한 볼륨 그룹 스냅샷 클래스를 생성합니다.
- 모든 PVC/볼륨이 동일한 SVM에 있는지 확인하여 VolumeGroupSnapshot을 생성할 수 있도록 합니다.

단계

- VolumeGroupSnapshot을 생성하기 전에 VolumeGroupSnapshotClass를 생성하십시오. 자세한 내용은 "VolumeGroupSnapshotClass"을 참조하십시오.

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- 기존 스토리지 클래스를 사용하여 필수 레이블이 포함된 PVC를 생성하거나 기존 PVC에 이러한 레이블을 추가합니다.

  다음 예는 `pvc1-group-snap`을 데이터 소스로 사용하고 `consistentGroupSnapshot: groupA`을 레이블로 사용하여 PVC를 생성합니다. 요구 사항에 따라 레이블 키와 값을 정의합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- PVC에 지정된 것과 동일한 레이블(`consistentGroupSnapshot: groupA`)을 사용하여 VolumeGroupSnapshot을 만듭니다.

  이 예에서는 볼륨 그룹 스냅샷을 생성합니다.

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

그룹 스냅샷을 사용하여 볼륨 데이터 복구

볼륨 그룹 스냅샷의 일부로 생성된 개별 스냅샷을 사용하여 개별 영구 볼륨을 복원할 수 있습니다. 볼륨 그룹 스냅샷을 하나의 단위로 복구할 수 없습니다.

볼륨 스냅샷 복원 ONTAP CLI를 사용하여 볼륨을 이전 스냅샷에 기록된 상태로 복원합니다.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

참고	스냅샷 복사본을 복원하면 기존 볼륨 구성이 덮어쓰여집니다. 스냅샷 복사본이 생성된 후 볼륨 데이터에 대한 변경 사항은 손실됩니다.

스냅샷에서 볼륨 제자리 복원

Trident는 `TridentActionSnapshotRestore` (TASR) CR을 사용하여 스냅샷에서 신속하게 제자리 볼륨 복원을 제공합니다. 이 CR은 명령형 Kubernetes 작업으로 작동하며 작업 완료 후에는 지속되지 않습니다.

자세한 내용은 "스냅샷에서 볼륨 제자리 복원"을 참조하십시오.

연결된 그룹 스냅샷이 있는 **PV** 삭제

그룹 볼륨 스냅샷을 삭제할 때:

- VolumeGroupSnapshots는 그룹 전체로 삭제할 수 있지만, 그룹 내의 개별 스냅샷은 삭제할 수 없습니다.
- PersistentVolumes가 해당 PersistentVolume에 대한 스냅샷이 존재하는 동안 삭제되면 Trident는 해당 볼륨을 "삭제 중" 상태로 이동합니다. 이는 볼륨을 안전하게 제거하기 전에 스냅샷을 먼저 제거해야 하기 때문입니다.
- 그룹 스냅샷을 사용하여 클론을 생성한 후 해당 그룹을 삭제하려는 경우, 클론 분할 작업이 시작되며 분할이 완료될 때까지 그룹을 삭제할 수 없습니다.

볼륨 스냅샷 컨트롤러를 배포합니다

Kubernetes 배포판에 스냅샷 컨트롤러와 CRD가 포함되어 있지 않은 경우 다음과 같이 배포할 수 있습니다.

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. 스냅샷 컨트롤러를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

참고	필요한 경우 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`를 열고 `namespace`를 네임스페이스로 업데이트하십시오.

관련 링크

- "VolumeGroupSnapshotClass"
- "볼륨 스냅샷"