모범 사례 및 권장 사항

Trident

NetApp
July 01, 2026

목차

모범 사례 및 권장 사항

구축

Trident를 배포할 때 여기에 나열된 권장 사항을 사용하십시오.

전용 네임스페이스에 배포

"네임스페이스"서로 다른 애플리케이션 간의 관리적 분리를 제공하고 리소스 공유를 방해하는 요소가 될 수 있습니다. 예를 들어, 한 네임스페이스의 PVC는 다른 네임스페이스에서 사용할 수 없습니다. Trident는 Kubernetes 클러스터의 모든 네임스페이스에 PV 리소스를 제공하며, 따라서 높은 권한을 가진 서비스 계정을 활용합니다.

또한 Trident Pod에 액세스하면 사용자가 스토리지 시스템 자격 증명 및 기타 민감한 정보에 액세스할 수 있습니다. 애플리케이션 사용자와 관리 애플리케이션이 Trident 객체 정의 또는 Pod 자체에 액세스할 수 없도록 하는 것이 중요합니다.

할당량 및 범위 제한을 사용하여 스토리지 사용량 제어

Kubernetes에는 애플리케이션의 리소스 사용량을 제한하는 강력한 메커니즘을 제공하는 두 가지 기능이 있습니다. "스토리지 할당량 메커니즘"를 사용하면 관리자가 네임스페이스별로 전역 및 스토리지 클래스별 용량 및 객체 개수 사용량 제한을 구현할 수 있습니다. 또한 "범위 제한"를 사용하면 PVC 요청이 프로비저너로 전달되기 전에 최소값과 최대값 범위 내에 있는지 확인할 수 있습니다.

이러한 값은 네임스페이스별로 정의되므로 각 네임스페이스는 리소스 요구 사항에 맞는 값을 정의해야 합니다. 에 대한 정보는 여기를 참조하십시오 "할당량을 활용하는 방법".

스토리지 구성

NetApp 포트폴리오의 각 스토리지 플랫폼은 컨테이너화된 애플리케이션이든 아니든 관계없이 애플리케이션에 도움이 되는 고유한 기능을 제공합니다.

플랫폼 개요

Trident는 ONTAP 및 Element와 함께 작동합니다. 모든 애플리케이션과 시나리오에 다른 것보다 더 적합한 플랫폼은 없지만, 플랫폼을 선택할 때는 애플리케이션의 요구 사항과 장치를 관리하는 팀을 고려해야 합니다.

사용하는 프로토콜과 호스트 운영 체제에 대한 기본 모범 사례를 따라야 합니다. 선택적으로, 특정 애플리케이션에 맞게 스토리지를 최적화하기 위해 백엔드, 스토리지 클래스 및 PVC 설정에 애플리케이션 모범 사례를 적용하는 것을 고려할 수 있습니다.

ONTAP 및 **Cloud Volumes ONTAP** 모범 사례

Trident용 ONTAP 및 Cloud Volumes ONTAP 구성 모범 사례를 알아보십시오.

다음 권장 사항은 Trident에서 동적으로 프로비저닝한 볼륨을 사용하는 컨테이너화된 워크로드에 대해 ONTAP를 구성하기 위한 지침입니다. 각 권장 사항은 사용 환경에 적합한지 검토하고 평가해야 합니다.

Trident 전용 **SVM** 사용

스토리지 가상 머신(SVM)은 ONTAP 시스템에서 테넌트 간의 격리 및 관리 분리를 제공합니다. 애플리케이션에 SVM을 전용으로 사용하면 권한 위임이 가능하고 리소스 사용량을 제한하기 위한 모범 사례를 적용할 수 있습니다.

SVM 관리에는 여러 가지 옵션이 있습니다.

- 백엔드 구성에서 클러스터 관리 인터페이스와 적절한 자격 증명을 제공하고 SVM 이름을 지정하십시오.
- ONTAP System Manager 또는 CLI를 사용하여 SVM용 전용 관리 인터페이스를 생성합니다.
- NFS 데이터 인터페이스와 관리 역할을 공유합니다.

각 경우에 인터페이스는 DNS에 등록되어 있어야 하며, Trident를 구성할 때 DNS 이름을 사용해야 합니다. 이렇게 하면 네트워크 ID 보존을 사용하지 않고도 SVM-DR과 같은 일부 재해 복구 시나리오를 구현할 수 있습니다.

SVM에 전용 관리 LIF를 사용할지 공유 관리 LIF를 사용할지 여부는 중요하지 않지만, 선택한 방식에 맞춰 네트워크 보안 정책을 수립해야 합니다. 관계없이, 관리 LIF는 DNS를 통해 액세스할 수 있어야 하며, "SVM-DR"Trident와 함께 사용될 경우 최대한의 유연성을 제공할 수 있습니다.

최대 볼륨 수 제한

ONTAP 스토리지 시스템에는 최대 볼륨 수가 있으며, 이는 소프트웨어 버전 및 하드웨어 플랫폼에 따라 다릅니다. 특정 플랫폼 및 ONTAP 버전에 대한 정확한 제한 사항을 확인하려면 "NetApp Hardware Universe"을 참조하십시오. 볼륨 수가 소진되면 Trident뿐만 아니라 모든 스토리지 요청에 대한 프로비저닝 작업이 실패합니다.

Trident의 `ontap-nas` 및 `ontap-san` 드라이버는 생성되는 각 Kubernetes 영구 볼륨(PV)에 대해 FlexVolume을 프로비저닝합니다. `ontap-nas-economy` 드라이버는 약 200개의 PV마다 하나의 FlexVolume을 생성합니다(50~300 사이에서 구성 가능). `ontap-san-economy` 드라이버는 약 100개의 PV마다 하나의 FlexVolume을 생성합니다(50~200 사이에서 구성 가능). Trident가 스토리지 시스템에서 사용 가능한 모든 볼륨을 소비하지 않도록 하려면 SVM에 제한을 설정해야 합니다. 명령줄에서 이 작업을 수행할 수 있습니다.

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

`max-volumes`의 값은 사용자 환경에 특정한 여러 기준에 따라 달라집니다.

- ONTAP 클러스터의 기존 볼륨 수
- 다른 애플리케이션에 대해 Trident 외부에서 프로비저닝할 것으로 예상되는 볼륨 수
- Kubernetes 애플리케이션에서 사용될 것으로 예상되는 영구 볼륨의 수

`max-volumes`값은 개별 `ONTAP` 노드가 아닌 `ONTAP` 클러스터의 모든 노드에 걸쳐 프로비저닝된 총 볼륨 수입니다. 따라서 `ONTAP` 클러스터 노드에 다른 노드보다 `Trident` 프로비저닝 볼륨이 훨씬 많거나 적을 수 있습니다.

예를 들어, 2노드 ONTAP 클러스터는 최대 2000개의 FlexVol 볼륨을 호스팅할 수 있습니다. 최대 볼륨 수를 1250으로 설정하는 것은 매우 적절해 보입니다. 그러나 한 노드에서만 "애그리게이트"이 SVM에 할당되거나, 한 노드에서 할당된 애그리게이트에 대해 프로비저닝이 불가능한 경우(예: 용량 부족), 다른 노드가 모든 Trident 프로비저닝 볼륨의 대상이

됩니다. 이는 `max-volumes` 값에 도달하기 전에 해당 노드의 볼륨 제한에 도달할 수 있음을 의미하며, Trident 및 해당 노드를 사용하는 다른 볼륨 작업 모두에 영향을 미칠 수 있습니다. 클러스터의 각 노드에서 **Trident**가 사용하는 **SVM**에 동일한 수의 애그리게이트가 할당되도록 하면 이러한 상황을 방지할 수 있습니다**.**

볼륨 복제

NetApp Trident는 `ontap-nas`, `ontap-san` 및 `solidfire-san` 스토리지 드라이버를 사용할 때 볼륨 복제를 지원합니다. `ontap-nas-flexgroup` 또는 `ontap-nas-economy` 드라이버를 사용할 때는 복제가 지원되지 않습니다. 기존 볼륨에서 새 볼륨을 생성하면 새 스냅샷이 생성됩니다.

> 다른 StorageClass와 연결된 PVC를 복제하지 마십시오. 호환성을 보장하고 예기치 않은 동작을 방지하려면 동일한 StorageClass 내에서 복제 작업을 수행하십시오.

Trident에서 생성한 볼륨의 최대 크기 제한

Trident에서 생성할 수 있는 볼륨의 최대 크기를 구성하려면, `limitVolumeSize` 매개변수를 `backend.json` 정의에 사용하십시오.

스토리지 어레이에서 볼륨 크기를 제어하는 것 외에도 Kubernetes 기능을 활용해야 합니다.

Trident에서 생성한 **FlexVols**의 최대 크기 제한

ontap-san-economy 및 ontap-nas-economy 드라이버의 풀로 사용되는 FlexVols의 최대 크기를 구성하려면, `limitVolumePoolSize` 매개변수를 `backend.json` 정의에 사용하십시오.

양방향 **CHAP**를 사용하도록 **Trident**를 구성합니다

백엔드 정의에서 CHAP 이니시에이터 및 대상 사용자 이름과 암호를 지정하고 Trident가 SVM에서 CHAP를 활성화하도록 설정할 수 있습니다. 백엔드 구성에서 `useCHAP` 매개 변수를 사용하면 Trident는 CHAP를 통해 ONTAP 백엔드에 대한 iSCSI 연결을 인증합니다.

SVM QoS 정책 생성 및 사용

SVM에 적용되는 ONTAP QoS 정책을 활용하면 Trident 프로비저닝된 볼륨이 소비할 수 있는 IOPS 수를 제한할 수 있습니다. 이는 "괴롭힘을 예방하다" 또는 제어 불능 상태의 컨테이너가 Trident SVM 외부의 워크로드에 영향을 미치는 것을 방지하는 데 도움이 됩니다.

몇 단계만 거치면 SVM에 대한 QoS 정책을 생성할 수 있습니다. 가장 정확한 정보는 사용 중인 ONTAP 버전의 설명서를 참조하십시오. 아래 예시는 SVM에서 사용 가능한 총 IOPS를 5000으로 제한하는 QoS 정책을 생성합니다.

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

또한, 사용 중인 ONTAP 버전에서 지원하는 경우, 컨테이너화된 워크로드에 일정량의 처리량을 보장하기 위해 QoS

최소값을 사용하는 것을 고려할 수 있습니다. 적응형 QoS는 SVM 수준 정책과 호환되지 않습니다.

컨테이너화된 워크로드에 할당되는 IOPS 수는 여러 요소에 따라 달라집니다. 그중에서도 다음과 같은 요소들이 포함됩니다:

- 스토리지 어레이를 사용하는 다른 워크로드. Kubernetes 배포와 관련이 없는 다른 워크로드가 스토리지 리소스를 사용하는 경우, 해당 워크로드에 실수로 악영향을 미치지 않도록 주의해야 합니다.
- 컨테이너에서 실행될 것으로 예상되는 워크로드입니다. 높은 IOPS 요구 사항을 가진 워크로드가 컨테이너에서 실행될 경우, 낮은 QoS 정책은 사용자 경험 저하로 이어질 수 있습니다.

SVM 레벨에서 할당된 QoS 정책은 해당 SVM에 프로비저닝된 모든 볼륨이 동일한 IOPS 풀을 공유하게 된다는 점을 기억하는 것이 중요합니다. 컨테이너화된 애플리케이션 중 하나 또는 소수의 애플리케이션이 높은 IOPS 요구 사항을 가질 경우, 다른 컨테이너화된 워크로드에 과도한 부담을 줄 수 있습니다. 이러한 경우, 외부 자동화 도구를 사용하여 볼륨별 QoS 정책을 할당하는 것을 고려해 볼 수 있습니다.

ONTAP 버전이 9.8 이전인 경우에*만* QoS 정책 그룹을 SVM에 할당해야 합니다.

Trident용 **QoS** 정책 그룹 생성

QoS(서비스 품질)는 중요한 워크로드의 성능이 경쟁 워크로드로 인해 저하되지 않도록 보장합니다. ONTAP QoS 정책 그룹은 볼륨에 대한 QoS 옵션을 제공하며, 사용자가 하나 이상의 워크로드에 대한 처리량 상한을 정의할 수 있도록 합니다. QoS에 대한 자세한 내용은 "QoS를 통해 처리량 보장"을 참조하십시오. QoS 정책 그룹은 백엔드 또는 스토리지 풀에 지정할 수 있으며, 해당 풀 또는 백엔드에 생성된 각 볼륨에 적용됩니다.

ONTAP에는 기존 QoS 정책 그룹과 적응형 QoS 정책 그룹 두 가지 유형이 있습니다. 기존 정책 그룹은 IOPS 기준으로 최대(또는 최신 버전에서는 최소) 처리량을 고정적으로 제공합니다. 적응형 QoS는 워크로드 크기에 따라 처리량을 자동으로 확장하여 워크로드 크기 변화에 따른 IOPS 대 TB/GB 비율을 유지합니다. 이는 대규모 배포 환경에서 수백 또는 수천 개의 워크로드를 관리할 때 상당한 이점을 제공합니다.

QoS 정책 그룹을 생성할 때 다음 사항을 고려하십시오.

- 백엔드 구성의 `qosPolicy` 블록에서 `defaults` 키를 설정해야 합니다. 다음은 백엔드 구성 예시입니다.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 0.0.0.0
dataLIF: 0.0.0.0
svm: svm0
username: user
password: pass
defaults:
  qosPolicy: standard-pg
storage:
  - labels:
      performance: extreme
    defaults:
      adaptiveQosPolicy: extremely-adaptive-pg
  - labels:
      performance: premium
    defaults:
      qosPolicy: premium-pg
```

- 볼륨당 정책 그룹을 적용해야 각 볼륨이 정책 그룹에서 지정한 전체 처리량을 얻을 수 있습니다. 공유 정책 그룹은 지원되지 않습니다.

QoS 정책 그룹에 대한 자세한 내용은 "ONTAP 명령 참조"를 참조하십시오.

Kubernetes 클러스터 구성원으로 스토리지 리소스 액세스 제한

Trident에서 생성한 NFS 볼륨, iSCSI LUN 및 FC LUN에 대한 액세스를 제한하는 것은 Kubernetes 배포 환경의 보안 태세에서 중요한 구성 요소입니다. 이를 통해 Kubernetes 클러스터에 속하지 않은 호스트가 해당 볼륨에 액세스하여 예기치 않게 데이터를 수정하는 것을 방지할 수 있습니다.

Kubernetes에서 네임스페이스는 리소스의 논리적 경계라는 점을 이해하는 것이 중요합니다. 동일한 네임스페이스 내의 리소스는 공유될 수 있다는 가정이 있지만, 중요한 것은 네임스페이스 간 기능이 없다는 것입니다. 즉, PV는 전역 객체이지만 PVC에 바인딩되면 동일한 네임스페이스에 있는 Pod에서만 액세스할 수 있습니다. 적절한 경우 네임스페이스를 사용하여 분리를 제공하는 것이 중요합니다.

Kubernetes 환경에서 데이터 보안과 관련하여 대부분의 조직이 가장 우려하는 사항은 컨테이너의 프로세스가 호스트에 마운트된 스토리지에 액세스할 수 있지만 컨테이너용이 아니라는 점입니다. "네임스페이스"는 이러한 유형의 침해를 방지하도록 설계되었습니다. 그러나 예외가 하나 있습니다. 바로 권한 있는 컨테이너입니다.

특권 컨테이너는 일반 컨테이너보다 훨씬 더 많은 호스트 수준 권한으로 실행되는 컨테이너입니다. 이러한 권한은 기본적으로 거부되지 않으므로 "Pod 보안 정책"을 사용하여 해당 기능을 비활성화해야 합니다.

Kubernetes와 외부 호스트 모두에서 액세스가 필요한 볼륨의 경우, 스토리지는 기존 방식으로 관리되어야 하며, PV는 관리자가 도입하고 Trident에서 관리하지 않아야 합니다. 이렇게 하면 Kubernetes와 외부 호스트 모두 연결이 끊어지고 더 이상 볼륨을 사용하지 않을 때만 스토리지 볼륨이 삭제됩니다. 또한, 사용자 지정 내보내기 정책을 적용하여 Kubernetes 클러스터 노드와 Kubernetes 클러스터 외부의 대상 서버에서 액세스할 수 있도록 할 수 있습니다.

전용 인프라 노드(예: OpenShift) 또는 사용자 애플리케이션 예약이 불가능한 노드가 있는 배포 환경의 경우, 스토리지

리소스에 대한 액세스를 더욱 제한하기 위해 별도의 내보내기 정책을 사용해야 합니다. 여기에는 해당 인프라 노드에 배포된 서비스(예: OpenShift 메트릭 및 로깅 서비스)와 인프라 노드가 아닌 곳에 배포된 표준 애플리케이션에 대한 내보내기 정책을 생성하는 것이 포함됩니다.

전용 엑스포트 정책 사용

각 백엔드에 대해 Kubernetes 클러스터에 있는 노드에만 액세스를 허용하는 엑스포트 정책이 있는지 확인해야 합니다. Trident는 엑스포트 정책을 자동으로 생성하고 관리할 수 있습니다. 이를 통해 Trident는 프로비저닝하는 볼륨에 대한 액세스를 Kubernetes 클러스터의 노드로 제한하고 노드 추가/삭제를 간소화합니다.

또는 수동으로 엑스포트 정책을 생성하고 각 노드 액세스 요청을 처리하는 하나 이상의 엑스포트 규칙을 추가할 수도 있습니다.

- `vserver export-policy create` ONTAP CLI 명령을 사용하여 엑스포트 정책을 생성합니다.
- `vserver export-policy rule create` ONTAP CLI 명령을 사용하여 엑스포트 정책에 규칙을 추가합니다.

이러한 명령을 실행하면 데이터에 액세스할 수 있는 Kubernetes 노드를 제한할 수 있습니다.

`showmount`애플리케이션 SVM에 대해 비활성화합니다

`showmount` 기능을 사용하면 `NFS` 클라이언트가 `SVM`에 사용 가능한 `NFS` 내보내기 목록을 쿼리할 수 있습니다. `Kubernetes` 클러스터에 배포된 `Pod`는 `showmount -e` 명령을 실행하여 액세스 권한이 없는 마운트를 포함하여 사용 가능한 마운트 목록을 받을 수 있습니다. 이는 그 자체로는 보안 침해가 아니지만, 권한이 없는 사용자가 `NFS` 내보내기에 연결하는 데 도움이 될 수 있는 불필요한 정보를 제공합니다.

SVM 레벨 ONTAP CLI 명령을 사용하여 `showmount` 비활성화해야 합니다.

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire 모범 사례

Trident용 SolidFire 스토리지 구성 모범 사례를 알아보십시오.

SolidFire 계정 생성

각 SolidFire 계정은 고유한 볼륨 소유자를 나타내며 고유한 CHAP(Challenge-Handshake Authentication Protocol) 자격 증명 세트를 받습니다. 계정에 할당된 볼륨은 계정 이름과 관련 CHAP 자격 증명을 사용하거나 볼륨 액세스 그룹을 통해 액세스할 수 있습니다. 계정에는 최대 2,000개의 볼륨을 할당할 수 있지만 볼륨은 하나의 계정에만 속할 수 있습니다.

QoS 정책을 생성합니다

여러 볼륨에 적용할 수 있는 표준화된 서비스 품질 설정을 생성하고 저장하려면 SolidFire QoS(Quality of Service) 정책을 사용하십시오.

볼륨별로 QoS 파라미터를 설정할 수 있습니다. QoS를 정의하는 세 가지 구성 가능한 파라미터(최소 IOPS, 최대 IOPS,

버스트 IOPS)를 설정하여 각 볼륨의 성능을 보장할 수 있습니다.

다음은 4Kb 블록 크기에 대한 최소, 최대 및 버스트 IOPS 값입니다.

IOPS 매개변수	정의	최소값	기본값	최대값**(4Kb)**
최소 IOPS	볼륨에 대해 보장되는 성능 수준입니다.	50	50	15000
최대 IOPS	성능은 이 한도를 초과하지 않습니다.	50	15000	200,000
버스트 IOPS	단기 버스트 시나리오에서 허용되는 최대 IOPS입니다.	50	15000	200,000

최대 IOPS와 버스트 IOPS는 최대 200,000까지 설정할 수 있지만, 볼륨의 실제 최대 성능은 클러스터 사용량과 노드별 성능에 따라 제한됩니다.

블록 크기와 대역폭은 IOPS 수에 직접적인 영향을 미칩니다. 블록 크기가 증가함에 따라 시스템은 더 큰 블록 크기를 처리하는 데 필요한 수준까지 대역폭을 증가시킵니다. 대역폭이 증가하면 시스템이 달성할 수 있는 IOPS 수는 감소합니다. QoS 및 성능에 대한 자세한 내용은 "SolidFire 서비스 품질"을 참조하십시오.

SolidFire 인증

Element는 CHAP와 VAG(Volume Access Groups)라는 두 가지 인증 방식을 지원합니다. CHAP는 CHAP 프로토콜을 사용하여 호스트를 백엔드에 인증합니다. Volume Access Groups는 프로비저닝하는 볼륨에 대한 액세스를 제어합니다. NetApp은 CHAP 인증 방식이 더 간단하고 확장성에 제한이 없으므로 CHAP 사용을 권장합니다.

향상된 CSI 프로비저너를 사용하는 Trident는 CHAP 인증을 지원합니다. VAG는 기존의 비CSI 운영 모드에서만 사용해야 합니다.

CHAP 인증(시작자가 의도된 볼륨 사용자인지 확인하는 인증)은 계정 기반 액세스 제어에서만 지원됩니다. CHAP를 사용하여 인증하는 경우 단방향 CHAP와 양방향 CHAP의 두 가지 옵션을 사용할 수 있습니다. 단방향 CHAP는 SolidFire 계정 이름과 시작자 암호를 사용하여 볼륨 액세스를 인증합니다. 양방향 CHAP 옵션은 볼륨이 계정 이름과 시작자 암호를 통해 호스트를 인증하고, 호스트는 계정 이름과 대상 암호를 통해 볼륨을 인증하는 가장 안전한 방법입니다.

하지만 CHAP를 활성화할 수 없고 VAG가 필요한 경우 액세스 그룹을 생성하고 호스트 이니시에이터와 볼륨을 액세스 그룹에 추가하십시오. 액세스 그룹에 추가하는 각 IQN은 CHAP 인증 여부와 관계없이 그룹의 각 볼륨에 액세스할 수 있습니다. iSCSI 이니시에이터가 CHAP 인증을 사용하도록 구성된 경우 계정 기반 액세스 제어가 사용됩니다. iSCSI 이니시에이터가 CHAP 인증을 사용하도록 구성되지 않은 경우 Volume Access Group 액세스 제어가 사용됩니다.

자세한 정보는 어디에서 찾을 수 있습니까?

다음은 모범 사례 문서의 일부 목록입니다. "NetApp 라이브러리"에서 최신 버전을 검색하십시오.

ONTAP

- "NFS 모범 사례 및 구현 가이드"
- "SAN 관리" (iSCSI용)
- "RHEL용 iSCSI Express 구성"

Element 소프트웨어

- "Linux용 SolidFire 구성"

NetApp HCI

- "NetApp HCI 배포 필수 조건"
- "NetApp Deployment Engine에 액세스합니다"

애플리케이션 모범 사례 정보

- "ONTAP의 MySQL 모범 사례"
- "SolidFire의 MySQL 모범 사례"
- "NetApp SolidFire 및 Cassandra"
- "SolidFire의 Oracle 모범 사례"
- "SolidFire의 PostgreSQL 모범 사례"

모든 애플리케이션에 특정 지침이 있는 것은 아니므로 NetApp 팀과 협력하고 "NetApp 라이브러리"를 사용하여 최신 문서를 찾는 것이 중요합니다.

Trident 통합

Trident를 통합하려면 드라이버 선택 및 배포, 스토리지 클래스 설계, 가상 풀 설계, 스토리지 프로비저닝에 대한 영구 볼륨 클레임(PVC) 영향, 볼륨 작업 및 Trident를 사용한 OpenShift 서비스 배포와 같은 설계 및 아키텍처 요소를 통합해야 합니다.

드라이버 선택 및 배포

스토리지 시스템의 백엔드 드라이버를 선택하고 배포합니다.

ONTAP 백엔드 드라이버

ONTAP 백엔드 드라이버는 사용되는 프로토콜과 스토리지 시스템에서 볼륨이 프로비저닝되는 방식에 따라 구분됩니다. 따라서 어떤 드라이버를 배포할지 결정할 때 신중하게 고려해야 합니다.

좀 더 자세히 설명하자면, 애플리케이션에 공유 스토리지(여러 파드가 동일한 PVC에 액세스하는 경우)가 필요한 구성 요소가 있다면 NAS 기반 드라이버가 기본 선택 사항이 되고, 블록 기반 iSCSI 드라이버는 공유 스토리지가 필요 없는 경우에 적합합니다. 애플리케이션 요구 사항과 스토리지 및 인프라 팀의 숙련도를 고려하여 프로토콜을 선택하십시오. 일반적으로 대부분의 애플리케이션에서 두 프로토콜 간의 차이는 크지 않으므로, 여러 파드가 동시에 액세스해야 하는 공유 스토리지가 필요한지 여부에 따라 결정되는 경우가 많습니다.

사용 가능한 ONTAP 백엔드 드라이버는 다음과 같습니다.

- `ontap-nas`: 프로비저닝된 각 PV는 완전한 ONTAP FlexVolume입니다.
- `ontap-nas-economy`: 프로비저닝된 각 PV는 qtree이며, FlexVolume당 구성 가능한 qtree 수를 가집니다 (기본값은 200개).
- `ontap-nas-flexgroup`: 각 PV는 전체 ONTAP FlexGroup로 프로비저닝되며 SVM에 할당된 모든 애그리게이트가 사용됩니다.
- `ontap-san`: 프로비저닝된 각 PV는 자체 FlexVolume 내의 LUN입니다.
- `ontap-san-economy`: 프로비저닝된 각 PV는 LUN이며, FlexVolume당 구성 가능한 LUN 수가 있습니다 (기본값은 100개).

세 가지 NAS 드라이버 중에서 선택하는 것은 애플리케이션에서 사용할 수 있는 기능에 몇 가지 영향을 미칩니다.

아래 표에서 모든 기능이 Trident를 통해 노출되는 것은 아닙니다. 일부 기능은 필요한 경우 스토리지 관리자가 프로비저닝 후 적용해야 합니다. 위첨자 각주는 기능 및 드라이버별 기능을 구분하여 보여줍니다.

ONTAP NAS 드라이버	스냅샷	클론	동적 엑스포트 정책	다중 연결	**QoS**	크기 조정	복제
`ontap-nas`	예	예	예 [5]	예	예 [1]	예	예 [1]
`ontap-nas-economy`	NO [3]	NO [3]	예 [5]	예	NO [3]	예	NO [3]
`ontap-nas-flexgroup`	예 [1]	아니요	예 [5]	예	예 [1]	예	예 [1]

Trident는 ONTAP용 SAN 드라이버 2개를 제공하며, 해당 기능은 아래와 같습니다.

ONTAP SAN 드라이버	스냅샷	클론	다중 연결	양방향 **CHAP**	**QoS**	크기 조정	복제
`ontap-san`	예	예	예 [4]	예	예 [1]	예	예 [1]
`ontap-san-economy`	예	예	예 [4]	예	NO [3]	예	NO [3]

> 위 표에 대한 각주: 예각주:1[]: Trident에서 관리하지 않음 예각주:2[]: Trident에서 관리하지만 PV 세분화는 지원하지 않음 아니오각주:3[]: Trident에서 관리하지 않으며 PV 세분화도 지원하지 않음 예각주:4[]: 원시 블록 볼륨에 대해 지원됨 예각주:5[]: Trident에서 지원됨

PV 단위로 세분화되지 않은 기능은 전체 FlexVolume에 적용되며 모든 PV(즉, 공유 FlexVols의 qtree 또는 LUN)는 공통 스케줄을 공유합니다.

위 표에서 볼 수 있듯이 `ontap-nas`와 `ontap-nas-economy`의 대부분의 기능은 동일합니다. 그러나 `ontap-nas-economy` 드라이버는 PV 단위로 스케줄을 제어하는 기능을 제한하기 때문에 특히 재해 복구 및 백업 계획에 영향을 미칠 수 있습니다. ONTAP 스토리지에서 PVC 클론 기능을 활용하려는 개발 팀의 경우 `ontap-nas`, `ontap-san` 또는 `ontap-san-economy` 드라이버를 사용할 때만 가능합니다.

> `solidfire-san` 드라이버는 PVC를 복제하는 기능도 갖추고 있습니다.

Cloud Volumes ONTAP 백엔드 드라이버

Cloud Volumes ONTAP는 파일 공유 및 NAS 및 SAN 프로토콜(NFS, SMB/CIFS 및 iSCSI)을 지원하는 블록 레벨 스토리지를 포함한 다양한 사용 사례를 위한 엔터프라이즈급 스토리지 기능과 함께 데이터 제어 기능을 제공합니다. Cloud Volume ONTAP와 호환되는 드라이버는 `ontap-nas`, `ontap-nas-economy`, `ontap-san` 및 `ontap-san-economy`입니다. 이러한 드라이버는 Azure용 Cloud Volume ONTAP, GCP용 Cloud Volume ONTAP에 적용됩니다.

Amazon FSx for ONTAP 백엔드 드라이버

Amazon FSx for NetApp ONTAP을 사용하면 익숙한 NetApp 기능, 성능 및 관리 기능을 활용하는 동시에 AWS에 데이터를 저장하는 간편성, 민첩성, 보안 및 확장성의 이점을 누릴 수 있습니다. FSx for ONTAP은 다양한 ONTAP 파일 시스템 기능과 관리 API를 지원합니다. Cloud Volume ONTAP과 호환되는 드라이버는 `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` 및 `ontap-san-economy`입니다.

NetApp HCI/SolidFire 백엔드 드라이버

`solidfire-san` NetApp HCI/SolidFire 플랫폼에서 사용되는 드라이버를 통해 관리자는 QoS 제한을 기반으로 Trident용 Element 백엔드를 구성할 수 있습니다. Trident에서 프로비저닝한 볼륨에 특정 QoS 제한을 설정하도록 백엔드를 설계하려면 백엔드 파일에서 `type` 매개변수를 사용하십시오. 관리자는 또한 `limitVolumeSize` 매개변수를 사용하여 스토리지에 생성할 수 있는 볼륨 크기를 제한할 수 있습니다. 현재 볼륨 크기 조정 및 볼륨 복제와 같은 Element 스토리지 기능은 `solidfire-san` 드라이버를 통해 지원되지 않습니다. 이러한 작업은 Element Software 웹 UI를 통해 수동으로 수행해야 합니다.

SolidFire 드라이버	스냅샷	클론	다중 연결	CHAP	QoS	크기 조정	복제
solidfire-san	예	예	예 [2]	예	예	예	예 [1]

> 각주: 예각주:1[]: Trident에서 관리하지 않음 예각주:2[]: 원시 블록 볼륨에 대해 지원됨

Azure NetApp Files 백엔드 드라이버

Trident는 `azure-netapp-files` 드라이버를 사용하여 "Azure NetApp Files" 서비스를 관리합니다.

이 드라이버 및 구성 방법에 대한 자세한 내용은 "Azure NetApp Files용 Trident 백엔드 구성"에서 확인할 수 있습니다.

Azure NetApp Files 드라이버	스냅샷	클론	다중 연결	QoS	확장	복제
azure-netapp-files	예	예	예	예	예	예 [1]

> 각주: 예각주:1[]: Trident에서 관리되지 않음

스토리지 클래스 설계

Kubernetes 스토리지 클래스 객체를 생성하려면 개별 스토리지 클래스를 구성하고 적용해야 합니다. 이 섹션에서는 애플리케이션에 맞는 스토리지 클래스를 설계하는 방법을 설명합니다.

특정 백엔드 활용

특정 스토리지 클래스 객체 내에서 필터링을 사용하여 해당 스토리지 클래스와 함께 사용할 스토리지 풀 또는 풀 집합을 결정할 수 있습니다. 스토리지 클래스에는 세 가지 필터 집합 `storagePools`, `additionalStoragePools` 및 /또는 `excludeStoragePools`을 설정할 수 있습니다.

`storagePools` 매개변수는 지정된 속성과 일치하는 풀 집합으로 스토리지를 제한하는 데 도움이 됩니다. `additionalStoragePools` 매개변수는 속성 및 `storagePools` 매개변수로 선택된 풀 집합과 함께 Trident가 프로비저닝에 사용하는 풀 집합을 확장하는 데 사용됩니다. 적절한 스토리지 풀 집합이 선택되도록 하려면 두 매개변수를 단독으로 또는 함께 사용할 수 있습니다.

`excludeStoragePools` 매개변수는 속성과 일치하는 나열된 풀 집합을 명시적으로 제외하는 데 사용됩니다.

QoS 정책 에뮬레이트

QoS(서비스 품질) 정책을 모방하도록 스토리지 클래스를 설계하려면 `media` 속성을 `hdd` 또는 `ssd`로 설정하여 스토리지 클래스를 생성하십시오. 스토리지 클래스에 지정된 `media` 속성을 기반으로 Trident는 `hdd` 또는 `ssd` 애그리게이트를 제공하는 적절한 백엔드를 선택하여 미디어 속성과 일치시키고 특정 애그리게이트에 볼륨 프로비저닝을 지시합니다. 따라서 `media` 속성을 `ssd`로 설정한 PREMIUM 스토리지 클래스를 생성하면 PREMIUM QoS 정책으로 분류할 수 있습니다. 마찬가지로 미디어 속성을 `hdd'로 설정한 STANDARD 스토리지 클래스를 생성하면 STANDARD QoS 정책으로 분류할 수 있습니다. 또한 스토리지 클래스의 ``IOPS" 속성을 사용하여 Element 어플라이언스로 프로비저닝을 리디렉션하는 것도 QoS 정책으로 정의할 수 있습니다.

특정 기능을 기반으로 백엔드 활용

스토리지 클래스는 씬 프로비저닝 및 씩 프로비저닝, 스냅샷, 클론, 암호화와 같은 기능이 활성화된 특정 백엔드에서 볼륨 프로비저닝을 수행하도록 설계할 수 있습니다. 사용할 스토리지를 지정하려면 필요한 기능이 활성화된 적절한 백엔드를 지정하는 스토리지 클래스를 생성하십시오.

가상 풀

가상 풀은 모든 Trident 백엔드에서 사용할 수 있습니다. Trident에서 제공하는 모든 드라이버를 사용하여 모든 백엔드에 대한 가상 풀을 정의할 수 있습니다.

가상 풀을 사용하면 관리자가 스토리지 클래스를 통해 참조할 수 있는 백엔드에 대한 추상화 계층을 생성하여 백엔드에 볼륨을 더욱 유연하고 효율적으로 배치할 수 있습니다. 동일한 서비스 클래스로 여러 백엔드를 정의할 수 있습니다. 또한 동일한 백엔드에 서로 다른 특성을 가진 여러 스토리지 풀을 생성할 수도 있습니다. 스토리지 클래스에 특정 레이블이 있는 선택기가 구성되면 Trident는 선택기의 모든 레이블과 일치하는 백엔드를 선택하여 볼륨을 배치합니다. 스토리지 클래스 선택기 레이블이 여러 스토리지 풀과 일치하는 경우 Trident는 그중 하나를 선택하여 볼륨을 프로비저닝합니다.

가상 풀 설계

백엔드를 생성할 때 일반적으로 매개변수 세트를 지정할 수 있습니다. 관리자가 동일한 스토리지 자격 증명을 사용하면서 다른 매개변수 세트를 지정하여 다른 백엔드를 생성하는 것이 불가능했습니다. 가상 풀이 도입되면서 이 문제가 해결되었습니다. 가상 풀은 백엔드와 Kubernetes 스토리지 클래스 사이에 도입된 추상화 계층으로, 관리자가 백엔드에 구애받지 않고 Kubernetes 스토리지 클래스에서 선택기로 참조할 수 있는 레이블과 함께 매개변수를 정의할

수 있도록 합니다. 가상 풀은 Trident에서 지원되는 모든 NetApp 백엔드에 대해 정의할 수 있습니다. 여기에는 SolidFire/NetApp HCI, ONTAP 및 Azure NetApp Files가 포함됩니다.

> 가상 풀을 정의할 때 백엔드 정의에서 기존 가상 풀의 순서를 변경하지 않는 것이 좋습니다. 또한 기존 가상 풀의 속성을 편집/수정하지 말고 새 가상 풀을 정의하는 것이 좋습니다.

다양한 서비스 수준**/QoS** 에뮬레이션

서비스 클래스를 에뮬레이션하기 위한 가상 풀을 설계할 수 있습니다. Azure NetApp Files용 Cloud Volume Service의 가상 풀 구현을 사용하여 다양한 서비스 클래스를 설정하는 방법을 살펴보겠습니다. Azure NetApp Files 백엔드에 서로 다른 성능 수준을 나타내는 여러 레이블을 구성합니다. `servicelevel` 애스펙트를 적절한 성능 수준으로 설정하고 각 레이블 아래에 필요한 다른 애스펙트를 추가합니다. 이제 서로 다른 가상 풀에 매핑되는 다양한 Kubernetes 스토리지 클래스를 생성합니다. `parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 지정합니다.

특정 측면 집합 할당

하나의 스토리지 백엔드에서 특정 속성을 가진 여러 개의 가상 풀을 설계할 수 있습니다. 이를 위해 백엔드에 여러 레이블을 구성하고 각 레이블 아래에 필요한 속성을 설정합니다. 이제 `parameters.selector` 필드를 사용하여 각 가상 풀에 매핑되는 서로 다른 Kubernetes 스토리지 클래스를 생성합니다. 백엔드에 프로비저닝되는 볼륨은 선택한 가상 풀에 정의된 속성을 갖게 됩니다.

스토리지 프로비저닝에 영향을 미치는 **PVC** 특성

PVC를 생성할 때 요청된 스토리지 클래스 외의 일부 매개변수가 Trident 프로비저닝 결정 프로세스에 영향을 미칠 수 있습니다.

액세스 모드

PVC를 통해 스토리지를 요청할 때 필수 입력 항목 중 하나는 액세스 모드입니다. 원하는 모드에 따라 스토리지 요청을 호스팅하는 백엔드가 선택될 수 있습니다.

Trident는 다음 매트릭스에 따라 지정된 액세스 방법과 함께 사용되는 스토리지 프로토콜을 일치시키려고 시도합니다. 이는 기본 스토리지 플랫폼과 무관합니다.

	ReadWriteOnce	**ReadOnlyMany**	**ReadWriteMany**
iSCSI	예	예	예(Raw 블록)
NFS	예	예	예

NFS 백엔드가 구성되지 않은 Trident 배포 환경에 ReadWriteMany PVC를 요청하면 볼륨이 프로비저닝되지 않습니다. 따라서 요청자는 애플리케이션에 적합한 액세스 모드를 사용해야 합니다.

볼륨 작업

영구 볼륨 수정

Kubernetes에서 영구 볼륨은 두 가지 예외를 제외하고는 변경 불가능한 객체입니다. 일단 생성되면 회수 정책과 크기는 수정할 수 있습니다. 하지만 그렇다고 해서 Kubernetes 외부에서 볼륨의 일부 측면을 수정하는 것이 불가능한 것은 아닙니다. 특정 애플리케이션에 맞게 볼륨을 사용자 지정하거나, 용량이 실수로 소모되는 것을 방지하거나, 또는 어떤 이유로든 볼륨을 다른 스토리지 컨트롤러로 이동해야 하는 경우에 이러한 수정이 필요할 수 있습니다.

현재 Kubernetes 내장 프로비저너는 NFS, iSCSI 또는 FC PV에 대한 볼륨 크기 조정 작업을 지원하지 않습니다. Trident는 NFS, iSCSI 및 FC 볼륨 모두의 확장을 지원합니다.

PV의 연결 세부 정보는 생성 후 수정할 수 없습니다.

필요 시 볼륨 스냅샷 생성

Trident는 CSI 프레임워크를 사용하여 온디맨드 볼륨 스냅샷 생성 및 스냅샷으로부터 PVC 생성을 지원합니다. 스냅샷은 데이터의 특정 시점 복사본을 유지하는 편리한 방법을 제공하며 Kubernetes의 소스 PV와는 독립적인 수명 주기를 가집니다. 이러한 스냅샷을 사용하여 PVC를 복제할 수 있습니다.

스냅샷에서 볼륨 생성

Trident는 볼륨 스냅샷에서 PersistentVolumes를 생성하는 기능도 지원합니다. 이를 위해 PersistentVolumeClaim을 생성하고 `datasource`볼륨을 생성할 필요한 스냅샷으로 지정하기만 하면 됩니다. Trident는 스냅샷에 있는 데이터로 볼륨을 생성하여 이 PVC를 처리합니다. 이 기능을 사용하면 리전 간 데이터 복제, 테스트 환경 생성, 손상되거나 오류가 발생한 프로덕션 볼륨 전체 교체 또는 특정 파일 및 디렉토리 검색 후 다른 연결된 볼륨으로 전송이 가능합니다.

클러스터에서 볼륨 이동

스토리지 관리자는 ONTAP 클러스터 내의 애그리게이트와 컨트롤러 간에 스토리지 소비자에게 중단 없이 볼륨을 이동할 수 있습니다. 대상 애그리게이트가 Trident에서 사용 중인 SVM에 접근 권한이 있는 애그리게이트인 한 이 작업은 Trident 또는 Kubernetes 클러스터에 영향을 미치지 않습니다. 중요한 점은 애그리게이트가 SVM에 새로 추가된 경우 Trident에 다시 추가하여 백엔드를 새로 고쳐야 한다는 것입니다. 이렇게 하면 Trident가 SVM의 인벤토리를 다시 생성하여 새 애그리게이트를 인식하게 됩니다.

하지만 백엔드 간에 볼륨을 이동하는 것은 Trident에서 자동으로 지원되지 않습니다. 여기에는 동일한 클러스터 내의 SVM 간, 클러스터 간 또는 다른 스토리지 플랫폼으로의 이동(해당 스토리지 시스템이 Trident에 연결되어 있는 경우에도)이 포함됩니다.

볼륨을 다른 위치로 복사한 경우, 볼륨 가져오기 기능을 사용하여 현재 볼륨을 Trident로 가져올 수 있습니다.

볼륨 확장

Trident는 NFS, iSCSI, 그리고 FC PV의 크기 조정을 지원합니다. 이를 통해 사용자는 Kubernetes 계층에서 직접 볼륨의 크기를 조정할 수 있습니다. 볼륨 확장은 ONTAP를 포함한 모든 주요 NetApp 스토리지 플랫폼과 SolidFire/NetApp HCI 백엔드에서 가능합니다. 나중에 확장이 가능하도록, 볼륨과 연결된 StorageClass에서 `allowVolumeExpansion`를 `true`로 설정하세요. `Persistent Volume`의 크기를 조정해야 할 때마다 `Persistent Volume Claim`의 `spec.resources.requests.storage` 주석을 원하는 볼륨 크기로 수정하세요. Trident가 스토리지 클러스터에서 볼륨 크기 조정을 자동으로 처리합니다.

기존 볼륨을 **Kubernetes**로 가져오기

볼륨 가져오기 기능을 사용하면 기존 스토리지 볼륨을 Kubernetes 환경으로 가져올 수 있습니다. 현재 이 기능은 `ontap-nas`, `ontap-nas-flexgroup`, `solidfire-san` 및 `azure-netapp-files` 드라이버에서 지원됩니다. 이 기능은 기존 애플리케이션을 Kubernetes로 포팅하거나 재해 복구 시나리오에서 유용합니다.

ONTAP 및 `solidfire-san`드라이버를 사용하는 경우 `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml`명령을 사용하여 기존 볼륨을 Kubernetes로 가져와 Trident에서 관리할 수 있습니다. 볼륨 가져오기 명령에 사용되는 PVC YAML 또는 JSON 파일은 Trident를 프로비저너로 식별하는 스토리지 클래스를 가리킵니다. NetApp HCI/SolidFire 백엔드를 사용하는 경우 볼륨 이름이 고유한지 확인하십시오. 볼륨 이름이 중복되는 경우 볼륨을 고유한 이름으로 복제하여 볼륨 가져오기 기능이 구분할 수 있도록 하십시오.

`azure-netapp-files` 드라이버를 사용하는 경우 `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` 명령을 사용하여 볼륨을 Kubernetes로 가져와 Trident에서 관리합니다. 이렇게 하면 고유한 볼륨 참조가 보장됩니다.

위 명령이 실행되면 Trident는 백엔드에서 볼륨을 찾아 크기를 읽습니다. 구성된 PVC의 볼륨 크기를 자동으로 추가하고 필요한 경우 덮어씁니다. 그런 다음 Trident는 새 PV를 생성하고 Kubernetes는 PVC를 PV에 바인딩합니다.

특정 가져온 PVC가 필요한 방식으로 컨테이너가 배포된 경우, 볼륨 가져오기 프로세스를 통해 PVC/PV 쌍이 바인딩될 때까지 컨테이너는 대기 상태로 유지됩니다. PVC/PV 쌍이 바인딩되면 다른 문제가 없는 한 컨테이너가 정상적으로 실행됩니다.

레지스트리 서비스

레지스트리용 스토리지 배포 및 관리에 대한 내용은 "netapp.io"의 "블로그"에 설명되어 있습니다.

로깅 서비스

다른 OpenShift 서비스와 마찬가지로 로깅 서비스는 플레이북에 제공되는 인벤토리 파일(일명 hosts)에 포함된 구성 매개변수를 사용하여 Ansible로 배포됩니다. 두 가지 설치 방법이 다루어집니다. 초기 OpenShift 설치 중 로깅 배포와 OpenShift 설치 후 로깅 배포입니다.

> Red Hat OpenShift 버전 3.9부터 공식 문서에서는 데이터 손상 문제로 인해 로깅 서비스에 NFS를 사용하지 않을 것을 권장합니다. 이는 Red Hat의 제품 테스트 결과를 기반으로 합니다. ONTAP NFS 서버에는 이러한 문제가 없으며 로깅 배포를 원활하게 지원할 수 있습니다. 궁극적으로 로깅 서비스에 사용할 프로토콜 선택은 사용자에게 달려 있지만, NetApp 플랫폼을 사용하는 경우 두 프로토콜 모두 훌륭하게 작동하며 NFS를 선호하는 경우 이를 피할 이유가 없습니다.

로깅 서비스에 NFS를 사용하려면 Ansible 변수 `openshift_enable_unsupported_configurations`를 `true`로 설정하여 설치 프로그램 오류를 방지해야 합니다.

시작하기

로깅 서비스는 선택적으로 애플리케이션뿐 아니라 OpenShift 클러스터 자체의 핵심 운영에도 배포할 수 있습니다. 변수 `openshift_logging_use_ops`를 `true`로 지정하여 운영 로깅을 배포하도록 선택하면 서비스 인스턴스가 두 개 생성됩니다. 운영 로깅 인스턴스를 제어하는 변수에는 "ops"가 포함되고, 애플리케이션용 인스턴스에는 포함되지 않습니다.

배포 방법에 따라 Ansible 변수를 구성하는 것은 기본 서비스에서 올바른 스토리지를 사용하도록 보장하는 데 중요합니다. 각 배포 방법에 대한 옵션을 살펴보겠습니다.

> 아래 표에는 로깅 서비스와 관련된 스토리지 구성에 필요한 변수만 포함되어 있습니다. "Red Hat OpenShift 로깅 문서"에서 배포 환경에 따라 검토, 구성 및 사용해야 하는 다른 옵션을 찾을 수 있습니다.

아래 표의 변수를 사용하면 Ansible 플레이북이 제공된 세부 정보를 사용하여 로깅 서비스용 PV 및 PVC를 생성합니다. 이 방법은 OpenShift 설치 후 구성 요소 설치 플레이북을 사용하는 것보다 유연성이 떨어지지만, 기존 볼륨이 있는 경우 하나의 옵션으로 사용할 수 있습니다.

변수	세부 정보
`openshift_logging_storage_kind`	`nfs`로 설정하여 설치 프로그램이 로깅 서비스를 위한 NFS PV를 생성하도록 합니다.
`openshift_logging_storage_host`	NFS 호스트의 호스트 이름 또는 IP 주소입니다. 이 값은 가상 머신의 dataLIF로 설정해야 합니다.
`openshift_logging_storage_nfs_directory`	NFS 내보내기의 마운트 경로입니다. 예를 들어 볼륨이 `/openshift_logging`로 접합된 경우 해당 경로를 이 변수에 사용합니다.
`openshift_logging_storage_volume_name`	생성할 PV의 이름(예: `pv_ose_logs`).
`openshift_logging_storage_volume_size`	예를 들어 NFS 내보내기의 크기 `100Gi`.

OpenShift 클러스터가 이미 실행 중이고 Trident가 배포 및 구성된 경우 설치 프로그램은 동적 프로비저닝을 사용하여 볼륨을 생성할 수 있습니다. 다음 변수를 구성해야 합니다.

변수	세부 정보
`openshift_logging_es_pvc_dynamic`	동적으로 프로비저닝된 볼륨을 사용하려면 true로 설정하십시오.
`openshift_logging_es_pvc_storage_class_name`	PVC에서 사용될 스토리지 클래스의 이름입니다.
`openshift_logging_es_pvc_size`	PVC에서 요청한 볼륨의 크기입니다.
`openshift_logging_es_pvc_prefix`	로깅 서비스에서 사용하는 PVC의 접두사입니다.
`openshift_logging_es_ops_pvc_dynamic`	`true`로 설정하여 운영 로깅 인스턴스에 동적으로 프로비저닝된 볼륨을 사용합니다.
`openshift_logging_es_ops_pvc_storage_class_name`	ops 로깅 인스턴스의 스토리지 클래스 이름입니다.
`openshift_logging_es_ops_pvc_size`	ops 인스턴스에 대한 볼륨 요청의 크기입니다.
`openshift_logging_es_ops_pvc_prefix`	ops 인스턴스 PVC의 접두사입니다.

로깅 스택을 배포합니다

초기 OpenShift 설치 과정의 일부로 로깅을 배포하는 경우 표준 배포 프로세스만 따르면 됩니다. Ansible은 필요한 서비스와 OpenShift 객체를 구성하고 배포하므로 Ansible 작업이 완료되는 즉시 서비스를 사용할 수 있습니다.

하지만 초기 설치 후 배포하는 경우에는 Ansible에서 컴포넌트 플레이북을 사용해야 합니다. 이 과정은 OpenShift 버전에 따라 약간 다를 수 있으므로 사용 중인 버전에 맞는 "Red Hat OpenShift Container Platform 3.11 문서"를 읽고 따르십시오.

메트릭 서비스

메트릭 서비스는 관리자에게 OpenShift 클러스터의 상태, 리소스 활용률 및 가용성에 대한 유용한 정보를 제공합니다. 또한 Pod 자동 스케일링 기능에 필수적이며, 많은 조직에서 메트릭 서비스의 데이터를 비용 청구 및/또는 사용량 보고 애플리케이션에 활용합니다.

로깅 서비스 및 OpenShift 전체와 마찬가지로 Ansible은 메트릭 서비스를 배포하는 데 사용됩니다. 또한 로깅 서비스와

마찬가지로 메트릭 서비스는 클러스터 초기 설정 시 또는 운영 후 구성 요소 설치 방법을 사용하여 배포할 수 있습니다. 다음 표에는 메트릭 서비스의 영구 스토리지를 구성할 때 중요한 변수가 나와 있습니다.

아래 표에는 메트릭 서비스와 관련된 스토리지 구성에 필요한 변수만 포함되어 있습니다. 문서에는 이 외에도 다양한 옵션이 있으므로 배포 환경에 맞게 검토, 구성 및 사용해야 합니다.

변수	세부 정보
`openshift_metrics_storage_kind`	`nfs`로 설정하여 설치 프로그램이 로깅 서비스를 위한 NFS PV를 생성하도록 합니다.
`openshift_metrics_storage_host`	NFS 호스트의 호스트 이름 또는 IP 주소입니다. 이 값은 SVM의 dataLIF로 설정해야 합니다.
`openshift_metrics_storage_nfs_directory`	NFS 내보내기의 마운트 경로입니다. 예를 들어 볼륨이 `/openshift_metrics`로 접합된 경우 해당 경로를 이 변수에 사용합니다.
`openshift_metrics_storage_volume_name`	생성할 PV의 이름(예: `pv_ose_metrics`).
`openshift_metrics_storage_volume_size`	예를 들어 NFS 내보내기의 크기 `100Gi`.

OpenShift 클러스터가 이미 실행 중이고 Trident가 배포 및 구성된 경우 설치 프로그램은 동적 프로비저닝을 사용하여 볼륨을 생성할 수 있습니다. 다음 변수를 구성해야 합니다.

변수	세부 정보
`openshift_metrics_cassandra_pvc_prefix`	메트릭 PVC에 사용할 접두사입니다.
`openshift_metrics_cassandra_pvc_size`	요청할 볼륨의 크기입니다.
`openshift_metrics_cassandra_storage_type`	메트릭에 사용할 스토리지 유형입니다. Ansible이 적절한 스토리지 클래스를 가진 PVC를 생성하려면 이 값을 dynamic으로 설정해야 합니다.
`openshift_metrics_cassanda_pvc_storage_class_name`	사용할 스토리지 클래스의 이름입니다.

메트릭 서비스 배포

hosts/inventory 파일에 적절한 Ansible 변수를 정의한 후 Ansible을 사용하여 서비스를 배포합니다. OpenShift 설치 시점에 배포하는 경우 PV가 자동으로 생성되어 사용됩니다. 컴포넌트 플레이북을 사용하여 OpenShift 설치 후 배포하는 경우 Ansible이 필요한 모든 PVC를 생성하고 Trident가 해당 PVC에 대한 스토리지를 프로비저닝한 후 서비스를 배포합니다.

위의 변수 및 배포 프로세스는 OpenShift의 각 버전에 따라 변경될 수 있습니다. 사용 중인 버전에 맞는 "Red Hat의 OpenShift 배포 가이드"을(를) 검토하고 따라 환경에 맞게 구성되도록 하십시오.

데이터 보호 및 재해 복구

Trident 및 Trident를 사용하여 생성된 볼륨에 대한 보호 및 복구 옵션에 대해 알아보십시오. 영구 저장 요구 사항이 있는 각 애플리케이션에 대해 데이터 보호 및 복구 전략을 수립해야 합니다.

Trident 복제 및 복구

재해 발생 시 Trident를 복원하기 위해 백업을 생성할 수 있습니다.

Trident 복제

Trident는 Kubernetes CRD를 사용하여 자체 상태를 저장 및 관리하고 Kubernetes 클러스터 etcd를 사용하여 메타데이터를 저장합니다.

단계

1. "Kubernetes: etcd 클러스터 백업"을 사용하여 Kubernetes 클러스터 etcd를 백업합니다.
2. 백업 아티팩트를 FlexVol 볼륨에 저장하세요.

> NetApp에서는 FlexVol이 있는 SVM을 SnapMirror 관계를 통해 다른 SVM으로 보호할 것을 권장합니다.

Trident 복구

Kubernetes CRD와 Kubernetes 클러스터 etcd 스냅샷을 사용하면 Trident를 복구할 수 있습니다.

단계

1. 타겟 SVM에서 Kubernetes etcd 데이터 파일과 인증서가 포함된 볼륨을 마스터 노드로 설정될 호스트에 마운트합니다.
2. Kubernetes 클러스터와 관련된 모든 필수 인증서를 `/etc/kubernetes/pki` 아래에 복사하고 etcd 멤버 파일을 `/var/lib/etcd` 아래에 복사하십시오.
3. "Kubernetes: etcd 클러스터 복원"을 사용하여 etcd 백업에서 Kubernetes 클러스터를 복원합니다.
4. `kubectl get crd`를 실행하여 모든 Trident 사용자 지정 리소스가 제대로 활성화되었는지 확인하고 Trident 객체를 검색하여 모든 데이터를 사용할 수 있는지 확인합니다.

SVM 복제 및 복구

Trident는 복제 관계를 구성할 수 없지만, 스토리지 관리자는 "ONTAP SnapMirror"를 사용하여 SVM을 복제할 수 있습니다.

재해가 발생할 경우 SnapMirror 타겟 SVM을 활성화하여 데이터 제공을 시작할 수 있습니다. 시스템 복구가 완료되면 운영 SVM으로 다시 전환할 수 있습니다.

이 작업 정보

SnapMirror SVM 복제 기능을 사용할 때 다음 사항을 고려하십시오.

- SVM-DR이 활성화된 각 SVM에 대해 별도의 백엔드를 생성해야 합니다.
- 필요한 경우에만 복제된 백엔드를 선택하도록 스토리지 클래스를 구성하여 SVM-DR을 지원하는 백엔드에 복제가 필요하지 않은 볼륨이 프로비저닝되지 않도록 하십시오.
- 애플리케이션 관리자는 복제와 관련된 추가 비용 및 복잡성을 이해하고 이 프로세스를 시작하기 전에 복구 계획을 신중하게 고려해야 합니다.

SVM 복제

"ONTAP: SnapMirror SVM 복제"를 사용하여 SVM 복제 관계를 생성할 수 있습니다.

SnapMirror를 사용하면 복제할 항목을 제어하는 옵션을 설정할 수 있습니다. Trident를 사용한 SVM 복구을 수행할 때 어떤 옵션을 선택했는지 알아야 합니다.

- "-identity-preserve true" 전체 SVM 구성을 복제합니다.
- "-discard-configs 네트워크" LIF 및 관련 네트워크 설정은 제외됩니다.
- "-identity-preserve false"볼륨과 보안 구성만 복제합니다.

Trident를 사용한 SVM 복구

Trident는 SVM 장애를 자동으로 감지하지 않습니다. 재해 발생 시 관리자는 수동으로 Trident 페일오버를 시작하여 새 SVM으로 전환할 수 있습니다.

단계

1. 예약 및 진행 중인 SnapMirror 전송을 취소하고, 복제 관계를 해제하고, 소스 SVM을 중지한 다음 SnapMirror 타겟 SVM을 활성화합니다.
2. SVM 복제를 구성할 때 `-identity-preserve false` 또는 `-discard-config network`를 지정한 경우 `Trident` 백엔드 정의 파일에서 `managementLIF` 및 `dataLIF`를 업데이트하십시오.
3. `storagePrefix`가 Trident 백엔드 정의 파일에 있는지 확인합니다. 이 매개변수는 변경할 수 없습니다. `storagePrefix`를 생략하면 백엔드 업데이트가 실패합니다.
4. 다음 명령어를 사용하여 필요한 모든 백엔드를 새 타겟 SVM 이름으로 업데이트하십시오.

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. `-identity-preserve false` 또는 `discard-config network`를 지정한 경우 모든 애플리케이션 Pod를 재시작해야 합니다.

> `-identity-preserve true`을 지정한 경우 타겟 SVM이 활성화되면 Trident에서 프로비저닝한 모든 볼륨이 데이터 제공을 시작합니다.

볼륨 복제 및 복구

Trident는 SnapMirror 복제 관계를 구성할 수 없지만 스토리지 관리자는 "ONTAP SnapMirror 복제 및 복구"를 사용하여 Trident에서 생성한 볼륨을 복제할 수 있습니다.

그런 다음 "tridentctl 볼륨 가져오기"을 사용하여 복구된 볼륨을 Trident로 가져올 수 있습니다.

> `ontap-nas-economy`, `ontap-san-economy` 또는 `ontap-flexgroup-economy` 드라이버에서는 가져오기가 지원되지 않습니다.

스냅샷 데이터 보호

다음 방법을 사용하여 데이터를 보호하고 복원할 수 있습니다.

- 영구 볼륨(PV)의 Kubernetes 볼륨 스냅샷을 생성하기 위한 외부 스냅샷 컨트롤러 및 CRD.

  "볼륨 스냅샷"

- ONTAP 스냅샷을 사용하여 볼륨의 전체 콘텐츠를 복원하거나 개별 파일 또는 LUN을 복구할 수 있습니다.

  "ONTAP 스냅샷"

Trident를 사용하여 상태 저장 애플리케이션의 페일오버 자동화

Trident의 강제 분리 기능은 Kubernetes 클러스터에서 비정상적인 노드로부터 볼륨을 자동으로 분리하여 데이터 손상을 방지하고 애플리케이션 가용성을 보장합니다. 이 기능은 노드가 응답하지 않거나 유지 보수를 위해 오프라인 상태가 되는 시나리오에서 특히 유용합니다.

강제 분리에 대한 세부 정보

강제 분리는 `ontap-san`, `ontap-san-economy`, `ontap-nas` 및 `ontap-nas-economy`에 대해서만 사용할 수 있습니다. 강제 분리를 활성화하기 전에 Kubernetes 클러스터에서 비정상 노드 종료(NGNS)를 활성화해야 합니다. NGNS는 Kubernetes 1.28 이상에서 기본적으로 활성화되어 있습니다. 자세한 내용은 "Kubernetes: 정상적이지 않은 노드 종료"을 참조하십시오.

> `ontap-nas` 또는 `ontap-nas-economy` 드라이버를 사용할 때는 백엔드 구성에서 `autoExportPolicy` 매개 변수를 `true`로 설정해야 Trident가 관리형 내보내기 정책을 사용하여 테인트가 적용된 Kubernetes 노드의 액세스를 제한할 수 있습니다.

> Trident는 Kubernetes NGNS에 의존하므로 모든 허용 불가능한 워크로드가 재배치될 때까지 비정상 노드에서 `out-of-service`테인트를 제거하지 마십시오. 테인트를 부적절하게 적용하거나 제거하면 백엔드 데이터 보호가 위험해질 수 있습니다.

Kubernetes 클러스터 관리자가 노드에 `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` 테인트를 적용하고 `enableForceDetach`가 `true`로 설정되면 Trident는 노드 상태를 확인하고 다음을 수행합니다.

1. 해당 노드에 마운트된 볼륨에 대한 백엔드 I/O 액세스를 중지합니다.
2. Trident 노드 객체를 `dirty`(새 게시에 안전하지 않음)로 표시합니다.

> Trident 컨트롤러는 Trident 노드 Pod에서 노드가 재인증(`dirty`로 표시된 후)될 때까지 새로운 볼륨 게시 요청을 거부합니다. 마운트된 `PVC`를 사용하여 예약된 모든 워크로드(클러스터 노드가 정상 상태이고 준비된 후에도)는 `Trident`가 노드 `clean`(새로운 게시를 위한 안전한 상태)를 검증할 때까지 허용되지 않습니다.

노드 상태가 복구되고 오염이 제거되면 Trident는 다음과 같이 동작합니다.

1. 노드에서 더 이상 사용되지 않는 게시된 경로를 식별하고 정리합니다.

2. 노드가 `cleanable`상태(서비스 중단 오류 표시가 제거되고 노드가 `Ready`상태인 경우)이고 모든 오래된 게시 경로가 정리되면 Trident는 해당 노드를 `clean`로 다시 승인하고 해당 노드에 새 게시 볼륨을 허용합니다.

자동 페일오버에 대한 세부 정보

"노드 상태 점검(NHC) 운영자"와의 통합을 통해 강제 분리 프로세스를 자동화할 수 있습니다. 노드 장애가 발생하면 NHC는 Trident 노드 복구(TNR)를 트리거하고 장애가 발생한 노드를 정의하는 TridentNodeRemediation CR을 Trident 네임스페이스에 생성하여 자동으로 강제 분리를 수행합니다. TNR은 노드 장애 발생 시에만 생성되며, 노드가 다시 온라인 상태가 되거나 삭제되면 NHC에서 제거됩니다.

실패한 노드 **Pod** 제거 프로세스

자동 장애 조치는 장애가 발생한 노드에서 제거할 워크로드를 선택합니다. TNR이 생성되면 TNR 컨트롤러는 해당 노드를 비정상 상태로 표시하여 새로운 볼륨 게시를 방지하고, 강제 분리가 지원되는 Pod와 해당 Pod의 볼륨 연결을 제거하기 시작합니다.

강제 분리에서 지원하는 모든 볼륨/PVC는 자동 페일오버에서 지원됩니다.

- 자동 내보내기 정책을 사용하는 NAS 및 NAS-economy 볼륨(SMB는 아직 지원되지 않음)
- SAN 및 SAN-economy 볼륨.

다음을 참조하십시오. 강제 분리에 대한 세부 정보.

기본 동작:

- force-detach에서 지원하는 볼륨을 사용하는 Pod는 장애가 발생한 노드에서 제거됩니다. Kubernetes는 이러한 Pod를 정상 노드로 다시 스케줄링합니다.
- 강제 분리 기능을 지원하지 않는 볼륨(Trident 이외의 볼륨 포함)을 사용하는 Pod는 장애가 발생한 노드에서 제거되지 않습니다.
- 스테이트리스 파드(PVC 제외)는 파드 어노테이션 `trident.netapp.io/podRemediationPolicy: delete`이 설정되어 있지 않으면 장애가 발생한 노드에서 제거되지 않습니다.

Pod 제거 동작 재정의:

Pod 제거 동작은 Pod 어노테이션을 사용하여 사용자 지정할 수 있습니다 `trident.netapp.io/podRemediationPolicy[retain, delete]`. 이러한 어노테이션은 페일오버 발생 시 검사되고 사용됩니다. 페일오버 후 어노테이션이 사라지지 않도록 하려면 Kubernetes 배포/복제 세트 Pod 사양에 어노테이션을 적용하세요:

- `retain` - 자동 장애 조치 중에 Pod는 장애가 발생한 노드에서 제거되지 않습니다.
- `delete` - 자동 장애 조치 중에 Pod는 장애가 발생한 노드에서 제거됩니다.

이러한 주석은 모든 포드에 적용할 수 있습니다.

- 강제 분리를 지원하는 볼륨의 경우 장애가 발생한 노드에서만 I/O 작업이 차단됩니다.
- 강제 분리를 지원하지 않는 볼륨의 경우 데이터 손상 및 다중 연결 문제가 발생할 위험이 있습니다.

TridentNodeRemediation CR

TridentNodeRemediation(TNR) CR은 장애가 발생한 노드를 정의합니다. TNR의 이름은 장애가 발생한 노드의 이름입니다.

TNR 예:

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

TNR 상태: 다음 명령을 사용하여 TNR 상태를 확인합니다.
`kubectl get tnr <name> -n <trident-namespace>`

TNR은 다음 상태 중 하나에 있을 수 있습니다.

- 복구 중:
  - 해당 노드에 마운트된 force-detach에서 지원하는 볼륨에 대한 백엔드 I/O 액세스를 중지합니다.
  - Trident 노드 객체가 더티(새 게시에 안전하지 않음)로 표시됩니다.
  - 노드에서 Pod 및 볼륨 연결을 제거합니다
- *NodeRecoveryPending*:
  - 컨트롤러가 노드가 다시 온라인 상태가 되기를 기다리고 있습니다.
  - 노드가 온라인 상태가 되면 publish-enforcement를 통해 노드가 깨끗하고 새로운 볼륨 게시를 위한 준비가 완료되었는지 확인합니다.
- 노드가 K8s에서 삭제되면 TNR 컨트롤러는 TNR을 제거하고 조정 작업을 중지합니다.
- 성공:
  - 모든 복구 및 노드 복구 단계가 성공적으로 완료되었습니다. 노드가 정상이며 새 볼륨 게시 준비가 되었습니다.
- 실패:
  - 복구 불가능한 오류입니다. 오류 원인은 CR의 status.message 필드에 설정되어 있습니다.

자동 페일오버 활성화

사전 요구 사항:

- 자동 장애 조치를 활성화하기 전에 강제 분리가 활성화되어 있는지 확인하십시오. 자세한 내용은 강제 분리에 대한 세부 정보을 참조하십시오.
- Kubernetes 클러스터에 노드 상태 확인(NHC)을 설치합니다.
  - "operator-sdk 설치".
  - 클러스터에 Operator Lifecycle Manager(OLM)가 설치되어 있지 않은 경우 설치하십시오 `operator-sdk olm install`.

- Node Health check Operator를 설치합니다 `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml`.

> 아래 [Integrating Custom Node Health Check Solutions] 섹션에 명시된 대로 노드 장애를 감지하는 다른 방법을 사용할 수도 있습니다.

자세한 내용은 "노드 상태 점검 오퍼레이터"을 참조하십시오.

단계

1. 클러스터의 워커 노드를 모니터링하기 위해 Trident 네임스페이스에 NodeHealthCheck(NHC) CR을 생성합니다. 예:

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
      duration: 0s
```

2. `trident` 네임스페이스에 노드 상태 점검 CR을 적용합니다.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

위의 CR은 노드 조건 Ready: false 및 Unknown에 대해 K8s 워커 노드를 감시하도록 구성되어 있습니다. 노드가 Ready: false 또는 Ready: Unknown 상태가 되면 Automated-Failover가 트리거됩니다.

CR의 `unhealthyConditions`는 0초 유예 기간을 사용합니다. 이로 인해 K8s가 노드에서 하트비트를 수신하지 못한 후 노드 조건을 Ready: false로 설정하면 즉시 자동 페일오버가 트리거됩니다. K8s는 기본적으로 마지막 하트비트 후

40초를 대기한 후 Ready: false로 설정합니다. 이 유예 기간은 K8s 배포 옵션에서 사용자 지정할 수 있습니다.

추가 구성 옵션은 "Node-Healthcheck-Operator 문서"을 참조하십시오.

추가 설정 정보

Trident가 강제 분리 기능을 활성화하여 설치되면 NHC와의 통합을 용이하게 하기 위해 Trident 네임스페이스에 두 개의 추가 리소스가 자동으로 생성됩니다: TridentNodeRemediationTemplate(TNRT) 및 ClusterRole.

TridentNodeRemediationTemplate(TNRT):

TNRT는 NHC 컨트롤러의 템플릿 역할을 하며, TNRT를 사용하여 필요에 따라 TNR 리소스를 생성합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

ClusterRole:

force-detach가 활성화되면 설치 중에 클러스터 역할이 추가됩니다. 이를 통해 NHC는 Trident 네임스페이스의 TNR에 대한 권한을 갖게 됩니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

K8s 클러스터 업그레이드 및 유지 관리

장애 조치를 방지하려면 노드가 다운되거나 재부팅될 것으로 예상되는 K8s 유지 관리 또는 업그레이드 중에 자동 장애 조치를 일시 중지하십시오. 위에서 설명한 NHC CR을 일시 중지하려면 해당 CR에 패치를 적용하면 됩니다.

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

이렇게 하면 자동 장애 조치가 일시 중지됩니다. 자동 장애 조치를 다시 활성화하려면 유지 관리가 완료된 후 사양에서 pauseRequests를 제거하십시오.

제한 사항

- I/O 작업은 force-detach에서 지원하는 볼륨에 대해 장애가 발생한 노드에서만 차단됩니다. force-detach에서 지원하는 볼륨/PVC를 사용하는 Pod만 자동으로 제거됩니다.
- 자동 페일오버 및 강제 분리 기능은 trident-controller pod 내부에서 실행됩니다. trident-controller를 호스팅하는 노드에 장애가 발생하면 K8s가 pod를 정상 노드로 이동할 때까지 자동 페일오버가 지연됩니다.

사용자 정의 노드 상태 확인 솔루션 통합

Node Healthcheck Operator를 대체 노드 장애 감지 도구로 교체하여 자동 장애 조치를 트리거할 수 있습니다. 자동 장애 조치 메커니즘과의 호환성을 보장하려면 사용자 지정 솔루션은 다음을 수행해야 합니다.

- 노드 장애가 감지되면 장애가 발생한 노드의 이름을 TNR CR 이름으로 사용하여 TNR을 생성합니다.
- 노드가 복구되고 TNR이 Succeeded 상태일 때 TNR을 삭제합니다.

보안

보안

여기에 나열된 권장 사항을 사용하여 Trident 설치의 보안을 확보하십시오.

Trident를 자체 네임스페이스에서 실행합니다

안정적인 스토리지를 보장하고 잠재적인 악의적 활동을 차단하려면 애플리케이션, 애플리케이션 관리자, 사용자 및 관리 애플리케이션이 Trident 객체 정의 또는 Pod에 액세스하는 것을 방지하는 것이 중요합니다.

다른 애플리케이션 및 사용자와 Trident를 분리하려면 항상 Trident를 자체 Kubernetes 네임스페이스에 설치하십시오(`trident`. Trident를 자체 네임스페이스에 배치하면 Kubernetes 관리 담당자만 Trident Pod 및 네임스페이스 CRD 객체에 저장된 아티팩트(해당하는 경우 백엔드 및 CHAP 암호)에 액세스할 수 있습니다. 관리자만 Trident 네임스페이스에 액세스하여 `tridentctl` 애플리케이션에 액세스할 수 있도록 해야 합니다.

ONTAP SAN 백엔드에서 **CHAP** 인증을 사용하십시오

Trident는 ONTAP SAN 워크로드에 대해 CHAP 기반 인증을 지원합니다(`ontap-san` 및 `ontap-san-economy` 드라이버 사용). NetApp은 호스트와 스토리지 백엔드 간의 인증을 위해 Trident와 함께 양방향 CHAP를 사용하는 것을 권장합니다.

SAN 스토리지 드라이버를 사용하는 ONTAP 백엔드의 경우 Trident는 `tridentctl`를 통해 양방향 CHAP를 설정하고 CHAP 사용자 이름과 암호를 관리할 수 있습니다. Trident가 ONTAP 백엔드에서 CHAP를 구성하는 방법을 이해하려면"ONTAP SAN 드라이버를 사용하여 백엔드를 구성할 준비를 하십시오"를 참조하십시오.

NetApp HCI 및 **SolidFire** 백엔드에서 **CHAP** 인증을 사용하십시오

NetApp에서는 호스트와 NetApp HCI 및 SolidFire 백엔드 간의 인증을 보장하기 위해 양방향 CHAP 배포를 권장합니다. Trident는 테넌트당 두 개의 CHAP 암호를 포함하는 secret 객체를 사용합니다. Trident가 설치되면 CHAP secret을 관리하고 해당 PV에 대한 `tridentvolume` CR 객체에 저장합니다. PV를 생성하면 Trident는 CHAP secret을 사용하여 iSCSI 세션을 시작하고 CHAP를 통해 NetApp HCI 및 SolidFire 시스템과 통신합니다.

Trident에서 생성한 볼륨은 어떤 볼륨 액세스 그룹과도 연결되지 않습니다.

NVE 및 **NAE**와 함께 **Trident** 사용

NetApp ONTAP는 디스크가 도난당하거나 반환되거나 용도가 변경될 경우 중요한 데이터를 보호하기 위해 저장 데이터 암호화를 제공합니다. 자세한 내용은 "NetApp Volume Encryption 구성 개요"를 참조하십시오.

- 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다.
  - NVE encryption 플래그를 `""`로 설정하여 NAE 지원 볼륨을 생성할 수 있습니다.
- 백엔드에서 NAE가 활성화되지 않은 경우 백엔드 구성에서 NVE 암호화 플래그가 `false`(기본값)로 설정되지 않는 한 Trident에서 프로비저닝된 모든 볼륨은 NVE가 활성화됩니다.

NAE가 활성화된 백엔드에서 Trident로 생성된 볼륨은 NVE 또는 NAE로 암호화되어야 합니다.

- Trident 백엔드 구성에서 NVE 암호화 플래그를 `true`로 설정하여 NAE 암호화를 재정의하고 볼륨별로 특정 암호화 키를 사용할 수 있습니다.
- NAE가 활성화된 백엔드에서 NVE 암호화 플래그를 `false`로 설정하면 NAE가 활성화된 볼륨이 생성됩니다. NVE 암호화 플래그를 `false`로 설정하여 NAE 암호화를 비활성화할 수는 없습니다.

- Trident에서 NVE 암호화 플래그를 명시적으로 `true`로 설정하여 수동으로 NVE 볼륨을 생성할 수 있습니다.

백엔드 구성 옵션에 대한 자세한 내용은 다음을 참조하십시오.

- "ONTAP SAN 구성 옵션"
- "ONTAP NAS 구성 옵션"

Linux Unified Key Setup(LUKS)

Trident에서 Linux Unified Key Setup(LUKS)을 활성화하여 ONTAP SAN 및 ONTAP SAN ECONOMY 볼륨을 암호화할 수 있습니다. Trident는 LUKS로 암호화된 볼륨에 대해 암호 순환 및 볼륨 확장을 지원합니다.

Trident에서 LUKS로 암호화된 볼륨은 "NIST"에서 권장하는 대로 aes-xts-plain64 암호 및 모드를 사용합니다.

LUKS 암호화는 ASA r2 시스템에서 지원되지 않습니다. ASA r2 시스템에 대한 자세한 내용은 "ASA r2 스토리지 시스템에 대해 알아보세요"를 참조하십시오.

시작하기 전에

- 워커 노드에는 cryptsetup 2.1 이상(단, 3.0 미만)이 설치되어 있어야 합니다. 자세한 내용은 "Gitlab: cryptsetup"를 참조하십시오.
- 성능 향상을 위해 NetApp에서는 워커 노드가 AES-NI(Advanced Encryption Standard New Instructions)를 지원하도록 권장합니다. AES-NI 지원 여부를 확인하려면 다음 명령을 실행하십시오.

```
grep "aes" /proc/cpuinfo
```

아무것도 반환되지 않으면 프로세서가 AES-NI를 지원하지 않는 것입니다. AES-NI에 대한 자세한 내용은 "Intel: Advanced Encryption Standard Instructions(AES-NI)"를 참조하십시오.

LUKS 암호화 활성화

ONTAP SAN 및 ONTAP SAN ECONOMY 볼륨의 경우 Linux Unified Key Setup(LUKS)을 사용하여 볼륨별 호스트 측 암호화를 활성화할 수 있습니다.

단계

1. 백엔드 구성에서 LUKS 암호화 속성을 정의합니다. ONTAP SAN의 백엔드 구성 옵션에 대한 자세한 내용은 "ONTAP SAN 구성 옵션"을 참조하십시오.

```
{
  "storage": [
    {
      "labels": {
        "luks": "true"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "true"
      }
    },
    {
      "labels": {
        "luks": "false"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "false"
      }
    }
  ]
}
```

2. `parameters.selector`를 사용하여 LUKS 암호화를 사용하는 스토리지 풀을 정의합니다. 예를 들면 다음과 같습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

3. LUKS 암호를 포함하는 비밀 키를 생성합니다. 예를 들면 다음과 같습니다.

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

제한 사항

LUKS로 암호화된 볼륨은 ONTAP 중복 제거 및 압축을 활용할 수 없습니다.

LUKS 볼륨 가져오기를 위한 백엔드 구성

LUKS 볼륨을 가져오려면 백엔드에서 `luksEncryption`을 (`true`로 설정해야 합니다. `luksEncryption` 옵션은 다음 예와 같이 볼륨이 LUKS 규격을 준수하는지(`true` 또는 LUKS 규격을 준수하지 않는지(`false`를 Trident에 알려줍니다.

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

LUKS 볼륨 가져오기를 위한 **PVC** 구성

LUKS 볼륨을 동적으로 가져오려면 annotation `trident.netapp.io/luksEncryption`을 `true`로 설정하고 이 예제와 같이 PVC에 LUKS 지원 스토리지 클래스를 포함하십시오.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

LUKS 암호 교체

LUKS 암호를 교체하고 교체를 확인할 수 있습니다.

> 볼륨, 스냅샷 또는 시크릿에서 더 이상 참조되지 않는다는 것을 확인하기 전까지는 암호를 잊지 마십시오. 참조된 암호를 분실하면 볼륨을 마운트할 수 없으며 데이터는 암호화된 상태로 유지되어 액세스할 수 없게 됩니다.

이 작업 정보

LUKS 암호 순환은 새 LUKS 암호가 지정된 후 해당 볼륨을 마운트하는 Pod가 생성될 때 발생합니다. 새 Pod가 생성될 때 Trident는 볼륨의 LUKS 암호를 시크릿에 있는 활성 암호와 비교합니다.

- 볼륨의 암호가 비밀의 활성 암호와 일치하지 않으면 순환이 발생합니다.
- 볼륨의 암호가 비밀 키에 있는 활성 암호와 일치하면 `previous-luks-passphrase` 매개변수는 무시됩니다.

단계

1. `node-publish-secret-name` 및 `node-publish-secret-namespace` StorageClass 매개변수를 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. 볼륨 또는 스냅샷에 있는 기존 암호를 확인합니다.

볼륨

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

스냅샷

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["A"]
```

3. 볼륨의 LUKS 암호를 업데이트하여 새 암호와 이전 암호를 지정하십시오. `previous-luke-passphrase-name`와 `previous-luks-passphrase`가 이전 암호와 일치하는지 확인하십시오.

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. 볼륨을 마운트하는 새 Pod를 생성합니다. 이는 로테이션을 시작하는 데 필요합니다.
5. 암호가 교체되었는지 확인하십시오.

볼륨

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

스냅샷

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

결과

볼륨 및 스냅샷에 새 암호만 반환되면 암호가 교체된 것입니다.

> 예를 들어 두 개의 암호문이 반환되면 `luksPassphraseNames: ["B", "A"]` 로테이션이 완료되지 않은 것입니다. 로테이션을 완료하기 위해 새 Pod를 트리거할 수 있습니다.

볼륨 확장 활성화

LUKS로 암호화된 볼륨에서 볼륨 확장을 활성화할 수 있습니다.

단계

1. `CSINodeExpandSecret` 기능 게이트를 활성화합니다(베타 1.25+). 자세한 내용은 "Kubernetes 1.25: 노드 기반 CSI 볼륨 확장을 위한 시크릿 사용"를 참조하십시오.
2. `node-expand-secret-name` 및 `node-expand-secret-namespace` StorageClass 매개변수를 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

결과

온라인 스토리지 확장을 시작하면 kubelet이 적절한 자격 증명을 드라이버에 전달합니다.

Kerberos 전송 중 암호화

Kerberos 전송 중 암호화를 사용하면 관리형 클러스터와 스토리지 백엔드 간의 트래픽에 대한 암호화를 활성화하여 데이터 액세스 보안을 개선할 수 있습니다.

Trident는 스토리지 백엔드로 ONTAP에 대한 Kerberos 암호화를 지원합니다.

- 온프레미스 **ONTAP** - Trident는 Red Hat OpenShift 및 업스트림 Kubernetes 클러스터에서 온프레미스 ONTAP 볼륨으로의 NFSv3 및 NFSv4 연결을 통해 Kerberos 암호화를 지원합니다.

NFS 암호화를 사용하는 볼륨을 생성, 삭제, 크기 조정, 스냅샷, 클론 복제, 읽기 전용 클론 복제 및 가져오기를 수행할 수 있습니다.

온프레미스 **ONTAP** 볼륨을 사용하여 전송 중 **Kerberos** 암호화 구성

관리형 클러스터와 온프레미스 ONTAP 스토리지 백엔드 간의 스토리지 트래픽에 Kerberos 암호화를 활성화할 수 있습니다.

> 온프레미스 ONTAP 스토리지 백엔드를 사용하는 NFS 트래픽에 대한 Kerberos 암호화는 `ontap-nas` 스토리지 드라이버를 통해서만 지원됩니다.

시작하기 전에

- `tridentctl` 유틸리티에 액세스할 수 있는지 확인하십시오.
- ONTAP 스토리지 백엔드에 대한 관리자 액세스 권한이 있는지 확인하십시오.
- ONTAP 스토리지 백엔드에서 공유할 볼륨의 이름을 알고 있는지 확인하십시오.
- NFS 볼륨에 대한 Kerberos 암호화를 지원하도록 ONTAP 스토리지 VM을 준비했는지 확인하십시오. 지침은 "dataLIF에서 Kerberos를 활성화합니다"을 참조하십시오.
- Kerberos 암호화와 함께 사용하는 모든 NFSv4 볼륨이 올바르게 구성되었는지 확인하십시오. "NetApp NFSv4 향상 기능 및 모범 사례 가이드"의 NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하십시오.

ONTAP 내보내기 정책을 추가 또는 수정합니다

기존 ONTAP 내보내기 정책에 규칙을 추가하거나 ONTAP 스토리지 VM 루트 볼륨과 업스트림 Kubernetes 클러스터와 공유되는 모든 ONTAP 볼륨에 대해 Kerberos 암호화를 지원하는 새 내보내기 정책을 생성해야 합니다. 추가하는 내보내기 정책 규칙 또는 새로 생성하는 내보내기 정책은 다음 액세스 프로토콜 및 액세스 권한을 지원해야 합니다.

액세스 프로토콜

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

액세스 세부 정보

볼륨에 대한 요구 사항에 따라 세 가지 버전의 Kerberos 암호화 중 하나를 구성할 수 있습니다.

- **Kerberos 5** - (인증 및 암호화)
- **Kerberos 5i** - (ID 보호 기능을 갖춘 인증 및 암호화)

- **Kerberos 5p** - (신원 및 개인정보 보호를 통한 인증 및 암호화)

적절한 액세스 권한으로 ONTAP 엑스포트 정책 규칙을 구성합니다. 예를 들어, 클러스터가 Kerberos 5i와 Kerberos 5p 암호화를 혼합하여 NFS 볼륨을 마운트하는 경우 다음 액세스 설정을 사용하십시오.

유형	읽기 전용 액세스	읽기/쓰기 액세스	슈퍼유저 액세스
UNIX	활성화됨	활성화됨	활성화됨
Kerberos 5i	활성화됨	활성화됨	활성화됨
Kerberos 5p	활성화됨	활성화됨	활성화됨

ONTAP 내보내기 정책 및 내보내기 정책 규칙을 생성하는 방법은 다음 문서를 참조하십시오.

- "엑스포트 정책을 생성합니다"
- "엑스포트 정책에 규칙 추가"

스토리지 백엔드를 생성합니다

Kerberos 암호화 기능을 포함하는 Trident 스토리지 백엔드 구성을 생성할 수 있습니다.

이 작업 정보

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 생성할 때 `spec.nfsMountOptions` 매개변수를 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다.

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

Kerberos 수준을 하나만 지정하십시오. 매개변수 목록에 두 개 이상의 Kerberos 암호화 수준을 지정하면 첫 번째 옵션만 사용됩니다.

단계

1. 관리형 클러스터에서 다음 예제를 사용하여 스토리지 백엔드 구성 파일을 생성합니다. 괄호 <> 안의 값은 사용자 환경에 맞는 정보로 바꾸십시오.

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

2. 이전 단계에서 생성한 구성 파일을 사용하여 백엔드를 생성하십시오.

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

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

```
tridentctl logs
```

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

스토리지 클래스를 생성합니다

Kerberos 암호화를 사용하여 볼륨을 프로비저닝하는 스토리지 클래스를 생성할 수 있습니다.

이 작업 정보

스토리지 클래스 객체를 생성할 때 `mountOptions` 매개변수를 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다.

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

Kerberos 수준을 하나만 지정하십시오. 매개변수 목록에 두 개 이상의 Kerberos 암호화 수준을 지정하면 첫 번째 옵션만 사용됩니다. 스토리지 백엔드 구성에서 지정한 암호화 수준이 스토리지 클래스 객체에서 지정한 수준과 다른 경우 스토리지 클래스 객체가 우선합니다.

단계

1. 다음 예제를 사용하여 StorageClass Kubernetes 객체를 생성합니다.

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

2. 스토리지 클래스를 생성합니다.

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

3. 스토리지 클래스가 생성되었는지 확인하십시오.

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

다음과 유사한 출력이 표시됩니다.

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

볼륨 프로비저닝

스토리지 백엔드와 스토리지 클래스를 생성한 후에는 볼륨을 프로비저닝할 수 있습니다. 자세한 내용은 "볼륨 프로비저닝"를 참조하십시오.

Azure NetApp Files 볼륨에서 전송 중 Kerberos 암호화 구성

관리형 클러스터와 단일 Azure NetApp Files 스토리지 백엔드 또는 Azure NetApp Files 스토리지 백엔드의 가상 풀 간의 스토리지 트래픽에 대해 Kerberos 암호화를 활성화할 수 있습니다.

시작하기 전에

- 관리형 Red Hat OpenShift 클러스터에서 Trident가 활성화되어 있는지 확인하십시오.
- `tridentctl` 유틸리티에 액세스할 수 있는지 확인하십시오.
- 요구 사항을 확인하고 "Azure NetApp Files 설명서"의 지침에 따라 Kerberos 암호화를 위한 Azure NetApp Files 스토리지 백엔드를 준비했는지 확인하십시오.
- Kerberos 암호화와 함께 사용하는 모든 NFSv4 볼륨이 올바르게 구성되었는지 확인하십시오. "NetApp NFSv4 향상 기능 및 모범 사례 가이드"의 NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하십시오.

스토리지 백엔드를 생성합니다

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

이 작업 정보

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

- `spec.kerberos` 필드를 사용하는 스토리지 백엔드 레벨
- `spec.storage.kerberos` 필드를 사용한 가상 풀 레벨

가상 풀 수준에서 구성을 정의할 때 스토리지 클래스의 레이블을 사용하여 풀이 선택됩니다.

어느 수준에서든 세 가지 버전의 Kerberos 암호화 중 하나를 지정할 수 있습니다.

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

단계

1. 관리형 클러스터에서 스토리지 백엔드를 정의해야 하는 위치(스토리지 백엔드 수준 또는 가상 풀 수준)에 따라 다음 예제 중 하나를 사용하여 스토리지 백엔드 구성 파일을 생성합니다. 괄호 <> 안의 값은 사용자 환경에 맞는 정보로 바꾸십시오.

스토리지 백엔드 수준 예

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

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

가상 풀 수준 예

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

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

2. 이전 단계에서 생성한 구성 파일을 사용하여 백엔드를 생성하십시오.

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

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

```
tridentctl logs
```

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

스토리지 클래스를 생성합니다

Kerberos 암호화를 사용하여 볼륨을 프로비저닝하는 스토리지 클래스를 생성할 수 있습니다.

단계

1. 다음 예제를 사용하여 StorageClass Kubernetes 객체를 생성합니다.

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

2. 스토리지 클래스를 생성합니다.

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

3. 스토리지 클래스가 생성되었는지 확인하십시오.

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

다음과 유사한 출력이 표시됩니다.

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

볼륨 프로비저닝

스토리지 백엔드와 스토리지 클래스를 생성한 후에는 볼륨을 프로비저닝할 수 있습니다. 자세한 내용은 "볼륨 프로비저닝"를 참조하십시오.