보안
Trident

NetApp
January 15, 2026

목차

보안 . . . 1
- 보안 . . . 1
  - Trident 자체 네임스페이스에서 실행 . . . 1
  - ONTAP SAN 백엔드에서 CHAP 인증 사용 . . . 1
  - NetApp HCI 및 SolidFire 백엔드에서 CHAP 인증 사용 . . . 1
  - NVE 및 NAE와 함께 Trident 사용 . . . 1
- Linux 통합 키 설정(LUKS) . . . 2
  - LUKS 암호화 활성화 . . . 2
  - LUKS 볼륨 가져오기를 위한 백엔드 구성 . . . 4
  - LUKS 볼륨 가져오기를 위한 PVC 구성 . . . 4
  - LUKS 암호 문구 회전 . . . 5
  - 볼륨 확장 활성화 . . . 7
- Kerberos 비행 중 암호화 . . . 7
  - 온프레미스 ONTAP 볼륨을 사용하여 비행 중 Kerberos 암호화 구성 . . . 8
  - Azure NetApp Files 볼륨을 사용하여 진행 중인 Kerberos 암호화 구성 . . . 12

보안

보안

여기에 나열된 권장 사항을 사용하여 Trident 설치가 안전한지 확인하세요.

Trident 자체 네임스페이스에서 실행

안정적인 저장소를 보장하고 잠재적인 악성 활동을 차단하려면 애플리케이션, 애플리케이션 관리자, 사용자 및 관리 애플리케이션이 Trident 객체 정의나 포드에 액세스하지 못하도록 하는 것이 중요합니다.

다른 애플리케이션과 사용자를 Trident 에서 분리하려면 항상 Trident 자체 Kubernetes 네임스페이스에 설치하십시오 .(`trident`). Trident 자체 네임스페이스에 배치하면 Kubernetes 관리 담당자만 Trident 포드와 네임스페이스 CRD 객체에 저장된 아티팩트(해당되는 경우 백엔드 및 CHAP 비밀 등)에 액세스할 수 있습니다. 관리자만 Trident 네임스페이스에 액세스할 수 있도록 허용해야 하며 이를 통해 액세스할 수 있습니다. `tridentctl` 애플리케이션.

ONTAP SAN 백엔드에서 CHAP 인증 사용

Trident ONTAP SAN 워크로드에 대한 CHAP 기반 인증을 지원합니다(사용 `ontap-san` 그리고 `ontap-san-economy` 운전자). NetApp 호스트와 스토리지 백엔드 간 인증을 위해 Trident 와 함께 양방향 CHAP를 사용할 것을 권장합니다.

SAN 스토리지 드라이버를 사용하는 ONTAP 백엔드의 경우 Trident 양방향 CHAP를 설정하고 CHAP 사용자 이름과 비밀번호를 관리할 수 있습니다. `tridentctl` . 참조하다"ONTAP SAN 드라이버로 백엔드 구성을 준비합니다." Trident ONTAP 백엔드에서 CHAP를 구성하는 방법을 이해합니다.

NetApp HCI 및 SolidFire 백엔드에서 CHAP 인증 사용

NetApp 호스트와 NetApp HCI 및 SolidFire 백엔드 간의 인증을 보장하기 위해 양방향 CHAP를 배포할 것을 권장합니다. Trident 테넌트당 두 개의 CHAP 암호가 포함된 비밀 객체를 사용합니다. Trident 가 설치되면 CHAP 비밀을 관리하고 저장합니다. `tridentvolume` 해당 PV에 대한 CR 객체입니다. PV를 생성하면 Trident CHAP 비밀을 사용하여 iSCSI 세션을 시작하고 CHAP를 통해 NetApp HCI 및 SolidFire 시스템과 통신합니다.

Trident 에서 생성된 볼륨은 어떤 볼륨 액세스 그룹과도 연관되지 않습니다.

NVE 및 NAE와 함께 Trident 사용

NetApp ONTAP 디스크가 도난당하거나 반환되거나 다른 용도로 사용되는 경우 중요한 데이터를 보호하기 위해 저장 데이터 암호화를 제공합니다. 자세한 내용은 다음을 참조하세요."NetApp 볼륨 암호화 구성 개요" .

- 백엔드에서 NAE가 활성화된 경우 Trident 에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다.
  - NVE 암호화 플래그를 설정할 수 있습니다. "" NAE 지원 볼륨을 생성합니다.
- 백엔드에서 NAE가 활성화되지 않은 경우 NVE 암호화 플래그가 설정되지 않는 한 Trident 에서 프로비저닝된 모든 볼륨은 NVE가 활성화됩니다. `false` (기본값) 백엔드 구성에서.

NAE 지원 백엔드의 Trident 에서 생성된 볼륨은 NVE 또는 NAE로 암호화되어야 합니다.

- NVE 암호화 플래그를 설정할 수 있습니다. `true` Trident 백엔드 구성에서 NAE 암호화를 재정의하고 볼륨별로 특정 암호화 키를 사용합니다.
- NVE 암호화 플래그를 다음으로 설정 `false` NAE 지원 백엔드에서는 NAE 지원 볼륨을 생성합니다. NVE 암호화 플래그를 설정하여 NAE 암호화를 비활성화할 수 없습니다. `false` .

- NVE 암호화 플래그를 명시적으로 설정하여 Trident 에서 NVE 볼륨을 수동으로 생성할 수 있습니다. `true` .

백엔드 구성 옵션에 대한 자세한 내용은 다음을 참조하세요.

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

Linux 통합 키 설정(LUKS)

Trident 에서 ONTAP SAN 및 ONTAP SAN ECONOMY 볼륨을 암호화하기 위해 Linux Unified Key Setup(LUKS)을 활성화할 수 있습니다. Trident LUKS로 암호화된 볼륨에 대한 암호 문구 순환과 볼륨 확장을 지원합니다.

Trident 에서 LUKS 암호화 볼륨은 권장하는 대로 aes-xts-plain64 암호 및 모드를 사용합니다."미국 국립표준기술원(NIST)" .

ASA r2 시스템에서는 LUKS 암호화가 지원되지 않습니다. ASA r2 시스템에 대한 정보는 다음을 참조하세요."ASA r2 스토리지 시스템에 대해 알아보세요" .

시작하기 전에

- 작업자 노드에는 cryptsetup 2.1 이상(3.0 미만)이 설치되어 있어야 합니다. 자세한 내용은 다음을 방문하세요 ."Gitlab: cryptsetup" .
- 성능상의 이유로 NetApp 작업자 노드가 AES-NI(Advanced Encryption Standard New Instructions)를 지원할 것을 권장합니다. AES-NI 지원을 확인하려면 다음 명령을 실행하세요.

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

아무것도 반환되지 않으면 프로세서가 AES-NI를 지원하지 않는 것입니다. AES-NI에 대한 자세한 내용은 다음을 방문하세요."Intel: 고급 암호화 표준 명령어(AES-NI)" .

LUKS 암호화 활성화

ONTAP SAN 및 ONTAP SAN ECONOMY 볼륨에 대해 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 다음을 참조하세요."ONTAP SAN 구성 옵션" .

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

2. 사용 `parameters.selector` LUKS 암호화를 사용하여 스토리지 풀을 정의합니다. 예를 들어:

```
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 암호문구를 포함하는 비밀번호를 생성합니다. 예를 들어:

```
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

LUKS 암호화 볼륨은 ONTAP 중복 제거 및 압축 기능을 활용할 수 없습니다.

LUKS 볼륨 가져오기를 위한 백엔드 구성

LUKS 볼륨을 가져오려면 다음을 설정해야 합니다. `luksEncryption` 에게(`true` 백엔드에서. 그만큼 `luksEncryption` 옵션은 볼륨이 LUKS 규격인지 Trident 알려줍니다.(`true`) 또는 LUKS 규격에 맞지 않음(`false`) 다음 예와 같습니다.

```
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 주석을 설정하세요. `trident.netapp.io/luksEncryption` 에게 `true` 이 예에서 보여지는 것처럼 PVC에 LUKS 지원 스토리지 클래스를 포함합니다.

```
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 문구 회전

LUKS 암호를 교체하고 교체를 확인할 수 있습니다.

> 볼륨, 스냅샷 또는 비밀에서 더 이상 참조되지 않는다는 것을 확인할 때까지 암호를 잊지 마세요. 참조된 암호문구가 손실되면 볼륨을 마운트할 수 없고 데이터는 암호화되어 액세스할 수 없게 됩니다.

이 작업에 관하여

LUKS 암호 문구 순환은 새로운 LUKS 암호 문구가 지정된 후 볼륨을 마운트하는 포드가 생성될 때 발생합니다. 새로운 포드가 생성되면 Trident 볼륨의 LUKS 암호를 비밀의 활성 암호와 비교합니다.

- 볼륨의 암호가 비밀의 활성 암호와 일치하지 않으면 회전이 발생합니다.
- 볼륨의 암호가 비밀의 활성 암호와 일치하는 경우 `previous-luks-passphrase` 매개변수는 무시됩니다.

단계

1. 추가하세요 `node-publish-secret-name` 그리고 `node-publish-secret-namespace` StorageClass 매개변수. 예를 들어:

```
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. 볼륨이나 스냅샷에 있는 기존 암호를 식별합니다.

   용량

```
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 비밀번호를 업데이트하여 새 암호와 이전 암호구를 지정합니다. 보장하다 `previous-luke-passphrase-name` 그리고 `previous-luks-passphrase` 이전 암호문구와 일치합니다.

```
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 마운트하여 새로운 포드를 만듭니다. 이는 회전을 시작하는 데 필요합니다.
5. 암호가 회전되었는지 확인하세요.

   용량

```
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

볼륨과 스냅샷에 새로운 암호문구만 반환되면 암호문구가 회전되었습니다.

> 예를 들어 두 개의 암호가 반환되는 경우 `luksPassphraseNames: ["B", "A"]` , 회전이 완료되지 않았습니다. 새로운 포드를 작동시켜 회전을 완료할 수 있습니다.

볼륨 확장 활성화

LUKS로 암호화된 볼륨에서 볼륨 확장을 활성화할 수 있습니다.

단계

1. 활성화 `CSINodeExpandSecret` 기능 게이트(베타 1.25+). 참조하다 "Kubernetes 1.25: CSI 볼륨의 노드 기반 확장을 위한 비밀 사용" 자세한 내용은.
2. 추가하세요 `node-expand-secret-name` 그리고 `node-expand-secret-namespace` StorageClass 매개변수. 예를 들어:

```
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 kubelet은 드라이버에 적절한 자격 증명을 전달합니다.

Kerberos 비행 중 암호화

Kerberos 전송 중 암호화를 사용하면 관리되는 클러스터와 스토리지 백엔드 간 트래픽에 대한 암호화를 활성화하여 데이터 액세스 보안을 강화할 수 있습니다.

Trident ONTAP 스토리지 백엔드로 사용할 때 Kerberos 암호화를 지원합니다.

- 온프레미스 **ONTAP** - Trident 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, 복제, 읽기 전용 복제 및 가져오기할 수 있습니다.

온프레미스 **ONTAP** 볼륨을 사용하여 비행 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 다음을 통해서만 지원됩니다. `ontap-nas` 저장 드라이버.

시작하기 전에

- 귀하가 다음에 액세스할 수 있는지 확인하십시오. `tridentctl` 공익사업.
- ONTAP 스토리지 백엔드에 대한 관리자 액세스 권한이 있는지 확인하세요.
- ONTAP 스토리지 백엔드에서 공유할 볼륨의 이름을 알고 있는지 확인하세요.
- NFS 볼륨에 대한 Kerberos 암호화를 지원하도록 ONTAP 스토리지 VM을 준비했는지 확인하세요. 참조하다 "dataLIF에서 Kerberos 활성화" 지침을 보려면.
- Kerberos 암호화와 함께 사용하는 모든 NFSv4 볼륨이 올바르게 구성되었는지 확인하세요. NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하세요. "NetApp NFSv4 개선 사항 및 모범 사례 가이드" .

ONTAP 내보내기 정책 추가 또는 수정

기존 ONTAP 내보내기 정책에 규칙을 추가하거나 ONTAP 스토리지 VM 루트 볼륨과 업스트림 Kubernetes 클러스터와 공유되는 모든 ONTAP 볼륨에 대한 Kerberos 암호화를 지원하는 새로운 내보내기 정책을 만들어야 합니다. 추가하는 내보내기 정책 규칙이나 만드는 새로운 내보내기 정책은 다음과 같은 액세스 프로토콜과 액세스 권한을 지원해야 합니다.

접근 프로토콜

NFS, NFSv3, NFSv4 액세스 프로토콜을 사용하여 내보내기 정책을 구성합니다.

접근 세부 정보

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

- **Kerberos 5** - (인증 및 암호화)
- **Kerberos 5i** - (신원 보호 기능이 있는 인증 및 암호화)
- **Kerberos 5p** - (신원 및 개인 정보 보호를 통한 인증 및 암호화)

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

유형	읽기 전용 액세스	읽기/쓰기 액세스	슈퍼유저 접근
유닉스	활성화됨	활성화됨	활성화됨

유형	읽기 전용 액세스	읽기/쓰기 액세스	슈퍼유저 접근
케르베로스 5i	활성화됨	활성화됨	활성화됨
케르베로스 5p	활성화됨	활성화됨	활성화됨

ONTAP 내보내기 정책과 내보내기 정책 규칙을 만드는 방법에 대한 자세한 내용은 다음 문서를 참조하세요.

- "수출 정책 만들기"
- "내보내기 정책에 규칙 추가"

스토리지 백엔드 생성

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

이 작업에 관하여

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

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

Kerberos 수준을 하나만 지정하세요. 매개변수 목록에서 두 개 이상의 Kerberos 암호화 수준을 지정하는 경우 첫 번째 옵션만 사용됩니다.

단계

1. 관리되는 클러스터에서 다음 예를 사용하여 스토리지 백엔드 구성 파일을 만듭니다. <> 괄호 안의 값을 사용자 환경의 정보로 바꾸세요.

```
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 만든 구성 파일을 사용하여 백엔드를 만듭니다.

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

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 파악하고 수정한 후에는 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스 생성

Kerberos 암호화를 사용하여 볼륨을 프로비저닝하기 위해 스토리지 클래스를 생성할 수 있습니다.

이 작업에 관하여

저장소 클래스 객체를 생성할 때 다음을 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다. `mountOptions` 매개변수:

- `mountOptions: sec=krb5`(인증 및 암호화)
- `mountOptions: sec=krb5i`(신원 보호 기능이 있는 인증 및 암호화)
- `mountOptions: sec=krb5p`(신원 및 개인 정보 보호를 통한 인증 및 암호화)

Kerberos 수준을 하나만 지정하세요. 매개변수 목록에서 두 개 이상의 Kerberos 암호화 수준을 지정하는 경우 첫 번째 옵션만 사용됩니다. 스토리지 백엔드 구성에서 지정한 암호화 수준이 스토리지 클래스 개체에서 지정한 수준과 다른 경우 스토리지 클래스 개체가 우선합니다.

단계

1. 다음 예를 사용하여 StorageClass Kubernetes 객체를 만듭니다.

```
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. 저장 클래스를 만듭니다.

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

3. 스토리지 클래스가 생성되었는지 확인하세요.

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

다음과 비슷한 출력이 표시됩니다.

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

공급량

스토리지 백엔드와 스토리지 클래스를 만든 후 이제 볼륨을 프로비저닝할 수 있습니다. 지침은 다음을 참조하세요. "볼륨 제공" .

Azure NetApp Files 볼륨을 사용하여 진행 중인 Kerberos 암호화 구성

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

시작하기 전에

- 관리되는 Red Hat OpenShift 클러스터에서 Trident 활성화했는지 확인하세요.
- 귀하가 다음에 액세스할 수 있는지 확인하십시오. `tridentctl` 공익사업.
- 요구 사항을 확인하고 다음 지침을 따르면 Kerberos 암호화를 위한 Azure NetApp Files 저장소 백엔드가 준비되었는지 확인할 수 있습니다. "Azure NetApp Files 설명서" .
- Kerberos 암호화와 함께 사용하는 모든 NFSv4 볼륨이 올바르게 구성되었는지 확인하세요. NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하세요. "NetApp NFSv4 개선 사항 및 모범 사례 가이드" .

스토리지 백엔드 생성

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

이 작업에 관하여

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

- *저장소 백엔드 수준*을 사용합니다. `spec.kerberos` 필드
- *가상 풀 레벨*을 사용하여 `spec.storage.kerberos` 필드

가상 풀 수준에서 구성을 정의하는 경우 스토리지 클래스의 레이블을 사용하여 풀이 선택됩니다.

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

- `kerberos: sec=krb5`(인증 및 암호화)
- `kerberos: sec=krb5i`(신원 보호 기능이 있는 인증 및 암호화)
- `kerberos: sec=krb5p`(신원 및 개인 정보 보호를 통한 인증 및 암호화)

단계

1. 관리되는 클러스터에서 스토리지 백엔드를 정의해야 하는 위치(스토리지 백엔드 수준 또는 가상 풀 수준)에 따라 다음 예제 중 하나를 사용하여 스토리지 백엔드 구성 파일을 만듭니다. <> 괄호 안의 값을 사용자 환경의 정보로 바꾸세요.

스토리지 백엔드 수준 예제

```
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 레벨 예시

```
---
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 만든 구성 파일을 사용하여 백엔드를 만듭니다.

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

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 파악하고 수정한 후에는 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스 생성

Kerberos 암호화를 사용하여 볼륨을 프로비저닝하기 위해 스토리지 클래스를 생성할 수 있습니다.

단계

1. 다음 예를 사용하여 StorageClass Kubernetes 객체를 만듭니다.

```
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. 저장 클래스를 만듭니다.

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

3. 스토리지 클래스가 생성되었는지 확인하세요.

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

다음과 비슷한 출력이 표시됩니다.

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

공급량

스토리지 백엔드와 스토리지 클래스를 만든 후 이제 볼륨을 프로비저닝할 수 있습니다. 지침은 다음을 참조하세요. "볼륨 제공" .