セキュリティ
Trident

NetApp
January 15, 2026

目次

セキュリティ

セキュリティ

ここで挙げた推奨事項に従って、 Trident のインストールが安全であることを確認してください。

Trident を独自の名前空間で実行する

信頼性の高いストレージを確保し、潜在的な悪意のあるアクティビティをブロックするには、アプリケーション、アプリケーション管理者、ユーザー、および管理アプリケーションがTridentオブジェクト定義またはポッドにアクセスできないようにすることが重要です。

他のアプリケーションやユーザーをTridentから分離するには、常にTrident を独自の Kubernetes 名前空間にインストールします。(`trident`）。 Trident を独自の名前空間に配置すると、Kubernetes 管理担当者だけがTridentポッドと、名前空間の CRD オブジェクトに保存されている成果物 (該当する場合はバックエンドや CHAP シークレットなど) にアクセスできるようになります。管理者のみがTrident名前空間にアクセスできるようにし、 `tridentctl`応用。

ONTAP SANバックエンドでCHAP認証を使用する

TridentはONTAP SANワークロードのCHAPベースの認証をサポートします（ `ontap-san`そして `ontap-san-economy`ドライバー)。 NetApp、ホストとストレージ バックエンド間の認証に、 Tridentを使用した双方向 CHAP を使用することを推奨しています。

SANストレージドライバを使用するONTAPバックエンドの場合、 Tridentは双方向CHAPを設定し、CHAPのユーザー名とシークレットを管理できます。 `tridentctl` 。参照"ONTAP SAN ドライバーを使用してバックエンドを構成する準備をする"Trident がONTAPバックエンドで CHAP を設定する方法を理解します。

NetApp HCIおよびSolidFireバックエンドでCHAP認証を使用する

NetApp、ホストとNetApp HCIおよびSolidFireバックエンド間の認証を確実にするために、双方向 CHAP を導入することを推奨しています。 Trident は、テナントごとに 2 つの CHAP パスワードを含む秘密オブジェクトを使用します。 Tridentがインストールされると、CHAPシークレットを管理し、それを `tridentvolume`それぞれの PV の CR オブジェクト。 PV を作成すると、 Trident はCHAP シークレットを使用して iSCSI セッションを開始し、CHAP 経由でNetApp HCIおよびSolidFireシステムと通信します。

> Tridentによって作成されたボリュームは、どのボリューム アクセス グループにも関連付けられていません。

Trident をNVE および NAE と併用する

NetApp ONTAP は、ディスクが盗難、返却、または再利用された場合に機密データを保護するために、保存データの暗号化を提供します。詳細については、"NetApp Volume Encryptionの設定 - 概要" 。

- バックエンドで NAE が有効になっている場合、 Tridentでプロビジョニングされたすべてのボリュームは NAE が有効になります。
  - NVE暗号化フラグを設定するには `""`NAE 対応ボリュームを作成します。

- NAEがバックエンドで有効になっていない場合、NVE暗号化フラグが設定されていない限り、 TridentでプロビジョニングされたボリュームはすべてNVE対応になります。 `false` (デフォルト値) をバックエンド構成で指定します。

> NAE 対応バックエンド上のTridentで作成されたボリュームは、NVE または NAE で暗号化されている必要があります。
>
> - NVE暗号化フラグを設定するには `true`Tridentバックエンド構成で NAE 暗号化をオーバーライドし、ボリュームごとに特定の暗号化キーを使用します。
> - NVE暗号化フラグを設定する `false`NAE 対応のバックエンドでは、NAE 対応のボリュームが作成されます。 NVE暗号化フラグを次のように設定してNAE暗号化を無効にすることはできません。 `false`。

- TridentでNVEボリュームを手動で作成するには、NVE暗号化フラグを明示的に設定します。 `true`。

バックエンド構成オプションの詳細については、以下を参照してください。

- "ONTAP SAN構成オプション"
- "ONTAP NAS 構成オプション"

Linux 統合キー設定 (LUKS)

Linux Unified Key Setup (LUKS) を有効にして、 Trident上のONTAP SAN およびONTAP SAN ECONOMY ボリュームを暗号化できます。 Trident は、 LUKS で暗号化されたボリュームのパスフレーズローテーションとボリューム拡張をサポートします。

Tridentでは、LUKSで暗号化されたボリュームは、aes-xts-plain64暗号とモードを使用します。これは、"NIST" 。

> LUKS 暗号化はASA r2 システムではサポートされていません。 ASA r2システムの詳細については、以下を参照してください。"ASA r2 ストレージシステムについて学ぶ" 。

開始する前に

- ワーカー ノードには、cryptsetup 2.1 以上 (3.0 未満) がインストールされている必要があります。詳細については、"Gitlab: 暗号化セットアップ" 。
- パフォーマンス上の理由から、 NetAppワーカー ノードで Advanced Encryption Standard New Instructions (AES-NI) をサポートすることを推奨しています。 AES-NI のサポートを確認するには、次のコマンドを実行します。

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

何も返されない場合、プロセッサは AES-NI をサポートしていません。 AES-NI の詳細については、以下をご覧ください。"インテル: 高度暗号化標準命令 (AES-NI)" 。

LUKS暗号化を有効にする

ONTAP SAN およびONTAP SAN ECONOMY ボリュームでは、Linux Unified Key Setup (LUKS) を使用して、ボリュームごとのホスト側暗号化を有効にすることができます。

手順

1. バックエンド構成で LUKS 暗号化属性を定義します。 ONTAP SANのバックエンド構成オプションの詳細については、以下を参照してください。"ONTAP SAN構成オプション" 。

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

2. 使用 `parameters.selector`LUKS 暗号化を使用してストレージ プールを定義します。例えば：

```
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 パスフレーズを含むシークレットを作成します。例えば：

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

制限事項

LUKS で暗号化されたボリュームは、 ONTAPの重複排除と圧縮を利用できません。

LUKSボリュームをインポートするためのバックエンド構成

LUKSボリュームをインポートするには、 `luksEncryption`に(`true`バックエンドで。その`luksEncryption`オプションはボリュームがLUKS準拠かどうかをTridentに伝える(`true`）またはLUKSに準拠していない(false) を次の例のように入力します。

```
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ボリュームをインポートするためのPVC構成

LUKSボリュームを動的にインポートするには、アノテーションを設定します`trident.netapp.io/luksEncryption`に `true`この例に示すように、PVC に LUKS 対応ストレージ クラスを含めます。

```
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パスフレーズをローテーションする

LUKS パスフレーズをローテーションし、ローテーションを確認することができます。

> パスフレーズがボリューム、スナップショット、またはシークレットによって参照されなくなったことを確認するまで、パスフレーズを忘れないようにしてください。参照されたパスフレーズが失われた場合、ボリュームをマウントできなくなり、データは暗号化されたままアクセスできなくなる可能性があります。

タスク概要

新しい LUKS パスフレーズが指定された後にボリュームをマウントするポッドが作成されると、LUKS パスフレーズのローテーションが発生します。新しいポッドが作成されると、 Trident はボリューム上の LUKS パスフレーズをシークレット内のアクティブなパスフレーズと比較します。

- ボリューム上のパスフレーズがシークレット内のアクティブなパスフレーズと一致しない場合は、ローテーションが発生します。
- ボリューム上のパスフレーズがシークレット内のアクティブなパスフレーズと一致する場合、 `previous-luks-passphrase`パラメータは無視されます。

手順

1. 追加する `node-publish-secret-name`そして `node-publish-secret-namespace`StorageClass パラメータ。例えば：

```
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. ボリュームまたはスナップショット上の既存のパスフレーズを識別します。

Volume

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

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

Snapshot

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

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

3. ボリュームの LUKS シークレットを更新して、新しいパスフレーズと以前のパスフレーズを指定します。確保する `previous-luke-passphrase-name`そして `previous-luks-passphrase`以前のパスフレーズと一致します。

```
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. ボリュームをマウントする新しいポッドを作成します。これは回転を開始するために必要です。

5. パスフレーズがローテーションされたことを確認します。

Volume

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

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

Snapshot

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

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

結果

ボリュームとスナップショットで新しいパスフレーズのみが返されたときに、パスフレーズがローテーションされました。

> たとえば、2つのパスフレーズが返された場合 `luksPassphraseNames: ["B", "A"]`、回転は不完全です。新しいポッドをトリガーして、回転を完了させることができます。

ボリューム拡張を有効にする

LUKS で暗号化されたボリュームでボリューム拡張を有効にすることができます。

手順

1. 有効にする `CSINodeExpandSecret`機能ゲート (ベータ 1.25 以上)。参照 "Kubernetes 1.25: CSIボリュームのノード駆動型拡張にSecretを使用する" 詳細については。
2. 追加する `node-expand-secret-name`そして `node-expand-secret-namespace`StorageClass パラメータ。例えば：

```
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
```

結果

オンライン ストレージ拡張を開始すると、kubelet は適切な資格情報をドライバーに渡します。

Kerberos のインフライト暗号化

Kerberos インフライト暗号化を使用すると、マネージド クラスターとストレージ バックエンド間のトラフィックの暗号化を有効にして、データ アクセスのセキュリティを向上させることができます。

Trident は、ストレージ バックエンドとしてのONTAPの Kerberos 暗号化をサポートしています。

- オンプレミス**ONTAP** - Trident は、 Red Hat OpenShift およびアップストリーム Kubernetes クラスターからオンプレミスONTAPボリュームへの NFSv3 および NFSv4 接続を介した Kerberos 暗号化をサポートします。

NFS 暗号化を使用するボリュームを作成、削除、サイズ変更、スナップショット、クローン、読み取り専用クローン、およびインポートできます。

オンプレミスの**ONTAP**ボリュームでインフライト **Kerberos** 暗号化を構成する

管理対象クラスターとオンプレミスのONTAPストレージ バックエンド間のストレージ トラフィックで Kerberos 暗号化を有効にできます。

> オンプレミスのONTAPストレージバックエンドを使用したNFSトラフィックのKerberos暗号化は、 `ontap-nas`ストレージ ドライバー。

開始する前に

- アクセスできることを確認してください `tridentctl`ユーティリティ。
- ONTAPストレージ バックエンドへの管理者アクセス権があることを確認します。
- ONTAPストレージ バックエンドから共有するボリュームの名前を必ず確認してください。
- NFS ボリュームの Kerberos 暗号化をサポートするようにONTAPストレージ VM を準備していることを確

認します。参照 "データLIFでKerberosを有効にする"手順についてはこちらをご覧ください。

- Kerberos 暗号化で使用する NFSv4 ボリュームが正しく構成されていることを確認します。 NetApp NFSv4ドメイン構成セクション（13ページ）を参照してください。 "NetApp NFSv4 の機能強化とベストプラクティスガイド" 。

ONTAPエクスポートポリシーを追加または変更する

既存のONTAPエクスポート ポリシーにルールを追加するか、 ONTAPストレージ VM ルート ボリュームと、アップストリーム Kubernetes クラスターと共有されているすべてのONTAPボリュームに対して Kerberos 暗号化をサポートする新しいエクスポート ポリシーを作成する必要があります。追加するエクスポート ポリシー ルール、または作成する新しいエクスポート ポリシーは、次のアクセス プロトコルとアクセス許可をサポートする必要があります。

アクセス プロトコル

NFS、NFSv3、および NFSv4 アクセス プロトコルを使用してエクスポート ポリシーを構成します。

アクセス詳細

ボリュームのニーズに応じて、3 つの異なるバージョンの Kerberos 暗号化のいずれかを構成できます。

- **Kerberos 5** - (認証と暗号化)
- **Kerberos 5i** - (ID保護を備えた認証と暗号化)
- **Kerberos 5p** - (アイデンティティとプライバシー保護を備えた認証と暗号化)

適切なアクセス権限を使用してONTAPエクスポート ポリシー ルールを設定します。たとえば、クラスターが Kerberos 5i と Kerberos 5p 暗号化を組み合わせて NFS ボリュームをマウントする場合は、次のアクセス設定を使用します。

タイプ	読み取り専用アクセス	読み取り/書き込みアクセス	スーパーユーザーアクセス
UNIX	有効	有効	有効
Kerberos 5i	有効	有効	有効
Kerberos 5p	有効	有効	有効

ONTAPエクスポート ポリシーとエクスポート ポリシー ルールを作成する方法については、次のドキュメントを参照してください。

- "エクスポート ポリシーの作成"
- "エクスポート ポリシーへのルールの追加"

ストレージバックエンドを作成する

Kerberos 暗号化機能を含むTridentストレージ バックエンド構成を作成できます。

タスク概要

Kerberos暗号化を設定するストレージバックエンド構成ファイルを作成するときに、次のオプションを使用して3つの異なるバージョンのKerberos暗号化のいずれかを指定できます。 `spec.nfsMountOptions`パラメータ:

- `spec.nfsMountOptions: sec=krb5`（認証と暗号化）

- `spec.nfsMountOptions: sec=krb5i`（ID保護を備えた認証と暗号化）
- `spec.nfsMountOptions: sec=krb5p`（アイデンティティとプライバシー保護を備えた認証と暗号化）

Kerberos レベルを 1 つだけ指定します。パラメータ リストで複数の Kerberos 暗号化レベルを指定した場合、最初のオプションのみが使用されます。

手順

1. マネージド クラスターで、次の例を使用してストレージ バックエンド構成ファイルを作成します。括弧<>内の値は、環境の情報で置き換えます。

```
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. 前の手順で作成した構成ファイルを使用して、バックエンドを作成します。

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

バックエンドの作成に失敗した場合、バックエンドの構成に問題があります。次のコマンドを実行すると、ログを表示して原因を特定できます。

```
tridentctl logs
```

構成ファイルの問題を特定して修正したら、create コマンドを再度実行できます。

ストレージクラスを作成する

Kerberos 暗号化を使用してボリュームをプロビジョニングするためのストレージ クラスを作成できます。

タスク概要

ストレージクラスオブジェクトを作成するときに、Kerberos暗号化の3つの異なるバージョンのいずれかを指定できます。 `mountOptions`パラメータ:

- `mountOptions: sec=krb5`（認証と暗号化）
- `mountOptions: sec=krb5i`（ID保護を備えた認証と暗号化）
- `mountOptions: sec=krb5p`（アイデンティティとプライバシー保護を備えた認証と暗号化）

Kerberos レベルを 1 つだけ指定します。パラメータ リストで複数の Kerberos 暗号化レベルを指定した場合、最初のオプションのみが使用されます。ストレージ バックエンド構成で指定した暗号化レベルがストレージ クラス オブジェクトで指定したレベルと異なる場合は、ストレージ クラス オブジェクトが優先されます。

手順

1. 次の例を使用して、StorageClass Kubernetes オブジェクトを作成します。

```
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. ストレージ クラスを作成します。

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

3. ストレージ クラスが作成されたことを確認します。

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

次のような出力が表示されます。

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

プロビジョニングボリューム

ストレージ バックエンドとストレージ クラスを作成したら、ボリュームをプロビジョニングできるようになります。手順については、 "ボリュームをプロビジョニングする" 。

Azure NetApp Filesボリュームでインフライト Kerberos 暗号化を構成する

マネージド クラスターと単一のAzure NetApp Filesストレージ バックエンドまたはAzure NetApp Filesストレージ バックエンドの仮想プール間のストレージ トラフィックに対して Kerberos 暗号化を有効にできます。

開始する前に

- 管理対象 Red Hat OpenShift クラスターでTridentが有効になっていることを確認します。
- アクセスできることを確認してください `tridentctl`ユーティリティ。
- 要件に注意し、以下の手順に従って、Kerberos暗号化用のAzure NetApp Filesストレージバックエンドを準備したことを確認します。 "Azure NetApp Files のドキュメント" 。
- Kerberos 暗号化で使用する NFSv4 ボリュームが正しく構成されていることを確認します。 NetApp NFSv4ドメイン構成セクション（13ページ）を参照してください。 "NetApp NFSv4 の機能強化とベストプラクティスガイド" 。

ストレージバックエンドを作成する

Kerberos 暗号化機能を含むAzure NetApp Filesストレージ バックエンド構成を作成できます。

タスク概要

Kerberos 暗号化を構成するストレージ バックエンド構成ファイルを作成するときに、次の 2 つのレベルのいずれかで適用されるように定義できます。

- *ストレージバックエンドレベル*は、 `spec.kerberos`分野
- *仮想プールレベル*を使用して `spec.storage.kerberos`分野

仮想プール レベルで構成を定義する場合、ストレージ クラスのラベルを使用してプールが選択されます。

どちらのレベルでも、Kerberos 暗号化の 3 つの異なるバージョンのいずれかを指定できます。

- `kerberos: sec=krb5`（認証と暗号化）
- `kerberos: sec=krb5i`（ID保護を備えた認証と暗号化）
- `kerberos: sec=krb5p`（アイデンティティとプライバシー保護を備えた認証と暗号化）

手順

1. 管理対象クラスターで、ストレージ バックエンドを定義する必要がある場所 (ストレージ バックエンド レベルまたは仮想プール レベル) に応じて、次のいずれかの例を使用してストレージ バックエンド構成ファイルを作成します。括弧<>内の値は、環境の情報で置き換えます。

ストレージバックエンドレベルの例

```
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
```

仮想プールレベルの例

```
---
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. 前の手順で作成した構成ファイルを使用して、バックエンドを作成します。

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

バックエンドの作成に失敗した場合、バックエンドの構成に問題があります。次のコマンドを実行すると、ログを表示して原因を特定できます。

```
tridentctl logs
```

構成ファイルの問題を特定して修正したら、create コマンドを再度実行できます。

ストレージクラスを作成する

Kerberos 暗号化を使用してボリュームをプロビジョニングするためのストレージ クラスを作成できます。

手順

1. 次の例を使用して、StorageClass Kubernetes オブジェクトを作成します。

```
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. ストレージ クラスを作成します。

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

3. ストレージ クラスが作成されたことを確認します。

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

次のような出力が表示されます。

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

プロビジョニングボリューム

ストレージ バックエンドとストレージ クラスを作成したら、ボリュームをプロビジョニングできるようになります。手順については、 "ボリュームをプロビジョニングする" 。