セキュリティ
Trident

NetApp
July 01, 2026

目次

セキュリティ

セキュリティ

ここに記載されている推奨事項を使用して、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

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オブジェクト定義やポッドにアクセスできないようにすることが重要です。これにより、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できます。

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てください(`trident`）。Trident を独自の名前空間に配置することで、Kubernetes 管理担当者のみが Trident ポッドおよび名前空間 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、したがって `tridentctl`アプリケーションにアクセスできるようにする必要があります。

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（ `ontap-san`および `ontap-san-economy`ドライバーを使用）のCHAPベースの認証をサポートしています。NetAppでは、ホストとストレージバックエンド間の認証にTridentで双方向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ーを使用する ONTAP バックエンドの場合、 Trident は双方向 CHAP を設定し、 CHAP ユーザー名とシークレットを `tridentctl`で管理できます。"ONTAP SANドライバを使用してバックエンドを設定する準備をします"を参照して、 Trident が ONTAP バックエンドで CHAP を設定する方法を理解してください。

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

NetAppは、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ます。Tridentは、テナントごとに2つのCHAPパスワードを含むシークレット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、それぞれのPVに対応する `tridentvolume`CRオブジェクトに保存します。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とNVEおよびNAEを使用

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"を参照してください。

- NAEがバックエンドで有効になっている場合、Tridentでプロビジョニングされたボリュームは、NAE対応になります。

    - NVE 暗号化フラグを `""`に設定すると、NAE 対応ボリュームを作成できます。

- NAEがバックエンドで有効になっていない場合、バックエンド構成でNVE暗号化フラグが `false`（デフォルト値）に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が有効になります。

> NAE 対応のバックエンドで Trident に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。
>
> - Trident バックエンド構成で NVE 暗号化フラグを `true`に設定すると、NAE 暗号化を上書きして、ボリュームごとに特定の暗号化キーを使用できます。
> - NAE対応のバックエンドでNVE暗号化フラグを `false`に設定すると、NAE対応のボリュームが作成されます。NVE暗号化フラグを `false`に設定してもNAE暗号化を無効にすることはできません。

- 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 `true`に設定します。

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 Unified Key Setup（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、"NIST"で推奨されているaes-xts-plain64暗号とモードを使用します。

> 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、"ASA r2ストレージシステムの詳細"を参照してください。

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：cryptsetup"を参照してください。
- パフォーマンス上の理由から、NetAppではワーカーノードで 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："Intel：Advanced Encryption Standard Instructions（AES-NI）"。

LUKS暗号化を有効にする

Linux Unified Key Setup（LUKS）を使用して、ONTAP SANおよびONTAP SAN ECONOMYボリュームのボリュームごとにホスト側の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 のバックエンド設定オプションの詳細については、"ONTAP SAN 構成オプション"を参照してください。

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

2. `parameters.selector`を使用して、LUKS暗号化を使用するストレージプールを定義します。例：

```
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：

```
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、バックエンドで `luksEncryption`を（`true`に設定する必要があります。 `luksEncryption`オプションは、次の例に示すように、ボリュームがLUKS準拠（（`true`）かLUKS非準拠（(false）かをTridentに通知します。

```
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 `trident.netapp.io/luksEncryption`を `true`に設定し、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でください。参照されている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. `node-publish-secret-name`と `node-publish-secret-namespace` StorageClassパラメータを追加します。例：

```
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特定します。

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。`previous-luke-passphrase-name`と `previous-luks-passphrase`が以前のパスフレーズと一致することを確認します。

```
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。これはローテーション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"]`、ローテーションは不完全です。新しいポッドをトリガーして、ローテーション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ストレージ拡張を有効にすることができます。

手順

1. `CSINodeExpandSecret`機能ゲートを有効にします（ベータ版 1.25 以降）。詳細については、"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"を参照してください。
2. `node-expand-secret-name`と `node-expand-secret-namespace` StorageClassパラメータを追加します。例：

```
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 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 および上流の Kubernetes クラスタからオンプレミス ONTAP ボリュームへの NFSv3 および NFSv4 接続での 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 **ONTAP** ボリュームでインフライト **Kerberos** 暗号化を設定する

管理対象クラスタとオンプレミス 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 暗号化を有効にすることができます。

> オンプレミス 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を使用する場合にのみサポートされます。

開始する前に

- `tridentctl`ユーティリティにアクセスできることを確認してください。
- 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てください。
- 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 を準備しておいてください。手順については "データLIFでKerberosを有効にする"を参照してください。

- 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 ドメイン設定セクション（13 ページ） "NetApp NFSv4 の機能強化とベストプラクティスガイド" を参照してください。

ONTAP エクスポートポリシーの追加または変更

ONTAP ストレージ VM ルートボリュームおよびアップストリーム Kubernetes クラスタと共有される ONTAP ボリュームに対して、Kerberos 暗号化をサポートする既存の ONTAP エクスポートポリシーにルールを追加するか、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権限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。

アクセスの詳細

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設定できます：

- **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** -（IDとプライバシー保護を使用した認証と暗号化）

適切なアクセス権限を持つ 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の作成方法については、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、 `spec.nfsMountOptions` パラメータを使用して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: 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`（ID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<>を環境からの情報に置き換えます：

```
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は、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、 `mountOptions`パラメータを使用して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: 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`（ID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ていることを確認します：

```
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
```

ボリュームのプロビジョニング

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"を参照してください。

Azure NetApp Files ボリュームでインフライト Kerberos 暗号化を設定する

マネージドクラスタと単一の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でKerberos暗号化を有効にすることができます。

開始する前に

- 管理対象の Red Hat OpenShift クラスタで Trident が有効になっていることを確認してください。
- `tridentctl`ユーティリティにアクセスできることを確認してください。
- Kerberos 暗号化用の Azure NetApp Files ストレージバックエンドを準備済みであることを確認してください。要件を確認し、 "Azure NetApp Files のドキュメント"の手順に従ってください。
- 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 ドメイン設定セクション（13 ページ） "NetApp NFSv4 の機能強化とベストプラクティスガイド"を参照してください。

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：

- `spec.kerberos`フィールドを使用した*ストレージバックエンドレベル*
- `spec.storage.kerberos`フィールドを使用した*仮想プールレベル*

仮想プールレベルで設定を定義する場合、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`（ID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<>を環境からの情報に置き換えます：

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は、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ていることを確認します：

```
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
```

ボリュームのプロビジョニング

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"を参照してください。