



# **Astra Trident をアップグレード**

## Astra Trident

NetApp  
April 04, 2024

# 目次

Astra Trident をアップグレード .....	1
Astra Trident をアップグレード .....	1
オペレータにアップグレードしてください .....	3
tridentctl を使用してアップグレードします .....	11

# Astra Trident をアップグレード

## Astra Trident をアップグレード

Astra Trident は四半期ごとにリリースサイクルを実施し、毎年 4 つのメジャーリリースをリリースしています。各新しいリリースは、以前のリリースに基づいてビルドされ、新機能とパフォーマンスの強化に加え、バグの修正や改善点が追加されています。ネットアップでは、Astra Tridentの新機能を活用するために、1年に1回以上アップグレードすることを推奨しています。

### アップグレード前の考慮事項

最新リリースの Astra Trident にアップグレードする際は、次の点を考慮してください。

- 特定のKubernetesクラスタ内のすべての名前スペースには、Astra Tridentインスタンスを1つだけインストールする必要があります。
- Trident 20.01 以降では、のベータ版のみが提供されます **"ボリューム Snapshot"** がサポートされます。Kubernetes 管理者は、従来のアルファスナップショットを保持するために、アルファスナップショットオブジェクトを安全にバックアップするか、ベータ版に変換するように注意する必要があります。
  - CSI のボリュームスナップショットは、Kubernetes 1.20 以降の GA 機能になりました。アップグレードする前に、を使用してアルファスナップショットCRDを削除する必要があります `tridentctl obliviate alpha-snapshot-crd` アルファスナップショット仕様のCRDを削除します。
  - ボリュームSnapshotのベータリリースでは、一連のCRDとSnapshotコントローラが変更されています。どちらもAstra Tridentをアップグレードする前にセットアップする必要があります。
  - 詳細については、を参照してください **"Kubernetesクラスタをアップグレードする前に知っておくべきこと"**。
- バージョン19.04以前からのアップグレードでは、いずれもAstra Tridentメタデータを自社から移行する必要があります `etcd` をCRDオブジェクトに追加します。を確認します **"Astra Tridentリリースに固有のドキュメント"** を参照して、アップグレードの仕組みを確認してください。
- アップグレードするときは、この作業を行うことが重要です `parameter.fsType` インチ StorageClasses Astra Tridentが使用。削除して再作成することができます StorageClasses 実行前のボリュームの中断はなし。
  - これは、強制の 要件 です **"セキュリティコンテキスト"** SAN ボリュームの場合。
  - **sample input**ディレクトリには、<https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-basic.yaml.template>などの例が含まれています[`storage-class-basic.yaml.template`] とリンク：<https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-bronze-default.yaml>[`storage-class-bronze-default.yaml`]をクリックします。詳細については、を参照してください **"既知の問題"**。

### ステップ1：バージョンを選択します

Astra Tridentバージョンは日付ベースです `YY.MM` 命名規則。「YY」は年の最後の2桁、「MM」は月です。ドットリリースは、の後に続きます `YY.MM.X` 条約。ここで、「X」はパッチレベルです。アップグレード前のバージョンに基づいて、アップグレード後のバージョンを選択します。

- インストールされているバージョンの4リリースウィンドウ内にある任意のターゲットリリースに直接アップグレードできます。たとえば、22.04 (22.04.1などのDOTリリースを含む) から23.04に直接アップグレードできます。
- 以前のリリースを使用している場合は、具体的な手順について、該当するリリースのドキュメントを参照してアップグレードを実行してください。そのためには、最初に4つのリリースウィンドウに対応する最新リリースにアップグレードする必要があります。たとえば'18.07を実行して'20.07リリースにアップグレードする場合は'次のように複数ステップのアップグレードプロセスを実行します
  - a. 最初のアップグレードは 18.07 から 19.07 へ。
  - b. その後 '19.07 から 20.07 にアップグレードします



OpenShift Container PlatformでTridentオペレータを使用してアップグレードする場合は、Trident 21.01.1以降にアップグレードする必要があります。21.01.0 でリリースされた Trident オペレータには、21.01.1 で修正された既知の問題が含まれています。詳細については、を参照してください "[GitHub の問題の詳細](#)"。

## ステップ2:元のインストール方法を決定します

通常は、最初のインストールと同じ方法でアップグレードする必要がありますが、可能です "[インストール方法を切り替えます](#)"。

Astra Tridentの最初のインストールに使用したバージョンを確認するには、次の手順を実行します。

1. 使用 `kubectl get pods - trident` ポッドを検査するために。
  - オペレータポッドがない場合は、を使用してAstra Tridentがインストールされています `tridentctl`。
  - オペレータポッドがある場合、Astra Tridentは手動またはHelmを使用してインストールされています。
2. オペレータポッドがある場合は、を使用します `kubectl describe tproc trident` をクリックし、Helmを使用してAstra Tridentがインストールされたかどうかを確認します。
  - Helmラベルがある場合は、Helmを使用してAstra Tridentがインストールされています。
  - Helmラベルがない場合は、Astra TridentをTridentオペレータを使用して手動でインストールしています。

## ステップ3：アップグレード方法を選択します

Astra Tridentは2つの方法でアップグレードできます。

オペレータを使用してアップグレードするタイミング

可能です "[Tridentオペレータを使用してアップグレードします](#)" 次の場合：

- オペレータまたはを使用してAstra Tridentを最初にインストールした `tridentctl`。
- CSI Tridentをアンインストールしても、インストールのメタデータは保持されます。
- CSIベースのAstra Tridentがインストールされている。の19.07以降のすべてのリリースはCSIベースです。Trident名前空間内のポッドを調べてバージョンを確認できます。
  - 23.01より前のバージョンでは、次の名前が使用されていました。 `trident-csi-*`

◦ 23.01以降でポッドの命名には次のものが使用されます。

- trident-controller-<generated id> コントローラポッドの場合
- trident-node-<operating system>-<generated id> ノードポッドの場合
- trident-operator-<generated id> オペレータポッド用



を使用している場合は、Tridentのアップグレードにオペレータを使用しないでください etcd-  
Tridentリリース（19.04以前）。

を使用してアップグレードするタイミング `tridentctl`

可能です 「tridentctl」を使用してAstra Tridentを最初にインストールした場合。

`tridentctl` は従来のAstra Tridentのインストール方法であり、複雑なカスタマイズを必要とするお客様に最適なオプションを提供します。詳細については、[を参照してください](#) "インストール方法を選択します"。

## 演算子に変更があります

Astra Tridentの21.01リリースでは、運用者のアーキテクチャが次のように変更されました。

- 演算子は `* cluster` を対象とした `*` になりました。Trident 演算子の以前のインスタンス（バージョン 20.04 ~ 20.10）は、`* 名前空間スコープ *` でした。クラスタを対象としたオペレータが有利な理由は次のとおりです。
  - リソースのアカウントビリティ：オペレータは、Astra Trident インストールに関連付けられたリソースをクラスタレベルで管理するようになりました。Astra Tridentのインストールの一環として、オペレータはを使用して複数のリソースを作成し、管理します `ownerReferences`。メンテナンス `ownerReferences` クラスタを対象としたリソースでは、OpenShiftなどの特定のKubernetesディストリビュータでエラーが発生する可能性があります。これは、クラスタを対象としたオペレータによって緩和されます。Trident リソースの自動修復とパッチ適用には、この要件が不可欠です。
  - アンインストール中のクリーンアップ：Astra Trident を完全に削除するには、関連するリソースをすべて削除する必要があります。ネームスペースを対象としたオペレータが、クラスタを対象としたリソース（`clusterRole`、`ClusterRoleBinding`、`PodSecurityPolicy` など）の削除で問題が発生し、クリーンアップが完了しない場合があります。クラスタを対象としたオペレータがこの問題を排除し、必要に応じて、Astra Trident を完全にアンインストールし、Aresh をインストールできます。
- `TridentProvisioner` が置き換えられました `TridentOrchestrator` Astra Tridentのインストールと管理に使用したカスタムリソース。また、`spec.namespace` に新しいフィールドが導入されます `TridentOrchestrator` 仕様Tridentのネームスペースは、`spec.namespace` を使用してからインストールまたはアップグレードするように指定できます `spec.namespace` フィールド。例を見てください ["こちらをご覧ください"](#)。

## オペレータにアップグレードしてください

オペレータを使用して手動またはHelmを使用して、既存のAstra Tridentインストールを簡単にアップグレードできます。

## Tridentオペレータを使用してアップグレード

通常は、最初にインストールしたときと同じ方法でAstra Tridentをアップグレードする必要があります。レビュー ["アップグレード方法を選択します"](#) をクリックしてください。

ネームスペースを対象としたオペレータ（バージョン20.07~20.10）を使用してインストールされたAstra Tridentのインスタンスからアップグレードする場合、Tridentオペレータは自動的に次の処理を行います。



- 移行 `tridentProvisioner` を追加します `tridentOrchestrator` 同じ名前のオブジェクト
- を削除します `TridentProvisioner` オブジェクトと `tridentprovisioner` CRD
- Astra Tridentを、使用しているクラスタを対象としたオペレータのバージョンにアップグレードします
- 最初にインストールされていたネームスペースと同じ場所にAstra Tridentをインストール

## クラスタを対象としたTridentオペレータ環境をアップグレード

クラスタを対象としたTridentオペレータインストールをアップグレードできます。すべてのAstra Tridentバージョン21.01以降では、クラスタを対象とした演算子を使用します。

作業を開始する前に

を実行しているKubernetesクラスタを使用していることを確認します ["サポートされるKubernetesバージョン"](#)。

手順

1. Astra Tridentのバージョンを確認します。

```
./tridentctl -n trident version
```

2. 現在のAstra Trident インスタンスのインストールに使用した Trident オペレータを削除たとえば、22.01からアップグレードする場合は、次のコマンドを実行します。

```
kubectl delete -f 22.01/trident-installer/deploy/bundle.yaml -n trident
```

3. を使用して初期インストールをカスタマイズした場合 `TridentOrchestrator` 属性を編集できます `TridentOrchestrator` インストールパラメータを変更するオブジェクト。これには、ミラーリングされたTridentおよびCSIイメージレジストリをオフラインモードに指定したり、デバッグログを有効にしたり、イメージプルシークレットを指定したりするための変更が含まれます。
4. 環境に適したバンドルYAMLファイルとAstra Tridentバージョンを使用してAstra Tridentをインストールします。たとえば、Kubernetes 1.27用Astra Trident 23.04をインストールする場合は、次のコマンドを実行します。

```
kubectl create -f 23.04.0/trident-installer/deploy/bundle_post_1_25.yaml
-n trident
```

Tridentでは、オペレータのインストールやKubernetesバージョンに関連するオブジェクトの作成に使用できるバンドルファイルが提供されています。



- クラスタでKubernetes 1.24以前を実行している場合は、を使用します ["Bundle\\_pre\\_1\\_25.yaml"](#)。
- クラスタでKubernetes 1.25以降を実行している場合は、を使用します ["bundle\\_post\\_1\\_25.yaml"](#)。

## 結果

Tridentのオペレータが、既存のAstra Tridentインストールを特定し、オペレータと同じバージョンにアップグレードします。

## 名前空間を対象としたオペレータインストールをアップグレードします

ネームスペースを対象としたオペレータ（バージョン20.07~20.10）を使用してインストールされたAstra Tridentのインスタンスから、クラスタを対象としたオペレータを対象としたインストールにアップグレードできます。

作業を開始する前に

からnamespace-scoped演算子をデプロイするために使用するバンドルYAMLファイルが必要です <https://github.com/NetApp/trident/tree/stable/vXX.XX/deploy/BUNDLE.YAML> ここで、vXX.XX は、バージョン番号およびです *BUNDLE.YAML* はバンドルYAMLファイル名です。

## 手順

1. を確認します TridentProvisioner 既存のTridentインストールのステータスはです Installed。

```
kubectl describe tprov trident -n trident | grep Message: -A 3
```

```
Message: Trident installed
Status:   Installed
Version:  v20.10.1
```



ステータスがになっている場合 `Updating` をクリックし、続行する前に解決してください。可能なステータス値のリストについては、を参照してください ["こちらをご覧ください"](#)。

2. を作成します TridentOrchestrator Tridentインストーラに付属のマニフェストを使用したCRD。

```
# Download the release required [23.04.0]
mkdir 23.04.0
cd 23.04.0
wget
https://github.com/NetApp/trident/releases/download/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. マニフェストを使用して、名前空間を対象とした演算子を削除します。

a. 正しいディレクトリにいることを確認します。

```
pwd
/root/20.10.1/trident-installer
```

b. namespace-scoped演算子を削除します。

```
kubectl delete -f deploy/<BUNDLE.YAML> -n trident

serviceaccount "trident-operator" deleted
clusterrole.rbac.authorization.k8s.io "trident-operator" deleted
clusterrolebinding.rbac.authorization.k8s.io "trident-operator"
deleted
deployment.apps "trident-operator" deleted
podsecuritypolicy.policy "tridentoperatorpods" deleted
```

c. Tridentオペレータが削除されたことを確認します。



```
kubectl get all -n trident
```

NAME	READY	STATUS	RESTARTS	AGE
pod/trident-csi-68d979fb85-dsrmn	6/6	Running	12	99d
pod/trident-csi-8jfhf	2/2	Running	6	105d
pod/trident-csi-jtnjz	2/2	Running	6	105d
pod/trident-csi-lcxvh	2/2	Running	8	105d

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
service/trident-csi	ClusterIP	10.108.174.125	<none>
34571/TCP, 9220/TCP	105d		

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AGE
daemonset.apps/trident-csi	3	3	3	3	
3					105d
					kubernetes.io/arch=amd64, kubernetes.io/os=linux

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/trident-csi	1/1	1	1	105d

NAME	DESIRED	CURRENT	READY
replicaset.apps/trident-csi-68d979fb85	1	1	1
105d			

#### 4. (オプション) インストールパラメータを変更する必要がある場合は、を更新します

TridentProvisioner 仕様これには、の値を変更するなどの変更が含まれます tridentImage、autosupportImage、プライベートイメージリポジトリ、および提供 imagePullSecrets)名前空間を対象とした演算子を削除した後、クラスタを対象とした演算子をインストールする前。更新可能なパラメータの一覧については、を参照してください ["設定オプション"](#)。

```
kubectl patch tprov <trident-provisioner-name> -n <trident-namespace> --type=merge -p '{"spec":{"debug":true}}'
```

#### 5. クラスタを対象としたTridentオペレータをインストールします。

##### a. 正しいディレクトリにいることを確認します。

```
pwd  
/root/23.04.0/trident-installer
```

##### b. クラスタを対象としたオペレータを同じ名前スペースにインストールします。

Tridentでは、オペレータのインストールやKubernetesバージョンに関連するオブジェクトの作成に使用できるバンドルファイルが提供されています。



- クラスタでKubernetes 1.24以前を実行している場合は、を使用します  
"Bundle\_pre\_1\_25.yaml"。
- クラスタでKubernetes 1.25以降を実行している場合は、を使用します  
"bundle\_post\_1\_25.yaml"。

```
kubectl create -f deploy/<BUNDLE.YAML>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#All tridentProvisioners will be removed, including the CRD itself
kubectl get tprov -n trident
Error from server (NotFound): Unable to list "trident.netapp.io/v1,
Resource=tridentprovisioners": the server could not find the
requested resource (get tridentprovisioners.trident.netapp.io)

#tridentProvisioners are replaced by tridentOrchestrator
kubectl get torc
NAME          AGE
trident      13s
```

- c. ネームスペースのTridentポッドを確認します。。 trident-controller ポッド名は、23.01で導入された命名規則を反映しています。

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

NAME	READY	STATUS	RESTARTS
trident-controller-79df798bdc-m79dc 1m41s	6/6	Running	0
trident-node-linux-xrst8 1m41s	2/2	Running	0
trident-operator-5574dbbc68-nthjv 1m52s	1/1	Running	0

- d. Tridentが目的のバージョンに更新されていることを確認

```
kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v23.04.0
```

## Helm ベースのオペレータインストールをアップグレードします

Helm ベースのオペレータインストールをアップグレードするには、次の手順を実行します。



Astra TridentがインストールされているKubernetesクラスタを1.24から1.25以降にアップグレードする場合は、value.yamlを更新して設定する必要があります  
excludePodSecurityPolicy 終了: true または、を追加します --set  
excludePodSecurityPolicy=true に移動します helm upgrade コマンドを実行してから  
クラスタをアップグレードしてください。

### 手順

1. 最新の Astra Trident リリースをダウンロード
2. を使用します helm upgrade コマンドを入力します trident-operator-23.04.0.tgz アップグレード後のバージョンが反映されます。

```
helm upgrade <name> trident-operator-23.04.0.tgz
```

初期インストール時にデフォルト以外のオプションを設定した場合（TridentイメージおよびCSIイメージのプライベートなミラーレジストリを指定するなど）は、を使用します --set これらのオプションがupgradeコマンドに含まれるようにするため、それらのオプションの値をdefaultにリセットします。



たとえば、のデフォルト値を変更するには、のように指定します `tridentDebug` を使用して、次のコマンドを実行します。

```
helm upgrade <name> trident-operator-23.04.0-custom.tgz --set  
tridentDebug=true
```

3. を実行します helm list グラフとアプリのバージョンが両方ともアップグレードされていることを確認します。を実行します tridentctl logs デバッグメッセージを確認します。

### 結果

Tridentのオペレータが、既存のAstra Tridentインストールを特定し、オペレータと同じバージョンにアップグレードします。

## オペレータ以外のインストールからアップグレードします

からTridentの最新リリースにアップグレードできます tridentctl インストール:

手順

1. 最新の Astra Trident リリースをダウンロード

```
# Download the release required [23.04.0]
mkdir 23.04.0
cd 23.04.0
wget
https://github.com/NetApp/trident/releases/download/v22.01.1/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

2. を作成します tridentorchestrator マニフェストからのCRD。

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. クラスタを対象としたオペレータを同じ名前スペースに導入します。

```
kubectl create -f deploy/<BUNDLE.YAML>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc 6/6     Running  0           150d
trident-node-linux-xrst8            2/2     Running  0           150d
trident-operator-5574dbbc68-nthjv   1/1     Running  0           1m30s
```

4. を作成します TridentOrchestrator Astra Tridentのインストール用にCR。

```

cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

```

```

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0           1m
trident-csi-xrst8                    2/2     Running   0           1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0           5m41s

```

## 5. Tridentが目的のバージョンにアップグレードされたことを確認

```

kubectl describe torc trident | grep Message -A 3

Message:          Trident installed
Namespace:        trident
Status:           Installed
Version:          v23.04.0

```

### 結果

既存のバックエンドと PVC は自動的に使用可能

## tridentctl を使用してアップグレードします

を使用すると、既存のAstra Tridentインストールを簡単にアップグレードできます tridentctl。

を使用して**Astra Trident**をアップグレードします tridentctl

Astra Trident のアンインストールと再インストールはアップグレードとして機能します。Trident をアンインストールしても、Astra Trident 環境で使用されている Persistent Volume Claim (PVC ; 永続的ボリューム要求) と Persistent Volume (PV ; 永続的ボリューム) は削除されません。Astra Trident がオフラインの間は、すでにプロビジョニング済みの PVS を引き続き使用でき、Astra Trident は、オンラインに戻った時点で作成された PVC に対してボリュームをプロビジョニングします。

作業を開始する前に

レビュー ["アップグレード方法を選択します"](#) を使用してアップグレードする前に tridentctl。

## 手順

1. のアンインストールコマンドを実行します `tridentctl` CRDと関連オブジェクトを除くAstra Tridentに関連付けられているすべてのリソースを削除する。

```
./tridentctl uninstall -n <namespace>
```

2. Astra Tridentを再インストールします。を参照してください "[tridentctl を使用して Astra Trident をインストールします](#)"。



アップグレードプロセスを中断しないでください。インストーラが完了するまで実行されることを確認します。

## を使用してボリュームをアップグレードする `tridentctl`

アップグレード後は、新しいTridentリリースで提供される豊富な機能（オンデマンドのボリュームSnapshotなど）を利用できます。を使用してボリュームをアップグレードできます `tridentctl upgrade` コマンドを実行します

レガシーボリュームがある場合は、NFSまたはiSCSIタイプからCSIタイプにアップグレードして、Astra Tridentのすべての新機能を使用する必要があります。Trident によってプロビジョニングされたレガシー PV は、従来の機能セットをサポートします。

作業を開始する前に

ボリュームをCSIタイプにアップグレードすることを決定する前に、次の点を考慮してください。

- 場合によっては、すべてのボリュームをアップグレードする必要はありません。以前に作成したボリュームには引き続きアクセスでき、正常に機能します。
- PV は、アップグレード時に展開 / 起動可能セットの一部としてマウントできます。展開 / 起動セットを停止する必要はありません。
- アップグレード時に、スタンドアロンの POD に PV を接続することはできません。ボリュームをアップグレードする前に、ポッドをシャットダウンする必要があります。
- アップグレードできるのは、PVC にバインドされているボリュームだけです。PVC にバインドされていないボリュームは、アップグレード前に削除およびインポートする必要があります。

## 手順

1. を実行します `kubectl get pv` をクリックしてPVSをリスト表示します。

```
kubectl get pv
NAME                                CAPACITY    ACCESS MODES    RECLAIM POLICY
STATUS    CLAIM                                STORAGECLASS    REASON    AGE
default-pvc-1-a8475                1073741824    RWO                                Delete
Bound    default/pvc-1                        standard
default-pvc-2-a8486                1073741824    RWO                                Delete
Bound    default/pvc-2                        standard
default-pvc-3-a849e                1073741824    RWO                                Delete
Bound    default/pvc-3                        standard
default-pvc-4-a84de                1073741824    RWO                                Delete
Bound    default/pvc-4                        standard
trident                             2Gi          RWO                                Retain
Bound    trident/trident                      19h
```

現在、Trident 20.07によって作成されたPVSのうちの4つが、を使用しています netapp.io/trident プロビジョニング担当者：

2. を実行します kubectl describe pv PVの詳細を確認します。

```
kubectl describe pv default-pvc-2-a8486

Name:                default-pvc-2-a8486
Labels:              <none>
Annotations:         pv.kubernetes.io/provisioned-by: netapp.io/trident
                    volume.beta.kubernetes.io/storage-class: standard
Finalizers:          [kubernetes.io/pv-protection]
StorageClass:        standard
Status:              Bound
Claim:               default/pvc-2
Reclaim Policy:      Delete
Access Modes:        RWO
VolumeMode:          Filesystem
Capacity:            1073741824
Node Affinity:       <none>
Message:
Source:
  Type:              NFS (an NFS mount that lasts the lifetime of a pod)
  Server:            10.xx.xx.xx
  Path:              /trid_1907_alpha_default_pvc_2_a8486
  ReadOnly:          false
```

PVはを使用して作成されました netapp.io/trident プロビジョニング担当者とプロビジョニングタイプはNFSです。Astra Tridentが提供する新機能をすべてサポートするには、このPVをCSIタイプにアップグレードする必要があります。

3. を実行します `tridentctl upgrade volume <name-of-trident-volume>` 従来のAstra TridentボリュームをCSI仕様にアップグレードするコマンド。

```
./tridentctl get volumes -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS | PROTOCOL |
BACKEND UUID           |  STATE  |  MANAGED  |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true         |          |
| default-pvc-3-a849e | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true         |          |
| default-pvc-1-a8475 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true         |          |
| default-pvc-4-a84de | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true         |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

./tridentctl upgrade volume default-pvc-2-a8486 -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS | PROTOCOL |
BACKEND UUID           |  STATE  |  MANAGED  |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true         |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

4. を実行します `kubectl describe pv` ボリュームがCSIボリュームであることを確認します。



```
kubectl describe pv default-pvc-2-a8486
Name:                default-pvc-2-a8486
Labels:              <none>
Annotations:         pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                    volume.beta.kubernetes.io/storage-class: standard
Finalizers:          [kubernetes.io/pv-protection]
StorageClass:        standard
Status:              Bound
Claim:               default/pvc-2
Reclaim Policy:      Delete
Access Modes:        RWO
VolumeMode:          Filesystem
Capacity:            1073741824
Node Affinity:       <none>
Message:
Source:
  Type:               CSI (a Container Storage Interface (CSI) volume
source)
  Driver:              csi.trident.netapp.io
  VolumeHandle:        default-pvc-2-a8486
  ReadOnly:            false
  VolumeAttributes:    backendUUID=c5a6f6a4-b052-423b-80d4-
8fb491a14a22

internalName=trid_1907_alpha_default_pvc_2_a8486
                    name=default-pvc-2-a8486
                    protocol=file
Events:               <none>
```

## 著作権に関する情報

Copyright © 2024 NetApp, Inc. All Rights Reserved. Printed in the U.S.このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および/または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

## 商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。