



Tridentを使用

Trident

NetApp
January 17, 2025

目次

Tridentを使用	1
ワーカーノードを準備します	1
バックエンドの構成と管理	13
ストレージクラスの作成と管理	166
ボリュームのプロビジョニングと管理	171

Tridentを使用

ワーカーノードを準備します

Kubernetesクラスタ内のすべてのワーカーノードが、ポッド用にプロビジョニングしたボリュームをマウントできる必要があります。ワーカーノードを準備するには、ドライバの選択に基づいて、NFS、iSCSI、NVMe/TCP、またはFCの各ツールをインストールする必要があります。

適切なツールを選択する

ドライバを組み合わせで使用している場合は、ドライバに必要なすべてのツールをインストールする必要があります。最新バージョンのRedHat CoreOSには、デフォルトでツールがインストールされています。

NFSツール

"[NFSツールのインストール](#)"を使用している場合： `ontap-nas`、`ontap-nas-economy`、`ontap-nas-flexgroup`、`azure-netapp-files`、`gcp-cvs`。

iSCSIツール

"[iSCSIツールをインストール](#)"を使用している場合： `ontap-san`、`ontap-san-economy`、`solidfire-san`。

NVMeツール

"[NVMeツールをインストールする](#)"を使用している場合 `ontap-san Non-Volatile Memory Express (NVMe) over TCP (NVMe/TCP) プロトコル`の場合。



NVMe/TCPにはONTAP 9.12以降を推奨します。

SCSI over FCツール

- SCSI over Fibre Channel (FC) は、Trident 24.10リリースの技術プレビュー機能です。*

"[FCツールのインストール](#)"を`sanType (SCSI over FC)`で`fc``使用している場合 `ontap-san`。

詳細については、[を参照してください "FCおよびFC-NVMe SANホストの構成方法"](#)。

ノードサービスの検出

Tridentは、ノードでiSCSIサービスまたはNFSサービスを実行できるかどうかを自動的に検出しようとしません。



ノードサービス検出で検出されたサービスが特定されますが、サービスが適切に設定されていることは保証されませ逆に、検出されたサービスがない場合も、ボリュームのマウントが失敗する保証はありません。

イベントを確認します

Tridentは、検出されたサービスを識別するためのイベントをノードに対して作成します。次のイベントを確認するには、`を`実行します。

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

検出されたサービスを確認

Tridentは、TridentノードCR上の各ノードで有効になっているサービスを識別します。検出されたサービスを表示するには、を実行します。

```
tridentctl get node -o wide -n <Trident namespace>
```

NFS ボリューム

オペレーティングシステム用のコマンドを使用して、NFSツールをインストールします。ブート時にNFSサービスが開始されていることを確認します。

RHEL 8以降

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```



NFSツールをインストールしたあとにワーカーノードをリブートして、コンテナにボリュームを接続する際の障害を回避します。

iSCSI ボリューム

Tridentでは、iSCSIセッションの確立、LUNのスキャン、マルチパスデバイスの検出、フォーマット、ポッドへのマウントを自動的に実行できます。

iSCSIの自己回復機能

ONTAPシステムの場合、Tridentは5分ごとにiSCSIの自己修復を実行し、次のことを実現します。

1. *希望するiSCSIセッションの状態と現在のiSCSIセッションの状態を識別します
2. *希望する状態と現在の状態を比較して、必要な修理を特定します。Tridentは、修理の優先順位と、修理をいつプリエンプトするかを決定します。
3. *現在のiSCSIセッションの状態を希望するiSCSIセッションの状態に戻すために必要な修復*を実行します。



自己修復アクティビティのログは、それぞれのデーモンセットポッドのコンテナにあり `trident-main``ます。ログを表示するには、Tridentのインストール時に `「true」` に設定しておく必要があります ``debug`。

Trident iSCSIの自己修復機能を使用すると、次のことを防止できます。

- ネットワーク接続問題 後に発生する可能性がある古いiSCSIセッションまたは正常でないiSCSIセッション。セッションが古くなった場合、Tridentは7分間待機してからログアウトし、ポータルとの接続を再確立します。



たとえば、ストレージコントローラでCHAPシークレットがローテーションされた場合にネットワークが接続を失うと、古い (*stale*) CHAPシークレットが保持されることがあります。自己修復では、これを認識し、自動的にセッションを再確立して、更新されたCHAPシークレットを適用できます。

- iSCSIセッションがありません
- LUNが見つかりません
- Tridentをアップグレードする前に考慮すべきポイント*
- ノード単位のigroup (23.04以降で導入) のみを使用している場合、iSCSIの自己修復によってSCSIバス内のすべてのデバイスに対してSCSI再スキャンが開始されます。
- バックエンドを対象としたigroup (23.04で廃止) のみを使用している場合、iSCSIの自己修復によってSCSIバス内の正確なLUN IDのSCSI再スキャンが開始されます。
- ノード単位のigroupとバックエンドを対象としたigroupが混在している場合、iSCSIの自己修復によってSCSIバス内の正確なLUN IDのSCSI再スキャンが開始されます。

iSCSIツールをインストール

使用しているオペレーティングシステム用のコマンドを使用して、iSCSIツールをインストールします。

作業を開始する前に

- Kubernetes クラスタ内の各ノードには一意の IQN を割り当てる必要があります。* これは必須の前提条件です *。
- RHCOSバージョン4.5以降またはRHEL互換のその他のLinuxディストリビューションをで使用している場合は、を使用します `solidfire-san Driver`およびElement OS 12.5以前。CHAP認証アルゴリズムがMD5 inに設定されていることを確認します `/etc/iscsi/iscsid.conf`。Element 12.7では、FIPS準拠のセキュアなCHAPアルゴリズムSHA1、SHA-256、およびSHA3-256が提供されています。

```
sudo sed -i 's/^\(node.session.auth.chap_algs\) .*/\1 = MD5/'  
/etc/iscsi/iscsid.conf
```

- iSCSI PVSを搭載したRHEL / RedHat CoreOSを実行するワーカーノードを使用する場合は、を指定します `discard StorageClass`のmountOptionを使用して、インラインのスペース再生を実行します。を参照してください ["Red Hat のドキュメント"](#)。

RHEL 8以降

1. 次のシステムパッケージをインストールします。

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-  
multipath
```

2. iscsi-initiator-utils のバージョンが 6.2.0.874-2.el7 以降であることを確認します。

```
rpm -q iscsi-initiator-utils
```

3. マルチパスを有効化：

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```



「/etc/multipath.conf」に「find_multipaths no」が「defVaults」に含まれていることを確認します。

4. 「iscsid」と「multipathd」が実行されていることを確認します。

```
sudo systemctl enable --now iscsid multipathd
```

5. 'iSCSI' を有効にして開始します

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. 次のシステムパッケージをインストールします。

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools  
scsistools
```

2. open-iscsi バージョンが 2.0.874-5ubuntu2.10 以降（bionic の場合）または 2.0.874-7.1ubuntu6.1 以降（Focal の場合）であることを確認します。

```
dpkg -l open-iscsi
```

3. スキャンを手動に設定：

```
sudo sed -i 's/^\(node.session.scan\).*\/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. マルチパスを有効化：

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```



「/etc/multipath.conf」に「find_multipaths no」が「defaults」に含まれていることを確認します。

5. 「open-iSCSI」 および「マルチパスツール」が有効で実行されていることを確認します。

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```



Ubuntu 18.04 では 'iSCSI デーモンを起動するために 'open-iscsi' を起動する前に 'iscsiadm' を持つターゲット・ポートを検出する必要がありますまたは 'iscsid' サービスを 'iscsid' を自動的に開始するように変更することもできます

iSCSI自己回復の設定または無効化

次のTrident iSCSI自己修復設定を構成して、古いセッションを修正できます。

- * iSCSIの自己修復間隔*：iSCSIの自己修復を実行する頻度を指定します（デフォルト：5分）。小さい数値を設定することで実行頻度を高めるか、大きい数値を設定することで実行頻度を下げることができます。



iSCSIの自己修復間隔を0に設定すると、iSCSIの自己修復が完全に停止します。iSCSIの自己修復を無効にすることは推奨しません。iSCSIの自己修復が意図したとおりに機能しない、またはデバッグ目的で機能しない特定のシナリオでのみ無効にする必要があります。

- * iSCSI自己回復待機時間*：正常でないセッションからログアウトして再ログインを試みるまでのiSCSI自己回復の待機時間を決定します（デフォルト：7分）。健全でないと識別されたセッションがログアウトされてから再度ログインしようとするまでの待機時間を長くするか、またはログアウトしてログインしてからログインするまでの時間を短くするように設定できます。

Helm

iSCSIの自己修復設定を構成または変更するには、`iscsiSelfHealingInterval` および `iscsiSelfHealingWaitTime` helmのインストール中またはhelmの更新中のパラメータ。

次の例では、iSCSIの自己修復間隔を3分、自己修復の待機時間を6分に設定しています。

```
helm install trident trident-operator-100.2410.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

Tridentctl

iSCSIの自己修復設定を構成または変更するには、`iscsi-self-healing-interval` および `iscsi-self-healing-wait-time` tridentctlのインストールまたは更新中のパラメータ。

次の例では、iSCSIの自己修復間隔を3分、自己修復の待機時間を6分に設定しています。

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

NVMe/TCPホリユウム

オペレーティングシステムに対応したコマンドを使用してNVMeツールをインストールします。



- NVMeにはRHEL 9以降が必要です。
- Kubernetesノードのカーネルバージョンが古すぎる場合や、使用しているカーネルバージョンに対応するNVMeパッケージがない場合は、ノードのカーネルバージョンをNVMeパッケージで更新しなければならないことがあります。

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```


インストールを確認します

インストールが完了したら、次のコマンドを使用して、Kubernetesクラスタ内の各ノードに一意的NQNが割り当てられていることを確認します。

```
cat /etc/nvme/hostnqn
```



Tridentでは、NVMeがダウンしてもパスがあきらめないように値が変更され`ctrl_device_tmo`ます。この設定は変更しないでください。

FCツールのインストール

オペレーティングシステム用のコマンドを使用して、FCツールをインストールします。

- FC PVSでRHEL / RedHat CoreOSを実行するワーカーノードを使用する場合は、StorageClassでmountOptionを指定し`discard`でインラインのスペース再生を実行します。を参照してください "[Red Hat のドキュメント](#)"。

RHEL 8以降

1. 次のシステムパッケージをインストールします。

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. マルチパスを有効化：

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```



「/etc/multipath.conf」に「find_multipaths no」が「defVaults」に含まれていることを確認します。

3. が実行中であることを確認し `multipathd` ます。

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. 次のシステムパッケージをインストールします。

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. マルチパスを有効化：

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```



「/etc/multipath.conf」に「find_multipaths no」が「defVaults」に含まれていることを確認します。

3. が有効で実行中であることを確認し `multipath-tools` ます。

```
sudo systemctl status multipath-tools
```

ファイバチャネル (FC) のサポート

Fibre Channel (FC ; ファイバチャネル) プロトコルをTridentで使用して、ONTAPシステムでストレージリソースをプロビジョニングおよび管理できるようになりました。

- SCSI over Fibre Channel (FC) は、Trident 24.10リリースの技術プレビュー機能です。*

ファイバチャネルは、その高いパフォーマンス、信頼性、拡張性から、エンタープライズストレージ環境で広く採用されているプロトコルです。ストレージデバイスに堅牢で効率的な通信チャネルを提供し、高速で安全なデータ転送を可能にします。SCSI over Fibre Channelを使用すると、既存のSCSIベースのストレージインフラストラクチャを活用しながら、ファイバチャネルの高パフォーマンスと長距離機能を活用できます。これにより、ストレージリソースを統合し、低レイテンシで大量のデータを処理できる、拡張性と効率性に優れたストレージエリアネットワーク (SAN) を構築できます。

TridentでFC機能を使用すると、次のことが可能になります。

- 配置仕様を使用してPVCを動的にプロビジョニングします。
- ボリュームのSnapshotを作成し、そのSnapshotから新しいボリュームを作成します。
- 既存のFC-PVCのクローンを作成します。
- 導入済みのボリュームのサイズを変更します。

前提条件

FCに必要なネットワークとノードを設定します。

ネットワーク設定

1. ターゲットインターフェイスのWWPNを取得します。詳細については、["network interface show"](#) を参照してください。
2. イニシエータ (ホスト) のインターフェイスのWWPNを取得します。

対応するホストオペレーティングシステムユーティリティを参照してください。

3. ホストとターゲットのWWPNを使用してFCスイッチにゾーニングを設定します。

詳細については、各スイッチベンダーのドキュメントを参照してください。

詳細については、次のONTAPドキュメントを参照してください。

- ["ファイバチャネルとFCoEのゾーニングの概要"](#)
- ["FCおよびFC-NVMe SANホストの構成方法"](#)

ワーカーノードを準備します

Kubernetesクラスタ内のすべてのワーカーノードが、ポッド用にプロビジョニングしたボリュームをマウントできる必要があります。ワーカーノードをFC用に準備するには、必要なツールをインストールする必要があります。

FCツールのインストール

オペレーティングシステム用のコマンドを使用して、FCツールをインストールします。

- FC PVSでRHEL / RedHat CoreOSを実行するワーカーノードを使用する場合は、StorageClassでmountOptionを指定し`discard`でインラインのスペース再生を実行します。を参照してください "[Red Hat のドキュメント](#)"。

RHEL 8以降

1. 次のシステムパッケージをインストールします。

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. マルチパスを有効化：

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```



「/etc/multipath.conf」に「find_multipaths no」が「defVaults」に含まれていることを確認します。

3. が実行中であることを確認し `multipathd` ます。

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. 次のシステムパッケージをインストールします。

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. マルチパスを有効化：

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```



「/etc/multipath.conf」に「find_multipaths no」が「defVaults」に含まれていることを確認します。

3. が有効で実行中であることを確認し `multipath-tools` ます。

```
sudo systemctl status multipath-tools
```

バックエンド構成の作成

ドライバおよび `fc`sanType` として Trident バックエンドを作成します ``ontap-san`。

参照先：

- ["バックエンドにONTAP SANドライバを設定する準備をします"](#)
- ["ONTAP のSAN構成オプションと例"](#)

FCマシヨウシタバックエンドコウセイノレイ

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  sanType: fcp
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

ストレージクラスを作成する。

詳細については、以下を参照してください。

- ["ストレージ構成オプション"](#)

ストレージクラスの例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fcp-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  protocol: "fcp"
  storagePool: "aggr1"
allowVolumeExpansion: True
```

バックエンドの構成と管理

バックエンドを設定

バックエンドは、Tridentとストレージシステム間の関係を定義します。Tridentは、そのストレージシステムとの通信方法や、Tridentがそのシステムからボリュームをプロビジョニングする方法を解説します。

Tridentは、ストレージクラスで定義された要件に一致するストレージプールをバックエンドから自動的に提供します。ストレージシステムにバックエンドを設定する方法について説明します。

- ["Azure NetApp Files バックエンドを設定します"](#)
- ["Google Cloud NetApp Volumeバックエンドの設定"](#)
- ["Cloud Volumes Service for Google Cloud Platform バックエンドを設定します"](#)
- ["NetApp HCI または SolidFire バックエンドを設定します"](#)
- ["バックエンドに ONTAP または Cloud Volumes ONTAP NAS ドライバを設定します"](#)
- ["バックエンドに ONTAP または Cloud Volumes ONTAP SAN ドライバを設定します"](#)
- ["Amazon FSx for NetApp ONTAPでTridentを使用"](#)

Azure NetApp Files の特長

Azure NetApp Files バックエンドを設定します

Azure NetApp FilesをTridentのバックエンドとして設定できます。Azure NetApp Filesバックエンドを使用してNFSボリュームとSMBボリュームを接続できます。Tridentは、Azure Kubernetes Services (AKS) クラスタの管理対象IDを使用したクレデンシャル管理もサポートしています。

Azure NetApp Filesドライバの詳細

Tridentには、クラスタと通信するための次のAzure NetApp Filesストレージドライバが用意されています。サポートされているアクセスモードは、*ReadWriteOnce(RWO)*、*ReadOnlyMany(ROX)*、*ReadWriteMany(RWX)*、*ReadWriteOncePod(RWOP)*です。

ドライバ	プロトコル	ボリュームモード	サポートされているアクセスモード	サポートされるファイルシステム
「azure-NetApp-files」と入力します	NFS SMB	ファイルシステム	RWO、ROX、RWX、RWOP	nfs、smb

考慮事項

- Azure NetApp Filesサービスでは、50GiB未満のボリュームはサポートされません。より小さいボリュームを要求すると、Tridentは50GiBのボリュームを自動的に作成します。
- Tridentでは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみがサポートさ

れます。

AKSの管理対象ID

Tridentでは、Azure Kubernetes Servicesクラスタがサポートされます"管理対象ID"。管理されたアイデンティティによって提供される合理的なクレデンシャル管理を利用するには、次のものがが必要です。

- AKSを使用して導入されるKubernetesクラスタ
- AKS Kubernetesクラスタに設定された管理対象ID
- 指定する "Azure" を含むTridentがインストールされています。 `cloudProvider`

Trident オペレータ

Trident演算子を使用してTridentをインストールするには、を `tridentorchestrator_cr.yaml`` に ` "Azure" `設定します `cloudProvider`。例：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

Helm

次の例では、環境変数を使用してTridentセットをAzureに `\$CP`インストールし `cloudProvider`ます。

```
helm install trident trident-operator-100.2410.0.tgz --create
--namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

`tridentctl`

次の例では、Tridentをインストールし、フラグをに `Azure`設定し `cloudProvider`ます。

```
tridentctl install --cloud-provider="Azure" -n trident
```

AKSのクラウドID

クラウドIDを使用すると、Kubernetesポッドは、明示的なAzureクレデンシャルを指定するのではなく、ワークロードIDとして認証することでAzureリソースにアクセスできます。

AzureでクラウドIDを活用するには、以下が必要です。

- AKSを使用して導入されるKubernetesクラスター
- AKS Kubernetesクラスターに設定されたワークロードIDとoidc-issuer
- ワークロードIDを指定 "Azure"、および `cloudIdentity` 指定するを含むTridentがインストールされている `cloudProvider`

Trident オペレータ

Trident演算子を使用してTridentをインストールするには、をに設定し、を `tridentorchestrator_cr.yaml` に `"Azure"` 設定 `cloudProvider` し `cloudIdentity` `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx` ます。

例：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  *cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' *
```

Helm

次の環境変数を使用して、* `cloud-provider` (CP) フラグと `cloud-identity` (CI) *フラグの値を設定します。

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx' "
```

次の例では、環境変数を使用してTridentをインストールし `cloudProvider`、をAzureに `CP` 設定し、をUSING THE環境変数 `CI` に設定し `cloudIdentity` ます。

```
helm install trident trident-operator-100.2410.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

`tridentctl`

次の環境変数を使用して、* `cloud provider` フラグと `cloud identity` *フラグの値を設定します。

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

次の例では、Tridentをインストールし、フラグをに設定し、`cloud-identity` を `CI` に `CP` 設定し `cloud-provider` ます。

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Azure NetApp Files バックエンドを設定する準備をします

Azure NetApp Files バックエンドを設定する前に、次の要件を満たしていることを確認する必要があります。

NFSボリュームと**SMB**ボリュームの前提条件

Azure NetApp Files を初めてまたは新しい場所で使用する場合は、Azure NetApp Files をセットアップしてNFSボリュームを作成するためにいくつかの初期設定が必要です。を参照してください "[Azure : Azure NetApp Files をセットアップし、NFSボリュームを作成します](#)"。

を設定して使用します "[Azure NetApp Files の特長](#)" バックエンドには次のものがが必要です。



- subscriptionID、tenantID、clientID、location`および `clientSecret AKS クラスタで管理対象IDを使用する場合はオプションです。
- tenantID、clientID`および `clientSecret は、AKSクラスタでクラウドIDを使用する場合はオプションです。

- 容量プール。を参照してください "[Microsoft : Azure NetApp Files 用の容量プールを作成します](#)"。
- Azure NetApp Files に委任されたサブネット。を参照してください "[Microsoft : サブネットをAzure NetApp Files に委任します](#)"。
- Azure NetApp Files が有効な Azure サブスクリプションのスクリプト ID。
- tenantID、clientID`および `clientSecret から "[アプリケーション登録](#)" Azure Active Directory で、Azure NetApp Files サービスに対する十分な権限がある。アプリケーション登録では、次のいずれかを使用します。
 - オーナーまたは寄与者のロール "[Azureで事前定義](#)"。
 - "[カスタム投稿者ロール](#)"(assignableScopes (サブスクリプションレベル))。次の権限がTridentで必要な権限のみに制限されています。カスタムロールを作成したら、"[Azureポータルを使用してロールを割り当てます](#)"を参照してください。

```

{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited
permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",
          "Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

```

```

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations/delete",

    "Microsoft.Features/features/read",
    "Microsoft.Features/operations/read",
    "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
    ],
    "notActions": [],
    "dataActions": [],
    "notDataActions": []
  }
]
}
}

```

- Azureがサポートされず location を1つ以上含むデータセンターを展開します ["委任されたサブネット"](#)。Trident 22.01の時点では location パラメータは、バックエンド構成ファイルの最上位にある必須フィールドです。仮想プールで指定された場所の値は無視されます。
- を使用してください Cloud Identity、client IDAから ["ユーザーが割り当てた管理ID"](#) そのIDを azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx。

SMBボリュームに関するその他の要件

SMBボリュームを作成するには、以下が必要です。

- Active Directoryが設定され、Azure NetApp Files に接続されています。を参照してください ["Microsoft : Azure NetApp Files のActive Directory接続を作成および管理します"](#)。
- Linuxコントローラノードと少なくとも1つのWindowsワーカーノードでWindows Server 2022を実行しているKubernetesクラスター。Tridentでは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみがサポートされます。
- Azure NetApp FilesがActive Directoryに対して認証できるように、Active Directoryクレデンシャルを含む少なくとも1つのTridentシークレット。シークレットを生成するには smbcreds :

```

kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'

```

- Windowsサービスとして設定されたCSIプロキシ。を設定します `csi-proxy` を参照してください "[GitHub: CSIプロキシ](#)" または "[GitHub: Windows向けCSIプロキシ](#)" Windowsで実行されているKubernetesノードの場合。

Azure NetApp Files バックエンド構成のオプションと例

Azure NetApp FilesのNFSおよびSMBバックエンド構成オプションについて説明し、構成例を確認します。

バックエンド構成オプション

Tridentはバックエンド構成（サブネット、仮想ネットワーク、サービスレベル、場所）を使用して、要求された場所で使用可能な容量プール上に、要求されたサービスレベルとサブネットに一致するAzure NetApp Files ボリュームを作成します。



Tridentでは、手動QoS容量プールはサポートされません。

Azure NetApp Filesバックエンドには、次の設定オプションがあります。

パラメータ	説明	デフォルト
「バージョン」		常に 1
'storageDriverName'	ストレージドライバの名前	「 azure-NetApp-files 」
backendName`	カスタム名またはストレージバックエンド	ドライバ名 + "_" + ランダムな文字
' スクリプト ID' 。	Azure サブスクリプションのサブスクリプション ID AKSクラスタで管理IDが有効になっている場合はオプションです。	
「 tenantID 」 。	アプリケーション登録からのテナント ID AKSクラスタで管理IDまたはクラウドIDを使用する場合はオプションです。	
「 clientID 」 。	アプリケーション登録からのクライアント ID AKSクラスタで管理IDまたはクラウドIDを使用する場合はオプションです。	
「 clientSecret 」 を入力します。	アプリケーション登録からのクライアントシークレット AKSクラスタで管理IDまたはクラウドIDを使用する場合はオプションです。	

パラメータ	説明	デフォルト
「サービスレベル」	「標準」、「プレミアム」、「ウルトラ」のいずれかです	"" (ランダム)
「ロケーション」	新しいボリュームを作成する Azure の場所の名前 AKS クラスタで管理 ID が有効になっている場合はオプションです。	
「resourceGroups」	検出されたリソースをフィルタリングするためのリソースグループのリスト	"" (フィルタなし)
「netappAccounts」のように入力します	検出されたリソースをフィルタリングするためのネットアップアカウントのリスト	"" (フィルタなし)
「capacityPools」	検出されたリソースをフィルタリングする容量プールのリスト	"" (フィルタなし、ランダム)
「virtualNetwork」	委任されたサブネットを持つ仮想ネットワークの名前	""
「サブネット」	「microsoft.Netapp/volumes」に委任されたサブネットの名前	""
「ネットワーク機能」	ボリューム用の VNet 機能のセットです。の場合もあります Basic または Standard。ネットワーク機能は一部の地域では使用できず、サブスクリプションで有効にする必要がある場合があります。を指定します networkFeatures この機能を有効にしないと、ボリュームのプロビジョニングが失敗します。	""
「nfsvMountOptions」のように入力します	NFS マウントオプションのきめ細かな制御。SMB ボリュームでは無視されます。NFS バージョン 4.1 を使用してボリュームをマウントするには、を参照してください nfsvers=4 カンマで区切って複数のマウントオプションリストを指定し、NFS v4.1 を選択します。ストレージクラス定義で設定されたマウントオプションは、バックエンド構成で設定されたマウントオプションよりも優先されます。	"nfsvers=3 "
「limitVolumeSize」と入力します	要求されたボリュームサイズがこの値を超えている場合はプロビジョニングが失敗します	"" (デフォルトでは適用されません)

パラメータ	説明	デフォルト
「バグトレースフラグ」	トラブルシューティング時に使用するデバッグフラグ。例： `{"API":false,"メソッド":"true,"検出":"true"}`トラブルシューティングを行って詳細なログダンプが必要な場合を除き、このオプションは使用しないでください。	null
nasType	NFSボリュームまたはSMBボリュームの作成を設定オプションはです nfs、 smb または null。 null に設定すると、デフォルトでNFSボリュームが使用されます。	nfs
supportedTopologies	このバックエンドでサポートされているリージョンとゾーンのリストを表します。詳細については、 を参照してください "CSI トポロジを使用します" 。	



ネットワーク機能の詳細については、[を参照してください "Azure NetApp Files ボリュームのネットワーク機能を設定します"](#)。

必要な権限とリソース

PVCの作成時に「No capacity pools found」エラーが表示される場合は、アプリケーション登録に必要な権限とリソース（サブネット、仮想ネットワーク、容量プール）が関連付けられていない可能性があります。デバッグを有効にすると、バックエンドの作成時に検出されたAzureリソースがTridentによってログに記録されず。適切なロールが使用されていることを確認します。

の値 resourceGroups、 netappAccounts、 capacityPools、 virtualNetwork および subnet 短縮名または完全修飾名を使用して指定できます。ほとんどの場合、短縮名は同じ名前の複数のリソースに一致する可能性があるため、完全修飾名を使用することを推奨します。

。 resourceGroups、 netappAccounts および capacityPools 値は、検出されたリソースのセットをこのストレージバックエンドで使用可能なリソースに制限するフィルタであり、任意の組み合わせで指定できます。完全修飾名の形式は次のとおりです。

を入力します	の形式で入力し
リソースグループ	< リソースグループ >
ネットアップアカウント	< リソースグループ >/< ネットアップアカウント >
容量プール	< リソースグループ >/< ネットアップアカウント >/< 容量プール >
仮想ネットワーク	< リソースグループ >/< 仮想ネットワーク >
サブネット	< resource group >/< 仮想ネットワーク >/< サブネット >

ボリュームのプロビジョニング

構成ファイルの特別なセクションで次のオプションを指定することで、デフォルトのボリュームプロビジョニングを制御できます。を参照してください [\[構成例\]](#) を参照してください。

パラメータ	説明	デフォルト
「 exportRule 」	新しいボリュームに対するエクスポートルール exportRule CIDR表記のIPv4アドレスまたはIPv4サブネットの任意の組み合わせをカンマで区切って指定する必要があります。SMBボリュームでは無視されます。	"0.0.0.0/0 "
「スナップショット方向」	.snapshot ディレクトリの表示を制御します	NFSv4の場合は「true」 NFSv3の場合は「false」
「 size 」	新しいボリュームのデフォルトサイズ	" 100G "
「 unixPermissions 」	新しいボリュームのUNIX権限（8進数の4桁）。SMBボリュームでは無視されます。	""（プレビュー機能、サブスクリプションでホワイトリスト登録が必要）

構成例

次の例は、ほとんどのパラメータをデフォルトのままにする基本的な設定を示しています。これは、バックエンドを定義する最も簡単な方法です。

最小限の構成

これは、バックエンドの絶対的な最小構成です。この構成では、Tridentは設定された場所でAzure NetApp Filesに委譲されたすべてのNetAppアカウント、容量プール、およびサブネットを検出し、それらのプールおよびサブネットの1つに新しいボリュームをランダムに配置します。は省略されているため、nasType nfs デフォルトが適用され、バックエンドでNFSボリュームがプロビジョニングされません。

この構成は、Azure NetApp Filesの使用を開始して試している段階で、実際にはプロビジョニングするボリュームに対して追加の範囲を設定することが必要な場合に適しています。

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
  tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
  clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
  clientSecret: SECRET
  location: eastus
```

AKSの管理対象ID

このバックエンド構成では、subscriptionID、tenantID、`clientID`および`clientSecret`は、管理対象IDを使用する場合はオプションです。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
```

AKSのクラウドID

このバックエンド構成では、tenantID、`clientID`および`clientSecret`は、クラウドIDを使用する場合はオプションです。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

容量プールフィルタを使用した特定のサービスレベル構成

このバックエンド構成では、容量プール内のAzureの場所`Ultra`にボリュームが配置され`eastus`ます。Tridentは、その場所のAzure NetApp Filesに委譲されたすべてのサブネットを自動的に検出し、そのいずれかに新しいボリュームをランダムに配置します。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
```

高度な設定

このバックエンド構成は、ボリュームの配置を単一のサブネットにまで適用する手間をさらに削減し、一部のボリュームプロビジョニングのデフォルト設定も変更します。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: 'true'
  size: 200Gi
  unixPermissions: '0777'
```

仮想プール構成

このバックエンド構成では、1つのファイルに複数のストレージプールを定義します。これは、異なるサービスレベルをサポートする複数の容量プールがあり、それらを表すストレージクラスを Kubernetes で作成する場合に便利です。プールを区別するために、仮想プールのラベルを使用しました performance。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
- application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
- labels:
  performance: gold
  serviceLevel: Ultra
  capacityPools:
  - ultra-1
  - ultra-2
  networkFeatures: Standard
- labels:
  performance: silver
  serviceLevel: Premium
  capacityPools:
  - premium-1
- labels:
  performance: bronze
  serviceLevel: Standard
  capacityPools:
  - standard-1
  - standard-2
```

サポートされるトポロジ構成

Tridentを使用すると、リージョンとアベイラビリティゾーンに基づいてワークロード用のボリュームを簡単にプロビジョニングできます。`supportedTopologies`このバックエンド構成のブロックは、バックエンドごとにリージョンとゾーンのリストを提供するために使用されます。ここで指定するリージョンとゾーンの値は、各Kubernetesクラスターノードのラベルのリージョンとゾーンの値と一致している必要があります。これらのリージョンとゾーンは、ストレージクラスで指定できる許容値のリストです。バックエンドで提供されるリージョンとゾーンのサブセットを含むストレージクラスの場合、Tridentは指定されたリージョンとゾーンにボリュームを作成します。詳細については、[を参照してください "CSI トポロジを使用します"](#)。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
supportedTopologies:
- topology.kubernetes.io/region: eastus
  topology.kubernetes.io/zone: eastus-1
- topology.kubernetes.io/region: eastus
  topology.kubernetes.io/zone: eastus-2
```

ストレージクラスの定義

次のようになります StorageClass 定義は、上記のストレージプールを参照してください。

を使用した定義の例 `parameter.selector` フィールド

を使用します `parameter.selector` を指定できます StorageClass ボリュームをホストするために使用される仮想プール。ボリュームには、選択したプールで定義された要素があります。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze"
allowVolumeExpansion: true
```

SMBボリュームの定義例

を使用します `nasType`、``node-stage-secret-name``および``node-stage-secret-namespace``を使用して、SMBボリュームを指定し、必要なActive Directoryクレデンシャルを指定できます。

デフォルト名前スペースの基本設定

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

名前スペースごとに異なるシークレットを使用する

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

ボリュームごとに異なるシークレットを使用する

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```




nasType: smb SMBボリュームをサポートするプールでフィルタリングします。nasType: nfs または nasType: null NFSプールに対してフィルタを適用します。

バックエンドを作成します

バックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

バックエンドの作成に失敗した場合は、バックエンドの設定に何か問題があります。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
```

構成ファイルで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。

Google Cloud NetAppボリューム

Google Cloud NetApp Volumeバックエンドの設定

Google Cloud NetApp VolumesをTridentのバックエンドとして設定できるようになりました。Google Cloud NetApp Volumeバックエンドを使用してNFSボリュームを接続できます。

Google Cloud NetApp Volumesドライバの詳細

Tridentは、クラスタと通信するためのドライバを提供します google-cloud-netapp-volumes。サポートされているアクセスモードは、*ReadWriteOnce(RWO)*、*ReadOnlyMany(ROX)*、*ReadWriteMany(RWX)*、*ReadWriteOncePod(RWOP)*です。

ドライバ	プロトコル	ボリュームモード	サポートされているアクセスモード	サポートされるファイルシステム
google-cloud-netapp-volumes	NFS	ファイルシステム	RWO、ROX、RWX、RWOP	nfs

GKEのクラウドID

クラウドIDを使用すると、Kubernetesポッドは、明示的なGoogle Cloudクレデンシャルを指定するのではなく、ワークロードIDとして認証することで、Google Cloudリソースにアクセスできます。

Google Cloudでクラウドアイデンティティを活用するには、以下が必要です。

- GKEを使用して導入されるKubernetesクラスタ。
- GKEクラスタに設定されたワークロードIDおよびoidc-issuer。
- ワークロードIDを指定 "GCP"、および `cloudIdentity` 指定するを含むTridentがインストールされ

ています `cloudProvider。

Trident オペレータ

Trident演算子を使用してTridentをインストールするには、をに設定し、を `tridentorchestrator_cr.yaml`に`"GCP`設定`cloudProvider`し`cloudIdentity`iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com`ます。`

例：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "GCP"
  cloudIdentity: 'iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com'
```

Helm

次の環境変数を使用して、* cloud-provider (CP) フラグと cloud-identity (CI) *フラグの値を設定します。

```
export CP="GCP"
export ANNOTATION="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

次の例では、環境変数を使用してTridentをインストールし、をGCPに設定し `cloudProvider`、を環境変数を使用 `$ANNOTATION`して`$CP`を設定し`cloudIdentity`ます。`

```
helm install trident trident-operator-100.2406.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

`tridentctl`

次の環境変数を使用して、* cloud provider フラグと cloud identity *フラグの値を設定します。

```
export CP="GCP"
export ANNOTATION="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

次の例では、Tridentをインストールし、フラグをに設定し、`cloud-identity`を`\$ANNOTATION`に`\$CP`設定し`cloud-provider`ます。

```
tridentctl install --cloud-provider=$CP --cloud
-identity="$ANNOTATION" -n trident
```

Google Cloud NetApp Volumeバックエンドを設定する準備

Google Cloud NetApp Volumeバックエンドを設定する前に、次の要件が満たされていることを確認する必要があります。

NFSボリュームノゼンテイジョウケン

Google Cloud NetApp Volumeを初めてまたは新しい場所で使用している場合は、Google Cloud NetApp VolumeをセットアップしてNFSボリュームを作成するために、いくつかの初期設定が必要です。を参照してください ["作業を開始する前に"](#)。

Google Cloud NetApp Volumeバックエンドを設定する前に、次の条件を満たしていることを確認してください。

- Google Cloud NetApp Volumes Serviceで設定されたGoogle Cloudアカウント。を参照してください ["Google Cloud NetAppボリューム"](#)。
- Google Cloudアカウントのプロジェクト番号。を参照してください ["プロジェクトの特定"](#)。
- NetApp Volume Admin) ロールが割り当てられたGoogle Cloudサービスアカウント (roles/netapp.admin。を参照してください ["IDおよびアクセス管理のロールと権限"](#)。
- GCNVアカウントのAPIキーファイル。を参照して ["サービスアカウントキーを作成します"](#)
- ストレージプール。を参照してください ["ストレージプールの概要"](#)。

Google Cloud NetApp Volumeへのアクセスの設定方法の詳細については、を参照してください ["Google Cloud NetApp Volumeへのアクセスをセットアップする"](#)。

Google Cloud NetApp Volumeのバックエンド構成オプションと例

Google Cloud NetApp VolumeのNFSバックエンド構成オプションについて説明し、構成例を確認します。

バックエンド構成オプション

各バックエンドは、1つのGoogle Cloudリージョンにボリュームをプロビジョニングします。他のリージョンにボリュームを作成する場合は、バックエンドを追加で定義します。

パラメータ	説明	デフォルト
「バージョン」		常に 1
'storageDriverName'	ストレージドライバの名前	の値は storageDriverName 「google-cloud-netapp-volumes」と指定する必要があります。

パラメータ	説明	デフォルト
backendName`	(オプション) ストレージバックエンドのカスタム名	ドライバ名 + "_" + API キーの一部
storagePools	ボリューム作成用のストレージプールを指定するオプションのパラメータ。	
「ProjectNumber」	Google Cloud アカウントのプロジェクト番号。この値は、Google Cloudポータルホームページにあります。	
「ロケーション」	TridentがGCNVボリュームを作成するGoogle Cloudの場所。リージョン間Kubernetesクラスタを作成する場合、で作成したボリュームは location、複数のGoogle Cloudリージョンのノードでスケジュールされているワークロードで使用できます。リージョン間トラフィックは追加コストを発生させます。	
「apiKey」と入力します	ロールが割り当てられたGoogle CloudサービスアカウントのAPIキー netapp.admin。このレポートには、Google Cloud サービスアカウントの秘密鍵ファイルのJSON形式のコンテンツが含まれています（バックエンド構成ファイルにそのままコピーされます）。には apiKey、、、の各キーのキーと値のペアを含める必要があります。type project_id client_email client_id auth_uri token_uri auth_provider_x509_cert_url、および client_x509_cert_url。	
「nfsvMountOptions」のように入力します	NFS マウントオプションのきめ細かな制御。	"nfsvers=3 "
「limitVolumeSize」と入力します	要求されたボリュームサイズがこの値を超えている場合はプロビジョニングが失敗します。	""（デフォルトでは適用されません）
「サービスレベル」	ストレージプールとそのボリュームのサービスレベル。値は flex、standard、premium、または `extreme` です。	
「ネットワーク」	GCNVボリュームに使用されるGoogle Cloudネットワーク。	
「バグトレースフラグ」	トラブルシューティング時に使用するデバッグフラグ。例：`{"api":false, "method":true}`トラブルシューティングを行って詳細なログダンプが必要な場合を除き、このオプションは使用しないでください。	null
supportedTopologies	このバックエンドでサポートされているリージョンとゾーンのリストを表します。詳細については、を参照してください "CSI トポロジを使用します" 。例： supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a	

ボリュームのプロビジョニングオプション

では、デフォルトのボリュームプロビジョニングを制御できます `defaults` 構成ファイルのセクション。

パラメータ	説明	デフォルト
「 <code>exportRule</code> 」	新しいボリュームのエクスポートルール。IPv4アドレスの任意の組み合わせをカンマで区切って指定する必要があります。	"0.0.0.0/0 "
「スナップショット方向」	「 <code>.snapshot</code> 」ディレクトリにアクセスします	NFSv4の場合は「 <code>true</code> 」 NFSv3の場合は「 <code>false</code> 」
「スナップショット予約」	Snapshot 用にリザーブされているボリュームの割合	"" (デフォルトの0を使用)
「 <code>unixPermissions</code> 」	新しいボリュームのUNIX権限 (8進数の4桁)。	""

構成例

次の例は、ほとんどのパラメータをデフォルトのままにする基本的な設定を示しています。これは、バックエンドを定義する最も簡単な方法です。


```
-----END PRIVATE KEY-----\n
```

```
---
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '123455380079'
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: '103346282737811234567'
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```



```
version: 1
storageDriverName: google-cloud-netapp-volumes
projectNumber: '123455380079'
location: europe-west6
serviceLevel: premium
storagePools:
- premium-pool1-europe-west6
- premium-pool2-europe-west6
apiKey:
  type: service_account
  project_id: my-gcnv-project
  client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
  client_id: '103346282737811234567'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```



```
znHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
znHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
znHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
XsYg6gyxy4zq7OlwWgLwGa==
-----END PRIVATE KEY-----
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '123455380079'
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: '103346282737811234567'
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: '10'
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
        serviceLevel: extreme
      defaults:
        snapshotReserve: '5'
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
        serviceLevel: premium
    - labels:
```

```
performance: standard
serviceLevel: standard
```

GKEのクラウドID

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

サポートされるトポロジ構成

Tridentを使用すると、リージョンとアベイラビリティゾーンに基づいてワークロード用のボリュームを簡単にプロビジョニングできます。`supportedTopologies`このバックエンド構成のブロックは、バックエンドごとにリージョンとゾーンのリストを提供するために使用されます。ここで指定するリージョンとゾーンの値は、各Kubernetesクラスターノードのラベルのリージョンとゾーンの値と一致している必要があります。これらのリージョンとゾーンは、ストレージクラスで指定できる許容値のリストです。バックエンドで提供されるリージョンとゾーンのサブセットを含むストレージクラスの場合、Tridentは指定されたリージョンとゾーンにボリュームを作成します。詳細については、[を参照してください "CSI トポロジを使用します"](#)。

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
- topology.kubernetes.io/region: asia-east1
  topology.kubernetes.io/zone: asia-east1-a
- topology.kubernetes.io/region: asia-east1
  topology.kubernetes.io/zone: asia-east1-b
```

次の手順

バックエンド構成ファイルを作成したら、次のコマンドを実行します。

```
kubectl create -f <backend-file>
```

バックエンドが正常に作成されたことを確認するには、次のコマンドを実行します。

```
kubectl get tridentbackendconfig
```

NAME	BACKEND NAME	BACKEND UUID
backend-tbc-gcnv	backend-tbc-gcnv	b2fd1ff9-b234-477e-88fd-713913294f65
Bound	Success	

バックエンドの作成に失敗した場合は、バックエンドの設定に何か問題があります。バックエンドについては、コマンドを使用して説明するか、次のコマンドを実行してログを表示して原因を特定できます `kubectl get tridentbackendconfig <backend-name>`。

```
tridentctl logs
```

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

その他の例

ストレージクラスの定義の例

以下は、上記のバックエンドを参照する基本的な定義です StorageClass。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

フィールドを使用した定義例 **parameter.selector** :

を使用する `parameter.selector` と、ボリュームのホストに使用される各に対してを指定できます StorageClass "仮想プール"。ボリュームには、選択したプールで定義された要素があります。

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=extreme"
  backendType: "google-cloud-netapp-volumes"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium"
  backendType: "google-cloud-netapp-volumes"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=standard"
  backendType: "google-cloud-netapp-volumes"

```

ストレージクラスの詳細については、[を参照してください](#) "ストレージクラスを作成する。"。

PVC定義の例PVCテイギノレイ

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc

```

PVCがバインドされているかどうかを確認するには、次のコマンドを実行します。


```
kubectl get pvc gcnv-nfs-pvc
```

NAME	STATUS	VOLUME	CAPACITY
gcnv-nfs-pvc	Bound	pvc-b00f2414-e229-40e6-9b16-ee03eb79a213	100Gi
ACCESS MODES	STORAGECLASS	AGE	
RWX	gcnv-nfs-sc	1m	

Google Cloudバックエンド用にCloud Volumes Service を設定します

提供されている構成例を使用して、TridentインストールのバックエンドとしてNetApp Cloud Volumes Service for Google Cloudを構成する方法を説明します。

Google Cloudドライバの詳細

Tridentは、クラスタと通信するためのドライバを提供します `gcp-cvs`。サポートされているアクセスモードは、*ReadWriteOnce(RWO)*、*ReadOnlyMany(ROX)*、*ReadWriteMany(RWX)*、*ReadWriteOncePod(RWOP)*です。

ドライバ	プロトコル	ボリュームモード	サポートされているアクセスモード	サポートされるファイルシステム
「gcp-cvs」	NFS	ファイルシステム	RWO、ROX、RWX、RWOP	nfs

TridentによるCloud Volumes Service for Google Cloudのサポートの詳細

Tridentでは"[サービスタイプ](#)"、次の2つのいずれかにCloud Volumes Serviceボリュームを作成できます。

- *** CVS-Performance ***：デフォルトのTridentサービスタイプ。パフォーマンスが最適化されたこのサービスタイプは、パフォーマンスを重視する本番環境のワークロードに最適です。CVS -パフォーマンスサービスタイプは、サイズが100GiB以上のボリュームをサポートするハードウェアオプションです。["3つのサービスレベル"](#)次のいずれかを選択できます。
 - standard
 - premium
 - extreme
- *** CVS ***：CVSサービスタイプは、中程度のパフォーマンスレベルに制限された高レベルの可用性を提供します。CVSサービスタイプは、ストレージプールを使用して1GiB未満のボリュームをサポートするソフトウェアオプションです。ストレージプールには最大50個のボリュームを含めることができ、すべてのボリュームでプールの容量とパフォーマンスを共有できます。のいずれかを選択できます ["2つのサービスレベル"](#)：
 - standardsw
 - zoneredundantstandardsw

必要なもの

を設定して使用します ["Cloud Volumes Service for Google Cloud"](#) バックエンドには次のものがが必要です。

- NetApp Cloud Volumes Service で設定されたGoogle Cloudアカウント
- Google Cloud アカウントのプロジェクト番号
- 「netappcloudvolumes .admin」 ロールを持つ Google Cloud サービスアカウント
- Cloud Volumes Service アカウントのAPIキーファイル

バックエンド構成オプション

各バックエンドは、1つの Google Cloud リージョンにボリュームをプロビジョニングします。他のリージョンにボリュームを作成する場合は、バックエンドを追加で定義します。

パラメータ	説明	デフォルト
「バージョン」		常に 1
'storageDriverName'	ストレージドライバの名前	"GCP-cvs"
backendName`	カスタム名またはストレージバックエンド	ドライバ名 + "_" + API キーの一部
'storageClass'	CVSサービスタイプを指定するためのオプションのパラメータ。CVSサービスタイプを選択するために使用し `software` ます。それ以外の場合、TridentはサービスタイプがCVS-Performanceとみなされ(`hardware` ます)。	
storagePools	CVSサービスタイプのみ。ボリューム作成用のストレージプールを指定するオプションのパラメータ。	
「 ProjectNumber 」	Google Cloud アカウントのプロジェクト番号。この値は、Google Cloudポータルホームページにあります。	
「 hostProjectNumber 」	共有VPCネットワークを使用する場合は必須です。このシナリオでは、 projectNumber は、サービスプロジェクトです hostProjectNumber は、ホストプロジェクトです。	
「 apiRegion 」 と入力します	TridentがCloud Volumes Serviceボリュームを作成するGoogle Cloudリージョン。リージョン間Kubernetesクラスタを作成する場合、で作成したボリュームは apiRegion、複数のGoogle Cloudリージョンのノードでスケジュールされているワークロードで使用できます。リージョン間トラフィックは追加コストを発生させます。	
「 apiKey 」 と入力します	を使用したGoogle CloudサービスアカウントのAPIキー netappcloudvolumes.admin ロール。このレポートには、Google Cloud サービスアカウントの秘密鍵ファイルのJSON形式のコンテンツが含まれています (バックエンド構成ファイルにそのままコピーされます)。	

パラメータ	説明	デフォルト
「ProxyURL」と入力します	CVSアカウントへの接続にプロキシサーバが必要な場合は、プロキシURLを指定します。プロキシサーバには、HTTP プロキシまたはHTTPS プロキシを使用できます。HTTPS プロキシの場合、プロキシサーバで自己署名証明書を使用するために証明書の検証はスキップされます。認証が有効になっているプロキシサーバはサポートされていません。	
「nfsvMountOptions」のように入力します	NFS マウントオプションのきめ細かな制御。	"nfsvers=3 "
「limitVolumeSize」と入力します	要求されたボリュームサイズがこの値を超えている場合はプロビジョニングが失敗します。	"" (デフォルトでは適用されません)
「サービスレベル」	新しいボリュームのCVS -パフォーマンスレベルまたはCVSサービスレベル。CVS -パフォーマンスの値は <code>standard</code> 、 <code>premium</code> または <code>extreme</code> です。CVSの値は <code>standardsw</code> または <code>zoneredundantstandardsw</code> 。	CVS -パフォーマンスのデフォルトは「Standard」です。CVSのデフォルトは"standardsw"です。
「ネットワーク」	Cloud Volumes Service ボリュームに使用するGoogle Cloudネットワーク。	デフォルト
「バグトレースフラグ」	トラブルシューティング時に使用するデバッグフラグ。例: <code>\{"api":false, "method":true\}</code> 。トラブルシューティングを行って詳細なログダンプが必要な場合を除き、このオプションは使用しないでください。	null
allowedTopologies	クロスリージョンアクセスを有効にするには、のStorageClass定義を使用します allowedTopologies すべてのリージョンを含める必要があります。例： - key: topology.kubernetes.io/region values: - us-east1 - europe-west1	

ボリュームのプロビジョニングオプション

では、デフォルトのボリュームプロビジョニングを制御できます defaults 構成ファイルのセクション。

パラメータ	説明	デフォルト
「exportRule」	新しいボリュームのエクスポートルール。CIDR 表記のIPv4 アドレスまたはIPv4 サブネットの任意の組み合わせをカンマで区切って指定する必要があります。	"0.0.0.0/0 "
「スナップショット方向」	「.snapshot」ディレクトリにアクセスします	いいえ
「スナップショット予約」	Snapshot 用にリザーブされているボリュームの割合	"" (CVS のデフォルト値をそのまま使用)

パラメータ	説明	デフォルト
「size」	新しいボリュームのサイズ。CVS - パフォーマンス最小値は100GiBです。CVS最小値は1GiBです。	CVS -パフォーマンスサービスのタイプはデフォルトで「100GiB」です。CVSサービスのタイプではデフォルトが設定されませんが、1GiB以上が必要です。

CVS -パフォーマンスサービスの種類の例

次の例は、CVS -パフォーマンスサービスタイプの設定例を示しています。

例 1：最小限の構成

これは、デフォルトの「標準」サービスレベルでデフォルトのCVSパフォーマンスサービスタイプを使用する最小バックエンド構成です。

```

---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com

```

例2：サービスレベルの設定

この例は、サービスレベルやボリュームのデフォルトなど、バックエンド構成オプションを示しています。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
proxyURL: http://proxy-server-hostname/
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 10Ti
serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '5'
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  size: 5Ti
```

例3：仮想プールの構成

この例では、を使用します storage 仮想プールおよびを設定します StorageClasses それはそれらを再度参照する。を参照してください [\[ストレージクラスの定義\]](#) をクリックして、ストレージクラスの定義方法を確認します。

ここでは、すべての仮想プールに対して特定のデフォルトが設定され、すべての仮想プールに対してが設定されます snapshotReserve 5%およびである exportRule を0.0.0.0/0に設定します。仮想プールは、で定義されます storage セクション。個々の仮想プールにはそれぞれ独自の定義があります serviceLevel をクリックすると、一部のプールでデフォルト値が上書きされます。プールを区別するために、仮想プールのラベルを使用しました `performance` および `protection`。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
nfsMountOptions: vers=3,proto=tcp,timeo=600
defaults:
  snapshotReserve: '5'
  exportRule: 0.0.0.0/0
labels:
  cloud: gcp
region: us-west2
storage:
- labels:
  performance: extreme
  protection: extra
  serviceLevel: extreme
```

```

defaults:
  snapshotDir: 'true'
  snapshotReserve: '10'
  exportRule: 10.0.0.0/24
- labels:
  performance: extreme
  protection: standard
  serviceLevel: extreme
- labels:
  performance: premium
  protection: extra
  serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '10'
- labels:
  performance: premium
  protection: standard
  serviceLevel: premium
- labels:
  performance: standard
  serviceLevel: standard

```

ストレージクラスの定義

次のStorageClass定義は、仮想プールの構成例に適用されます。を使用します `parameters.selector` では、ボリュームのホストに使用する仮想プールをストレージクラスごとに指定できます。ボリュームには、選択したプールで定義された要素があります。

ストレージクラスの例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=standard"
```



```
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

- 最初のストレージクラス (cvs-extreme-extra-protection) を最初の仮想プールにマッピングします。スナップショット予約が 10% の非常に高いパフォーマンスを提供する唯一のプールです。
- 最後のStorageClass(cvs-extra-protection) は、10%のスナップショットリザーブを提供するストレージプールを呼び出します。Tridentは、選択する仮想プールを決定し、スナップショット予約の要件を確実に満たします。

CVSサービスタイプの例

次の例は、CVSサービスタイプの設定例を示しています。

例1：最小構成

これは、を使用するバックエンドの最小構成です storageClass CVSサービスタイプとデフォルトを指定するには standardsw サービスレベル：

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
storageClass: software
apiRegion: us-east4
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
serviceLevel: standardsw
```

例2：ストレージプールの構成

このバックエンド設定の例では、を使用して storagePools ストレージプールを設定します。

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
data.iam.gserviceaccount.com
  client_id: '107071413297115343396'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

次の手順

バックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

バックエンドの作成に失敗した場合は、バックエンドの設定に何か問題があります。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
```

構成ファイルで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。

NetApp HCI または SolidFire バックエンドを設定します

Trident環境でElementバックエンドを作成して使用方法について説明します。

Element ドライバの詳細

Tridentは、クラスタと通信するためのストレージドライバを提供します `solidfire-san`。サポートされているアクセスモードは、`ReadWriteOnce(RWO)`、`ReadOnlyMany(ROX)`、`ReadWriteMany(RWX)`、`ReadWriteOncePod(RWOP)`です。

```
`solidfire-san`ストレージドライバは、  
_file_and_block_volumeモードをサポートしています。volumeModeの場合  
`Filesystem`、Tridentはボリュームを作成し、ファイルシステムを作成します。ファイルシ  
ステムのタイプは StorageClass で指定されます。
```

ドライバ	プロトコル	ボリュームモード	サポートされているアクセスモード	サポートされるファイルシステム
「olidfire -san」	iSCSI	ブロック	RWO、ROX、RWX、RWOP	ファイルシステムがありません。raw ブロックデバイスです。
「olidfire -san」	iSCSI	ファイルシステム	RWO、RWOP	「xfs」、「ext3」、「ext4」

作業を開始する前に

Elementバックエンドを作成する前に、次の情報が必要になります。

- Element ソフトウェアを実行する、サポート対象のストレージシステム。
- NetApp HCI / SolidFire クラスタ管理者またはボリュームを管理できるテナントユーザのクレデンシャル。
- すべての Kubernetes ワーカーノードに適切な iSCSI ツールをインストールする必要があります。を参照してください "[ワーカーノードの準備情報](#)"。

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
「バージョン」		常に 1

パラメータ	説明	デフォルト
'storageDriverName'	ストレージドライバの名前	常に「solidfire-san-」
backendName`	カスタム名またはストレージバックエンド	「iSCSI_」 + ストレージ (iSCSI) IP アドレス SolidFire
「エンドポイント」	テナントのクレデンシャルを使用する SolidFire クラスターの MVIP	
「VIP」	ストレージ (iSCSI) の IP アドレスとポート	
「ラベル」	ボリュームに適用する任意の JSON 形式のラベルのセット。	「」
「tenantname」	使用するテナント名 (見つからない場合に作成)	
「InitiatorIFCace」	iSCSI トラフィックを特定のホストインターフェイスに制限します	デフォルト
UseCHAP'	CHAPを使用してiSCSIを認証します。TridentはCHAPを使用します。	正しいです
「アクセスグループ」	使用するアクセスグループ ID のリスト	「trident」という名前のアクセスグループの ID を検索します。
「タイプ」	QoS の仕様	
「limitVolumeSize」と入力します	要求されたボリュームサイズがこの値を超えている場合、プロビジョニングが失敗します	"" (デフォルトでは適用されません)
「バグトレースフラグ」	トラブルシューティング時に使用するデバッグフラグ。例: {"API": false、"method": true}	null



トラブルシューティングを行い、詳細なログダンプが必要な場合を除き、「ebugTraceFlags」は使用しないでください。

例1：のバックエンド構成 solidfire-san 3種類のボリュームを備えたドライバ

次の例は、CHAP 認証を使用するバックエンドファイルと、特定の QoS 保証を適用した 3 つのボリュームタイプのモデリングを示しています。その場合 'ストレージ・クラスを定義して 'iops`storage クラス・パラメータを使用します

```

---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"
TenantName: "<tenant>"
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000

```

例2：のバックエンドとストレージクラスの設定 solidfire-san 仮想プールを備えたドライバ

この例は、仮想プールとともに、それらを参照するStorageClassesとともに構成されているバックエンド定義ファイルを示しています。

ストレージプールに存在するラベルを、プロビジョニング時にバックエンドストレージLUNにコピーしますTrident。ストレージ管理者は、仮想プールごとにラベルを定義したり、ボリュームをラベルでグループ化したりできます。

以下に示すバックエンド定義ファイルの例では、すべてのストレージプールに対して特定のデフォルトが設定されています。これにより、が設定されます type シルバー。仮想プールは、で定義されます storage セクション。この例では、一部のストレージプールが独自のタイプを設定し、一部のプールが上記のデフォルト値を上書きします。

```

---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"

```

```
TenantName: "<tenant>"
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
- labels:
  performance: gold
  cost: '4'
  zone: us-east-1a
  type: Gold
- labels:
  performance: silver
  cost: '3'
  zone: us-east-1b
  type: Silver
- labels:
  performance: bronze
  cost: '2'
  zone: us-east-1c
  type: Bronze
- labels:
  performance: silver
  cost: '1'
  zone: us-east-1d
```

次のStorageClass定義は、上記の仮想プールを参照しています。を使用する `parameters.selector` 各ストレージクラスは、ボリュームのホストに使用できる仮想プールを呼び出します。ボリュームには、選択した仮

想プール内で定義された要素があります。

最初のStorageClass(solidfire-gold-four) が最初の仮想プールにマッピングされます。これは、ゴールドのパフォーマンスとゴールドのパフォーマンスを提供する唯一のプールです Volume Type QoS。最後のStorageClass(solidfire-silver) は、Silverパフォーマンスを提供するストレージプールを呼び出します。Tridentが選択する仮想プールを決定し、ストレージ要件が満たされるようにします。


```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold; cost=4"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=3"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze; cost=2"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=1"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
  fsType: "ext4"
```

詳細については、こちらをご覧ください

- ["ボリュームアクセスグループ"](#)

ONTAP SAN ドライバ

ONTAP SAN ドライバの概要

ONTAP および Cloud Volumes ONTAP SAN ドライバを使用した ONTAP バックエンドの設定について説明します。

ONTAP SAN ドライバの詳細

Tridentは、ONTAPクラスタと通信するための次のSANストレージドライバを提供します。サポートされているアクセスモードは、*ReadWriteOnce*(RWO)、*ReadOnlyMany*(ROX)、*ReadWriteMany*(RWX)、*ReadWriteOncePod*(RWOP)です。

ドライバ	プロトコル	ボリュームモード	サポートされているアクセスモード	サポートされるファイルシステム
「ontap - san」	iSCSI	ブロック	RWO、ROX、RWX、RWOP	ファイルシステムなし。rawブロックデバイスです
「ontap - san」	iSCSI	ファイルシステム	RWO、RWOP ROXおよびRWXは、ファイルシステムボリュームモードでは使用できません。	「xfs」、「ext3」、「ext4」
「ontap - san」	NVMe/FC	ブロック	RWO、ROX、RWX、RWOP	ファイルシステムなし。rawブロックデバイスです
	を参照してください NVMe/TCP に関するその他の考慮事項。			
「ontap - san」	NVMe/FC	ファイルシステム	RWO、RWOP ROXおよびRWXは、ファイルシステムボリュームモードでは使用できません。	「xfs」、「ext3」、「ext4」
	を参照してください NVMe/TCP に関するその他の考慮事項。			

ドライバ	プロトコル	ボリュームモード	サポートされているアクセスモード	サポートされるファイルシステム
「ONTAP - SAN - エコノミー」	iSCSI	ブロック	RWO、ROX、RWX、RWOP	ファイルシステムなし。rawブロックデバイスです
「ONTAP - SAN - エコノミー」	iSCSI	ファイルシステム	RWO、RWOP ROXおよびRWXは、ファイルシステムボリュームモードでは使用できません。	「xfs」、「ext3」、「ext4」



- 使用 `ontap-san-economy` 永続的ボリュームの使用数が次の値よりも多いと予想される場合のみ "サポートされるONTAPの制限"。
- 使用 `ontap-nas-economy` 永続的ボリュームの使用数が次の値よりも多いと予想される場合のみ "サポートされるONTAPの制限" および `ontap-san-economy` ドライバは使用できません。
- 使用しないでください `ontap-nas-economy` データ保護、ディザスタリカバリ、モビリティのニーズが予想される場合。

ユーザ権限

Tridentは、ONTAP管理者またはSVM管理者（通常はクラスタユーザ、`vsadmin`SVMユーザ`、または別の名前で同じロールのユーザを使用）として実行することを想定しています ``admin`。Amazon FSx for NetApp ONTAP環境では、Tridentは、クラスタユーザまたは `vsadmin`SVMユーザ` を使用するONTAP管理者またはSVM管理者、または同じロールの別の名前のユーザとして実行される必要があります ``fsxadmin`。この ``fsxadmin`` ユーザは、クラスタ管理者ユーザに代わる限定的なユーザです。



パラメータを使用する場合は `limitAggregateUsage`、クラスタ管理者の権限が必要です。TridentでAmazon FSx for NetApp ONTAPを使用している場合、`limitAggregateUsage`` パラメータはユーザアカウントと ``fsxadmin`` ユーザアカウントでは機能しません ``vsadmin`。このパラメータを指定すると設定処理は失敗します。

ONTAP内でTridentドライバが使用できる、より制限の厳しいロールを作成することは可能ですが、推奨しません。Tridentの新リリースでは、多くの場合、考慮すべきAPIが追加で必要になるため、アップグレードが難しく、エラーも起こりやすくなります。

NVMe/TCPに関するその他の考慮事項

Tridentは、次のドライバを使用してNon-Volatile Memory Express (NVMe) プロトコルをサポートします `ontap-san`。

- IPv6
- NVMeボリュームのSnapshotとクローン
- NVMeボリュームのサイズ変更
- Tridentの外部で作成されたNVMeボリュームをインポートして、そのライフサイクルをTridentで管理でき

るようになる

- NVMeネイティブマルチパス
- Kubernetesノードのグレースフルシャットダウンまたはグレースフルシャットダウン (24.06)

Tridentは以下をサポートしていません。

- NVMeでネイティブにサポートされるDH-HMAC-CHAP
- Device Mapper (DM; デバイスマッパー) マルチパス
- LUKS暗号化

バックエンドに**ONTAP SAN**ドライバを設定する準備をします

ONTAP SANドライバでONTAPバックエンドを構成するための要件と認証オプションを理解します。

要件

すべてのONTAPバックエンドについて、TridentでSVMにアグリゲートを少なくとも1つ割り当てる必要があります。

複数のドライバを実行し、1つまたは複数のドライバを参照するストレージクラスを作成することもできます。たとえば 'ONTAP-SAN' ドライバを使用する「-dev」クラスと 'ONTAP-SAN-エコノミー' 'one' を使用する「デフォルト」クラスを設定できます

すべてのKubernetesワーカーノードに適切なiSCSIツールをインストールしておく必要があります。を参照してください ["ワーカーノードを準備します"](#) を参照してください。

ONTAPバックエンドの認証

Tridentには、ONTAPバックエンドの認証に2つのモードがあります。

- **credential based** : 必要な権限を持つ ONTAP ユーザのユーザ名とパスワード。ONTAP バージョンとの互換性を最大限に高めるために 'admin' または vsadmin などの事前定義されたセキュリティ・ログイン・ロールを使用することを推奨します
- **証明書ベース** : Tridentは、バックエンドにインストールされている証明書を使用してONTAPクラスタと通信することもできます。この場合、バックエンド定義には、Base64 でエンコードされたクライアント証明書、キー、および信頼された CA 証明書 (推奨) が含まれている必要があります。

既存のバックエンドを更新して、クレデンシャルベースの方式と証明書ベースの方式を切り替えることができます。ただし、一度にサポートされる認証方法は1つだけです。別の認証方式に切り替えるには、バックエンド設定から既存の方式を削除する必要があります。



クレデンシャルと証明書の両方を*指定しようとする、バックエンドの作成が失敗し、構成ファイルに複数の認証方法が指定されているというエラーが表示されます。

クレデンシャルベースの認証を有効にします

TridentがONTAPバックエンドと通信するには、SVMを対象としたクラスタを対象とした管理者に対するクレデンシャルが必要です。や vsadmin`などの事前定義された標準のロールを使用することを推奨します

`admin。これにより、今後のONTAPリリースで使用する機能APIが公開される可能性がある将来のTridentリリースとの前方互換性が確保されます。Tridentでは、カスタムのセキュリティログインロールを作成して使用できますが、推奨されません。

バックエンド定義の例は次のようになります。

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

バックエンド定義は、クレデンシャルがプレーンテキストで保存される唯一の場所であることに注意してください。バックエンドが作成されると、ユーザ名とパスワードが Base64 でエンコードされ、Kubernetes シークレットとして格納されます。クレデンシャルの知識が必要なのは、バックエンドの作成または更新だけです。この処理は管理者専用で、Kubernetes / ストレージ管理者が実行します。

証明書ベースの認証を有効にします

新規または既存のバックエンドは証明書を使用して ONTAP バックエンドと通信できます。バックエンド定義には 3 つのパラメータが必要です。

- `clientCertificate` : Base64 でエンコードされたクライアント証明書の値。
- `clientPrivateKey` : Base64 でエンコードされた、関連付けられた秘密鍵の値。
- `trustedCACertificate`: 信頼された CA 証明書の Base64 エンコード値。信頼された CA を使用する場合は、このパラメータを指定する必要があります。信頼された CA が使用されていない場合は無視してかまいません。

一般的なワークフローは次の手順で構成されます。

手順

1. クライアント証明書とキーを生成します。生成時に、ONTAP ユーザとして認証するように Common Name (CN ; 共通名) を設定します。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key  
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. 信頼された CA 証明書を ONTAP クラスタに追加します。この処理は、ストレージ管理者がすでに行っている可能性があります。信頼できる CA が使用されていない場合は無視します。

```
security certificate install -type server -cert-name <trusted-ca-cert-  
name> -vserver <vserver-name>  
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled  
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca  
<cert-authority>
```

3. ONTAP クラスタにクライアント証明書とキーをインストールします (手順 1)。

```
security certificate install -type client-ca -cert-name <certificate-  
name> -vserver <vserver-name>  
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. ONTAP セキュリティ・ログイン・ロールが 'cert' 認証方式をサポートしていることを確認します

```
security login create -user-or-group-name admin -application ontapi  
-authentication-method cert  
security login create -user-or-group-name admin -application http  
-authentication-method cert
```

5. 生成された証明書を使用して認証をテスト ONTAP 管理 LIF > と <vserver name> は、管理 LIF の IP アドレスおよび SVM 名に置き換えてください。

```
curl -X POST -Lk https://<ONTAP-Management-  
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key  
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp  
xmlns="http://www.netapp.com/filer/admin" version="1.21"  
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Base64 で証明書、キー、および信頼された CA 証明書をエンコードする。

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 前の手順で得た値を使用してバックエンドを作成します。

```
cat cert-backend.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "SanBackend",
  "managementLIF": "1.2.3.4",
  "svm": "vserver_test",
  "clientCertificate": "Faaaakkkkeeee...Vaaalllluuueeeee",
  "clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
  "trustedCACertificate": "QNFinfO...SiqOyN",
  "storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |         0 |
+-----+-----+-----+-----+
+-----+-----+
```

認証方法を更新するか、クレデンシャルをローテーションして

既存のバックエンドを更新して、別の認証方法を使用したり、クレデンシャルをローテーションしたりできます。これはどちらの方法でも機能します。ユーザ名とパスワードを使用するバックエンドは証明書を使用するように更新できますが、証明書を使用するバックエンドはユーザ名とパスワードに基づいて更新できます。これを行うには、既存の認証方法を削除して、新しい認証方法を追加する必要があります。次に必要なパラメータを含む更新されたbackend.jsonファイルを使用してtridentctl backend updateを実行します

```

cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |          UUID          |
STATE | VOLUMES |
+-----+-----+-----+
+-----+-----+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |      9 |
+-----+-----+-----+
+-----+-----+

```



パスワードのローテーションを実行する際には、ストレージ管理者が最初に ONTAP でユーザのパスワードを更新する必要があります。この後にバックエンドアップデートが続きます。証明書のローテーションを実行する際に、複数の証明書をユーザに追加することができます。その後、バックエンドが更新されて新しい証明書が使用されるようになります。この証明書に続く古い証明書は、ONTAP クラスタから削除できます。

バックエンドを更新しても、すでに作成されているボリュームへのアクセスは中断されず、その後のボリューム接続にも影響しません。バックエンドの更新が成功すると、TridentがONTAPバックエンドと通信し、以降のボリューム処理を処理できるようになります。

Trident用のカスタムONTAPロールの作成

Tridentで処理を実行するためにONTAP adminロールを使用する必要がないように、最小Privilegesを持つONTAPクラスタロールを作成できます。Tridentバックエンド構成にユーザ名を含めると、Trident作成したONTAPクラスタロールが使用されて処理が実行されます。

Tridentカスタムロールの作成の詳細については、を参照してください["Tridentカスタムロールジェネレータ"](#)。

ONTAP CLIノシヨウ

1. 次のコマンドを使用して新しいロールを作成します。

```
security login role create <role_name\> -cmddirname "command" -access all  
-vserver <svm_name\>
```

2. Tridentユーザのユーザ名を作成します。

```
security login create -username <user_name\> -application ontapi  
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver  
<svm_name\> -comment "user_description"
```

3. ユーザにロールをマッピングします。

```
security login modify username <user_name\> -vserver <svm_name\> -role  
<role_name\> -application ontapi -application console -authmethod  
<password\>
```

System Managerの使用

ONTAPシステムマネージャで、次の手順を実行します。

1. カスタムロールの作成：

- a. クラスタレベルでカスタムロールを作成するには、*[クラスタ]>[設定]*を選択します。

(または) SVMレベルでカスタムロールを作成するには、*[ストレージ]>[Storage VM]>[設定]>[ユーザとロール]*を選択し `required SVM` ます。

- b. の横にある矢印アイコン (→*) を選択します。
- c. [Roles]*で[+Add]*を選択します。
- d. ロールのルールを定義し、*[保存]*をクリックします。

2. ロールを **Trident** ユーザにマップする:+[ユーザとロール]ページで次の手順を実行します。

- a. で[アイコンの追加]*を選択します。
- b. 必要なユーザ名を選択し、* Role *のドロップダウンメニューでロールを選択します。
- c. [保存 (Save)] をクリックします。

詳細については、次のページを参照してください。

- ["ONTAPの管理用のカスタムロール"または"カスタムロールの定義"](#)
- ["ロールとユーザを使用する"](#)

双方向 **CHAP** を使用して接続を認証します

Tridentでは、ドライバと `ontap-san-economy`` ドライバの双方向CHAPを使用してiSCSIセッションを認証できます `ontap-san`。これには、バックエンド定義でオプションを有効にする必要があります `useCHAP` ます。に設定する `true` と、TridentはSVMのデフォルトのイニシエータセキュリティを双方向CHAPに設定し、

ユーザ名とシークレットをバックエンドファイルに設定します。接続の認証には双方向 CHAP を使用することを推奨します。次の設定例を参照してください。

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
```



「useCHAP」パラメータは、1 回だけ設定できるブール型のオプションです。デフォルトでは false に設定されています。true に設定したあとで、false に設定することはできません。

「useCHAP=true」に加えて、「chapInitiatorSecret」、「chapTargetInitiatorSecret」、「chapTargetUsername」、および「chapUsername」フィールドもバックエンド定義に含める必要があります。シークレットは 'tridentctl update を実行してバックエンドを作成した後に変更できます

動作の仕組み

true に設定する `useCHAP` と、ストレージ管理者は Trident にストレージバックエンドで CHAP を構成するように指示します。これには次のものが含まれます。

- SVM で CHAP をセットアップします。
 - SVM のデフォルトのイニシエータセキュリティタイプが none（デフォルトで設定）*で、*ボリュームに既存の LUN がない場合、Trident はデフォルトのセキュリティタイプをに設定し CHAP、CHAP イニシエータとターゲットのユーザ名とシークレットの設定に進みます。
 - SVM に LUN が含まれている場合、Trident は SVM で CHAP を有効にしません。これにより、SVM にすでに存在する LUN へのアクセスが制限されなくなります。
- CHAP イニシエータとターゲットのユーザ名とシークレットを設定します。これらのオプションは、バックエンド構成で指定する必要があります（上記を参照）。

バックエンドが作成されると、Trident は対応する CRD を作成し tridentbackend、CHAP シークレットとユーザ名を Kubernetes シークレットとして格納します。このバックエンドで Trident によって作成されたすべての PVS がマウントされ、CHAP 経由で接続されます。

クレデンシャルをローテーションし、バックエンドを更新

CHAP 証明書を更新するには 'backend.json ファイルの CHAP パラメータを更新しますこれには 'CHAP シークレットを更新し 'tridentctl update コマンドを使用してこれらの変更を反映する必要があります



バックエンドのCHAPシークレットを更新する場合は、を使用してバックエンドを更新する必要があります `tridentctl`。Tridentではこれらの変更を反映できないため、CLI / ONTAP UIでストレージクラスタのクレデンシャルを更新しないでください。

```
cat backend-san.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "ontap_san_chap",
  "managementLIF": "192.168.0.135",
  "svm": "ontap_iscsi_svm",
  "useCHAP": true,
  "username": "vsadmin",
  "password": "password",
  "chapInitiatorSecret": "cl9qxUpDaTeD",
  "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLsd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
|  NAME          | STORAGE DRIVER |                               UUID                               |
STATE | VOLUMES |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

既存の接続は影響を受けず、SVM上のTridentによってクレデンシャルが更新されてもアクティブなままです。新しい接続では更新されたクレデンシャルが使用され、既存の接続は引き続きアクティブになります。古いPVSを切断して再接続すると、更新されたクレデンシャルが使用されます。

ONTAP のSAN構成オプションと例


Tridentのインストール時にONTAP SANドライバを作成して使用方法について説明します。このセクションでは、バックエンドの構成例と、バックエンドをStorageClassesにマッピングするための詳細を示します。

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
「バージョン」		常に 1
'storageDriverName'	ストレージドライバの名前	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy
backendName`	カスタム名またはストレージバックエンド	ドライバ名+"_" + dataLIF
「管理 LIF」	クラス管理LIFまたはSVM管理LIFのIPアドレス。Fully Qualified Domain Name (FQDN; 完全修飾ドメイン名) を指定できます。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。シームレスなMetroClusterスイッチオーバーについては、を参照して [mcc-best] ください。	「10.0.0.1」、 「[2001:1234:abcd::fe]」
「重複排除	プロトコル LIF の IP アドレス。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。* iSCSIの場合は指定しないでください。 Trident は、を使用して" ONTAP の選択的LUNマップ "、マルチパスセッションの確立に必要なiSCSI LIFを検出します。が明示的に定義されている場合は、警告が生成され`dataLIF`ます。MetroClusterの場合は省略してください。*を参照してください [mcc-best] 。	SVMの派生物です
'VM'	使用する Storage Virtual Machine * MetroClusterの場合は省略してください。* [mcc-best] 。	SVM 「管理 LIF」 が指定されている場合に生成されます
「 useCHAP 」	CHAPを使用してONTAP SANドライバのiSCSIを認証します (プリアン)。バックエンドで指定されたSVMのデフォルト認証として双方向CHAPを設定して使用する場合は、Tridentのをに設定し`true`ます。詳細については、を参照してください " バックエンドにONTAP SANドライバを設定する準備をします "。	「偽」
「 chapInitiatorSecret 」	CHAP イニシエータシークレット。「 useCHAP = TRUE 」の場合は必須	""
「ラベル」	ボリュームに適用する任意の JSON 形式のラベルのセット	""
「 chapTargetInitiatorSecret 」	CHAP ターゲットイニシエータシークレット。「 useCHAP = TRUE 」の場合は必須	""

パラメータ	説明	デフォルト
「chapUsername」	インバウンドユーザ名。「useCHAP = TRUE」の場合は必須	""
「chapTargetUsername」	ターゲットユーザ名。「useCHAP = TRUE」の場合は必須	""
「clientCertificate」をクリックします	クライアント証明書の Base64 エンコード値。証明書ベースの認証に使用されます	""
「clientPrivateKey」	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます	""
「trustedCacertificate」	信頼された CA 証明書の Base64 エンコード値。任意。証明書ベースの認証に使用されます。	""
「ユーザ名」	ONTAP クラスタとの通信に必要なユーザ名。クレデンシャルベースの認証に使用されます。	""
「password」と入力します	ONTAP クラスタとの通信にパスワードが必要です。クレデンシャルベースの認証に使用されます。	""
'VM'	使用する Storage Virtual Machine	SVM 「管理 LIF」が指定されている場合に生成されます
'storagePrefix'	SVM で新しいボリュームをプロビジョニングする際に使用するプレフィックスを指定します。あとから変更することはできません。このパラメータを更新するには、新しいバックエンドを作成する必要があります。	trident

パラメータ	説明	デフォルト
「集約」	<p>プロビジョニング用のアグリゲート（オプション。設定する場合は SVM に割り当てる必要があります）。ドライバの場合 <code>ontap-nas-flexgroup</code>、このオプションは無視されます。割り当てられていない場合は、使用可能ないずれかのアグリゲートを使用して FlexGroup ボリュームをプロビジョニングできます。</p> <p> SVMでアグリゲートが更新されると、Tridentコントローラを再起動せずにSVMをポーリングすることで、Tridentでアグリゲートが自動的に更新されます。ボリュームをプロビジョニングするようにTridentで特定のアグリゲートを設定している場合、アグリゲートの名前を変更するかSVMから移動すると、SVMアグリゲートのポーリング中にTridentでバックエンドが障害状態になります。アグリゲートをSVMにあるアグリゲートに変更するか、アグリゲートを完全に削除してバックエンドをオンラインに戻す必要があります。</p>	""
「 <code>AggreglimitateUsage</code> 」と入力します	<p>使用率がこの割合を超えている場合は、プロビジョニングが失敗します。Amazon FSx for NetApp ONTAP バックエンドを使用している場合は、<code>limitAggregateUsage`</code>を指定しないでください。指定された <code>`vsadmin`</code> には <code>`fsxadmin`</code>、アグリゲートの使用量を取得してTridentを使用して制限するために必要な権限が含まれていません。</p>	""（デフォルトでは適用されません）
「 <code>limitVolumeSize</code> 」と入力します	<p>要求されたボリュームサイズがこの値を超えている場合、プロビジョニングが失敗します。また、LUNで管理するボリュームの最大サイズも制限します。</p>	""（デフォルトでは適用されません）
<code>'lunsPerFlexvol</code>	<p>FlexVol あたりの最大 LUN 数。有効な範囲は 50、200 です</p>	100
「バグトレースフラグ」	<p>トラブルシューティング時に使用するデバッグフラグ。例： <code>{"api" : false, "method" : true}</code></p> <p>トラブルシューティングを行い、詳細なログダンプが必要な場合を除き、は使用しないでください。</p>	null

パラメータ	説明	デフォルト
「useREST」	<p>ONTAP REST API を使用するためのブーリアンパラメータ。</p> <p>useREST`に設定する `true`と、Tridentはバックエンドとの通信にONTAP REST APIを使用します。に設定する `false`と、Tridentはバックエンドとの通信にONTAP ZAPI呼び出しを使用します。この機能にはONTAP 9.11.1以降が必要です。また、使用するONTAPログインロールには、アプリケーションへのアクセス権が必要です `ontap`。これは、事前に定義された役割と役割によって実現され vsadmin cluster-admin ます。Trident 24.06リリースおよびONTAP 9.15.1以降では、useREST`がデフォルトで設定されて `true`います。ONTAP ZAPI呼び出しを使用するには、を `false`変更してください。</p> <p>`useREST` useREST はNVMe/TCPに完全修飾されています。</p>	true ONTAP 9.15.1以降の場合は、それ以外の場合は false。
sanType	<p>iSCSI、nvme`NVMe/TCP、または `fc`SCSI over Fibre Channel (FC ; SCSI over Fibre Channel) に対してを選択します `iscsi`。「FCP」(SCSI over FC) は、Trident 24.10リリースの技術プレビュー機能です。</p>	iscsi 空白の場合
formatOptions	<p>を使用して、`formatOptions`コマンドのコマンドライン引数を指定します。この引数 `mkfs`は、ボリュームがフォーマットされるたびに適用されます。これにより、好みに応じてボリュームをフォーマットできます。デバイスパスを除いて、mkfsコマンドオプションと同様にformatOptionsを指定してください。例：「-E nodiscard」</p> <ul style="list-style-type: none"> • `ontap-san`および `ontap-san-economy`ドライバでのみサポートされています。* 	
limitVolumePoolSize	<p>ONTAP SANエコノミーバックエンドでLUNを使用する場合の、要求可能な最大FlexVolサイズ。</p>	"" (デフォルトでは適用されません)
denyNewVolumePools	<p>バックエンドがLUNを格納するために新しいFlexVolボリュームを作成することを制限します ontap-san-economy。新しいPVのプロビジョニングには、既存のFlexVolのみが使用されます。</p>	

formatOptionsの使用に関する推奨事項

Tridentでは、フォーマット処理を高速化するために、次のオプションを推奨しています。

-E nodiscard :

- keep : mkfsの時点でブロックを破棄しないでください (ブロックの破棄は、最初はソリッドステートデバイスやスパーズ/シンプロビジョニングされたストレージで有効です)。これは廃止されたオプション「-K」に代わるもので、すべてのファイルシステム (xfs、ext3、およびext4) に適用できます。

ボリュームのプロビジョニング用のバックエンド構成オプション

これらのオプションを使用して、のデフォルトプロビジョニングを制御できます defaults 設定のセクション。例については、以下の設定例を参照してください。

パラメータ	説明	デフォルト
「平和の配分」	space-allocation for LUN のコマンドを指定します	"正しい"
「平和のための準備」を参照してください	スペースリザーベーションモード：「none」（シン） または「volume」（シック）	"なし"
「ナップショットポリシー」	使用する Snapshot ポリシー	"なし"
「QOSPolicy」	作成したボリュームに割り当てる QoS ポリシーグループ。ストレージプール/バックエンドごとに QOSPolicy または adaptiveQosPolicy のいずれかを選択します。TridentでQoSポリシーグループを使用するには、ONTAP 9.8以降が必要です。共有されていないQoSポリシーグループを使用し、ポリシーグループが各コンスチチュエントに個別に適用されるようにします。QoSポリシーグループを共有すると、すべてのワークロードの合計スループットの上限が適用されます。	""
「adaptiveQosPolicy」を参照してください	アダプティブ QoS ポリシーグループ：作成したボリュームに割り当てます。ストレージプール/バックエンドごとに QOSPolicy または adaptiveQosPolicy のいずれかを選択します	""
「スナップショット予約」	Snapshot 用にリザーブされているボリュームの割合	次の場合は「0」 snapshotPolicy は「none」、 それ以外の場合は「」です。
'plitOnClone	作成時にクローンを親からスプリットします	いいえ
「暗号化」	新しいボリュームでNetApp Volume Encryption（NVE）を有効にします。デフォルトはです。`false`このオプションを使用するには、クラスタで NVE のライセンスが設定され、有効になっている必要があります。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でNAEが有効になります。詳細については、を参照してください" TridentとNVEおよびNAEとの連携 "。	いいえ
luksEncryption	LUKS暗号化を有効にします。を参照してください" Linux Unified Key Setup（LUKS；統合キーセットアップ）を使用 "。 LUKS暗号化はNVMe/TCPではサポートされません。	""
'ecurityStyle'	新しいボリュームのセキュリティ形式	unix
階層ポリシー	「none」を使用する階層化ポリシー	ONTAP 9.5より前のSVM-DR設定の場合は「snapshot-only」

パラメータ	説明	デフォルト
nameTemplate	カスタムボリューム名を作成するためのテンプレート。	""

ボリュームプロビジョニングの例

デフォルトが定義されている例を次に示します。

```

---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'

```



ドライバを使用して作成されたすべてのボリュームについて、`ontap-san` TridentはLUNメタデータに対応するために10%の容量をFlexVolに追加します。LUNは、ユーザがPVCで要求したサイズとまったく同じサイズでプロビジョニングされます。Tridentは、FlexVolに10%を追加します（ONTAPでは使用可能なサイズとして表示されます）。ユーザには、要求した使用可能容量が割り当てられます。また、利用可能なスペースがフルに活用されていないかぎり、LUNが読み取り専用になることもありません。これは、ONTAPとSANの経済性には該当しません。

定義されたバックエンドの場合 snapshotReserve、Tridentは次のようにボリュームのサイズを計算します。

$$\text{Total volume size} = [(\text{PVC requested size}) / (1 - (\text{snapshotReserve percentage}) / 100)] * 1.1$$

1.1は、LUNメタデータに対応するためにFlexVolに追加される10%のTridentです。= 5%、PVC要求= 5GiBの場合、`snapshotReserve`ボリュームの合計サイズは5.79GiB、使用可能なサイズは5.5GiBです。`volume show`次の例のような結果が表示されます。

Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
		_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4	online	RW	10GB	5.00GB	0%
		_pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d	online	RW	5.79GB	5.50GB	0%
		_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba	online	RW	1GB	511.8MB	0%

3 entries were displayed.

現在、既存のボリュームに対して新しい計算を行うには、サイズ変更だけを使用します。

最小限の設定例

次の例は、ほとんどのパラメータをデフォルトのままにする基本的な設定を示しています。これは、バックエンドを定義する最も簡単な方法です。



TridentでAmazon FSx on NetApp ONTAPを使用している場合は、IPアドレスではなく、LIFのDNS名を指定することを推奨します。

ONTAP SANの例

これは、ontap-san ドライバ。

```

---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>

```

ONTAP SANの経済性の例

```

---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>

```

1. 例

スイッチオーバーやスイッチバックの実行中にバックエンド定義を手動で更新する必要がないようにバックエンドを設定できます。"[SVMのレプリケーションとリカバリ](#)"。

シームレスなスイッチオーバーとスイッチバックを実現するには、managementLIF を省略します。dataLIF および svm パラメータ例：

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

証明書ベースの認証の例

この基本的な設定例では、clientCertificate、clientPrivateKey`および`trustedCACertificate（信頼されたCAを使用している場合はオプション）が入力されます backend.json およびは、クライアント証明書、秘密鍵、信頼されたCA証明書のbase64エンコード値をそれぞれ取得します。

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

双方向CHAPの例

次の例では、 useCHAP をに設定します true。

ONTAP SAN CHAPの例

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
```

ONTAP SANエコノミーCHAPの例

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
```

NVMe/TCPの例

ONTAPバックエンドでNVMeを使用するSVMを設定しておく必要があります。これはNVMe/TCPの基本的なバックエンド構成です。

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

nameTemplateを使用したバックエンド構成の例

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults: {
  "nameTemplate":
    "{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.RequestName}}"
  },
  "labels": {"cluster": "ClusterA", "PVC":
    "{{.volume.Namespace}}_{{.volume.RequestName}}"}
}
```

<code> ONTAP SANエコノミー</code>ドライバのformatOptionsの例

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: ''
svm: svm1
username: ''
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: "-E nodiscard"
```

仮想プールを使用するバックエンドの例

これらのサンプルバックエンド定義ファイルでは、次のような特定のデフォルトがすべてのストレージプールに設定されています。spaceReserve「なし」の場合は、spaceAllocationとの誤り encryption 実行されます。仮想プールは、ストレージセクションで定義します。

Tridentでは、[Comments]フィールドにプロビジョニングラベルが設定されます。FlexVol にコメントが設定されます。Tridentは、仮想プールに存在するすべてのラベルをプロビジョニング時にストレージボリュームにコピーします。ストレージ管理者は、仮想プールごとにラベルを定義したり、ボリュームをラベルでグループ化したりできます。

これらの例では、一部のストレージプールが独自の spaceReserve、spaceAllocation`および`encryption 値、および一部のプールはデフォルト値よりも優先されます。



```

---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
- labels:
  protection: gold
  creditpoints: '40000'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
    adaptiveQosPolicy: adaptive-extreme
- labels:
  protection: silver
  creditpoints: '20000'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
    qosPolicy: premium
- labels:
  protection: bronze
  creditpoints: '5000'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'

```


ONTAP SANの経済性の例

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
labels:
  store: san_economy_store
region: us_east_1
storage:
- labels:
  app: oracledb
  cost: '30'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
- labels:
  app: postgresdb
  cost: '20'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
- labels:
  app: mysqldb
  cost: '10'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
- labels:
  department: legal
  creditpoints: '5000'
```

```
zone: us_east_1c
defaults:
  spaceAllocation: 'true'
  encryption: 'false'
```

NVMe/TCPの例

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: 'false'
  encryption: 'true'
storage:
- labels:
  app: testApp
  cost: '20'
  defaults:
    spaceAllocation: 'false'
    encryption: 'false'
```

バックエンドを **StorageClasses** にマッピングします

次のStorageClass定義は、[\[仮想プールを使用するバックエンドの例\]](#)。を使用する `parameters.selector` フィールドでは、各StorageClassがボリュームのホストに使用できる仮想プールを呼び出します。ボリュームには、選択した仮想プール内で定義された要素があります。

- `protection-gold` StorageClassは、`ontap-san` バックエンド：ゴールドレベルの保護を提供する唯一のプールです。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- protection-not-gold StorageClassは、内の2番目と3番目の仮想プールにマッピングされます。ontap-san バックエンド：これらは、ゴールド以外の保護レベルを提供する唯一のプールです。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- app-mysqldb StorageClassは内の3番目の仮想プールにマッピングされます ontap-san-economy バックエンド：これは、mysqldbタイプアプリケーション用のストレージプール構成を提供する唯一のプールです。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- protection-silver-creditpoints-20k StorageClassは内の2番目の仮想プールにマッピングされます ontap-san バックエンド：シルバーレベルの保護と20000クレジットポイントを提供する唯一のプールです。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- creditpoints-5k StorageClassは内の3番目の仮想プールにマッピングされます ontap-san バックエンドと内の4番目の仮想プール ontap-san-economy バックエンド：これらは、5000クレジットポイントを持つ唯一のプールオフリングです。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- my-test-app-sc StorageClassはにマッピングされます testAPP 内の仮想プール ontap-san ドライバ sanType: nvme。これは唯一のプールサービスです。 testApp。

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Tridentが選択する仮想プールを決定し、ストレージ要件が満たされるようにします。

ONTAP NAS ドライバ

ONTAP NAS ドライバの概要

ONTAP および Cloud Volumes ONTAP の NAS ドライバを使用した ONTAP バックエンドの設定について説明します。

ONTAP NASドライバの詳細

Tridentは、ONTAPクラスタと通信するための次のNASストレージドライバを提供します。サポートされているアクセスモードは、*ReadWriteOnce(RWO)*、*ReadOnlyMany(ROX)*、*ReadWriteMany(RWX)*、*ReadWriteOncePod(RWOP)*です。

ドライバ	プロトコル	ボリュームモード	サポートされているアクセスモード	サポートされるファイルシステム
「ONTAP - NAS」	NFS SMB	ファイルシステム	RWO、ROX、RWX、RWOP	""、nfs、smb
「ONTAP - NAS - エコノミー」	NFS SMB	ファイルシステム	RWO、ROX、RWX、RWOP	""、nfs、smb
「ONTAP-NAS-flexgroup」	NFS SMB	ファイルシステム	RWO、ROX、RWX、RWOP	""、nfs、smb



- 使用 `ontap-san-economy` 永続的ボリュームの使用数が次の値よりも多いと予想される場合のみ **"サポートされるONTAPの制限"**。
- 使用 `ontap-nas-economy` 永続的ボリュームの使用数が次の値よりも多いと予想される場合のみ **"サポートされるONTAPの制限"** および `ontap-san-economy` ドライバは使用できません。
- 使用しないでください `ontap-nas-economy` データ保護、ディザスタリカバリ、モビリティのニーズが予想される場合。

ユーザ権限

Tridentは、ONTAP管理者またはSVM管理者（通常はクラスタユーザ、`vsadmin`SVMユーザ`、または別の名前で同じロールのユーザを使用）として実行することを想定しています ``admin`。

Amazon FSx for NetApp ONTAP環境では、Tridentは、クラスタユーザまたは `vsadmin`SVMユーザ` を使用するONTAP管理者またはSVM管理者、または同じロールの別の名前のユーザとして実行される必要があります ``fsxadmin`。この ``fsxadmin`` ユーザは、クラスタ管理者ユーザに代わる限定的なユーザです。



パラメータを使用する場合は `limitAggregateUsage`、クラスタ管理者の権限が必要です。TridentでAmazon FSx for NetApp ONTAPを使用している場合、`limitAggregateUsage`` パラメータはユーザアカウントと ``fsxadmin`` ユーザアカウントでは機能しません ``vsadmin`。このパラメータを指定すると設定処理は失敗します。

ONTAP内でTridentドライバが使用できる、より制限の厳しいロールを作成することは可能ですが、推奨しません。Tridentの新リリースでは、多くの場合、考慮すべきAPIが追加で必要になるため、アップグレードが難しく、エラーも起こりやすくなります。

ONTAP NAS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

ONTAP NASドライバでONTAPバックエンドを設定するための要件、認証オプション、およびエクスポートポリシーを理解します。

要件

- すべてのONTAPバックエンドについて、TridentでSVMにアグリゲートを少なくとも1つ割り当てる必要があります。
- 複数のドライバを実行し、どちらか一方を参照するストレージクラスを作成できます。たとえば、を使用するGoldクラスを設定できます `ontap-nas` ドライバとを使用するBronzeクラス `ontap-nas-economy` 1つ。
- すべてのKubernetesワーカーノードに適切なNFSツールをインストールしておく必要があります。を参照してください "[こちらをご覧ください](#)" 詳細：
- Tridentでは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみがサポートされます。詳細については、を参照してください [SMBボリュームをプロビジョニングする準備をします](#)。

ONTAPバックエンドの認証

Tridentには、ONTAPバックエンドの認証に2つのモードがあります。

- **Credential-based**：このモードでは、ONTAPバックエンドに十分な権限が必要です。事前定義されたセキュリティログインロールに関連付けられたアカウントを使用することを推奨します。例：`admin` または `vsadmin` ONTAP のバージョンとの互換性を最大限に高めるため。
- **証明書ベース**：このモードでは、TridentがONTAPクラスタと通信するために、バックエンドに証明書をインストールする必要があります。この場合、バックエンド定義には、Base64 でエンコードされたクライアント証明書、キー、および信頼された CA 証明書（推奨）が含まれている必要があります。

既存のバックエンドを更新して、クレデンシャルベースの方式と証明書ベースの方式を切り替えることができます。ただし、一度にサポートされる認証方法は1つだけです。別の認証方式に切り替えるには、バックエンド設定から既存の方式を削除する必要があります。



クレデンシャルと証明書の両方を*指定しようとすると、バックエンドの作成が失敗し、構成ファイルに複数の認証方法が指定されているというエラーが表示されます。

クレデンシャルベースの認証を有効にします

TridentがONTAPバックエンドと通信するには、SVMを対象としたクラスタを対象とした管理者に対するクレデンシャルが必要です。や `vsadmin`` などの事前定義された標準のロールを使用することを推奨します ``admin`。これにより、今後のONTAPリリースで使用する機能APIが公開される可能性がある将来のTridentリリースとの前方互換性が確保されます。Tridentでは、カスタムのセキュリティログインロールを作成して使用できますが、推奨されません。

バックエンド定義の例は次のようになります。

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

バックエンド定義は、クレデンシャルがプレーンテキストで保存される唯一の場所であることに注意してください。バックエンドが作成されると、ユーザ名とパスワードが Base64 でエンコードされ、Kubernetes シークレットとして格納されます。クレデンシャルの知識が必要なのは、バックエンドの作成と更新だけです。この処理は管理者専用で、Kubernetes / ストレージ管理者が実行します。

証明書ベースの認証を有効にします

新規または既存のバックエンドは証明書を使用して ONTAP バックエンドと通信できます。バックエンド定義には 3 つのパラメータが必要です。

- `clientCertificate` : Base64 でエンコードされたクライアント証明書の値。
- `clientPrivateKey` : Base64 でエンコードされた、関連付けられた秘密鍵の値。
- `trustedCACertificate`: 信頼された CA 証明書の Base64 エンコード値。信頼された CA を使用する場合は、このパラメータを指定する必要があります。信頼された CA が使用されていない場合は無視してかまいません。

一般的なワークフローは次の手順で構成されます。

手順

1. クライアント証明書とキーを生成します。生成時に、ONTAP ユーザとして認証するように Common

Name (CN ; 共通名) を設定します。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key  
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. 信頼された CA 証明書を ONTAP クラスタに追加します。この処理は、ストレージ管理者がすでに行っている可能性があります。信頼できる CA が使用されていない場合は無視します。

```
security certificate install -type server -cert-name <trusted-ca-cert-  
name> -vserver <vserver-name>  
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled  
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca  
<cert-authority>
```

3. ONTAP クラスタにクライアント証明書とキーをインストールします (手順 1)。

```
security certificate install -type client-ca -cert-name <certificate-  
name> -vserver <vserver-name>  
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. ONTAP セキュリティ・ログイン・ロールが 'cert' 認証方式をサポートしていることを確認します

```
security login create -user-or-group-name vsadmin -application ontapi  
-authentication-method cert -vserver <vserver-name>  
security login create -user-or-group-name vsadmin -application http  
-authentication-method cert -vserver <vserver-name>
```

5. 生成された証明書を使用して認証をテスト ONTAP 管理 LIF > と <vserver name> は、管理 LIF の IP アドレスおよび SVM 名に置き換えてください。LIF のサービスポリシーが「default-data-management」に設定されていることを確認する必要があります。

```
curl -X POST -Lk https://<ONTAP-Management-  
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key  
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp  
xmlns="http://www.netapp.com/filer/admin" version="1.21"  
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Base64 で証明書、キー、および信頼された CA 証明書をエンコードする。


```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 前の手順で得た値を使用してバックエンドを作成します。

```
cat cert-backend-updated.json
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "NasBackend",
  "managementLIF": "1.2.3.4",
  "dataLIF": "1.2.3.8",
  "svm": "vserver_test",
  "clientCertificate": "Faaaakkkkeeee...Vaaalllluuueeeee",
  "clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
  "storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |         9 |
+-----+-----+-----+-----+
+-----+-----+

```

認証方法を更新するか、クレデンシャルをローテーションして

既存のバックエンドを更新して、別の認証方法を使用したり、クレデンシャルをローテーションしたりできます。これはどちらの方法でも機能します。ユーザ名とパスワードを使用するバックエンドは証明書を使用するように更新できますが、証明書を使用するバックエンドはユーザ名とパスワードに基づいて更新できます。これを行うには、既存の認証方法を削除して、新しい認証方法を追加する必要があります。次に、更新されたbackend.jsonファイルに必要なパラメータが含まれたものを使用して実行します `tridentctl update backend`。

```

cat cert-backend-updated.json
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "NasBackend",
  "managementLIF": "1.2.3.4",
  "dataLIF": "1.2.3.8",
  "svm": "vserver_test",
  "username": "vsadmin",
  "password": "password",
  "storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |      9 |
+-----+-----+-----+-----+
+-----+-----+

```



パスワードのローテーションを実行する際には、ストレージ管理者が最初に ONTAP でユーザのパスワードを更新する必要があります。この後にバックエンドアップデートが続きます。証明書のローテーションを実行する際に、複数の証明書をユーザに追加することができます。その後、バックエンドが更新されて新しい証明書が使用されるようになります。この証明書に続く古い証明書は、ONTAP クラスタから削除できます。

バックエンドを更新しても、すでに作成されているボリュームへのアクセスは中断されず、その後のボリューム接続にも影響しません。バックエンドの更新が成功すると、TridentがONTAPバックエンドと通信し、以降のボリューム処理を処理できるようになります。

Trident用のカスタムONTAPロールの作成

Tridentで処理を実行するためにONTAP adminロールを使用する必要がないように、最小Privilegesを持つONTAPクラスタロールを作成できます。Tridentバックエンド構成にユーザ名を含めると、Trident作成したONTAPクラスタロールが使用されて処理が実行されます。

Tridentカスタムロールの作成の詳細については、を参照してください["Tridentカスタムロールジェネレータ"](#)。

ONTAP CLIノシヨウ

1. 次のコマンドを使用して新しいロールを作成します。

```
security login role create <role_name\> -cmddirname "command" -access all  
-vserver <svm_name\>
```

2. Tridentユーザのユーザ名を作成します。

```
security login create -username <user_name\> -application ontapi  
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver  
<svm_name\> -comment "user_description"
```

3. ユーザにロールをマッピングします。

```
security login modify username <user_name\> -vserver <svm_name\> -role  
<role_name\> -application ontapi -application console -authmethod  
<password\>
```

System Managerの使用

ONTAPシステムマネージャで、次の手順を実行します。

1. カスタムロールの作成：

- a. クラスタレベルでカスタムロールを作成するには、*[クラスタ]>[設定]*を選択します。

(または) SVMレベルでカスタムロールを作成するには、*[ストレージ]>[Storage VM]>[設定]>[ユーザとロール]*を選択し`required SVM`ます。

- b. の横にある矢印アイコン (→*) を選択します。
- c. [Roles]*で[+Add]*を選択します。
- d. ロールのルールを定義し、*[保存]*をクリックします。

2. ロールをTridentユーザにマップする:+[ユーザとロール]ページで次の手順を実行します。

- a. で[アイコンの追加]*を選択します。
- b. 必要なユーザ名を選択し、* Role *のドロップダウンメニューでロールを選択します。
- c. [保存 (Save)] をクリックします。

詳細については、次のページを参照してください。

- ["ONTAPの管理用のカスタムロール"または"カスタムロールの定義"](#)
- ["ロールとユーザを使用する"](#)

NFS エクスポートポリシーを管理します

Tridentは、NFSエクスポートポリシーを使用して、プロビジョニングするボリュームへのアクセスを制御します。

Tridentでエクスポートポリシーを使用する場合は、次の2つのオプションがあります。

- Tridentでは、エクスポートポリシー自体を動的に管理できます。この処理モードでは、許可可能なIPアドレスを表すCIDRブロックのリストをストレージ管理者が指定します。Tridentは、これらの範囲に該当する該当するノードIPを公開時に自動的にエクスポートポリシーに追加します。または、CIDRを指定しない場合は、パブリッシュ先のボリュームで見つかったグローバル対象のユニキャストIPがすべてエクスポートポリシーに追加されます。
- ストレージ管理者は、エクスポートポリシーを作成したり、ルールを手動で追加したりできます。Tridentでは、設定で別のエクスポートポリシー名を指定しないかぎり、デフォルトのエクスポートポリシーが使用されます。

エクスポートポリシーを動的に管理

Tridentでは、ONTAPバックエンドのエクスポートポリシーを動的に管理できます。これにより、ストレージ管理者は、明示的なルールを手動で定義するのではなく、ワーカーノードのIPで許容されるアドレススペースを指定できます。エクスポートポリシーの管理が大幅に簡易化され、エクスポートポリシーを変更しても、ストレージクラスタに対する手動の操作は不要になります。さらに、ボリュームをマウントしていて、指定された範囲のIPを持つワーカーノードだけにストレージクラスタへのアクセスを制限し、きめ細かく自動化された管理をサポートします。



ダイナミックエクスポートポリシーを使用する場合は、Network Address Translation (NAT; ネットワークアドレス変換) を使用しないでください。NATを使用すると、ストレージコントローラは実際のIPホストアドレスではなくフロントエンドのNATアドレスを認識するため、エクスポートルールに一致しない場合はアクセスが拒否されます。



Trident 24.10では、`ontap-nas`ストレージドライバは以前のリリースと同様に動作します。ONTAP NASドライバに変更はありません。Trident 24.10では、ボリュームベースのきめ細かなアクセス制御が可能なのはストレージドライバだけ`ontap-nas-economy`です。

例

2つの設定オプションを使用する必要があります。バックエンド定義の例を次に示します。

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
- 192.168.0.0/24
autoExportPolicy: true
```



この機能を使用する場合は、SVMのルートジャンクションに、ノードのCIDRブロックを許可するエクスポートルール（デフォルトのエクスポートポリシーなど）を含む事前に作成したエクスポートポリシーがあることを確認する必要があります。1つのSVMをTrident専用にするには、必ずNetAppのベストプラクティスに従ってください。

ここでは、上記の例を使用してこの機能がどのように動作するかについて説明します。

- `autoExportPolicy` がに設定されてい `true` ます。これは、Tridentが、このバックエンドを使用してsvmに対してプロビジョニングされたボリュームごとにエクスポートポリシーを作成し、アドレスブロックを使用してルールの追加と削除を処理すること `autoexportCIDRs` を示します `svm1`。ボリュームがノードに接続されるまでは、そのボリュームへの不要なアクセスを防止するルールのない空のエクスポートポリシーが使用されます。ボリュームがノードに公開されると、Tridentは、指定したCIDRブロック内のノードIPを含む基盤となるqtreeと同じ名前のエクスポートポリシーを作成します。これらのIPは、親FlexVolで使用されるエクスポートポリシーにも追加されます。

◦ 例：

- バックエンドUUID 403b5326-8482-40dB-96d0-d83fb3f4daec
 - `autoExportPolicy` に設定 `true`
 - ストレージプレフィックス `trident`
 - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
 - `svm_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c` という名前のqtree Tridentでは、という名前のFlexVolのエクスポートポリシー、という名前のqtreeのエクスポートポリシー、`trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c` およびという名前の空のエクスポートポリシー `trident_empty` がSVM上に作成されます `trident-403b5326-8482-40db96d0-d83fb3f4daec`。FlexVolエクスポートポリシーのルールは、qtreeエクスポートポリシーに含まれるすべてのルールのスーパーセットになります。空のエクスポートポリシーは、関連付けられていないボリュームで再利用されます。
- `autoExportCIDRs` アドレスブロックのリストが含まれます。このフィールドは省略可能で、デフォルト値は `["0.0.0.0/0", ":::0"]` です。定義されていない場合、Tridentは、パブリケーションを使用して、ワーカーノード上で見つかったグローバルスコープのユニキャストアドレスをすべて追加します。

この例では `192.168.0.0/24`、アドレス空間が提供されています。これは、パブリケーションでこのアドレス範囲に含まれるKubernetesノードIPが、Tridentが作成するエクスポートポリシーに追加されることを示します。Tridentは、実行するノードを登録すると、ノードのIPアドレスを取得し、で指定されたアドレスブロックと照合し `autoExportCIDRs` ます。公開時に、IPをフィルタリングした後、Tridentは公開先ノードのクライアントIPのエクスポートポリシールールを作成します。

バックエンドの作成後に `autoExportPolicy` および `autoExportCIDRs` を更新できます自動的に管理されるバックエンドに新しいCIDRsを追加したり、既存のCIDRsを削除したりできます。CIDRsを削除する際は、既存の接続が切断されないように注意してください。バックエンドに対して「`autoExportPolicy`」を無効にし、手動で作成したエクスポートポリシーに戻すこともできます。これには、バックエンド構成で「`exportPolicy`」パラメータを設定する必要があります。

Tridentがバックエンドを作成または更新した後、または対応するCRDを `tridentbackend` 使用してバックエンドをチェックでき `tridentctl` ます。

```

./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileType: ext4

```

ノードを削除すると、Tridentはすべてのエクスポートポリシーをチェックして、そのノードに対応するアクセスルールを削除します。Tridentは、管理対象バックエンドのエクスポートポリシーからこのノードIPを削除することで、不正なマウントを防止します。ただし、このIPがクラスタ内の新しいノードで再利用される場合を除きます。

既存のバックエンドがある場合は、を使用してバックエンドを更新する `tridentctl update backend` と、Tridentがエクスポートポリシーを自動的に管理するようになります。これにより、バックエンドのUUIDとqtree名に基づいて、必要に応じてという名前の新しいエクスポートポリシーが2つ作成されます。バックエンドにあるボリュームは、アンマウントして再度マウントしたあとに、新しく作成したエクスポートポリシーを使用します。



自動管理されたエクスポートポリシーを使用してバックエンドを削除すると、動的に作成されたエクスポートポリシーが削除されます。バックエンドが再作成されると、そのバックエンドは新しいバックエンドとして扱われ、新しいエクスポートポリシーが作成されます。

稼働中のノードのIPアドレスが更新された場合は、そのノードでTridentポッドを再起動する必要があります。その後、Tridentは管理しているバックエンドのエクスポートポリシーを更新して、IPの変更を反映します。

SMB ボリュームをプロビジョニングする準備をします

多少の準備が必要な場合は、次のツールを使用してSMBボリュームをプロビジョニングできます。 ontap-nas ドライバ。



を作成するには、SVMでNFSプロトコルとSMB / CIFSプロトコルの両方を設定する必要があります。 ontap-nas-economy オンプレミスのONTAP用のSMBボリューム。これらのプロトコルのいずれかを設定しないと、原因 SMBボリュームの作成が失敗します。



`autoExportPolicy`SMBボリュームではサポートされません。

作業を開始する前に

SMBボリュームをプロビジョニングする前に、以下を準備しておく必要があります。

- Linuxコントローラノードと少なくとも1つのWindowsワーカーノードでWindows Server 2022を実行しているKubernetesクラスター。Tridentでは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみがサポートされます。
- Active Directoryクレデンシャルを含む少なくとも1つのTridentシークレット。シークレットを生成するには `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user  
--from-literal password='password'
```

- Windowsサービスとして設定されたCSIプロキシ。を設定します`csi-proxy`を参照してください "[GitHub: CSIプロキシ](#)" または "[GitHub: Windows向けCSIプロキシ](#)" Windowsで実行されているKubernetesノードの場合。

手順

1. オンプレミスのONTAPでは、必要に応じてSMB共有を作成することも、Tridentで共有を作成することもできます。



Amazon FSx for ONTAPにはSMB共有が必要です。

SMB管理共有は、のいずれかの方法で作成できます "[Microsoft管理コンソール](#)" 共有フォルダスナップインまたはONTAP CLIを使用します。ONTAP CLIを使用してSMB共有を作成するには、次の手順を実行します

- a. 必要に応じて、共有のディレクトリパス構造を作成します。

。 `vserver cifs share create` コマンドは、共有の作成時に `-path` オプションで指定されているパスを確認します。指定したパスが存在しない場合、コマンドは失敗します。

- b. 指定したSVMに関連付けられているSMB共有を作成します。

```
vserver cifs share create -vserver vserver_name -share-name  
share_name -path path [-share-properties share_properties,...]  
[other_attributes] [-comment text]
```

- c. 共有が作成されたことを確認します。

```
vserver cifs share show -share-name share_name
```



を参照してください "[SMB 共有を作成](#)" 詳細については、

2. バックエンドを作成する際に、SMBボリュームを指定するように次の項目を設定する必要があります。ONTAP バックエンド構成オプションのすべてのFSXについては、を参照してください "[FSX \(ONTAP の構成オプションと例\)](#)"。

パラメータ	説明	例
smbShare	次のいずれかを指定できます。Microsoft管理コンソールまたはONTAP CLIを使用して作成されたSMB共有の名前、TridentでSMB共有を作成できるようにする名前、ボリュームへの共通の共有アクセスを禁止する場合はパラメータを空白のままにします。オンプレミスのONTAPでは、このパラメータはオプションです。このパラメータはAmazon FSx for ONTAPバックエンドで必須であり、空にすることはできません。	smb-share
nasType	をに設定する必要があります smb . nullの場合、デフォルトはです nfs 。	smb
'securityStyle'	新しいボリュームのセキュリティ形式。をに設定する必要があります ntfs または mixed SMB ボリューム	ntfs または mixed SMBボリュームの場合
「 unixPermissions 」	新しいボリュームのモード。* SMBボリュームは空にしておく必要があります。*	""

ONTAP NASの設定オプションと例



Tridentのインストール時にONTAP NASドライバを作成して使用方法について説明します。このセクションでは、バックエンドの構成例と、バックエンドをStorageClassesにマッピングするための詳細を示します。

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
「バージョン」		常に 1
'storageDriverName'	ストレージドライバの名前	「ontap-nas」、 「ontap-nas-economy」、 「ontap-nas-flexgroup」、 「ontap-san」、 「ontap-san-economy」
backendName`	カスタム名またはストレージバックエンド	ドライバ名+"_" + dataLIF
「管理 LIF 」	クラスタまたはSVM管理LIFのIPアドレス完全修飾ドメイン名 (FQDN) を指定できます。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。シームレスなMetroClusterスイッチオーバーについては、を参照して [mcc-best] ください。	「 10.0.0.1 」、 「 [2001:1234:abcd::fefe] 」

パラメータ	説明	デフォルト
「重複排除	<p>プロトコル LIF の IP アドレス。指定することをお勧めします dataLIF。指定しない場合、TridentはSVMからデータLIFをフェッチします。NFSマウント処理に使用するFully Qualified Domain Name (FQDN；完全修飾ドメイン名) を指定して、ラウンドロビンDNSを作成して複数のデータLIF間で負荷を分散することができます。初期設定後に変更できます。を参照してください。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります</p> <p>[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。* MetroClusterの場合は省略してください。*を参照してください[mcc-best]。</p>	指定されたアドレス、または指定されていない場合はSVMから取得されるアドレス (非推奨)
'VM'	<p>使用する Storage Virtual Machine</p> <p>* MetroClusterの場合は省略してください。* [mcc-best]。</p>	SVM 「管理 LIF」 が指定されている場合に生成されます
「autoExportPolicy」を参照してください	エクスポートポリシーの自動作成と更新を有効にします[ブーリアン]。オプションと `autoExportCIDRs` オプションを使用する `autoExportPolicy` と、Tridentでエクスポートポリシーを自動的に管理できます。	いいえ
「autoExportCIDR」	が有効な場合にKubernetesのノードIPをフィルタリングするCIDRのリスト autoExportPolicy。オプションと `autoExportCIDRs` オプションを使用する `autoExportPolicy` と、Tridentでエクスポートポリシーを自動的に管理できます。	["0.0.0.0/0","::/0"]
「ラベル」	ボリュームに適用する任意の JSON 形式のラベルのセット	""
「clientCertificate」をクリックします	クライアント証明書の Base64 エンコード値。証明書ベースの認証に使用されます	""
「clientPrivateKey」	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます	""
「trustedCacertificate」	信頼された CA 証明書の Base64 エンコード値。任意。証明書ベースの認証に使用されます	""
「ユーザ名」	クラスタ / SVM に接続するためのユーザ名。クレデンシャルベースの認証に使用されます	
「password」と入力します	クラスタ / SVM に接続するためのパスワード。クレデンシャルベースの認証に使用されます	

パラメータ	説明	デフォルト
'storagePrefix'	<p>SVM で新しいボリュームをプロビジョニングする際に使用するプレフィックスを指定します。設定後に更新することはできません</p> <p> qtree-nas-economyとstoragePrefixをONTAP 24文字以上で使用する場合、ボリューム名にはストレージプレフィックスは含まれませんが、qtreeにはストレージプレフィックスが埋め込まれます。</p>	"トライデント"
「集約」	<p>プロビジョニング用のアグリゲート（オプション。設定する場合は SVM に割り当てる必要があります）。ドライバの場合 <code>ontap-nas-flexgroup</code>、このオプションは無視されます。割り当てられていない場合は、使用可能ないずれかのアグリゲートを使用してFlexGroupボリュームをプロビジョニングできます。</p> <p> SVMでアグリゲートが更新されると、Tridentコントローラを再起動せずにSVMをポーリングすることで、Tridentでアグリゲートが自動的に更新されます。ボリュームをプロビジョニングするようにTridentで特定のアグリゲートを設定している場合、アグリゲートの名前を変更するかSVMから移動すると、SVMアグリゲートのポーリング中にTridentでバックエンドが障害状態になります。アグリゲートをSVMにあるアグリゲートに変更するか、アグリゲートを完全に削除してバックエンドをオンラインに戻す必要があります。</p>	""
「AggreglimitateUsage」と入力します	<p>使用率がこの割合を超えている場合は、プロビジョニングが失敗します。* Amazon FSX for ONTAP * には適用されません</p>	""（デフォルトでは適用されません）

パラメータ	説明	デフォルト
flexgroupAggregateList	<p>プロビジョニング用のアグリゲートのリスト（オプション。設定されている場合はSVMに割り当てる必要があります）。SVMに割り当てられたすべてのアグリゲートを使用して、FlexGroupボリュームがプロビジョニングされます。ONTAP - NAS - FlexGroup * ストレージドライバでサポートされています。</p> <p> SVMでアグリゲートリストが更新されると、Tridentコントローラを再起動せずにSVMをポーリングすることで、Trident内のアグリゲートリストが自動的に更新されます。ボリュームをプロビジョニングするようにTridentで特定のアグリゲートリストを設定している場合、アグリゲートリストの名前を変更するかSVMから移動すると、Tridentアグリゲートのポーリング中にバックエンドが障害状態になります。アグリゲートリストをSVM上のアグリゲートリストに変更するか、アグリゲートリストを完全に削除してバックエンドをオンラインに戻す必要があります。</p>	""
「limitVolumeSize」と入力します	<p>要求されたボリュームサイズがこの値を超えている場合、プロビジョニングが失敗します。また、qtreesに対して管理するボリュームの最大サイズを制限し、オプションを使用`qtreesPerFlexvol`するとFlexVolあたりの最大qtree数をカスタマイズできます。</p>	""（デフォルトでは適用されません）
「バグトレースフラグ」	<p>トラブルシューティング時に使用するデバッグフラグ。例：{"api": false、"method": true}</p> <p>使用しないでください debugTraceFlags トラブルシューティングを実行していて、詳細なログダンプが必要な場合を除きます。</p>	null
nasType	<p>NFSボリュームまたはSMBボリュームの作成を設定オプションはです nfs、 smb またはnull。nullに設定すると、デフォルトでNFSボリュームが使用されます。</p>	nfs
「nfsvMountOptions」のように入力します	<p>NFSマウントオプションをカンマで区切ったリスト。Kubernetes永続ボリュームのマウントオプションは通常ストレージクラスで指定されますが、ストレージクラスにマウントオプションが指定されていない場合、Tridentはストレージバックエンドの構成ファイルに指定されているマウントオプションを使用してフォールバックします。ストレージクラスまたは構成ファイルでマウントオプションが指定されていない場合、Tridentは関連付けられた永続ボリュームにマウントオプションを設定しません。</p>	""

パラメータ	説明	デフォルト
qtreesPerFlexvol	FlexVol あたりの最大 qtree 数。有効な範囲は [50、300] です。	"200"
smbShare	次のいずれかを指定できます。Microsoft管理コンソールまたはONTAP CLIを使用して作成されたSMB共有の名前、TridentでSMB共有を作成できるようにする名前、ボリュームへの共通の共有アクセスを禁止する場合はパラメータを空白のままにします。オンプレミスのONTAPでは、このパラメータはオプションです。このパラメータはAmazon FSx for ONTAPバックエンドで必須であり、空にすることはできません。	smb-share
「useREST」	ONTAP REST API を使用するためのブーリアンパラメータ。useRESTに設定する `true` と、Tridentはバックエンドとの通信にONTAP REST APIを使用します。に設定する `false` と、Tridentはバックエンドとの通信にONTAP ZAPI呼び出しを使用します。この機能にはONTAP 9.11.1以降が必要です。また、使用するONTAPログインロールには、アプリケーションへのアクセス権が必要です `ontap`。これは、事前に定義された役割と役割によって実現され vsadmin cluster-admin ます。Trident 24.06リリースおよびONTAP 9.15.1以降では、userRESTがデフォルトでに設定されて `true` います。ONTAP ZAPI呼び出しを使用するには、をに `false` 変更してください。 `useREST`	true ONTAP 9.15.1以降の場合は、それ以外の場合は false。
limitVolumePoolSize	ONTAP NASエコノミーバックエンドでqtreeを使用する場合の、要求可能なFlexVolの最大サイズ。	"" (デフォルトでは適用されません)
denyNewVolumePools	を制限し `ontap-nas-economy` バックエンドがqtreeを格納するために新しいFlexVolボリュームを作成することです。新しいPVのプロビジョニングには、既存のFlexVolのみが使用されます。	

ボリュームのプロビジョニング用のバックエンド構成オプション

これらのオプションを使用して、のデフォルトプロビジョニングを制御できます defaults 設定のセクション。例については、以下の設定例を参照してください。

パラメータ	説明	デフォルト
「平和の配分」	qtreeに対するスペース割り当て	"正しい"
「平和のための準備」を参照してください	スペースリザーベーションモード：「none」 (シン) または「volume」 (シック)	"なし"
「ナップショットポリシー」	使用する Snapshot ポリシー	"なし"

パラメータ	説明	デフォルト
「QOSPolicy」	作成したボリュームに割り当てる QoS ポリシーグループ。ストレージプール/バックエンドごとに QOSPolicy または adaptiveQosPolicy のいずれかを選択します	""
「adaptiveQosPolicy」を参照してください	アダプティブ QoS ポリシーグループ：作成したボリュームに割り当てます。ストレージプール/バックエンドごとに QOSPolicy または adaptiveQosPolicy のいずれかを選択します。経済性に影響する ONTAP - NAS ではサポートされません。	""
「スナップショット予約」	Snapshot 用にリザーブされているボリュームの割合	次の場合は「0」 snapshotPolicy は「none」、 それ以外の場合は「」です。
'splitOnClone	作成時にクローンを親からスプリットします	いいえ
「暗号化」	新しいボリュームで NetApp Volume Encryption (NVE) を有効にします。デフォルトはです。`false` このオプションを使用するには、クラスタで NVE のライセンスが設定され、有効になっている必要があります。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で NAE が有効になります。詳細については、 を参照してください"TridentとNVEおよびNAEとの連携" 。	いいえ
階層ポリシー	「none」を使用する階層化ポリシー	ONTAP 9.5より前のSVM-DR設定の場合は「snapshot-only」
「unixPermissions」	新しいボリュームのモード	NFSボリュームの場合は「777」、SMBボリュームの場合は空（該当なし）
「スナップショット方向」	にアクセスする権限を管理します。 .snapshot ディレクトリ	NFSv4の場合は「true」 NFSv3の場合は「false」
「exportPolicy」と入力します	使用するエクスポートポリシー	デフォルト
'securityStyle'	新しいボリュームのセキュリティ形式。NFSのサポート mixed および unix セキュリティ形式SMBはをサポートします mixed および ntfs セキュリティ形式	NFSのデフォルトはです unix。SMBのデフォルトはです ntfs。
nameTemplate	カスタムボリューム名を作成するためのテンプレート。	""



TridentでQoSポリシーグループを使用するには、ONTAP 9.8以降が必要です。共有されていないQoSポリシーグループを使用し、ポリシーグループが各コンスチチュエントに個別に適用されるようにします。QoSポリシーグループを共有すると、すべてのワークロードの合計スループットの上限が適用されます。

ボリュームプロビジョニングの例

デフォルトが定義されている例を次に示します。

```

---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: '10'

```

と `ontap-nas-flexgroups` については、`ontap-nas` Trident 新しい計算式を使用して、FlexVol が `snapshotReserve` の割合と PVC で正しくサイジングされるようになりました。ユーザが PVC を要求すると、Trident は新しい計算を使用して、より多くのスペースを持つ元の FlexVol を作成します。この計算により、ユーザは要求された PVC 内の書き込み可能なスペースを受信し、要求されたスペースよりも少ないスペースを確保できます。v21.07 より前のバージョンでは、ユーザが PVC を要求すると（5GiB など）、`snapshotReserve` が 50% に設定されている場合、書き込み可能なスペースは 2.5GiB のみになります。これは、ユーザが要求したのはボリューム全体であり、その割合であるため `snapshotReserve` です。Trident 21.07 では、ユーザが要求するのは書き込み可能なスペースであり、Trident ではボリューム全体に対する割合として定義されます。`snapshotReserve` これはには適用されませ `ontap-nas-economy` ん。この機能の仕組みについては、次の例を参照してください。

計算は次のとおりです。

```

Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)

```

snapshotReserve = 50%、PVC 要求 = 5GiB の場合、ボリュームの合計サイズは $2/0.5 = 10\text{GiB}$ であり、使用可能なサイズは 5GiB であり、これが PVC 要求で要求されたサイズです。volume show コマンドは '次の例のような結果を表示する必要があります

Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
	_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4		online	RW	10GB	5.00GB	0%
	_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba		online	RW	1GB	511.8MB	0%

2 entries were displayed.

以前のインストールからの既存のバックエンドでは、Tridentのアップグレード時に前述のようにボリュームがプロビジョニングされます。アップグレード前に作成したボリュームについては、変更が反映されるようにボリュームのサイズを変更する必要があります。たとえば、以前のと2GiBのPVCで`snapshotReserve=50`は、1GiBの書き込み可能なスペースを提供するボリュームが作成されました。たとえば、ボリュームのサイズを3GiBに変更すると、アプリケーションの書き込み可能なスペースが6GiBのボリュームで3GiBになります。

最小限の設定例

次の例は、ほとんどのパラメータをデフォルトのままにする基本的な設定を示しています。これは、バックエンドを定義する最も簡単な方法です。



ネットアップ ONTAP で Trident を使用している場合は、IP アドレスではなく LIF の DNS 名を指定することを推奨します。

ONTAP NASエコノミーの例

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

ONTAP NAS FlexGroupの例

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

MetroClusterの例

スイッチオーバーやスイッチバックの実行中にバックエンド定義を手動で更新する必要がないようにバックエンドを設定できます。"[SVMのレプリケーションとリカバリ](#)"。

シームレスなスイッチオーバーとスイッチバックを実現するには、managementLIF を省略します。dataLIF および svm パラメータ例：

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

SMBボリュームの例

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```


証明書ベースの認証の例

これは、バックエンドの最小限の設定例です。`clientCertificate`、`clientPrivateKey` および `trustedCACertificate`（信頼されたCAを使用している場合はオプション）が入力されます。`backend.json` およびは、クライアント証明書、秘密鍵、信頼されたCA証明書のbase64エンコード値をそれぞれ取得します。

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

自動エクスポートポリシーの例

この例は、動的なエクスポートポリシーを使用してエクスポートポリシーを自動的に作成および管理するようにTridentに指示する方法を示しています。これは、ドライバと `ontap-nas-flexgroup` ドライバで同じように機能し `ontap-nas-economy` ます。

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

IPv6アドレスの例

この例は、を示しています managementLIF IPv6アドレスを使用している。

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

SMBボリュームを使用したAmazon FSx for ONTAPの例

。 smbShare SMBボリュームを使用するFSx for ONTAPの場合、パラメータは必須です。

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

nameTemplateを使用したバックエンド構成の例

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults: {
  "nameTemplate":
    "{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.R
    equestName}}"
  },
  "labels": {"cluster": "ClusterA", "PVC":
    "{{.volume.Namespace}}_{{.volume.RequestName}}"}
}
```

仮想プールを使用するバックエンドの例

以下に示すサンプルのバックエンド定義ファイルでは、次のような特定のデフォルトがすべてのストレージプールに設定されています。spaceReserve「なし」の場合は、spaceAllocationとの誤り encryption 実行されます。仮想プールは、ストレージセクションで定義します。

Tridentでは、[Comments]フィールドにプロビジョニングラベルが設定されます。コメントは、のFlexVolまたはのFlexGroup ontap-nas-flexgroup`で設定します `ontap-nas。Tridentは、仮想プールに存在するすべてのラベルをプロビジョニング時にストレージボリュームにコピーします。ストレージ管理者は、仮想プールごとにラベルを定義したり、ボリュームをラベルでグループ化したりできます。

これらの例では、一部のストレージプールが独自の spaceReserve、spaceAllocation`および `encryption 値、および一部のプールはデフォルト値よりも優先されます。

ONTAP NASの例

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: 'false'
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
  app: msoffice
  cost: '100'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
    adaptiveQosPolicy: adaptive-premium
- labels:
  app: slack
  cost: '75'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  department: legal
  creditpoints: '5000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
```

```
  app: wordpress
  cost: '50'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
  app: mysqldb
  cost: '25'
  zone: us_east_1d
  defaults:
    spaceReserve: volume
    encryption: 'false'
    unixPermissions: '0775'
```

ONTAP NAS FlexGroupの例

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
  protection: gold
  creditpoints: '50000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  protection: gold
  creditpoints: '30000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  protection: silver
  creditpoints: '20000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
  protection: bronze
  creditpoints: '10000'
  zone: us_east_1d
```

```
defaults:  
  spaceReserve: volume  
  encryption: 'false'  
  unixPermissions: '0775'
```

```

---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: nas_economy_store
region: us_east_1
storage:
- labels:
  department: finance
  creditpoints: '6000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  protection: bronze
  creditpoints: '5000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  department: engineering
  creditpoints: '3000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
  department: humanresource
  creditpoints: '2000'
  zone: us_east_1d
  defaults:

```



```
spaceReserve: volume
encryption: 'false'
unixPermissions: '0775'
```

バックエンドを **StorageClasses** にマッピングします

次のStorageClass定義は、を参照してください。 [\[仮想プールを使用するバックエンドの例\]](#)。を使用する `parameters.selector` フィールドでは、各StorageClassがボリュームのホストに使用できる仮想プールを呼び出します。ボリュームには、選択した仮想プール内で定義された要素があります。

- `protection-gold` StorageClassは、 `ontap-nas-flexgroup` バックエンド：ゴールドレベルの保護を提供する唯一のプールです。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- `protection-not-gold` StorageClassは、内の3番目と4番目の仮想プールにマッピングされます。 `ontap-nas-flexgroup` バックエンド：金色以外の保護レベルを提供する唯一のプールです。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- `app-mysqldb` StorageClassは内の4番目の仮想プールにマッピングされます。 `ontap-nas` バックエンド：これは、`mysqldb`タイプアプリ用のストレージプール構成を提供する唯一のプールです。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- [t] protection-silver-creditpoints-20k StorageClassは、ontap-nas-flexgroup バックエンド：シルバーレベルの保護と20000クレジットポイントを提供する唯一のプールです。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- 。 creditpoints-5k StorageClassは、ontap-nas バックエンドと内の2番目の仮想プール ontap-nas-economy バックエンド：これらは、5000クレジットポイントを持つ唯一のプールオフリングです。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Tridentが選択する仮想プールを決定し、ストレージ要件が満たされるようにします。

更新 dataLIF 初期設定後

初期設定後にデータLIFを変更するには、次のコマンドを実行して、更新されたデータLIFを新しいバックエンドJSONファイルに指定します。

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-with-updated-dataLIF>
```



PVCが1つ以上のポッドに接続されている場合は、対応するすべてのポッドを停止してから、新しいデータLIFを有効にするために稼働状態に戻す必要があります。

NetApp ONTAP 対応の Amazon FSX

Amazon FSx for NetApp ONTAPでTridentを使用

"NetApp ONTAP 対応の Amazon FSX" は、NetApp ONTAP ストレージオペレーティングシステムを基盤とするファイルシステムの起動や実行を可能にする、フルマネージドのAWSサービスです。FSX for ONTAP を使用すると、使い慣れたネットアップの機能、パフォーマンス、管理機能を活用しながら、AWSにデータを格納するためのシンプルさ、即応性、セキュリティ、拡張性を活用できます。FSX for ONTAP は、ONTAP ファイルシステムの機能と管理APIをサポートしています。

Amazon FSx for NetApp ONTAPファイルシステムをTridentと統合すると、Amazon Elastic Kubernetes Service (EKS) で実行されているKubernetesクラスタが、ONTAPを基盤とするブロックおよびファイルの永続ボリュームをプロビジョニングできるようになります。

ファイルシステムは、オンプレミスの ONTAP クラスタに似た、Amazon FSX のプライマリリソースです。各 SVM 内には、ファイルとフォルダをファイルシステムに格納するデータコンテナである 1 つ以上のボリュームを作成できます。Amazon FSx for NetApp ONTAP を使用すると、Data ONTAP はクラウド内の管理対象ファイルシステムとして提供されます。新しいファイルシステムのタイプは * NetApp ONTAP * です。

TridentとAmazon FSx for NetApp ONTAPを使用すると、Amazon Elastic Kubernetes Service (EKS) で実行されているKubernetesクラスタが、ONTAPを基盤とするブロックおよびファイルの永続ボリュームをプロビジョニングできるようになります。

要件

"Tridentの要件"FSx for ONTAPとTridentを統合するには、さらに次のものがが必要です。

- 既存の Amazon EKS クラスタまたは 'kubect!' がインストールされた自己管理型 Kubernetes クラスタ
- クラスタのワーカーノードから到達可能な既存のAmazon FSx for NetApp ONTAPファイルシステムおよびStorage Virtual Machine (SVM) 。
- 準備されているワーカーノード "NFSまたはiSCSI"。



Amazon LinuxおよびUbuntuで必要なノードの準備手順を実行します "Amazon Machine Images の略" (AMIS) EKS の AMI タイプに応じて異なります。

考慮事項

- SMBボリューム：
 - SMBボリュームは、を使用してサポートされます ontap-nas ドライバーのみ。

- SMBボリュームは、Trident EKSアドオンではサポートされません。
- Tridentでは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみがサポートされます。詳細については、[を参照してください "SMBボリュームをプロビジョニングする準備をします"](#)。
- Trident 24.02より前のバージョンでは、自動バックアップが有効になっているAmazon FSxファイルシステム上に作成されたボリュームは、Tridentで削除できませんでした。Trident 24.02以降でこの問題を回避するには、AWS FSx for ONTAPのバックエンド構成ファイルで、`apiRegion`AWS、AWS、およびAWS`apiKey``を ``secretKey`` 指定します ``fsxFilesystemID``。



TridentにIAMロールを指定する場合は、`apiKey``、および `secretKey`` の各フィールドをTridentに明示的に指定する必要はありません ``apiRegion``。詳細については、[を参照してください "FSX \(ONTAP の構成オプションと例\)"](#)。

認証

Tridentには2つの認証モードがあります。

- クレデンシャルベース（推奨）：クレデンシャルをAWS Secrets Managerに安全に格納します。ファイルシステムのユーザ、またはSVM用に設定されているユーザを使用できます `fsxadmin vsadmin``。



Tridentは、SVMユーザ、または別の名前と同じロールのユーザとして実行することを想定しています `vsadmin``。Amazon FSx for NetApp ONTAPには、ONTAPクラスタユーザに代わる限定的なユーザが `admin``、`fsxadmin`` があります。Tridentでの使用を強くお勧めしません `vsadmin``。

- 証明書ベース：Tridentは、SVMにインストールされている証明書を使用してFSxファイルシステム上のSVMと通信します。

認証を有効にする方法の詳細については、使用しているドライバタイプの認証を参照してください。

- ["ONTAP NAS認証"](#)
- ["ONTAP SAN認証"](#)

詳細については、[こちらをご覧ください](#)

- ["Amazon FSX for NetApp ONTAP のドキュメント"](#)
- ["Amazon FSX for NetApp ONTAP に関するブログ記事です"](#)

IAMロールとAWS Secretを作成する

KubernetesポッドがAWSリソースにアクセスするように設定するには、明示的なAWSクレデンシャルを指定する代わりに、AWS IAMロールとして認証します。



AWS IAMロールを使用して認証するには、EKSを使用してKubernetesクラスタを導入する必要があります。

AWS Secret Managerシークレットの作成

次の例では、Trident CSIクレデンシャルを格納するAWSシークレットマネージャシークレットを作成します。

```
aws secretsmanager create-secret --name trident-secret --description "Trident CSI credentials" --secret-string "{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

IAMポリシーの作成

次の例は、AWS CLIを使用してIAMポリシーを作成します。

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy-document file://policy.json --description "This policy grants access to Trident CSI to FSxN and Secret manager"
```

ポリシーJSONファイル：

```
policy.json:
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx>DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-id>:secret:<aws-secret-manager-name>"
    }
  ],
  "Version": "2012-10-17"
}
```

次の例では、EKSでサービスアカウント用のIAMロールを作成します。

```
eksctl create iamserviceaccount --name trident-controller --namespace trident
--cluster <my-cluster> --role-name <AmazonEKS_FSxN_CSI_DriverRole> --role-only
--attach-policy-arn arn:aws:iam::aws:policy/service-
role/AmazonFSxNCSIDriverPolicy --approve
```

Astra Trident をインストール

Astra Tridentは、KubernetesでのAmazon FSx for NetApp ONTAPストレージ管理を合理化し、開発者や管理者がアプリケーションの導入に集中できるようにします。

Astra Tridentは次のいずれかの方法でインストールできます。

- Helm
- EKSアドオン

```
If you want to make use of the snapshot functionality, install the CSI
snapshot controller add-on. Refer to
https://docs.aws.amazon.com/eks/latest/userguide/csi-snapshot-
controller.html.
```

Helmを使用してAstra Tridentをインストール

1. Astra Tridentインストーラパッケージをダウンロード

Astra Tridentインストーラパッケージには、Tridentオペレータの導入とAstra Tridentのインストールに必要なものがすべて含まれています。最新バージョンのAstra TridentインストーラをGitHubの[Assets]セクションからダウンロードして展開します。

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xvf trident-installer-24.10.0.tar.gz
cd trident-installer
```

2. 次の環境変数を使用して、* cloud provider フラグと cloud identity *フラグの値を設定します。

```
export CP="AWS"
export CI="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'"
```

次の例では、Astra Tridentをインストールし、フラグを \$CP、`cloud-identity`に`\$CI`設定して`cloud-provider`ます。

```
helm install trident trident-operator-100.2410.0.tgz --set
cloudProvider=$CP --set cloudIdentity=$CI --namespace trident
```

コマンドを使用して、名前、ネームスペース、グラフ、ステータス、アプリケーションのバージョン、リビジョン番号など、インストールの詳細を確認できます `helm list`。

```
helm list -n trident
```

NAME	NAMESPACE	REVISION	UPDATED
STATUS	CHART		APP VERSION
trident-operator	trident	1	2024-10-14 14:31:22.463122
+0300 IDT	deployed	trident-operator-100.2410.0	24.10.0

EKSアドオンを使用してAstra Tridentをインストール

Astra Trident EKSアドオンには、最新のセキュリティパッチ、バグ修正が含まれており、AWSによってAmazon EKSとの連携が検証されています。EKSアドオンを使用すると、Amazon EKSクラスタの安全性と安定性を一貫して確保し、アドオンのインストール、構成、更新に必要な作業量を削減できます。

前提条件

AWS EKS用のAstra Tridentアドオンを設定する前に、次の条件を満たしていることを確認してください。

- アドオンサブスクリプションがあるAmazon EKSクラスタアカウント
- AWS MarketplaceへのAWS権限：
"aws-marketplace:ViewSubscriptions",
"aws-marketplace:Subscribe",
"aws-marketplace:Unsubscribe
- AMIタイプ：Amazon Linux 2 (AL2_x86_64) またはAmazon Linux 2 Arm (AL2_ARM_64)
- ノードタイプ：AMDまたはARM
- 既存のAmazon FSx for NetApp ONTAPファイルシステム

AWS向けのAstra Tridentアドオンを有効にする

EKSクラスタ

次のコマンド例は、Astra Trident EKSアドオンをインストールします。

```
eksctl create addon --cluster clusterName --name netapp_trident-operator  
--version v24.10.0-eksbuild  
eksctl create addon --cluster clusterName --name netapp_trident-operator  
--version v24.10.0-eksbuild.1 (専用バージョンを使用)
```



オプションのパラメータを設定する場合 `cloudIdentity` は、EKSアドオンを使用してTridentをインストールするときに指定して `cloudProvider` ください。

管理コンソール

1. でAmazon EKSコンソールを開きます <https://console.aws.amazon.com/eks/home#/clusters>。
2. 左側のナビゲーションペインで、*[クラスタ]*をクリックします。
3. NetApp Trident CSIアドオンを設定するクラスタの名前をクリックします。
4. をクリックし、[その他のアドオンの入手]*をクリックします。
5. [アドオンの選択]ページで、次の手順を実行します。
 - a. [AWS Marketplace EKS-addons]セクションで、* NetApp Trident *チェックボックスを選択します。
 - b. 「*次へ*」をクリックします。
6. [Configure selected add-ons* settings]ページで、次の手順を実行します。
 - a. 使用する*バージョン*を選択します。
 - b. では、[Not set]*のままにします。
 - c. *オプションの構成設定*を展開し、*アドオン構成スキーマ*に従って、*構成値*セクションのconfigurationValuesパラメータを前の手順で作成したrole-arnに設定します（値は次の形式にする必要があります `eks.amazonaws.com/role-arn:arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole`）。

[Conflict resolution method]で[Override]を選択すると、既存のアドオンの1つ以上の設定をAmazon EKSアドオン設定で上書きできます。このオプションを有効にしない場合、既存の設定と競合すると、操作は失敗します。表示されたエラーメッセージを使用して、競合のトラブルシューティングを行うことができます。このオプションを選択する前に、Amazon EKSアドオンが自己管理に必要な設定を管理していないことを確認してください。
7. [次へ]*を選択します。
8. [確認して追加]ページで、*[作成]*を選択します。

アドオンのインストールが完了すると、インストールされているアドオンが表示されます。

AWS CLI

1. ファイルを作成し add-on.json ます。


```
add-on.json
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v24.10.0-eksbuild.1",
  "serviceAccountRoleArn": "arn:aws:iam::123456:role/astratrident-
role",
  "configurationValues": "{\"cloudIdentity\":
'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/astratrident-
role'\",
  \"cloudProvider\": \"AWS\"}"
}
```

2. Install the Astra<xmt-block0> Trident</xmt-block> EKS add-on

```
aws eks create-addon --cli-input-json file://add-on.json
```

Astra Trident EKSアドオンの更新

EKSクラスタ

- お使いのFSxN Trident CSIアドオンの現在のバージョンを確認してください。をクラスタ名に置き換え `my-cluster` ます。

```
eksctl get addon --name netapp_trident-operator --cluster my-cluster
```

出力例：

```
NAME                                VERSION                                STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v24.10.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- 前の手順の出力で `update available` で返されたバージョンにアドオンを更新します。
`eksctl update addon --name netapp_trident-operator --version v24.10.0-eksbuild.1 --cluster my-cluster --force`

オプションを削除し、いずれかのAmazon EKSアドオン設定が既存の設定と競合している場合 `--force`、Amazon EKSアドオンの更新は失敗します。競合の解決に役立つエラーメッセージが表示されます。このオプションを指定する前に、管理する必要がある設定がAmazon EKSアドオンで管理されていないことを確認してください。これらの設定はこのオプションで上書きされます。この設定のその他のオプションの詳細については、を参照してください "[アドオン](#)". Amazon EKS Kubernetesフィールド管理の詳細については、を参照してください "[Kubernetesフィールド管理](#)".

管理コンソール

- Amazon EKSコンソールを開き <https://console.aws.amazon.com/eks/home#/clusters> ます。
- 左側のナビゲーションペインで、*[クラスタ]*をクリックします。
- NetApp Trident CSIアドオンを更新するクラスタの名前をクリックします。
- [アドオン]タブをクリックします。
- NetApp Trident をクリックし、Edit *をクリックします。
- [選択したアドオン設定の構成]ページで、次の手順を実行します。
 - 使用する*バージョン*を選択します。
 - [Optional configuration settings]*を展開し、必要に応じて変更します。
 - [変更の保存 *] をクリックします。

AWS CLI

次の例では、EKSアドオンを更新します。

```
aws eks update-addon --cluster-name my-cluster netapp_trident-operator vpc-cni
--addon-version v24.10.0-eksbuild.1 \
--service-account-role-arn arn:aws:iam::111122223333:role/role-name
--configuration-values '{} ' --resolve-conflicts --preserve
```

Astra Trident EKSアドオンのアンインストールと削除

Amazon EKSアドオンを削除するには、次の2つのオプションがあります。

- クラスタにアドオンソフトウェアを保持–このオプションを選択すると、Amazon EKSによる設定の管理が削除されます。また、Amazon EKSが更新を通知し、更新を開始した後にAmazon EKSアドオンを自動的に更新する機能も削除されます。ただし、クラスタ上のアドオンソフトウェアは保持されます。このオプションを選択すると、アドオンはAmazon EKSアドオンではなく自己管理型インストールになります。このオプションを使用すると、アドオンのダウンタイムは発生しません。アドオンを保持するには、コマンドのオプションをそのまま使用し `--preserve` ます。
- クラスタからアドオンソフトウェアを完全に削除する–クラスターに依存するリソースがない場合にのみ、Amazon EKSアドオンをクラスターから削除することをお勧めします。コマンドからオプションを削除してアドオンを削除し `--preserve delete` ます。



アドオンにIAMアカウントが関連付けられている場合、IAMアカウントは削除されません。

EKSクラスタ

次のコマンドは、Astra Trident EKSアドオンをアンインストールします。

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

管理コンソール

1. でAmazon EKSコンソールを開きます <https://console.aws.amazon.com/eks/home#/clusters>。
2. 左側のナビゲーションペインで、*[クラスタ]*をクリックします。
3. NetApp Trident CSIアドオンを削除するクラスタの名前をクリックします。
4. タブをクリックし、[Astra Trident by NetApp]をクリックします。
5. [削除 (Remove)]をクリックします。
6. [Remove netapp_trident-operator confirmation]*ダイアログで、次の手順を実行します。
 - a. Amazon EKSでアドオンの設定を管理しないようにするには、*[クラスタに保持]*を選択します。クラスタにアドオンソフトウェアを残して、アドオンのすべての設定を自分で管理できるようにする場合は、この手順を実行します。
 - b. 「netapp_trident -operator *」と入力します。
 - c. [削除 (Remove)]をクリックします。

AWS CLI

をクラスタの名前に置き換え `my-cluster`、次のコマンドを実行します。

```
aws eks delete-addon --cluster-name my-cluster --addon-name netapp_trident-operator --preserve
```

ストレージバックエンドの設定

ONTAP SANとNASドライバの統合

バックエンドファイルは、次の例に示すように、AWS Secret Managerに保存されているSVMのクレデンシャル

ル（ユーザ名とパスワード）を使用して作成できます。

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFileSystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFileSystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

バックエンドの作成については、次のページを参照してください。

- ["バックエンドに ONTAP NAS ドライバを設定します"](#)
- ["バックエンドに ONTAP SAN ドライバを設定します"](#)

FSx for ONTAP ドライバの詳細

次のドライバを使用して、TridentとAmazon FSx for NetApp ONTAPを統合できます。

- `ontap-san`：プロビジョニングされる各PVは、それぞれのAmazon FSx for NetApp ONTAPボリューム内のLUNです。ブロックストレージに推奨されます。
- `ontap-nas`：プロビジョニングされる各PVは、完全なAmazon FSx for NetApp ONTAPボリュームです。NFSとSMBで推奨されます。
- 「ONTAP と SAN の経済性」：プロビジョニングされた各 PV は、NetApp ONTAP ボリュームの Amazon FSX ごとに構成可能な数の LUN を持つ LUN です。
- 「ONTAP-NAS-エコノミー」：プロビジョニングされた各 PV は `qtree` であり、NetApp ONTAP ボリュームの Amazon FSX ごとに設定可能な数の `qtree` があります。
- 「ONTAP-NAS-flexgroup」：プロビジョニングされた各 PV は、NetApp ONTAP FlexGroup ボリューム用の完全な Amazon FSX です。

ドライバーの詳細については、[こちら](#)を参照してください。"[NASドライバ](#)" および "[SANドライバ](#)"。

構成例

Secret Managerを使用したAWS FSx for ONTAPの設定

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFileSystemID: fs-xxxxxxxxxx
  managementLIF:
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxx:secret:secret-
name"
    type: awsarn
```

SMBホリウムノストレエシクラスノセツテイ

を使用します `nasType`、`node-stage-secret-name`および`node-stage-secret-namespace``を使用して、SMBボリュームを指定し、必要なActive Directoryクレデンシャルを指定できます。SMBボリュームは、を使用してサポートされます `ontap-nas` ドライバーのみ。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: nas-smb-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

バックエンドの高度な設定と例

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	例
「バージョン」		常に 1
'storageDriverName'	ストレージドライバの名前	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy
backendName`	カスタム名またはストレージバックエンド	ドライバ名 + "_" + データ LIF

パラメータ	説明	例
「管理 LIF」	<p>クラスタまたはSVM管理LIFのIPアドレス完全修飾ドメイン名（FQDN）を指定できます。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]などの角かっこで定義する必要があります。aws`フィールドでを指定する場合は`fsxFilesystemID、を指定する必要はありません。managementLIF`。TridentはAWSからSVM情報を取得するためです。`managementLIF`そのため、SVMの下のユーザ（vsadminなど）のクレデンシャルを指定し、そのユーザにロールが割り当てられている必要があります `vsadmin`。</p>	<p>「10.0.0.1」、「[2001:1234:abcd::fefe]」</p>
「重複排除	<p>プロトコル LIF の IP アドレス。* ONTAP NASドライバ*: データLIFを指定することを推奨します。指定しない場合、TridentはSVMからデータLIFをフェッチします。NFSマウント処理に使用するFully Qualified Domain Name（FQDN；完全修飾ドメイン名）を指定して、ラウンドロビンDNSを作成して複数のデータLIF間で負荷を分散することができます。初期設定後に変更できます。を参照してください。* ONTAP SANドライバ*: iSCSIには指定しないでくださいTridentは、ONTAP選択的LUNマップを使用して、マルチパスセッションの確立に必要なiSCSI LIFを検出します。データLIFが明示的に定義されている場合は警告が生成されます。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]などの角かっこで定義する必要があります。</p>	

パラメータ	説明	例
「 autoExportPolicy 」を参照してください	エクスポートポリシーの自動作成と更新を有効にします[ブーリアン]。オプションと `autoExportCIDRs` オプションを使用する `autoExportPolicy` と、Tridentでエクスポートポリシーを自動的に管理できます。	「偽」
「 autoExportCI` 」	が有効な場合にKubernetesのノードIPをフィルタリングするCIDRのリスト autoExportPolicy。オプションと `autoExportCIDRs` オプションを使用する `autoExportPolicy` と、Tridentでエクスポートポリシーを自動的に管理できます。	「[0.0.0.0/0]、 「::/0」 」
「ラベル」	ボリュームに適用する任意のJSON形式のラベルのセット	""
「 clientCertificate 」をクリックします	クライアント証明書の Base64 エンコード値。証明書ベースの認証に使用されます	""
「 clientPrivateKey 」	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます	""
「 trustedCacertifate 」	信頼された CA 証明書の Base64 エンコード値。任意。証明書ベースの認証に使用されます。	""
「ユーザ名」	クラスタまたはSVMに接続するためのユーザ名。クレデンシャルベースの認証に使用されます。たとえば、vsadminのように指定します。	
「 password 」と入力します	クラスタまたはSVMに接続するためのパスワード。クレデンシャルベースの認証に使用されます。	
'VM'	使用する Storage Virtual Machine	SVM管理LIFが指定されている場合に生成されます。
'storagePrefix'	SVM で新しいボリュームをプロビジョニングする際に使用するプレフィックスを指定します。作成後に変更することはできません。このパラメータを更新するには、新しいバックエンドを作成する必要があります。	trident

パラメータ	説明	例
「AggreglimitateUsage」と入力します	* Amazon FSx for NetApp ONTAP には指定しないでください。*指定されたと vsadmin`には `fsxadmin、アグリゲートの使用量を取得してTridentを使用して制限するために必要な権限が含まれていません。	使用しないでください。
「limitVolumeSize」と入力します	要求されたボリュームサイズがこの値を超えている場合、プロビジョニングが失敗します。また、qtreeおよびLUN用に管理するボリュームの最大サイズも制限します qtreesPerFlexvol オプションを使用すると、FlexVol あたりの最大qtree数をカスタマイズできます。	"" (デフォルトでは適用されません)
lunsPerFlexvol	FlexVol あたりの最大LUN数。有効な範囲は50、200です。SANのみ。	"100"
「バグトレースフラグ」	トラブルシューティング時に使用するデバッグフラグ。例：{"API": false、"method": true}は使用されません debugTraceFlags トラブルシューティングを実行していて、詳細なログダンプが必要な場合を除きます。	null
「nfsvMountOptions」のように入力します	NFSマウントオプションをカンマで区切ったリスト。Kubernetes永続ボリュームのマウントオプションは通常ストレージクラスで指定されますが、ストレージクラスにマウントオプションが指定されていない場合、Tridentはストレージバックエンドの構成ファイルに指定されているマウントオプションを使用してフォールバックします。ストレージクラスまたは構成ファイルでマウントオプションが指定されていない場合、Tridentは関連付けられた永続ボリュームにマウントオプションを設定しません。	""
nasType	NFSボリュームまたはSMBボリュームの作成を設定オプションはです nfs、smb、またはnull。*をに設定する必要があります smb SMBボリューム。*をnullに設定すると、デフォルトでNFSボリュームが使用されます。	nfs

パラメータ	説明	例
qtreesPerFlexvol`	FlexVol あたりの最大 qtree 数。有効な範囲は [50、300] です。	"200"
smbShare	次のいずれかを指定できます。Microsoft管理コンソールまたはONTAP CLIを使用して作成されたSMB共有の名前、またはTridentにSMB共有の作成を許可する名前。このパラメータは、Amazon FSx for ONTAPバックエンドに必要です。	smb-share
「useREST`」	ONTAP REST API を使用するためのブーリアンパラメータ。技術レビュー useREST は技術レビューとして提供されており、本番環境のワークロードには推奨されません。に設定する true`と、TridentはONTAP REST APIを使用してバックエンドと通信します。この機能にはONTAP 9.11.1以降が必要です。また、使用するONTAPログインロールには、アプリケーションへのアクセス権が必要です`ontap。これは、事前に定義された役割と役割によって実現されるvsadmin cluster-admin ます。	「偽」
aws	AWS FSx for ONTAPの構成ファイルでは、次の項目を指定できます。 - fsxFilesystemID：AWS FSxファイルシステムのIDを指定します。 - apiRegion：AWS APIリージョン名。 - apikey：AWS APIキー。 - secretKey：AWSシークレットキー。	"" "" ""
credentials	AWS Secret Managerに保存するFSx SVMのクレデンシャルを指定します。 - name：シークレットのAmazon Resource Name (ARN)。SVMのクレデンシャルが含まれています。 - type:に設定 awsarn。 を参照してください " AWS Secrets Managerシークレットの作成 " を参照してください。	

ボリュームのプロビジョニング用のバックエンド構成オプション

これらのオプションを使用して、のデフォルトプロビジョニングを制御できます defaults 設定のセクション。例については、以下の設定例を参照してください。

パラメータ	説明	デフォルト
「平和の配分」	space-allocation for LUN のコマンドを指定します	「真」
「平和のための準備」を参照してください	スペースリザベーションモード： 「none」（シン）または「volume」（シック）	「NONE」
「ナップショットポリシー」	使用する Snapshot ポリシー	「NONE」
「QOSPolicy」	作成したボリュームに割り当てる QoS ポリシーグループ。ストレージプールまたはバックエンドごとに、QOSPolicyまたはadaptiveQosPolicyのいずれかを選択します。TridentでQoSポリシーグループを使用するには、ONTAP 9.8以降が必要です。共有されていないQoSポリシーグループを使用し、ポリシーグループが各コンスチチュエントに個別に適用されるようにします。QoSポリシーグループを共有すると、すべてのワークロードの合計スループットの上限が適用されます。	「」
「adaptiveQosPolicy」を参照してください	アダプティブ QoS ポリシーグループ：作成したボリュームに割り当てます。ストレージプールまたはバックエンドごとに、QOSPolicyまたはadaptiveQosPolicyのいずれかを選択します。経済性に影響する ONTAP - NAS ではサポートされません。	「」
「スナップショット予約」	スナップショット "0" 用に予約されたボリュームの割合	状況 snapshotPolicy はです none、else 「」
'plitOnClone	作成時にクローンを親からスプリットします	「偽」

パラメータ	説明	デフォルト
「暗号化」	新しいボリュームでNetApp Volume Encryption (NVE) を有効にします。デフォルトはです。 `false`このオプションを使用するには、クラスタで NVE のライセンスが設定され、有効になっている必要があります。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でNAEが有効になります。詳細については、を参照してください" Trident とNVEおよびNAEとの連携 "。	「偽」
luksEncryption	LUKS暗号化を有効にします。を参照してください " Linux Unified Key Setup (LUKS ; 統合キーセットアップ) を使用 "。SANのみ。	""
階層ポリシー	使用する階層化ポリシー none	snapshot-only ONTAP 9.5より前のSVM-DR構成の場合
「 unixPermissions 」	新しいボリュームのモード。* SMB ボリュームは空にしておきます。*	「」
'securityStyle'	新しいボリュームのセキュリティ形式。NFSのサポート mixed および unix セキュリティ形式SMBはをサポートします mixed および ntfs セキュリティ形式	NFSのデフォルトはです unix 。SMBのデフォルトはです ntfs。

SMBボリュームをプロビジョニングする準備をします

を使用してSMBボリュームをプロビジョニングできます `ontap-nas` ドライバ。をクリックしてください [ONTAP SANとNASドライバの統合](#) 次の手順を実行します。

作業を開始する前に

SMBボリュームをプロビジョニングする前に `ontap-nas` ドライバー、あなたは以下を持っている必要があります。

- Linuxコントローラノードと少なくとも1つのWindowsワーカーノードでWindows Server 2019を実行しているKubernetesクラスタ。Tridentでは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみがサポートされます。
- Active Directoryクレデンシャルを含む少なくとも1つのTridentシークレット。シークレットを生成するには `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Windowsサービスとして設定されたCSIプロキシ。を設定します `csi-proxy` を参照してください "[GitHub:](#)

CSIプロキシ" または "GitHub: Windows向けCSIプロキシ" Windowsで実行されているKubernetesノードの場合。

手順

1. SMB共有を作成SMB管理共有は、のいずれかの方法で作成できます "[Microsoft管理コンソール](#)" 共有フォルダスナップインまたはONTAP CLIを使用します。ONTAP CLIを使用してSMB共有を作成するには、次の手順を実行します

- a. 必要に応じて、共有のディレクトリパス構造を作成します。

。 `vserver cifs share create` コマンドは、共有の作成時に `-path` オプションで指定されているパスを確認します。指定したパスが存在しない場合、コマンドは失敗します。

- b. 指定したSVMに関連付けられているSMB共有を作成します。

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

- c. 共有が作成されたことを確認します。

```
vserver cifs share show -share-name share_name
```



を参照してください "[SMB 共有を作成](#)" 詳細については、

2. バックエンドを作成する際に、SMBボリュームを指定するように次の項目を設定する必要があります。ONTAP バックエンド構成オプションのすべてのFSXについては、を参照してください "[FSX \(ONTAP の構成オプションと例\)](#)" 。

パラメータ	説明	例
smbShare	次のいずれかを指定できます。Microsoft管理コンソールまたはONTAP CLIを使用して作成されたSMB共有の名前、またはTridentにSMB共有の作成を許可する名前。このパラメータは、Amazon FSx for ONTAPバックエンドに必要です。	smb-share
nasType	をに設定する必要があります smb . nullの場合、デフォルトは <code>dfs</code> です nfs 。	smb
'securityStyle'	新しいボリュームのセキュリティ形式。をに設定する必要があります ntfs または mixed SMB ボリューム	ntfs または mixed SMB ボリュームの場合

パラメータ	説明	例
「unixPermissions」	新しいボリュームのモード。* SMBボリュームは空にしておく必要があります。*	""

ストレージクラスとPVCを設定する

Kubernetes StorageClassオブジェクトを設定してストレージクラスを作成し、Tridentでボリュームのプロビジョニング方法を指定します。設定したKubernetes StorageClassを使用してPVへのアクセスを要求するPersistentVolume (PV) とPersistentVolumeClaim (PVC) を作成します。その後、PVをポッドにマウントできます。

ストレージクラスを作成する。

Kubernetes StorageClassオブジェクトの設定

では、"[Kubernetes StorageClassオブジェクト](#)"そのクラスで使用されるプロビジョニングツールとしてTridentが指定され、ボリュームのプロビジョニング方法がTridentに指示されます。例：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
```

ストレージクラスとパラメータおよびパラメータとの連携によるTridentによるボリュームのプロビジョニング方法の詳細については PersistentVolumeClaim、を参照してください"[Kubernetes オブジェクトと Trident オブジェクト](#)"。

ストレージクラスを作成する。

手順

1. これはKubernetesオブジェクトなので、`kubectl` をクリックしてKubernetesで作成します。

```
kubectl create -f storage-class-ontapnas.yaml
```

2. KubernetesとTridentの両方で「basic-csi」ストレージクラスが表示され、Tridentがバックエンドでプールを検出していることを確認します。

```
kubectl get sc basic-csi
NAME          PROVISIONER          AGE
basic-csi     csi.trident.netapp.io 15h
```

PVおよびPVCの作成

A "[永続ボリューム](#)" (PV) は、Kubernetesクラスタ上のクラスタ管理者がプロビジョニングする物理ストレージリソースです。。 "[PersistentVolumeClaim](#)" (PVC) は、クラスタ上のPersistentVolumeへのアクセス要求です。

PVCは、特定のサイズまたはアクセスモードのストレージを要求するように設定できます。クラスタ管理者は、関連付けられているStorageClassを使用して、PersistentVolumeのサイズとアクセスモード（パフォーマンスやサービスレベルなど）以上を制御できます。

PVとPVCを作成したら、ポッドにボリュームをマウントできます。

マニフェストの例

PersistentVolumeサンプルマニフェスト

このサンプルマニフェストは、StorageClassに関連付けられた10Giの基本PVを示しています。 basic-csi。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/my/host/path"
```

PersistentVolumeClaim サンプルマニフェスト

次に、基本的なPVC設定オプションの例を示します。

RWOアクセスを備えたPVC

この例は、という名前のStorageClassに関連付けられたRWXアクセスを持つ基本的なPVCを示しています。basic-csi。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

NVMe / TCP対応PVC

この例は、という名前のStorageClassに関連付けられたNVMe/TCPの基本的なPVCとRWOアクセスを示しています。protection-gold。

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san-nvme
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: protection-gold
```

PVおよびPVCの作成

手順

1. PVを作成します。


```
kubectl create -f pv.yaml
```

2. PVステータスを確認します。

```
kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS  REASON    AGE
pv-storage    4Gi       RWO           Retain          Available
7s
```

3. PVCを作成します。

```
kubectl create -f pvc.yaml
```

4. PVCステータスを確認します。

```
kubectl get pvc
NAME          STATUS  VOLUME          CAPACITY  ACCESS MODES  STORAGECLASS  AGE
pvc-storage  Bound  pv-name         2Gi       RWO           storageclass  5m
```

ストレージクラスとパラメータおよびパラメータとの連携によるTridentによるボリュームのプロビジョニング方法の詳細については [PersistentVolumeClaim](#)、を参照してください"[Kubernetes オブジェクトと Trident オブジェクト](#)"。

Trident属性

これらのパラメータは、特定のタイプのボリュームのプロビジョニングに使用する Trident で管理されているストレージプールを決定します。

属性	を入力します	値	提供	リクエスト	でサポートされます
メディア ^1	文字列	HDD、ハイブリッド、SSD	プールにはこのタイプのメディアが含まれています。ハイブリッドは両方を意味します	メディアタイプが指定されました	ONTAPNAS、ONTAPNAS エコノミー、ONTAP-NAS-flexgroup、ONTAPSAN、solidfire-san-SAN、solidfire-san-SAN のいずれかに対応しています

属性	を入力します	値	提供	リクエスト	でサポートされます
プロビジョニングタイプ	文字列	シン、シック	プールはこのプロビジョニング方法をサポートします	プロビジョニング方法が指定されました	シック：All ONTAP ; thin : All ONTAP & solidfire-san-SAN
backendType	文字列	ONTAPNAS、ONTAPNASエコノミー、ONTAP-NAS-flexgroup、ONTAPSAN、solidfire-san-SAN、solidfire-san-SAN、GCP-cvs、azure-NetApp-files、ONTAP-SAN-bエコノミー	プールはこのタイプのバックエンドに属しています	バックエンドが指定されて	すべてのドライバ
Snapshot	ブール値	true false	プールは、Snapshot を含むボリュームをサポートします	Snapshot が有効なボリューム	ONTAP-NAS、ONTAP-SAN、solidfire-san、gcvcs
クローン	ブール値	true false	プールはボリュームのクローニングをサポートします	クローンが有効なボリューム	ONTAP-NAS、ONTAP-SAN、solidfire-san、gcvcs
暗号化	ブール値	true false	プールでは暗号化されたボリュームをサポート	暗号化が有効なボリューム	ONTAP-NAS、ONTAP-NAS-エコノミー、ONTAP-NAS-FlexArray グループ、ONTAP-SAN
IOPS	整数	正の整数	プールは、この範囲内で IOPS を保証する機能を備えています	ボリュームで IOPS が保証されました	solidfire - SAN

^1 ^ : ONTAP Select システムではサポートされていません

サンプルアプリケーションのデプロイ

サンプルアプリケーションをデプロイします。

手順

1. ボリュームをポッドにマウントします。

```
kubectl create -f pv-pod.yaml
```

次に、PVCをポッドに接続するための基本的な設定例を示します。基本設定：

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
        claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```



進捗状況は次を使用して監視できます。 `kubectl get pod --watch`。

2. ボリュームがマウントされていることを確認します。 `/my/mount/path`。

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

```
Filesystem                                                    Size
Used Avail Use% Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06 1.1G
320K 1.0G 1% /my/mount/path
```

1. ポッドを削除できるようになりました。Podアプリケーションは存在しなくなりますが、ボリュームは残ります。

```
kubectl delete pod task-pv-pod
```

EKSクラスタでのAstra Trident EKSアドオンの設定

Astra Tridentは、KubernetesでのAmazon FSx for NetApp ONTAPストレージ管理を合理化し、開発者や管理者がアプリケーションの導入に集中できるようにします。Astra Trident EKSアドオンには、最新のセキュリティパッチ、バグ修正が含まれており、AWSによってAmazon EKSとの連携が検証されています。EKSアドオンを使用すると、Amazon EKSクラスタの安全性と安定性を一貫して確保し、アドオンのインストール、構成、更新に必要な作業量を削減できます。

前提条件

AWS EKS用のAstra Tridentアドオンを設定する前に、次の条件を満たしていることを確認してください。

- アドオンサブスクリプションがあるAmazon EKSクラスタアカウント
- AWS MarketplaceへのAWS権限：
"aws-marketplace:ViewSubscriptions",
"aws-marketplace:Subscribe",
"aws-marketplace:Unsubscribe
- AMIタイプ：Amazon Linux 2 (AL2_x86_64) またはAmazon Linux 2 Arm (AL2_ARM_64)
- ノードタイプ：AMDまたはARM
- 既存のAmazon FSx for NetApp ONTAPファイルシステム

手順

1. EKS Kubernetesクラスタで、*アドオン*タブに移動します。

The screenshot shows the AWS EKS console interface for a cluster named 'tri-env-eks'. At the top right, there are buttons for 'Delete cluster', 'Upgrade version', and 'View dashboard'. Below this is a notification banner about the end of standard support for Kubernetes version 1.30 on July 28, 2025, with an 'Upgrade now' button. The main content area is divided into sections: 'Cluster info' (Status: Active, Kubernetes version: 1.30, Support period: Standard support until July 28, 2025, Provider: EKS), 'Cluster health issues' (0 issues), and 'Upgrade insights' (0 insights). Below these is a navigation bar with tabs for Overview, Resources, Compute, Networking, Add-ons (1), Access, Observability, Update history, and Tags. A notification banner indicates 'New versions are available for 1 add-on.' The 'Add-ons (3)' section is active, showing a search bar with 'Find add-on', filters for 'Any category' and 'Any status', and '3 matches'.

2. [AWS Marketplace add-ons]*にアクセスし、_storage_categoryを選択します。

AWS Marketplace add-ons (1) ↻

Discover, subscribe to and configure EKS add-ons to enhance your EKS clusters.

Filtering options

Any category ▾ NetApp, Inc. ▾ Any pricing model ▾ [Clear filters](#)

NetApp, Inc. ✕ < 1 >

NetApp **NetApp Trident** □

NetApp Trident streamlines Amazon FSx for NetApp ONTAP storage management in Kubernetes to let your developers and administrators focus on application deployment. FSx for ONTAP flexibility, scalability, and integration capabilities make it the ideal choice for organizations seeking efficient containerized storage workflows. [Product details](#)

Standard Contract

Category storage	Listed by NetApp, Inc.	Supported versions 1.31, 1.30, 1.29, 1.28, 1.27, 1.26, 1.25, 1.24, 1.23	Pricing starting at View pricing details
----------------------------	--	---	--

[Cancel](#) [Next](#)

3. NetApp Trident *を探し、Astra Tridentアドオンのチェックボックスを選択します。
4. 必要なアドオンのバージョンを選択します。

NetApp Trident [Remove add-on](#)

Listed by NetApp	Category storage	Status ✔ Ready to install
----------------------------	---------------------	------------------------------

i You're subscribed to this software [View subscription](#) ✕

You can view the terms and pricing details for this product or choose another offer if one is available.

Version
Select the version for this add-on.

Select IAM role
Select an IAM role to use with this add-on. To create a new custom role, follow the instructions in the [Amazon EKS User Guide](#).

[↻](#)

▶ **Optional configuration settings**

[Cancel](#) [Previous](#) [Next](#)

5. ノードから継承するIAMロールオプションを選択します。

Review and add

Step 1: Select add-ons

Edit

Selected add-ons (1)

Find add-on

Add-on name	Type	Status
netapp_trident-operator	storage	Ready to install

Step 2: Configure selected add-ons settings

Edit

Selected add-ons version (1)

Add-on name	Version	IAM role for service account (IRSA)
netapp_trident-operator	v24.10.0-eksbuild.1	Not set

EKS Pod Identity (0)

Add-on name	IAM role	Service account
No Pod Identity associations None of the selected add-on(s) have Pod Identity associations.		

Cancel

Previous

Create

6. *オプションの構成設定*を展開し、*アドオン構成スキーマ*に従って、*構成値*セクションのconfigurationValuesパラメーターを前の手順で作成したrole-arnに設定します（値は次の形式にする必要があります eks.amazonaws.com/role-arn:arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole）。[Conflict resolution method]で[Override]を選択すると、既存のアドオンの1つ以上の設定をAmazon EKSアドオン設定で上書きできます。このオプションを有効にしない場合、既存の設定と競合すると、操作は失敗します。表示されたエラーメッセージを使用して、競合のトラブルシューティングを行うことができます。このオプションを選択する前に、Amazon EKSアドオンが自己管理に必要な設定を管理していないことを確認してください。

▼ Optional configuration settings

Add-on configuration schema

Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.

```
default: {},
"examples": [
  {
    "cloudIdentity": ""
  }
],
"properties": {
  "cloudIdentity": {
    "default": "",
    "examples": [
      ""
    ],
    "title": "The cloudIdentity Schema",
    "type": "string"
  }
}
```

Configuration values [Info](#)

Specify any additional JSON or YAML configurations that should be applied to the add-on.

```
1 {
2   "cloudIdentity": "'eks.amazonaws.com/role-arn: arn:aws:iam
3 }
   ::186785786363:role/tri-env-eks-trident-controller-role'"
```

7. 「* Create *」を選択します。
8. アドオンのステータスが `_Active_` であることを確認します。

Add-ons (1) [Info](#) View details Edit Remove Get more add-ons

Q netapp × Any categ... Any status 1 match < 1 >

NetApp **NetApp Trident** ○

NetApp Trident streamlines Amazon FSx for NetApp ONTAP storage management in Kubernetes to let your developers and administrators focus on application deployment. FSx for ONTAP flexibility, scalability, and integration capabilities make it the ideal choice for organizations seeking efficient containerized storage workflows. [Product details](#)

Category	Status	Version	EKS Pod Identity	IAM role for service account (IRSA)
storage	✔ Active	v24.10.0-eksbuild.1	-	Not set

Listed by [NetApp, Inc.](#)

[View subscription](#)

CLIを使用したAstra Trident EKSアドオンのインストールとアンインストール

CLIを使用してAstra Trident EKSアドオンをインストールします。

次のコマンド例は、Astra Trident EKSアドオンをインストールします（専用バージョンを使用）。

```
eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version v24.10.0-eksbuild
eksctl create addon --cluster clusterName --name netapp_trident-operator --version v24.10.0-eksbuild.1
```

CLIを使用してAstra Trident EKSアドオンをアンインストールします。

次のコマンドは、Astra Trident EKSアドオンをアンインストールします。

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

kubectl を使用してバックエンドを作成します

バックエンドは、Tridentとストレージシステム間の関係を定義します。Tridentは、そのストレージシステムとの通信方法や、Tridentがそのシステムからボリュームをプロビジョニングする方法を解説します。Tridentをインストールしたら、次の手順でバックエンドを作成します。TridentBackendConfig`Custom Resource Definition (CRD) を使用すると、Kubernetesインターフェイスから直接Tridentバックエンドを作成および管理できます。これは、またはKubernetesディストリビューション用の同等のCLIツールを使用して実行できます `kubectl`。

TridentBackendConfig

TridentBackendConfig(tbc, tbconfig, tbackendconfig)は、を使用してTridentバックエンドを管理できるフロントエンドの名前空間CRDです。`kubectl`Kubernetes管理者やストレージ管理者は、Kubernetes CLIを使用して直接バックエンドを作成、管理できるようになりました(`tridentctl`た。専用のコマンドラインユーティリティは必要ありません)。

「TridentBackendConfig」オブジェクトを作成すると、次のようになります。

- バックエンドは、指定した設定に基づいてTridentによって自動的に作成されます。これは内部的には (tbc、tridentbackend) CRとして表され `TridentBackend` ます。
- は TridentBackendConfig、Tridentによって作成されたに一意にバインドされます TridentBackend。

各「TridentBackendConfig」は、「TridentBackend」を使用して1対1のマッピングを維持します。前者はバックエンドの設計と構成をユーザに提供するインターフェイスで、後者はTridentが実際のバックエンドオブジェクトを表す方法です。



`TridentBackend`CRSはTridentによって自動的に作成されます。これらは*変更しないでください。バックエンドを更新するには、オブジェクトを変更し `TridentBackendConfig` ます。

「TridentBackendConfig」CRの形式については、次の例を参照してください。


```

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret

```

の例を確認することもできます ["Trident インストーラ"](#) 目的のストレージプラットフォーム / サービスの設定例を示すディレクトリ。

。spec バックエンド固有の設定パラメータを使用します。この例では、バックエンドはを使用します ontap-san storage driver およびでは、に示す構成パラメータを使用します。ご使用のストレージドライバの設定オプションのリストについては、["ストレージドライバのバックエンド設定情報"](#)。

「PEC」セクションには、「credentials」フィールドと「deletionPolicy」フィールドも含まれています。これらのフィールドは、「TridentBackendConfig」CR に新しく導入されました。

- **credentials** : このパラメータは必須フィールドで、ストレージシステム / サービスとの認証に使用されるクレデンシャルが含まれています。ユーザが作成した Kubernetes Secret に設定されます。クレデンシャルをプレーンテキストで渡すことはできないため、エラーになります。
- **DeletionPolicy**: 「TridentBackendConfig」が削除されたときに何が起るかを定義します。次の2つの値のいずれかを指定できます。
 - 「削除」 : これにより、「TridentBackendConfig」CR とそれに関連付けられたバックエンドの両方が削除されます。これがデフォルト値です。
 - 「管理」 : 「TridentBackendConfig」CR を削除しても、バックエンド定義は引き続き表示され、「tridentctl」で管理できます。削除ポリシーを「retain」に設定すると、ユーザは以前のリリース (21.04 より前) にダウングレードし、作成されたバックエンドを保持できます。このフィールドの値は、「TridentBackendConfig」が作成された後で更新できます。



バックエンドの名前は 'PEC.backendName' を使用して設定されます指定しない場合、バックエンドの名前は「TridentBackendConfig」オブジェクト (metadata.name) の名前に設定されます。'PEC.backendName' を使用してバックエンド名を明示的に設定することをお勧めします



で作成されたバックエンドに tridentctl は、関連付けられたオブジェクトはありません `TridentBackendConfig`。このようなバックエンドを管理するには、kubect1 CR を作成し `TridentBackendConfig` ます。同一の設定パラメータ (、 `spec.storagePrefix` `spec.storageDriverName` など) を指定するように注意する必要があります `spec.backendName`。Tridentは、新しく作成されたを既存のバックエンドに自動的にバインドし `TridentBackendConfig` ます。

手順の概要

kubectl' を使用して新しいバックエンドを作成するには '次の手順を実行する必要があります

1. を作成し "Kubernetes Secret" ます。シークレットには、Tridentがストレージクラスタ/サービスと通信するために必要なクレデンシャルが含まれています。
2. 「TridentBackendConfig」オブジェクトを作成します。ストレージクラスタ/サービスの詳細を指定し、前の手順で作成したシークレットを参照します。

バックエンドを作成したら、「kubectl get tbc <tbc-name> -n <trident-namespace>`」を使用してバックエンドのステータスを確認し、詳細を収集できます。

手順 1 : Kubernetes Secret を作成します

バックエンドのアクセスクレデンシャルを含むシークレットを作成します。ストレージサービス/プラットフォームごとに異なる固有の機能です。次に例を示します。

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: t@Ax@7q(>
```

次の表に、各ストレージプラットフォームの Secret に含める必要があるフィールドをまとめます。

ストレージプラットフォームのシークレットフィールド概要	秘密	Field 概要の略
Azure NetApp Files の特長	ClientID	アプリケーション登録からのクライアント ID
Cloud Volumes Service for GCP	private_key_id です	秘密鍵の ID。CVS 管理者ロールを持つ GCP サービスアカウントの API キーの一部
Cloud Volumes Service for GCP	private_key を使用します	秘密鍵CVS 管理者ロールを持つ GCP サービスアカウントの API キーの一部
Element (NetApp HCI / SolidFire)	エンドポイント	テナントのクレデンシャルを使用する SolidFire クラスタの MVIP

ストレージプラットフォームのシークレットフィールド概要	秘密	Field 概要の略
ONTAP	ユーザ名	クラスタ / SVM に接続するためのユーザ名。クレデンシャルベースの認証に使用されます
ONTAP	パスワード	クラスタ / SVM に接続するためのパスワード。クレデンシャルベースの認証に使用されます
ONTAP	clientPrivateKey	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます
ONTAP	chapUsername のコマンド	インバウンドユーザ名。useCHAP = true の場合は必須。「ONTAP-SAN'」と「ONTAP-SAN-エコノミー」の場合
ONTAP	chapInitiatorSecret	CHAP イニシエータシークレット。useCHAP = true の場合は必須。「ONTAP-SAN'」と「ONTAP-SAN-エコノミー」の場合
ONTAP	chapTargetUsername のコマンド	ターゲットユーザ名。useCHAP = true の場合は必須。「ONTAP-SAN'」と「ONTAP-SAN-エコノミー」の場合
ONTAP	chapTargetInitiatorSecret	CHAP ターゲットイニシエータシークレット。useCHAP = true の場合は必須。「ONTAP-SAN'」と「ONTAP-SAN-エコノミー」の場合

このステップで作成されたシークレットは、次のステップで作成された「TridentBackendConfig」オブジェクトの「PEC.credentials」フィールドで参照されます。

手順2：を作成します TridentBackendConfig **CR**

これで「TridentBackendConfig」CRを作成する準備ができました。この例では'ONTAP-SAN'ドライバを使用するバックエンドは'次に示す TridentBackendConfig オブジェクトを使用して作成されます

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret

```

手順3：のステータスを確認します TridentBackendConfig **CR**

これで「TridentBackendConfig」CRが作成され、ステータスを確認できるようになりました。次の例を参照してください。

```

kubectl -n trident get tbc backend-tbc-ontap-san
NAME                                BACKEND NAME                BACKEND UUID
PHASE    STATUS
backend-tbc-ontap-san  ontap-san-backend          8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8    Bound    Success

```

バックエンドが正常に作成され、「TridentBackendConfig」CRにバインドされました。

フェーズには次のいずれかの値を指定できます。

- Bound: TridentBackendConfig CRはバックエンドに関連付けられており、そのバックエンドにはが含まれています configRef をに設定します TridentBackendConfig crのuid
- Unbound : "" を使用して表現されています「TridentBackendConfig」オブジェクトはバックエンドにバインドされません。新しく作成されたすべての TridentBackendConfig' CRS は、デフォルトでこのフェーズに入ります。フェーズが変更された後、再度 Unbound に戻すことはできません。
- Deleting: TridentBackendConfig CR deletionPolicy が削除対象に設定されました。をクリックします TridentBackendConfig CRが削除され、削除状態に移行します。
 - バックエンドに永続的ボリューム要求 (PVC) が存在しない場合、を削除する TridentBackendConfig`と、TridentはバックエンドとCRを削除します `TridentBackendConfig。
 - バックエンドに1つ以上のPVCが存在する場合は、削除状態になります。次に 'TridentBackendConfig'CR が削除フェーズに入りますバックエンドおよび TridentBackendConfig は、すべてのPVCが削除された後にのみ削除されます。
- lost : 「TridentBackendConfig」CRに関連付けられているバックエンドが誤って削除されたか、意図的に削除されました。「TridentBackendConfig」CRには削除されたバックエンドへの参照がありま

す。「TridentBackendConfig」CRは、「\$eletionPolicy」の値に関係なく削除できます。

- Unknown：TridentはCRに関連付けられたバックエンドの状態または存在を特定できません TridentBackendConfig。たとえば、APIサーバが応答していない場合やCRDが見つからない場合 `tridentbackends.trident.netapp.io` などです。これには介入が必要な場合があります

この段階では、バックエンドが正常に作成されます。など、いくつかの操作を追加で処理することができます "[バックエンドの更新とバックエンドの削除](#)"。

(オプション) 手順 4：詳細を確認します

バックエンドに関する詳細情報を確認するには、次のコマンドを実行します。

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

NAME	BACKEND NAME	BACKEND UUID	
PHASE	STATUS	STORAGE DRIVER	DELETION POLICY
backend-tbc-ontap-san	ontap-san-backend	8d24fce7-6f60-4d4a-8ef6-	
bab2699e6ab8	Bound	Success	ontap-san delete

さらに、「TridentBackendConfig」のYAML / JSON ダンプを取得することもできます。

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: "2021-04-21T20:45:11Z"
  finalizers:
  - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound

```

backendInfo`CRに`応答して作成されたバックエンドの`TridentBackendConfig`とが`backendUUID`格納され`backendName`ます。この`lastOperationStatus`フィールドには、CRの最後の操作のステータスが表示されます。このステータス`TridentBackendConfig`は、ユーザーがトリガーした場合（ユーザーがで何かを変更した場合など）、またはTridentによってトリガーされた場合`spec`（Tridentの再起動中など）です。成功または失敗のいずれかです。phase`CRとバックエンド間の関係のステータスを表します`TridentBackendConfig。上の例では、のphase`値がバインドされています。つまり、CRがバックエンドに関連付けられていることを意味します`TridentBackendConfig。

イベントログの詳細を取得するには、「`kubectl -n trident describe describe tbc <tbc -cr-name>`」コマンドを実行します。



tridentctl を使用して '関連付けられた TridentBackendConfig' オブジェクトを含むバックエンドを更新または削除することはできません。「tridentctl」と「TridentBackendConfig」の切り替えに関連する手順を理解するには、次の手順に従います。["こちらを参照してください"](#)。

バックエンドの管理

kubectl を使用してバックエンド管理を実行します

kubectl を使用してバックエンド管理操作を実行する方法について説明します

バックエンドを削除します

を削除することで、TridentBackendConfig（に基づいて）バックエンドを削除または保持するようにTridentに指示し `deletionPolicy` します。バックエンドを削除するには、`delete` に設定されていることを確認します `deletionPolicy`。のみを削除するには TridentBackendConfig、`retain` に設定されていることを確認します `deletionPolicy`。これにより、バックエンドが引き続き存在し、を使用して管理できます `tridentctl`。

次のコマンドを実行します。

```
kubectl delete tbc <tbc-name> -n trident
```

Tridentでは、で使用されていたKubernetesシークレットは削除されません TridentBackendConfig。Kubernetes ユーザは、シークレットのクリーンアップを担当します。シークレットを削除するときは注意が必要です。シークレットは、バックエンドで使用されていない場合にのみ削除してください。

既存のバックエンドを表示します

次のコマンドを実行します。

```
kubectl get tbc -n trident
```

`tridentctl get backend -n trident` または `tridentctl get backend -o yaml -n trident` を実行して、存在するすべてのバックエンドのリストを取得することもできます。このリストには 'tridentctl' で作成されたバックエンドも含まれます

バックエンドを更新します

バックエンドを更新する理由はいくつかあります。

- ストレージシステムのクレデンシャルが変更されている。クレデンシャルを更新するには、オブジェクトで使用されるKubernetes Secretを `TridentBackendConfig` 更新する必要があります。Tridentは、提供された最新のクレデンシャルでバックエンドを自動的に更新します。次のコマンドを実行して、Kubernetes Secret を更新します。

```
kubectl apply -f <updated-secret-file.yaml> -n trident
```

- パラメータ（使用する ONTAP SVM の名前など）を更新する必要があります。
 - 更新できます TridentBackendConfig 次のコマンドを使用して、Kubernetesから直接オブジェクトを作成します。

```
kubectl apply -f <updated-backend-file.yaml>
```

- または、既存の TridentBackendConfig 次のコマンドを使用してCRを実行します。

```
kubectl edit tbc <tbc-name> -n trident
```



- バックエンドの更新に失敗した場合、バックエンドは最後の既知の設定のまま残ります。ログを表示して原因を確認するには、「`kubectl get tbc <tbc-name> -o yaml -n trident`」または「`kubectl describe tbc <tbc-name> -n trident`」を実行します。
- 構成ファイルで問題を特定して修正したら、`update` コマンドを再実行できます。

tridentctl を使用してバックエンド管理を実行します

tridentctl を使用してバックエンド管理操作を実行する方法について説明します

バックエンドを作成します

を作成したら "[バックエンド構成ファイル](#)"を使用して、次のコマンドを実行します。

```
tridentctl create backend -f <backend-file> -n trident
```

バックエンドの作成に失敗した場合は、バックエンドの設定に何か問題があります。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 -n trident
```

構成ファイルの問題を特定して修正したら、再度 `create` コマンドを実行します

バックエンドを削除します

Tridentからバックエンドを削除するには、次の手順を実行します。

1. バックエンド名を取得します。

```
tridentctl get backend -n trident
```

2. バックエンドを削除します。

```
tridentctl delete backend <backend-name> -n trident
```




TridentでプロビジョニングされたボリュームとこのバックエンドからSnapshotが残っている場合、バックエンドを削除すると、そのバックエンドで新しいボリュームがプロビジョニングされなくなります。バックエンドは「削除」状態のままになり、Tridentは削除されるまでそれらのボリュームとスナップショットを管理し続けます。

既存のバックエンドを表示します

Tridentが認識しているバックエンドを表示するには、次の手順を実行します。

- 概要を取得するには、次のコマンドを実行します。

```
tridentctl get backend -n trident
```

- すべての詳細を確認するには、次のコマンドを実行します。

```
tridentctl get backend -o json -n trident
```

バックエンドを更新します

新しいバックエンド構成ファイルを作成したら、次のコマンドを実行します。

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

バックエンドの更新が失敗した場合、バックエンドの設定に問題があるか、無効な更新を試行しました。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 -n trident
```

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

バックエンドを使用するストレージクラスを特定します

ここでは 'バックエンド・オブジェクトの tridentctl 出力と同じ JSON を使用して回答で実行できる質問の例を示しますこれには 'jq' ユーティリティが使用されますこのユーティリティをインストールする必要があります

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name, storageClasses: [.storage[].storageClasses]|unique}]'
```

これは、「TridentBackendConfig」を使用して作成されたバックエンドにも適用されます。

バックエンド管理オプション間を移動します

Tridentでバックエンドを管理するさまざまな方法について説明します。

バックエンドを管理するためのオプション

を導入しました `TridentBackendConfig` 管理者は現在、バックエンドを2つの方法で管理できるようになっています。これには、次のような質問があります。

- tridentctl を使用して作成したバックエンドは 'TridentBackendConfig' で管理できますか
- 「TridentBackendConfig」を使用して作成したバックエンドは、「tridentctl」を使用して管理できますか。

管理 tridentctl を使用してバックエンドを TridentBackendConfig

このセクションでは 'tridentBackendConfig' オブジェクトを作成して Kubernetes インターフェイスから直接 'tridentctl' を使用して作成されたバックエンドの管理に必要な手順について説明します

これは、次のシナリオに該当します。

- 既存のバックエンドには TridentBackendConfig を使用して作成されたためです tridentctl。
- 「tridentctl」で作成された新しいバックエンドと、その他の「TridentBackendConfig」オブジェクトが存在します。

どちらのシナリオでも、バックエンドは引き続き存在し、Tridentはボリュームをスケジューリングして処理します。管理者には次の2つの選択肢があります。

- tridentctl を使用して 'バックエンドを使用して作成したバックエンドを管理します
- tridentctl を使用して作成されたバックエンドを新しい TridentBackendConfig オブジェクトにバインドしますこれは 'バックエンドが tridentctl' ではなく 'kubectl' を使用して管理されることを意味します

「kubectl」を使用して既存のバックエンドを管理するには、既存のバックエンドにバインドする「TridentBackendConfig」を作成する必要があります。その仕組みの概要を以下に示します。

1. Kubernetes Secret を作成します。シークレットには、Tridentがストレージクラスタ/サービスと通信するために必要なクレデンシャルが含まれています。
2. 「TridentBackendConfig」オブジェクトを作成します。ストレージクラスタ/サービスの詳細を指定し、前の手順で作成したシークレットを参照します。同一の構成パラメータ ('PEC.backendName'`PEC.storagePrefix``PEC.storageDriverName') を指定するように注意する必要があります`PEC.backendName' は '既存のバックエンドの名前に設定する必要があります

手順 0 : バックエンドを特定します

を作成します TridentBackendConfig 既存のバックエンドにバインドする場合は、バックエンド設定を取得する必要があります。この例では、バックエンドが次の JSON 定義を使用して作成されているとします。

```
tridentctl get backend ontap-nas-backend -n trident
+-----+-----+
+-----+-----+-----+-----+
```

NAME	STORAGE DRIVER	UUID
STATE VOLUMES		
ontap-nas-backend	ontap-nas	52f2eb10-e4c6-4160-99fc-96b3be5ab5d7
online	25	

```
cat ontap-nas-backend.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
  "backendName": "ontap-nas-backend",
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "admin-password",

  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {"store": "nas_store"},
  "region": "us_east_1",
  "storage": [
    {
      "labels": {"app": "msoffice", "cost": "100"},
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {"app": "mysqldb", "cost": "25"},
      "zone": "us_east_1d",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
      }
    }
  ]
}
```

```
]
}
```

手順 1 : Kubernetes Secret を作成します

次の例に示すように、バックエンドのクレデンシャルを含むシークレットを作成します。

```
cat tbc-ontap-nas-backend-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password

kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

手順2 : を作成します TridentBackendConfig CR

次の手順では ' (この例のように) 事前に存在する 'ONTAP-NAS-backend' に自動的にバインドされる 'TridentBackendConfig' CR を作成します 次の要件が満たされていることを確認します。

- 「'PEC.backendName'」に同じバックエンド名が定義されています。
- 設定パラメータは元のバックエンドと同じです。
- 仮想プール (存在する場合) は、元のバックエンドと同じ順序である必要があります。
- クレデンシャルは、プレーンテキストではなく、Kubernetes Secret を通じて提供されます。

この場合、「TridentBackendConfig」は次のようになります。

```
cat backend-tbc-ontap-nas.yaml
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
    app: msoffice
    cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
    app: mysqlldb
    cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'

kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

手順3：のステータスを確認します TridentBackendConfig **CR**

「TridentBackendConfig」が作成された後、そのフェーズは「バインド」されている必要があります。また、既存のバックエンドと同じバックエンド名と UUID が反映されている必要があります。

```

kubect1 get tbc tbc-ontap-nas-backend -n trident
NAME                                BACKEND NAME                BACKEND UUID
PHASE    STATUS
tbc-ontap-nas-backend  ontap-nas-backend          52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7    Bound    Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+-----+-----+
+-----+-----+-----+-----+
|          NAME          | STORAGE DRIVER |          UUID          |
| STATE  | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ontap-nas-backend    | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |          25 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

これで 'バックエンドは 'tbc-ontap/nas-backend' TridentBackendConfig' オブジェクトを使用して完全に管理されます

管理 TridentBackendConfig を使用してバックエンドを tridentctl

tridentBackendConfig を使用して作成されたバックエンドを一覧表示するには 'tridentctl を使用しますまた、管理者は、「TridentBackendConfig」を削除し、「pec.deletionPolicy」が「re」に設定されていることを確認することで、「tridentctl」を使用してこのようなバックエンドを完全に管理することもできます。

手順 0 : バックエンドを特定します

たとえば '次のバックエンドが TridentBackendConfig を使用して作成されたとします

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                                BACKEND NAME          BACKEND UUID
PHASE  STATUS  STORAGE DRIVER  DELETION POLICY
backend-tbc-ontap-san  ontap-san-backend  81abcb27-ea63-49bb-b606-
0a5315ac5f82  Bound  Success  ontap-san  delete

tridentctl get backend ontap-san-backend -n trident
+-----+-----+
+-----+-----+-----+-----+
|          NAME          | STORAGE DRIVER |          UUID
| STATE  | VOLUMES |
+-----+-----+
+-----+-----+-----+-----+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

出力からはそのことがわかります TridentBackendConfig は正常に作成され、バックエンドにバインドされています（バックエンドのUUIDを確認してください）。

手順1：確認します deletionPolicy がに設定されます retain

の値を見てみましょう deletionPolicy。これはに設定する必要があり `retain` ます。これにより、CRが削除されてもバックエンド定義が存在し、で管理できるように `TridentBackendConfig` なり `tridentctl` ます。

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                                BACKEND NAME          BACKEND UUID
PHASE  STATUS  STORAGE DRIVER  DELETION POLICY
backend-tbc-ontap-san  ontap-san-backend  81abcb27-ea63-49bb-b606-
0a5315ac5f82  Bound  Success  ontap-san  delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                                BACKEND NAME          BACKEND UUID
PHASE  STATUS  STORAGE DRIVER  DELETION POLICY
backend-tbc-ontap-san  ontap-san-backend  81abcb27-ea63-49bb-b606-
0a5315ac5f82  Bound  Success  ontap-san  retain
```



「削除ポリシー」が「再取得」に設定されていない限り、次の手順に進まないでください。

手順2: を削除します TridentBackendConfig CR

最後の手順は、「TridentBackendConfig」CRを削除することです。「削除ポリシー」が「取得」に設定されていることを確認したら、削除を続行できます。

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-----+-----+-----+
+-----+-----+-----+
|          NAME          | STORAGE DRIVER |          UUID          |
| STATE | VOLUMES |          |
+-----+-----+-----+
+-----+-----+-----+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |          33 |
+-----+-----+-----+
+-----+-----+-----+
```

オブジェクトが削除されると、`TridentBackendConfig` Tridentは実際にはバックエンド自体を削除せずにオブジェクトを削除します。

ストレージクラスの作成と管理

ストレージクラスを作成する。

Kubernetes StorageClassオブジェクトを設定してストレージクラスを作成し、Tridentでボリュームのプロビジョニング方法を指定します。

Kubernetes StorageClassオブジェクトの設定

は、"[Kubernetes StorageClassオブジェクト](#)"そのクラスで使用されるプロビジョニングツールとしてTridentを識別し、ボリュームのプロビジョニング方法をTridentに指示します。例：


```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

ストレージクラスとパラメータおよびパラメータとの連携によるTridentによるボリュームのプロビジョニング方法の詳細については PersistentVolumeClaim、を参照してください"[Kubernetes オブジェクトと Trident オブジェクト](#)".

ストレージクラスを作成する。

StorageClassオブジェクトを作成したら、ストレージクラスを作成できます。 [[ストレージクラスノサンプル](#)] に、使用または変更できる基本的なサンプルを示します。

手順

1. これはKubernetesオブジェクトなので、 `kubectl` をクリックしてKubernetesで作成します。

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. KubernetesとTridentの両方で「basic-csi」ストレージクラスが表示され、Tridentがバックエンドでプールを検出していることを確認します。

```

kubect1 get sc basic-csi
NAME          PROVISIONER          AGE
basic-csi     csi.trident.netapp.io 15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}

```

ストレージクラスノサンプル

Tridentが提供し ["特定のバックエンド向けのシンプルなストレージクラス定義"](#)ます。

または、 `sample-input/storage-class-csi.yaml.templ` インストーラに付属しており、`BACKEND_TYPE` ストレージドライバの名前を指定します。

```

./tridentctl -n trident get backend
+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |         0 |
+-----+-----+-----+-----+
+-----+-----+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml

```

ストレージクラスを管理する

既存のストレージクラスを表示したり、デフォルトのストレージクラスを設定したり、ストレージクラスバックエンドを識別したり、ストレージクラスを削除したりできます。

既存のストレージクラスを表示します

- 既存の Kubernetes ストレージクラスを表示するには、次のコマンドを実行します。

```
kubectl get storageclass
```

- Kubernetes ストレージクラスの詳細を表示するには、次のコマンドを実行します。

```
kubectl get storageclass <storage-class> -o json
```

- Tridentの同期されたストレージクラスを表示するには、次のコマンドを実行します。

```
tridentctl get storageclass
```

- 同期されたTridentのストレージクラスの詳細を表示するには、次のコマンドを実行します。

```
tridentctl get storageclass <storage-class> -o json
```

デフォルトのストレージクラスを設定する

Kubernetes 1.6 では、デフォルトのストレージクラスを設定する機能が追加されています。永続ボリューム要求（PVC）に永続ボリュームが指定されていない場合に、永続ボリュームのプロビジョニングに使用するストレージクラスです。

- ストレージクラスの定義でアノテーションの「storageclass.kubernetes.io/is-default-class」を true に設定して、デフォルトのストレージクラスを定義します。仕様に依じて、それ以外の値やアノテーションがない場合は false と解釈されます。
- 次のコマンドを使用して、既存のストレージクラスをデフォルトのストレージクラスとして設定できます。

```
kubectl patch storageclass <storage-class-name> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 同様に、次のコマンドを使用して、デフォルトのストレージクラスアノテーションを削除できます。

```
kubectl patch storageclass <storage-class-name> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

また、このアノテーションが含まれている Trident インストーラバンドルにも例があります。



クラスタ内のデフォルトのストレージクラスは一度に1つだけにしてください。Kubernetes では、技術的に複数のストレージを使用することはできますが、デフォルトのストレージクラスがまったくない場合と同様に動作します。

ストレージクラスのバックエンドを特定します

これは、Tridentバックエンドオブジェクト用に出力するJSONを使用して回答できる質問の例 `tridentctl` です。これはユーティリティを使用し `jq` します。このユーティリティは、最初にインストールする必要がある場合があります。

```
tridentctl get storageclass -o json | jq '[.items[] | {storageClass: .Config.name, backends: [.storage]|unique}]'
```

ストレージクラスを削除する

Kubernetes からストレージクラスを削除するには、次のコマンドを実行します。

```
kubectl delete storageclass <storage-class>
```

「<storage-class>」は、ご使用のストレージクラスに置き換えてください。

このストレージクラスを使用して作成された永続ボリュームは変更されず、Tridentで引き続き管理されます。



Tridentでは、作成するボリュームに対して空白が適用され `fsType``ます。iSCSIバックエンドの場合は、StorageClassで強制することを推奨します `parameters.fsType``。既存のストレージクラスを削除し、指定したで再作成してください `parameters.fsType``。

ボリュームのプロビジョニングと管理

ボリュームをプロビジョニングする

設定したKubernetes StorageClassを使用してPVへのアクセスを要求するPersistentVolume (PV) とPersistentVolumeClaim (PVC) を作成します。その後、PVをポッドにマウントできます。

概要

A "[永続ボリューム_](#)" (PV) は、Kubernetesクラスタ上のクラスタ管理者がプロビジョニングする物理ストレージリソースです。。 "[PersistentVolumeClaim_](#)" (PVC) は、クラスタ上のPersistentVolumeへのアクセス要求です。

PVCは、特定のサイズまたはアクセスモードのストレージを要求するように設定できます。クラスタ管理者は、関連付けられているStorageClassを使用して、PersistentVolumeのサイズとアクセスモード（パフォーマンスやサービスレベルなど）以上を制御できます。

PVとPVCを作成したら、ポッドにボリュームをマウントできます。

マニフェストの例

PersistentVolumeサンプルマニフェスト

このサンプルマニフェストは、StorageClassに関連付けられた10Giの基本PVを示しています。basic-csi。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

PersistentVolumeClaim サンプルマニフェスト

次に、基本的なPVC設定オプションの例を示します。

RWOアクセスを備えたPVC

次の例は、という名前のStorageClassに関連付けられた、RWOアクセスが設定された基本的なPVCを示しています。basic-csi。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

NVMe / TCP対応PVC

この例は、という名前のStorageClassに関連付けられたNVMe/TCPの基本的なPVCとRWOアクセスを示しています。protection-gold。

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san-nvme
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: protection-gold
```

PODマニフェストのサンプル

次の例は、PVCをポッドに接続するための基本的な設定を示しています。

キホンセット

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
        claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```


NVMe/TCPの基本構成

```
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
    volumeMounts:
    - mountPath: "/usr/share/nginx/html"
      name: task-pv-storage
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
  - name: task-pv-storage
    persistentVolumeClaim:
      claimName: pvc-san-nvme
```

PVおよびPVCの作成

手順

1. PVを作成します。

```
kubectl create -f pv.yaml
```

2. PVステータスを確認します。

```
kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS  REASON    AGE
pv-storage    4Gi       RWO           Retain          Available
7s
```

3. PVCを作成します。

```
kubectl create -f pvc.yaml
```

4. PVCステータスを確認します。

```
kubectl get pvc
NAME          STATUS  VOLUME          CAPACITY  ACCESS  MODES  STORAGECLASS  AGE
pvc-storage  Bound  pv-name  2Gi      RWO                                5m
```

5. ボリュームをポッドにマウントします。

```
kubectl create -f pv-pod.yaml
```



進捗状況は次を使用して監視できます。 `kubectl get pod --watch`。

6. ボリュームがマウントされていることを確認します。 `/my/mount/path`。

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

7. ポッドを削除できるようになりました。Podアプリケーションは存在しなくなりますが、ボリュームは残ります。

```
kubectl delete pod task-pv-pod
```

ストレージクラスとパラメータおよびパラメータとの連携によるTridentによるボリュームのプロビジョニング方法の詳細については [PersistentVolumeClaim](#)、を参照してください"[Kubernetes オブジェクトと Trident オブジェクト](#)"。

ボリュームを展開します

Tridentを使用すると、Kubernetesユーザは作成後にボリュームを拡張できます。ここでは、iSCSI ボリュームと NFS ボリュームの拡張に必要な設定について説明します。

iSCSI ボリュームを展開します

CSI プロビジョニングを使用して、iSCSI Persistent Volume (PV) を拡張できます。



iSCSI ボリュームの拡張は 'ONTAP-SAN' ONTAP-SAN-エコノミー 'olidfire-SAN' ドライバによってサポートされており 'Kubernetes 1.16 以降が必要です

手順 1 : ボリュームの拡張をサポートするようにストレージクラスを設定する

StorageClass定義を編集して `allowVolumeExpansion` フィールドからに移動します `true`。

```
cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

既存の StorageClass の場合は 'allowVolumeExpansion' パラメータを含めるように編集します

手順 2 : 作成した **StorageClass** を使用して **PVC** を作成します

PVC定義を編集し、`spec.resources.requests.storage` 新たに必要となったサイズを反映するには、元のサイズよりも大きくする必要があります。

```
cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Tridentは永続的ボリューム (PV) を作成し、この永続的ボリューム要求 (PVC) に関連付けます。

```
kubectl get pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound     pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO          ontap-san    8s

kubectl get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY  STATUS    CLAIM                STORAGECLASS  REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi      RWO
Delete        Bound     default/san-pvc     ontap-san    10s
```

手順 3 : PVC を接続するポッドを定義します

サイズを変更するポッドにPVを接続します。iSCSI PV のサイズ変更には、次の 2 つのシナリオがあります。

- PVがポッドに接続されている場合、Tridentはストレージバックエンド上のボリュームを拡張し、デバイスを再スキャンして、ファイルシステムのサイズを変更します。
- 接続されていないPVのサイズを変更しようとする、Tridentはストレージバックエンド上のボリュームを拡張します。PVC がポッドにバインドされると、Trident はデバイスを再スキャンし、ファイルシステムのサイズを変更します。展開操作が正常に完了すると、Kubernetes は PVC サイズを更新します。

この例では、ポッドが作成され、「1-pvc」が使用されます。

```
kubectl get pod
NAME          READY    STATUS    RESTARTS   AGE
ubuntu-pod    1/1     Running   0          65s

kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:    default
StorageClass:  ontap-san
Status:       Bound
Volume:       pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:       <none>
Annotations:  pv.kubernetes.io/bind-completed: yes
              pv.kubernetes.io/bound-by-controller: yes
              volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:   [kubernetes.io/pvc-protection]
Capacity:    1Gi
Access Modes: RWO
VolumeMode:  Filesystem
Mounted By:  ubuntu-pod
```

ステップ 4 : PV を展開します

1Gi から 2Gi に作成された PV のサイズを変更するには、PVC の定義を編集し、「`PEC.resources.request.storage`」を 2Gi に更新します。

```
kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
  ...
```

手順 5 : 拡張を検証する

PVC、PV、およびTridentボリュームのサイズを確認することで、拡張が正常に機能したことを検証できます。

```

kubect1 get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO          ontap-san    11m
kubect1 get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY  STATUS    CLAIM          STORAGECLASS  REASON  AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi      RWO
Delete        Bound     default/san-pvc  ontap-san    12m
tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          |  STATE  |  MANAGED  |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san    |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

NFS ボリュームを拡張します

Tridentでは、`ontap-nas-economy` `ontap-nas-flexgroup`、`gcp-cvs` `azure-netapp-files` およびバックエンドでプロビジョニングされるNFS PVsのボリューム拡張がサポートされます `ontap-nas`。

手順 1 : ボリュームの拡張をサポートするようにストレージクラスを設定する

NFS PV のサイズを変更するには 'まず' `allowVolumeExpansion` フィールドを `true` に設定してボリュームを拡張できるようにストレージ・クラスを構成する必要があります

```

cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true

```

このオプションを指定せずにすでにストレージ・クラスを作成している場合は `kubect1 Edit storageclass` を使用して既存のストレージ・クラスを編集するだけで `ボリュームの拡張が可能になります`

手順 2 : 作成した **StorageClass** を使用して **PVC** を作成します

```
cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

TridentはこのPVC用に20MiBのNFS PVを作成する必要があります。

```
kubectl get pvc
NAME                STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
ontapnas20mb       Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi       RWO            ontapnas       9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS    CLAIM                STORAGECLASS   REASON   AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi       RWO            Delete           Bound    default/ontapnas20mb  ontapnas   2m42s
```

ステップ 3 : **PV** を展開します

新しく作成した20MiBのPVのサイズを1GiBに変更するには、そのPVCを編集してを設定します
spec.resources.requests.storage 1GiBへ :

```
kubectl edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  ...
```

手順 4 : 拡張を検証する

PVC、PV、およびTridentボリュームのサイズを確認することで、サイズ変更が正しく機能したかどうかを検証できます。


```

kubect1 get pvc ontapnas20mb
NAME                STATUS      VOLUME
CAPACITY  ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb  Bound        pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi
RWO                ontapnas          4m44s

kubect1 get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                CAPACITY  ACCESS MODES
RECLAIM POLICY     STATUS    CLAIM                STORAGECLASS  REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi      RWO
Delete            Bound     default/ontapnas20mb  ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          |  STATE  | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file      | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

ボリュームをインポート

tridentctl import を使用して、既存のストレージボリュームを Kubernetes PV としてインポートできます。

概要と考慮事項

Tridentにボリュームをインポートする目的は次のとおりです。

- アプリケーションをコンテナ化し、既存のデータセットを再利用する
- 一時的なアプリケーションにはデータセットのクローンを使用
- 障害が発生したKubernetesクラスタを再構築します
- ディザスタリカバリ時にアプリケーションデータを移行

考慮事項

ボリュームをインポートする前に、次の考慮事項を確認してください。

- Tridentでインポートできるのは、RW（読み取り/書き込み）タイプのONTAPボリュームのみです。DP（データ保護）タイプのボリュームはSnapMirrorデスティネーションボリュームです。ボリュームをTrident

にインポートする前に、ミラー関係を解除する必要があります。

- アクティブな接続がないボリュームをインポートすることを推奨します。アクティブに使用されているボリュームをインポートするには、ボリュームのクローンを作成してからインポートを実行します。



Kubernetesは以前の接続を認識せず、アクティブなボリュームをポッドに簡単に接続できるため、これはブロックボリュームで特に重要です。その結果、データが破損する可能性があります。

- PVCで指定する必要がありますが、`StorageClass` Tridentはインポート時にこのパラメータを使用しません。ストレージクラスは、ボリュームの作成時に、ストレージ特性に基づいて使用可能なプールから選択するために使用されます。ボリュームはすでに存在するため、インポート時にプールを選択する必要はありません。そのため、PVCで指定されたストレージクラスと一致しないバックエンドまたはプールにボリュームが存在してもインポートは失敗しません。
- 既存のボリュームサイズはPVCで決定され、設定されます。ストレージドライバによってボリュームがインポートされると、PVはClaimRefを使用してPVCに作成されます。
 - 再利用ポリシーは、最初から設定されています retain PVにあります。KubernetesがPVCとPVを正常にバインドすると、再利用ポリシーがストレージクラスの再利用ポリシーに合わせて更新されます。
 - ストレージクラスの再利用ポリシーが`delete`にすると、PVが削除されるとストレージボリュームが削除されます。
- デフォルトでは、TridentはPVCを管理し、バックエンドでFlexVolとLUNの名前を変更します。フラグを渡して管理対象外のボリュームをインポートできます `--no-manage`。を使用する場合 `--no-manage`、Tridentはオブジェクトのライフサイクル中、PVCまたはPVに対して追加の操作を実行しません。PVが削除されてもストレージボリュームは削除されず、ボリュームのクローンやボリュームのサイズ変更などのその他の処理も無視されます。



このオプションは、コンテナ化されたワークロードにKubernetesを使用するが、Kubernetes以外でストレージボリュームのライフサイクルを管理する場合に便利です。

- PVCとPVにアノテーションが追加されます。このアノテーションは、ボリュームがインポートされたこと、およびPVCとPVが管理されていることを示す二重の目的を果たします。このアノテーションは変更または削除しないでください。

ボリュームをインポートします

を使用できます `tridentctl import` をクリックしてボリュームをインポートします。

手順

1. Persistent Volume Claim (PVC; 永続的ボリューム要求) ファイルを作成します (例: `pvc.yaml`) をクリックします。PVCファイルには、が含まれている必要があります `name`、`namespace`、`accessModes` および `storageClassName`。必要に応じて、を指定できます `unixPermissions` 定義されています。

最小仕様の例を次に示します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```



PV名やボリュームサイズなどの追加のパラメータは指定しないでください。これにより原因、インポートコマンドが失敗する可能性があります。

2. コマンドを使用して `tridentctl import`、ボリュームを含むTridentバックエンドの名前と、ストレージ上のボリュームを一意に識別する名前（ONTAP FlexVol、Element Volume、Cloud Volumes Serviceパスなど）を指定します。`-f`PVCファイルへのパスを指定するには、引数が必要です。

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
```

例

サポートされているドライバについて、次のボリュームインポートの例を確認してください。

ONTAP NASおよびONTAP NAS FlexGroup

Tridentは、ドライバと `ontap-nas-flexgroup` ドライバを使用したボリュームインポートをサポートしている `ontap-nas` ます



- `ontap-nas-economy` ドライバで `qtree` をインポートおよび管理できない。
- `ontap-nas` および `ontap-nas-flexgroup` ドライバでボリューム名の重複が許可されていません。

「`ontap/nas`」ドライバで作成される各ボリュームは、ONTAP クラスタ上の FlexVol です。「`ontap/nas`」ドライバを使用して FlexVol をインポートする方法は同じです。ONTAP クラスタにすでに存在する FlexVol は 'ONTAP-NAS'PVC としてインポートできます同様に、FlexGroup ボリュームは「`ONTAP-NAS-flexgroup`」PVC としてインポートできます。

ONTAP NASの例

次の例は、管理対象ボリュームと管理対象外ボリュームのインポートを示しています。

管理対象ボリューム

次の例は、という名前のボリュームをインポートします managed_volume という名前のバックエンドで ontap_nas :

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
file	pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	online	standard	true

管理対象外のボリューム

引数を使用した場合 --no-manage、Tridentはボリュームの名前を変更しません。

次に、をインポートする例を示します unmanaged_volume をクリックします ontap_nas バックエンド:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file> --no-manage
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
file	pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	online	standard	false

ONTAP SAN

Tridentは、ドライバと `ontap-san-economy` ドライバを使用したボリュームインポートをサポートしている `ontap-san` ます

Tridentでは、単一のLUNを含むONTAP SAN FlexVolをインポートできます。これは、ドライバと一致している `ontap-san` ます。ドライバは、PVCごとにFlexVolを作成し、FlexVol内にLUNを作成します。TridentはFlexVol

をインポートし、PVC定義に関連付けます。

ONTAP SANの例

次の例は、管理対象ボリュームと管理対象外ボリュームのインポートを示しています。

管理対象ボリューム

管理対象ボリュームの場合、TridentはFlexVolの名前を形式に、FlexVol内のLUNの名前をに`lun0`変更`pvc-<uuid>`します。

次の例は、をインポートします `ontap-san-managed` にあるFlexVol `ontap_san_default` バックエンド：

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-basic-import.yaml -n trident -d
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic          |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

管理対象外のボリューム

次に、をインポートする例を示します `unmanaged_example_volume` をクリックします `ontap_san` バックエンド：

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE  | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

次の例に示すように、KubernetesノードのIQNとIQNを共有するigroupにLUNをマッピングすると、エラーが

表示されます。LUN already mapped to initiator(s) in this group。ボリュームをインポートするには、イニシエータを削除するか、LUNのマッピングを解除する必要があります。

Vserver	Igroup	Protocol	OS Type	Initiators
svm0	k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3	iscsi	linux	iqn.1994-05.com.redhat:4c2e1cf35e0
svm0	unmanaged-example-igroup	mixed	linux	iqn.1994-05.com.redhat:4c2e1cf35e0

要素 (Element)

Tridentは、NetApp Elementソフトウェアとドライバを使用したNetApp HCIボリュームインポートをサポートしています solidfire-san。



Element ドライバではボリューム名の重複がサポートされます。ただし、ボリューム名が重複している場合、Tridentはエラーを返します。回避策としてボリュームをクローニングし、一意のボリューム名を指定して、クローンボリュームをインポートします。

要素の例

次に、をインポートする例を示します element-managed バックエンドのボリューム element_default。

```
tridentctl import volume element_default element-managed -f pvc-basic-import.yaml -n trident -d
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
block	pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe	d3ba047a-ea0b-43f9-9c42-e38e58301c49	10 GiB	online	basic-element	true

Google Cloud Platform の 1 つです

Tridentはドライバを使用したボリュームインポートをサポートして `gcp-cvs` ます。



NetApp Cloud Volumes Serviceから作成されたボリュームをGoogle Cloud Platformにインポートするには、ボリュームパスでボリュームを特定します。ボリュームパスは、ボリュームのエクスポートパスのの続く部分です :/。たとえば、エクスポートパスがの場合などです 10.0.0.1:/adroit-jolly-swift、ボリュームのパスはです adroit-jolly-swift。

Google Cloud Platformの例

次に、をインポートする例を示します gcp-cvs バックエンドのボリューム gcpcvs_YEppr を指定します adroit-jolly-swift。

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-
file> -n trident
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
	pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55	e1a6e65b-299e-4568-ad05-4f0a105c888f	93 GiB	online	gcp-storage	true
					file	

Azure NetApp Files の特長

Tridentはドライバを使用したボリュームインポートをサポートしてい`azure-netapp-files`ます。



Azure NetApp Filesボリュームをインポートするには、ボリュームパスでボリュームを特定します。ボリュームパスは、ボリュームのエクスポートパスのの続く部分です :/。たとえば、マウントパスがの場合などです 10.0.0.2:/importvoll1、ボリュームのパスはです importvoll1。

Azure NetApp Filesの例

次に、をインポートする例を示します azure-netapp-files バックエンドのボリューム azurenetappfiles_40517 を指定します importvoll1。

```
tridentctl import volume azurenetappfiles_40517 importvoll1 -f <path-to-pvc-file> -n trident
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE  | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage  |
file      | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

ボリュームの名前とラベルをカスタマイズする

Tridentでは、作成したボリュームにわかりやすい名前とラベルを割り当てることができます。これにより、ボリュームを特定し、それぞれのKubernetesリソース（PVC）に簡単にマッピングできます。また、バックエンドレベルでテンプレートを定義してカスタムボリューム名とカスタムラベルを作成することもできます。作成、インポート、またはクローンを作成するボリュームは、テンプレートに準拠します。

作業を開始する前に

カスタマイズ可能なボリューム名とラベルのサポート：

1. ボリュームの作成、インポート、クローニングの各処理。
2. ontap-nas-economyドライバの場合、qtreeボリュームの名前だけがテンプレート名に準拠します。
3. ontap-san-economyドライバの場合、名前テンプレートに準拠するのはLUN名のみです。

制限

1. カスタマイズ可能なボリューム名は、ONTAPオンプレミスドライバとのみ互換性があります。
2. カスタマイズ可能なボリューム名は、既存のボリュームには適用されません。

カスタマイズ可能なボリューム名の主な動作

1. 名前テンプレートの無効な構文が原因でエラーが発生した場合、バックエンドの作成は失敗します。ただし、テンプレートアプリケーションが失敗した場合は、既存の命名規則に従ってボリュームに名前が付けられます。
2. バックエンド構成の名前テンプレートを使用してボリュームの名前が指定されている場合、ストレージプレフィックスは適用されません。任意のプレフィックス値をテンプレートに直接追加できます。

名前テンプレートとラベルを使用したバックエンド構成の例

カスタム名テンプレートは、ルートレベルまたはプールレベルで定義できます。

ルートレベルの例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
    "{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.RequestName}}"
  },
  "labels": {"cluster": "ClusterA", "PVC":
    "{{.volume.Namespace}}_{{.volume.RequestName}}"}
}
```

プールレベルの例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels":{"labelname":"label1", "name": "{{ .volume.Name }}"},
      "defaults":
      {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
        }}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels":{"cluster":"label2", "name": "{{ .volume.Name }}"},
      "defaults":
      {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
        }}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

名前テンプレートの例

例1:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

例2:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}"
```

考慮すべきポイント

1. ボリュームインポートの場合、既存のボリュームに特定の形式のラベルがある場合にのみラベルが更新されます。例：{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}。
2. 管理対象ボリュームのインポートの場合、ボリューム名はバックエンド定義のルートレベルで定義された名前テンプレートの後に続きます。
3. Tridentでは、storageプレフィックスを指定したスライス演算子の使用はサポートされていません。
4. テンプレートによってボリューム名が一意にならない場合、Tridentではいくつかのランダムな文字が追加されて一意のボリューム名が作成されます。
5. NASエコノミーボリュームのカスタム名の長さが64文字を超える場合、Tridentは既存の命名規則に従ってボリュームに名前を付けます。他のすべてのONTAPドライバでは、ボリューム名が名前の上限を超えると、ボリュームの作成プロセスが失敗します。

ネームスペース間でNFSボリュームを共有します

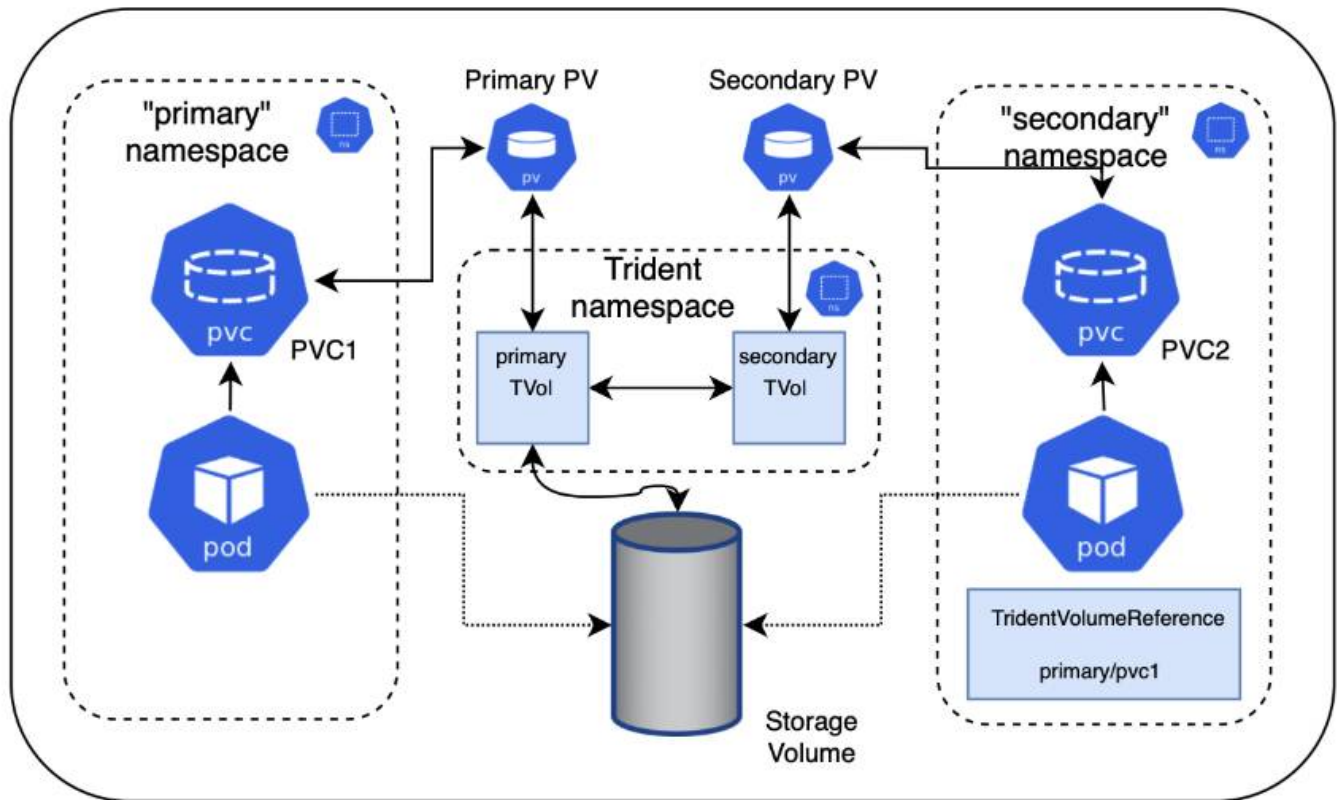
Tridentを使用すると、プライマリネームスペースにボリュームを作成し、1つ以上のセカンダリネームスペースで共有できます。

の機能

TridentVolumeReference CRを使用すると、1つ以上のKubernetesネームスペース間でReadWriteMany (RWX) NFSボリュームを安全に共有できます。このKubernetesネイティブ解決策には、次のようなメリットがあります。

- セキュリティを確保するために、複数のレベルのアクセス制御が可能です
- すべてのTrident NFSボリュームドライバで動作
- tridentctlやその他の非ネイティブのKubernetes機能に依存しません

この図は、2つのKubernetesネームスペース間でのNFSボリュームの共有を示しています。



クイックスタート

NFSボリューム共有はいくつかの手順で設定できます。

1

ボリュームを共有するようにソースPVCを設定します

ソース名前空間の所有者は、ソースPVCのデータにアクセスする権限を付与します。

2

デスティネーション名前空間にCRを作成する権限を付与します

クラスタ管理者が、デスティネーション名前空間の所有者にTridentVolumeReference CRを作成する権限を付与します。

3

デスティネーション名前空間にTridentVolumeReferenceを作成します

宛先名前空間の所有者は、送信元PVCを参照するためにTridentVolumeReference CRを作成します。

4

宛先名前空間に下位PVCを作成します

宛先名前空間の所有者は、送信元PVCからのデータソースを使用する下位PVCを作成します。

ソース名前空間とデスティネーション名前空間を設定します

セキュリティを確保するために、名前空間間共有では、ソース名前空間の所有者、クラスタ管理

者、および宛先名前空間の所有者によるコラボレーションとアクションが必要です。ユーザーロールは各手順で指定します。

手順

1. ソース名前空間の所有者：PVCを作成します (pvc1) をソース名前空間に追加し、デスティネーション名前空間との共有権限を付与します (namespace2) を使用します shareToNamespace アノテーション

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Tridentは、PVとそのバックエンドNFSストレージボリュームを作成します。



- カンマ区切りリストを使用して、複数の名前空間にPVCを共有できます。例：
trident.netapp.io/shareToNamespace:
namespace2, namespace3, namespace4。
- *を使用して、すべての名前空間に共有できます *。例：
trident.netapp.io/shareToNamespace: *
- PVCを更新してを含めることができます shareToNamespace アノテーションはいつでも作成できます。

2. *クラスタ管理者：*カスタムロールとkubecfgを作成して、デスティネーション名前空間の所有者にTridentVolumeReference CRを作成する権限を付与します。
3. *デスティネーション名前空間所有者：*ソース名前空間を参照するデスティネーション名前空間にTridentVolumeReference CRを作成します pvc1。

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

- 宛先名前空間の所有者：PVCを作成します (pvc2) をデスティネーション名前スペースに展開します (namespace2)を使用します shareFromPVC 送信元PVCを指定する注釈。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```



宛先PVCのサイズは、送信元PVCのサイズ以下である必要があります。

結果

TridentはデスティネーションPVCのアノテーションを読み取り shareFromPVC、ソースPVストレージリソースを共有する独自のストレージリソースのない下位ボリュームとしてデスティネーションPVを作成します。宛先PVCとPVは、通常どおりバインドされているように見えます。

共有ボリュームを削除

複数の名前スペースで共有されているボリュームは削除できます。Tridentは、ソース名前スペース上のボリュームへのアクセスを削除し、そのボリュームを共有する他の名前スペースへのアクセスを維持します。このボリュームを参照している名前スペースをすべて削除すると、Tridentによってボリュームが削除されません。

使用 `tridentctl get` 下位のボリュームを照会する

を使用する[`tridentctl`ユーティリティを使用すると、を実行できます `get` コマンドを使用して下位のボリュームを取得します。詳細については、[リンク:./trident-reference/tridentctl.html](https://docs.netapp.com/us-en/trident-reference/tridentctl.html)を参照してください

[tridentctl コマンドとオプション]。

Usage:

```
tridentctl get [option]
```

フラグ:

- `-h, --help`: ボリュームのヘルプ。
- `--parentOfSubordinate string`: クエリを下位のソースボリュームに制限します。
- `--subordinateOf string`: クエリをボリュームの下位に制限します。

制限

- Tridentでは、デスティネーション名前スペースが共有ボリュームに書き込まれないようにすることはできません。共有ボリュームのデータの上書きを防止するには、ファイルロックなどのプロセスを使用する必要があります。
- を削除しても、送信元PVCへのアクセスを取り消すことはできません `shareToNamespace` または `shareFromNamespace` 注釈またはを削除します `TridentVolumeReference CR`。アクセスを取り消すには、下位PVCを削除する必要があります。
- Snapshot、クローン、およびミラーリングは下位のボリュームでは実行できません。

を参照してください。

名前スペース間のボリュームアクセスの詳細については、次の資料を参照してください。

- [にアクセスします "名前スペース間でのボリュームの共有：名前スペース間のボリュームアクセスを許可する場合は「Hello」と入力します"](#)。
- [のデモをご覧ください "ネットアップTV"](#)。

CSI トポロジを使用します

Tridentでは、を使用して、Kubernetesクラスタ内のノードを選択的に作成して接続できます ["CSI トポロジ機能"](#)。

概要

CSI トポロジ機能を使用すると、領域およびアベイラビリティゾーンに基づいて、ボリュームへのアクセスをノードのサブセットに制限できます。現在、クラウドプロバイダは、Kubernetes 管理者がゾーンベースのノードを生成できるようになっています。ノードは、リージョンによって異なるアベイラビリティゾーンに配置することも、リージョンによって配置することもできます。マルチゾーンアーキテクチャでワークロード用のボリュームのプロビジョニングを容易にするために、TridentではCSIトポロジを使用しています。



CSI トポロジ機能の詳細については、[を参照してください "こちらをご覧ください"](#)。

Kubernetes には、2つの固有のボリュームバインドモードがあります。

- `VolumeBindingMode` をに設定する `Immediate` と、Tridentはトポロジを認識せずにボリュームを作成します。ボリュームバインディングと動的プロビジョニングは、PVCが作成されるときに処理されます。これはデフォルト `VolumeBindingMode` であり、トポロジの制約を適用しないクラスタに適しています。永続ボリュームは、要求元ポッドのスケジュール要件に依存することなく作成されます。
- VolumeBindingMode を「WaitForFirstConsumer」に設定すると、PVCの永続ボリュームの作成とバインドは、PVCを使用するポッドがスケジュールされて作成されるまで遅延されます。これにより、トポロジの要件に応じたスケジュールの制約を満たすようにボリュームが作成されます。



「WaitForFirstConsumer」バインディングモードでは、トポロジラベルは必要ありません。これはCSIトポロジ機能とは無関係に使用できます。

必要なもの

CSIトポロジを使用するには、次のものがが必要です。

- を実行するKubernetesクラスタ ["サポートされるKubernetesバージョン"](#)

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- クラスタ内のノードには、トポロジ対応と `topology.kubernetes.io/zone` を示すラベルを付ける必要があります(`topology.kubernetes.io/region` ます。これらのラベル*は、Tridentをトポロジ対応にするためにTridentをインストールする前に、クラスタ内のノード*に設定しておく必要があります。


```
kubectl get nodes -o=jsonpath='{range .items[*]}[.metadata.name],
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kuber-
netes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kuber-
netes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kuber-
netes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

手順 1 : トポロジ対応バックエンドを作成する

Tridentストレージバックエンドは、アベイラビリティゾーンに基づいて選択的にボリュームをプロビジョニングするように設計できます。各バックエンドは、サポートされているゾーンとリージョンのリストを表すオプションのブロックを運ぶことができます `supportedTopologies`。ストレージクラスがそのようなバックエンドを使用する場合、ボリュームは、サポートされているリージョン/ゾーンでスケジューリングされているアプリケーションから要求された場合にのみ作成されます。

バックエンド定義の例を次に示します。

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-a
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-b
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {"topology.kubernetes.io/region": "us-east1",
     "topology.kubernetes.io/zone": "us-east1-a"},
    {"topology.kubernetes.io/region": "us-east1",
     "topology.kubernetes.io/zone": "us-east1-b"}
  ]
}
```



`supportedTopologies`は、バックエンドごとにリージョンとゾーンのリストを提供するために使用されます。これらのリージョンとゾーンは、StorageClassで指定できる許容値のリストを表します。バックエンドで提供されるリージョンとゾーンのサブセットを含むストレージクラスの場合、Tridentはバックエンドにボリュームを作成します。

また 'ストレージ・プールごとに 'supportedTopologies を定義することもできます次の例を参照してください。

```

---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-a
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-b
storage:
- labels:
    workload: production
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b

```

この例では、「region」および「zone」ラベルはストレージプールの場所を表しています。「topology.kubernetes.io/region」と「topology.kubernetes.io/zone」は、ストレージプールの消費元を決定します。

手順 2：トポロジを認識するストレージクラスを定義する

クラスタ内のノードに提供されるトポロジラベルに基づいて、トポロジ情報を含めるように StorageClasses を定義できます。これにより、作成された PVC 要求の候補となるストレージプール、および Trident によってプロビジョニングされたボリュームを使用できるノードのサブセットが決まります。

次の例を参照してください。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"
```

前述のStorageClass定義では、volumeBindingMode`がに設定されて`WaitForFirstConsumer`います。このStorageClassで要求されたPVCは、ポッドで参照されるまで処理されません。およびに、`allowedTopologies`使用するゾーンとリージョンを示します。StorageClassは`netapp-san-us-east1`、上記で定義したバックエンドにPVCを作成し`san-backend-us-east1`ます。

ステップ 3 : PVC を作成して使用する

StorageClass を作成してバックエンドにマッピングすると、PVC を作成できるようになりました。

以下の例「PEC」を参照してください。

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

このマニフェストを使用してPVCを作成すると、次のような結果になります。

```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san      Pending
2s
kubect1 describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type      Reason              Age   From
  ----      -
  Normal    WaitForFirstConsumer 6s    persistentvolume-controller
waiting
for first consumer to be created before binding

```

Trident でボリュームを作成して PVC にバインドするには、ポッド内の PVC を使用します。次の例を参照してください。

```

apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: topology.kubernetes.io/region
                operator: In
                values:
                  - us-east1
            preferredDuringSchedulingIgnoredDuringExecution:
              - weight: 1
                preference:
                  matchExpressions:
                    - key: topology.kubernetes.io/zone
                      operator: In
                      values:
                        - us-east1-a
                        - us-east1-b
      securityContext:
        runAsUser: 1000
        runAsGroup: 3000
        fsGroup: 2000
    volumes:
      - name: voll
        persistentVolumeClaim:
          claimName: pvc-san
    containers:
      - name: sec-ctx-demo
        image: busybox
        command: [ "sh", "-c", "sleep 1h" ]
        volumeMounts:
          - name: voll
            mountPath: /data/demo
        securityContext:
          allowPrivilegeEscalation: false

```

この podSpec は 'us-east1' 領域に存在するノード上のポッドをスケジュールするよう Kubernetes に指示し 'us-east1-a' または 'us-east1-b' ゾーン内に存在する任意のノードから選択します

次の出力を参照してください。

```

kubect1 get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP              NODE
NOMINATED NODE READINESS GATES
app-pod-1    1/1     Running   0           19s   192.168.25.131 node2
<none>      <none>
kubect1 get pvc -o wide
NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES STORAGECLASS          AGE   VOLUMEMODE
pvc-san      Bound   pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b 300Mi
RWO          netapp-san-us-east1  48s   Filesystem

```

バックエンドを更新して追加 supportedTopologies

既存のバックエンドは 'tridentctl backend update' を使用して 'supportedTopologies' のリストを含むように更新できますこれは、すでにプロビジョニングされているボリュームには影響せず、以降の PVC にのみ使用されます。

詳細については、こちらをご覧ください

- ["コンテナのリソースを管理"](#)
- ["ノードセレクタ"](#)
- ["アフィニティと非アフィニティ"](#)
- ["塗料および耐性"](#)

スナップショットを操作します

永続ボリューム (PV) のKubernetesボリュームSnapshotを使用すると、ボリュームのポイントインタイムコピーを作成できます。Tridentを使用して作成したボリュームのSnapshotの作成、Tridentの外部で作成したSnapshotのインポート、既存のSnapshotからの新しいボリュームの作成、Snapshotからのボリュームデータのリカバリを実行できます。

概要

ボリュームSnapshotは、でサポートされます ontap-nas、ontap-nas-flexgroup、ontap-san、ontap-san-economy、solidfire-san、gcp-cvs`および`azure-netapp-files ドライバ。

作業を開始する前に

スナップショットを操作するには、外部スナップショットコントローラとカスタムリソース定義 (CRD) が必要です。Kubernetesオーケストレーションツール (例: Kubeadm、GKE、OpenShift) の役割を担っています。

KubernetesディストリビューションにスナップショットコントローラとCRDが含まれていない場合は、を参照してください [ボリュームSnapshotコントローラの導入](#)。



GKE環境でオンデマンドボリュームスナップショットを作成する場合は、スナップショットコントローラを作成しないでください。GKEでは、内蔵の非表示のスナップショットコントローラを使用します。

ボリューム **Snapshot** を作成します

手順

1. を作成します `VolumeSnapshotClass`。詳細については、を参照してください "[ボリュームSnapshotクラス](#)"。
 - は `driver`Trident CSIドライバ`を示しています。
 - `deletionPolicy` は、です `Delete` または `Retain`。に設定すると `Retain``を使用すると、ストレージクラスタの基盤となる物理Snapshotが、の場合でも保持されます `VolumeSnapshot オブジェクトが削除された。

例

```
cat snap-sc.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

2. 既存のPVCのスナップショットを作成します。

例

- 次に、既存のPVCのスナップショットを作成する例を示します。

```
cat snap.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

- 次の例は、という名前のPVCのボリュームSnapshotオブジェクトを作成します。 `pvc1 Snapshot`の名前には設定されます `pvc1-snap`。ボリュームSnapshotはPVCに似ており、に関連付けられています `VolumeSnapshotContent` 実際のスナップショットを表すオブジェクト。


```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                AGE
pvc1-snap           50s
```

- 次の情報を確認できます。VolumeSnapshotContent のオブジェクト pvc1-snap ボリュームSnapshot。ボリュームSnapshotの詳細を定義します。。 Snapshot Content Name このSnapshotを提供するVolumeSnapshotContentオブジェクトを特定します。。 Ready To Use パラメータは、スナップショットを使用して新しいPVCを作成できることを示します。

```
kubectl describe volumesnapshots pvc1-snap
Name:                pvc1-snap
Namespace:          default
.
.
.
Spec:
  Snapshot Class Name:  pvc1-snap
  Snapshot Content Name: snapcontent-e8d8a0ca-9826-11e9-9807-525400f3f660
  Source:
    API Group:
    Kind:             PersistentVolumeClaim
    Name:             pvc1
Status:
  Creation Time:      2019-06-26T15:27:29Z
  Ready To Use:      true
  Restore Size:      3Gi
.
.
```

ボリュームSnapshotからPVCを作成

を使用できます dataSource という名前のVolumeSnapshotを使用してPVCを作成するには <pvc-name> データのソースとして。作成された PVC は、ポッドに接続して、他の PVC と同様に使用できます。



PVCはソースボリュームと同じバックエンドに作成されます。を参照してください ["KB : Trident PVCスナップショットからPVCを作成することは代替バックエンドではできない"](#)。

次に、を使用してPVCを作成する例を示します。 pvc1-snap をデータソースとして使用します。

```
cat pvc-from-snap.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

ボリュームSnapshotのインポート

Tridentでは、クラスタ管理者が"[Kubernetesの事前プロビジョニングされたSnapshotプロセス](#)"を使用して、オブジェクトを作成したり、Tridentの外部で作成されたSnapshotをインポートしたりできます
VolumeSnapshotContent。

作業を開始する前に

TridentでSnapshotの親ボリュームが作成またはインポートされている必要があります。

手順

1. *クラスタ管理者：*バックエンドSnapshotを参照するオブジェクトを作成します
VolumeSnapshotContent。これにより、TridentでSnapshotワークフローが開始されます。
 - バックエンドスナップショットの名前を annotations として
trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">。
 - で指定します <name-of-parent-volume-in-trident>/<volume-snapshot-content-name>
snapshotHandle。この情報は、呼び出しで外部スナップショットによってTridentに提供される唯一
の情報です ListSnapshots。



◦ <volumeSnapshotContentName> CRの命名規則のため、バックエンドスナップ
ショット名が常に一致するとは限りません。

例

次の例では、VolumeSnapshotContent バックエンドスナップショットを参照するオブジェクト
snap-01。

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01" # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>

```

2. クラスタ管理者：VolumeSnapshot を参照するCR VolumeSnapshotContent オブジェクト。これにより、VolumeSnapshot 指定された名前空間内。

例

次の例では、VolumeSnapshot CR名 import-snap を参照しています。VolumeSnapshotContent 名前付き import-snap-content。

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content

```

3. *内部処理（アクション不要）：*外部スナップショットは、新しく作成されたを認識して VolumeSnapshotContent`呼び出しを実行します`ListSnapshots。Tridentによってが作成され`TridentSnapshot`ます。
 - 外部スナップショットは、VolumeSnapshotContent 終了：readyToUse および VolumeSnapshot 終了：true。
 - Tridentのリターン readyToUse=true。
4. 任意のユーザー：PersistentVolumeClaim 新しい VolumeSnapshot`を参照してください`spec.dataSource（または spec.dataSourceRef）nameは VolumeSnapshot 名前。

例

次に、を参照するPVCを作成する例を示します。VolumeSnapshot 名前付き import-snap。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Snapshotを使用してボリュームデータをリカバリします

snapshotディレクトリは、を使用してプロビジョニングされるボリュームの互換性を最大限に高めるため、デフォルトでは非表示になっています ontap-nas および ontap-nas-economy ドライバ。を有効にします .snapshot スナップショットからデータを直接リカバリするディレクトリ。

ボリュームを以前のSnapshotに記録されている状態にリストアするには、ボリュームSnapshotリストアONTAP CLIを使用します。

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```



Snapshotコピーをリストアすると、既存のボリューム設定が上書きされます。Snapshotコピーの作成後にボリュームデータに加えた変更は失われます。

Snapshotが関連付けられているPVを削除する

スナップショットが関連付けられている永続ボリュームを削除すると、対応する Trident ボリュームが「削除状態」に更新されます。ボリュームSnapshotを削除してTridentボリュームを削除します。

ボリュームSnapshotコントローラの導入

KubernetesディストリビューションにスナップショットコントローラとCRDが含まれていない場合は、次のように導入できます。

手順

1. ボリュームのSnapshot作成

```
cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. スナップショットコントローラを作成します。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```



必要に応じて、を開きます `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` およびを更新します namespace に移動します。

関連リンク

- ["ボリューム Snapshot"](#)
- ["ボリュームSnapshotクラス"](#)

著作権に関する情報

Copyright © 2025 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。