



バックエンドの構成と管理

Trident

NetApp
March 05, 2026

目次

バックエンドの構成と管理	1
バックエンドを構成する	1
Azure NetApp Files	1
Azure NetApp Filesバックエンドを構成する	1
Azure NetApp Filesバックエンドを構成する準備をする	5
Azure NetApp Filesバックエンド構成オプションと例	8
Google Cloud NetApp Volumes	21
Google Cloud NetApp Volumesバックエンドを構成する	21
Google Cloud NetApp Volumesバックエンドを構成する準備をする	24
Google Cloud NetApp Volumes のバックエンド構成オプションと例	24
Google Cloud バックエンド用のCloud Volumes Serviceを構成する	38
Google Cloud ドライバーの詳細	38
Google Cloud のCloud Volumes Serviceに対するTrident のサポートについて学ぶ	39
バックエンド構成オプション	39
ボリュームのプロビジョニング オプション	41
CVS-Performance サービスタイプの例	41
CVS サービスタイプの例	47
次の手順	49
NetApp HCIまたはSolidFireバックエンドを構成する	50
要素ドライバーの詳細	50
開始する前に	50
バックエンド構成オプション	50
例1: 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ー	51
例2: バックエンドとストレージクラスの構成 `solidfire-san` 仮想プールを備えたドライバー	52
詳細情報の参照	55
ONTAP SAN ドライバー	55
ONTAP SAN ドライバーの概要	55
ONTAP SAN ドライバーを使用してバックエンドを構成する準備をする	57
ONTAP SAN 構成オプションと例	65
ONTAP NAS ドライバー	85
ONTAP NAS ドライバーの概要	85
ONTAP NASドライバーを使用してバックエンドを構成する準備をする	87
ONTAP NAS の構成オプションと例	99
Amazon FSx for NetApp ONTAP	121
Amazon FSx for NetApp ONTAPでTrident を使用する	121
IAMロールとAWSシークレットを作成する	124
Tridentをインストールする	130
ストレージバックエンドを構成する	138
ストレージクラスとPVCを構成する	148

サンプルアプリケーションをデプロイする	153
EKS クラスターでTrident EKS アドオンを構成する	154
kubectl でバックエンドを作成する	157
TridentBackendConfig	157
手順の概要	159
ステップ1: Kubernetesシークレットを作成する	159
ステップ2: 作成する `TridentBackendConfig` CR	161
ステップ3: ステータスを確認する `TridentBackendConfig` CR	161
(オプション) ステップ4: 詳細を取得する	162
バックエンドを管理する	164
kubectl を使用してバックエンド管理を実行する	164
tridentctl でバックエンド管理を実行する	165
バックエンド管理オプション間を移動する	166

バックエンドの構成と管理

バックエンドを構成する

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。これは、Tridentにそのストレージシステムと通信する方法と、そこからボリュームをプロビジョニングする方法を指示します。

Tridentは、ストレージクラスによって定義された要件に一致するバックエンドからストレージプールを自動的に提供します。ストレージシステムのバックエンドを構成する方法を学習します。

- ["Azure NetApp Filesバックエンドを構成する"](#)
- ["Google Cloud NetApp Volumesバックエンドを構成する"](#)
- ["Google Cloud Platform バックエンド用のCloud Volumes Serviceを構成する"](#)
- ["NetApp HCIまたはSolidFireバックエンドを構成する"](#)
- ["ONTAPまたはCloud Volumes ONTAP NAS ドライバーを使用してバックエンドを構成する"](#)
- ["ONTAPまたはCloud Volumes ONTAP SAN ドライバーを使用してバックエンドを構成する"](#)
- ["Amazon FSx for NetApp ONTAPでTridentを使用する"](#)

Azure NetApp Files

Azure NetApp Filesバックエンドを構成する

Azure NetApp Files をTridentのバックエンドとして構成できます。Azure NetApp Files バックエンドを使用して、NFS および SMB ボリュームを接続できます。Tridentは、Azure Kubernetes Services (AKS) クラスターのマネージド ID を使用した資格情報の管理もサポートしています。

Azure NetApp Filesドライバーの詳細

Tridentは、クラスターと通信するために次のAzure NetApp Filesストレージドライバーを提供します。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	サポートされているファイルシステム
azure-netapp-files	NFS SMB	Filesystem	RWO、ROX、RWX、RWOP	nfs、smb

考慮事項

- Azure NetApp Filesサービスは、50 GiB 未満のボリュームをサポートしていません。より小さいボリュームが要求された場合、Tridentは自動的に50 GiBのボリュームを作成します。

- Trident 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 ボリュームのみをサポートします。

AKS のマネージド ID

Tridentのサポート"マネージドID" Azure Kubernetes Services クラスター用。マネージド ID によって提供される合理化された資格情報管理を活用するには、次のものがが必要です。

- AKS を使用してデプロイされた Kubernetes クラスター
- AKS Kubernetes クラスターで構成されたマネージド ID
- Tridentがインストールされており、cloudProvider`指定する ` "Azure"。

Tridentオペレーター

Tridentオペレータを使用してTridentをインストールするには、tridentorchestrator_cr.yaml`設定する `cloudProvider`に ` "Azure"。例えば：

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

舵

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

```
helm install trident trident-operator-100.2506.0.tgz --create
--namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code>トライデントctl</code>

次の例では、Tridentをインストールし、cloudProvider`フラグを `Azure:

```
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 資格情報を提供する代わりに、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るようになります。

Azure でクラウド ID を活用するには、次のものがが必要です。

- AKS を使用してデプロイされた Kubernetes クラスター
- AKS Kubernetes クラスターで構成されたワークロード ID と oidc-issuer
- Tridentがインストールされており、`cloudProvider`指定する`"Azure"`そして`cloudIdentity`ワークロードIDの指定

Tridentオペレーター

Tridentオペレーターを使用してTridentをインストールするには、`tridentorchestrator_cr.yaml`に設定する `cloudProvider` に `"Azure"`、そして設定 `cloudIdentity` に `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`。

例えば：

```
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-xxxxxxxxxxxx' # Edit
```

舵

次の環境変数を使用して、**cloud-provider (CP)** および **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-xxxxxxxxxxxx'"
```

次の例では、Tridentをインストールし、`cloudProvider` 環境変数を使用してAzureへ `CP`、そして、`cloudIdentity` 環境変数を使用する `CI`：

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>トライデントctl</code>

次の環境変数を使用して、クラウド プロバイダー および クラウド ID 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-xxxxxxxxxxxx"
```

次の例では、Tridentをインストールし、`cloud-provider` フラグを `CP`、そして `cloud-identity` に `CI`：

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

Azure NetApp Filesバックエンドを構成する準備をする

Azure NetApp Filesバックエンドを構成する前に、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およびSMBボリュームの前提条件

Azure NetApp Files を初めて使用するか、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、いくつかの初期構成が必要です。参照 ["Azure: Azure NetApp Files をセットアップして NFS ボリュームを作成する"](#)。

設定して使用するには ["Azure NetApp Files"](#)バックエンドでは、次のものがが必要です。



- subscriptionID、tenantID、clientID、location、そして `clientSecret` AKS クラスターでマネージド ID を使用する場合はオプションです。
- tenantID、clientID、そして `clientSecret` AKS クラスターでクラウド ID を使用する場合はオプションです。

- 容量プール。参照["Microsoft: Azure NetApp Filesの容量プールを作成する"](#)。
- Azure NetApp Filesに委任されたサブネット。参照["Microsoft: Azure NetApp Filesにサブネットを委任する"](#)。
- `subscriptionID` Azure NetApp Filesが有効になっている Azure サブスクリプションから。
- tenantID、clientID、そして `clientSecret` から["アプリ登録"](#) Azure Active Directory で、Azure NetApp Filesサービスに対する十分な権限を持っていること。アプリ登録では次のいずれかを使用する必要があります。
 - 所有者または貢献者の役割["Azureによって事前定義済み"](#)。
 - あ["カスタム貢献者ロール"](#)サブスクリプションレベルで(assignableScopes) に、Tridentに必要な権限のみに制限された以下の権限が付与されます。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る"](#)。

```

{
  "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/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat

```

```

ions/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": []
}
]
}
}
}

```

- アズール `location` 少なくとも1つを含む **"委任されたサブネット"**。Trident22.01の時点で、`location` パラメータは、バックエンド構成ファイルの最上位レベルの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。
- 使用するには Cloud Identity、取得 `client ID` から **"ユーザー割り当てマネージド ID"** そしてそのIDを `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`。

SMBボリュームの追加要件

SMB ボリュームを作成するには、次のものがが必要です。

- Active Directory が構成され、Azure NetApp Filesに接続されています。参照["Microsoft: Azure NetApp Filesの Active Directory 接続の作成と管理"](#)。
- Linux コントローラー ノードと、Windows Server 2022 を実行する少なくとも 1 つの Windows ワーカー ノードを備えた Kubernetes クラスター。Trident 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 ボリュームのみをサポートします。
- Azure NetApp Files が Active Directory に対して認証できるように、Active Directory 資格情報を含む少なくとも 1 つのTridentシークレット。秘密を生成する `smbcreds`:

```

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

```

- Windows サービスとして構成された CSI プロキシ。設定するには `csi-proxy`、参照["GitHub: CSIプロキシ"](#)または["GitHub: Windows 用 CSI プロキシ"](#)Windows 上で実行されている Kubernetes ノード用。

Azure NetApp Filesバックエンド構成オプションと例

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学習し、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で、要求されたサービス レベルとサブネットに一致する容量プールにAzure NetApp Files ボリュームを作成します。



* NetApp Trident 25.06 リリース以降、手動 QoS 容量プールがテクニカル プレビューとしてサポートされます。*

Azure NetApp Filesバックエンドは、次の構成オプションを提供します。

パラメータ	説明	デフォルト
version		常に1
storageDriverName	ストレージ ドライバーの名前	「azure-netapp-files」
backendName	カスタム名またはストレージバックエンド	ドライバー名 + "_" + ランダムな文字
subscriptionID	Azure サブスクリプションのサブスクリプション ID。AKS クラスタでマネージド ID が有効になっている場合はオプションです。	
tenantID	アプリ登録からのテナント ID 。AKS クラスタでマネージド ID またはクラウド ID が使用される場合はオプションです。	
clientID	アプリ登録からのクライアント ID。AKS クラスタでマネージド ID またはクラウド ID が使用される場合はオプションです。	
clientSecret	アプリ登録からのクライアント シークレット。AKS クラスタでマネージド ID またはクラウド ID が使用される場合はオプションです。	
serviceLevel	1つ Standard、 Premium、 または Ultra	「」 (ランダム)
location	新しいボリュームが作成される Azure の場所の名前。AKS クラスタでマネージド ID が有効になっている場合はオプションです。	

パラメータ	説明	デフォルト
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	[] (フィルターなし)
netappAccounts	検出されたリソースをフィルタリングするためのNetAppアカウントのリスト	[] (フィルターなし)
capacityPools	検出されたリソースをフィルタリングするための容量プールのリスト	[] (フィルターなし、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	""
subnet	委任先のサブネットの名前 Microsoft.Netapp/volumes	""
networkFeatures	ボリュームのVNet機能のセット。Basic`または `Standard。ネットワーク機能はすべての地域で利用できるわけではなく、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。指定`networkFeatures`この機能が有効になっていないと、ボリュームのプロビジョニングが失敗します。	""
nfsMountOptions	NFS マウント オプションのきめ細かな制御。SMB ボリュームの場合は無視されます。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、`nfsvers=4`カンマ区切りのマウント オプション リストで、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 オプションをオーバーライドします。	「nfsvers=3」
limitVolumeSize	要求されたボリューム サイズがこの値を超える場合、プロビジョニングに失敗します	"" (デフォルトでは強制されません)
debugTraceFlags	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, "discovery": true}。 トラブルシューティングを行っており、詳細なログ ダンプが必要な場合を除き、これを使用しないでください。	ヌル

パラメータ	説明	デフォルト
nasType	NFS または SMB ボリュームの作成を構成します。オプションは nfs、`smb` または null。null に設定すると、デフォルトで NFS ボリュームになります。	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、" CSI トポロジを使用する "。	
qosType	QoS タイプ（自動または手動）を表します。* Trident 25.06 のテクニカルプレビュー*	オート
maxThroughput	許容される最大スループットを MiB/秒単位で設定します。手動 QoS 容量プールでのみサポートされます。* Trident 25.06 のテクニカルプレビュー*	4 MiB/sec



ネットワーク機能の詳細については、以下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構成する](#)"。

必要な権限とリソース

PVC の作成時に「容量プールが見つかりません」というエラーが表示される場合は、アプリの登録に必要なアクセス許可とリソース (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。デバッグが有効になっている場合、Trident はバックエンドの作成時に検出された Azure リソースをログに記録します。適切なロールが使用されていることを確認します。

の値は resourceGroups、netappAccounts、capacityPools、virtualNetwork、そして `subnet` 短い名前または完全修飾名を使用して指定できます。短い名前は同じ名前を持つ複数のリソースと一致する可能性があるため、ほとんどの場合、完全修飾名が推奨されます。

その resourceGroups、netappAccounts、そして `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ものに制限するフィルターであり、任意の組み合わせで指定できます。完全修飾名は、次の形式に従います。

タイプ	フォーマット
リソース グループ	<リソース グループ>
NetApp アカウント	<リソース グループ>/<netapp アカウント>
容量プール	<リソース グループ>/<netapp 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	<リソース グループ>/<仮想ネットワーク>/<サブネット>

ボリュームプロビジョニング

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により、デフォルトのボリュームのプロビジョニングを制御できます。参照 [\[構成例\]](#) 詳細については。

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。 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コンマで区切ったリストにする必要があります。 SMB ボリュームの場合は無視されます。	「0.0.0.0/0」
snapshotDir	.snapshotディレクトリの可視性を制御します	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 (4桁の 8 進数)。SMB ボリュームの場合は無視されます。	「」 (プレビュー機能、サブスクリプションでホワイトリストへの登録が必要)

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構成を示しています。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

これは絶対的に最小限のバックエンド構成です。この構成では、Trident は構成された場所にある Azure NetApp Files に委任されたすべての NetApp アカウント、容量プール、およびサブネットを検出し、それらのプールとサブネット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。なぜなら `nasType` 省略された場合、`nfs` デフォルトが適用され、バックエンドは NFS ボリュームをプロビジョニングします。

この構成は、Azure NetApp Files を使い始めて試してみる場合に最適ですが、実際には、プロビジョニングするボリュームの範囲をさらに指定することが必要になります。

```
---
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 のマネージド ID

このバックエンド構成では、subscriptionID、tenantID、clientID、そして`clientSecret`マネージド 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
```

AKS のクラウド ID

このバックエンド構成では、tenantID、clientID、そして`clientSecret`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ー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、Azureの`eastus`場所`Ultra`容量プール。Tridentは、その場所でAzure NetApp Filesに委任されたすべてのサブネットを自動的に検出し、そのうちの1つ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
```

手動 QoS 容量プールを使用したバックエンドの例

このバックエンド構成では、Azureの `eastus` 手動 QoS 容量プールのある場所。* NetApp Trident 25.06 のテクニカル プレビュー*。

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

高度な設定

このバックエンド構成により、ボリュームの配置範囲が単一のサブネットにさらに縮小され、一部のボリューム プロビジョニングのデフォルトも変更されます。

```
---
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"
```

仮想プールの構成

このバックエンド構成では、単一のファイルで複数のストレージ 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。仮想プールラベルは、以下の基準に基づいてプールを区別するために使用されました。 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
```

サポートされているトポロジ構成

Trident は、リージョンと可用性ゾーンに基づいてワークロードのボリュームのプロビジョニングを容易にします。その `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する必要があります。これらのリージョンとゾーンは、ストレージ クラスで提供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、"[CSIトポロジを使用する](#)"。

```
---
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 `StorageClass` 定義は上記のストレージ プールを参照します。

定義例 `parameter.selector` 分野

使用 `parameter.selector` それぞれ指定できます `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側面が含まれます。

```
---
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

使用 `nasType`、`node-stage-secret-name`、そして `node-stage-secret-namespace`、SMB ボリュームを指定し、必要な Active Directory 資格情報を提供できます。

デフォルトの名前空間での基本設定

```
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"
```

名前空間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秘密を使用する

```
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のフィルター。`nasType: nfs`または`nasType: null`NFS プールのフィルター。

バックエンドを作成する

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

```
tridentctl logs
```

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

Google Cloud NetApp Volumes

Google Cloud NetApp Volumesバックエンドを構成する

Google Cloud NetApp Volumes をTridentのバックエンドとして構成できるようになりました。Google Cloud NetApp Volumes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。

Google Cloud NetApp Volumesドライバの詳細

Tridentは`google-cloud-netapp-volumes`クラスターと通信するためのドライバー。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	サポートされているファイルシステム
google-cloud-netapp-volumes	NFS SMB	Filesystem	RWO、ROX、RWX、RWOP	nfs、smb

GKE のクラウド ID

Cloud Identity を使用すると、Kubernetes ポッドは、明示的な Google Cloud 認証情報を提供する代わりに、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るようになります。

Google Cloud でクラウド ID を活用するには、次のものがが必要です。

- GKE を使用してデプロイされた Kubernetes クラスター。
- GKE クラスターに構成されたワークロード ID と、ノードプールに構成された GKE メタデータ サーバー。

- Google Cloud NetApp Volumes管理者 (roles/netapp.admin) ロールまたはカスタムロールを持つ GCP サービス アカウント。
- 「GCP」を指定する cloudProvider と、新しい GCP サービス アカウントを指定する cloudIdentity を含むTrident がインストールされています。以下に例を示します。

Tridentオペレーター

Tridentオペレーターを使用してTridentをインストールするには、`tridentorchestrator_cr.yaml`を設定する `cloudProvider` に `"GCP"`、そして設定 `cloudIdentity` に `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com`。

例えば：

```
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'
```

舵

次の環境変数を使用して、**cloud-provider (CP)** および **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、Tridentをインストールし、`cloudProvider` 環境変数を使用してGCPへ `$CP`、そして、`cloudIdentity` 環境変数を使用する `$ANNOTATION`：

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

<code>トライデントctl</code>

次の環境変数を使用して、クラウド プロバイダー および クラウド ID 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、`cloud-provider` フラグを `$CP`、そして `cloud-identity` に `$ANNOTATION`：

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

Google Cloud NetApp Volumesバックエンドを構成する準備をする

Google Cloud NetApp Volumesバックエンドを構成する前に、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ボリュームの前提条件

Google Cloud NetApp Volumes を初めて使用するか、新しい場所で使用する場合は、Google Cloud NetApp Volumes をセットアップして NFS ボリュームを作成するための初期設定が必要です。参照["開始する前に"](#)。

Google Cloud NetApp Volumesバックエンドを構成する前に、次のものを用意してください。

- Google Cloud NetApp Volumesサービスで構成された Google Cloud アカウント。参照["Google Cloud NetApp Volumes"](#)。
- Google Cloud アカウントのプロジェクト番号。参照["プロジェクトの特定"](#)。
- NetApp Volumes 管理者権限を持つ Google Cloud サービス アカウント(roles/netapp.admin) 役割。参照["アイデンティティと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。参照["サービスアカウントキーを作成する"](#)
- ストレージ プール。参照["ストレージプールの概要"](#)。

Google Cloud NetApp Volumesへのアクセスを設定する方法の詳細については、以下を参照してください。"[Google Cloud NetApp Volumesへのアクセスを設定する](#)"。

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

Google Cloud NetApp Volumesのバックエンド構成オプションについて学習し、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、単一の 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には、追加のバックエンドを定義できます。

パラメータ	説明	デフォルト
version		常に1
storageDriverName	ストレージ ドライバーの名前	の価値 storageDriverName 「google-cloud-netapp-volumes」 として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	ドライバー名 + "_" + API キーの一部
storagePools	ボリューム作成用のストレージ プールを指定するために使用されるオプションのパラメーター。	
projectNumber	Google Cloud アカウントのプロジェクト番号。値は Google Cloud ポータルのホームページにあります。	
location	Trident が GCNV ボリュームを作成する Google Cloud のロケーション。クロスリージョン Kubernetes クラスタを作成する場合、`location` 複数の Google Cloud リージョンにまたがるノードでスケジュールされたワークロードで使用できます。リージョン間のトラフィックには追加コストが発生します。	
apiKey	Google Cloud サービスアカウントの API キーと netapp.admin`役割。これには、Google Cloud サービス アカウントの秘密鍵ファイルの JSON 形式の内容 (バックエンド構成ファイルにそのままコピーされます) が含まれます。その `apiKey` 次の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`、そして `client_x509_cert_url`。	
nfsMountOptions	NFS マウント オプションのきめ細かな制御。	「nfsvers=3」
limitVolumeSize	要求されたボリューム サイズがこの値を超える場合、プロビジョニングは失敗します。	"" (デフォルトでは強制されません)
serviceLevel	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。	
labels	ボリュームに適用する任意の JSON 形式のラベルのセット	""
network	GCNV ボリュームに使用される Google Cloud ネットワーク。	
debugTraceFlags	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おり、詳細なログ ダンプが必要な場合を除き、これを使用しないでください。	ヌル
nasType	NFS または SMB ボリュームの作成を構成します。オプションは nfs、`smb` または null。null に設定すると、デフォルトで NFS ボリュームになります。	nfs

パラメータ	説明	デフォルト
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、" CSIトポロジを使用する "。例えば： 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

デフォルトのボリュームプロビジョニングは、`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。任意の IPv4 アドレスの組み合わせをコンマで区切ったリストにする必要があります。	「0.0.0.0/0」
snapshotDir	アクセス`.snapshot`ディレクトリ	NFSv4の場合は「true」、NFSv3の場合は「false」
snapshotReserve	スナップショット用に予約されているボリュームの割合	"" (デフォルトの0を受け入れます)
unixPermissions	新しいボリュームの UNIX 権限 (4桁の 8 進数)。	""

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構成を示しています。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

これは絶対的に最小限のバックエンド構成です。この構成では、Trident は構成された場所にあるGoogle Cloud NetApp Volumesに委任されたすべてのストレージ プールを検出し、それらのプールの1つに新しいボリュームをランダムに配置します。なぜなら `nasType` 省略された場合、`nfs` デフォルトが適用され、バックエンドは NFS ボリュームをプロビジョニングします。

この構成は、Google Cloud NetApp Volumesを使い始めて試してみる場合に最適ですが、実際には、プロビジョニングするボリュームのスコープをさらに指定する必要がある可能性が高くなります。

```

---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    XsYg6gyxy4zq7OlwWgLwGa==\n
    -----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

```

SMBボリュームの構成

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    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/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

StoragePoolsフィルターを使用した構成



```

---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/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
  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

```

仮想プールの構成

このバックエンド構成では、単一のファイルで複数の仮想プールを定義します。仮想プールは、`storage`セクション。異なるサービス レベルをサポートする複数のストレージ プールがあり、Kubernetes でそれらを表すストレージ クラスを作成する場合に役立ちます。仮想プール ラベルは、プールを区別するために使用されます。例えば、以下の例では `performance`ラベルと `serviceLevel`タイプは仮想プールを区別するために使用されます。

一部のデフォルト値をすべての仮想プールに適用できるように設定し、個々の仮想プールのデフォルト値を上書きすることもできます。次の例では、`snapshotReserve`そして `exportRule`すべての仮想プールのデフォルトとして機能します。

詳細については、"[仮想プール](#)"。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/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
```

サポートされているトポロジ構成

Trident は、リージョンと可用性ゾーンに基づいてワークロードのボリュームのプロビジョニングを容易にします。その `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する必要があります。これらのリージョンとゾーンは、ストレージ クラスで提供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、"[CSIトポロジを使用する](#)"。

```
---
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、バックエンドの構成に問題があります。バックエンドを記述するには、`kubectl get tridentbackendconfig <backend-name>` 次のコマンドを実行して、コマンドを実行するか、ログを表示して原因を特定します。

```
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。

ストレージクラスの定義

以下は基本的な `StorageClass` 上記のバックエンドを参照する定義。

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

定義例 `parameter.selector` 分野：

使用 `parameter.selector` それぞれ指定できます `StorageClass` その"[仮想プール](#)"ボリュームをホストするために使用されます。ボリュームには、選択したプールで定義された側面が含まれます。

```
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、以下を参照してください。"[ストレージクラスを作成する](#)"。

SMBボリュームの定義例

使用 `nasType`、`node-stage-secret-name`、そして `node-stage-secret-namespace`、SMB ボリュームを指定し、必要な Active Directory 資格情報を提供できます。任意の権限または権限のない任意の Active Directory ユーザー/パスワードをノード ステージ シークレットに使用できます。

デフォルトの名前空間での基本設定

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

名前空間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: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  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秘密を使用する

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  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のフィルター。`nasType: nfs`または`nasType: null`NFS プールのフィルター。

PVC定義の例

```
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 バックエンド用のCloud Volumes Serviceを構成する

提供されているサンプル構成を使用して、TridentインストールのバックエンドとしてNetApp Cloud Volumes Service for Google Cloud を構成する方法を学習します。

Google Cloud ドライバーの詳細

Tridentは`gcp-cvs`クラスターと通信するためのドライバー。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	サポートされているファイルシステム
gcp-cvs	NFS	Filesystem	RWO、ROX、RWX、RWOP	nfs

Google Cloud のCloud Volumes Serviceに対するTrident のサポートについて学ぶ

Tridentは、2つのうちのいずれかでCloud Volumes Serviceボリュームを作成できます。"サービスの種類":

- **CVS-Performance:** デフォルトのTridentサービス タイプ。このパフォーマンス最適化されたサービス タイプは、パフォーマンスを重視する運用ワークロードに最適です。CVS-Performance サービス タイプは、最小 100 GiB サイズ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。次のいずれかを選択できます"[3つのサービスレベル](#)":
 - standard
 - premium
 - extreme
- **CVS:** CVS サービス タイプは、限定的から中程度のパフォーマンス レベルで、高いゾーン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 1 GiB という小さな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がプールの容量とパフォーマンスを共有します。次のいずれかを選択できます"[2つのサービスレベル](#)":
 - standardsw
 - zoneredundantstandardsw

要件

設定して使用するには "[Cloud Volumes Service for Google Cloud](#)"バックエンドでは、次のものがが必要です。

- NetApp Cloud Volumes Serviceが設定された Google Cloud アカウント
- Google Cloud アカウントのプロジェクト番号
- Google Cloudサービスアカウント `netappcloudvolumes.admin` 役割
- 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、単一の 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には、追加のバックエンドを定義できます。

パラメータ	説明	デフォルト
version		常に1
storageDriverName	ストレージ ドライバーの名前	「gcp-cvs」
backendName	カスタム名またはストレージバックエンド	ドライバー名 + "_" + API キーの一部
storageClass	CVS サービス タイプを指定するために使用されるオプション パラメーター。使用 <code>software`CVS`</code> サービス タイプを選択します。それ以外の場合、TridentはCVS-Performanceサービスタイプを想定します。(`hardware`)。	

パラメータ	説明	デフォルト
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 プールを指定するために使用されるオプションのパラメーター。	
projectNumber	Google Cloud アカウントのプロジェクト番号。値は Google Cloud ポータルの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` 奉仕プロジェクトであり、`hostProjectNumber` ホストプロジェクトです。	
apiRegion	Trident が Cloud Volumes Service ボリュームを作成する Google Cloud リージョン。リージョンをまたがる Kubernetes クラスタを作成する場合、`apiRegion` 複数の Google Cloud リージョンにまたがるノードでスケジュールされたワークロードで使用できます。リージョン間のトラフィックには追加コストが発生します。	
apiKey	Google Cloud サービスアカウントの API キーと `netappcloudvolumes.admin` 役割。これには、Google Cloud サービス アカウントの秘密鍵ファイルの JSON 形式の内容（バックエンド構成ファイルにそのままコピーされます）が含まれます。	
proxyURL	CVS アカウントに接続するためにプロキシ サーバーが必要な場合のプロキシ URL。プロキシ サーバーは、HTTP プロキシまたは HTTPS プロキシのいずれかになります。HTTPS プロキシの場合、プロキシサーバーで自己署名証明書を使用できるようにするために、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ーはサポートされていません。	
nfsMountOptions	NFS マウント オプションのきめ細かな制御。	「nfsvers=3」
limitVolumeSize	要求されたボリューム サイズがこの値を超える場合、プロビジョニングは失敗します。	"" (デフォルトでは強制されません)
serviceLevel	新しいボリュームの CVS-Performance または CVS サービス レベル。CVS パフォーマンス値は standard、premium、または extreme。CVS 値は standardsw` または `zoneredundantstandardsw。	CVS-Performance のデフォルトは「標準」です。CVS のデフォルトは「standardsw」です。
network	Cloud Volumes Service ボリュームに使用される Google Cloud ネットワーク。	"デフォルト"
debugTraceFlags	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おり、詳細なログ ダンプが必要な場合を除き、これを使用しないでください。	ヌル

パラメータ	説明	デフォルト
allowedTopologies	リージョン間のアクセスを有効にするには、allowedTopologies`すべての地域を含める必要があります。例えば： `- 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

デフォルトのボリュームプロビジョニングは、`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コンマで区切ったリストにする必要があります。	「0.0.0.0/0」
snapshotDir	アクセス`.snapshot`ディレクトリ	"間違い"
snapshotReserve	スナップショット用に予約されているボリュームの割合	"" (CVS のデフォルトの 0 を受け入れます)
size	新しいボリュームのサイズ。CVS パフォーマンスの最小値は 100 GiB です。CVS の最小値は 1 GiB です。	CVS-Performance サービス タイプのデフォルトは「100GiB」です。CVS サービス タイプではデフォルトは設定されませんが、最小 1 GiB が必要です。

CVS-Performance サービスタイプの例

次の例は、CVS-Performance サービス タイプのサンプル構成を示しています。

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-Performance サービス タイプを使用する最小の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: サービスレベル構成

このサンプルは、サービス レベルやボリュームのデフォルトなどの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

このサンプルでは `storage` 仮想プールを構成し、`StorageClasses` それらを参照します。参照[\[ストレージクラスの定義\]](#)ストレージ クラスがどのように定義されているかを確認します。

ここでは、すべての仮想プールに特定のデフォルトが設定され、`snapshotReserve` 5%で `exportRule` `0.0.0.0/0` に変更します。仮想プールは、`storage` セクション。各仮想プールは独自の `serviceLevel` 一部のプールではデフォルト値が上書きされます。仮想プールラベルは、以下の基準に基づいてプールを区別するために使用されました。`performance` そして `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。使用 parameters.selector、ボリュームをホストするために使用される仮想プールを StorageClass 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側面が含まれます。

```

---
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StorageClass(cvs-extreme-extra-protection) は最初の仮想プールにマップされます。これは、スナップショット予約が 10% で極めて優れた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予約を提供するストレージプールを呼び出します。Trident は、どの仮想プールが選択されるか決定し、スナップショット予約要件が満たされていることを確認します。

CVS サービスタイプの例

次の例は、CVS サービス タイプのサンプル構成を示しています。

例1: 最小構成

これは、storageClass CVSサービスタイプとデフォルトを指定する `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

このサンプルのバックエンド構成では、`storagePools`ストレージ プールを構成します。

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

```
tridentctl logs
```

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

NetApp HCIまたはSolidFireバックエンドを構成する

Tridentインストールで Element バックエンドを作成して使用方法を学習します。

要素ドライバーの詳細

Tridentは `solidfire-san` クラスターと通信するためのストレージ ドライバー。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。

その `solidfire-san` ストレージ ドライバーは、ファイル および ブロック ボリューム モードをサポートします。のために `Filesystem` volumeMode では、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種類は、StorageClass によって指定されます。

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	サポートされているファイルシステム
solidfire-san	iSCSI	ブロック	RWO、ROX、RWX、RWOP	ファイルシステムがありません。生のブロックデバイス。
solidfire-san	iSCSI	Filesystem	RWO、RWOP	xfs、ext3、ext4

開始する前に

Element バックエンドを作成する前に、次のものがが必要です。

- Element ソフトウェアを実行するサポートされているストレージ システム。
- ボリュームを管理できるNetApp HCI/ SolidFireクラスター管理者またはテナント ユーザーの資格情報。
- すべての Kubernetes ワーカーノードに適切な iSCSI ツールがインストールされている必要があります。参照["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

バックエンドの構成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
version		常に1
storageDriverName	ストレージ ドライバーの名前	いつも「solidfireさん」

パラメータ	説明	デフォルト
backendName	カスタム名またはストレージバックエンド	「solidfire_」 + ストレージ (iSCSI) IPアドレス
Endpoint	テナント資格情報を持つSolidFire クラスターの MVIP	
SVIP	ストレージ (iSCSI) IPアドレスとポート	
labels	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 (見つからない場合は作成)	
InitiatorIFace	iSCSIトラフィックを特定のホスト インターフェースに制限する	"デフォルト"
UseCHAP	CHAP を使用して iSCSI を認証します。 Trident はCHAP を使用しません。	true
AccessGroups	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
Types	QoS仕様	
limitVolumeSize	要求されたボリューム サイズがこの値を超える場合、プロビジョニングに失敗します	"" (デフォルトでは強制されません)
debugTraceFlags	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>	ヌル



使用しないでください `debugTraceFlags` ただし、トラブルシューティングを行っており、詳細なログ ダンプが必要な場合を除きます。

例1: 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ー

この例では、CHAP 認証を使用し、特定の QoS 保証を備えた 3 つのボリューム タイプをモデル化するバックエンド ファイルを示します。おそらく、これらのそれぞれを消費するためのストレージクラスを定義することになるでしょう。`IOPS` ストレージ クラス パラメータ。

```

---
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: バックエンドとストレージクラスの構成 `solidfire-san` 仮想プールを備えたドライバー

この例では、仮想プールと、それらを参照する StorageClasses が構成されたバックエンド定義ファイルを示します。

Trident は、プロビジョニング時にストレージ プールにあるラベルをバックエンド ストレージ LUN にコピーします。便宜上、ストレージ管理者は仮想プールごとにラベルを定義し、ラベルごとにボリュームをグループ化できます。

以下に示すサンプルのバックエンド定義ファイルでは、すべてのストレージプールに特定のデフォルトが設定されており、`type` シルバーで。仮想プールは、`storage` セクション。この例では、一部のストレージ プールは独自のタイプを設定し、一部のプールは上記で設定された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ます。使用して `parameters.selector` フィールドでは、各 StorageClass はボリュームをホストするため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で定義された側面が設定されます。

最初のStorageClass(solidfire-gold-four) は最初の仮想プールにマップされます。これはゴールドパフォーマンスを提供する唯一のプールです Volume Type QoS`ゴールドの。最後のStorageClass(`solidfire-silver) は、シルバー 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 はどの仮想プールが選択されるか決定し、ストレージ要件が満たされていることを確認します。

```
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

```

詳細情報の参照

- ["ボリュームアクセスグループ"](#)

ONTAP SAN ドライバー

ONTAP SAN ドライバーの概要

ONTAPおよびCloud Volumes ONTAP SAN ドライバーを使用してONTAPバックエンドを構成する方法について学習します。

ONTAP SAN ドライバーの詳細

Trident は、ONTAPクラスタと通信するための次の SAN ストレージ ドライバーを提供します。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	サポートされているファイルシステム
ontap-san	iSCSI SCSI over 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
ontap-san	iSCSI SCSI over FC	Filesystem	RWO、RWOP ROX と 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。	xfx、ext3、ext4
ontap-san	NVMe/TCP 参照NVMe/TCPに関する追加の考慮事項。	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

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	サポートされているファイルシステム
ontap-san	NVMe/TCP 参照NVMe/TC Pに関する追加の考慮事項。	Filesystem	RWO、RWOP ROX と 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-economy	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
ontap-san-economy	iSCSI	Filesystem	RWO、RWOP ROX と 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` 永続ボリュームの使用数が"[サポートされているONTAPボリュームの制限](#)"。
- 使用 `ontap-nas-economy` 永続ボリュームの使用数が"[サポートされているONTAPボリュームの制限](#)"そして `ontap-san-economy` ドライバーは使用できません。
- 使用しないでください `ontap-nas-economy` データ保護、災害復旧、モビリティの必要性が予想される場合。
- NetApp、ontap-san を除くすべてのONTAPドライバーで Flexvol autogrow を使用することは推奨されていません。回避策として、Trident はスナップショット リザーブの使用をサポートし、それに応じて Flexvol ボリュームを拡張します。

ユーザー権限

Tridentは、通常、ONTAPまたはSVM管理者として実行することを想定しています。`admin` クラスターユーザーまたは `vsadmin` SVM ユーザー、または同じロールを持つ別の名前のユーザー。Amazon FSx for NetApp ONTAPの導入では、Tridentはクラスターを使用してONTAPまたはSVM管理者として実行されることが想定されています。`fsxadmin` ユーザーまたは `vsadmin` SVM ユーザー、または同じロールを持つ別の名前のユーザー。その `fsxadmin` ユーザーは、クラスター管理者ユーザーの限定的な代替です。



を使用する場合 `limitAggregateUsage` パラメータには、クラスター管理者の権限が必要です。Amazon FSx for NetApp ONTAPをTridentで使用する場合、`limitAggregateUsage` パラメータは、`vsadmin` そして `fsxadmin` ユーザーアカウント。このパラメータを指定すると、構成操作は失敗します。

ONTAP内でTridentドライバーが使用できる、より制限の厳しいロールを作成することは可能ですが、お勧めしません。Tridentのほとんどの新しいリリースでは、考慮する必要がある追加のAPI が呼び出されるため、アップグレードが困難になり、エラーが発生しやすくなります。

NVMe/TCPに関する追加の考慮事項

Tridentは、不揮発性メモリエクスプレス (NVMe) プロトコルをサポートしています。`ontap-san`ドライバーには以下が含まれます:

- IPv6
- NVMeボリュームのスナップショットとクローン
- 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する
- NVMeネイティブマルチパス
- K8sノードの正常または異常シャットダウン (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ているDH-HMAC-CHAP
- デバイスマッパー (DM) マルチパス
- LUKS暗号化



NVMe はONTAP REST API でのみサポートされ、ONTAPI (ZAPI) ではサポートされません。

ONTAP SAN ドライバーを使用してバックエンドを構成する準備をする

ONTAP SAN ドライバーを使用して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バックエンドでは、Trident少なくとも1つのアグリゲートをSVMに割り当てる必要があります。



"[ASA r2 システム](#)"ストレージ層の実装は他のONTAPシステム (ASA、AFF、FAS) とは異なります。ASA r2 システムでは、集約の代わりにストレージ可用性ゾーンが使用されます。参照"[事項を](#)"ASA r2 システムでSVMにアグリゲートを割り当てる方法に関するナレッジベースの記事。

複数のドライバーを実行し、いずれかを指すストレージクラスを作成することもできることに注意してください。例えば、`san-dev`を使用するクラス`ontap-san`運転手と`san-default`を使用するクラス`ontap-san-economy`1つ。

すべてのKubernetes ワーカーノードに適切なiSCSI ツールがインストールされている必要があります。参照"[ワーカーノードを準備する](#)" 詳細については。

ONTAPバックエンドを認証する

Trident は、ONTAPバックエンドを認証する2つのモードを提供します。

- 資格情報ベース: 必要な権限を持つONTAPユーザーのユーザー名とパスワード。次のような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お勧めします。`admin`または`vsadmin`ONTAPバージョンとの最大限の互換性を確保するためです。
- 証明書ベース: Trident は、バックエンドにインストールされた証明書を使用してONTAPクラスターと通信することもできます。ここで、バックエンド定義には、クライアント証明書、キー、および信頼されたCA証明書（使用する場合、推奨）のBase64 エンコードされた値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、資格情報ベースの方法と証明書ベースの方法間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。別の認証方法に切り替えるには、バックエンド構成から既存の方法を削除する必要があります。



資格情報と証明書の両方を提供しようとすると、構成ファイルに複数の認証方法が提供されているというエラーが発生し、バックエンドの作成が失敗します。

資格情報ベースの認証を有効にする

Trident、ONTAPバックエンドと通信するために、SVM スコープ/クラスタ スコープの管理者の認証情報が必要です。次のような標準の事前定義されたロールを利用することをお勧めします。`admin`または`vsadmin`。これにより、将来のTridentリリースで使用される機能APIを公開する可能性のある将来のONTAPリリースとの前方互換性が確保されます。カスタムセキュリティログインロールを作成してTridentで使用することは可能ですが、お勧めしません。

サンプルのバックエンド定義は次のようになります。

ヤムル

```
---
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、資格情報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留意してください。バックエンドが作成されると、ユーザー名とパスワードは Base64 でエンコードされ、Kubernetes シークレットとして保存されます。バックエンドの作成または更新は、資格情報に関する知識が必要となる唯一のステップです。したがって、これは Kubernetes/ストレージ管理者によって実行される管理者専用の操作です。

証明書ベースの認証の有効化

新規および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`: クライアント証明書の Base64 エンコードされた値。
- `clientPrivateKey`: 関連付けられた秘密キーの Base64 エンコードされた値。
- `trustedCACertificate`: 信頼された CA 証明書の Base64 エンコードされた値。信頼できる 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。信頼できる CA が使用されていない場合は、これを無視できます。

一般的なワークフローには次の手順が含まれます。

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、認証するONTAPユーザーに共通名 (CN) 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. 信頼できる CA 証明書をONTAPクラスタに追加します。これはストレージ管理者によってすでに処理され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. クライアント証明書とキー（手順 1 から）をONTAPクラスタにインストールします。

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



このコマンドを実行すると、ONTAPは証明書の入力を求めます。手順1で生成された `k8senv.pem` ファイルの内容を貼り付け、`END`を入力してインストールを完了します。

4. ONTAPセキュリティログインロールがサポートしていることを確認する `cert` 認証方法。

```
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します。 <ONTAP Management LIF> と <vserver name> を管理 LIF IP と SVM 名に置き換えます。

```
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. 証明書、キー、および信頼された CA 証明書を Base64 でエンコードします。

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

7. 前の手順で取得し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...Vaaalllluuuuueeee",
  "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、資格情報をローテーションする

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、資格情報をローテーションしたりすることができます。これは両方向に機能します。ユーザー名/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。また、証明書を使用するバックエンドは、ユーザー名/パスワードベースに更新できます。これを行うには、既存の認証方法を削除し、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実行します。tridentctl backend update

```

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 |
+-----+-----+-----+
+-----+-----+

```



パスワードをローテーションする場合、ストレージ管理者はまずONTAP上のユーザーのパスワードを更新する必要があります。続いてバックエンドの更新が行われます。証明書をローテーションする場合、ユーザーに複数の証明書を追加できます。その後、バックエンドは新しい証明書を使用するように更新され、その後、古い証明書を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、その後に行われたボリューム接続にも影響はありません。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、将来のボリューム操作を処理できることを示します。

Trident用のカスタムONTAPロールを作成する

最小限の権限を持つONTAPクラスタ ロールを作成すると、Tridentで操作を実行するためにONTAP管理者ロールを使用する必要がなくなります。Tridentバックエンド構成にユーザー名を含めると、Tridentは作成したONTAPクラスタ ロールを使用して操作を実行します。

参照 ["Tridentカスタムロールジェネレーター"](#) Tridentカスタム ロールの作成の詳細については、こちらをご覧ください。

ONTAP CLIの使用

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ユーザーのユーザー名を作成します。

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

3. ロールをユーザーにマップします。

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

System Managerを使用

ONTAP System Manager で次の手順を実行します。

1. カスタムロールを作成する:

- 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 > ストレージVM >** を選択します。 **required SVM > 設定 > ユーザーとロール**。

- b. ユーザーとロール*の横にある矢印アイコン (→*) を選択します。
- c. 役割*の下の+追加*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. 役割をTridentユーザーにマップします: + ユーザーと役割 ページで次の手順を実行します。

- a. ユーザー*の下の追加アイコン+*を選択します。
- b. 必要なユーザー名を選択し、*役割*のドロップダウン メニューで役割を選択します。
- c. *保存*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"](#)
- ["役割とユーザーを操作する"](#)

双方向CHAPによる接続の認証

Tridentは、双方向CHAPを使用してiSCSIセッションを認証できます。`ontap-san`そして`ontap-san-economy`ドライバ。これには、`useCHAP`バックエンド定義のオプション。に設定すると`true`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、バックエンド ファイルから

ユーザー名とシークレットを設定します。NetApp、接続の認証に双方向 CHAP を使用することを推奨しています。次のサンプル構成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
```



その `useCHAP` パラメータは一度だけ設定できるブールオプションです。デフォルトでは `false` に設定されています。`true` に設定した後は、`false` に設定することはできません。

に加えて `useCHAP=true`、`chapInitiatorSecret`、`chapTargetInitiatorSecret`、`chapTargetUsername`、そして `chapUsername` フィールドはバックエンド定義に含める必要があります。バックエンドを作成した後、次のコマンドを実行することでシークレットを変更できます。
`tridentctl update`。

仕組み

設定により `useCHAP=true` に設定すると、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 を設定する:
 - 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）であり、かつポリシーーム内に既存のLUNが存在しない場合、Tridentはデフォルトのセキュリティタイプを次のように設定します。CHAP CHAP イニシエーターとターゲットのユーザー名とシークレットの構成に進みます。
 - SVM に LUN が含まれている場合、Trident はSVM 上で CHAP を有効にしません。これにより、SVM 上にすでに存在する LUN へのアクセスが制限されなくなります。
- CHAP イニシエーターとターゲットのユーザー名とシークレットを構成します。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 `tridentbackend` CRD は、CHAP シークレットとユーザー名を Kubernetes シークレットとして保存します。このバックエンドで Tridentによって作成されるすべての PV は、CHAP 経由でマウントおよび接続されます。

認証情報をローテーションしてバックエンドを更新する

CHAP認証情報を更新するには、CHAPパラメータを更新します。`backend.json`ファイル。これにはCHAPシークレットを更新し、`tridentctl update`これらの変更を反映するコマンド。



バックエンドのCHAPシークレットを更新する場合は、`tridentctl`バックエンドを更新します。Trident はこれらの変更を取得できないため、ONTAP CLI またはONTAP System Manager を使用してストレージ クラスターの資格情報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ません。SVM 上のTridentによって資格情報が更新されると、既存の接続は引き続きアクティブなままになります。新しい接続では更新された資格情報が使用され、既存の接続は引き続きアクティブなままになります。古い PV を切断して再接続すると、更新された資格情報が使用されるようになります。

ONTAP SAN 構成オプションと例

TridentインストールでONTAP SAN ドライバーを作成して使用方法を学習します。このセクションでは、バックエンドを StorageClasses にマッピングするためのバックエンド構成の例と詳細について説明します。

"ASA r2 システム"ストレージ層の実装は他のONTAPシステム (ASA、AFF、FAS) とは異なります。これらのバリエーションは、記載されている特定のパラメータの使用に影響します。"ASA r2 システムと他のONTAP システムの違いについて詳しくは、[こちらをご覧ください。](#)"



のみ `ontap-san` ドライバー (iSCSI および NVMe/TCP プロトコル付き) は、ASA r2 システムでサポートされています。

Tridentバックエンド構成では、システムがASA r2であることを指定する必要はありません。選択すると `ontap-san` として `storageDriverName` Trident は、ASA r2 または従来のONTAPシステムを自動的に検出します。以下の表に示すように、一部のバックエンド構成パラメータはASA r2 システムには適用されません。

バックエンド構成オプション

バックエンドの構成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
version		常に1
storageDriverName	ストレージ ドライバーの名前	ontap-san`または `ontap-san-economy
backendName	カスタム名またはストレージバックエンド	ドライバー名 + "_" + dataLIF
managementLIF	<p>クラスタまたは SVM 管理 LIF の IP アドレス。</p> <p>完全修飾ドメイン名 (FQDN) を指定できます。</p> <p>Trident がIPv6 フラグを使用して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アドレスは角括弧で囲んで定義する必要があります。例: [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。</p> <p>シームレスなMetroClusterスイッチオーバーについては、MetroClusterの例。</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> <p> 「vsadmin」の資格情報を使用している場合は、managementLIF SVMの認証情報である必要があります。「admin」認証情報を使用する場合は、`managementLIF` クラスタのものである必要があります。</p> </div>	"10.0.0.1"、"[2001:1234:abcd::fefe]"
dataLIF	<p>プロトコル LIF の IP アドレス。Trident がIPv6 フラグを使用して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アドレスは角括弧で囲んで定義する必要があります。例: [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"ONTAP選択的 LUN マップ"マルチパス セッションを確立するために必要な iSCSI LIF を検出します。警告が発生するのは、`dataLIF`明示的に定義されています。*Metrocluster の場合は省略します。*参照MetroClusterの例。</p>	SVMによって導出された

パラメータ	説明	デフォルト
svm	使用するストレージ仮想マシン *Metrocluster の場合は省略。*参照 MetroClusterの例 。	SVMの場合導出される `managementLIF`指定されている
useCHAP	CHAP を使用してONTAP SAN ドライバーの iSCSI を認証します [ブール値]。設定 `true` Trident が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 を設定して使用できるようにします。参照 "ONTAP SAN ドライバーを使用してバックエンドを構成する準備をする" 詳細については、 FCP または NVMe/TCP ではサポートされません。	false
chapInitiatorSecret	CHAP イニシエーター シークレット。必須の場合 useCHAP=true	""
labels	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 イニシエーター シークレット。必須の場合 useCHAP=true	""
chapUsername	受信ユーザー名。必須の場合 useCHAP=true	""
chapTargetUsername	ターゲットユーザー名。必須の場合 useCHAP=true	""
clientCertificate	クライアント証明書の Base64 エンコードされた値。証明書ベースの認証に使用	""
clientPrivateKey	クライアント秘密キーの Base64 エンコードされた値。証明書ベースの認証に使用	""
trustedCACertificate	信頼された CA 証明書の Base64 エンコードされた値。オプション。証明書ベースの認証に使用されます。	""
username	ONTAPクラスタと通信するために必要なユーザー名。資格情報ベースの認証に使用されます。Active Directory認証については、 "Active Directory の認証情報を使用して、バックエンド SVM に対してTrident を認証する" 。	""
password	ONTAPクラスタと通信するために必要なパスワード。資格情報ベースの認証に使用されます。Active Directory認証については、 "Active Directory の認証情報を使用して、バックエンド SVM に対してTrident を認証する" 。	""
svm	使用するストレージ仮想マシン	SVMの場合導出される `managementLIF`指定されている
storagePrefix	SVM で新しいボリュームをプロビジョニングするときに使用されるプレフィックス。後で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
aggregate	<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)。のために `ontap-nas-flexgroup` ドライバーの場合、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、使用可能なアグリゲートのいずれかを使用してFlexGroupボリュームをプロビジョニングできます。</p> <div style="border: 1px solid gray; padding: 10px; margin: 10px 0;"> <p> SVM でアグリゲートが更新されると、Tridentコントローラーを再起動せずに、SVM をポーリングすることによってTridentでも自動的に更新されます。Tridentで特定のアグリゲートをボリュームのプロビジョニング用に構成した場合、アグリゲートの名前が変更されたり、SVM から移動されたりすると、SVM アグリゲートをポーリングしているときに、Tridentでバックエンドが障害状態になります。バックエンドをオンラインに戻すには、アグリゲートを SVM 上に存在するものに変更するか、完全に削除する必要があります。</p> </div> <p>• ASA r2 システムでは指定しないでください*。</p>	""
limitAggregateUsage	<p>使用率がこのパーセンテージを超える場合、プロビジョニングは失敗します。Amazon FSx for NetApp ONTAPバックエンドを使用している場合は、指定しないでください。limitAggregateUsage。提供された `fsxadmin` そして `vsadmin` 集計使用量を取得し、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。* ASA r2 システムでは指定しないでください*。</p>	"" (デフォルトでは強制されません)
limitVolumeSize	<p>要求されたボリューム サイズがこの値を超える場合、プロビジョニングは失敗します。また、LUN に対して管理するボリュームの最大サイズも制限します。</p>	"" (デフォルトでは強制されません)
lunsPerFlexvol	<p>Flexvolあたりの最大LUN数は[50, 200]の範囲でなければなりません</p>	100
debugTraceFlags	<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} トラブルシューティングを行っており、詳細なログ ダンプが必要な場合を除き、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ル パラメータ。</p> <div style="border: 1px solid gray; padding: 10px; margin: 10px 0;"> <p> `useREST` に設定すると `true`、Trident は ONTAP REST API を使用してバックエンドと通信します。 `false` Trident は、バックエンドとの通信に ONTAPI (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。さらに、使用する ONTAP ロールには、`ontapi` 応用。これは、事前に定義された `vsadmin` そして `cluster-admin` 役割。 Trident 24.06 リリースおよび ONTAP 9.15.1 以降では、`useREST` 設定されている `true` デフォルト; 変更 `useREST` に `false` ONTAPI (ZAPI) 呼び出しを使用します。 </p> </div> <p> `useREST` NVMe/TCP に完全対応しています。 </p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p>  NVMe は ONTAP REST API でのみサポートされ、ONTAPI (ZAPI) ではサポートされません。 </p> </div> <p> 指定されている場合、常に `true` ASA r2 システムの場合。 </p>	true`ONTAP 9.15.1以降の場合、それ以外の場合 `false。
sanType	<p> 選択するには使用 `iscsi` iSCSI の場合、 `nvme` NVMe/TCP の場合または `fcp` SCSI over Fibre Channel (FC) 用。 </p>	`iscsi` 空白の場合

パラメータ	説明	デフォルト
formatOptions	<p>使用 `formatOptions` コマンドライン引数を指定するには `mkfs` 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 オプションと同様の formatOptions を指定してください。例: "-E nodiscard"</p> <p>対応機種 `ontap-san` そして `ontap-san-economy` iSCSI プロトコルを使用したドライバー。さらに、iSCSI および NVMe/TCP プロトコルを使用する場合、ASA r2 システムでもサポートされます。</p>	
limitVolumePoolSize	ontap-san-economy バックエンドで LUN を使用する場合の、要求可能な FlexVol の最大サイズ。	"" (デフォルトでは強制されません)
denyNewVolumePools	制限 `ontap-san-economy` バックエンドが LUN を格納するための新しい FlexVol ボリュームを作成できないようにします。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。	

formatOptionsの使用に関する推奨事項

Trident 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ます。

-E 破棄なし:

- 保持し、mkfs 時にブロックを破棄しないでください (最初にブロックを破棄することは、ソリッドステート デバイスおよびスパス/シン プロビジョニング ストレージで役立ちます)。これは非推奨のオプション「-K」に代わるもので、すべてのファイルシステム (xfs、ext3、ext4) に適用できます。

Active Directory の認証情報を使用して、バックエンド SVM に対して Trident を認証する

Active Directory (AD) 認証情報を使用してバックエンド SVM に対して認証するように Trident を設定できます。AD アカウントが SVM にアクセスする前に、クラスタまたは SVM への AD ドメイン コントローラ アクセスを設定する必要があります。AD アカウントを使用してクラスタを管理するには、ドメイン トンネルを作成する必要があります。参照 ["ONTAPでActive Directoryドメインコントローラのアクセスを構成する"](#) 詳細については。

手順

1. バックエンド SVM のドメイン ネーム システム (DNS) 設定を構成します。

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. 次のコマンドを実行して、Active Directory に SVM のコンピュータ アカウントを作成します。

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. このコマンドを使用して、クラスタまたはSVMを管理するためのADユーザーまたはグループを作成しま

す。

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. Tridentバックエンド設定ファイルで、username そして password パラメータをそれぞれ AD ユーザー名またはグループ名とパスワードに渡します。

ボリュームのプロビジョニングのためのバックエンド構成オプション

デフォルトのプロビジョニングは、以下のオプションを使用して制御できます。`defaults`構成のセクション。例については、以下の構成例を参照してください。

パラメータ	説明	デフォルト
spaceAllocation	LUNのスペース割り当て	"true" 指定されている場合は、 true ASA r2 システムの場合。
spaceReserve	スペース予約モード。「なし」(薄い)または「ボリューム」(厚い)。設定`none`ASA r2 システムの場合。	"なし"
snapshotPolicy	使用するスナップショット ポリシー。設定`none`ASA r2 システムの場合。	"なし"
qosPolicy	作成され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 グループが各構成要素に個別に適用されるようにする必要があります。共有 QoS ポリシー グループは、すべてのワークロードの合計スループットの上限を適用します。	""
adaptiveQosPolicy	作成されたボリュームに割り当てるアダプティブ 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	""
snapshotReserve	スナップショット用に予約されているボリュームの割合。* ASA r2 システムでは指定しないでください*。	「0」の場合`snapshotPolicy`は「なし」、それ以外の場合は「」
splitOnClone	クローン作成時に親からクローンを分割する	"間違い"
encryption	新しいボリュームでNetAppボリューム暗号化 (NVE) を有効にします。デフォルトは false。このオプションを使用するには、NVE のライセンスを取得し、クラスターで有効にする必要があります。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、以下を参照してください。 "Trident がNVE および NAE と連携する仕組み" 。	"false" 指定されている場合は、 true ASA r2 システムの場合。

パラメータ	説明	デフォルト
luksEncryption	LUKS 暗号化を有効にします。参照" Linux Unified Key Setup (LUKS) を使用する "。	"" 設定 `false` ASA r2 システムの場合。
tieringPolicy	階層化ポリシーは「なし」を使用します。* ASA r2 システムでは指定しないでください。*	
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

デフォルトを定義した例を次に示します。

```

---
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'

```



作成されたすべてのボリュームについて `ontap-san` ドライバーにより、Trident は LUN メタデータに対応するために FlexVol に 10 パーセントの容量を追加します。LUN は、ユーザーが PVC で要求した正確なサイズでプロビジョニングされます。Trident は FlexVol に 10 パーセントを追加します (ONTAP では使用可能なサイズとして表示されます)。ユーザーは要求した使用可能な容量を取得できるようになります。この変更により、使用可能なスペースが完全に使用されない限り、LUN が読み取り専用になることも防止されます。これは ontap-san-economy には適用されません。

定義するバックエンドの場合 `snapshotReserve` Trident はボリュームのサイズを次のように計算します。

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve percentage) / 100)] * 1.1
```

にTridentがFlexVolに追加する10%の容量です。のために snapshotReserve= 5%、PVC 要求 = 5 GiB の場合、ボリュームの合計サイズは 5.79 GiB、使用可能なサイズは 5.5 GiB になります。その `volume show` コマンドを実行すると、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使用する唯一の方法は、サイズ変更です。

最小限の構成例

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構成を示しています。これはバックエンドを定義する最も簡単な方法です。



Tridentを搭載したNetApp ONTAPでAmazon FSx を使用している場合、NetApp、LIF に IP アドレスではなく DNS 名を指定することを推奨しています。

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>
```

MetroClusterの例

バックエンドを設定することで、スイッチオーバーとスイッチバック後にバックエンド定義を手動で更新する必要がなくなります。["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行うには、SVMを次のように指定します。``managementLIF``そして省略する ``svm`` パラメータ。例えば：

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

ONTAP SANエコノミーの例

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

証明書ベースの認証の例

この基本構成例では `clientCertificate`、`clientPrivateKey`、そして `trustedCACertificate`（信頼できるCAを使用する場合はオプション）が入力されます。`backend.json` クライアント証明書、秘密キー、信頼できる CA 証明書の base64 エンコードされた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: c19qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

双方向CHAPの例

これらの例では、`useCHAP``に設定 `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バックエンドに NVMe が設定された SVM が必要です。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
```

SCSI over FC (FCP) の例

ONTAPバックエンドに FC が設定された SVM が必要です。これは FC の基本的なバックエンド構成です。

```
---  
version: 1  
backendName: fcp-backend  
storageDriverName: ontap-san  
managementLIF: 10.0.0.1  
svm: svm_fc  
username: vsadmin  
password: password  
sanType: fcp  
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}}_{{.vo\
      lume.RequestName}}"
labels:
  cluster: ClusterA
PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

ontap-san-economy ドライバーの 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
```

仮想プールを備えたバックエンドの例

これらのサンプルバックエンド定義ファイルでは、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。`spaceReserve` どれも、`spaceAllocation` 偽で、そして `encryption` 偽です。仮想プールはストレージ セクションで定義されます。

Trident は、「コメント」フィールドにプロビジョニング ラベルを設定します。コメントは FlexVol volume に設定され、Trident はプロビジョニング時に仮想プールに存在するすべてのラベルをストレージ ボリュームにコピーします。便宜上、ストレージ管理者は仮想プールごとにラベルを定義し、ラベルごとにボリュームをグ

ループ化できます。

これらの例では、一部のストレージプールは独自の `spaceReserve`、`spaceAllocation`、そして `encryption` 値があり、一部のプールはデフォルト値を上書きします。



```
---
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"
```

```

---
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にマッピングする

以下のStorageClass定義は、[\[仮想プールを備えたバックエンドの例\]](#)。使用して`parameters.selector`フィールドでは、各StorageClassはボリュームをホストするため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で定義された側面が設定されます。

- その`protection-gold`ストレージクラスは、`ontap-san`バックエンド。これは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"
```

- その `protection-not-gold` ストレージクラスは、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"
```

- その `app-mysqldb` StorageClass は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 タイプのアプリにストレージ 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"
```

- その `protection-silver-creditpoints-20k` ストレージクラスは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。これは、シルバー レベルの保護と 20,000 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"
```

- その `creditpoints-5k` StorageClassは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で提供される唯一のプールです。

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

- その `my-test-app-sc` StorageClassは `testAPP` 仮想プール `ontap-san` ドライバー付き `sanType: nvme`。これは唯一のプールの提供です `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 はどの仮想プールが選択されるか決定し、ストレージ要件が満たされていることを確認します。

ONTAP NAS ドライバー

ONTAP NAS ドライバーの概要

ONTAPおよびCloud Volumes ONTAP NAS ドライバーを使用してONTAPバックエンド

を構成する方法について学習します。

ONTAP NAS ドライバーの詳細

Trident は、ONTAP クラスタと通信するための次の NAS ストレージ ドライバーを提供します。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	サポートされているファイルシステム
ontap-nas	NFS SMB	Filesystem	RWO、ROX、RWX、RWOP	""、nfs、smb
ontap-nas-economy	NFS SMB	Filesystem	RWO、ROX、RWX、RWOP	""、nfs、smb
ontap-nas-flexgroup	NFS SMB	Filesystem	RWO、ROX、RWX、RWOP	""、nfs、smb



- 使用 `ontap-san-economy` 永続ボリュームの使用数が"[サポートされているONTAPボリュームの制限](#)"。
- 使用 `ontap-nas-economy` 永続ボリュームの使用数が"[サポートされているONTAPボリュームの制限](#)"そして `ontap-san-economy` ドライバーは使用できません。
- 使用しないでください `ontap-nas-economy` データ保護、災害復旧、モビリティの必要性が予想される場合。
- NetApp、ontap-san を除くすべてのONTAPドライバーで Flexvol autogrow を使用することは推奨されていません。回避策として、Trident はスナップショット リザーブの使用をサポートし、それに応じて Flexvol ボリュームを拡張します。

ユーザー権限

Trident は、通常、ONTAP または SVM 管理者として実行することを想定しています。`admin` クラスターユーザーまたは `vsadmin` SVM ユーザー、または同じロールを持つ別の名前のユーザー。

Amazon FSx for NetApp ONTAP の導入では、Trident はクラスタを使用して ONTAP または SVM 管理者として実行されることが想定されています。`fsxadmin` ユーザーまたは `vsadmin` SVM ユーザー、または同じロールを持つ別の名前のユーザー。その `fsxadmin` ユーザーは、クラスター管理者ユーザーの限定的な代替です。



を使用する場合 `limitAggregateUsage` パラメータには、クラスター管理者の権限が必要です。Amazon FSx for NetApp ONTAP を Trident で使用する場合、`limitAggregateUsage` パラメータは、`vsadmin` そして `fsxadmin` ユーザーアカウント。このパラメータを指定すると、構成操作は失敗します。

ONTAP 内で Trident ドライバーが使用できる、より制限の厳しいロールを作成することは可能ですが、お勧めしません。Trident のほとんどの新しいリリースでは、考慮する必要がある追加の API が呼び出されるため、アップグレードが困難になり、エラーが発生しやすくなります。

ONTAP NAS ドライバーを使用してバックエンドを構成する準備をする

ONTAP NAS ドライバーを使用してONTAPバックエンドを構成するための要件、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バックエンドでは、Trident少なくとも1つのアグリゲートをSVMに割り当てる必要があります。
- 複数のドライバーを実行し、いずれかを指すストレージクラスを作成できます。たとえば、`ontap-nas` ドライバーとブロンズクラスは`ontap-nas-economy`1つ。
- すべてのKubernetes ワーカーノードに適切なNFS ツールがインストールされている必要があります。参照["ここをクリックしてください。"](#)詳細についてはこちらをご覧ください。
- Trident 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 ボリュームのみをサポートします。参照 [SMBボリュームのプロビジョニングの準備](#) 詳細については。

ONTAPバックエンドを認証する

Trident は、ONTAPバックエンドを認証する2つのモードを提供します。

- 認証情報ベース: このモードでは、ONTAPバックエンドに対する十分な権限が必要です。次のような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お勧めします。`admin`または`vsadmin`ONTAPバージョンとの最大限の互換性を確保するためです。
- 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がインストールされている必要があります。ここで、バックエンド定義には、クライアント証明書、キー、および信頼されたCA証明書（使用する場合）のBase64 エンコードされた値が含まれている必要があります（推奨）。

既存のバックエンドを更新して、資格情報ベースの方法と証明書ベースの方法間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。別の認証方法に切り替えるには、バックエンド構成から既存の方法を削除する必要があります。



資格情報と証明書の両方を提供しようとすると、構成ファイルに複数の認証方法が提供されているというエラーが発生し、バックエンドの作成が失敗します。

資格情報ベースの認証を有効にする

Trident、ONTAPバックエンドと通信するために、SVM スコープ/クラスタ スコープの管理者の認証情報が必要です。次のような標準の事前定義されたロールを利用することをお勧めします。`admin`または`vsadmin`。これにより、将来のTridentリリースで使用される機能APIを公開する可能性のある将来のONTAPリリースとの前方互換性が確保されます。カスタムセキュリティログインロールを作成してTridentで使用することは可能ですが、お勧めしません。

サンプルのバックエンド定義は次のようになります。

ヤムル

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
    "name": "secret-backend-creds"
  }
}
```

バックエンド定義は、資格情報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留意してください。バックエンドが作成されると、ユーザー名とパスワードは Base64 でエンコードされ、Kubernetes シークレットとして保存されます。バックエンドの作成/更新は、資格情報に関する知識が必要となる唯一のステップです。したがって、これは Kubernetes/ストレージ管理者によって実行される管理者専用の操作です。

証明書ベースの認証を有効にする

新規および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`: クライアント証明書の Base64 エンコードされた値。
- `clientPrivateKey`: 関連付けられた秘密キーの Base64 エンコードされた値。
- `trustedCACertificate`: 信頼された CA 証明書の Base64 エンコードされた値。信頼できる 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。信頼できる CA が使用されていない場合は、これを無視できます。

一般的なワークフローには次の手順が含まれます。

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、認証するONTAPユーザーに共通名 (CN) 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. 信頼できる CA 証明書をONTAPクラスタに追加します。これはストレージ管理者によってすでに処理され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. クライアント証明書とキー（手順 1 から）をONTAPクラスタにインストールします。

```
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サポートしていることを確認する `cert` 認証方法。

```
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します。< ONTAP Management LIF> と <vserver name> を管理 LIF IP と SVM 名に置き換えます。LIFのサービスポリシーが次のように設定されていることを確認する必要があります。 default-data-management。

```
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. 証明書、キー、および信頼された CA 証明書を Base64 でエンコードします。

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

7. 前の手順で取得し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...Vaaallllluuuueeee",
  "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、資格情報をローテーションする

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、資格情報をローテーションしたりすることができます。これは両方向に機能します。ユーザー名/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。また、証明書を使用するバックエンドは、ユーザー名/パスワードベースに更新できます。これを行うには、既存の認証方法を削除し、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
NasBackend	ontap-nas	98e19b74-aec7-4a3d-8dcf-128e5033b214

```

STATE | VOLUMES |
online | 9 |

```



パスワードをローテーションする場合、ストレージ管理者はまずONTAP上のユーザーのパスワードを更新する必要があります。続いてバックエンドの更新が行われます。証明書をローテーションする場合、ユーザーに複数の証明書を追加できます。その後、バックエンドは新しい証明書を使用するように更新され、その後、古い証明書を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、その後に行われたボリューム接続にも影響はありません。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、将来のボリューム操作を処理できることを示します。

Trident用のカスタムONTAPロールを作成する

最小限の権限を持つONTAPクラスタロールを作成すると、Tridentで操作を実行するためにONTAP管理者ロールを使用する必要がなくなります。Tridentバックエンド構成にユーザー名を含めると、Tridentは作成したONTAPクラスタロールを使用して操作を実行します。

参照"[Tridentカスタムロールジェネレーター](#)" Tridentカスタムロールの作成の詳細については、こちらをご覧ください。

ONTAP CLIの使用

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ユーザーのユーザー名を作成します。

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

3. ロールをユーザーにマップします。

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

System Managerを使用

ONTAP System Manager で次の手順を実行します。

1. カスタムロールを作成する:

- 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 > ストレージVM >** を選択します。 **required SVM > 設定 > ユーザーとロール**。

- b. ユーザーとロール*の横にある矢印アイコン (→*) を選択します。
- c. 役割*の下の+追加*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. 役割をTridentユーザーにマップします: + ユーザーと役割 ページで次の手順を実行します。

- a. ユーザー*の下の追加アイコン+*を選択します。
- b. 必要なユーザー名を選択し、*役割*のドロップダウン メニューで役割を選択します。
- c. *保存*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"](#)
- ["役割とユーザーを操作する"](#)

NFSエクスポートポリシーを管理する

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、エクスポート ポリシーを操作するときに2つのオプションが提供されます。

- Trident はエクスポート ポリシー自体を動的に管理できます。この動作モードでは、ストレージ管理者は許容 IP アドレスを表す CIDR ブロックのリストを指定します。Trident は、公開時にこれらの範囲内にある該当するノード IP をエクスポート ポリシーに自動的に追加します。あるいは、CIDR が指定されていない場合は、ボリュームが公開されるノードで見つかったすべてのグローバル スコープのユニキャスト IP がエクスポート 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、ルールを手動で追加できます。構成で別のエクスポート ポリシー名が指定されていない限り、Trident はデフォルトのエクスポート ポリシーを使用します。

エクスポートポリシーを動的に管理する

Trident 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する機能を提供します。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 IP に許可されるアドレス空間を指定できるようになります。これにより、エクスポート ポリシーの管理が大幅に簡素化され、エクスポート ポリシーを変更する際にストレージ クラスターで手動で介入する必要がなくなります。さらに、これにより、ボリュームをマウントしており、指定された範囲内の IP を持つワーカー ノードのみにストレージ クラスターへのアクセスが制限され、きめ細かな自動管理がサポートされます。



動的エクスポート ポリシーを使用する場合は、ネットワーク アドレス変換 (NAT) を使用しないでください。NAT では、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 NAT アドレスを認識するため、エクスポート ルールに一致するものが見つからない場合はアクセスが拒否されます。

例

使用する必要がある構成オプションが2つあります。バックエンドの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 CIDR ブロックを許可するエクスポート ルールを含むエクスポート ポリシーが以前に作成されていることを確認する必要があります (デフォルトのエクスポート ポリシーなど)。SVM を Trident 専用にする場合は、常に NetApp が推奨するベスト プラクティスに従ってください。

上記の例を使用して、この機能がどのように機能するかを説明します。

- `autoExportPolicy` `設定されている` `true`。これは、Tridentがこのバックエンドで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することを示します。 `svm1` SVMを使用してルールの追加と削除を処理します `autoexportCIDRs` アドレス ブロック。ボリュームがノードに接続されるまで、そのボリュームへの不要なアクセスを防ぐためのルールのない空のエクスポート ポリシーがボリュームで使用されます。ボリュームがノードに公開されると、Trident は指定された CIDR ブロック内のノード IP を含む基礎となる `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 volumeで使用されるエクスポートポリシーにも追加されます。
 - 例えば：
 - 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
 - `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`という名前の`qtree`は、FlexVolという名前のエクスポートポリシーを作成します。 `trident-403b5326-8482-40db96d0-d83fb3f4daec`、`qtree`のエクスポートポリシー `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`、および空のエクスポートポリシー `trident_empty` SVM 上。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、接続されていないボリュームによって再利用されます。
- `autoExportCIDRs` `アドレス ブロックのリストが含まれます。このフィールドはオプションであり、デフォルトは ["0.0.0.0/0", ":::/0"] になります。定義されていない場合、Trident はパブリケーションを持つワーカー ノードで見つかったすべてのグローバル スコープのユニキャスト アドレスを追加します。

この例では、`192.168.0.0/24` `アドレス空間が提供されます。これは、公開されているこのアドレス範囲内にある 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、そのノードを登録する際に、そのノードのIPアドレスを取得し、それを以下のアドレスブロックと照合します。``autoExportCIDRs``公開時に、IP をフィルタリングした後、Trident は公開先の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。

更新できます `autoExportPolicy` `そして` `autoExportCIDRs` `バックエンドを作成した後。自動的に管理されるバックエンドに新しい CIDR を追加したり、既存の CIDR を削除したりできます。CIDR を削除するときは、既存の接続が切断されないように注意してください。無効にすることもできます `autoExportPolicy` `バックエンドにエクスポート ポリシーを手動で作成し、フォールバックします。これには、``exportPolicy``バックエンド構成のパラメータ。

Tridentがバックエンドを作成または更新した後、以下のコマンドでバックエンドを確認できます。``tridentctl``または対応する``tridentbackend``CRD:

```
./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
```

ノードが削除されると、Trident はすべてのエクスポート ポリシーをチェックし、そのノードに対応するアクセスルールを削除します。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 を削除することにより、この IP がクラスター内の新しいノードによって再利用されない限り、Trident は不正なマウントを防止します。

既存のバックエンドの場合は、バックエンドを次のように更新します。tridentctl update backend Trident がエクスポート ポリシーを自動的に管理することを保証します。これにより、必要に応じて、バックエンドの UUID と qtree 名に基づいて名前が付けられた 2 つの新しいエクスポート ポリシーが作成されます。バックエンドに存在するボリュームは、アンマウントされて再度マウントされた後、新しく作成されたエクスポート ポリシーを使用します。



自動管理エクスポート ポリシーを持つ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 ポリシーも削除されます。バックエンドが再作成されると、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。

ライブ ノードの IP アドレスが更新された場合は、ノード上の Trident ポッドを再起動する必要があります。その後、Trident は、この IP 変更を反映するために、管理するバックエンドのエクスポート ポリシーを更新します。

SMB ボリュームのプロビジョニングの準備

少しの追加の準備をすれば、SMB ボリュームをプロビジョニングできます。`ontap-nas` ドライバー。



SVM で NFS 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 ONTAP オンプレミス クラスターの SMB ボリューム。これらのプロトコルのいずれかを構成しないと、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。

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、次のものがが必要です。

- Linux コントローラー ノードと、Windows Server 2022 を実行する少なくとも 1 つの Windows ワーカー ノードを備えた Kubernetes クラスター。Trident 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 ボリュームのみをサポートします。
- Active Directory 資格情報を含む少なくとも 1 つのTridentシークレット。秘密を生成する smbcreds:

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

- Windows サービスとして構成された CSI プロキシ。設定するには `csi-proxy`、参照["GitHub: CSIプロキシ"](#)または["GitHub: Windows 用 CSI プロキシ"](#)Windows 上で実行されている Kubernetes ノード用。

手順

1. オンプレミスのONTAPの場合、オプションで SMB 共有を作成するか、Trident で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、次の2つの方法のいずれかで作成できます。["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。

その `vservers cifs share create` コマンドは、共有の作成中に `-path` オプションで指定されたパスをチェックします。指定されたパスが存在しない場合、コマンドは失敗します。

- b. 指定された SVM に関連付けられた SMB 共有を作成します。

```
vservers cifs share create -vservers vservers_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。

```
vservers cifs share show -share-name share_name
```



参照["SMB共有を作成する"](#)詳細についてはこちらをご覧ください。

2. バックエンドを作成するときは、SMB ボリュームを指定するために以下を構成する必要があります。FSx for ONTAPのバックエンド構成オプションについては、以下を参照してください。["FSx for 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 がSMB 共有を作成できるようにする名前、またはボリュームへの共通共有アクセスを防止するためにパラメータを空白のままにしておくことができます。このパラメータは、オンプレミスのONTAPではオプションです。このパラメータはAmazon FSx for ONTAPバックエンドに必須であり、空白にすることはできません。	smb-share
nasType	設定する必要があります smb . nullの場合、デフォルトは nfs 。	smb
securityStyle	新しいボリュームのセキュリティ スタイル。設定する必要があります `ntfs` または `mixed` SMB ボリュームの場合。	`ntfs` または `mixed` SMB ボリューム用
unixPermissions	新しいボリュームのモード。 SMB ボリュームの場合は空のままにする必要があります。	""

安全なSMBを有効にする

25.06リリース以降、NetApp Tridentは、以下の方法で作成されたSMBボリュームの安全なプロビジョニングをサポートします。`ontap-nas`そして`ontap-nas-economy`バックエンド。セキュアSMBを有効にすると、アクセス制御リスト (ACL) を使用して、Active Directory (AD) ユーザーおよびユーザー グループにSMB共有への制御されたアクセスを提供できます。

覚えておくべきポイント

- インポート `ontap-nas-economy` ボリュームはサポートされていません。
- 読み取り専用クローンのみがサポートされています `ontap-nas-economy` ボリューム。
- Secure SMB が有効になっている場合、Trident はバックエンドに記載されているSMB共有を無視しません。
- PVC アノテーション、ストレージクラス アノテーション、およびバックエンド フィールドを更新しても、SMB共有ACLは更新されません。
- クローン PVC の注釈で指定されたSMB共有ACLは、ソースPVCのACLよりも優先されます。
- 安全なSMBを有効にする際には、有効なADユーザーを提供するようにしてください。無効なユーザーはACLに追加されません。
- バックエンド、ストレージクラス、PVCで同じADユーザーに異なる権限を指定した場合、権限の優先順位はPVC、ストレージクラス、バックエンドの順になります。
- セキュアSMBは以下でサポートされています `ontap-nas` 管理対象ボリュームのインポートには適用され、管理対象外ボリュームのインポートには適用されません。

手順

1. 次の例に示すように、TridentBackendConfig で adAdminUser を指定します。

```

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret

```

2. ストレージ クラスに注釈を追加します。

追加する `trident.netapp.io/smbShareAdUser` ストレージ クラスにアノテーションを追加して、安全な SMB を確実に有効にします。注釈に指定されたユーザー値 `trident.netapp.io/smbShareAdUser` 指定されたユーザー名と同じである必要があります `smbcreds` 秘密。次のいずれかを選択できます `smbShareAdUserPermission: full_control`、`change`、または `read`。デフォルトの権限は `full_control`。

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate

```

1. PVCを作成します。

次の例では、PVC を作成します。

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc

```

ONTAP NAS の構成オプションと例

TridentインストールでONTAP NAS ドライバーを作成して使用方法を学習します。このセクションでは、バックエンドを StorageClasses にマッピングするためのバックエンド構成の例と詳細について説明します。

バックエンド構成オプション

バックエンドの構成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
version		常に1
storageDriverName	ストレージドライバーの名前	ontap-nas、ontap-nas-economy、またはontap-nas-flexgroup
backendName	カスタム名またはストレージバックエンド	ドライバー名 + "_" + dataLIF
managementLIF	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。Trident がIPv6 フラグを使用して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アドレスは角括弧で囲んで定義する必要があります。例: [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、 MetroClusterの例 。	"10.0.0.1"、"[2001:1234:abcd::fefe]"

パラメータ	説明	デフォルト
dataLIF	プロトコル LIF の IP アドレス。NetAppは以下を指定することを推奨しています dataLIF。指定されない場合、Trident はSVM から dataLIF を取得します。NFS マウント操作に使用する完全修飾ドメイン名 (FQDN) を指定できるため、複数のデータ LIF 間で負荷分散を行うラウンドロビン DNS を作成できます。初期設定後も変更可能です。参照。Trident がIPv6 フラグを使用して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 アドレスは角括弧で囲んで定義する必要があります。例: [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。*Metrocluster の場合は省略します。*参照MetroClusterの例。	指定されたアドレス、または指定されていない場合は SVM から派生したアドレス (非推奨)
svm	使用するストレージ仮想マシン *Metrocluster の場合は省略。*参照MetroClusterの例。	SVMの場合導出される `managementLIF` 指定されている
autoExportPolicy	自動エクスポート ポリシーの作成と更新を有効にします [ブール値]。使用して `autoExportPolicy` そして `autoExportCIDRs` オプションを使用すると、Trident はエクスポート ポリシーを自動的に管理できます。	false
autoExportCIDRs	KubernetesのノードIPをフィルタリングするためのCIDRのリスト `autoExportPolicy` が有効になります。使用して `autoExportPolicy` そして `autoExportCIDRs` オプションを使用すると、Trident はエクスポート ポリシーを自動的に管理できます。	["0.0.0.0/0", ":::0"]
labels	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	クライアント証明書の Base64 エンコードされた値。証明書ベースの認証に使用	""
clientPrivateKey	クライアント秘密キーの Base64 エンコードされた値。証明書ベースの認証に使用	""
trustedCACertificate	信頼された CA 証明書の Base64 エンコードされた値。オプション。証明書ベースの認証に使用	""
username	クラスター/SVM に接続するためのユーザー名。資格情報ベースの認証に使用されます。Active Directory認証については、" Active Directory の認証情報を使用して、バックエンド SVM に対してTrident を認証する "。	
password	クラスター/SVM に接続するためのパスワード。資格情報ベースの認証に使用されます。Active Directory認証については、" Active Directory の認証情報を使用して、バックエンド SVM に対してTrident を認証する "。	

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ときに使用されるプレフィックス。設定後は更新できません</p> <p> ontap-nas-economy と 24 文字以上の storagePrefix を使用する場合、ボリューム名にはストレージプレフィックスが埋め込まれますが、qtree には埋め込まれません。</p>	"トライデント"
aggregate	<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)。のために `ontap-nas-flexgroup` ドライバーの場合、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、使用可能なアグリゲートのいずれかを使用して 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 コントローラーを再起動せずに、SVM をポーリングすることによって Trident でも自動的に更新されます。Trident で特定のアグリゲートをボリュームのプロビジョニング用に構成した場合、アグリゲートの名前が変更されたり、SVM から移動されたりすると、SVM アグリゲートをポーリングしているときに、Trident でバックエンドが障害状態になります。バックエンドをオンラインに戻すには、アグリゲートを SVM 上に存在するものに変更するか、完全に削除する必要があります。</p>	""
limitAggregateUsage	<p>使用率がこのパーセンテージを超える場合、プロビジョニングは失敗します。* Amazon FSx for ONTAP には適用されません*。</p>	"" (デフォルトでは強制されません)

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。設定する場合は、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は、FlexGroupボリュームのプロビジョニングに使用されます。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 で集計リストが更新されると、Tridentコントローラーを再起動せずに、SVM をポーリングすることによってTridentの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ためにTridentで特定の集約リストを設定した場合、集約リストの名前が変更されたり、SVM から移動されたりすると、SVM 集約をポーリングしているときに、Tridentでバックエンドが障害状態になります。バックエンドをオンラインに戻すには、集約リストをSVM 上に存在するリストに変更するか、集約リストを完全に削除する必要があります。</p>	""
limitVolumeSize	<p>要求されたボリューム サイズがこの値を超える場合、プロビジョニングは失敗します。また、qtreeの管理対象となるボリュームの最大サイズを制限し、`qtreesPerFlexvol` オプションにより、FlexVol volumeあたりの qtree の最大数をカスタマイズできます。</p>	"" (デフォルトでは強制されません)
debugTraceFlags	<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} 使用しないでください `debugTraceFlags` ただし、トラブルシューティングを行っており、詳細なログ ダンプが必要な場合を除きます。</p>	ヌル
nasType	<p>NFS または SMB ボリュームの作成を構成します。オプションは nfs、`smb` または null。null に設定すると、デフォルトで NFS ボリュームになります。</p>	nfs
nfsMountOptions	<p>NFS マウント オプションのコンマ区切りリスト。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 クラスでマウント 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で指定されたマウント オプションを使用します。ストレージ クラスまたは構成ファイルにマウント 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>	""
qtreesPerFlexvol	<p>FlexVolあたりの最大Qtree数は、[50, 300]の範囲でなければなりません</p>	「200」

パラメータ	説明	デフォルト
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 がSMB 共有を作成できるようにする名前、またはボリュームへの共通共有アクセスを防止するためにパラメータを空白のままにしておくことができます。このパラメータは、オンプレミスのONTAPではオプションです。このパラメータはAmazon FSx for ONTAPバックエンドに必須であり、空白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ルパラメータ。useREST`に設定すると `true、Trident はONTAP REST APIを使用してバックエンドと通信します。false Trident は、バックエンドとの通信にONTAPI (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。さらに、使用するONTAPログインロールには、`ontapi` 応用。これは、事前に定義された `vsadmin` として `cluster-admin` 役割。Trident 24.06リリースおよびONTAP 9.15.1以降では、`useREST` 設定されている `true` デフォルト; 変更 `useREST` に `false` ONTAPI (ZAPI) 呼び出しを使用します。	true`ONTAP 9.15.1以降の場合、それ以外の場合 `false。
limitVolumePoolSize	ontap-nas-economy 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。	"" (デフォルトでは強制されません)
denyNewVolumePools	制限 `ontap-nas-economy` バックエンドが Qtree を格納するための新しいFlexVolボリュームを作成できないようにします。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。	
adAdminUser	SMB 共有へのフルアクセス権を持つ Active Directory 管理者ユーザーまたはユーザーグループ。このパラメータを使用して、SMB 共有への完全な制御権限を持つ管理者権限を付与します。	

ボリュームのプロビジョニングのためのバックエンド構成オプション

デフォルトのプロビジョニングは、以下のオプションを使用して制御できます。`defaults` 構成のセクション。例については、以下の構成例を参照してください。

パラメータ	説明	デフォルト
spaceAllocation	Qtreeのスペース割り当て	"真実"
spaceReserve	スペース予約モード。「なし」（薄い）または「ボリューム」（厚い）	"なし"
snapshotPolicy	使用するスナップショットポリシー	"なし"

パラメータ	説明	デフォルト
qosPolicy	作成され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	作成されたボリュームに割り当てるアダプティブ 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。ontap-nas-economy ではサポートされていません。	""
snapshotReserved	スナップショット用に予約されているボリュームの割合	「0」の場合 `snapshotPolicy` は「なし」、それ以外の場合は「」
splitOnClone	クローン作成時に親からクローンを分割する	"間違い"
encryption	新しいボリュームで NetApp ボリューム暗号化 (NVE) を有効にします。デフォルトは false。このオプションを使用するには、NVE のライセンスを取得し、クラスターで有効にする必要があります。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、以下を参照してください。 "Trident が NVE および NAE と連携する仕組み" 。	"間違い"
tieringPolicy	「なし」を使用する階層化ポリシー	
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)
snapshotDir	アクセスを制御します `snapshot` ディレクトリ	NFSv4 の場合は「true」、NFSv3 の場合は「false」
exportPolicy	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スタイル。NFS サポート `mixed` として `unix` セキュリティスタイル。SMB サポート `mixed` として `ntfs` セキュリティスタイル。	NFS のデフォルトは unix。SMB のデフォルトは 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構成要素に個別に適用されるようにする必要があります。共有 QoS ポリシーグループは、すべてのワークロードの合計スループットの上限を適用します。

ボリュームプロビジョニングの例

デフォルトを定義した例を次に示します。

```

---
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"

```

のために `ontap-nas``そして ``ontap-nas-flexgroups`Trident`、新しい計算を使用し、`SnapshotReserve` のパーセンテージと `PVC` に合わせて `FlexVol` のサイズが適切に設定されるようになりました。ユーザーが `PVC` を要求すると、`Trident` は新しい計算方法を用いて、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、ユーザーは `PVC` で要求した書き込み可能なスペースを確実に受け取り、要求したスペースよりも少ないスペースを受け取ることはありません。v21.07より前のバージョンでは、ユーザーが `SnapshotReserve` を50%に設定して `PVC` (例えば5GiB) を要求した場合、書き込み可能なスペースは2.5GiBしか得られませんでした。これは、ユーザーが要求したのは全巻であり、``snapshotReserve`` それはそのパーセンテージです。 `Trident 21.07` では、ユーザーが要求するのは書き込み可能なスペースであり、`Trident` はそれを定義します。 ``snapshotReserve`` 全体の量の割合として数値を表示します。これは適用されません ``ontap-nas-economy`。これがどのように機能するかを確認するには、次の例を参照してください

計算は次のようになります。

```

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

```

`snapshotReserve = 50%`、`PVC` リクエスト = 5 GiB の場合、ボリュームの合計サイズは $5/0.5 = 10$ GiB となり、使用可能なサイズはユーザーが `PVC` リクエストで要求した 5 GiB になりますその ``volume show`` コマンドを実行すると、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_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は、Tridentをアップグレードする際に、上記のようにボリュームをプロビジョニングします。アップグレード前に作成したボリュームについては、変更を反映させるためにボリュームのサイズを変更する必要があります。例えば、2GiBのPVCで `snapshotReserve=50` 以前は、1 GiB 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。例えば、ボリュームを3GiBにサイズ変更すると、6GiBのボリュームで3GiBの書き込み可能領域がアプリケーションに提供されます。

最小限の構成例

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構成を示しています。これはバックエンドを定義する最も簡単な方法です。



Tridentを搭載したNetApp ONTAPでAmazon FSx を使用している場合は、LIF に IP アドレスではなく DNS 名を指定することをお勧めします。

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

バックエンドを設定することで、スイッチオーバーとスイッチバック後にバックエンド定義を手動で更新する必要がなくなります。["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行うには、SVMを次のように指定します。``managementLIF``そして省略する ``dataLIF``そして ``svm``パラメータ。例えば：

```
---  
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

これは最小限のバックエンド構成の例です。clientCertificate、clientPrivateKey、そしてtrustedCACertificate（信頼できるCAを使用する場合はオプション）が入力されます。`backend.json`クライアント証明書、秘密キー、信頼できるCA証明書のbase64エンコードされた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

この例では、動的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ます。これは、`ontap-nas-economy`そして`ontap-nas-flexgroup`ドライバー。

```
---
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 `managementLIF` IPv6 アドレスを使用します。

```
---  
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

その `smbShare` SMB ボリュームを使用する FSx for ONTAPにはこのパラメータが必要です。

```
---  
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}}_{{.vo\
      lume.RequestName}}"
  labels:
    cluster: ClusterA
    PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

仮想プールを備えたバックエンドの例

以下に示すサンプルのバックエンド定義ファイルでは、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。`spaceReserve` どれも、`spaceAllocation` 偽で、そして `encryption` 偽です。仮想プールはストレージ セクションで定義されます。

Trident は、「コメント」フィールドにプロビジョニング ラベルを設定します。FlexVol にコメントが設定されている ontap-nas` または FlexGroup `ontap-nas-flexgroup。Trident は、プロビジョニング時に仮想プールに存在するすべてのラベルをストレージ ボリュームにコピーします。便宜上、ストレージ管理者は仮想プールごとにラベルを定義し、ラベルごとにボリュームをグループ化できます。

これらの例では、一部のストレージプールは独自の spaceReserve、spaceAllocation、そして `encryption` 値があり、一部のプールはデフォルト値を上書きします。

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: mysqlldb
  cost: "25"
  zone: us_east_1d
  defaults:
    spaceReserve: volume
    encryption: "false"
    unixPermissions: "0775"
```

```
---
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にマッピングする

以下のStorageClass定義は、[\[仮想プールを備えたバックエンドの例\]](#)。使用して `parameters.selector` フィールドでは、各 StorageClass はボリュームをホストするため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で定義された側面が設定されます。

- その `protection-gold` ストレージクラスは、`ontap-nas-flexgroup` バックエンド。これらは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"
```

- その `protection-not-gold` ストレージクラスは、`ontap-nas-flexgroup` 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"
```

- その `app-mysqldb` ストレージクラスは、`ontap-nas` バックエンド。これは、mysqldb タイプの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"
```

- The `protection-silver-creditpoints-20k` ストレージクラスは、`ontap-nas-flexgroup` バックエンド。これは、シルバー レベルの保護と 20,000 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"
```

- その `creditpoints-5k` ストレージクラスは、`ontap-nas` バックエンドと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で提供される唯一のプールです。

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

Trident はどの仮想プールが選択されるか決定し、ストレージ要件が満たされていることを確認します。

アップデート `dataLIF` 初期設定後

次のコマンドを実行して、更新された dataLIF を含む新しいバックエンド JSON ファイルを提供することにより、初期構成後に dataLIF を変更できます。

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



PVC が 1 つまたは複数のポッドに接続されている場合、新しい dataLIF を有効にするには、対応するすべてのポッドを停止してから再度起動する必要があります。

セキュアな中小企業の例

ontap-nas ドライバーを使用したバックエンド構成

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

ontap-nas-economy ドライバーを使用したバックエンド構成

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

ストレージプールを使用したバックエンド構成

```

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret

```

ontap-nas ドライバーを使用したストレージクラスの例

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate

```



必ず追加してください `annotations` 安全な SMB を有効にします。バックエンドまたは PVC で設定された構成に関係なく、アノテーションがないとセキュア SMB は機能しません。

ontap-nas-economy ドライバーを使用したストレージクラスの例

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate

```

単一の AD ユーザーによる PVC の例

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc

```

複数の AD ユーザーによる PVC の例

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAPでTridentを使用する

"Amazon FSx for NetApp ONTAP"は、NetApp ONTAPストレージオペレーティングシステムを搭載したファイルシステムを起動および実行できるようにする、完全に管理されたAWSサービスです。FSx for ONTAPを使用すると、使い慣れたNetAppの機能、パフォーマンス、管理機能を活用しながら、AWSにデータを保存するシンプルさ、俊敏性、セキュリティ、スケーラビリティを活用できます。FSx for ONTAPは、ONTAPファイルシステム機能と管理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と統合して、Amazon Elastic Kubernetes Service (EKS) で実行されているKubernetesクラスターがONTAPによってサポートされ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することができます。

ファイルシステムは、オンプレミスのONTAPクラスターに類似した、Amazon FSxの主要なリソースです。各SVM内に、ファイルシステム内のファイルとフォルダーを保存するデータコンテナであるボリューム

を1つまたは複数のボリュームを作成できます。Amazon FSx for NetApp ONTAP はクラウド内のマネージド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タイプは * NetApp ONTAP* と呼ばれます。

Trident を Amazon FSx for NetApp ONTAP と併用することで、Amazon Elastic Kubernetes Service (EKS) で実行されている Kubernetes クラスターが ONTAP によってサポートされ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

に加えて ["Tridentの要件"](#) FSx for ONTAP を Trident と統合するには、次のものがが必要です。

- 既存の Amazon EKS クラスターまたはセルフマネージド Kubernetes クラスターで `kubectl` インストールされました。
- クラスターのワーカーノードからアクセスできる既存の Amazon FSx for NetApp ONTAP ファイルシステムとストレージ仮想マシン (SVM)。
- 準備されたワーカーノード **"NFS または iSCSI"**。



Amazon Linux および Ubuntu に必要なノード準備手順に従ってください。 ["Amazon マシンイメージ"](#) (AMI) は EKS AMI タイプによって異なります。

考慮事項

- SMB ボリューム:
 - SMB ボリュームは、`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ていません。
 - Trident 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 ボリュームのみをサポートします。参照 ["SMB ボリュームのプロビジョニングの準備"](#) 詳細については。
- Trident 24.02 より前では、自動バックアップが有効になっている Amazon FSx ファイルシステムで作成されたボリュームは、Trident では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、`fsxFilesystemID`、`AWS apiRegion`、`AWS apiKey`、`AWS secretKey` AWS FSx for ONTAP のバックエンド構成ファイル内。



Trident に IAM ロールを指定する場合は、`apiRegion`、`apiKey`、そして `secretKey` フィールドを Trident に明示的に渡します。詳細については、["FSx for ONTAP の構成オプションと例"](#)。

Trident SAN/iSCSI と EBS-CSI ドライバーの同時使用

AWS (EKS、ROSA、EC2、またはその他のインスタンス) で `ontap-san` ドライバー (iSCSI など) を使用する予定の場合、ノードに必要なマルチパス構成が Amazon Elastic Block Store (EBS) CSI ドライバーと競合する可能性があります。同じノード上の EBS ディスクに干渉せずにマルチパスが機能することを保証するには、マルチパス設定で EBS を除外する必要があります。この例では、`multipath.conf` EBS ディスクをマルチパスから除外しながら必要な Trident 設定を含むファイル:

```

defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
}

```

認証

Trident は2 つの認証モードを提供します。

- 認証情報ベース (推奨): 認証情報を AWS Secrets Manager に安全に保存します。使用することができます `fsxadmin` ファイルシステムのユーザーまたは `vsadmin` SVM 用に構成されたユーザー。



Tridentは、`vsadmin` SVM ユーザーとして、または同じロールを持つ別の名前のユーザーとして、Amazon FSx for NetApp ONTAPには `fsxadmin` ONTAPの限定的な代替品であるユーザー `admin` クラスター ユーザー。強くお勧めします `vsadmin` Trident付き。

- 証明書ベース: Trident は、SVM にインストールされた証明書を使用して、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、ドライバー タイプの認証を参照してください。

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

テスト済みの Amazon マシンイメージ (AMI)

EKS クラスターはさまざまなオペレーティングシステムをサポートしていますが、AWS は特定の Amazon マシンイメージ (AMI) をコンテナと EKS 用に最適化しています。次の AMI はNetApp Trident 25.02 でテストされています。

アミ	NAS	NAS-エコノミー	iSCSI	iSCSIエコノミー
AL2023_x86_64_標準	はい	はい	はい	はい
AL2_x86_64	はい	はい	○*	○*
ボトルロケット_x86_64	はい**	はい	該当なし	該当なし
AL2023_ARM_64_標準	はい	はい	はい	はい
AL2_ARM_64	はい	はい	○*	○*

ボトルロケットアー ム64	はい**	はい	該当なし	該当なし
------------------	------	----	------	------

- * ノードを再起動せずにPVを削除することはできません
- ** Tridentバージョン 25.02 の NFSv3 では動作しません。



必要な AMI がここにリストされていない場合、それはサポートされていないという意味ではなく、単にテストされていないという意味です。このリストは、動作することがわかっている AMI のガイドとして機能します。

テストに使用したデバイス:

- EKS version: 1.32
- インストール方法: Helm 25.06 および AWS アドオン 25.06
- NAS については、NFSv3 と NFSv4.1 の両方がテストされました。
- SAN の場合、NVMe-oF ではなく iSCSI のみがテストされました。

実行されたテスト:

- 作成: ストレージクラス、PVC、ポッド
- 削除: ポッド、PVC (通常、qtree/lun - エコノミー、AWS バックアップ付き NAS)

詳細情報の参照

- ["Amazon FSx for NetApp ONTAPドキュメント"](#)
- ["Amazon FSx for NetApp ONTAPに関するブログ投稿"](#)

IAMロールとAWSシークレットを作成する

明示的な AWS 認証情報を提供する代わりに、AWS IAM ロールとして認証することで、Kubernetes ポッドが AWS リソースにアクセスするように設定できます。



AWS IAM ロールを使用して認証するには、EKS を使用して Kubernetes クラスタをデプロイする必要があります。

AWS Secrets Managerシークレットを作成する

Trident はストレージを管理するために FSx vserver に対して API を発行するため、そのためには資格情報が必要になります。これらの認証情報を渡す安全な方法は、AWS Secrets Manager シークレットを使用することです。したがって、まだお持ちでない場合は、vsadmin アカウントの認証情報を含む AWS Secrets Manager シークレットを作成する必要があります。

この例では、Trident CSI 認証情報を保存するための AWS Secrets Manager 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ポリシーを作成する

Trident を正しく実行するには AWS 権限も必要です。したがって、Trident に必要な権限を与えるポリシーを作成する必要があります。

次の例では、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
Secrets manager"
```

ポリシー **JSON** の例:

```

{
  "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"
}

```

サービス アカウントの関連付け (IRSA) 用の **Pod Identity** または **IAM** ロールを作成する

EKS Pod Identity を持つ AWS Identity and Access Management (IAM) ロール、またはサービスアカウントの関連付け (IRSA) の IAM ロールを引き受けるように Kubernetes サービスアカウントを設定できます。サービスアカウントを使用するように設定されたすべてのポッドは、ロールがアクセス権限を持つすべての AWS サービスにアクセスできるようになります。

ポッドのアイデンティティ

Amazon EKS ポッドアイデンティティの関連付けは、Amazon EC2 インスタンスプロファイルが Amazon EC2 インスタンスに認証情報を提供するのと同様に、アプリケーションの認証情報を管理する機能を提供します。

EKS クラスターに **Pod Identity** をインストールします:

AWS コンソールまたは次の AWS CLI コマンドを使用して、Pod ID を作成できます。

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

詳細については、"[Amazon EKS ポッドアイデンティティエージェントを設定する](#)"。

trust-relationship.json を作成:

EKS サービス プリンシパルが Pod Identity に対してこのロールを引き受けることができるように、trust-relationship.json を作成します。次に、次の信頼ポリシーを持つロールを作成します。

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

trust-relationship.json ファイル:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

IAM ロールにロールポリシーをアタッチします:

前の手順のロールポリシーを、作成した IAM ロールにアタッチします。

```
aws iam attach-role-policy \  
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \  
  --role-name fsxn-csi-role
```

ポッド ID の関連付けを作成します:

IAM ロールとTridentサービス アカウント (trident-controller) の間にポッド ID の関連付けを作成します。

```
aws eks create-pod-identity-association \  
  --cluster-name <EKS_CLUSTER_NAME> \  
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \  
  --namespace trident --service-account trident-controller
```

サービス アカウントの関連付け (IRSA) の IAM ロール

AWS CLI の使用:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \  
  --assume-role-policy-document file://trust-relationship.json
```

trust-relationship.json ファイル:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

次の値を更新します `trust-relationship.json` ファイル：

- **<account_id>** - AWS アカウントID
- **<oidc_provider>** - EKS クラスターの OIDC。oidc_provider は次のコマンドを実行して取得できません。

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer" \
  --output text | sed -e "s/^https:\\/\\/\\/"
```

IAM ロールを **IAM** ポリシーにアタッチします：

ロールが作成されたら、次のコマンドを使用して、(上記の手順で作成した) ポリシーをロールにアタッチします。

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

OIDC プロバイダーが関連付けられていることを確認します：

OIDC プロバイダーがクラスターに関連付けられていることを確認します。次のコマンドを使用して確認できます。

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

出力が空の場合は、次のコマンドを使用して IAM OIDC をクラスターに関連付けます。

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name  
--approve
```

eksctl を使用している場合、次の例を使用して EKS のサービス アカウントの 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 <IAM-Policy ARN> --approve
```

Tridentをインストールする

Trident は、Kubernetes での Amazon FSx for NetApp ONTAP ストレージ管理を合理化し、開発者と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。

次のいずれかの方法で Trident をインストールできます。

- 舵
- EKS アドオン

スナップショット機能を利用する場合は、CSI スナップショット コントローラー アドオンをインストールします。参照 ["CSI ボリュームのスナップショット機能を有効にする"](#) 詳細についてはこちらをご覧ください。

Helm 経由で Trident をインストールする

ポッドのアイデンティティ

1. Trident Helm リポジトリを追加します。

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 次の例を使用してTrident をインストールします。

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

使用することができます `helm list` 名前、名前空間、チャート、ステータス、アプリのバージョン、リビジョン番号などのインストールの詳細を確認するコマンド。

```
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-
100.2502.0	25.02.0		

サービス アカウント アソシエーション (IRSA)

1. Trident Helm リポジトリを追加します。

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. クラウド プロバイダー と クラウド ID の値を設定します。

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 \ --set cloudProvider="AWS" \ --set cloudIdentity="'eks.amazonaws.com/role-arn: arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>' " \ --namespace trident \ --create-namespace
```

使用することができます `helm list` 名前、名前空間、チャート、ステータス、アプリのバージョン、リビジョン番号などのインストールの詳細を確認するコマンド。

```
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-
100.2506.0	25.06.0		

iSCSI を使用する予定の場合は、クライアント マシンで iSCSI が有効になっていることを確認してください。AL2023 ワーカー ノード OS を使用している場合は、helm インストールで node prep パラメータを追加することで、iSCSI クライアントのインストールを自動化できます。



```
helm install trident-operator netapp-trident/trident-operator  
--version 100.2502.1 --namespace trident --create-namespace --  
set nodePrep={iscsi}
```

EKS アドオン経由で Trident をインストールする

Trident EKS アドオンには最新のセキュリティパッチとバグ修正が含まれており、Amazon EKS で動作することが AWS によって検証されています。EKS アドオンを使用すると、Amazon EKS クラスターの安全性と安定性を常に確保し、アドオンのインストール、設定、更新に必要な作業量を削減できます。

前提条件

AWS EKS の Trident アドオンを設定する前に、以下があ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付きの Amazon EKS クラスターアカウント
- AWS マーケットプレイスへの 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

管理コンソール

1. 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、[クラスター] を選択します。
3. NetApp Trident CSI アドオンを構成するクラスターの名前を選択します。
4. *アドオン*を選択し、*さらにアドオンを取得*を選択します。
5. アドオンを選択するには、次の手順に従います。
 - a. **AWS Marketplace** アドオン セクションまで下にスクロールし、検索ボックスに「* Trident」* と入力します。
 - b. Trident by NetAppボックスの右上隅にあるチェック ボックスを選択します。
 - c. *次へ*を選択します。
6. *選択したアドオンの構成*設定ページで、次の操作を行います。



Pod Identity 関連付けを使用している場合は、これらの手順をスキップしてください。

- a. 使用したい*バージョン*を選択してください。
- b. IRSA 認証を使用している場合は、オプション構成設定で使用可能な構成値を必ず設定してください。
 - 使用したい*バージョン*を選択してください。
 - アドオン構成スキーマ に従い、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 (値は次の形式である必要があります)。

```
{  
  
  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",  
  "cloudProvider": "AWS"  
  
}
```

+

競合解決方法として [オーバーライド] を選択した場合、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せず、既存の設定と競合する場合、操作は失敗します。結果のエラー 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する必要がある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. *確認と追加*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、インストールされたアドオンが表示されます。

AWS CLI

1.作成する `add-on.json` ファイル:

Pod Identity の場合は、次の形式を使用します。

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

IRSA 認証の場合は、次の形式を使用します:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```



交換する `<role ARN>` 前の手順で作成されたロールの ARN を使用します。

2.Trident EKS アドオンをインストールします。

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

eksctl

次のコマンド例は、Trident EKS アドオンをインストールします。

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Trident EKS アドオンを更新する

管理コンソール

1. Amazon EKS コンソールを開く <https://console.aws.amazon.com/eks/home#/clusters>。
2. 左側のナビゲーション ペインで、[クラスター] を選択します。
3. NetApp Trident CSI アドオンを更新するクラスターの名前を選択します。
4. *アドオン*タブを選択します。
5. *Trident by NetApp*を選択し、*編集*を選択します。
6. *Trident by NetAppの構成 *ページで、次の操作を行います。
 - a. 使用したい*バージョン*を選択してください。
 - b. *オプションの構成設定*を展開し、必要に応じて変更します。
 - c. *変更を保存*を選択します。

AWS CLI

次の例では、EKS アドオンを更新します。

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
  \"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- FSxN Trident CSI アドオンの現在のバージョンを確認します。交換する `my-cluster` クラスター名に置き換えます。

```
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	v25.6.0-eksbuild.1	ACTIVE	0
{"cloudIdentity":"'eks.amazonaws.com/role-arn: arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}			

- 前の手順の出力の 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
v25.6.0-eksbuild.1 --cluster my-cluster --force
```

削除すると `--force` オプションと Amazon EKS アドオン設定のいずれかが既存の設定と競合する場合、Amazon EKS アドオンの更新は失敗し、競合を解決するための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、Amazon EKS アドオンが管理する必要がある設定を管理していないことを確認してください。このオプションによってそれらの設定が上書きされるためです。この設定の他のオプションの詳細については、"[アドオン](#)"。Amazon EKS Kubernetes フィールド管理の詳細については、以下を参照してください。"[Kubernetes フィールド管理](#)"。

Trident EKS アドオンをアンインストール/削除する

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。

- クラスター上のアドオンソフトウェアを保持する – このオプションを選択すると、Amazon EKS によるすべての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、アドオンソフトウェアはクラスター上に保持されます。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。保持する `--preserve` アドオンを保持するには、コマンドにオプションを追加します。
- クラスターからアドオンソフトウェアを完全に削除する – NetApp、クラスター上に Amazon EKS アドオンに依存するリソースがない場合にのみ、クラスターから Amazon EKS アドオンを削除することをお勧めします。削除する `--preserve` オプションから `delete` アドオンを削除するコマンド。



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

管理コンソール

1. 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、[クラスター] を選択します。
3. NetApp Trident CSI アドオンを削除するクラスターの名前を選択します。
4. アドオン*タブを選択し、Trident by NetApp*を選択します。*
5. *削除*を選択します。
6. **netapp_trident-operator** の削除確認 ダイアログで、次の操作を行います。
 - a. Amazon EKS によるアドオン設定の管理を停止する場合は、[クラスターで保持] を選択します。アドオン ソフトウェアをクラスター上に保持して、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、これを実行してください。
 - b. **netapp_trident-operator** と入力します。
 - c. *削除*を選択します。

AWS CLI

交換する `my-cluster` クラスターの名前に変更し、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
```

eksctl

次のコマンドは、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
```

ストレージバックエンドを構成する

ONTAP SANおよびNASドライバーの統合

ストレージ バックエンドを作成するには、JSON または YAML 形式で構成ファイルを作成する必要があります。ファイルでは、必要なストレージのタイプ (NAS または SAN)、ストレージの取得元となるファイル システム、SVM、および認証方法を指定する必要があります。次の例は、NAS ベースのストレージを定義し、AWS シークレットを使用して使用する SVM への認証情報を保存する方法を示しています。

ヤムル

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
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"
    "namespace": "trident"
  },
  "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"
    }
  }
}
```

次のコマンドを実行して、Tridentバックエンド構成 (TBC) を作成し、検証します。

- yaml ファイルから trident バックエンド構成 (TBC) を作成し、次のコマンドを実行します。

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Trident バックエンド構成 (TBC) が正常に作成されたことを確認します。

```
Kubectl get tbc -n trident
```

NAME	BACKEND NAME	BACKEND UUID
PHASE	STATUS	
backend-tbc-ontap-nas	tbc-ontap-nas	933e0071-66ce-4324-
b9ff-f96d916ac5e9	Bound	Success

FSx for ONTAP ドライバーの詳細

次のドライバーを使用して、Trident を Amazon FSx for NetApp ONTAP と統合できます。

- `ontap-san`: プロビジョニングされた各 PV は、独自の 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 および SMB に推奨されます。
- `ontap-san-economy`: プロビジョニングされる各 PV は、Amazon FSx for NetApp ONTAP ボリュームごとに設定可能な数の LUN を持つ LUN です。
- `ontap-nas-economy`: プロビジョニングされる各 PV は qtree であり、Amazon FSx for NetApp ONTAP ボリューム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る各 PV は、完全な Amazon FSx for NetApp ONTAP FlexGroup ボリュームです。

ドライバーの詳細については、"[NAS ドライバー](#)"そして"[SAN ドライバー](#)"。

設定ファイルが作成されたら、次のコマンドを実行して EKS 内に作成します。

```
kubect1 create -f configuration_file
```

ステータスを確認するには、次のコマンドを実行します。

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

NAME	BACKEND NAME	BACKEND UUID
PHASE STATUS		
backend-fsx-ontap-nas f2f4c87fa629 Bound	backend-fsx-ontap-nas Success	7a551921-997c-4c37-a1d1-

バックエンドの高度な構成と例

バックエンドの構成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
version		常に1
storageDriverName	ストレージ ドライバーの名前	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	ドライバー名 + "_" + dataLIF
managementLIF	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。 Trident が IPv6 フラグを使用して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] のように角括弧で定義する必要があります。あなたが提供した `fsxFilesystemID` の下で `aws` フィールドに入力する必要はありません `managementLIF` TridentはSVMを取得するため `managementLIF` AWS からの情報。そのため、SVM下のユーザーの資格情報（例：vsadmin）を提供する必要があり、ユーザーは `vsadmin` 役割。	"10.0.0.1"、 "[2001:1234:abcd::fefe]"

パラメータ	説明	例
dataLIF	<p>プロトコル LIF の IP アドレス。 * ONTAP NAS ドライバー*: NetApp、dataLIF を指定することを推奨しています。指定されない場合、Trident は SVM から dataLIF を取得します。NFS マウント操作に使用する完全修飾ドメイン名 (FQDN) を指定できるため、複数のデータ LIF 間で負荷分散を行うラウンドロビン DNS を作成できます。初期設定後も変更可能です。参照。 * ONTAP SAN ドライバー*: iSCSI の場合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を確立するために必要な iSCSI LIF を検出します。dataLIF が明示的に定義されている場合、警告が生成されます。Trident が IPv6 フラグを使用して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] のように角括弧で定義する必要があります。</p>	
autoExportPolicy	<p>自動エクスポート ポリシーの作成と更新を有効にします [ブール値]。使用して `autoExportPolicy` そして `autoExportCIDRs` オプションを使用すると、Trident はエクスポートポリシーを自動的に管理できます。</p>	false
autoExportCIDRs	<p>Kubernetes のノード IP をフィルタリングするための CIDR のリスト `autoExportPolicy` が有効になります。使用して `autoExportPolicy` そして `autoExportCIDRs` オプションを使用すると、Trident はエクスポートポリシーを自動的に管理できます。</p>	"["0.0.0.0/0", ":::/0"]"
labels	<p>ボリュームに適用する任意の JSON 形式のラベルのセット</p>	""
clientCertificate	<p>クライアント証明書の Base64 エンコードされた値。証明書ベースの認証に使用</p>	""
clientPrivateKey	<p>クライアント秘密キーの Base64 エンコードされた値。証明書ベースの認証に使用</p>	""

パラメータ	説明	例
trustedCACertificate	信頼された CA 証明書の Base64 エンコードされた値。オプション。証明書ベースの認証に使用されません。	""
username	クラスタまたは SVM に接続するためのユーザー名。資格情報ベースの認証に使用されます。たとえば、vsadmin。	
password	クラスターまたは SVM に接続するためのパスワード。資格情報ベースの認証に使用されます。	
svm	使用するストレージ仮想マシン	SVM 管理 LIF が指定されている場合に派生されます。
storagePrefix	SVM で新しいボリュームをプロビジョニングするときに使用されるプレフィックス。作成後は変更できません。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
limitAggregateUsage	* Amazon FSx for NetApp ONTAP には指定しないでください。*提供された `fsxadmin` そして `vsadmin` 集計使用量を取得し、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	要求されたボリュームサイズがこの値を超える場合、プロビジョニングは失敗します。また、qtree と LUN を管理するボリュームの最大サイズを制限し、`qtreesPerFlexvol` オプションにより、FlexVol volume あたりの qtree の最大数をカスタマイズできます。	"" (デフォルトでは強制されません)
lunsPerFlexvol	Flexvol ボリュームあたりの最大 LUN 数は、[50、200] の範囲にする必要があります。SAN のみ。	「100」
debugTraceFlags	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> 使用しないでください `debugTraceFlags` ただし、トラブルシューティングを行っており、詳細なログ ダンプが必要な場合を除きます。	ヌル

パラメータ	説明	例
nfsMountOptions	NFS マウント オプションのコンマ区切りリスト。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 クラスでマウント 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で指定されたマウント オプションを使用します。ストレージ クラスまたは構成ファイルにマウント 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 または SMB ボリュームの作成を構成します。オプションは nfs、smb、または null。設定する必要があります `smb`SMB ボリュームの場合。null に設定すると、デフォルトで NFS ボリュームになります。	nfs
qtreesPerFlexvol	FlexVol volumeあたりの最大Qtree 数は、[50, 300]の範囲でなければなりません	"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 がSMB 共有を作成できるようにするための名前。このパラメータは、Amazon FSx for ONTAPバックエンドに必須です。	smb-share
useREST	ONTAP REST API を使用するためのブール パラメーター。に設定すると `true`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。さらに、使用するONTAPログインロールには、`ontap`応用。これは、事前に定義された `vsadmin`そして `cluster-admin`役割。	false

パラメータ	説明	例
aws	AWS FSx for ONTAPの設定ファイルでは以下を指定できます。 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 秘密キー。	"" "" ""
credentials	AWS Secrets Manager に保存する FSx SVM 認証情報を指定します。 - name: SVM の認証情報が含まれるシークレットの Amazon リソースネーム (ARN)。 - type: に設定 awsarn。参照" AWS Secrets Managerシークレットを作成する " 詳細についてはこちらをご覧ください。	

ボリュームのプロビジョニングのためのバックエンド構成オプション

デフォルトのプロビジョニングは、以下のオプションを使用して制御できます。`defaults`構成のセクション。例については、以下の構成例を参照してください。

パラメータ	説明	デフォルト
spaceAllocation	LUNのスペース割り当て	true
spaceReserve	スペース予約モード。「なし」(薄い) または「ボリューム」(厚い)	none
snapshotPolicy	使用するスナップショットポリシー	none
qosPolicy	作成され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構成要素に個別に適用されるようにする必要があります。共有 QoS ポリシーグループは、すべてのワークロードの合計スループットの上限を適用します。	""

パラメータ	説明	デフォルト
adaptiveQosPolicy	作成されたボリュームに割り当てるアダプティブ 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。ontap-nas-economy ではサポートされていません。	""
snapshotReserve	スナップショット用に予約されているボリュームの割合「0」	もし snapshotPolicy`は`none、else「」
splitOnClone	クローン作成時に親からクローンを分割する	false
encryption	新しいボリュームでNetAppボリューム暗号化 (NVE) を有効にします。デフォルトは false。このオプションを使用するには、NVE のライセンスを取得し、クラスターで有効にする必要があります。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、以下を参照してください。" TridentがNVEおよびNAEと連携する仕組み "。	false
luksEncryption	LUKS 暗号化を有効にします。参照" Linux Unified Key Setup (LUKS)を使用する "。SANのみ。	""
tieringPolicy	使用する階層化ポリシー none	
unixPermissions	新しいボリュームのモード。SMB ボリュームの場合は空白のままにします。	""
securityStyle	新しいボリュームのセキュリティスタイル。NFSサポート`mixed`そして`unix`セキュリティスタイル。SMBサポート`mixed`そして`ntfs`セキュリティスタイル。	NFSのデフォルトは unix。SMBのデフォルトは ntfs。

SMBボリュームのプロビジョニングの準備

SMBボリュームをプロビジョニングするには、`ontap-nas`ドライバ。完了する前に[ONTAP SANおよびNASドライバーの統合次](#)の手順を実行します。

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`ドライバーには次のものがが必要です。

- Linux コントローラー ノードと、Windows Server 2019 を実行する少なくとも 1 つの Windows ワーカー ノードを備えた Kubernetes クラスター。Trident 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 ボリュームのみをサポートします。

- Active Directory 資格情報を含む少なくとも 1 つの Trident シークレット。秘密を生成する smbcreds:

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

- Windows サービスとして構成された CSI プロキシ。設定するには `csi-proxy`、参照["GitHub: CSI プロキシ"](#)または["GitHub: Windows 用 CSI プロキシ"](#) Windows 上で実行されている Kubernetes ノード用。

手順

1. SMB 共有を作成します。SMB 管理共有は、次の 2 つの方法のいずれかで作成できます。["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。

その `vserver cifs share create` コマンドは、共有の作成中に `-path` オプションで指定されたパスをチェックします。指定されたパスが存在しない場合、コマンドは失敗します。

- b. 指定された SVM に関連付けられた 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
```



参照["SMB共有を作成する"](#)詳細についてはこちらをご覧ください。

2. バックエンドを作成するときは、SMB ボリュームを指定するために以下を構成する必要があります。FSx for ONTAP のバックエンド構成オプションについては、以下を参照してください。["FSx for 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 が SMB 共有を作成できるようにするための名前。このパラメータは、Amazon FSx for ONTAP バックエンドに必須です。	smb-share

パラメータ	説明	例
nasType	設定する必要があります smb . null の場合、デフォルトは nfs 。	smb
securityStyle	新しいボリュームのセキュリティスタイル。設定する必要があります `ntfs` または `mixed` SMB ボリュームの場合。	`ntfs` または `mixed` SMB ボリューム用
unixPermissions	新しいボリュームのモード。 SMB ボリュームの場合は空のままにする必要があります。	""

ストレージクラスとPVCを構成する

Kubernetes StorageClass オブジェクトを構成し、ボリュームのプロビジョニング方法を Trident に指示するストレージクラスを作成します。構成された Kubernetes StorageClass を使用して PV へのアクセスを要求する 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 オブジェクトを構成する

その "**Kubernetes StorageClass オブジェクト**" オブジェクトは、そのクラスに使用されるプロビジョナーとして Trident を識別し、ボリュームをプロビジョニングする方法を Trident に指示します。NFS を使用するボリュームの Storageclass を設定するには、この例を使用します (属性の完全なリストについては、以下の Trident 属性セクションを参照してください)。

```

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

```

iSCSI を使用するボリュームの Storageclass を設定するには、次の例を使用します。

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

AWS BottlerocketでNFSv3ボリュームをプロビジョニングするには、必要な `mountOptions` ストレージクラスに:

```
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"
mountOptions:
  - nfsvers=3
  - nolock
```

参照"[KubernetesとTridentオブジェクト](#)"ストレージクラスがどのように相互作用するかの詳細については、PersistentVolumeClaim Trident がボリュームをプロビジョニングする方法を制御するためのパラメータ。

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` Kubernetes で作成します。

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

2. これで、Kubernetes とTridentの両方に **basic-csi** ストレージ クラスが表示され、Trident はバックエンドでプールを検出しているはずで。

```
kubectl get sc basic-csi
```

NAME	PROVISIONER	AGE
basic-csi	csi.trident.netapp.io	15h

PVCを作成する

あ "永続ボリュームクレーム"(PVC) は、クラスター上の 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構成できます。関連付けられた StorageClass を使用すると、クラスター管理者は PersistentVolume のサイズやアクセス モードだけでなく、パフォーマンスやサービス レベルなどの制御も行えます。

PVC を作成したら、ボリュームをポッドにマウントできます。

サンプルマニフェスト

PersistentVolumeClaim サンプルマニフェスト

これらの例は、基本的な PVC 構成オプションを示しています。

RWX アクセス付き PVC

この例では、RWXアクセスを持つ基本的なPVCが、StorageClassに関連付けられています。basic-csi。

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

iSCSI を使用した PVC の例

この例では、RWOアクセスを持つiSCSI用の基本PVCが、StorageClassに関連付けられています。protection-gold。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

PVCを作成する

手順

1. PVCを作成

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

2. 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

参照"[KubernetesとTridentオブジェクト](#)"ストレージクラスがどのように相互作用するかの詳細については、PersistentVolumeClaim Trident がボリュームをプロビジョニングする方法を制御するためのパラメータ。

Tridentの属性

これらのパラメータは、特定のタイプのボリュームをプロビジョニングするためにどの Trident 管理ストレージ プールを使用するかを決定します。

属性	タイプ	値	オファー	要求	支援
メディア ¹	string	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ません。ハイブリッドとは両方を意味します。	指定されたメディアタイプ	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san
プロビジョニングタイプ	string	薄い、厚い	プールはこのプロビジョニング方法をサポートしています	指定されたプロビジョニング方法	厚い：すべてオンタップ、薄い：すべてオンタップとsolidfireさん
バックエンドタイプ	string	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san、gcp-cvs、azure-netapp-files、ontap-san-economy	プールはこのタイプのバックエンドに属します	バックエンドが指定されました	すべてのドライバー
Snapshot	ブール	真、偽	プールはスナップショット付きのボリュームをサポートします	スナップショットが有効になっているボリューム	ontap-nas、ontap-san、solidfire-san、gcp-cvs
クローン	ブール	真、偽	プールはボリュームのクローン作成をサポート	クローンが有効なボリューム	ontap-nas、ontap-san、solidfire-san、gcp-cvs

属性	タイプ	値	オファー	要求	支援
暗号化	ブール	真、偽	プールは暗号化されたボリュームをサポートします	暗号化が有効になっているボリューム	ontap-nas、ontap-nas-economy、ontap-nas-flexgroups、ontap-san
IOPS	整数	正の整数	プールはこの範囲のIOPSを保証できる	ボリュームはこれらのIOPSを保証	solidfireさん

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をデプロイする

ストレージクラスとPVCが作成されると、PVをポッドにマウントできます。このセクションでは、PVをポッドに接続するためのコマンドと構成の例を示します。

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
```

以下の例は、PVCをポッドに接続するための基本構成を示しています。基本構成:

```
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
```



進捗状況は以下で確認できます。 `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する `/my/mount/path`。

```
kubectl exec -it 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
```

これでポッドを削除できます。 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 pv-pod
```

EKS クラスターでTrident EKS アドオンを構成する

NetApp Trident は、 Kubernetes でのAmazon FSx for NetApp ONTAPストレージ管理を合理化し、開発者と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。 NetApp Trident EKS アドオンには最新のセキュリティパッチとバグ修正が含まれており、Amazon EKS で動作することがAWSによって検証されています。 EKS アドオンを使用すると、Amazon EKS クラスターの安全性と安定性を常に確保し、アドオンのインストール、設定、更新に必要な作業量を削減できます。

前提条件

AWS EKS のTridentアドオンを設定する前に、以下があることを確認してください。

- アドオンを操作する権限を持つ Amazon EKS クラスターアカウント。参照["Amazon EKS アドオン"](#)。
- AWS マーケットプレイスへの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 ポッドがAWSリソースにアクセスできるようにするには、IAM ロールとAWSシークレットを作成してください。手順については、["IAMロールとAWSシークレットを作成する"](#)。

2. EKS Kubernetes クラスターで、[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'. A notification banner at the top states: 'End of standard support for Kubernetes version 1.30 is July 28, 2025. On that date, your cluster will enter the extended support period with additional fees. For more information, see the pricing page [3].' Below this is a 'Cluster info' section with a table:

Status Active	Kubernetes version 1.30	Support period Standard support until July 28, 2025	Provider EKS
-------------------------	-----------------------------------	---	------------------------

Below the table are 'Cluster health issues' (0) and 'Upgrade insights' (0). A navigation bar includes 'Overview', 'Resources', 'Compute', 'Networking', 'Add-ons' (selected), 'Access', 'Observability', 'Update history', and 'Tags'. A notification says 'New versions are available for 1 add-on.' Below that is an 'Add-ons (3)' section with a search bar, filters for 'Any category', 'Any status', and '3 matches'. Buttons for 'View details', 'Edit', 'Remove', and 'Get more add-ons' are present.

3. AWS Marketplace アドオン に移動し、storage カテゴリを選択します。

The screenshot shows the 'AWS Marketplace add-ons (1)' page. It features a search bar with 'Find add-on' and filtering options: 'Any category', 'NetApp, Inc.', and 'Any pricing model'. A 'Clear filters' button is also present. Below the filters, a list of add-ons is shown, with 'NetApp Trident' selected. The details for 'NetApp Trident' are as follows:

Category storage	Listed by NetApp, Inc. [3]	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 [3]
----------------------------	--------------------------------------	---	--

At the bottom right, there are 'Cancel' and 'Next' buttons.

4. * NetApp Trident* を見つけて、Trident アドオンのチェックボックスをオンにし、次へ をクリックします。

5. アドオンの希望するバージョンを選択します。

Configure selected add-ons settings

Configure the add-ons for your cluster by selecting settings.

NetApp Trident

Listed by **NetApp** | Category storage | Status Ready to install [Remove add-on](#)

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.
v25.6.0-eksbuild.1

Optional configuration settings

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

6. 必要なアドオン設定を構成します。

Review and add

Step 1: Select add-ons

Selected add-ons (1)

Find add-on < 1 >

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

Step 2: Configure selected add-ons settings

Selected add-ons version (1)

< 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)

< 1 >

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](#)

7. IRSA（サービスアカウントのIAMロール）を使用している場合は、追加の構成手順を参照してください。"ここをクリックしてください。"。

8. *作成*を選択します。

9. アドオンのステータスが **アクティブ** であることを確認します。

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

10. 次のコマンドを実行して、Tridentがクラスターに正しくインストールされていることを確認します。

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

11. セットアップを続行し、ストレージバックエンドを構成します。詳細については、"[ストレージバックエンドを構成する](#)"。

CLI を使用して**Trident EKS** アドオンをインストール/アンインストールする

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

次のコマンド例は、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 v25.6.0-eksbuild.1 (専用版あり)
```

CLI を使用して**NetApp 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、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 でバックエンドを作成する

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。これは、Tridentにそのストレージシステムと通信する方法と、そこからボリュームをプロビジョニングする方法を指示します。Tridentをインストールしたら、次のステップはバックエンドを作成することです。その`TridentBackendConfig`カスタムリソース定義(CRD)を使用すると、Kubernetesインターフェースを介してTridentバックエンドを直接作成および管理できます。これは次のように行うことができます`kubectl`または、Kubernetesディストリビューションに相当するCLIツール。

TridentBackendConfig

TridentBackendConfig(tbc、tbconfig、tbackendconfig)は、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 およびストレージ管理者は、専用のコマンドラインユーティリティを必要とせずに、Kubernetes CLI を介して直接バックエンドを作成および管理できるようになりました。 (`tridentctl`)。

の作成時に `TridentBackendConfig` オブジェクトの場合、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 `TridentBackend` (`tbe`、 `tridentbackend`) CR。
- その `TridentBackendConfig` 一意に結びついている `TridentBackend` Trident によって作成されました。

それぞれ `TridentBackendConfig` 1対1のマッピングを維持する `TridentBackend` 前者は、バックエンドを設計および構成するためにユーザーに提供されるインターフェースであり、後者は Trident が実際のバックエンドオブジェクトを表現する方法です。



`TridentBackend` CR は Trident によって自動的に作成されます。これらを変更しないでください。バックエンドを更新したい場合は、`TridentBackendConfig` 物体。

フォーマットについては次の例を参照してください。 `TridentBackendConfig` CR:

```
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
```

また、以下の例もご覧ください。 ["トライデントインストーラー"](#) 必要なストレージプラットフォーム/サービスのサンプル構成のディレクトリ。

その `spec` バックエンド固有の構成パラメータを受け取ります。この例では、バックエンドは `ontap-san` ストレージドライバーであり、ここに表されている構成パラメータを使用します。ご希望のストレージドライバー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ーのバックエンド構成情報"](#)。

その `spec` セクションには以下も含まれています `credentials` そして `deletionPolicy` 新たに導入されたフィールド `TridentBackendConfig` CR:

- `credentials`: このパラメータは必須フィールドであり、ストレージシステム/サービスでの認証に使用される資格情報が含まれます。これは、ユーザーが作成した Kubernetes シークレットに設定されます。資格情報はプレーンテキストで渡すことができず、エラーが発生します。
- `deletionPolicy`: このフィールドは、`TridentBackendConfig` 削除されます。次の 2 つの値のいずれかを取ることができます。

- delete: これにより、両方の `TridentBackendConfig` CR および関連するバックエンド。これがデフォルト値です。
- retain: とき `TridentBackendConfig` CR が削除されても、バックエンドの定義はそのまま残り、`tridentctl`。削除ポリシーを設定する `retain` ユーザーは以前のリリース (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` が作成されます。



バックエンドの名前は次のように設定されます。 `spec.backendName`。指定しない場合は、バックエンドの名前は `TridentBackendConfig` オブジェクト (metadata.name)。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。 `spec.backendName`。



作成されたバックエンド `tridentctl` 関連する `TridentBackendConfig` 物体。このようなバックエンドを管理するには、 `kubectl` 作成することで `TridentBackendConfig` CR。同一の設定パラメータ (例えば `spec.backendName`、 `spec.storagePrefix`、 `spec.storageDriverName`、 等々)。Tridentは新しく作成された `TridentBackendConfig` 既存のバックエンドを使用します。

手順の概要

新しいバックエンドを作成するには `kubectl`、次の操作を行う必要があります。

1. 作成する "Kubernetes シークレット" シークレットには、Trident がストレージ クラスター/サービスと通信するために必要な資格情報が含まれています。
2. 作成する `TridentBackendConfig` 物体。これには、ストレージ クラスター/サービスに関する詳細が含まれており、前の手順で作成されたシークレットが参照されます。

バックエンドを作成したら、次のようにしてそのステータスを確認できます。 `kubectl get tbc <tbc-name> -n <trident-namespace>` 追加の詳細情報を収集します。

ステップ1: Kubernetesシークレットを作成する

バックエンドのアクセス資格情報を含むシークレットを作成します。これは各ストレージ サービス/プラットフォームに固有です。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: password
```

この表は、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たものです。

ストレージプラットフォームの秘密フィールドの説明	秘密	フィールドの説明
Azure NetApp Files	クライアントID	アプリ登録からのクライアントID
Cloud Volumes Service for GCP	秘密鍵ID	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	秘密鍵	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
エレメント (NetApp HCI/ SolidFire)	エンドポイント	テナント資格情報を持つSolidFire クラスタの MVIP
ONTAP	ユーザ名	クラスタ/SVM に接続するためのユーザー名。資格情報ベースの認証に使用される
ONTAP	パスワード	クラスター/SVM に接続するためのパスワード。資格情報ベースの認証に使用される
ONTAP	クライアント秘密鍵	クライアント秘密キーの Base64 エンコードされた値。証明書ベースの認証に使用される
ONTAP	chapユーザー名	受信ユーザー名。 useCHAP=true の場合は必須です。のために ontap-san`そして `ontap-san-economy
ONTAP	chapInitiatorSecret	CHAP イニシエーター シークレット。 useCHAP=true の場合は必須です。のために ontap-san`そして `ontap-san-economy
ONTAP	chapターゲットユーザー名	ターゲットユーザー名。 useCHAP=true の場合は必須です。のために ontap-san`そして `ontap-san-economy

ストレージプラットフォームの秘密フィールドの説明	秘密	フィールドの説明
ONTAP	chapTargetInitiatorSecret	CHAP ターゲット イニシエーターシークレット。 useCHAP=true の場合は必須です。のために ontap-san`そして `ontap-san-economy

このステップで作成されたシークレットは、`spec.credentials`の分野 `TridentBackendConfig` 次のステップで作成されるオブジェクト。

ステップ2: 作成する `TridentBackendConfig` CR

これで作成準備ができました `TridentBackendConfig` CR。この例では、`ontap-san` ドライバーは、`TridentBackendConfig` 以下に示すオブジェクト:

```
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
```

ステップ3: ステータスを確認する `TridentBackendConfig` CR

これで、`TridentBackendConfig` CR、ステータスを確認できます。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。

フェーズは次のいずれかの値を取ることができます。

- **Bound** : その `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 `configRef` に設定 `TridentBackendConfig` CR の uid。
- **Unbound**: 表現方法 ""。その `TridentBackendConfig` オブジェクトはバックエンドにバインドされていません。すべて新しく作成されました `TridentBackendConfig` CR はデフォルトでこのフェーズにあります。フェーズが変更された後は、再びアンバウンドに戻ることはできません。
- **Deleting** : その `TridentBackendConfig` CR の `deletionPolicy` 削除するように設定されました。いつ `TridentBackendConfig` CR が削除されると、削除中状態に移行します。
 - バックエンドに永続ボリュームクレーム (PVC) が存在しない場合は、`TridentBackendConfig` Trident はバックエンドと `TridentBackendConfig` CR。
 - バックエンドに 1 つ以上の PVC が存在する場合、削除状態になります。その `TridentBackendConfig` その後、CR も削除フェーズに入ります。バックエンドと `TridentBackendConfig` すべての PVC が削除された後にのみ削除されます。
- **Lost**: に関連付けられたバックエンド `TridentBackendConfig` CR が誤ってまたは故意に削除され、`TridentBackendConfig` CR には削除されたバックエンドへの参照がまだ残っています。その `TridentBackendConfig` CR は、`deletionPolicy` 値。
- **Unknown**: Trident は、関連付けられたバックエンドの状態または存在を判別できません。`TridentBackendConfig` CR。たとえば、API サーバーが応答しない場合や、`tridentbackends.trident.netapp.io` CRD がありません。これには介入が必要になる可能性があります。

この段階で、バックエンドが正常に作成されました。追加で処理できる操作がいくつかあります。"[バックエンドの更新と削除](#)"。

(オプション) ステップ4: 詳細を取得する

バックエンドの詳細情報を取得するには、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、YAML/JSON ダンプを取得することもできます。`TridentBackendConfig`。

```
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`に含まれている `backendName`そして `backendUUID`バックエンドは、`TridentBackendConfig` CR。その lastOperationStatus`フィールドは、最後の操作のステータスを表します。 `TridentBackendConfig` CRはユーザーによってトリガーされる（例えば、ユーザーが spec）またはTridentによってトリガーされます（たとえば、Trident の再起動時）。成功または失敗のいずれかになります。 phase`の関係の状態を表します `TridentBackendConfig` CR とバックエンド。上記の例では、`phase`値はBoundです。つまり、 `TridentBackendConfig` CR はバックエンドに関連付けられています。

実行することができます `kubectl -n trident describe tbc <tbc-cr-name>` イベント ログの詳細を取得するコマンド。



関連付けられたバックエンドを更新または削除することはできません。TridentBackendConfig`オブジェクト使用 `tridentctl`。切り替え手順を理解するには tridentctl`そして `TridentBackendConfig`、[こちらをご覧ください](#)。

バックエンドを管理する

kubectl を使用してバックエンド管理を実行する

バックエンド管理操作を実行する方法について学習します。 kubectl。

バックエンドを削除する

削除することで TridentBackendConfig、Trident にバックエンドを削除/保持するように指示します (deletionPolicy)。バックエンドを削除するには、deletionPolicy`削除するように設定されています。削除するには `TridentBackendConfig、必ず deletionPolicy`保持するように設定されています。これにより、バックエンドがまだ存在し、以下を使用して管理できることが保証されます。
`tridentctl。

次のコマンドを実行します。

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

Tridentは、使用されていたKubernetes Secretsを削除しません。TridentBackendConfig。Kubernetes ユーザーはシークレットをクリーンアップする責任があります。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する必要があります。

既存のバックエンドを表示する

次のコマンドを実行します。

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

実行することもできます tridentctl get backend -n trident`または `tridentctl get backend -o yaml -n trident`存在するすべてのバックエンドのリストを取得します。このリストには、
`tridentctl。

バックエンドを更新する

バックエンドを更新する理由は複数考えられます。

- ストレージシステムへの資格情報が変更されました。資格情報を更新するには、`TridentBackendConfig` オブジェクトを更新する必要があります。Trident は、提供された最新の資格情報を使用してバックエンドを自動的に更新します。Kubernetes シークレットを更新するには、次のコマンドを実行します。

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

- パラメータ (使用されている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 経由でオブジェクトを直接実行します。

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

- あるいは、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が失敗した場合、バックエンドは最後の既知の構成のままになります。ログを表示して原因を特定するには、次のコマンドを実行します。 `kubectl get tbc <tbc-name> -o yaml -n trident` または `kubectl describe tbc <tbc-name> -n trident`。
- 構成ファイルの問題を特定して修正したら、更新コマンドを再実行できます。

tridentctl でバックエンド管理を実行する

バックエンド管理操作を実行する方法について学習します。 `tridentctl`。

バックエンドを作成する

作成後"[バックエンド設定ファイル](#)"、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、バックエンドの構成に問題があります。次のコマンドを実行すると、ログを表示して原因を特定できます。

```
tridentctl logs -n trident
```

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

バックエンドを削除する

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 がこのバックエンドからプロビジョニングしたボリュームとスナップショットがまだ存在する場合、バックエンドを削除すると、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「削除中」の状態が存在し続けます。

既存のバックエンドを表示する

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
```

- すべての詳細を取得するには、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する

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は、バックエンドの構成に問題があるか、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
```

設定ファイルの問題を特定して修正したら、`update`再度コマンドを実行します。

バックエンドを使用するストレージクラスを識別する

これはJSONで答えられる質問の例です。`tridentctl`バックエンド オブジェクトの出力。これは、`jq`インストールする必要があるユーティリティ。

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

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

バックエンド管理オプション間を移動する

Tridentでバックエンドを管理するさまざまな方法について学びます。

バックエンドを管理するためのオプション

の導入により `TridentBackendConfig` 管理者は、バックエンドを管理する 2 つの独自の方法を利用できるようになりました。これにより、次の疑問が生じます。

- バックエンドは以下を使用して作成できます `tridentctl`` 管理される `TridentBackendConfig`?
- バックエンドは以下を使用して作成できます `TridentBackendConfig`` 管理するには `tridentctl`?

管理 `tridentctl`` バックエンドを使用する `TridentBackendConfig`

このセクションでは、以下の手順で作成されたバックエンドを管理するために必要な手順について説明します。 `tridentctl` Kubernetes インターフェースから直接作成することで `TridentBackendConfig` オブジェクト。

これは次のシナリオに適用されます。

- 既存のバックエンドには `TridentBackendConfig`` なぜなら、それらは `tridentctl`。
- 作成された新しいバックエンド `tridentctl``、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 がボリュームをスケジュールして操作します。管理者はここで 2 つの選択肢のいずれかを選択できます。

- 引き続き使用 `tridentctl` これを使用して作成されたバックエンドを管理します。
- 作成したバックエンドをバインドする `tridentctl`` 新しい `TridentBackendConfig` 物体。そうすることで、バックエンドは次のように管理されることとなります。 `kubectl` そしてそうではない `tridentctl`。

既存のバックエンドを管理するには `kubectl` を作成する必要があります `TridentBackendConfig` 既存のバックエンドにバインドします。その仕組みの概要は次のとおりです。

1. Kubernetes シークレットを作成します。シークレットには、Trident がストレージ クラスター/サービスと通信するために必要な資格情報が含まれています。
2. 作成する `TridentBackendConfig`` 物体。これには、ストレージ クラスター/サービスに関する詳細が含まれており、前の手順で作成されたシークレットが参照されます。同一の設定パラメータ（例えば `spec.backendName`、`spec.storagePrefix`、`spec.storageDriverName`、等々）。 `spec.backendName` 既存のバックエンドの名前に設定する必要があります。

ステップ0: バックエンドを特定する

を作成するには `TridentBackendConfig` 既存のバックエンドにバインドする場合は、バックエンドの構成を取得する必要があります。この例では、次の JSON 定義を使用してバックエンドが作成されたと仮定します。


```

{
  "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"
      }
    }
  ]
}

```

ステップ1: Kubernetesシークレットを作成する

次の例に示すように、バックエンドの資格情報を含む Secret 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
```

ステップ2: 作成する `TridentBackendConfig` CR

次のステップは、`TridentBackendConfig` 既存のCRに自動的にバインドされる `ontap-nas-backend` (この例のように)。次の要件が満たされていることを確認してください。

- 同じバックエンド名が定義されている `spec.backendName`。
- 構成パラメータは元のバックエンドと同一です。
- 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を維持する必要があります。
- 資格情報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` 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
```

ステップ3: ステータスを確認する `TridentBackendConfig` CR

その後 `TridentBackendConfig` 作成された場合、そのフェーズは `Bound`。また、既存のバックエンドと同じバックエンド名と UUID を反映する必要があります。

```

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 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

バックエンドは、tbc-ontap-nas-backend `TridentBackendConfig` 物体。

管理 TridentBackendConfig `バックエンドを使用する` `tridentctl`

`tridentctl` 作成されたバックエンドを一覧表示するために使用できます
`TridentBackendConfig`。さらに、管理者は、次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。`tridentctl` 削除することで `TridentBackendConfig`、そして確認する
`spec.deletionPolicy` 設定されている `retain`。

ステップ0: バックエンドを特定する

例えば、次のバックエンドが次のように作成されたとします。 TridentBackendConfig:

```
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、`TridentBackendConfig`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]。

ステップ1: 確認 `deletionPolicy` 設定されている `retain`

の値を見てみましょう `deletionPolicy`。これを設定する必要があります `retain`。これにより、`TridentBackendConfig` CRが削除されても、バックエンドの定義はそのまま残り、`tridentctl`。

```
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
```



以下の場合を除き、次のステップに進まないでください。`deletionPolicy`設定されている`retain`。

ステップ2: 削除する`TridentBackendConfig`CR

最後のステップは、`TridentBackendConfig` CR。確認後、`deletionPolicy`設定されている`retain`削除を進めることができます。

```
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 |
+-----+-----+-----+
+-----+-----+-----+
```

削除すると、`TridentBackendConfig`オブジェクトを削除すると、Trident はバックエンド自体を実際に削除せずに、単にオブジェクトを削除します。

著作権に関する情報

Copyright © 2026 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。