バックエンドの構成と管理
Trident

NetApp
March 02, 2026

This PDF was generated from https://docs.netapp.com/ja-jp/trident/trident-use/backends.html on March 02, 2026. Always check docs.netapp.com for the latest.

目次

バックエンドの構成と管理

バックエンドを設定

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

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

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

Azure NetApp Files の特長

Azure NetApp Files バックエンドを設定します

Azure NetApp Files を Trident のバックエンドとして使用します。このバックエンドは、NFS および SMB ボリュームをサポートします。Trident は、Azure Kubernetes Service（AKS）クラスターのマネージド ID とワークロード ID をサポートします。

サポートされている **Azure** クラウド環境

Tridentは、複数のAzureクラウド環境でAzure NetApp Filesバックエンドをサポートします。

サポートされている Azure クラウドは次のとおりです：

- Azure コマーシャル
- Azure Government（Azure Government / MAG）

Trident を導入する場合、または Azure NetApp Files バックエンドを設定する場合は、Azure Resource Manager と認証エンドポイントが Azure クラウド環境と一致していることを確認してください。

Azure NetApp Files ドライバのサポートを確認

Tridentは、次のAzure NetApp Filesストレージドライバを提供します。

サポートされているアクセス モードには、*ReadWriteOnce*（RWO）、*ReadOnlyMany*（ROX）、*ReadWriteMany*（RWX）、および *ReadWriteOncePod*（RWOP）が含まれます。

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

考慮事項の確認

- Azure NetApp Filesは、50 GiB未満のボリュームをサポートしていません。Tridentは、より小さいボリュームが要求された場合、50 GiBのボリュームを作成します。
- Tridentでは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみがサポートされます。
- 非商用 Azure クラウドでの Azure NetApp Files の導入には、クラウド固有の Azure Resource Manager と認証エンドポイントが必要です。Trident とすべてのバックエンド構成で、Azure クラウド環境に適したエンドポイントを使用していることを確認してください。

AKS のマネージド **ID** を使用する

Tridentは、AKSクラスタ用の"管理対象ID"をサポートしています。

> `tridentctl`を使用して `Azure NetApp Files`
> バックエンドを作成または管理する場合は、正しい `Azure`
> クラウド環境用に構成されていることを確認してください。

マネージド ID を使用するには、次のものが必要です：

- AKSを使用して導入されるKubernetesクラスタ
- AKS Kubernetes クラスタで設定された管理対象 ID
- Tridentがインストールされ、 `cloudProvider`が `"Azure"`に設定されている

Trident オペレータ

編集 `tridentorchestrator_cr.yaml`して、 `cloudProvider`を `"Azure"`に設定します。

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

Helm

次の例では、Tridentをインストールし、環境変数 `$CP`を使用して `cloudProvider`を設定します：

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

<code>tridentctl</code>

次の例では Trident をインストールし、 `cloud-provider` フラグを `Azure`に設定します：

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

AKS のワークロード **ID** を使用する

ワークロード ID を使用すると、Kubernetes ポッドはワークロード 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。

`tridentctl`を使用して `Azure NetApp Files`
バックエンドを作成または管理する場合は、正しい `Azure`
クラウド環境用に構成されていることを確認してください。

ワークロード ID を使用するには、次のものが必要です：

- AKSを使用して導入されるKubernetesクラスタ
- AKS Kubernetesクラスタに設定されたワークロードIDとoidc-issuer
- Tridentがインストールされ、 `cloudProvider`が `"Azure"`に設定され、 `cloudIdentity`がワークロードIDの値に設定されている

Trident オペレータ

編集 `tridentorchestrator_cr.yaml`して `cloudProvider`を `"Azure"`に設定します。 `cloudIdentity`を `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`に設定します。

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

Helm

次の環境変数を使用して、*クラウド プロバイダ（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-
xxxxxxxxxxx'"
```

次の例では Trident をインストールし、 `cloudProvider`を `$CP`を使用して設定し、 `cloudIdentity`を `$CI`を使用して設定します：

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

<code>tridentctl</code>

次の環境変数を使用して、*クラウド プロバイダ*および*cloud identity*フラグの値を設定します：

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

次の例では Trident をインストールし、 `cloud-provider`を `$CP`に、 `cloud-identity`を `$CI`に設定します：

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

Azure NetApp Files バックエンドを設定する準備をします

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

サポートされている Azure クラウド環境

Tridentは、複数のAzureクラウド環境でAzure NetApp Filesバックエンドをサポートします。

サポートされている Azure クラウドは次のとおりです：

- Azure コマーシャル
- Azure Government（Azure Government / MAG）

環境を準備する際には、Azure サブスクリプション、ID 構成、Azure NetApp Files リソースが適切な Azure クラウド環境に作成されていることを確認してください。

NFSボリュームとSMBボリュームの前提条件

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

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

> - `subscriptionID`、 `tenantID`、 `clientID`、 `location`および `clientSecret` AKS クラスタで管理対象IDを使用する場合はオプションです。
> - `tenantID`、 `clientID`および `clientSecret` は、AKSクラスタでクラウドIDを使用する場合はオプションです。
> - 非商用 Azure クラウドでの Azure NetApp Files の導入には、クラウド固有の Azure Resource Manager と認証エンドポイントが必要です。Trident とすべてのバックエンド構成で、Azure クラウド環境に適したエンドポイントを使用していることを確認してください。

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

カスタムコントリビュータロール

```
{
  "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": []
      }
    ]
  }
}
```

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

SMBボリュームに関するその他の要件

SMBボリュームを作成するには、以下が必要です。

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

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

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

Azure NetApp Files バックエンド構成のオプションと例

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

バックエンド構成オプション

Tridentは、バックエンド構成（サブネット、仮想ネットワーク、サービスレベル、場所）を使用して、要求された場所で使用可能であり、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上にAzure NetApp Filesボリュームを作成します。

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

パラメータ	説明	デフォルト
「バージョン」	バックエンド構成のバージョン。	常に 1
'torageDriverName'	ストレージドライバの名前	「 azure-NetApp-files 」
backendName`	ストレージ バックエンドのカスタム名	ドライバ名 + "_" + ランダムな文字
' スクリプト ID' 。	Azure サブスクリプションのサブスクリプション ID AKSクラスタで管理IDが有効になっている場合はオプションです。	
「 tenantID 」。	アプリケーション登録からのテナント ID AKSクラスタで管理IDまたはクラウドIDを使用する場合はオプションです。	
「 clientID 」。	アプリケーション登録からのクライアント ID AKSクラスタで管理IDまたはクラウドIDを使用する場合はオプションです。	
「 clientSecret 」を入力します。	アプリケーション登録からのクライアントシークレット AKSクラスタで管理IDまたはクラウドIDを使用する場合はオプションです。	
「サービスレベル」	「標準」、「プレミアム」、「ウルトラ」のいずれかです	"" （ランダム）
「ロケーション」	新しいボリュームを作成する Azure の場所の名前 AKSクラスタで管理IDが有効になっている場合はオプションです。	

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	"[]" （フィルタなし）
「 netappAccounts 」のように入力します	検出されたリソースをフィルタリングするためのネットアップアカウントのリスト	"[]" （フィルタなし）
「 capacityPools	検出されたリソースをフィルタリングする容量プールのリスト	"[]" （フィルタなし、ランダム）
「 virtualNetwork 」	委任されたサブネットを持つ仮想ネットワークの名前	""
「サブネット」	「 icrosoft.Netapp/volumes` 」に委任されたサブネットの名前	""
「ネットワーク機能」	ボリュームのVNet機能のセット。 `Basic`または `Standard`の場合があります。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。 `networkFeatures`を指定すると、この機能が有効になっていない場合、ボリュームのプロビジョニングが失敗します。	""
「 nfsvMountOptions 」のように入力します	NFSマウントオプションのきめ細かな制御。SMBボリューム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、カンマ区切りのマウントオプションリストに `nfsvers=4`を含めて、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 」と入力します	要求されたボリュームサイズがこの値を超えている場合はプロビジョニングが失敗します	"" （デフォルトでは適用されません）
「バグトレースフラグ」	トラブルシューティング時に使用するデバッグフラグ。例 : `\{"API":false," メソッド ":true," 検出 ":true}`トラブルシューティングを行って詳細なログダンプが必要な場合を除き、このオプションは使用しないでください。	null

パラメータ	説明	デフォルト
`nasType`	NFSボリュームまたはSMBボリュームの作成を設定オプションはです `nfs`、 `smb` またはnull。nullに設定すると、デフォルトでNFSボリュームが使用されます。	`nfs`
`supportedTopologies`	このバックエンドでサポートされているリージョンとゾーンのリストを表します。詳細については、を参照してください "CSI トポロジを使用します"。	
`qosType`	QoSタイプ（自動または手動）を表します。	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。	`4 MiB/sec`

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

Azure クラウド環境を検討する（**26.02**）

26.02リリース以降、TridentはAzure NetApp Filesバックエンドの作成と管理を複数のAzureクラウド環境でサポートします。

サポートされている Azure クラウドは次のとおりです：

- Azure コマーシャル
- Azure Government（Azure Government / MAG）

Tridentを展開する場合、またはAzure NetApp Filesバックエンドを作成する場合は、Azure Resource Managerと認証エンドポイントがAzureクラウド環境と一致していることを確認してください。エンドポイントが一致しない場合、 `tridentctl`認証できず、バックエンドの作成に失敗します。

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、アプリ登録に必要な権限とリソース（サブネット、仮想ネットワーク、容量プール）が関連付けられていない可能性があります。デバッグ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。

`resourceGroups`、 `netappAccounts`、 `capacityPools`、 `virtualNetwork`、および `subnet`の値は、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の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。

vNet がAzure NetApp Files (ANF) ストレージ アカウントとは異なるリソース グループにある場合は、バックエンドの resourceGroups リストを構成するときに、仮想ネットワークのリソース グループを指定します。

`resourceGroups`、 `netappAccounts`、および `capacityPools`の値は、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：

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

ボリュームのプロビジョニング

構成ファイルの特別なセクションで次のオプションを指定することにより、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。詳細については、[構成例]を参照してください。

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

構成例

次の例は、ほとんどのパラメータをデフォルトのままにする基本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、サブネットを検出し、それらのプールとサブネットの1つに新しいボリュームをランダムに配置します。 `nasType`が省略されているため、 `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の使用を開始して試している段階で、実際にはプロビジョニングするボリュームに対して追加の範囲を設定することが必要な場合に適しています。

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

AKSの管理対象ID

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

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/eastus-anf-subnet
```

AKSのクラウド**ID**

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

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

容量プールフィルタを使用した特定のサービスレベル構成

このバックエンド構成では、Azureの `eastus`ロケーションの `Ultra`容量プールにボリュームを配置します。Tridentは、そのロケーションで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つにランダムに新しいボリュームを配置します。

```
---
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。

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

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

仮想プール構成

このバックエンド構成では、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 `performance`に基づいてプールを区別するために使用されました。

```
---
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:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

サポートされるトポロジ構成

Tridentは、リージョンとアベイラビリティゾーンに基づいてワークロードのボリュームのプロビジョニングを容易にします。このバックエンド構成の `supportedTopologies`ブロックは、バックエンドごとの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 トポロジを使用します" を参照してください。

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

ストレージクラスの定義

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

を使用した定義の例 `parameter.selector` フィールド

`parameter.selector`を使用すると、
`StorageClass`ごとに、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: 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

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

デフォルトネームスペースの基本設定

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

ネームスペースごとに異なるシークレットを使用する

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

ボリュームごとに異なるシークレットを使用する

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

`nasType: smb` SMB ボリュームをサポートするプールのフィルタ。
`nasType: nfs`または `nasType: null` NFS プールのフィルタ。

バックエンドを作成します

バックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

非商用Azureクラウドを使用する場合は、`tridentctl`がAzureクラウド環境のAzure Resource Managerと認証エンドポイントを使用するように設定されていることを確認してください。バックエンドの作成に失敗した場合は、バックエンドの設定を確認し、ログを表示して原因を特定します：

```
tridentctl logs
```

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

Google Cloud NetAppボリューム

NASワークロード用にGoogle Cloud NetApp Volumesを設定する

Google Cloud NetApp Volumes を Trident のバックエンドとして設定し、ファイルベースのストレージボリュームをプロビジョニングできます。Trident は、Google Cloud NetApp Volumes 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、Google Cloud NetApp Volumes で NAS ワークロードと SAN ワークロードに別々のバックエンドを使用します。`google-cloud-netapp-volumes`バックエンドはファイルベースのプロトコルのみをサポートしており、iSCSI ボリュームのプロビジョニングには使用できません。

iSCSIブロックボリュームをプロビジョニングするには、`google-cloud-netapp-volumes-san`バックエンドを使用します。これは、SANワークロード専用に設計された別のバックエンドタイプです。

NASボリュームとiSCSIブロックボリューム

Google Cloud NetApp Volumes は、アプリケーションがデータにアクセスして管理する方法が異なる NAS とブロックストレージの両方をサポートします。

NAS ボリュームはファイルベースのストレージを提供し、NFS や SMB などの標準ファイル プロトコルを介してアクセスされます。ボリュームは共有ファイルシステムとしてマウントされ、複数のポッドまたはノードからの同時アクセスをサポートします。

iSCSI ブロック ボリュームは、生のブロック ストレージを提供し、Kubernetes ノードに接続されたブロック デバイスとしてアクセスされます。ブロック ストレージは通常、ワークロードでブロック レベルのアクセスまたはアプリケーション管理の I/O 動作が必要な場合に使用されます。

これは次の環境に適用されます：

- Trident 26.02以降
- Google Kubernetes Engine （ GKE ）
- Google Cloud NetApp Volumes NAS プール
- NFSおよびSMBワークロード

ブロック（iSCSI）ワークロードについては、ブロックストレージ（iSCSI）を設定するを参照してください。

Google Cloud NetApp Volumesドライバの詳細

Trident は、Google Cloud NetApp Volumes から NAS ストレージをプロビジョニングするための `google-cloud-netapp-volumes`ドライバを提供します。

ドライバーは次のアクセス モードをサポートしています：

- ReadWriteOnce（RWO）
- ReadOnlyMany（ROX）
- ReadWriteMany（RWX）
- ReadWriteOncePod（RWOP）

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

Google Kubernetes Engine のクラウド ID

Cloud Identity を使用すると、静的な Google Cloud 認証情報を使用する代わりに、ワークロード ID として認証することで、Kubernetes ワークロードが Google Cloud リソースにアクセスできるようになります。

Google Cloud NetApp Volumes でクラウド ID を使用するには、次のものが必要です：

- Google Kubernetes Engine（GKE）を使用してデプロイされた Kubernetes クラスタ
- GKE クラスタでワークロード ID が有効になっており、ノード プールでメタデータ サーバーが有効になっている
- Google Cloud NetApp Volumes Admin ロールを持つ Google Cloud サービスアカウント(`roles/netapp.admin`または同等のカスタムロール
- Tridentがインストールされ、クラウド プロバイダが `GCP`に設定され、クラウドIDアノテーションが設定されている

Trident オペレータ

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`を編集して `cloudProvider`を `GCP`に設定し、 `cloudIdentity`を GKE サービス アカウントに設定します。

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

Helm

Helm を使用して Trident をインストールする際に、クラウド プロバイダとクラウド ID を設定します。

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

Tridentctl

クラウド プロバイダとクラウド ID を指定して Trident をインストールします。

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
  -n trident
```

Trident NAS バックエンドを設定する

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

NAS ボリュームのプロビジョニング

NASボリュームは、 `google-cloud-netapp-volumes`バックエンドを使用してプロビジョニングされ、NFSおよびSMBプロトコルをサポートします。

NFS ボリューム用の StorageClass

NFS ボリュームをプロビジョニングするには、 `nasType`を `nfs`に設定します。

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

SMBボリューム用のStorageClass

`nasType`、 `csi.storage.k8s.io/node-stage-secret-name`、および `csi.storage.k8s.io/node-stage-secret-namespace`を使用して、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。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-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"
allowVolumeExpansion: true
```

PersistentVolumeClaim の例（RWX）

NAS ボリュームは同時アクセスをサポートしており、一般的には
`ReadWriteMany`でプロビジョニングされます。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

PersistentVolumeClaim の例（RWO）

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

NASボリュームは `volumeMode: Filesystem`を使用します。

SAN ワークロード用に Google Cloud NetApp Volumes を設定

Tridentを設定して、Google Cloud NetApp VolumesからiSCSIプロトコルを使用してブロックストレージボリュームをプロビジョニングできます。SANボリュームは、`google-cloud-netapp-volumes-san`ストレージドライバーを使用して*Flex Unified*ストレージプールからプロビジョニングされます。

このドライバーはブロックワークロード専用であり、NASプロトコルをサポートしていません。

`google-cloud-netapp-volumes-san`バックエンドは、iSCSIブロックボリュームのプロビジョニングに必要です。 `google-cloud-netapp-volumes`バックエンドはNASプロトコルのみをサポートしており、SANワークロードには使用できません。

NASボリュームと**iSCSI**ブロックボリューム

Google Cloud NetApp Volumes は、アプリケーションがデータにアクセスして管理する方法が異なる NAS とブロックストレージの両方をサポートします。

NAS ボリュームはファイルベースのストレージを提供し、NFS または SMB を使用して共有ファイルシステムとしてマウントされます。これらのボリュームは、複数のポッドまたはノードが同じデータに同時にアクセスする必要がある場合によく使用されます。

iSCSIブロックボリュームは、rawブロックストレージを提供し、ブロックデバイスとしてKubernetesノードに接続されます。各ボリュームは論理ユニット番号（LUN）としてプロビジョニングされ、iSCSIプロトコルを使用してアクセスされます。ブロックストレージは通常、ワークロードでブロックレベルのアクセスまたはアプリケーション管理のI/O動作が必要な場合に使用されます。

Flex Unified Google Cloud NetApp Volumes プールを基盤とする Trident 管理の iSCSI ストレージを使用して、Google Kubernetes Engine にブロック指向のワークロードをデプロイできます。

これは次の環境に適用されます：

- Trident 26.02以降
- Google Kubernetes Engine （ GKE ）
- Google Cloud NetApp Volumes **Flex Unified** ストレージプール
- iSCSIベースのブロックワークロード

Trident 26.02では、SANワークロードでサポートされるのはFlexサービスレベルのみです。

ストレージアーキテクチャの概要

SANワークロードの場合、TridentはFlex Unified ストレージプールにiSCSI論理ユニット番号（LUN）を作成して、ブロックストレージをプロビジョニングします。

各Kubernetes PersistentVolumeは単一のLUNに対応します。Tridentは、作成、ホストマッピング、接続、クリーンアップなど、LUNのライフサイクル全体を管理します。

Flex Unified ストレージプール

Flex Unified ストレージプールは、iSCSIプロトコルを使用してブロックストレージを提供し、SANプロビジョニングに必要です。

Trident 26.02の場合：

- **Flex Unified REGIONAL** プールのみがサポートされています
- Flex Unified *ZONAL*プールは、Trident 26.02.1以降でサポートされます
- SANワークロードでは*Flex*サービスレベルのみがサポートされます

ブロックボリューム

ブロック ボリュームは iSCSI LUN としてプロビジョニングされ、Kubernetes ノードにブロック デバイスとして提示されます。

ブロックボリューム：

- iSCSIプロトコルを使用する
- ファイルシステムとrawブロックのプレゼンテーションをサポート
- Tridentによって接続され、管理されます
- 複数の Kubernetes アクセスモードをサポート

アクセスモード

Tridentによってプロビジョニングされたブロックボリュームは、次のアクセスモードをサポートします：

- `ReadWriteOnce`（RWO）
- `ReadOnlyMany`（ROX）
- `ReadWriteOncePod`（RWOP）
- `ReadWriteMany`（RWX）、次の場合にのみサポートされます： `volumeMode: Block`

volumeModeの動作

`volumeMode`フィールドは、ブロックボリュームの公開方法を制御します：

- `Filesystem` Trident がボリュームをフォーマットしてマウントします。
- `Block` Tridentはデバイスを接続し、rawブロックデバイスとして公開します。

Trident SANバックエンドを設定する

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

SAN ワークロード用の **StorageClass** を作成します

SANバックエンドを設定したら、
`google-cloud-netapp-volumes-san`ドライバを参照するStorageClassを作成します。

ファイルシステムのタイプは、バックエンドではなく StorageClass で定義されます。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

サポートされているファイルシステムの種類：

- `ext4`（デフォルト）
- `ext3`
- `xfs`

SANドライバはFlexサービスレベルのみをサポートし、 `exportRule`、 `unixPermissions`、 `nasType`、 `snapshotDir`、 `nfsMountOptions`などのNAS固有のバックエンドパラメータや階層化関連の設定は使用しません。

サポートされている操作

`google-cloud-netapp-volumes-san`ドライバーを使用してプロビジョニングされたブロックボリュームは、次の機能をサポートします：

- 作成
- 削除
- クローン
- スナップショット
- サイズ変更
- インポート

ブロックボリュームのプロビジョニング

ReadWriteOnce（RWO）

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadWriteOncePod（RWOP）

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadOnlyMany（ROX）

ROXの一般的なパターンは、既存のReadWriteOnceボリュームを複製し、クローンを読み取り専用としてマウントすることです。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

ReadWriteMany（RWX）— rawブロックのみ

ReadWriteManyは、 `volumeMode: Block`の場合にのみサポートされます。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

追加GiBオーバープロビジョニング動作

Google Cloud NetApp Volumes ブロックボリュームには、内部メタデータのオーバーヘッドが含まれます。このオーバーヘッドにより、プロビジョニングされた容量と比較して、カーネルに表示されるデバイスサイズが小さくなります。

テストの結果：

- 初期作成時に約 300 KiB のオーバーヘッド

- サイズ変更後、最大約 107 MiB のオーバーヘッド

Google Cloud NetApp Volumes は GiB 単位の割り当てのみを受け入れるため、Trident は次の方法で、使用可能なデバイスサイズが常に PVC 要求を満たすか超えることを保証します（：）

- 要求されたサイズを次の整数 GiB に切り上げる
- 1 GiBのバッファを追加する

例：

- PVC リクエスト：100 GiB
- Google Cloud NetApp Volumes でのプロビジョニングサイズ：101 GiB
- アプリケーションから見える使用可能領域：少なくとも 100 GiB

これにより、内部メタデータのオーバーヘッドを考慮した後でも、アプリケーションが常に要求された容量を受け取ることが保証されます。

Podの例

ファイルシステムにマウントされたブロックボリューム（**RWO**）

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

Raw ブロックデバイス **(RWX)**

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

アタッチとマウントの動作

Google Cloud NetApp Volumes からプロビジョニングされた SAN ボリュームの場合：

- Trident は、Flex Unified ストレージプールに論理ユニット番号（LUN）を作成します。
- 公開中、Trident は LUN をノードごとのホストグループにマップします。
- ノードステージング中、Trident：
  - iSCSIターゲットにログインします
  - LUNを検出します
  - マルチパスを設定します
- もし `volumeMode: Filesystem`、Tridentは必要に応じてデバイスをフォーマットし、マウントします。
- `volumeMode: Block`の場合、Tridentはデバイスを接続し、フォーマットやマウントを行わずにポッドに直接公開します。

> SAN ブロックボリュームでは、分散ロックや書き込み調整は提供されません。ブロックボリュームが複数のノードからアクセスされる場合（ReadWriteMany with `volumeMode: Block`）、アプリケーションまたはファイルシステムは同時実行性を管理する必要があります。

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

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

NFSボリュームノゼンテイジョウケン

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

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

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

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

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

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

バックエンド構成オプション

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

パラメータ	説明	デフォルト
「バージョン」		常に 1
'torageDriverName'	ストレージドライバの名前	の値は `storageDriverName` 「google-cloud-netapp-volumes」と指定する必要があります。
backendName`	（オプション）ストレージバックエンドのカスタム名	ドライバ名 + "_" + API キーの一部
`storagePools`	ボリューム作成用のストレージプールを指定するオプションのパラメータ。	
「 ProjectNumber 」	Google Cloud アカウントのプロジェクト番号。この値は、Google Cloudポータルのホームページにあります。	

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

ボリュームのプロビジョニングオプション

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

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

構成例

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

最小限の構成

これは、バックエンドの絶対的な最小構成です。この構成では、Tridentは設定された場所でGoogle Cloud NetApp Volumeに委譲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。は省略されているため、 `nasType` `nfs` デフォルトが適用され、バックエンドでNFSボリュームがプロビジョニングされます。

この構成は、Google Cloud NetApp Volumeの使用を開始して試用す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に対して追加の範囲設定が必要になることがよくあります。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/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
  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ホリユウムノセツテイ

```
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フイルタヲシヨウシタセツテイ

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/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
```

仮想プール構成

このバックエンド構成では、1つのファイルに複数の仮想プールが定義されます。仮想プールは、セクションで定義し `storage` ます。さまざまな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、それらを表すストレージクラスをKubernetesで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。たとえば、次の例では `performance` 、仮想プールを区別するためにラベルと `serviceLevel` タイプが使用されています。

また、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たり、個 々 の仮想プールのデフォルト値を上書きしたりすることもできます。次の例では、 `snapshotReserve` `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、を参照してください "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-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
```

```
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

GKEのクラウドID

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

サポートされるトポロジ構成

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

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

次の手順

バックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

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

```
kubectl get tridentbackendconfig

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

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

```
tridentctl logs
```

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

ストレージクラスの定義

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

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

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

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

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

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

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

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

SMBボリュームの定義例

`node-stage-secret-name`、およびを使用する `nasType` `node-stage-secret-namespace`と、SMBボリュームを指定し、必要なActive Directoryクレデンシャルを指定できます。権限の有無にかかわらず、すべてのActive Directoryユーザ/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。

デフォルトネームスペースの基本設定

```
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"
```

ネームスペースごとに異なるシークレットを使用する

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: 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: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

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

PVC定義の例PVCテイギノレイ

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

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

```
kubectl get pvc gcnv-nfs-pvc

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

Google Cloud NetApp Volumes の自動階層化を構成する

このページでは、Tridentを使用してGoogle Cloud NetApp Volumesの自動階層化を設定する方法について説明します。自動階層化は、ボリュームのプロビジョニング中にTridentバックエンドパラメータとPersistentVolumeClaimアノテーションを使用して設定します。

概要

自動階層化により Trident は非アクティブなデータをパフォーマンス層から容量層に自動的に移動するボリュームをプロビジョニングできます。これにより、頻繁にアクセスされるデータのパフォーマンスを維持しながら、ストレージコストを削減できます。

Trident はボリューム作成時にのみ自動階層化設定を適用します。プロビジョニング後の変更は Trident 26.02 ではサポートされていません。

概念

自動階層化

自動階層化では、アクセス パターンに基づいて、アクセス頻度の低いデータをパフォーマンス層から容量層に移動します。データの移動は非同期的に行われ、即時には行われません。

階層化ポリシー

階層化ポリシーは、ボリュームに対して自動階層化を有効にするかどうかを決定します。

以下のポリシーがサポートされています：* `auto`：アクセスパターンに基づいて自動階層化を有効にします * `none`：自動階層化を無効にします

冷房日数

冷却日数は、データ ブロックが階層化の対象となる前に非アクティブのままにしておく必要がある最小日数を指定します。冷却日数は、階層化ポリシーが `auto`に設定されている場合にのみ適用されます。

構成モデル

構成スコープ

自動階層化は複数のスコープで設定できます：

- ストレージプールのスコープ 環境プールからプロビジョニングされたすべてのボリューム。
- ボリュームスコープ 環境単一のボリュームに適用されます（PersistentVolumeClaim アノテーションを使用）。

Tridentは、各設定が定義されている場所に基づいて有効な構成を決定します。

設定の優先順位

複数のスコープで同じ設定が定義されている場合、Tridentは次の優先順位を適用します：

1. PersistentVolumeClaim アノテーション
2. Trident バックエンド構成
3. ストレージ プールのデフォルト

より高い優先順位で定義された設定は、より低いレベルの値を上書きします。

Trident 26.02 でサポートされている機能

Trident 26.02は、Google Cloud NetApp Volumesの以下の自動階層化機能をサポートします：

- ボリュームのプロビジョニング時の自動階層化の有効化または無効化
- Tridentバックエンド構成での階層化ポリシーの定義
- PVC アノテーションを使用したボリュームごとの階層化ポリシーと冷却日数の上書き
- 自動階層化が有効になっているボリュームのクーリング日数の設定

Trident 26.02でサポートされていない機能

次の処理はサポートされていません。

- ボリューム作成後の自動階層化設定の変更
- Kubernetes アップデートを使用して既存のボリュームの階層化ポリシーを変更する
- Trident 管理のプロビジョニング ワークフロー外での自動階層化設定の適用

バックエンド構成パラメータ

以下のパラメータは、Trident バックエンド構成で定義された場合の自動階層化動作を制御します：

パラメータ	必須	説明
tieringPolicy	いいえ	ボリュームの階層化ポリシー（(`auto`または `none`）
tieringMinimumCoolingDays	いいえ	データが階層化されるまでの非アクティブ日数（範囲：2～183、デフォルト：31）

PersistentVolumeClaim アノテーションを使用したボリュームレベルのオーバーライド

サポートされている注釈

PersistentVolumeClaim 注釈を使用すると、ボリュームごとに自動階層化設定をオーバーライドできます。

アノテーション	説明
`trident.netapp.io/tieringPolicy`	ボリュームの階層化ポリシーを上書きします
`trident.netapp.io/tieringMinimumCoolingDays`	ボリュームの冷却日数の値を上書きします

例：自動階層化オーバーライド付き**PersistentVolumeClaim**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

動作と制限

プロビジョニング動作

- 自動階層化設定はボリュームの作成時にのみ評価され、適用されます。
- Tridentは、プロビジョニング後に階層化設定を調整しません。
- 階層化ポリシーが `none`に設定されている場合、冷却日は無視されます。

プラットフォームの制限

- 自動階層化は NAS ボリューム（NFS および SMB）でのみサポートされます。
- ブロックボリューム（iSCSI）は自動階層化をサポートしていません。
- Google Cloud NetApp Volumes ストレージ プールでは、Google Cloud で自動階層化が有効になっている必要があります。

サポートされている値

- `tieringMinimumCoolingDays`の有効範囲：2~183
- デフォルト値：31

NetApp HCI または SolidFire バックエンドを設定します

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

Elementドライバの詳細

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

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

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

作業を開始する前に

Elementバックエンドを作成する前に、次の情報が必要になります。

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

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
「バージョン」		常に 1
'torageDriverName'	ストレージドライバの名前	常に「SolidFire - SAN」
backendName`	カスタム名またはストレージバックエンド	「SolidFire _」+ストレージ（iSCSI）IPアドレス
「エンドポイント」	テナントのクレデンシャルを使用する SolidFire クラスタの MVIP	
「 VIP 」	ストレージ（ iSCSI ）の IP アドレスとポート	
「ラベル」	ボリュームに適用する任意の JSON 形式のラベルのセット。	""
「 tenantname 」	使用するテナント名（見つからない場合に作成）	
「 InitiatorIFCace 」	iSCSI トラフィックを特定のホストインターフェイスに制限します	デフォルト
UseCHAP'	CHAPを使用してiSCSIを認証します。TridentはCHAPを使用します。	正しいです
「アクセスグループ」	使用するアクセスグループ ID のリスト	「Trident」という名前のアクセスグループのIDを検索します。
「タイプ」	QoS の仕様	
「 limitVolumeSize 」と入力します	要求されたボリュームサイズがこの値を超えている場合、プロビジョニングが失敗します	"" （デフォルトでは適用されません）
「バグトレースフラグ」	トラブルシューティング時に使用するデバッグフラグ。例： {"api"：false、"method"：true}	null

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

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

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

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

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

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

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

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

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
  - labels:
      performance: gold
      cost: "4"
    zone: us-east-1a
    type: Gold
  - labels:
      performance: silver
      cost: "3"
    zone: us-east-1b
    type: Silver
  - labels:
      performance: bronze
      cost: "2"
    zone: us-east-1c
    type: Bronze
  - labels:
```

```
      performance: silver
      cost: "1"
    zone: us-east-1d
```

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

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

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold; cost=4
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=3
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze; cost=2
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=1
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
  fsType: ext4
```

詳細については、こちらをご覧ください

- "ボリュームアクセスグループ"

ONTAP SAN ドライバ

ONTAP SANドライバの概要

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

ONTAP SANドライバの詳細

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

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

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

警告

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

ユーザ権限

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

`fsxadmin`ユーザは、クラスタ管理者ユーザに代わる限定的なユーザです。

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

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

NVMe/TCPに関するその他の考慮事項

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

- IPv6
- NVMeボリュームのSnapshotとクローン
- NVMeボリュームのサイズ変更
- Tridentの外部で作成されたNVMeボリュームをインポートして、そのライフサイクルをTridentで管理できるようにする
- NVMeネイティブマルチパス
- Kubernetesノードのグレースフルシャットダウンまたはグレースフルシャットダウン（24.06）

Tridentは以下をサポートしていません。

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

メモ	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。

バックエンドに**ONTAP SAN**ドライバを設定する準備をします

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

要件

すべての ONTAP バックエンドでは、Trident では少なくとも 1 つのアグリゲートを SVM に割り当てる必要があります。

メモ	"ASA r2システム"ストレージ層の実装において他のONTAPシステム (ASA、 AFF、 FAS) と異なります。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"これ"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。

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

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

ONTAPバックエンドの認証

Tridentには、ONTAPバックエンドの認証に2つのモードがあります。

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

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

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

クレデンシャルベースの認証を有効にします

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

バックエンド定義の例は次のようになります。

YAML

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

JSON

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

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

証明書ベースの認証の有効化

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

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

一般的なワークフローは次の手順で構成されます。

手順

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

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

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

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

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

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

メモ	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 'cert' 認証方式をサポートしていることを確認します

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

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

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

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

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

7. 前の手順で得た値を使用してバックエンドを作成します。

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

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

認証方法を更新するか、クレデンシャルをローテーションして

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

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

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

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

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

Trident用のカスタムONTAPロールの作成

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

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

ONTAP CLIノシヨウ

1. 次のコマンドを使用して新しいロールを作成します。

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

2. Tridentユーザのユーザ名を作成します。

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

3. ユーザにロールをマッピングします。

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

System Managerの使用

ONTAPシステムマネージャで、次の手順を実行します。

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

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

   b. の横にある矢印アイコン（→*）を選択します。
   c. [Roles]*で[+Add]*を選択します。
   d. ロールのルールを定義し、*[保存]*をクリックします。
2. ロールを**Trident**ユーザにマップする:+[ユーザとロール]ページで次の手順を実行します。
   a. で**[**アイコンの追加**]**+*を選択します。
   b. 必要なユーザ名を選択し、* Role *のドロップダウンメニューでロールを選択します。
   c. [保存（ Save ）] をクリックします。

詳細については、次のページを参照してください。

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

双方向**CHAP**による接続の認証

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

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

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

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

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

仕組み

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

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

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

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

警告	バックエンドのCHAPシークレットを更新する場合は、を使用してバックエンドを更新する必要があります `tridentctl`。ONTAP CLIまたはONTAPシステムマネージャを使用してストレージクラスタのクレデンシャルを更新しないでください。Tridentではこれらの変更を反映できません。

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

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

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

ONTAP のSAN構成オプションと例

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

"ASA r2システム"ストレージ層の実装において他のONTAPシステム (ASA、 AFF、 FAS) と異なります。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。"。

メモ	のみ `ontap-san`ドライバー (iSCSI、NVMe/TCP、および FC 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 またはその他の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

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
「バージョン」		常に 1
'torageDriverName'	ストレージドライバの名前	`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	ドライバ名+"_"+ dataLIF
「管理 LIF 」	クラスタ管理LIFまたはSVM管理LIFのIPアドレス。 Fully Qualified Domain Name（FQDN；完全修飾ドメイン名）を指定できます。 IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`。 シームレスなMetroClusterスイッチオーバーについては、を参照してMetroClusterの例ください。 メモ 「vsadmin」のクレデンシャルを使用する場合はSVMのクレデン `managementLIF`シャル、「admin」のクレデンシャルを使用する場合はクラスタのクレデンシャル `managementLIF`を使用する必要があります。	"10.0.0.1 ","[2001：1234：abcd：：fe]"

パラメータ	説明	デフォルト
「重複排除	プロトコル LIF の IP アドレス。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`。* iSCSIの場合は指定しないでください。**Trident**は、を使用して**"ONTAP の選択的LUNマップ"**、マルチパスセッションの確立に必要な**iSCSI LIF**を検出します。が明示的に定義されている場合は、警告が生成され **`dataLIF`**ます。 MetroClusterの場合は省略してください。*を参照してくださいMetroClusterの例。	SVMの派生物です
'VM'	使用する Storage Virtual Machine * MetroClusterの場合は省略してください。* MetroClusterの例。	SVM 「管理 LIF 」が指定されている場合に生成されます
「 useCHAP 」	CHAPを使用してONTAP SANドライバのiSCSIを認証します（ブーリアン）。バックエンドで指定されたSVMのデフォルト認証として双方向CHAPを設定して使用する場合は、Tridentのをに設定し `true`ます。詳細については、を参照してください "バックエンドにONTAP SANドライバを設定する準備をします" 。**FCP** または **NVMe/TCP** ではサポートされません。	「偽」
「 chapInitiatorSecret 」	CHAP イニシエータシークレット。「 useCHAP = TRUE 」の場合は必須	""
「ラベル」	ボリュームに適用する任意の JSON 形式のラベルのセット	""
「 chapTargetInitiatorSecret 」	CHAP ターゲットイニシエータシークレット。「 useCHAP = TRUE 」の場合は必須	""
「 chapUsername 」	インバウンドユーザ名。「 useCHAP = TRUE 」の場合は必須	""
「 chapTargetUsername 」	ターゲットユーザ名。「 useCHAP = TRUE 」の場合は必須	""
「 clientCertificate 」をクリックします	クライアント証明書の Base64 エンコード値。証明書ベースの認証に使用されます	""
「 clientPrivateKey 」	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます	""
「 trustedCacertifate 」	信頼された CA 証明書の Base64 エンコード値。任意。証明書ベースの認証に使用されます。	""

パラメータ	説明	デフォルト
「ユーザ名」	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 」と入力します	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"。	""
'VM'	使用する Storage Virtual Machine	SVM 「管理 LIF 」が指定されている場合に生成されます
'toragePrefix'	SVM で新しいボリュームをプロビジョニングする際に使用するプレフィックスを指定します。あとから変更することはできません。このパラメータを更新するには、新しいバックエンドを作成する必要があります。	`trident`
「集約」	プロビジョニング用のアグリゲート（オプション。設定する場合は SVM に割り当てる必要があります）。ドライバの場合 `ontap-nas-flexgroup`、このオプションは無視されます。割り当てられていない場合は 、使用可能ないずれかのアグリゲートを使用してFlexGroupボリュームをプロビジョニングできます。 メモ: SVMでアグリゲートが更新されると、Tridentコントローラを再起動せずにSVMをポーリングすることで、Tridentでアグリゲートが自動的に更新されます。ボリュームをプロビジョニングするようにTridentで特定のアグリゲートを設定している場合、アグリゲートの名前を変更するかSVMから移動すると、SVMアグリゲートのポーリング中にTridentでバックエンドが障害状態になります。アグリゲートをSVMにあるアグリゲートに変更するか、アグリゲートを完全に削除してバックエンドをオンラインに戻す必要があります。 **ASA r2** システムには指定しないでください。	""

パラメータ	説明	デフォルト
「AggreglimitateUsage 」と入力します	使用率がこの割合を超えている場合は、プロビジョニングが失敗します。Amazon FSx for NetApp ONTAP バックエンドを使用している場合は、を指定しないで `limitAggregateUsage`ください。指定されたと `vsadmin`には `fsxadmin`、アグリゲートの使用量を取得してTridentを使用して制限するために必要な権限が含まれていません。**ASA r2** システムには指定しないでください。	""（デフォルトでは適用されません）
「limitVolumeSize 」と入力します	要求されたボリュームサイズがこの値を超えている場合、プロビジョニングが失敗します。また、LUNで管理するボリュームの最大サイズも制限します。	""（デフォルトでは適用されません）
'lunsPerFlexvol	FlexVol あたりの最大 LUN 数。有効な範囲は 50 、200 です	`100`
「バグトレースフラグ」	トラブルシューティング時に使用するデバッグフラグ。例： {"api"：false、"method"：true} トラブルシューティングを行い、詳細なログダンプが必要な場合を除き、は使用しないでください。	`null`

パラメータ	説明	デフォルト
「 useREST` 」	ONTAP REST API を使用するためのブール パラメーター。 `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`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。 `useREST`NVMe/TCP に完全対応しています。 メモ: 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。 指定されている場合、常に **`true`ASA r2** システムの場合。	true ONTAP 9.15.1以降の場合は、それ以外の場合は false。
sanType	iSCSI、 nvme`NVMe/TCP、または `fcp`SCSI over Fibre Channel（FC；SCSI over Fibre Channel）に対してを選択します `iscsi。	iscsi 空白の場合

パラメータ	説明	デフォルト
`formatOptions`	を使用して、 `formatOptions`コマンドのコマンドライン引数を指定します。この引数 `mkfs`は、ボリュームがフォーマットされるたびに適用されます。これにより、好みに応じてボリュームをフォーマットできます。デバイスパスを除いて、mkfsコマンドオプションと同様にformatOptionsを指定してください。例：「-E nodiscard」 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。**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。	
`limitVolumePoolSize`	ONTAP SANエコノミーバックエンドでLUNを使用する場合の、要求可能な最大FlexVolサイズ。	"" （デフォルトでは適用されません）
`denyNewVolumePools`	バックエンドがLUNを格納するために新しいFlexVolボリュームを作成することを制限します `ontap-san-economy`。新しいPVのプロビジョニングには、既存のFlexVolのみが使用されます。	

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ています。

- **-E nodiscard (ext3、ext4):** mkfs 時にブロックを破棄しません (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ース / 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、ext3、ext4 ファイル システムに適用できます。
- **-K (xfs):** mkfs 時にブロックを破棄しません。このオプションは xfs ファイル システムに適用できます。

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ーを管理するには、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用のバックエンド構成オプション

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

パラメータ	説明	デフォルト
「平和の配分」	space-allocation for LUN のコマンドを指定します	"true" 指定されている場合は、 **`true` ASA r2** システムの場合。
「平和のための準備」を参照してください	スペースリザベーションモード：「none」（シン）または「volume」（シック）。設定 **`none` ASA r2** システムの場合。	"なし"
「ナプショットポリシー」	使用するSnapshotポリシー。設定 **`none` ASA r2** システムの場合。	"なし"
「 QOSPolicy 」	作成したボリュームに割り当てる QoS ポリシーグループ。ストレージプール / バックエンドごとに QOSPolicy または adaptiveQosPolicy のいずれかを選択します。TridentでQoSポリシーグループを使用するには、ONTAP 9 .8以降が必要です。共有されていないQoSポリシーグループを使用し、ポリシーグループが各コンスティチュエントに個別に適用されるようにします。QoSポリシーグループを共有すると、すべてのワークロードの合計スループットの上限が適用されます。	""
「 adaptiveQosPolicy 」を参照してください	アダプティブ QoS ポリシーグループ：作成したボリュームに割り当てます。ストレージプール / バックエンドごとに QOSPolicy または adaptiveQosPolicy のいずれかを選択します	""
「スナップショット予約」	Snapshot用にリザーブされているボリュームの割合。**ASA r2** システムには指定しないでください。	次の場合は「0」 `snapshotPolicy` は「none」、それ以外の場合は「」です。
'plitOnClone	作成時にクローンを親からスプリットします	いいえ
「暗号化」	新しいボリュームでNetApp Volume Encryption（NVE）を有効にします。デフォルトはです。 `false`このオプションを使用するには、クラスタで NVE のライセンスが設定され、有効になっている必要があります。バックエンドでNAEが有効になっている場合、Tridentでプロビジョニングされたすべてのボリュームでが有効になります。詳細については、を参照してください"TridentとNVEおよびNAEとの連携"。	"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。を参照してください "Linux Unified Key Setup（LUKS；統合キーセットアップ）を使用"。	"" 設定 **`false`** **ASA r2** システムの場合。
階層ポリシー	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

デフォルトが定義されている例を次に示します。

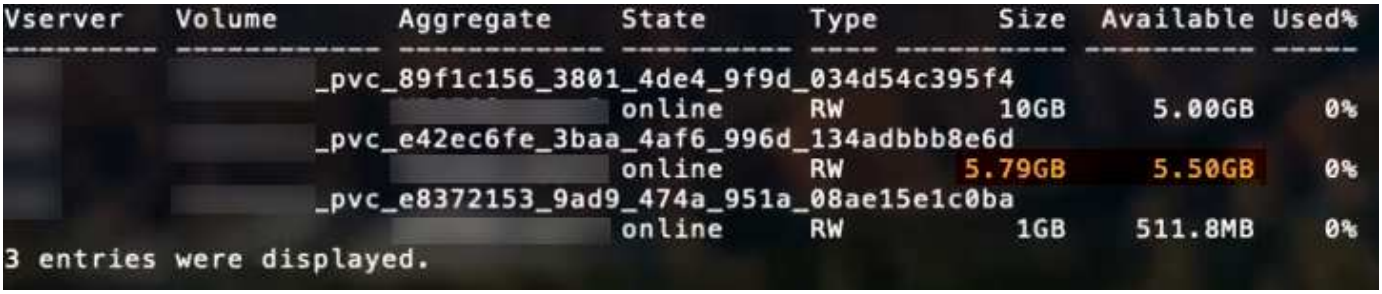
```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: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

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

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

```
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。 `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です。 `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。



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

最小限の設定例

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

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

ONTAP SANの例

これは、 `ontap-san` ドライバ。

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

MetroClusterの例

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

スイッチオーバーとスイッチバックをシームレスに実行するには、を使用してSVMを指定し `managementLIF`、パラメータは省略します `svm`。例：

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

ONTAP SANの経済性の例

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

証明書ベースの認証の例

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

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

双方向CHAPの例

次の例では、 `useCHAP` をに設定します `true`。

ONTAP SAN CHAPの例

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

ONTAP SANエコノミーCHAPの例

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

NVMe/TCPの例

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

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

SCSI over FC（FCP）の例

ONTAPバックエンドでFCを使用してSVMを設定しておく必要があります。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エコノミードライバの**formatOptions**の例

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

仮想プールを使用するバックエンドの例

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

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

たりできます。

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

ONTAP SANの例

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

ONTAP SANの経済性の例

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
```

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

NVMe/TCPの例

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

バックエンドを **StorageClasses** にマッピングします

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ONTAP NAS ドライバ

ONTAP NASドライバの概要

ONTAP および Cloud Volumes ONTAP の NAS ドライバを使用した ONTAP バックエン

ドの設定について説明します。

ONTAP NASドライバの詳細

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

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

警告

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

ユーザ権限

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

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

メモ

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

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

ONTAP NASドライバを使用してバックエンドを設定する準備をします

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

25.10リリース以降、 NetApp Tridentは以下をサポートします。"NetApp AFXストレージシステム" 。NetApp AFX ストレージ システムは、ストレージ層の実装において他のONTAPシステム (ASA、 AFF、 FAS) とは異なります。

メモ	のみ `ontap-nas`AFX システムではドライバー (NFS プロトコル付き) がサポートされていますが、SMB プロトコルはサポートされていません。

Tridentバックエンド構成では、システムが AFX であることを指定する必要がありません。選択すると `ontap-nas`として `storageDriverName`Trident はAFX システムを自動的に検出します。

要件

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

ONTAPバックエンドの認証

Tridentには、ONTAPバックエンドの認証に2つのモードがあります。

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

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

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

クレデンシャルベースの認証を有効にします

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

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

バックエンド定義の例は次のようになります。

YAML

```yaml
---
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

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

証明書ベースの認証を有効にします

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

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

せん。

一般的なワークフローは次の手順で構成されます。

手順

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

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

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

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

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

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

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

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

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

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

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

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

7. 前の手順で得た値を使用してバックエンドを作成します。

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

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

認証方法を更新するか、クレデンシャルをローテーションして

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

```
cat cert-backend-updated.json
```

```
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}
```

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

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

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

Trident用のカスタムONTAPロールの作成

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

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

ONTAP CLIノシヨウ

1. 次のコマンドを使用して新しいロールを作成します。

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

2. Tridentユーザのユーザ名を作成します。

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

3. ユーザにロールをマッピングします。

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

System Managerの使用

ONTAPシステムマネージャで、次の手順を実行します。

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

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

   b. の横にある矢印アイコン（→*）を選択します。
   c. [Roles]*で[+Add]*を選択します。
   d. ロールのルールを定義し、*[保存]*をクリックします。
2. ロールを**Trident**ユーザにマップする:+[ユーザとロール]ページで次の手順を実行します。
   a. で**[**アイコンの追加**]**+*を選択します。
   b. 必要なユーザ名を選択し、* Role *のドロップダウンメニューでロールを選択します。
   c. [保存（ Save ）] をクリックします。

詳細については、次のページを参照してください。

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

NFS エクスポートポリシーを管理します

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

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

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

エクスポートポリシーを動的に管理

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

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

例

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

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

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

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

- `autoExportPolicy`がに設定されてい `true`ます。これは、Tridentが、このバックエンドを使用してSVMに対してプロビジョニングされたボリュームごとにエクスポートポリシーを作成し、アドレスブロックを使用してルールの追加と削除を処理すること `autoexportCIDRs`を示します `svm1`。ボリュームがノードに接続されるまでは、そのボリュームへの不要なアクセスを防止するルールのない空のエクスポートポリシーが使用されます。ボリュームがノードに公開されると、Tridentは、指定したCIDRブロック内のノードIPを含む基盤となるqtreeと同じ名前のエクスポートポリシーを作成します。これらのIPは、親FlexVol 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：
    - バックエンドUUID 403b5326-8482-40dB-96d0-d83fb3f4daec
    - `autoExportPolicy`に設定 `true`
    - ストレージプレフィックス `trident`
    - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - svm_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1cという名前のqtree Tridentでは、という名前のFlexVolのエクスポートポリシー、という名前のqtreeのエクスポートポリシー、`trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`およびという名前の空のエクスポートポリシー `trident_empty`がSVM上に作成されます `trident-403b5326-8482-40db96d0-d83fb3f4daec`。FlexVolエクスポートポリシーのルールは、qtreeエクスポートポリシーに含まれるすべてのルールのスーパーセットになります。空のエクスポートポリシーは、関連付けられていないボリュームで再利用されます。
- `autoExportCIDRs`アドレスブロックのリストが含まれます。このフィールドは省略可能で、デフォルト値は ["0.0.0.0/0" 、 "::/0" です。定義されていない場合、Tridentは、パブリケーションを使用して、ワーカーノード上で見つかったグローバルスコープのユニキャストアドレスをすべて追加します。

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

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

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

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

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

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

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

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

SMBボリュームをプロビジョニングする準備をします

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

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

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

作業を開始する前に

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

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

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

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

手順

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

   メモ | Amazon FSx for ONTAPにはSMB共有が必要です。

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

   a. 必要に応じて、共有のディレクトリパス構造を作成します。

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

   b. 指定したSVMに関連付けられているSMB共有を作成します。

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

   c. 共有が作成されたことを確認します。

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

   メモ | を参照してください "SMB 共有を作成" 詳細については、

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

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

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。

覚えておいてください

- 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 共有を無視します。
- 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。
- セキュアSMBを有効にする際は、有効なADユーザーを指定してください。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することで、セキュアSMBを確実に有効にすることができます。アノテーションに指定されたユーザー値は `trident.netapp.io/smbShareAdUser` で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。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の設定オプションと例

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

25.10リリース以降、 NetApp Tridentは以下をサポートします。"NetApp AFXストレージシステム" 。NetApp AFX ストレージ システムは、ストレージ層の実装において他のONTAPベースのシステム (ASA、 AFF、 FAS) とは異なります。

メモ	のみ `ontap-nas`ドライバー (NFS プロトコル付き) はNetApp AFX システムでサポートされていますが、SMB プロトコルはサポートされていません。

Tridentバックエンド構成では、システムがNetApp AFX ストレージ システムであることを指定する必要がありません。選択すると `ontap-nas`として `storageDriverName`Trident はAFX ストレージ 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 AFX ストレージ 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

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
「バージョン」		常に 1

パラメータ	説明	デフォルト
'torageDriverName'	ストレージドライバの名前 メモ NetApp AFXシステムの場合のみ`ontap-nas`サポートされています。	`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	ドライバ名+"_"+ dataLIF
「管理 LIF 」	クラスタまたはSVM管理LIFのIPアドレス完全修飾ドメイン名（FQDN）を指定できます。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`。シームレスなMetroClusterスイッチオーバーについては、を参照してMetroClusterの例ください。	"10.0.0.1 ","[2001：1234：abcd：：fe]"
「重複排除	プロトコル LIF の IP アドレス。を指定することを推奨しますNetApp `dataLIF`。指定しない場合、TridentはSVMからデータLIFをフェッチします。NFSのマウント処理に使用するFully Qualified Domain Name（FQDN；完全修飾ドメイン名）を指定すると、ラウンドロビンDNSを作成して複数のデータLIF間で負荷を分散できます。初期設定後に変更できます。を参照してください 。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`。* MetroClusterの場合は省略してください。*を参照してください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から取得されるアドレス（非推奨）
'VM'	使用する Storage Virtual Machine * MetroClusterの場合は省略してください。* MetroClusterの例。	SVM 「管理 LIF 」が指定されている場合に生成されます
「 autoExportPolicy 」を参照してください	エクスポートポリシーの自動作成と更新を有効にします[ブーリアン]。オプションと `autoExportCIDRs`オプションを使用する `autoExportPolicy`と、Tridentでエクスポートポリシーを自動的に管理できます。	いいえ
「 autoExportCI` 」	が有効な場合にKubernetesのノードIPをフィルタリングするCIDRのリスト `autoExportPolicy`。オプションと `autoExportCIDRs`オプションを使用する `autoExportPolicy`と、Tridentでエクスポートポリシーを自動的に管理できます。	["0.0.0.0/0","::/0"]'
「ラベル」	ボリュームに適用する任意の JSON 形式のラベルのセット	""
「 clientCertificate 」をクリックします	クライアント証明書の Base64 エンコード値。証明書ベースの認証に使用されます	""

パラメータ	説明	デフォルト
「clientPrivateKey」	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます	""
「trustedCacertifate」	信頼された CA 証明書の Base64 エンコード値。任意。証明書ベースの認証に使用されます	""
「ユーザ名」	クラスタ/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 」と入力します	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"。	
'toragePrefix'	SVM で新しいボリュームをプロビジョニングする際に使用するプレフィックスを指定します。設定後に更新することはできません メモ: qtree-nas-economyとstoragePrefixをONTAP 24文字以上で使用する場合、ボリューム名にはストレージプレフィックスは含まれませんが、qtreeにはストレージプレフィックスが埋め込まれます。	"トライデント"

パラメータ	説明	デフォルト
「集約」	プロビジョニング用のアグリゲート（オプション。設定する場合は SVM に割り当てる必要があります）。ドライバの場合 `ontap-nas-flexgroup`、このオプションは無視されます。割り当てられていない場合は 、使用可能ないずれかのアグリゲートを使用してFlexGroupボリュームをプロビジョニングできます。 メモ: SVMでアグリゲートが更新されると、Tridentコントローラを再起動せずにSVMをポーリングすることで、Tridentでアグリゲートが自動的に更新されます。ボリュームをプロビジョニングするようにTridentで特定のアグリゲートを設定している場合、アグリゲートの名前を変更するかSVMから移動すると、SVMアグリゲートのポーリング中にTridentでバックエンドが障害状態になります。アグリゲートをSVMにあるアグリゲートに変更するか、アグリゲートを完全に削除してバックエンドをオンラインに戻す必要があります。 **AFX** ストレージ システムには指定しないでください。	""
「AggreglimitateUsage 」と入力します	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*。**AFX** ストレージ システムには指定しないでください。	""（デフォルトでは適用されません）

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

パラメータ	説明	デフォルト
qtreesPerFlexvol`	FlexVol あたりの最大 qtree 数。有効な範囲は [50 、300] です。	"200"
`smbShare`	次のいずれかを指定できます。Microsoft管理コンソールまたはONTAP CLIを使用して作成されたSMB共有の名前、TridentでSMB共有を作成できるようにする名前、ボリュームへの共通の共有アクセスを禁止する場合はパラメータを空白のままにします。オンプレミスのONTAPでは、このパラメータはオプションです。このパラメータはAmazon FSx for ONTAPバックエンドで必須であり、空にすることはできません。	`smb-share`
「 useREST` 」	ONTAP REST API を使用するためのブール パラメーター。 `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`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。指定されている場合、常に **`true`AFX** ストレージ システム用。	`true` ONTAP 9.15.1以降の場合は、それ以外の場合は `false`。
`limitVolumePoolSize`	ONTAP NASエコノミーバックエンドでqtreeを使用する場合の、要求可能なFlexVolの最大サイズ。	"" （デフォルトでは適用されません）
`denyNewVolumePools`	を制限し `ontap-nas-economy`バックエンドがqtreeを格納するために新しいFlexVolボリュームを作成することます。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共有へのフルコントロール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用のバックエンド構成オプション

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

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

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

メモ	TridentでQoSポリシーグループを使用するには、ONTAP 9 .8以降が必要です。共有されていないQoSポリシーグループを使用し、ポリシーグループが各コンスティチュエントに個別に適用されるようにします。QoSポリシーグループを共有すると、すべてのワークロードの合計スループットの上限が適用されます。

ボリュームプロビジョニングの例

デフォルトが定義されている例を次に示します。

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

のために ontap-nas`そして `ontap-nas-flexgroups`Trident、新しい計算を使用して、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 内で要求した書き込み可能な領域を確実に受け取り、要求した領域よりも少ない領域を受け取ることがなくなります。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

計算は次のとおりです。

```
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。 `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。

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`以前の設定では、1GiBの書き込み可能領域を持つボリュームが作成されていました。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。

最小限の設定例

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

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

ONTAP NASエコノミーの例

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

ONTAP NAS FlexGroupの例

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

MetroClusterの例

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

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

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

SMBボリュームの例

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

証明書ベースの認証の例

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

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

自動エクスポートポリシーの例

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

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

IPv6アドレスの例

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

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

SMBボリュームを使用したAmazon FSx for ONTAPの例

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

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

nameTemplateを使用したバックエンド構成の例

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

仮想プールを使用するバックエンドの例

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

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

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

ONTAP NASの例

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

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

ONTAP NAS FlexGroupの例

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

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

ONTAP NASエコノミーの例

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

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

バックエンドを **StorageClasses** にマッピングします

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

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

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

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

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

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

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

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

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

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

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

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

更新 `dataLIF` 初期設定後

初期設定後にdataLIFを変更するには、次のコマンドを実行して新しいバックエンドJSONファイルに更新されたdataLIFを指定します。

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

メモ	PVCが1つ以上のポッドに接続されている場合、新しいデータLIF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稼働状態に戻す必要があります。

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
```

メモ

必ず追加してください `annotations`セキュア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
```

NetApp ONTAP 対応の Amazon FSX

Amazon FSx for NetApp ONTAPでTridentを使用

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

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

ファイルシステムは、オンプレミスの ONTAP クラスタに似た、 Amazon FSX のプライマリリソースです。各 SVM 内には、ファイルとフォルダをファイルシステムに格納するデータコンテナである 1 つ以上のボリュ

ームを作成できます。Amazon FSx for NetApp ONTAPは、クラウド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のタイプは * NetApp ONTAP * です。

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

要件

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

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

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

考慮事項

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

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

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には2つの認証モードがあります。

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

  警告

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

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

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

- "ONTAP NAS認証"
- "ONTAP SAN認証"

テスト済みの**Amazon**マシンイメージ（**AMIS**）

EKSクラスタ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ではコンテナとEKS用に特定のAmazon Machine Images（AMIS）が最適化されています。次の AMI は NetApp Trident 25.02 でテストされています。

亜美	NAS	NASエコノミー	iSCSI	iSCSIエコノミー
AL2023_x86_64_STANDARD	はい。	はい。	はい。	はい。
AL2_x86_64	はい。	はい。	はい*	はい*
BOTTLEROCKET_x86_64	はい**	はい。	N/A	N/A
AL2023_ARM_64_STANDARD	はい。	はい。	はい。	はい。
AL2_ARM_64	はい。	はい。	はい*	はい*

BOTTLEROCKET_ARM_64	はい**	はい。	N/A	N/A

- * 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。

メモ	目的のAMIがここにリストされていない場合、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ことを意味します。このリストは、AMI が動作することがわかっている場合のガイドとして機能します。

テスト実施項目：

- 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の両方をテストしました。
- SANについてはiSCSIのみをテストし、NVMe-oFはテストしませんでした。

実行されたテスト：

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

詳細については、こちらをご覧ください

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

IAMロールとAWS Secretを作成する

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

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

AWS Secrets Managerシークレットの作成

TridentはFSx SVMに対してAPIを発行してストレージを管理するため、そのためにはクレデンシャルが必要になります。これらのクレデンシャルを安全に渡すには、AWS Secrets Managerシークレットを使用します。そのため、AWS Secrets Managerシークレットをまだ作成していない場合は、vsadminアカウントのクレデンシャルを含むシークレットを作成する必要があります。

次の例では、Trident CSIクレデンシャルを格納する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ポリシーの作成

Tridentを正しく実行するには、AWSの権限も必要です。そのため、必要な権限をTridentに付与するポリシーを作成する必要があります。

次の例は、AWS CLIを使用してIAMポリシーを作成します。

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

Kubernetes サービスアカウントを設定して、EKS ポッド ID またはサービスアカウントの関連付け (IRSA) 用の IAM ロールを持つ AWS Identity and Access Management (IAM) ロールを引き受けることができます。これにより、この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、その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ポッド ID に対してこのロールを引き受けられるように、trust-relationship.json を作成します。次に、以下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** の関連付けを作成する:

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"
        }
      }
    }
  ]
}
```

ファイルの次の値を更新し `trust-relationship.json`ます。

- **<account_id>**-お客様のAWSアカウントID
- **<oidc_provider>**- EKSクラスタのOIDC。oidc_providerを取得するには、次のコマンドを実行します。

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

- IAMポリシーにIAMロールを関連付ける*：

ロールを作成したら、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関連付けます。

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

- OICDプロバイダが関連付けられていることを確認します*：

OIDCプロバイダがクラスタ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

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

次のいずれかの方法でTridentをインストールできます。

- Helm
- 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アドオンをインストールします。詳細については、を参照してください "CSIボリュームのスナップショット機能を有効にする" 。

Helmを使用した**Trident**のインストール

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
```

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

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                           APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    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
```

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

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2510.0    25.10.0
```

メモ

iSCSI を使用する場合は、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。AL2023Worker node OS を使用している場合は、helm インストール時に 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アドオンを使用して**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、バグ修正が含まれており、AWSによってAmazon EKSと連携することが検証されています。EKSアドオンを使用すると、Amazon EKSクラスタの安全性と安定性を一貫して確保し、アドオンのインストール、構成、更新に必要な作業量を削減できます。

前提条件

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

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

AWS向け**Trident**アドオンを有効にする

管理コンソール

1. でAmazon EKSコンソールを開きます https://console.aws.amazon.com/eks/home#/clusters。
2. 左側のナビゲーションペインで、*[クラスタ]*を選択します。
3. NetApp Trident CSIアドオンを設定するクラスタの名前を選択します。
4. *アドオン*を選択し、*追加のアドオン*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. [Configure selected add-ons* settings]ページで、次の手順を実行します。

メモ	**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. 使用する*バージョン*を選択します。
   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。
      - 使用する*バージョン*を選択します。
      - アドオン構成スキーマ に従って、構成値 セクションの **configurationValues** パラメータを、前の手順で作成した role-arn に設定します (値は次の形式である必要があります)。

```
{

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

}
```

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

7. [次へ]*を選択します。
8. [確認して追加]ページで、*[作成]*を選択します。

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

AWS CLI

1.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:

メモ　ビジネス

```
{
  "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"
  }
}
```

メモ　を、前の手順で作成したロールのARNに置き換えます `<role ARN>`。

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

次の例では、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アドオンの更新

管理コンソール

1. Amazon EKSコンソールを開き https://console.aws.amazon.com/eks/home#/clustersます。
2. 左側のナビゲーションペインで、*[クラスタ]*を選択します。
3. NetApp Trident CSIアドオンを更新するクラスタ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、 Edit *を選択します。
6. [Configure Trident by NetApp *]ページで、次の手順を実行します。
   a. 使用する*バージョン*を選択します。
   b. [Optional configuration settings]*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

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

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

  出力例：

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- 前の手順の出力でupdate availableで返されたバージョンにアドオンを更新します。

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

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

Trident EKSアドオンのアンインストール/削除

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

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

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

管理コンソール

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

AWS CLI

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

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

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の設定

ONTAP SANとNASドライバの統合

ストレージバックエンドを作成するには、JSONまたはYAML形式の構成ファイルを作成する必要があります。ファイルには、使用するストレージのタイプ（NASまたはSAN）、ファイルの取得元のファイルシステム、SVM、およびその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にクレデンシャルを格納する方法を示しています。

YAML

```
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:xxxxxxxx: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:xxxxxxxx: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）を作成および検証します。

- 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ドライバの詳細

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

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

ドライバーの詳細については、を参照してください。 "NASドライバ" および "SANドライバ"。

構成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。

```
kubectl 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   backend-fsx-ontap-nas   7a551921-997c-4c37-a1d1-
f2f4c87fa629   Bound   Success
```

バックエンドの高度な設定と例

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	例
「バージョン」		常に 1
'torageDriverName'	ストレージドライバの名前	`ontap-nas`、 `ontap-nas-economy`、 `ontap-nas-flexgroup`、 `ontap-san`、 `ontap-san-economy`
backendName`	カスタム名またはストレージバックエンド	ドライバ名+"_"+ dataLIF
「管理 LIF 」	クラスタまたはSVM管理LIFのIPアドレス完全修飾ドメイン名（FQDN）を指定できます。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、[28e8：d9fb：a825：b7bf：69a8：d02f：9e7b：3555]などの角かっこで定義する必要があります。 `aws`フィールドでを指定する場合は `fsxFilesystemID`、を指定する必要はありませ `managementLIF`ん。TridentはAWSからSVM情報を取得するためです。 `managementLIF`そのため、SVMの下のユーザ（vsadminなど）のクレデンシャルを指定し、そのユーザにロールが割り当てられている必要があり `vsadmin` ます。	"10.0.0.1 ","[2001：1234：abcd：：fe]"

パラメータ	説明	例
「重複排除	プロトコル LIF の IP アドレス。* ONTAP NASドライバ*：NetAppではdataLIFの指定を推奨しています。指定しない場合、TridentはSVMからデータLIFをフェッチします。NFSのマウント処理に使用するFully Qualified Domain Name（FQDN；完全修飾ドメイン名）を指定すると、ラウンドロビンDNSを作成して複数のデータLIF間で負荷を分散できます。初期設定後に変更できます。を参照してください 。* ONTAP SANドライバ*: iSCSIには指定しないでくださいTridentは、ONTAP選択的LUNマップを使用して、マルチパスセッションの確立に必要なiSCI LIFを検出します。データLIFが明示的に定義されている場合は警告が生成されます。IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、[28e8：d9fb：a825：b7bf：69a8：d02f：9e7b：3555]などの角かっこで定義する必要があります。	
「 autoExportPolicy 」を参照してください	エクスポートポリシーの自動作成と更新を有効にします[ブーリアン]。オプションと `autoExportCIDRs`オプションを使用する `autoExportPolicy` と、Tridentでエクスポートポリシーを自動的に管理できます。	「偽」
「 autoExportCI` 」	が有効な場合にKubernetesのノードIPをフィルタリングするCIDRのリスト `autoExportPolicy`。オプションと `autoExportCIDRs`オプションを使用する `autoExportPolicy` と、Tridentでエクスポートポリシーを自動的に管理できます。	"["0.0.0.0/0"、"：：/0"]"
「ラベル」	ボリュームに適用する任意の JSON 形式のラベルのセット	""
「 clientCertificate 」をクリックします	クライアント証明書の Base64 エンコード値。証明書ベースの認証に使用されます	""
「 clientPrivateKey 」	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます	""

パラメータ	説明	例
「 trustedCacertifate 」	信頼された CA 証明書の Base64 エンコード値。任意。証明書ベースの認証に使用されます。	""
「ユーザ名」	クラスタまたはSVMに接続するためのユーザ名。クレデンシャルベースの認証に使用されます。たとえば、vsadminのように指定します。	
「 password 」と入力します	クラスタまたはSVMに接続するためのパスワード。クレデンシャルベースの認証に使用されます。	
'VM'	使用する Storage Virtual Machine	SVM管理LIFが指定されている場合に生成されます。
'toragePrefix'	SVM で新しいボリュームをプロビジョニングする際に使用するプレフィックスを指定します。作成後に変更することはできません。このパラメータを更新するには、新しいバックエンドを作成する必要があります。	`trident`
「 AggreglimitateUsage 」と入力します	* Amazon FSx for NetApp ONTAP には指定しないでください。*指定されたと `vsadmin` には `fsxadmin`、アグリゲートの使用量を取得してTridentを使用して制限するために必要な権限が含まれていません。	使用しないでください。
「 limitVolumeSize 」と入力します	要求されたボリュームサイズがこの値を超えている場合、プロビジョニングが失敗します。また、qtreeおよびLUNに対して管理するボリュームの最大サイズを制限し、オプションを使用すると、FlexVol volumeあたりのqtreeの最大数をカスタマイズできます。 `qtreesPerFlexvol`	"" （デフォルトでは適用されません）
'lunsPerFlexvol	FlexVol volumeあたりの最大LUN数は[50、200]の範囲で指定する必要があります。SANのみ。	“100”
「バグトレースフラグ」	トラブルシューティング時に使用するデバッグフラグ。例： {"api"：false、"method"：true} 使用しないでください `debugTraceFlags` トラブルシューティングを実行していて、詳細なログダンプが必要な場合を除きます。	null

パラメータ	説明	例
「 nfsvMountOptions 」のように入力します	NFSマウントオプションをカンマで区切ったリスト。Kubernetes永続ボリュームのマウントオプションは通常ストレージクラスで指定されますが、ストレージクラスにマウントオプションが指定されていない場合、Tridentはストレージバックエンドの構成ファイルに指定されているマウントオプションを使用してフォールバックします。ストレージクラスまたは構成ファイルでマウントオプションが指定されていない場合、Tridentは関連付けられた永続ボリュームにマウントオプションを設定しません。	""
`nasType`	NFSボリュームまたはSMBボリュームの作成を設定オプションはです `nfs`、 `smb`、またはnull。*をに設定する必要があります `smb` SMBボリューム。*をnullに設定すると、デフォルトでNFSボリュームが使用されます。	`nfs`
qtreesPerFlexvol`	FlexVol volumeあたりの最大qtree数は[50、300]の範囲で指定する必要があります。	`"200"`
`smbShare`	次のいずれかを指定できます。Microsoft管理コンソールまたはONTAP CLIを使用して作成されたSMB共有の名前、またはTridentにSMB共有の作成を許可する名前。このパラメータは、Amazon FSx for ONTAPバックエンドに必要です。	`smb-share`
「 useREST` 」	ONTAP REST API を使用するためのブーリアンパラメータ。に設定する `true`と、TridentはONTAP REST APIを使用してバックエンドと通信します。この機能にはONTAP 9.11.1以降が必要です。また、使用するONTAPログインロールには、アプリケーションへのアクセス権が必要です `ontap` 。これは、事前に定義された役割と役割によって実現され `vsadmin` `cluster-admin` ます。	「偽」

パラメータ	説明	例
`aws`	AWS FSx for ONTAPの構成ファイルでは、次の項目を指定できます。 - `fsxFilesystemID`：AWS FSxファイルシステムのIDを指定します。 - `apiRegion`：AWS APIリージョン名。 - `apikey`：AWS APIキー。 - `secretKey`：AWSシークレットキー。	 "" "" ""
`credentials`	AWS Secrets Managerに保存するFSx SVMのクレデンシャルを指定します。- `name`：シークレットのAmazonリソース名（ARN）。SVMのクレデンシャルが含まれています。- `type`：に設定します `awsarn`。詳細については、を参照してください "AWS Secrets Managerシークレットの作成" 。	

ボリュームのプロビジョニング用のバックエンド構成オプション

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

パラメータ	説明	デフォルト
「平和の配分」	space-allocation for LUN のコマンドを指定します	「真」
「平和のための準備」を参照してください	スペースリザベーションモード：「none」（シン）または「volume」（シック）	「 NONE 」
「ナプショットポリシー」	使用する Snapshot ポリシー	「 NONE 」

パラメータ	説明	デフォルト
「 QOSPolicy 」	作成したボリュームに割り当てるQoS ポリシーグループ。ストレージプールまたはバックエンドごとに、QOSPolicyまたはadaptiveQosPolicyのいずれかを選択します。TridentでQoSポリシーグループを使用するには、ONTAP 9 .8以降が必要です。共有されていないQoSポリシーグループを使用し、ポリシーグループが各コンスティチュエントに個別に適用されるようにします。QoSポリシーグループを共有すると、すべてのワークロードの合計スループットの上限が適用されます。	""
「 adaptiveQosPolicy 」を参照してください	アダプティブ QoS ポリシーグループ：作成したボリュームに割り当てます。ストレージプールまたはバックエンドごとに、QOSPolicyまたはadaptiveQosPolicyのいずれかを選択します。経済性に影響するONTAP - NAS ではサポートされません。	""
「スナップショット予約」	Snapshot「0」用にリザーブされているボリュームの割合	がの `none`場合 `snapshotPolicy else、""
'plitOnClone	作成時にクローンを親からスプリットします	「偽」
「暗号化」	新しいボリュームでNetApp Volume Encryption（NVE）を有効にします。デフォルトはです。`false`このオプションを使用するには、クラスタで NVE のライセンスが設定され、有効になっている必要があります。バックエンドでNAEが有効になっている場合、Tridentでプロビジョニングされたすべてのボリュームで NAEが有効になります。詳細については、を参照してください"TridentとNVEおよびNAEとの連携"。	「偽」
`luksEncryption`	LUKS暗号化を有効にします。を参照してください "Linux Unified Key Setup（LUKS；統合キーセットアップ）を使用"。SANのみ。	""
階層ポリシー	使用する階層化ポリシー `none`	
「 unixPermissions 」	新しいボリュームのモード。* SMBボリュームは空にしておきます。*	""

パラメータ	説明	デフォルト
'ecurityStyle'	新しいボリュームのセキュリティ形式。NFSのサポート `mixed` および `unix` セキュリティ形式SMBはをサポートします `mixed` および `ntfs` セキュリティ形式	NFSのデフォルトはです `unix` 。SMBのデフォルトはです `ntfs`。

SMBボリュームのプロビジョニング

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とNASドライバの統合 次の手順を実行します。 "SMBボリュームをプロビジョニングする準備をします"。

ストレージクラスと**PVC**を設定する

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

ストレージクラスを作成する。

Kubernetes StorageClassオブジェクトの設定

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、必要なをストレージクラスに追加し`mountOptions`ます。

```
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
```

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

ストレージクラスを作成する。

手順

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

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

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

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

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

PVCの作成

```
https://kubernetes.io/docs/concepts/storage/persistent-
volumes["PersistentVolumeClaim_"^]（PVC）は、クラスタ上のPersistentVolumeへのアクセス要求です。
```

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

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。

マニフェストの例

PersistentVolumeClaimサンプルマニフェスト

次に、基本的なPVC設定オプションの例を示します。

RWXアクセスを備えた**PVC**

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

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: 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 を作成します。

```
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
```

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

Trident属性

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

属性	を入力します	値	提供	リクエスト	でサポートされます
メディア ^1	文字列	HDD 、ハイブリッド、 SSD	プールにはこのタイプのメディアが含まれています。ハイブリッドは両方を意味します	メディアタイプが指定されました	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup 、 ONTAPSAN 、 solidfire-san-SAN 、 solidfire-san-SAN のいずれかに対応しています
プロビジョニングタイプ	文字列	シン、シック	プールはこのプロビジョニング方法をサポートします	プロビジョニング方法が指定されました	シック：All ONTAP ；thin ：All ONTAP & solidfire-san-SAN
backendType	文字列	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、solidfire-san 、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ています	バックエンドが指定されて	すべてのドライバ

属性	を入力します	値	提供	リクエスト	でサポートされます
Snapshot	ブール値	true false	プールは、Snapshot を含むボリュームをサポートします	Snapshot が有効なボリューム	ontap-nas 、ontapさん、solidfireさん
クローン	ブール値	true false	プールはボリュームのクローニングをサポートします	クローンが有効なボリューム	ontap-nas 、ontapさん、solidfireさん
暗号化	ブール値	true false	プールでは暗号化されたボリュームをサポート	暗号化が有効なボリューム	ONTAP-NAS 、ONTAP-NAS-エコノミー 、ONTAP-NAS-FlexArray グループ、 ONTAP-SAN
IOPS	整数	正の整数	プールは、この範囲内で IOPS を保証する機能を備えています	ボリュームで IOPS が保証されました	solidfire - SAN

^1 ^ ： ONTAP Select システムではサポートされていません

サンプルアプリケーションのデプロイ

ストレージクラスとPVCが作成されたら、そのPVをポッドにマウントできます。ここでは、PVをポッドに接続するためのコマンドと設定例を示します。

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
```

次に、PVCをポッドに接続するための基本的な設定例を示します。基本設定：

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

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

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

```
kubectl exec -it 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
```

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

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

EKSクラスタでのTrident EKSアドオンの設定

NetApp Tridentは、KubernetesでAmazon FSx for NetApp ONTAPストレージ管理を合理化し、開発者や管理者がアプリケーションの導入に集中できるようにします。NetApp Trident EKSアドオンには、最新のセキュリティパッチ、バグ修正が含まれており、AWSによってAmazon EKSと連携することが検証されています。EKSアドオンを使用する

と、Amazon EKSクラスタの安全性と安定性を一貫して確保し、アドオンのインストール、構成、更新に必要な作業量を削減できます。

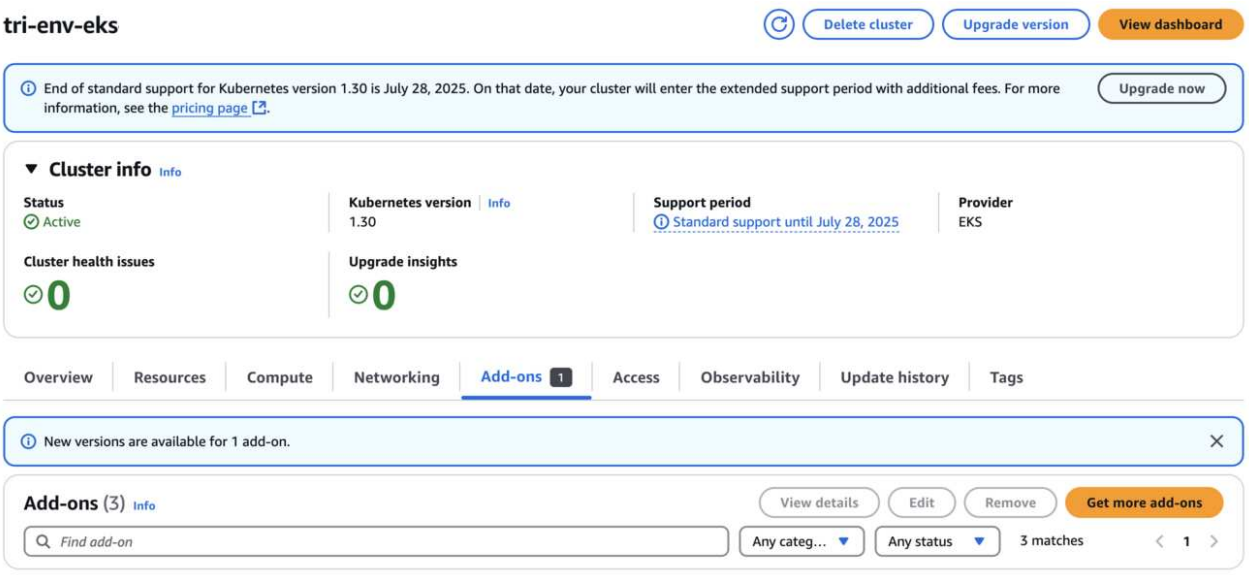
前提条件

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

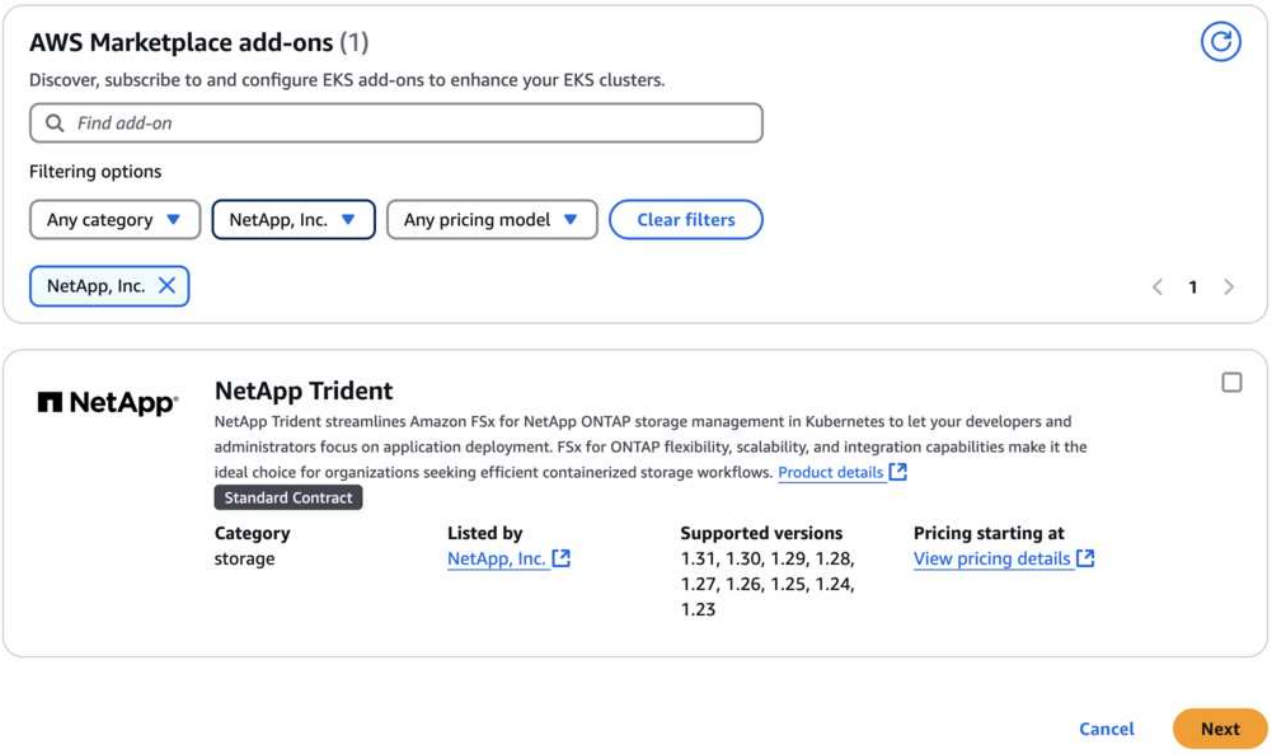
- アドオンを使用する権限を持つAmazon EKSクラスタアカウント。を参照してください "Amazon EKSアドオン"。
- AWS MarketplaceへのAWS権限：
  ```
  "aws-marketplace:ViewSubscriptions",
  "aws-marketplace:Subscribe",
  "aws-marketplace:Unsubscribe
  ```
- AMIタイプ：Amazon Linux 2（AL2_x86_64）またはAmazon Linux 2 Arm（AL2_ARM_64）
- ノードタイプ：AMDまたはARM
- 既存のAmazon FSx for NetApp ONTAPファイルシステム

手順

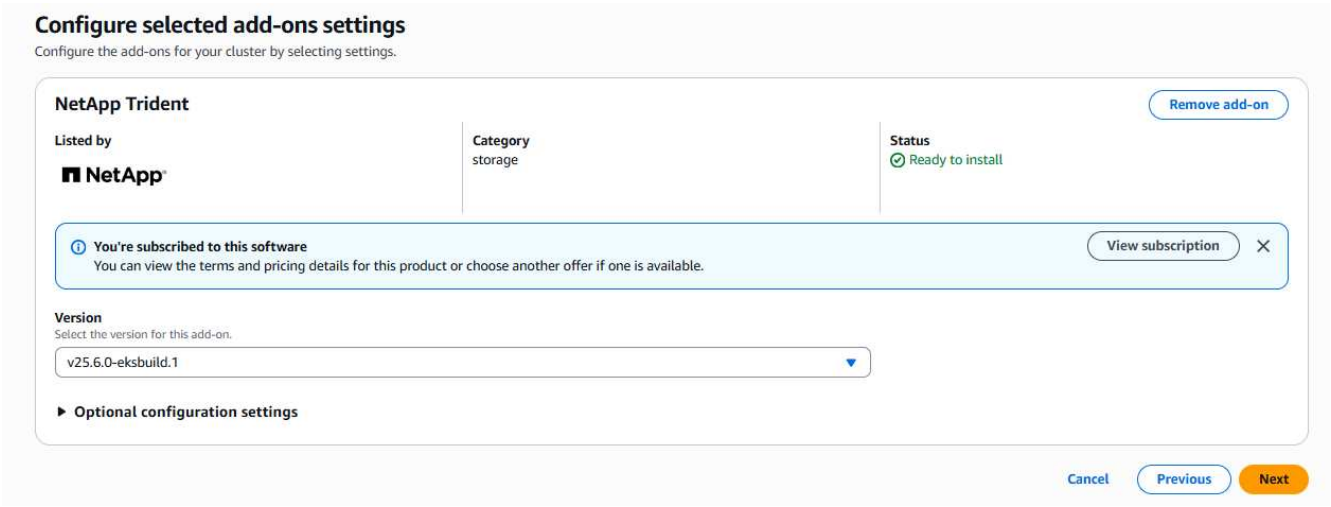
1. EKSポッドがAWSリソースにアクセスできるようにするために、IAMロールとAWSシークレットを作成してください。手順については、を参照してください"IAMロールとAWS Secretを作成する"。
2. EKS Kubernetesクラスタで、*[アドオン]*タブに移動します。



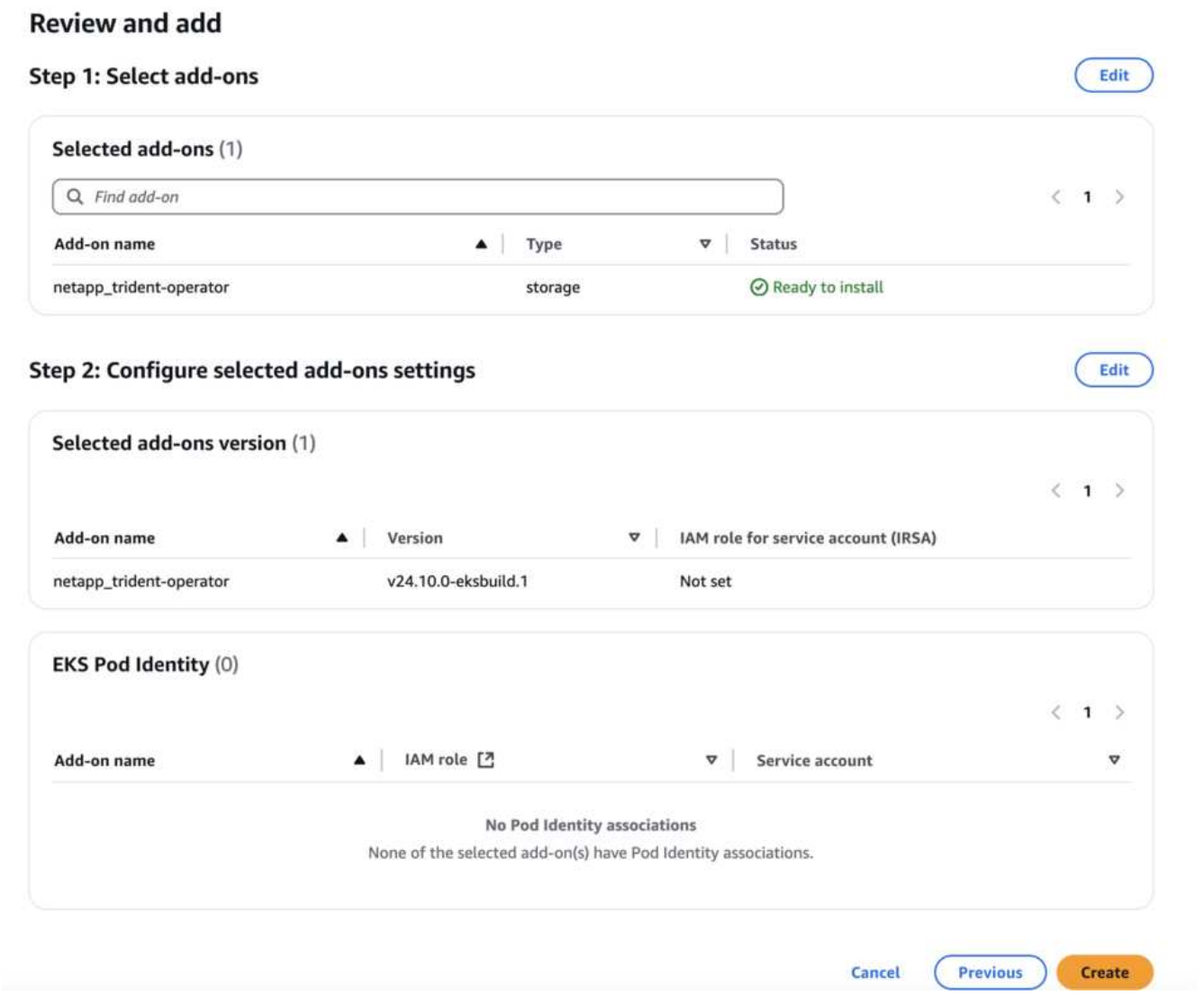
3. [AWS Marketplace add-ons]*にアクセスし、_storage_categoryを選択します。



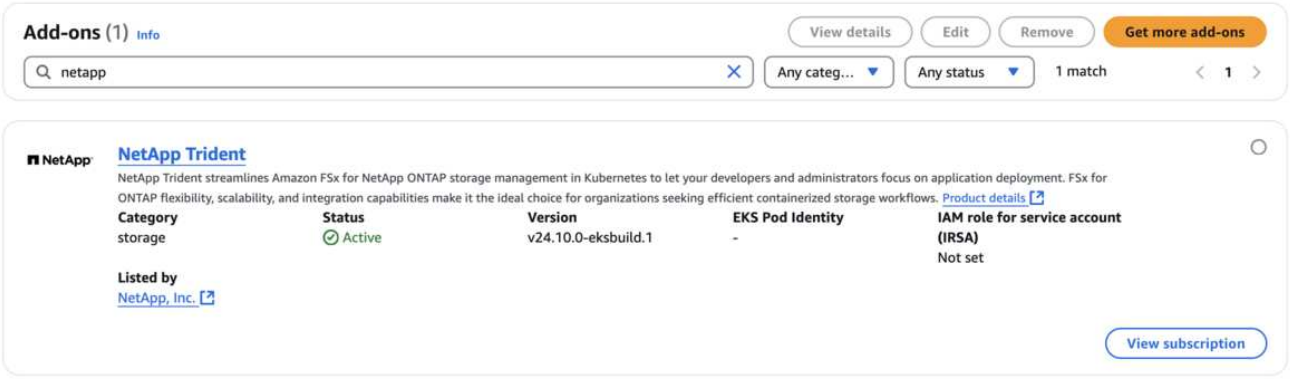
4. NetApp Trident を探し、**Trident**アドオンのチェックボックスを選択して Next *をクリックします。
5. 必要なアドオンのバージョンを選択します。



6. 必要なアドオン設定を構成します。



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。"こちらをご覧ください" 。
8. 「 * Create * 」を選択します。
9. アドオンのステータスが_Active_であることを確認します。



10. 次のコマンドを実行して、Tridentがクラスタ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設定します。詳細については、を参照してください "ストレージバックエンドの設定"。

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

CLIを使用して**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）

以下のコマンド例は Trident EKS アドオンバージョン 25.6.1 をインストールします：
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1`（専用バージョンを使用）

以下のコマンド例は Trident EKS アドオンバージョン 25.6.2 をインストールします：
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1`（専用バージョンを使用）

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 を使用してバックエンドを作成します

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

TridentBackendConfig

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

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

- バックエンドは、指定した設定に基づいてTridentによって自動的に作成されます。これは内部的には

(tbe、 tridentbackend）CRとして表され `TridentBackend`ます。

- は TridentBackendConfig、Tridentによって作成されたに一意にバインドされます TridentBackend。

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

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

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

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

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

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

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

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

は、「 TridentBackendConfig 」が作成された後で更新できます。

メモ	バックエンドの名前は 'PEC.backendName' を使用して設定されます指定しない場合、バックエンドの名前は「 TridentBackendConfig 」オブジェクト（ metadata.name ）の名前に設定されます。'PEC.backendName' を使用してバックエンド名を明示的に設定することをお勧めします
ヒント	で作成されたバックエンドに `tridentctl`は、関連付けられたオブジェクトはありません `TridentBackendConfig`。このようなバックエンドをで管理するには、 `kubectl`CRを作成し `TridentBackendConfig`ます。同一の設定パラメータ（、、 `spec.storagePrefix` `spec.storageDriverName`など）を指定するように注意する必要があります `spec.backendName`。Tridentは、新しく作成されたを既存のバックエンドに自動的にバインドし `TridentBackendConfig`ます。

手順の概要

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

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

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

手順 1 ： Kubernetes Secret を作成します

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

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

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: password
```

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

ストレージプラットフォームのシークレットフィールド概要	秘密	**Field** 概要の略
Azure NetApp Files の特長	ClientID	アプリケーション登録からのクライアント ID
Element （ NetApp HCI / SolidFire ）	エンドポイント	テナントのクレデンシャルを使用する SolidFire クラスタの MVIP
ONTAP	ユーザ名	クラスタ / SVM に接続するためのユーザ名。クレデンシャルベースの認証に使用されます
ONTAP	パスワード	クラスタ / SVM に接続するためのパスワード。クレデンシャルベースの認証に使用されます
ONTAP	clientPrivateKey	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます
ONTAP	chapUsername のコマンド	インバウンドユーザ名。useCHAP = true の場合は必須。「 ONTAP-SAN' 」と「 ONTAP-SAN-エコノミー 」の場合
ONTAP	chapInitiatorSecret	CHAP イニシエータシークレット。useCHAP = true の場合は必須。「 ONTAP-SAN' 」と「 ONTAP-SAN-エコノミー 」の場合
ONTAP	chapTargetUsername のコマンド	ターゲットユーザ名。useCHAP = true の場合は必須。「 ONTAP-SAN' 」と「 ONTAP-SAN-エコノミー 」の場合
ONTAP	chapTargetInitiatorSecret	CHAP ターゲットイニシエータシークレット。useCHAP = true の場合は必須。「 ONTAP-SAN' 」と「 ONTAP-SAN-エコノミー 」の場合

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

手順2：を作成します `TridentBackendConfig` CR

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

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

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

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

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

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

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

フェーズには次のいずれかの値を指定できます。

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

すべての PVC が削除された後にのみ削除されます。

- `lost` ：「 TridentBackendConfig 」 CR に関連付けられているバックエンドが誤って削除されたか、意図的に削除されました。「 TridentBackendConfig 」 CR には削除されたバックエンドへの参照があります。「 TridentBackendConfig 」 CR は、「 $eleetionPolicy 」の値に関係なく削除できます。
- `Unknown`：TridentはCRに関連付けられたバックエンドの状態または存在を特定できません `TridentBackendConfig`。たとえば、APIサーバが応答していない場合やCRDが見つからない場合 `tridentbackends.trident.netapp.io`などです。これには介入が必要な場合があります

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

（オプション）手順 4 ：詳細を確認します

バックエンドに関する詳細情報を確認するには、次のコマンドを実行します。

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

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

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

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

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

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

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

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

バックエンドの管理

kubectl を使用してバックエンド管理を実行します

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

バックエンドを削除します

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

次のコマンドを実行します。

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

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

既存のバックエンドを表示します

次のコマンドを実行します。

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

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

バックエンドを更新します

バックエンドを更新する理由はいくつかあります。

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

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

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

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

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

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

メモ

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

tridentctl を使用してバックエンド管理を実行します

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

バックエンドを作成します

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

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

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

```
tridentctl logs -n trident
```

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

バックエンドを削除します

Tridentからバックエンドを削除するには、次の手順を実行します。

1. バックエンド名を取得します。

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

2. バックエンドを削除します。

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

メモ	TridentでプロビジョニングされたボリュームとこのバックエンドからSnapshotが残っている場合、バックエンドを削除すると、そのバックエンドで新しいボリュームがプロビジョニングされなくなります。バックエンドは引き続き「Deleting」状態になります。

既存のバックエンドを表示します

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

- 概要を取得するには、次のコマンドを実行します。

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

- すべての詳細を確認するには、次のコマンドを実行します。

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

バックエンドを更新します

新しいバックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

バックエンドの更新が失敗した場合、バックエンドの設定に問題があるか、無効な更新を試行しました。次のコマンドを実行すると、ログを表示して原因を特定できます。

```
tridentctl logs -n trident
```

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

バックエンドを使用するストレージクラスを特定します

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

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

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

バックエンド管理オプション間を移動します

Tridentでバックエンドを管理するさまざまな方法について説明します。

バックエンドを管理するためのオプション

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

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

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

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

これは、次のシナリオに該当します。

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

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

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

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

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

手順 **0** ：バックエンドを特定します

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

```
tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

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

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
  "backendName": "ontap-nas-backend",
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "admin-password",
  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {
    "store": "nas_store"
  },
  "region": "us_east_1",
  "storage": [
    {
      "labels": {
        "app": "msoffice",
        "cost": "100"
      },
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {
        "app": "mysqldb",
        "cost": "25"
      },
      "zone": "us_east_1d",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
      }
    }
  ]
}
```

手順 **1** ： **Kubernetes Secret** を作成します

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

```
cat tbc-ontap-nas-backend-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password
```

```
kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

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

次の手順では '（この例のように）事前に存在する 'ONTAP-NAS-backend' に自動的にバインドされる 'TridentBackendConfig'CR を作成します次の要件が満たされていることを確認します。

- 「 'PEC.backendName' 」に同じバックエンド名が定義されています。
- 設定パラメータは元のバックエンドと同じです。
- 仮想プール（存在する場合）は、元のバックエンドと同じ順序である必要があります。
- クレデンシャルは、プレーンテキストではなく、 Kubernetes Secret を通じて提供されます。

この場合、「 TridentBackendConfig 」は次のようになります。

```
cat backend-tbc-ontap-nas.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'
```

```
kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

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

「 TridentBackendConfig 」が作成された後、そのフェーズは「バインド」されている必要があります。また、既存のバックエンドと同じバックエンド名と UUID が反映されている必要があります。

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                   BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

これで ' バックエンドは 'tbc-ontap/nas-backend`TridentBackendConfig' オブジェクトを使用して完全に管理されます

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

tridentBackendConfig を使用して作成されたバックエンドを一覧表示するには 'tridentctl を使用しますまた、管理者は、「 TridentBackendConfig 」を削除し、「 pec.deletionPolicy` 」が「 re 」に設定されていることを確認することで、「 tridentctl 」を使用してこのようなバックエンドを完全に管理することもできます。

手順 **0** ：バックエンドを特定します

たとえば ' 次のバックエンドが TridentBackendConfig を使用して作成されたとします

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

出力からはそのことがわかります `TridentBackendConfig` は正常に作成され、バックエンドにバインドされています（バックエンドのUUIDを確認してください）。

手順**1**：確認します `deletionPolicy` がに設定されます `retain`

の価値を見てみましょう `deletionPolicy`。これはに設定する必要があり `retain`ます。これにより、CRが削除されてもバックエンド定義が存在し、で管理できるように `TridentBackendConfig`なり `tridentctl`ます。

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

メモ 「削除ポリシー」が「再取得」に設定されていない限り、次の手順に進まないでください。

手順**2**：を削除します `TridentBackendConfig` **CR**

最後の手順は、「 TridentBackendConfig 」 CR を削除することです。「削除ポリシー」が「取得」に設定されていることを確認したら、削除を続行できます。

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

オブジェクトが削除されると、 `TridentBackendConfig`Tridentは実際にはバックエンド自体を削除せずにオブジェクトを削除します。

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権を有し、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権については、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。