



バックエンドの構成と管理

Trident

NetApp
January 14, 2026

目次

バックエンドの構成と管理	1
バックエンドを設定	1
Azure NetApp Files	1
Azure NetApp Files バックエンドを設定します	1
Azure NetApp Files バックエンドを設定する準備をします	5
Azure NetApp Files バックエンド構成のオプションと例	8
Google Cloud NetAppボリューム	20
Google Cloud NetApp Volumeバックエンドの設定	20
Google Cloud NetApp Volumeバックエンドを設定する準備	23
Google Cloud NetApp Volumeのバックエンド構成オプションと例	23
Google Cloudバックエンド用にCloud Volumes Service を設定します	37
Google Cloudドライバの詳細	37
TridentによるCloud Volumes Service for Google Cloudのサポートの詳細	38
バックエンド構成オプション	38
ボリュームプロビジョニングオプション	40
CVS -パフォーマンスサービスの種類の例	40
CVSサービスタイプの例	46
次の手順	48
NetApp HCI または SolidFire バックエンドを設定します	49
Elementドライバの詳細	49
開始する前に	49
バックエンド構成オプション	49
例1：3つのボリュームタイプを持つドライバのバックエンド構成 solidfire-san	50
例2：仮想プールを使用するドライバのバックエンドとストレージクラスの構成 solidfire-san	51
詳細情報	54
ONTAP SANドライバ	54
ONTAP SANドライバの概要	54
ONTAP SANドライバを使用してバックエンドを設定する準備をします	56
ONTAP SANの設定オプションと例	64
ONTAP NASドライバ	83
ONTAP NASドライバの概要	83
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	85
ONTAP NASの設定オプションと例	95
Amazon FSx for NetApp ONTAP	114
Amazon FSx for NetApp ONTAPでTridentを使用	114
IAMロールとAWS Secretを作成する	116
Trident をインストール	120
ストレージバックエンドの設定	127
ストレージクラスとPVCを設定する	137

サンプルアプリケーションのデプロイ	142
EKSクラスタでのTrident EKSアドオンの設定	143
kubectl を使用してバックエンドを作成します	148
TridentBackendConfig	148
手順の概要	150
手順 1：Kubernetes Secret を作成します	150
ステップ2：CRを作成する TridentBackendConfig	151
手順3：CRのステータスを確認する TridentBackendConfig	152
(オプション) 手順 4：詳細を確認します	153
バックエンドの管理	155
kubectl を使用してバックエンド管理を実行します	155
tridentctl を使用してバックエンド管理を実行します	156
バックエンド管理オプション間を移動します	158

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

Azure NetApp Files

Azure NetApp Files バックエンドを設定します

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

Azure NetApp Files ドライバの詳細

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

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

考慮事項

- Azure NetApp Filesサービスでは、50GiB未満のボリュームはサポートされません。より小さいボリュームを要求すると、Tridentは50GiBのボリュームを自動的に作成します。

- Tridentでは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみがサポートされます。

AKSの管理対象ID

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

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

Trident オペレータ

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

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

Helm

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

```
helm install trident trident-operator-100.2502.0.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、フラグをに `Azure` 設定し `cloudProvider` ます。

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

AKSのクラウドID

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

AzureでクラウドIDを活用するには、以下が必要です。

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

Trident オペレータ

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

例えば：

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

Helm

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

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

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

```
helm install trident trident-operator-100.2502.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code> tridentctl </code>

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

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

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

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

Azure NetApp Files バックエンドを設定する準備をします

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

NFSボリュームとSMBボリュームの前提条件

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

バックエンドを設定して使用するには ["Azure NetApp Files"](#)、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` AKS クラスタでクラウドIDを使用する場合はオプションです。
-
- 容量プール。を参照してください ["Microsoft : Azure NetApp Files 用の容量プールを作成します"。](#)
 - Azure NetApp Files に委任されたサブネット。を参照してください ["Microsoft : サブネットを Azure NetApp Files に委任します"。](#)
 - `subscriptionID` Azure NetApp Filesを有効にしたAzureサブスクリプションから削除します。
 - `tenantID` `clientID` `clientSecret` Azure NetApp Filesサービスへの十分な権限を持つ、Azure Active Directory内から["アプリケーション登録"](#)。アプリケーション登録では、次のいずれかを使用します。
 - 所有者ロールまたは寄与者ロール["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

```

    "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": []
}
]
}
}

```

- 少なくとも1つを含む "委任されたサブネット" Azure location。Trident 22.01では、この `location` パラメータはバックエンド構成ファイルの最上位レベル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。
- を使用するに Cloud Identity` は、から "ユーザーが割り当てた管理ID" を取得し `client ID`、でそのIDを指定します `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx`。

SMBボリュームに関するその他の要件

SMBボリュームを作成するには、以下が必要です。

- Active Directoryが設定され、Azure NetApp Files に接続されています。を参照してください ["Microsoft : Azure NetApp Files のActive Directory接続を作成および管理します"](#)。
- Linuxコントローラノードと少なくとも1つのWindowsワーカーノードでWindows Server 2022を実行しているKubernetesクラスタ。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 `smbccreds` :

```

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。を設定するには `csi-proxy`、Windowsで実行されているKubernetesノードについて、またはを["GitHub: Windows向けCSIプロキシ"](#)参照してください["GitHub: CSIプロキシ"](#)。

Azure NetApp Files バックエンド構成のオプションと例

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

バックエンド構成オプション

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



Tridentでは、手動QoS容量プールはサポートされません。

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

パラメータ	製品説明	デフォルト
version		常に 1
storageDriverName	ストレージドライバの名前	「azure-NetApp-files」
backendName	カスタム名またはストレージバックエンド	ドライバ名 + "_" + ランダムな文字
subscriptionID	AzureサブスクリプションからのサブスクリプションID管理されたIDがAKSクラスタで有効になっている場合はオプションです。	
tenantID	AKSクラスタで管理IDまたはクラウドIDが使用されている場合は、アプリ登録からのテナントIDはオプションです。	
clientID	管理対象IDまたはクラウドIDがAKSクラスタで使用されている場合、アプリ登録からのクライアントIDはオプションです。	
clientSecret	アプリ登録からのクライアントシークレット管理されたIDまたはクラウドIDがAKSクラスタで使用されている場合はオプションです。	
serviceLevel	、`Premium` または `Ultra` のいずれか `Standard`	"" (ランダム)
location	新しいボリュームが作成されるAzureの場所の名前AKSクラスタで管理IDが有効になっている場合はオプションです。	
resourceGroups	検出されたリソースをフィルタリングするためのリソースグループのリスト	"[]" (フィルタなし)

パラメータ	製品説明	デフォルト
netappAccounts	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	""
subnet	委任先のサブネットの名前 Microsoft.Netapp/volumes	""
networkFeatures	ボリュームのVNet機能のセットは `Basic`、または `Standard` です。 ネットワーク機能は一部の地域では使用できず、サブスクリプションで有効にする必要がある場合があります。この機能を有効にしないタイミングを指定する `networkFeatures` と、ボリュームのプロビジョニングが失敗します。	""
nfsMountOptions	NFSマウントオプションのきめ細かな制御。SMBボリュームでは無視されます。NFSバージョン4.1を使用してボリュームをマウントするには、カンマで区切ったマウントオプションのリストにを追加してNFS v4.1を `nfsvers=4` 選択します。ストレージクラス定義で設定されたマウントオプションは、バックエンド構成で設定されたマウントオプションよりも優先されます。	"nfsvers=3"
limitVolumeSize	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)
debugTraceFlags	トラブルシューティング時に使用するデバッグフラグ。例: `{"api": false, "method": true, "discovery": true}` トラブルシューティングを行って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 `nfs`、`smb` または `null` です。 `null` に設定すると、デフォルトで NFS ボリュームが使用されます。	nfs

パラメータ	製品説明	デフォルト
supportedTopologies	このバックエンドでサポートされているリージョンとゾーンのリストを表します。詳細については、を参照してください "CSI トポロジを使用します" 。	



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

必要な権限とリソース

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

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

``resourceGroups`、`netappAccounts`、`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リソースに制限するフィルタで、任意の組み合わせで指定できます。完全修飾名の形式は次のとおりです。`

タイプ	形式
リソースグループ	<code><リソースグループ></code>
ネットアップアカウント	<code><リソースグループ>/<ネットアップアカウント></code>
容量プール	<code><リソースグループ>/<ネットアップアカウント>/<容量プール></code>
仮想ネットワーク	<code><リソースグループ>/<仮想ネットワーク></code>
サブネット	<code><resource group>/<仮想ネットワーク>/<サブネット></code>

ボリュームのプロビジョニング

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

パラメータ	製品説明	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` IPv4アドレスまたはIPv4サブネットをCIDR表記で任意に組み合わせたリスト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"
snapshotDir	.snapshot ディレクトリの表示を制御します	NFSv4の場合は「true」 NFSv3の場合は「false」
size	新しいボリュームのデフォルトサイズ	"100G"
unixPermissions	新しいボリュームのUNIX権限 (8進数の4桁)。SMBボリュームでは無視されます。	"" (プレビュー機能、サブスクリプションでホワイトリスト登録が必要)

構成例

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

最小限の構成

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

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、`tenantID`、`clientID`、が`clientSecret`省略されてい`subscriptionID`、ます。これらは、管理対象IDを使用する場合はオプションです。

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

AKSのクラウドID

このバックエンド構成では、クラウドIDを使用する場合はオプションである、、`clientID`、が``clientSecret``省略されて``tenantID``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
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

容量プールフィルタを使用した特定のサービスレベル構成

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

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

高度な設定

このバックエンド構成は、ボリュームの配置を单一のサブネットにまで適用する手間をさらに削減し、一部のボリュームプロビジョニングのデフォルト設定も変更します。

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

仮想プール構成

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

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

サポートされるトポジ構成

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カー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: 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

以下の`StorageClass`定義は、上記のストレージプールを表しています。

フィールド名: `parameter.selector`

を使用する`parameter.selector`と、ボリュームのホストに使用する仮想プールごとにを指定できます`StorageClass`。ボリュームには、選択したプールで定義された要素があります。

```

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

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

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

```

SMBボリュームの定義例

`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。

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に対してフィルタを適用します。
`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>
```

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

```
tridentctl logs
```

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

Google Cloud NetAppボリューム

Google Cloud NetApp Volumeバックエンドの設定

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

Google Cloud NetApp Volumes ドライバの詳細

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

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

GKEのクラウドID

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

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

- GKEを使用して導入されるKubernetesクラスタ。
- GKEクラスタに設定されたワーカーロードID、およびノードプールに設定されたGKEメタデータサーバ。

- Google Cloud NetAppのボリューム管理者 (roles/gcp.admin NetApp) ロールまたはカスタムロールを持つGCPサービスアカウント。
- 新しいGCPサービスアカウントを指定するcloudProviderとcloudIdentityを含むTridentがインストールされます。以下に例を示します。

Trident オペレータ

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

例えば：

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

Helm

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

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

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

```
helm install trident trident-operator-100.2502.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

<code> tridentctl </code>

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

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

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

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

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

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

NFSボリュームノゼンティジョウケン

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 ["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	製品説明	デフォルト
version		常に 1

パラメータ	製品説明	デフォルト
storageDriverName	ストレージドライバの名前	の値は storageDriverName 「google-cloud-netapp-volumes」と指定する必要があります。
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。	
location	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	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。	
nfsMountOptions	NFS マウントオプションのきめ細かな制御。	"nfsvers=3 "
limitVolumeSize	要求されたボリュームサイズがこの値を超え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)
serviceLevel	ストレージプールとそのボリュームのサービスレベル。値は flex、standard、premium、または `extreme`です。	
network	GCNVボリュームに使用されるGoogle Cloudネットワーク。	
debugTraceFlags	トラブルシューティング時に使用するデバッグフラグ。例：`{"api":false, "method":true}` トラブルシューティングを行って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 nfs、`smb` またはnullです。 nullに設定すると、デフォルトでNFSボリュームが使用されます。	nfs

パラメータ	製品説明	デフォルト
supportedTopologies	<p>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：</p> <pre>supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a</pre>	

ボリュームプロビジョニングオプション

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

パラメータ	製品説明	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"
snapshotDir	ディレクトリへのアクセス .snapshot	NFSv4の場合は「true」 NFSv3の場合は「false」
snapshotReserve	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-----\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzzE4jK3b1/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzzE4jK3b1/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzzE4jK3b1/qp8B4Kws8zX5ojY9m\n
    XsYg6gyxy4zq70lwWgLwGa==\n
    -----END PRIVATE KEY-----\n

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

```

SMBホリュウムノセツティ

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
      https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
      https://www.googleapis.com/robot/v1/metadata/x509/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

StoragePools フィルタフショウシタセツティ

```

-----
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzzE4jK3b1/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzzE4jK3b1/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzzE4jK3b1/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq70lwWgLwGa==
    -----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-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlZZE4jK3b1/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlZZE4jK3b1/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlZZE4jK3b1/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlZZE4jK3b1/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カーノ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、["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に対してフィルタを適用します。
`nasType: nfs` または `nasType: null` NFSプールのフィルタ。

PVC定義の例

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

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

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

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

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

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

Google Cloudドライバの詳細

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

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

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

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

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

必要なもの

バックエンドを設定して使用するには "Cloud Volumes Service for Google Cloud"、次のものが必要です。

- NetApp Cloud Volumes Service で設定されたGoogle Cloudアカウント
- Google Cloud アカウントのプロジェクト番号
- ロールが割り当てられたGoogle Cloudサービスアカウント netappcloudvolumes.admin
- Cloud Volumes Service アカウントのAPIキーファイル

バックエンド構成オプション

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

パラメータ	製品説明	デフォルト
version		常に 1
storageDriverName	ストレージドライバの名前	"GCP-cvs"
backendName	カスタム名またはストレージバックエンド	ドライバ名 + "_" + API キーの一部
storageClass	CVSサービスタイプを指定するためのオプションのパラメータ。CVSサービスタイプを選択するために使用し `software` ます。それ以外の場合、TridentはサービスタイプがCVS-Performanceとみなされ(`hardware` ます)。	
storagePools	CVSサービスタイプのみ。ボリューム作成用のストレージプールを指定するオプションのパラメータ。	

パラメータ	製品説明	デフォルト
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。	
hostProjectNumber	共有VPCネットワークを使用する場合は必須です。このシナリオでは、`projectNumber`はサービスプロジェクト、`hostProjectNumber`はホストプロジェクトです。	
apiRegion	TridentがCloud Volumes Serviceボリュームを作成するGoogle Cloudリージョン。リージョン間Kubernetesクラスタを作成する場合、で作成したボリュームは apiRegion、複数のGoogle Cloudリージョンのノードでスケジュールされているワークフローで使用できます。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	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 netappcloudvolumes.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）。	
proxyURL	CVSアカウントへの接続にプロキシサーバが必要な場合は、プロキシURLを指定します。プロキシサーバには、HTTP プロキシまたは HTTPS プロキシを使用できます。HTTPS プロキシの場合、プロキシサーバで自己署名証明書を使用するために証明書の検証はスキップされます。認証が有効になっているプロキシサーバはサポートされていません。	
nfsMountOptions	NFS マウントオプションのきめ細かな制御。	"nfsvers=3 "
limitVolumeSize	要求されたボリュームサイズがこの値を超え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)
serviceLevel	新しいボリュームのCVS -パフォーマンスレベルまたはCVSサービスレベル。CVS-Performanceの値は standard、`'premium`または`'extreme`です。CVS値は`standardsw`または`'zoneredundantstandardsw`です。	CVS -パフォーマンスのデフォルトは「Standard」です。CVSのデフォルトは"standardsw"です。
network	Cloud Volumes Service ボリュームに使用するGoogle Cloudネットワーク。	デフォルト
debugTraceFlags	トラブルシューティング時に使用するデバッグフラグ。例：`'{"api":false, "method":true}` トラブルシューティングを行って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	製品説明	デフォルト
allowedTopologies	リージョン間アクセスを有効にするには、 のStorageClass定義 allowedTopologies にすべてのリージョンが含まれている必要があります。例： `- key: topology.kubernetes.io/region values: - us-east1 - europe-west1	

ボリュームプロビジョニングオプション

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

パラメータ	製品説明	デフォルト
exportRule	新しいボリュームのエクスポートルール。CIDR 表記の IPv4 アドレスまたは IPv4 サブネットの任意の組み合わせをカンマで区切って指定する必要があります。	"0.0.0.0/0"
snapshotDir	ディレクトリへのアクセス .snapshot	いいえ
snapshotReserve	Snapshot 用にリザーブされているボリュームの割合	"" (CVS のデフォルト値をそのまま使用)
size	新しいボリュームのサイズ。CVS - パフォーマンス最小値は100GiBです。CVS最小値は1GiBです。	CVS - パフォーマンスサービスのタイプはデフォルトで「100GiB」です。CVSサービスのタイプではデフォルトが設定されませんが、1GiB以上が必要です。

CVS -パフォーマンスサービスの種類の例

次の例は、CVS -パフォーマンスサービスタイプの設定例を示しています。

例 1：最小限の構成

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

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

例2：サービスレベルの設定

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

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

例3：仮想プールの構成

この例では、を使用して、`storage`仮想プールとを参照するを設定し`StorageClasses`ます。ストレージクラスの定義方法については、を参照して[\[ストレージクラスの定義\]](#)ください。

ここでは、すべての仮想プールに特定のデフォルトが設定されます。これにより、が5%に設定され、が`exportRule` `0.0.0.0/0`に設定され `snapshotReserve` ます。仮想プールは、セクションで定義し `storage` ます。個々の仮想プールはそれぞれ独自に定義され `serviceLevel`、一部のプールはデフォルト値を上書きします。仮想プールラベルを使用して、および`protection`に基づいてプールを区別しました `performance`。

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

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

ストレージクラスの定義

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

ストレージクラスの例

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

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

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

CVSサービスタイプの例

次の例は、CVSサービスタイプの設定例を示しています。

例1：最小構成

これは、CVSサービスタイプとデフォルトのサービスレベルを指定するために`standardsw`を使用する最小のバックエンド構成`storageClass`です。

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

例2：ストレージプールの構成

このバックエンド構成の例では、を使用して `storagePools` ストレージプールを構成しています。

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

次の手順

バックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

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

```
tridentctl logs
```

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

NetApp HCI または SolidFire バックエンドを設定します

Trident環境でElementバックエンドを作成して使用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
<code>solidfire-san</code>	iSCSI	ブロック	RWO、ROX、RWX、RWOP	ファイルシステムがありません。 <code>raw</code> ブロックデバイスです。
<code>solidfire-san</code>	iSCSI	ファイルシステム	RWO、RWOP	<code>xfs</code> 、 <code>ext3</code> 、 <code>ext4</code>

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	製品説明	デフォルト
<code>version</code>		常に 1

パラメータ	製品説明	デフォルト
storageDriverName	ストレージドライバの名前	常に「SolidFire - SAN」
backendName	カスタム名またはストレージバックエンド	「SolidFire _」+ストレージ(iSCSI) IPアドレス
Endpoint	テナントのクレデンシャルを使用するSolidFireクラスタのMVIP	
SVIP	ストレージ(iSCSI)のIPアドレスとポート	
labels	ボリュームに適用する任意のJSON形式のラベルのセット。	""
TenantName	使用するテナント名(見つからない場合に作成)	
InitiatorIFace	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
AccessGroups	使用するアクセスグループIDのリスト	「Trident」という名前のアクセスグループのIDを検索します。
Types	QoSの仕様	
limitVolumeSize	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)
debugTraceFlags	トラブルシューティング時に使用するデバッグフラグ。例: {"api": false, "method": true}	null



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

例1：3つのボリュームタイプを持つドライバのバックエンド構成 solidfire-san

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。次に、ストレージクラスパラメータを使用して各ストレージクラスを使用するように定義します IOPS。

```
---
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：仮想プールを使用するドライバのバックエンドとストレージクラスの構成 solidfire-san

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。

以下に示すサンプルのバックエンド定義ファイルでは、すべてのストレージプールに特定のデフォルトが設定されており、そのデフォルトはAt Silverに設定されて `type` います。仮想プールは、セクションで定義し `storage` ます。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。フィールドを使用して

parameters.selector、各StorageClass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

```

詳細情報

- ・"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 の 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。	ext4、ext3、ext3

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 / TCP 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、RW OP	ファイルシステムなし。rawブロックデバイスです
ontap-san	NVMe / TCP 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-economy	iSCSI	ブロック	RWO、ROX、RWX、RW OP	ファイルシステムなし。rawブロックデバイスです
ontap-san-economy	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

- 永続的ボリュームの使用数がよりも多くなると予想される場合にのみ使用します `ontap-san-economy` ["サポートされるONTAPの制限"](#)。
- 永続的ボリュームの使用数がよりも多いと予想され、`ontap-san-economy` ドライバを使用できない場合にのみ ["サポートされるONTAPの制限"](#) 使用して `ontap-nas-economy` ください。
- データ保護、ディザスタリカバリ、モビリティの必要性が予想される場合は使用しない `ontap-nas-economy` でください。
- 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。



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るDH-HMAC-CHAP
- Device Mapper (DM ; デバイスマッパー) マルチパス
- LUKS暗号化

ONTAP SANドライバを使用してバックエンドを設定する準備をします

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。

ASA r2システムでSVMにアグリゲートを割り当てる方法については、次のナレッジベースの記事を参照してください。["SVM管理者がCLIを使用してストレージユニットを作成すると、「ストレージサービスに使用できる候補アグリゲートがありません」というエラーが発生して失敗します。"](#)

複数のドライバを実行し、1つまたは複数のドライバを参照するストレージクラスを作成することもできます

す。たとえば、ドライバを使用するクラス `ontap-san` と、ドライバ `san-default` を使用するクラスを `ontap-san-economy` 設定できます `san-dev`。

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。詳細については、を参照してください ["ワーカーノードを準備します"](#)。

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 : 必要な権限を持つONTAPユーザのユーザ名とパスワード。ONTAPのバージョンと最大限の互換性を確保するために、や `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 `admin` ます。
- 証明書ベース : Tridentは、バックエンドにインストールされている証明書を使用してONTAPクラスタと通信することもできます。この場合、バックエンド定義には、Base64でエンコードされたクライアント証明書、キー、および信頼されたCA証明書（推奨）が含まれている必要があります。

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



クレデンシャルと証明書の両方を*指定しようとすると、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を有効にする

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

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

一般的なワークフローは次の手順で構成されます。

手順

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

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

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

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

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

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

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

```
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": "Faaaakkkeeee...Vaaallluuuueeee",
  "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、実行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 `tridentctl backend update` ます。

```

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

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

```

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

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

Trident用のカスタムONTAPロールの作成

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

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

ONTAP CLI ノショウ

1. 次のコマンドを使用して新しいロールを作成します。

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

2. Tridentユーザのユーザ名を作成します。

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

3. ユーザにロールをマッピングします。

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

System Managerの使用

ONTAPシステムマネージャで、次の手順を実行します。

1. カスタムロールの作成：

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

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

- b. の横にある矢印アイコン (→*) を選択します。

- c. [Roles]*で[+Add]*を選択します。

- d. ロールのルールを定義し、*[保存]*をクリックします。

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

- a. で【アイコンの追加】+*を選択します。

- b. 必要なユーザ名を選択し、* Role *のドロップダウンメニューでロールを選択します。

- c. [保存 (Save)]をクリックします。

詳細については、次のページを参照してください。

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

双方向 CHAP を使用して接続を認証します

Tridentでは、ドライバと ontap-san-economy、ドライバの双方向CHAPを使用してiSCSIセッションを認証できます `ontap-san`。これには、バックエンド定義でオプションを有効にする必要があり `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: c19qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```



`useCHAP` パラメータはブール値のオプションで、一度だけ設定できます。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。

さらに `useCHAP=true`、`chapInitiatorSecret`、`chapTargetInitiatorSecret`、`chapTargetUsername`、および `chapUsername` フィールドを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、を実行してバックエンドを作成したあとに変更できます `tridentctl update`。

仕組み

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

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

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

クレデンシャルをローテーションし、バックエンドを更新

CHAP クレデンシャルを更新するには、ファイルの CHAP パラメータを更新し `backend.json` ます。そのためには、CHAP シークレットを更新し、コマンドを使用して変更を反映する必要があります `tridentctl update` ます。

バックエンドのCHAPシークレットを更新する場合は、を使用してバックエンドを更新する必要があります `tridentctl`。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": "c19qxUpDaTeD",
  "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あります。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 SANの設定オプションと例

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。

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	製品説明	デフォルト
version		常に 1
storageDriveName	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
managementLIF	<p>クラスタ管理LIFまたはSVM管理LIFのIPアドレス。</p> <p>Fully Qualified Domain Name (FQDN; 完全修飾ドメイン名) を指定できます。</p> <p>IPv6フラグを使用してTridentがインストールされている場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります</p> <p>[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。</p> <p>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。</p> <p> 「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。</p>	"10.0.0.1 ", "[2001:1234:abcd:fe]"
dataLIF	<p>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</p> <p>[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。</p>	SVMの派生物です
svm	使用するStorage Virtual Machine * MetroClusterでは省略*を参照してください MetroClusterの例 。	SVMが指定されている場合に派生managementLIF

パラメータ	製品説明	デフォルト
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 "ONTAP SAN ドライバを使用してバックエンドを設定する準備をします" 。	false
chapInitiatorSecret	CHAP イニシエータシークレット。必要な場合 useCHAP=true	""
labels	ボリュームに適用する任意の JSON 形式のラベルのセット	""
chapTargetInitiatorSecret	CHAP ターゲットイニシエータシークレット。必要な場合 useCHAP=true	""
chapUsername	インバウンドユーザ名。必要な場合 useCHAP=true	""
chapTargetUsername	ターゲットユーザ名。必要な場合 useCHAP=true	""
clientCertificate	クライアント証明書の Base64 エンコード値。証明書ベースの認証に使用されます	""
clientPrivatekey	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます	""
trustedCACertificate	信頼された CA 証明書の Base64 エンコード値。オプション。証明書ベースの認証に使用されます。	""
username	ONTAP クラスタとの通信に必要なユーザ名。クレデンシャルベースの認証に使用されます。	""
password	ONTAP クラスタとの通信にパスワードが必要です。クレデンシャルベースの認証に使用されます。	""
svm	使用する Storage Virtual Machine	SVMが指定されている場合に派生 managementLIF
storagePrefix	SVM で新しいボリュームをプロビジョニングする際に使用するプレフィックスを指定します。あとから変更することはできません。このパラメータを更新するには、新しいバックエンドを作成する必要があります。	trident

パラメータ	製品説明	デフォルト
aggregate	<p>プロビジョニング用のアグリゲート（オプション。設定する場合は SVM に割り当てる必要があります）。ドライバの場合 <code>ontap-nas-flexgroup</code>、このオプションは無視されます。割り当てられていない場合は、使用可能ないずれかのアグリゲートを使用してFlexGroupボリュームをプロビジョニングできます。</p> <p> SVMでアグリゲートが更新されると、Tridentコントローラを再起動せずにSVMをポーリングすることで、Tridentでアグリゲートが自動的に更新されます。ボリュームをプロビジョニングするようにTridentで特定のアグリゲートを設定している場合、アグリゲートの名前を変更するかSVMから移動すると、SVMアグリゲートのポーリング中にTridentでバックエンドが障害状態になります。アグリゲートをSVMにあるアグリゲートに変更するか、アグリゲートを完全に削除してバックエンドをオンラインに戻す必要があります。</p> <ul style="list-style-type: none"> ASA R2*には指定しないでください。 	""
limitAggregateUsage	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 <code>limitAggregateUsage</code> 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*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	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
debugTraceFlags	トラブルシューティング時に使用するデバッグフラグ。例： <code>{"api": false, "method": true}</code> 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、は使用しないでください。	null

パラメータ	製品説明	デフォルト
useREST	<p>ONTAP REST API を使用するためのブーリアンパラメータ。</p> <p>useREST`に設定する `true`と、Trident はONTAP REST APIを使用してバックエンドと通信します。に設定する `false`と、Trident はONTAPI (ZAPI) 呼び出し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`ontapi`。これは、事前に定義された役割と役割によって実現され vsadmin cluster-admin ます。Trident 24.06リリースおよびONTAP 9.151以降では、が`useREST`デフォルトでに設定されて `true`います。 `false`ONTAPI (ZAPI) 呼び出しを使用するようにに変更してください。</p> <p>`useREST`</p> <p>useREST はNVMe/TCPに完全修飾されています。*指定されている場合は、ASA R2*の場合は常に設定され `true`ます。</p>	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
formatOptions	<p>を使用して、`formatOptions`コマンドのコマンドライン引数を指定します。この引数 `mkfs`は、ボリュームがフォーマットされるたびに適用されます。これにより、好みに応じてボリュームをフォーマットできます。デバイスパスを除いて、mkfsコマンドオプションと同様にformatOptionsを指定してください。例：「-E nodiscard」</p> <ul style="list-style-type: none"> `ontap-san`および`ontap-san-economy` ドライバでのみサポートされています。* 	
limitVolumePoolSize	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では、フォーマット処理を高速化するために、次のオプションを推奨しています。

-E nodiscard :

- **keep** : mkfsの時点でブロックを破棄しないでください (ブロックの破棄は、最初はソリッドステートデバイスやスパース/シンプロビジョニングされたストレージで有効です)。これは廃止されたオプション「-

K」に代わるもので、すべてのファイルシステム（xfs、ext3、およびext4）に適用できます。

ボリュームのプロビジョニング用のバックエンド構成オプション

設定のセクションで、これらのオプションを使用してデフォルトのプロビジョニングを制御できます defaults。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	製品説明	デフォルト
spaceAllocation	space-allocation for LUN のコマンドを指定します	「true」*を指定すると、ASA R2* の場合はに設定され `true` ます。
spaceReserve	スペースリザベーションモード：「none」（シン）または「volume」（シック）。* ASA R2* の場合はに設定し `none` ます。	"なし"
snapshotPolicy	使用するSnapshotポリシー。* ASA R2* の場合はに設定し `none` ます。	"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イ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	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	""
snapshotReserve	Snapshot用にリザーブされているボリュームの割合。* ASA R2* には指定しないでください。	が「none」の場合は「0」 snapshotPolicy、それ以外の場合は「」
splitOnClone	作成時にクローンを親からスプリットします	いいえ
encryption	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 "。	「false」*を指定すると、ASA R2* の場合はに設定されます true。
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 "。	"" ASA R2の場合はに設定します false。
tieringPolicy	「none」を使用する階層化ポリシー* ASA R2 *には指定しないでください。	

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

ボリュームプロビジョニングの例

デフォルトが定義されている例を次に示します。

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

① 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
```

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

Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
	_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4		online	RW	10GB	5.00GB	0%
	_pvc_e42ec6fe_3baa_4af6_996d_134adb8e6d		online	RW	5.79GB	5.50GB	0%
	_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba		online	RW	1GB	511.8MB	0%
3 entries were displayed.							

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

最小限の設定例

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



TridentでAmazon FSx on NetApp ONTAPを使用している場合、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

これはドライバを使用した基本的な設定です `ontap-san`。

```
---
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

スイッチオーバー後およびスイッチバック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`。例えば：

```
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）が入力され `backend.json`、それぞれクライアント証明書、秘密鍵、および信頼されたCA証明書のbase64でエンコードされた値が使用されます。

```
---  
version: 1  
storageDriverName: ontap-san  
backendName: DefaultSANBackend  
managementLIF: 10.0.0.1  
svm: svm_iscsi  
useCHAP: true  
chapInitiatorSecret: c19qxIm36DKyawxy  
chapTargetInitiatorSecret: rqxigXgkesIpwxyz  
chapTargetUsername: iJF4heBRT0TCwxyz  
chapUsername: uh2aNCLSd6cNwxyz  
clientCertificate: ZXROZXJwYXB...ICMgJ3BhcGVyc2  
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX  
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

双方向CHAPの例

これらの例では、がに設定され`true`たバックエンドが作成され`useCHAP`ます。

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: svml
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、none、spaceAllocation`false、false`encryption`など、すべてのストレージプールに特定のデフォルトが設定されています`spaceReserve。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で独自の、`spaceAllocation`および`encryption`の値が設定され、`spaceReserve`、一部のプールでデフォルト値が上書きされます。

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: mysql ldb  
    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、を参照して[\[仮想プールを使用するバックエンドの例\]](#)ください。フィールドを使用して parameters.selector、各StorageClass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。

- protection-gold`StorageClassはバックエンドの最初の仮想プールにマッピングされます`ontap-san。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、バックエンドの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。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はバックエンドの3番目の仮想プールにマッピングされます `ontap-san-economy。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はバックエンドの2番目の仮想プールにマッピングされます `ontap-san。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、バックエンドの3番目の仮想プールとバックエンドの4番目の仮想プール `ontap-san-economy`にマッピングされます `ontap-san。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、を使用してドライバの `sanType: nvme`仮想プールに `ontap-san`マッピングされます `testAPP。これは唯一のプール `testApp`です。

```

---
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-economy	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

- 永続的ボリュームの使用数がよりも多くなると予想される場合にのみ使用します`ontap-san-economy`["サポートされるONTAPの制限"](#)。
- 永続的ボリュームの使用数がよりも多いと予想され、`ontap-san-economy` ドライバを使用できない場合にのみ["サポートされるONTAPの制限"](#)を使用して`ontap-nas-economy`ください。
- データ保護、ディザスタリカバリ、モビリティの必要性が予想される場合は使用しない`ontap-nas-economy`でください。
- 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期的なユーザです。

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

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

ONTAP NAS ドライバを使用してバックエンドを設定する準備をします

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

要件

- すべてのONTAP バックエンドでは、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、ドライバを使用するGoldクラスと、ドライバを使用するBronzeクラスを `ontap-nas-economy`、`ontap-nas` 設定できます。
- 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。["ここをクリック" 詳細](#)については、を参照してください。
- 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。ONTAP のバージョンと最大限の互換性を確保するために、や `vsadmin`、などの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 `admin`。
- 証明書ベース：このモードでは、Trident がONTAP クラスタと通信するために、バックエンドに証明書をインストールする必要があります。この場合、バックエンド定義には、Base64 でエンコードされたクライアント証明書、キー、および信頼された CA 証明書（推奨）が含まれている必要があります。

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



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

クレデンシャルベースの認証を有効にします

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

バックエンド定義の例は次のようにになります。

YAML

```
---
```

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

JSON

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

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

証明書ベースの認証を有効にします

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

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

一般的なワークフローは次の手順で構成されます。

手順

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

Name (CN ; 共通名) を設定します。

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

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

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

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

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

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

```
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に設定されていることを確認する必要があります `default-data-management` 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": "Faaaakkkeeee...Vaaallluuuueeee",
  "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、実行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 `tridentctl update backend` 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",
  "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 |          |
+-----+-----+-----+
+-----+-----+
```

i 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けます。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 ノショウ

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 자체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限り、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く、指定された範囲のIPを持つワーカーノードだけにストレージクラスタへのアクセスを制限し、きめ細かく自動化された管理をサポートします。



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

例

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

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



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

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

- `autoExportPolicy` が設定されている `true` です。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。
 - 例えば：
 - バックエンド UUID 403b5326-8482-40dB-96d0-d83fb3f4daec
 - `autoExportPolicy` に設定 `true`
 - ストレージプレフィックス `trident`
 - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
 - `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。

バックエンドを作成した後で、バックエンドのおよびを `autoExportCIDRs` 更新できます `autoExportPolicy`。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無効にして、手動で作成したエクスポートポリシーにフォールバックすることもできます `autoExportPolicy`。この場合、バックエンド設定でパラメータを設定する必要があります `exportPolicy`。

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。



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

少し準備をするだけで、ドライバを使用してSMBボリュームをプロビジョニングできます `ontap-nas`。



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。



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

開始する前に

SMBボリュームをプロビジョニングする前に、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 `smbccreds` :

```
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。を設定するには `csi-proxy`、Windowsで実行されているKubernetesノードについて、またはを["GitHub: Windows向けCSIプロキシ"](#)参照してください["GitHub: CSIプロキシ"](#)。

手順

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



Amazon FSx for ONTAPにはSMB共有が必要です。

SMB管理共有は、共有フォルダスナップインを使用するか、ONTAP CLIを使用して作成できます["Microsoft管理コンソール"](#)。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。

コマンドは `vserver cifs share create`、共有の作成時に `-path` オプションで指定されたパスをチェックします。指定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
```



詳細については、を参照して["SMB共有を作成する"](#)ください。

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。FSx for ONTAPのバックエンド構成オプションについては、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	製品説明	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	*に設定する必要があります <code>smb</code> 。* <code>null</code> の場合、デフォルトはになります <code>nfs</code> 。	<code>smb</code>
securityStyle	新しいボリュームのセキュリティ形式。* SMBボリュームの場合はまたは <code>mixed</code> 、に設定する必要があります <code>ntfs</code> 。*	<code>ntfs</code> 、SMBボリュームの場合はまたは <code>mixed</code>
unixPermissions	新しいボリュームのモード。* SMBボリュームは空にしておく必要があります。*	""

ONTAP NASの設定オプションと例

Tridentのインストール時にONTAP NASドライバを作成して使用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。

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	製品説明	デフォルト
version		常に 1
storageDriveName	ストレージドライバの名前	<code>ontap-nas</code> 、 <code>ontap-nas-economy</code> 、または <code>ontap-nas-flexgroup</code>
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
managementLIF	クラスタまたはSVM管理LIFのIPアドレス完全修飾ドメイン名 (FQDN) を指定できます。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、のように角っこで定義する必要があります <code>[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]</code> 。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。	<code>"10.0.0.1","[2001:1234:abcd:fe]"</code>

パラメータ	製品説明	デフォルト
dataLIF	<p>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、のように角っこで定義する必要があります</p> <p>[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。* MetroClusterの場合は省略してください。*を参照してくださいMetroClusterの例。</p>	指定されたアドレス、または指定されていない場合はSVMから取得されるアドレス (非推奨)
svm	使用するStorage Virtual Machine * MetroClusterでは省略*を参照してください MetroClusterの例 。	SVMが指定されている場合に派生managementLIF
autoExportPolicy	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。	正しくない
autoExportCIDRs	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"]
labels	ボリュームに適用する任意の JSON 形式のラベルのセット	""
clientCertificate	クライアント証明書の Base64 エンコード値。証明書ベースの認証に使用されます	""
clientPrivateKey	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます	""
trustedCACertificate	信頼された CA 証明書の Base64 エンコード値。オプション。証明書ベースの認証に使用されます	""
username	クラスタ / SVM に接続するためのユーザ名。クレデンシャルベースの認証に使用されます	
password	クラスタ / SVM に接続するためのパスワード。クレデンシャルベースの認証に使用されます	

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

パラメータ	製品説明	デフォルト
flexgroupAggregateList	<p>プロビジョニング用のアグリゲートのリスト（オプション）。設定されている場合はSVMに割り当てる必要があります）。SVMに割り当てられたすべてのアグリゲートを使用して、FlexGroupボリュームがプロビジョニングされます。ONTAP - NAS - FlexGroup *ストレージドライバーでサポートされています。</p> <p> SVMでアグリゲートリストが更新されると、Tridentコントローラを再起動せずにSVMをポーリングすること で、Trident内のアグリゲートリストが自動的に更新されます。ボリュームをプロビジョニングするようにTridentで特定のアグリゲートリストを設定している場合、アグリゲートリストの名前を変更するかSVMから移動すると、Tridentアグリゲートのポーリング中にバックエンドが障害状態になります。アグリゲートリストをSVM上のアグリゲートリストに変更するか、アグリゲートリストを完全に削除してバックエンドをオンラインに戻す必要があります。</p>	""
limitVolumeSize	<p>要求されたボリュームサイズがこの値を超えている場合、プロビジョニングが失敗します。また、qtree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あたりの最 大qtree数をカスタマイズできます。</p> <p>qtreesPerFlexvol</p>	"" (デフォルトでは適用されませ ん)
debugTraceFlags	トラブルシューティング時に使用するデバッグフラグ。例： {"api" : false, "method" : true} 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、は使用しない `debugTraceFlags` 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 `nfs`、`smb` または `null` です。 `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
nfsMountOptions	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	製品説明	デフォルト
qtreesPerFlexVol	FlexVolあたりの最大 qtree 数。有効な範囲は [50、300] です。	"200"
smbShare	次のいずれかを指定できます。Microsoft管理コンソールまたはONTAP CLIを使用して作成されたSMB共有の名前、TridentでSMB共有を作成できるようにする名前、ボリュームへの共通の共有アクセスを禁止する場合はパラメータを空白のままにします。オンプレミスのONTAPでは、このパラメータはオプションです。このパラメータはAmazon FSx for ONTAPバックエンドで必須であり、空にすることはできません。	smb-share
useREST	ONTAP REST API を使用するためのブーリアンパラメータ。useRESTに設定する `true` と、TridentはONTAP REST APIを使用してバックエンドと通信します。に設定する `false` と、TridentはONTAPI (ZAPI) 呼び出し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 `ontapi`。これは、事前に定義された役割と役割によって実現され vsadmin cluster-admin ます。Trident 24.06リリースおよびONTAP 9.151以降では、が useREST デフォルトで設定されて `true` います。 `false` ONTAPI (ZAPI) 呼び出しを使用するように変更してください。 `useREST`	true ONTAP 9.15.1以降の場合は、それ以外の場合は false。
limitVolumePoolSize	ONTAP NASエコノミーバックエンドでqtreeを使用する場合の、要求可能なFlexVolの最大サイズ。	"" (デフォルトでは適用されません)
denyNewVolumePools	を制限し `ontap-nas-economy` バックエンドがqtreeを格納するために新しいFlexVolボリュームを作成することです。新しいPVのプロビジョニングには、既存のFlexVolのみが使用されます。	

ボリュームのプロビジョニング用のバックエンド構成オプション

設定のセクションで、これらのオプションを使用してデフォルトのプロビジョニングを制御できます defaults。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	製品説明	デフォルト
spaceAllocation	qtreeに対するスペース割り当て	"正しい"
spaceReserve	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"
snapshotPolicy	使用する Snapshot ポリシー	"なし"

パラメータ	製品説明	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	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。	""
snapshotReserve	Snapshot 用にリザーブされているボリュームの割合	が「none」の場合は「0」 snapshotPolicy、それ以外の場合は「」
splitOnClone	作成時にクローンを親からスプリットします	いいえ
encryption	新しいボリュームでNetApp Volume Encryption (NVE) を有効にします。デフォルトはです。`false`このオプションを使用するには、クラスタで NVE のライセンスが設定され、有効になっている必要があります。バックエンドでNAEが有効になっている場合、TridentでプロビジョニングされたすべてのボリュームでNAEが有効になります。詳細については、を参照してください "TridentとNVEおよびNAEとの連携" 。	いいえ
tieringPolicy	「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）
snapshotDir	ディレクトリへのアクセスを管理します。 .snapshot	NFSv4の場合は「true」NFSv3の場合は「false」
exportPolicy	使用するエクスポートポリシー	デフォルト
securityStyle	新しいボリュームのセキュリティ形式。NFSのサポート `mixed` と `unix` セキュリティ形式。SMBのサポート `mixed` と `ntfs` セキュリティ形式。	NFSのデフォルトはです `unix`。SMBのデフォルトはです `ntfs`。
nameTemplate	カスタムボリューム名を作成するためのテンプレート。	""

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

ボリュームプロビジョニングの例

デフォルトが定義されている例を次に示します。

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

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

計算は次のとおりです。

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

`snapshotReserve = 50%`、`PVC 要求 = 5GiB` の場合、ボリュームの合計サイズは $5 / 0.5 = 10\text{GiB}$ であり、使用可能なサイズは 5GiB であり、これが PVC 要求で要求されたサイズです。`volume show` 次の例のような結果が表示されます。

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

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

最小限の設定例

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



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

ONTAP NASの経済性の例

```
---
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

スイッチオーバー後およびスイッチバック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、パラメータと svm`パラメータを省略します `dataLIF。例えば：

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

SMBボリュームの例

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

証明書ベースの認証の例

これは最小限のバックエンド構成の例です。clientCertificate、clientPrivateKey、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）に値が入力されbackend.json、それぞれクライアント証明書、秘密鍵、および信頼されたCA証明書のBase64でエンコードされた値が使用されます。

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXROZXJwYXB...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

次に、IPv6アドレスの使用例を示し `managementLIF` ます。

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

SMBボリュームを使用したAmazon FSx for ONTAPの例

`smbShare` SMBボリュームを使用するFSx for ONTAPでは、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: ZXROZXJwYXB...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、すべてのストレージプールに特定のデフォルトが設定されています (at none、at spaceAllocation false、at false encryptionなど) `spaceReserve。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で独自の、 `spaceAllocation` および `encryption` の値が設定され `spaceReserve`、一部のプールでデフォルト値が上書きされます。

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: mysqladb
  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の経済性の例

```
---  
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[\[仮想プールを使用するバックエンドの例\]](#)。フィールドを使用して parameters.selector、各StorageClass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。

- protection-gold`StorageClassは、バックエンドの最初と2番目の仮想プールにマッピングされます `ontap-nas-flexgroup。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、バックエンドの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。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はバックエンドの4番目の仮想プールにマッピングされます `ontap-nas。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"
```

- protection-silver-creditpoints-20k`StorageClassはバックエンドの3番目の仮想プールにマッピングされます `ontap-nas-flexgroup。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、バックエンドの3番目の仮想プールとバックエンドの2番目の仮想プール `ontap-nas-economy`にマッピングされます `ontap-nas。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。

初期設定後に更新 dataLIF

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>
```



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。

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAPでTridentを使用

"Amazon FSx for NetApp ONTAP"は、NetApp ONTAPストレージオペレーティングシステムを基盤とするファイルシステムを起動して実行できる、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クラスタまたはがインストールされた自己管理型Kubernetesクラスタ kubectl。
- 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)。
- 用に準備されたワーカーノード"NFSまたはiSCSI"。



EKS AMIタイプに応じて、Amazon LinuxおよびUbuntu (AMIS) で必要なノードの準備手順に従って ["Amazon Machine Images の略"](#)ください。

考慮事項

- SMBボリューム：

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



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

認証

Tridentには2つの認証モードがあります。

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



Tridentは、SVMユーザ、または別の名前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AMIsはTrident 24.10でテストされています。

アーミ	NAS	NASエコノミー	SAN	SANエコノミー
AL2023_x86_64_標準	はい	はい	はい	はい
AL2_x86_64	はい	はい	はい**	はい**
ボトルロケット_x86_64	はい*	はい	該当なし	該当なし

AL2023_ARM_64_標準	はい	はい	はい	はい
AL2_ARM_64	はい	はい	はい**	はい**
ボトルロケットアーム64	はい*	はい	該当なし	該当なし

- *マウントオプションでは「nolock」を使用する必要があります。

- **ノードを再起動せずにPVを削除できません



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、動作が確認されているAMISのガイドとして機能します。

テスト実施項目：

- EKS version: 1.30
- インストール方法：HelmとAWSアドオンとして
- 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）

詳細情報

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

IAMロールとAWS Secretを作成する

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



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

AWS 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"\n    --secret-string\n    "{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}\"\n
```

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 AmazonFSxNCSIReaderPolicy --policy\n    -document file://policy.json\n    --description "This policy grants access to Trident CSI to FSxN and\n    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用の**IAM**ロールを作成する

ポリシーを作成したら、Tridentが実行されるサービスアカウントに割り当てるロールを作成するときに使用します。

AWS CLI

```
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"\n
--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ています。

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

- OIDC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

次の例では、EKSでサービスアカウント用のIAMロールを作成します。

```
eksctl create iamserviceaccount --name trident-controller --namespace  
trident \  
--cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole  
--role-only \  
--attach-policy-arn <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

1. Tridentインストーラパッケージのダウンロード

Tridentインストーラパッケージには、Tridentオペレータの導入とTridentのインストールに必要なすべてのものが含まれています。GitHubのAssetsセクションから最新バージョンのTridentインストーラをダウンロード

ードして展開します。

```
wget
https://github.com/NetApp/trident/releases/download/v25.02.0/trident-
installer-25.02.0.tar.gz
tar -xf trident-installer-25.02.0.tar.gz
cd trident-installer
```

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

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

```
helm install trident trident-operator-100.2502.0.tgz \
--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-100.2502.0	25.02.0

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_Linux_64 ARM)
- 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

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> \
--service-account-role-arn arn:aws:iam::<account_id>:role/<role_name>
--force
```

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. [アドオンの選択]ページで、次の手順を実行します。
 - a. [AWS Marketplace EKS-addons]セクションで、* Trident by NetApp *チェックボックスを選択します。
 - b. 「* 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。
 - a. 使用する*バージョン*を選択します。
 - b. では、[Not set]*のままにします。
 - c. Add-on構成スキーマ*に従って、* Configuration Values *セクションのconfigurationValues/パラメーターを前の手順で作成したrole-arnに設定します（値は次の形式にする必要があります。

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

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

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

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

AWS CLI

1. ファイルを作成し add-on.json ます。

```
{  
  "clusterName": "<eks-cluster>",  
  "addonName": "netapp_trident-operator",  
  "addonVersion": "v25.02.1-eksbuild.1",  
  "serviceAccountRoleArn": "<role ARN>",  
  "configurationValues": {  
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",  
    "cloudProvider": "AWS"  
  }  
}
```



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
```

Trident EKSアドオンの更新

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て`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	v25.02.1-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 v25.02.1-eksbuild.1 --cluster my-cluster --force
```

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

管理コンソール

- Amazon EKSコンソールを開き <https://console.aws.amazon.com/eks/home#/clusters> ます。
- 左側のナビゲーションペインで、*[クラスタ]*を選択します。
- 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。
- [アドオン]タブを選択します。
- Trident by NetApp を選択し、Edit *を選択します。
- [Configure Trident by NetApp *]ページで、次の手順を実行します。
 - 使用する*バージョン*を選択します。
 - [Optional configuration settings]*を展開し、必要に応じて変更します。
 - 「変更を保存」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 my-cluster netapp_trident-operator
vpc-cni --addon-version v25.02.1-eksbuild.1 \
--service-account-role-arn <role-ARN> --configuration-values '{}'
--resolve-conflicts --preserve
```

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。



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。

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
```

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
```

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 b9ff-f96d916ac5e9	tbc-ontap-nas Bound	933e0071-66ce-4324-
	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-san-economy：プロビジョニングされた各PVは、Amazon FSx for NetApp ONTAPボリュームごとに設定可能なLUN数を持つLUNです。
- ontap-nas-economy：プロビジョニングされる各PVはqtreeであり、Amazon FSx for NetApp ONTAPボリュームごとにqtree数を設定できます。
- ontap-nas-flexgroup：プロビジョニングされる各PVは、完全なAmazon FSx for NetApp ONTAP FlexGroupボリュームです。

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

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	製品説明	例
version		常に 1
storageDriverName	ストレージドライバの名前	ontap-nas ontap-nas-economy、、 ontap-nas-flexgroup、、 ontap-san ontap-san-economy
backendName	カスタム名またはストレージバックエンド	ドライバ名+"_" + dataLIF
managementLIF	クラスタまたはSVM管理LIFのIPアドレス完全修飾ドメイン名 (FQDN) を指定できます。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]などの角っこで定義する必要があります。aws`フィールドを指定する場合は`fsxFilesystemID`を指定する必要があります`managementLIF`。TridentはAWSからSVM情報を取得するためです。 `managementLIF`そのため、SVMの下のユーザ (vsadminなど) のクレデンシャルを指定し、そのユーザにロールが割り当てられる必要があります`vsadmin`ます。	"10.0.0.1 ", "[2001 : 1234 : abcd : fe]"

パラメータ	製品説明	例
dataLIF	<p>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。</p>	
autoExportPolicy	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。	false
autoExportCIDRs	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"]"
labels	ボリュームに適用する任意の JSON 形式のラベルのセット	""
clientCertificate	クライアント証明書の Base64 エンコード値。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	製品説明	例
trustedCACertificate	信頼された CA 証明書の Base64 エンコード値。オプション。証明書ベースの認証に使用されます。	""
username	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	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。	
svm	使用する Storage Virtual Machine	SVM管理LIFが指定されている場合に生成されます。
storagePrefix	SVM で新しいボリュームをプロビジョニングする際に使用するプレフィックスを指定します。作成後に変更することはできません。このパラメータを更新するには、新しいバックエンドを作成する必要があります。	trident
limitAggregateUsage	* 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	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”
debugTraceFlags	トラブルシューティング時に使用するデバッグフラグ。例： {"api" : false, "method" : true} トラブルシューティングを行って詳細なログダンプが必要な場合を除き、は使用しない`debugTraceFlags`でください。	null

パラメータ	製品説明	例
nfsMountOptions	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 `nfs`、`smb` または `null` です。* SMBボリュームの場合は `smb`。* `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ます。	false

パラメータ	製品説明	例
aws	<p>AWS FSx for ONTAPの構成ファイルでは次のように指定できます。 - : AWS FSxファイルシステムのIDを指定します。</p> <p>fsxFilesystemID- apiRegion : AWS APIリージョン名。 -</p> <p>apikey : AWS APIキー。 -</p> <p>secretKey : AWSシークレットキー。</p>	<pre>""</pre> <pre>""</pre> <pre>""</pre>
credentials	<p>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"。</p>	

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

設定のセクションで、これらのオプションを使用してデフォルトのプロビジョニングを制御できます defaults。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	製品説明	デフォルト
spaceAllocation	space-allocation for LUN のコマンドを指定します	true
spaceReserve	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
snapshotPolicy	使用する Snapshot ポリシー	none
qosPolicy	作成したボリュームに割り当てる QoS ポリシーグループ。ストレージプールまたはバックエンドごとに、QOSPolicyまたはadaptiveQosPolicyのいずれかを選択します。TridentでQoSポリシーグループを使用するには、ONTAP 9.8以降が必要です。共有されていないQoSポリシーグループを使用し、ポリシーグループが各コンステイ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。	""

パラメータ	製品説明	デフォルト
adaptiveQosPolicy	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。	""
snapshotReserve	Snapshot 「0」 用にリザーブされているボリュームの割合	がの <code>none</code> 、場合 <code>snapshotPolicy else</code> 、 <code>""</code>
splitOnClone	作成時にクローンを親からスプリットします	<code>false</code>
encryption	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。 <code>false</code> 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" 。	<code>false</code>
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。	""
tieringPolicy	使用する階層化ポリシー	<code>none</code>
unixPermissions	新しいボリュームのモード。* SMB ボリュームは空にしておきます。*	""
securityStyle	新しいボリュームのセキュリティ形式。NFSのサポート <code>'mixed'</code> と <code>'unix'</code> セキュリティ形式。SMBのサポート <code>'mixed'</code> と <code>'ntfs'</code> セキュリティ形式。	NFSのデフォルトはです <code>unix</code> 。SMBのデフォルトはです <code>ntfs</code> 。

SMBボリュームをプロビジョニングする準備をします

ドライバを使用してSMBボリュームをプロビジョニングできます `ontap-nas`。完了する前に、次の手順を実行して[ONTAP SANとNASドライバの統合](#)ください。

開始する前に

ドライバを使用してSMBボリュームをプロビジョニングする`'ontap-nas'`には、次の準備が必要です。

- Linuxコントローラノードと少なくとも1つのWindowsワーカーノードでWindows Server 2019を実行して

いるKubernetesクラスタ。Tridentでは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみがサポートされます。

- Active Directoryクレデンシャルを含む少なくとも1つのTridentシークレット。シークレットを生成するには `smbccreds` :

```
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。を設定するには `csi-proxy`、Windowsで実行されているKubernetesノードについて、またはを["GitHub: Windows向けCSIプロキシ"](#)参照してください["GitHub: CSIプロキシ"](#)。

手順

- SMB共有を作成SMB管理共有は、共有フォルダスナップインを使用するか、ONTAP CLIを使用して作成できます["Microsoft管理コンソール"](#)。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

- 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。

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

- 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]
```

- 共有が作成されたことを確認します。

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



詳細については、を参照して["SMB共有を作成する"](#)ください。

- 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。FSx for ONTAPのバックエンド構成オプションについては、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	製品説明	例
smbShare	次のいずれかを指定できます。Microsoft管理コンソールまたはONTAP CLIを使用して作成されたSMB共有の名前、またはTridentにSMB共有の作成を許可する名前。このパラメータは、Amazon FSx for ONTAPバックエンドに必要です。	smb-share
nasType	*に設定する必要があります smb。*nullの場合、デフォルトはになります nfs。	smb
securityStyle	新しいボリュームのセキュリティ形式。 * SMBボリュームの場合は または mixed`に設定する必要が あります `ntfs。*	ntfs`SMBボリュームの場合はま たは `mixed
unixPermissions	新しいボリュームのモード。 SMBボリュームは空にしておく必 要があります。*	""

ストレージクラスとPVCを設定する

Kubernetes StorageClassオブジェクトを設定してストレージクラスを作成し、Tridentでボリュームのプロビジョニング方法を指定します。設定したKubernetes StorageClassを使用してPVへのアクセスを要求する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

は、"Kubernetes StorageClassオブジェクト"そのクラスで使用されるプロビジョニングツールとしてTridentを識別し、ボリュームのプロビジョニング方法をTridentに指示します。例えば：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  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、を使用して `kubectl` 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

PersistentVolumeサンプルマニフェスト

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

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

PersistentVolumeClaimサンプルマニフェスト

次に、基本的なPVC設定オプションの例を示します。

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
```

NVMe / TCP対応PVC

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

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

PVおよびPVCの作成

手順

1. PVCを作成

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

2. PVCステータスを確認します。

```
kubectl get pvc
```

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

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	タイプ	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

属性	タイプ	値	提供	リクエスト	でサポートされます
backendType	文字列	ONTAPNAS、ONTAPNASエコノミー、ONTAP-NAS-flexgroup、ONTAPSAN、solidfire-san-SAN、solidfire-san-SAN、GCP-cvs、azure-NetApp-files、ONTAP-SAN-bエコノミー	プールはこのタイプのバックエンドに属しています	バックエンドが指定されています	すべてのドライバ
Snapshot	ブール値	true false	プールは、Snapshot を含むボリュームをサポートします	Snapshot が有効なボリューム	ONTAP-NAS, ONTAP-SAN, solidfire-san-, gcvs
クローン	ブール値	true false	プールはボリュームのクローンングをサポートします	クローンが有効なボリューム	ONTAP-NAS, ONTAP-SAN, solidfire-san-, gcvs
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
```



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

2. ボリュームがに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。

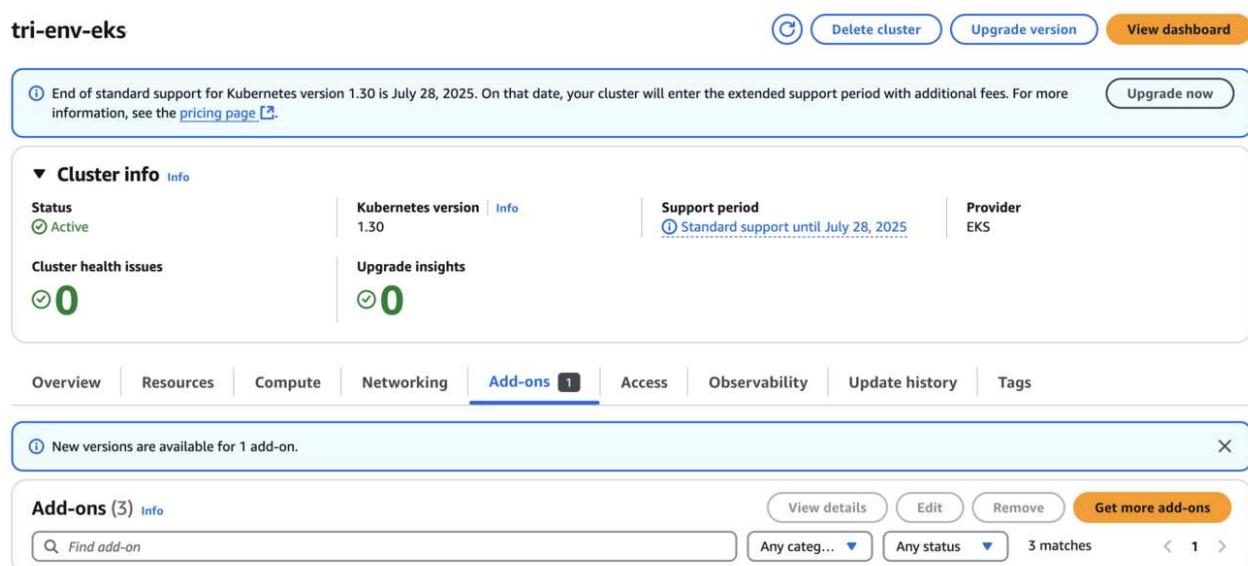
前提条件

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_Linux_64 ARM)
- 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

- 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"。
- EKS Kubernetesクラスタで、*[アドオン]*タブに移動します。



The screenshot shows the AWS EKS Cluster Management console. At the top, there is a cluster name 'tri-env-eks' and several buttons: 'Delete cluster', 'Upgrade version', and 'View dashboard'. Below this is a message box stating 'End of standard support for Kubernetes version 1.30 is July 28, 2025. On that date, your cluster will enter the extended support period with additional fees. For more information, see the [pricing page](#)' and a 'Upgrade now' button. The main area is titled 'Cluster info' with tabs for 'Info', 'Status' (Active), 'Kubernetes version' (1.30), 'Support period' (Standard support until July 28, 2025), and 'Provider' (EKS). Below this are sections for 'Cluster health issues' (0) and 'Upgrade insights' (0). At the bottom, a navigation bar includes 'Overview', 'Resources', 'Compute', 'Networking', 'Add-ons' (selected), 'Access', 'Observability', 'Update history', and 'Tags'. A notification bar at the bottom indicates 'New versions are available for 1 add-on.' The 'Add-ons' section shows 3 matches, with buttons for 'View details', 'Edit', 'Remove', and 'Get more add-ons'.

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

AWS Marketplace add-ons (1)



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

Find add-on

Filtering options

Any category ▾

NetApp, Inc. ▾

Any pricing model ▾

Clear filters

NetApp, Inc.

< 1 >



NetApp Trident

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

Standard Contract

Category
storage

Listed by
[NetApp, Inc.](#)

Supported versions
1.31, 1.30, 1.29, 1.28,
1.27, 1.26, 1.25, 1.24,
1.23

Pricing starting at
[View pricing details](#)

Cancel

Next

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。

NetApp Trident

Remove add-on

Listed by



Category

storage

Status

Ready to install



You're subscribed to this software

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

View subscription

Version

Select the version for this add-on.

v24.10.0-eksbuild.1

Select IAM role

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

Not set



► Optional configuration settings

Cancel

Previous

Next

6. ノードから継承するIAMロールオプションを選択します。

Review and add

Step 1: Select add-ons

Edit

Selected add-ons (1)

Find add-on

< 1 >

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

Step 2: Configure selected add-ons settings

Edit

Selected add-ons version (1)

< 1 >

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

EKS Pod Identity (0)

< 1 >

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

Cancel

Previous

Create

7. アドオン構成スキーマ*に従い、* Configuration Values *セクションのConfiguration Values/パラメーターを前の手順で作成したrole-arnに設定します（手順1）。値は次の形式にする必要があります。

{

```
"cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'"
```

}

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

▼ Optional configuration settings

Add-on configuration schema

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

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

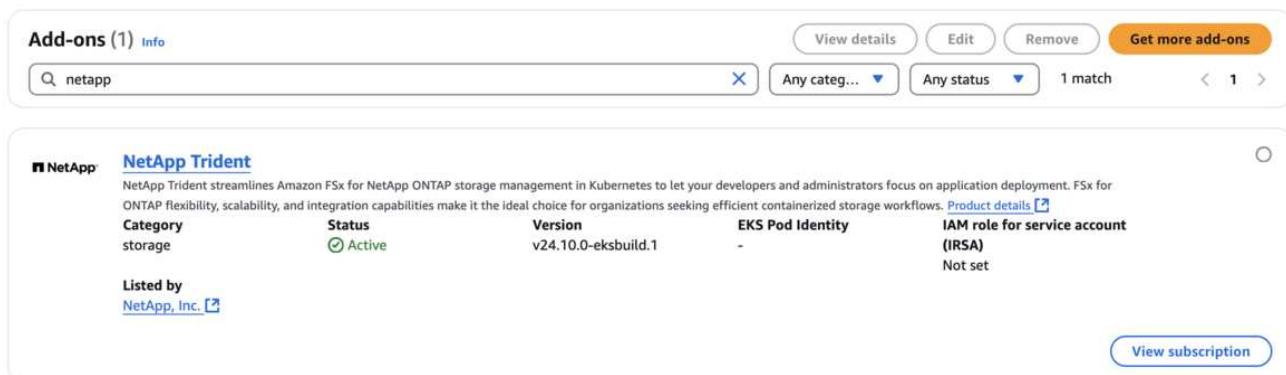
Configuration values | Info

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

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

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。



The screenshot shows the AWS Lambda console with the following details for the NetApp Trident add-on:

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

Additional details:
Listed by NetApp, Inc. [\[View\]](#)
View subscription

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。

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

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.02.1-eksbuild.1
```

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す(`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)。

オブジェクトを作成すると、`TridentBackendConfig`次の処理が実行されます。

- ・ 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。

それそれが`TridentBackendConfig`との1対1のマッピングを保持し`TridentBackend`ます。前者はバックエンドを設計および設定するためにユーザーに提供されるインターフェイスです。後者はTridentが実際のバックエンドオブジェクトを表す方法です。



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

CRの形式については、次の例を参照して`TridentBackendConfig`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

```

必要なストレージプラットフォーム/サービスの設定例については、ディレクトリにある例を参照し ["Tridentインストーラ"](#) てください。

は `spec`、バックエンド固有の設定パラメータを取得します。この例では、バックエンドでストレージドライバを使用し `ontap-san`、次の表に示す設定パラメータを使用しています。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、を参照してください["ストレージドライバのバックエンド設定情報"](#)。

この `'spec'` セクションには、CRで新たに導入されたフィールドと `'deletionPolicy'` フィールド `'TridentBackendConfig'` も含まれてい `'credentials'` ます。

- `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するクレデンシャルが含まれます。ユーザが作成した Kubernetes Secret に設定されます。クレデンシャルをプレーンテキストで渡すことはできないため、エラーになります。
- `deletionPolicy` : このフィールドは、が削除されたときの動作を定義します `TridentBackendConfig`。次の 2 つの値のいずれかを指定できます。
 - `delete` : これにより、CRと関連するバックエンドの両方が削除され `'TridentBackendConfig'` ます。これがデフォルト値です。
 - `retain` : CRが削除されても、`TridentBackendConfig` バックエンド定義は引き続き存在し、で管理できます。 `tridentctl` 削除ポリシーを設定する `'retain'` と、ユーザは以前のリリース (21.04より前のリリース) にダウングレードして、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、の作成後に更新できます `'TridentBackendConfig'`。



バックエンドの名前はを使用して設定され `'spec.backendName'` ます。指定しない場合、バックエンドの名前はオブジェクトの名前 (`metadata.name`) に設定され `'TridentBackendConfig'` ます。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 `'spec.backendName'` ます。



で作成されたバックエンドに `tridentctl` は、関連付けられたオブジェクトはありません `'TridentBackendConfig'`。このようなバックエンドをで管理するには、`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

を使用して新しいバックエンドを作成するには `kubectl`、次の手順を実行します。

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するために必要なクレデンシャルが含まれています。
2. オブジェクトを作成し `TridentBackendConfig` 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、を使用してそのステータスを確認し、追加の詳細情報を収集できます `kubectl get tbc <tbc-name> -n <trident-namespace>`。

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	ClientId	アプリケーション登録からのクライアント ID
Cloud Volumes Service for GCP	private_key_id です	秘密鍵の ID。CVS 管理者ロールを持つ GCP サービスアカウントの API キーの一部
Cloud Volumes Service for GCP	private_key を使用します	秘密鍵CVS 管理者ロールを持つ GCP サービスアカウントの API キーの一部

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
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。および ontap-san-economy の場合 `ontap-san
ONTAP	chapInitiatorSecret	CHAP イニシエータシークレット。useCHAP = true の場合は必須。および ontap-san-economy の場合 `ontap-san
ONTAP	chapTargetUsername のコマンド	ターゲットユーザ名。useCHAP = true の場合は必須。および ontap-san-economy の場合 `ontap-san
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。および ontap-san-economy の場合 `ontap-san

このステップで作成したシークレットは、次のステップで作成したオブジェクトのフィールド `TridentBackendConfig` で参照され `spec.credentials` ます。

ステップ2：CRを作成する TridentBackendConfig

これでCRを作成する準備ができ TridentBackendConfig ました。この例では、ドライバを使用するバックエンドが `ontap-san`、次のオブジェクトを使用して作成され `TridentBackendConfig` 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：CRのステータスを確認する TridentBackendConfig

CRを作成したので TridentBackendConfig、ステータスを確認できます。次の例を参照してください。

```

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

```

バックエンドが正常に作成され、CRにバインドされました `TridentBackendConfig` 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CRのuidがセットされ `TridentBackendConfig` て `configRef` ます。
- **Unbound**: を使用して表されます `""`。 `TridentBackendConfig` オブジェクトはバックエンドにバインドされていません。デフォルトでは、新しく作成されたすべての `TridentBackendConfig` CRSがこのフェーズになります。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**: CR `deletionPolicy` は `TridentBackendConfig` 削除するように設定されています。 CRが削除されると `TridentBackendConfig` CRは削除ステートに移行します。
 - 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。 TridentBackendConfig` その後、 CR は削除フェーズに入ります。 バックエンドとは `TridentBackendConfig` すべての PVC が削除された後にのみ削除されます。
- **Lost** : CR に関連付けられているバックエンドが TridentBackendConfig` 誤ってまたは故意に削除され、 `TridentBackendConfig` CR には削除されたバックエンドへの参照が残っています。

`TridentBackendConfig` CRは、値に関係なく削除できます `deletionPolicy`。

- 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、追加で処理できる処理がいくつかあります["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	Bound	ontap-san-backend	8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
	Success	ontap-san	delete

さらに、のyaml/jsonダンプを取得することもできます `TridentBackendConfig`。

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

```

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

```

backendInfo`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、コマンドを実行し `kubectl -n trident describe tbc <tbc-cr-name>` ます。



を使用して、関連付けられたオブジェクトを tridentctl` 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 `TridentBackendConfig`。とを TridentBackendConfig` 切り替える手順について説明します `tridentctl "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

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。 kubectl

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
```

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 `tridentctl get backend -n trident。このリストには、で作成
されたバックエンドも含まれ `tridentctl` 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。
 - 次のコマンドを使用して、Kubernetesから直接オブジェクトを更新できます

TridentBackendConfig。

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

- または、次のコマンドを使用して既存のCRに変更を加えることもできます
TridentBackendConfig。

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

-  • バックエンドの更新に失敗した場合、バックエンドは最後の既知の設定のまま残ります。ログを表示して原因を特定するには、またはを `kubectl describe tbc <tbc-name> -n trident` 実行し `kubectl get tbc <tbc-name> -o yaml -n trident` 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

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。tridentctl

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"、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、コマンドをもう一度実行できます create。

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。

- バックエンド名を取得します。

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

- バックエンドを削除します。

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



TridentでプロビジョニングされたボリュームとこのバックエンドからSnapshotが残っている場合、バックエンドを削除すると、そのバックエンドで新しいボリュームがプロビジョニングされなくなります。バックエンドは引き続き「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、コマンドをもう一度実行できます update。

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

これは、バックエンドオブジェクト用に出力するJSONで回答できる質問の例 `tridentctl` です。これは、インストールする必要があるユーティリティを使用し `jq` ます。

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

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

の導入により `TridentBackendConfig`、管理者はバックエンドを2つの独自の方法で管理できるようになりました。これには、次のような質問があります。

- を使用して作成したバックエンドはで管理 `TridentBackendConfig` でき `tridentctl` ますか。
- を使用して作成したバックエンドはを使用して管理 `tridentctl` でき `TridentBackendConfig` ますか。

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

このセクションでは、オブジェクトを作成してKubernetesインターフェイスから直接 `TridentBackendConfig`、作成されたバックエンドを管理するために必要な手順について説明し `tridentctl` ます。

これは、次のシナリオに該当します。

- を使用して作成された `tridentctl`、既存のバックエンドにはがありません。
`TridentBackendConfig`
- 他のオブジェクトが存在するときに、 `TridentBackendConfig`、で作成された新しいバックエンド
`tridentctl`。

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。

- を使用して作成されたバックエンドの管理に引き続き使用し `tridentctl` ます。
- を使用して作成したバックエンドを新しいオブジェクトに `TridentBackendConfig`、バインドし
`tridentctl` ます。これは、バックエンドがではなくを使用して管理されることを意味します
`kubectl tridentctl`。

を使用して既存のバックエンドを管理するには `kubectl`、既存のバックエンドにバインドするを作成する必要があります `TridentBackendConfig`。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. オブジェクトを作成し `TridentBackendConfig` 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`。`spec.backendName` 既存のバックエンドの名前に設定する必要があります。

手順 0：バックエンドを特定します

既存のバックエンドにバインドするを作成するには `TridentBackendConfig`、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": "mysqladb",
        "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：CRを作成する TridentBackendConfig

次の手順では、（この例のように）既存のCRを自動的にバインドする`ontap-nas-backend`を作成し、`TridentBackendConfig`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。

- には、同じバックエンド名が定義されています`spec.backendName`。
- 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：CRのステータスを確認する TridentBackendConfig

が作成されたら TridentBackendConfig、そのフェーズはにする必要があります Bound。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、バックエンドはオブジェクトを使用して完全に管理され `tbc-ontap-nas-backend` `TridentBackendConfig` ます。

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

`tridentctl` を使用して作成されたバックエンドの一覧表示に使用でき
`TridentBackendConfig` ます。さらに、管理者は、を削除して、がに設定されている
`retain` ことを確認する `spec.deletionPolicy` ことで、
`TridentBackendConfig` 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
`tridentctl`。

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を使用して作成されたとし `TridentBackendConfig` 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、が正常に作成され、バックエンドにバインドされていることがわかり `TridentBackendConfig` ます ([Observe the backend's UUID]) 。

手順1：Confirmがに設定されている `retain` ことを確認 `deletionPolicy`

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

```

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

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

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

```



がに設定され `retain` ていない場合は、次の手順に進まないで `deletionPolicy` ください。

手順2：CRを削除する TridentBackendConfig

最後のステップはCRを削除することです TridentBackendConfig。がに設定されている retain ことを確認したら `deletionPolicy`、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隨時、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 5225.227-19（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。