



Astra Trident を使用

Astra Trident

NetApp
April 16, 2024

目次

Astra Trident を使用	1
ワーカーノードを準備します	1
バックエンドを設定	5
kubectl を使用してバックエンドを作成します	79
kubectl を使用してバックエンド管理を実行します	86
tridentctl を使用してバックエンド管理を実行します	87
バックエンド管理オプション間を移動します	89
ストレージクラスを管理する	95
ボリューム操作を実行する	97
ネームスペース間でNFSボリュームを共有します	122
Astra Trident を監視	126

Astra Trident を使用

ワーカーノードを準備します

Kubernetes クラスタ内のすべてのワーカーノードが、ポッド用にプロビジョニングしたボリュームをマウントできる必要があります。を使用する場合 `ontap-nas`、`ontap-nas-economy` または `ontap-nas-flexgroup` バックエンドの1つを推進するのに、ワーカーノードにはNFSツールが必要です。それ以外の場合は iSCSI ツールが必要です。

最新バージョンの RedHat CoreOS には、デフォルトで NFS と iSCSI の両方がインストールされています。



NFS ツールまたは iSCSI ツールをインストールしたあとは、必ずワーカーノードをリブートしてください。リブートしないと、ボリュームをコンテナに接続できないことがあります。

ノードサービスの検出

22.07以降、Astra Tridentは、ノードでiSCSIサービスまたはNFSサービスを実行できるかどうかを自動的に検出しようとします。Astra Tridentが、検出されたサービスを特定するためのイベントをノードに対して作成次のイベントを確認するには、コマンドを使用します。

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

Tridentは、TridentノードCRの各ノードで有効になっているサービスも特定します。検出されたサービスを表示するには、次のコマンドを使用します。

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



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

NFS ボリューム

プロトコル	オペレーティングシステム	コマンド
NFS	RHEL / CentOS 7.	<code>sudo yum install -y nfs-utils</code>
NFS	Ubuntu	<code>sudo apt-get install -y nfs-common</code>



ブート時に NFS サービスが開始されていることを確認してください。

iSCSI ボリューム

iSCSI ボリュームを使用するときは、次の点に注意してください。

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

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

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

プロトコル	オペレーティングシステム	コマンド
iSCSI	RHEL/CentOS	<ol style="list-style-type: none"> 次のシステムパッケージをインストールします。 <pre>sudo yum install -y lsscsi iscsi-initiator- utils sg3_utils device- mapper-multipath</pre> iscsi-initiator-utils のバージョンが 6.2.0.874-2.el7 以降であることを確認します。 <pre>rpm -q iscsi-initiator- utils</pre> スキャンを手動に設定： <pre>sudo sed -i 's/^\(node.session.scan \).*\/\1 = manual/' /etc/iscsi/iscsid.conf</pre> マルチパスを有効化： <pre>sudo mpathconf --enable --with_multipathd y --find_multipaths n</pre> <div>  <p>確認します etc/multipath.conf が含まれます find_multipaths no の下 defaults。</p> </div> を確認します iscsid および multipathd 実行中： <pre>sudo systemctl enable --now iscsid multipathd</pre> を有効にして開始します iscsi： <pre>sudo systemctl enable --now iscsi</pre>

プロトコル	オペレーティングシステム	コマンド
iSCSI	Ubuntu	<ol style="list-style-type: none"> 次のシステムパッケージをインストールします。 <pre>sudo apt-get install -y open-iscsi lsscsi sg3- utils multipath-tools scsitools</pre> open-iscsi バージョンが 2.0.874-5ubuntu2.10 以降（bionic の場合）または 2.0.874-7.1ubuntu6.1 以降（Focal の場合）であることを確認します。 <pre>dpkg -l open-iscsi</pre> スキャンを手動に設定： <pre>sudo sed -i 's/^\(node.session.scan \).*\/\1 = manual/' /etc/iscsi/iscsid.conf</pre> マルチパスを有効化： <pre>sudo tee /etc/multipath.conf < ←'EOF' defaults { user_friendly_names yes find_multipaths no } EOF sudo systemctl enable --now multipath- tools.service sudo service multipath- tools restart</pre> <div>  <p>確認します etc/multipath.conf が含まれます find_multipaths no の下 defaults。</p> </div> を確認します open-iscsi および multipath-tools 有効になっていて実行中： <pre>sudo systemctl status multipath-tools</pre>



Ubuntu 18.04の場合は、ターゲットポートをで検出する必要があります iscsiadm 開始する前に open-iscsi iSCSIデーモンを開始します。または、を変更することもできます iscsi サービスを開始します iscsid 自動的に。

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

バックエンドを設定

バックエンドは、Astra Trident とストレージシステムの関係性を定義します。Trident がストレージシステムとの通信方法を Trident から指示し、Astra Trident がボリュームをプロビジョニングする方法も解説します。Astra Trident は、ストレージクラスが定義した要件に合わせて、バックエンドからストレージプールを自動的に提供します。お使いのストレージシステムのタイプに基づいたバックエンドの設定の詳細については、こちらを参照してください。

- ["Azure NetApp Files バックエンドを設定します"](#)
- ["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"](#)
- ["バックエンドに ONTAP または Cloud Volumes ONTAP SAN ドライバを設定します"](#)
- ["Amazon FSX for NetApp ONTAP で Astra Trident を使用"](#)

Azure NetApp Files バックエンドを設定します

Azure NetApp Files (ANF) を Astra Tridentのバックエンドとして設定できます。ANFバックエンドを使用してNASボリュームとSMBボリュームを接続できます。

- ["準備"](#)
- ["設定オプションと例"](#)

考慮事項

- Azure NetApp Files サービスでは、100GB未満のボリュームはサポートされません。100 GB のボリュームが小さい場合は、Trident が自動的に作成します。
- Astra Tridentは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみをサポート
- Astra TridentはWindows ARM アーキテクチャをサポートしていません。

Azure NetApp Files バックエンドを設定する準備をします

ANF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。

Azure NetApp Files を初めて使用する場合や新しい場所で使用する場合は、いくつかの初期設定が必要です。

- Azure NetApp Files をセットアップしてNFSボリュームを作成する方法については、[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"](#)。
- Azure NetApp Files を設定してSMBボリュームを追加するには、以下を参照してください。 ["Azure : Azure NetApp Files 用のSMBボリュームを作成します"](#)。

要件

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

- subscriptionID Azure NetApp Files を有効にしたAzureサブスクリプションから選択します。
- tenantID、 clientID`および `clientSecret から ["アプリケーション登録"](#) Azure Active Directory で、 Azure NetApp Files サービスに対する十分な権限がある。アプリケーション登録では、次のいずれかを使用します。
 - オーナーまたは寄与者のロール ["Azureで事前定義"](#)
 - A ["カスタム投稿者ロール"](#) をサブスクリプションレベルで選択します (assignableScopes)以下のアクセス許可は、Astra Tridentが必要とするものに限定されます。カスタムロールを作成したあと、["Azureポータルを使用してロールを割り当てます"](#)。

```
{
  "id": "/subscriptions/<subscription-id>/providers/Microsoft.Authorization/roleDefinitions/<role-definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [

"Microsoft.NetApp/netAppAccounts/capacityPools/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/read",
",
"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/write",
",
"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/delete",
",
```



```

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/GetMetadata/action",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTargets/read",

                                "Microsoft.Network/virtualNetworks/read",
                                "Microsoft.Network/virtualNetworks/subnets/read",

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

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

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

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

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

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

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

```

- Azureがサポートされます location を1つ以上含むデータセンターを展開します **"委任されたサブネット"**。Trident 22.01の時点では location パラメータは、バックエンド構成ファイルの最上位にある必須フィールドです。仮想プールで指定された場所の値は無視されます。

SMBボリュームに関するその他の要件

- 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クラスター。Astra Tridentは、Windowsノードで実行されているポッドにマウントされたSMBボリュームのみをサポート
- Active Directoryのクレデンシャルを含むTridentシークレットが少なくとも1つあり、ANFがActive Directoryに認証できるようになっている。シークレットを生成します `smbcreds` :

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

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

Azure NetApp Files バックエンド構成のオプションと例

ANF用のNFSとSMBのバックエンド構成オプションについて説明し、設定例を確認してください。

Astra 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、要求された場所で利用可能で、要求されたサービスレベルとサブネットに一致する容量プールにANFボリュームを作成します。



Astra Trident は、手動 QoS 容量プールをサポートしていません。

バックエンド構成オプション

ANFバックエンドには次の設定オプションがあります。

パラメータ	説明	デフォルト
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のサブスクリプション ID	
tenantID	アプリケーション登録からのテナント ID	
clientID	アプリケーション登録からのクライアント ID	
clientSecret	アプリケーション登録からのクライアントシークレット	
serviceLevel	の1つ Standard、 Premium`または `Ultra	"" (ランダム)

パラメータ	説明	デフォルト
location	新しいボリュームを作成する Azure の場所の名前	
resourceGroups	検出されたリソースをフィルタリングするためのリソースグループのリスト	"[]" (フィルタなし)
netappAccounts	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	に委任されたサブネットの名前 Microsoft.Netapp/volumes	""
networkFeatures	ボリューム用のVNet機能のセットです。の場合もあります Basic または Standard。 ネットワーク機能は一部の地域では使用できず、サブスクリプションで有効にする必要がある場合があります。を指定します networkFeatures この機能を有効にしないと、ボリュームのプロビジョニングが失敗します。	""
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、を参照してください nfsvers=4 カンマで区切って複数のマウントオプションリストを指定し、NFS v4.1 を選択します。 ストレージクラス定義で設定されたマウントオプションは、バックエンド構成で設定されたマウントオプションよりも優先されます。	"nfsvers=3 "
limitVolumeSize	要求されたボリュームサイズがこの値を超えている場合はプロビジョニングが失敗します	"" (デフォルトでは適用されません)

パラメータ	説明	デフォルト
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 オプションはです nfs、 smb または null。nullに設定すると、デフォルトでNFSボリュームが使用されます。	nfs



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」エラーが発生した場合、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。デバッグが有効になっている場合、Astra Tridentはバックエンドの作成時に検出されたAzureリソースをログに記録します。適切なロールが使用されていることを確認します。

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

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

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

ボリュームのプロビジョニング

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

パラメータ	説明	デフォルト
exportRule	<p>新しいボリュームに対するエクスポートルール</p> <p>exportRule CIDR表記のIPv4アドレスまたはIPv4サブネットの任意の組み合わせをカンマで区切って指定する必要があります。</p> <p>SMBボリュームでは無視されます。</p>	"0.0.0.0/0 "
snapshotDir	.snapshot ディレクトリの表示を制御します	いいえ
size	新しいボリュームのデフォルトサイズ	" 100G "
unixPermissions	<p>新しいボリュームのUNIX権限（8進数の4桁）。</p> <p>SMBボリュームでは無視されます。</p>	""（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）



ANFバックエンドで作成されたすべてのボリュームに対して、Astra Tridentは、ストレージプール上にあるラベルを、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、ストレージプールごとにラベルを定義し、ストレージプール内に作成されたすべてのボリュームをグループ化できます。これは、バックエンド構成で提供されるカスタマイズ可能な一連のラベルに基づいてボリュームを区別するのに便利です。

構成例

例 1：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、ANF に委譲されたネットアップアカウント、容量プール、サブネットがすべて検出され、それらのプールまたはサブネットの 1 つに新しいボリュームがランダムに配置されます。理由 `nasType` は省略されています `nfs` デフォルトが適用され、バックエンドが NFS ボリュームにプロビジョニングされます。

この構成は、ANF の利用を開始して何を試してみるときに理想的ですが、実際には、プロビジョニングするボリュームの範囲をさらに設定することを検討しています。

```
{
  "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"
}
```

例 2：容量プールフィルタを使用した特定のサービスレベル設定

このバックエンド構成では、Azure にボリュームが配置されます `eastus` の場所 `Ultra` 容量プール：Astra Trident は、ANF に委譲されたすべてのサブネットをその場所で自動的に検出し、いずれかのサブネット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"
  ],
}
```

例 3：高度な設定

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

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

例 4：仮想ストレージプールの構成

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。


```

{
  "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"
  },
  "location": "eastus",

  "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"]
    }
  ]
}

```

ストレージクラスの定義

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

を使用した定義の例 `parameter.selector` フィールド

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

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

SMBボリュームの定義例

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

例1：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"
```

例2：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}
```

例3：ボリュームごとに異なるシークレットを使用する

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



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

バックエンドを作成します

バックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

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

```
tridentctl logs
```

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

GCP バックエンドの CVS を設定します

提供されている設定例を使用して、ネットアップCloud Volumes Service (CVS) for Google Cloud Platform (GCP) をAstra Tridentインストールのバックエンドとして設定する方法を説明します。

CVS for GCPのAstra Tridentサポートについてご確認ください

Astra Tridentは、デフォルトのCVSサービスタイプがonに設定されたボリュームをサポートしています **"GCP"**。Astra Tridentは、CVSサービスタイプで許可されている最小値に関係なく、100GiB未満のCVSボリュームをサポートしていません。したがって、Tridentは要求されたボリュームが最小サイズよりも小さい場合、自動的に100GiBのボリュームを作成します。

必要なもの

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

- ネットアップ CVS 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
- を使用するGoogle Cloudサービスアカウント `netappcloudvolumes.admin` ロール
- CVS サービスアカウントの API キーファイル

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
version		常に 1
storageDriverName	ストレージドライバの名前	"GCP-cvs"

パラメータ	説明	デフォルト
backendName	カスタム名またはストレージバックエンド	ドライバ名 + "_" + API キーの一部
storageClass	ストレージのタイプから選択します hardware（パフォーマンス最適化済み）または software（CVSサービスタイプ）	
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。	
apiRegion	CVS アカウント地域。バックエンドがボリュームをプロビジョニングするリージョンです。	
apiKey	を使用したGoogle CloudサービスアカウントのAPIキー netappcloudvolumes.admin ロール。このレポートには、Google Cloud サービスアカウントの秘密鍵ファイルの JSON 形式のコンテンツが含まれています（バックエンド構成ファイルにそのままコピーされます）。	
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 サービスレベル。「Standard」、「Premium」、「Extreme」のいずれかです。	標準
network	CVSボリュームに使用するGCPネットワーク	デフォルト

パラメータ	説明	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

共有VPCネットワークを使用する場合は、両方のポートを使用します `projectNumber` および `hostProjectNumber` を指定する必要があります。その場合は、 `projectNumber` は、サービスプロジェクトです `hostProjectNumber` は、ホストプロジェクトです。

。 `apiRegion` Astra TridentがCVSボリュームを作成するGCPリージョンを表します。クロスリージョンのKubernetesクラスタを作成する場合、で作成されたCVSボリューム `apiRegion` 複数のGCP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がかかることに注意してください。

- クロスリージョンアクセスを有効にするには、のStorageClass定義を使用します `allowedTopologies` すべてのリージョンを含める必要があります。例：

```
- key: topology.kubernetes.io/region
  values:
    - us-east1
    - europe-west1
```

- `storageClass` は、必要なを選択するためのオプションのパラメータです "[CVS サービスタイプ](#)"。基本CVSサービスタイプから選択できます (`storageClass=software`) またはCVS -パフォーマンスサービスのタイプ (`storageClass=hardware`) を使用します。これは、デフォルトでTridentが使用します。必ずを指定してください `apiRegion` それぞれのCVSを提供します `storageClass` バックエンドの定義に含まれています。

Astra Trident は、 Google Cloud 上の基本 CVS サービスタイプと統合されている ベータ版の機能 で、本番環境のワークロード向けではありません。Trident は、 CVS パフォーマンスサービスタイプでは完全にサポートされている ** で、デフォルトで使用されます。

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。

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、各ボリュームのデフォルトのプロビジョニング方法を制御できます。以下の設定例を参照してください。

パラメータ	説明	デフォルト
exportRule	新しいボリュームのエクスポートルール	"0.0.0.0/0 "


```
jY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qP8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qP8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qP8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qP8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qP8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRA  
GzllZE4jK3bl/qP8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4j  
K3bl/qP8B4Kws8zX5ojY9m\nnXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE  
KEY-----\n",  
    "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"  
}
```



```

sAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSa
PIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZN
chRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz1z
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHi
sIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOgu
SaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyA
ZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz
1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrth
HisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbO
guSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKe
yAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRA
Gz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrthHisIsAbOguSaPIKeyAZNchRAGz1zZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
    "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 4：仮想ストレージプールの構成

この例は、仮想ストレージプールとともに設定されたバックエンド定義ファイルを示しています
StorageClasses それはそれらを再度参照する。

以下に示すバックエンド定義ファイルの例では、すべてのストレージプールに対して特定のデフォルトが設定されています。これにより、が設定されます `snapshotReserve 5%` およびである `exportRule` を `0.0.0.0/0` に設定します。仮想ストレージプールは、で定義されます `storage` セクション。この例では、個々のストレージプールが独自に設定されています `'serviceLevel'` をクリックすると、一部のプールでデフォルト値が上書きされます。

```
{
  "version": 1,
  "storageDriverName": "gcp-cvs",
  "projectNumber": "012345678901",
  "apiRegion": "us-west2",
  "apiKey": {
    "type": "service_account",
    "project_id": "my-gcp-project",
    "private_key_id": "1234567890123456789012345678901234567890",
    "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisI
sAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSa
PIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZN
chRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzll
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHi
sIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOgu
SaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyA
ZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz
llZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtr
HisIsAbOguSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbO
guSaPIKeyAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKe
yAZNchRAGzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRA
GzllZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
    "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"
    }
]
}

```

次の StorageClass 定義は、上記のストレージプールを参照してください。を使用します
`parameters.selector` フィールドでは、ボリュームのホストに使用される仮想プールをストレージクラスごとに指定できます。ボリュームには、選択したプールで定義された要素があります。

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

```

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

```

```

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

```

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

ネットアップが提供する Trident インストールで Element バックエンドを作成して使用方法をご確認ください。

必要なもの

- 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"](#)。

知っておくべきこと

。solidfire-san ストレージドライバは、ボリュームモード（fileとblock）の両方をサポートしています。をクリックします Filesystem volumeMode、Astra Tridentがボリュームを作成し、ファイルシステムを作成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
solidfire-san	iSCSI	ブロック	RWO 、 ROX 、 RWX	ファイルシステムがありません。raw ブロックデバイスです。
solidfire-san	iSCSI	ブロック	RWO 、 ROX 、 RWX	ファイルシステムがありません。raw ブロックデバイスです。
solidfire-san	iSCSI	ファイルシステム	RWO 、 ROX	xfs、 ext3、 ext4
solidfire-san	iSCSI	ファイルシステム	RWO 、 ROX	xfs、 ext3、 ext4



Astra Trident は強化された CSI プロビジョニング担当者として機能する場合、CHAP を使用します。CSI のデフォルトである CHAP を使用している場合は、これ以上の準備は必要ありません。を明示的に設定することを推奨します UseCHAP CSI以外のTridentでCHAPを使用するオプション。それ以外は、を参照してください ["こちらをご覧ください"](#)。



ボリュームアクセスグループは、従来の非 CSI フレームワークである Astra Trident でのみサポートされています。CSI モードで動作するように設定されている場合、Astra Trident は CHAP を使用します。

どちらでもない場合 AccessGroups または UseCHAP が設定され、次のいずれかのルールが適用されます。

- デフォルトの場合は trident アクセスグループが検出され、アクセスグループが使用されます。
- アクセスグループが検出されず、Kubernetes バージョンが 1.7 以降の場合は、CHAP が使用されます。

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
version		常に 1
storageDriverName	ストレージドライバの名前	常に「solidfire-san-」
backendName	カスタム名またはストレージバックエンド	「iSCSI_」 + ストレージ（iSCSI）IP アドレス SolidFire
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	正しいです
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



使用しないでください debugTraceFlags トラブルシューティングを実行していて、詳細なログダンプが必要な場合を除きます。



Astra Trident は、ボリュームを作成すると、ストレージプール上のすべてのラベルを、プロビジョニング時にバックアップストレージ LUN にコピーします。ストレージ管理者は、ストレージプールごとにラベルを定義し、ストレージプール内に作成されたすべてのボリュームをグループ化できます。これにより、バックエンド構成で提供されるカスタマイズ可能な一連のラベルに基づいてボリュームを簡単に区別できます。

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

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

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

例2：のバックエンドとストレージクラスの設定 solidfire-san 仮想ストレージプール用のドライバ

この例は、仮想ストレージプールで設定されたバックエンド定義ファイルと、それらを参照する StorageClasses を示しています。

以下に示すバックエンド定義ファイルの例では、すべてのストレージプールに対して特定のデフォルトが設定されています。これにより、が設定されます type シルバー。仮想ストレージプールは、で定義されます storage セクション。この例では、一部のストレージプールで独自のタイプが設定されており、一部のプールでは上記で設定したデフォルト値が上書きされます。

```

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

最初のストレージクラス (solidfire-gold-four) を選択すると、最初の仮想ストレージプールにマッピングされます。ゴールドのパフォーマンスを提供する唯一のプール volume Type QoS 金の。最後のストレージクラス (solidfire-silver) Silverパフォーマンスを提供するストレージプールをすべて特定します。Trident が、どの仮想ストレージプールを選択するかを判断し、ストレージ要件を確実に満たすようにします。

```

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

```

詳細については、こちらをご覧ください

- ["ボリュームアクセスグループ"](#)

バックエンドに **ONTAP SAN** ドライバを設定します

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

- ["準備"](#)
- ["設定と例"](#)

ユーザ権限

Tridentは、通常はを使用して、ONTAP 管理者またはSVM管理者のどちらかとして実行される必要があります。admin クラスターユーザまたは `vsadmin` SVMユーザ、または同じロールを持つ別の名前のユーザ。Amazon FSX for NetApp ONTAP 環境では、Astra Tridentは、クラスターを使用して、ONTAP 管理者またはSVM管理者のどちらかとして実行されるものと想定しています `fsxadmin` ユーザまたは `vsadmin` SVMユーザ、または同じロールを持つ別の名前のユーザ。。 `fsxadmin` このユーザは、クラスター管理者ユーザを限定的に置き換えるものです。



を使用する場合 `limitAggregateUsage` クラスター管理者権限が必要です。Amazon FSX for NetApp ONTAP をAstra Tridentとともに使用している場合は、を参照してください `limitAggregateUsage` パラメータはでは機能しません `vsadmin` および `fsxadmin` ユーザアカウント：このパラメータを指定すると設定処理は失敗します。

ONTAP 内では、Trident ドライバが使用できるより制限的な役割を作成することもできますが、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。

バックエンドに**ONTAP SAN**ドライバを設定する準備をします

ONTAP SAN ドライバを使用して ONTAP バックエンドを設定するための準備方法について説明します。ONTAP バックエンドすべてに対して、Astra Trident が SVM に少なくとも 1 つのアグリゲートを割り当てておく必要があります。

複数のドライバを実行し、1 つまたは複数のドライバを参照するストレージクラスを作成することもできます。たとえば、を設定できます `san-dev` を使用するクラス `ontap-san` ドライバおよび `A san-default` を使用するクラス `ontap-san-economy` 1つ。

すべてのKubernetesワーカーノードに適切なiSCSIツールをインストールしておく必要があります。を参照してください ["こちらをご覧ください"](#) 詳細：

認証

Astra Trident には、ONTAP バックエンドを認証する 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。など、事前定義されたセキュリティログインロールを使用することを推奨します `admin` または `vsadmin` ONTAP のバージョンとの互換性を最大限に高めるため。
- **証明書ベース** : Astra Trident は、バックエンドにインストールされた証明書を使用して ONTAP クラスター

と通信することもできます。この場合、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 を対象とした管理者またはクラスタを対象とした管理者のクレデンシャルが必要です。などの標準の事前定義されたロールを使用することを推奨します admin または vsadmin。これにより、今後のリリースの ONTAP との互換性が今後のリリースの Astra Trident で使用される機能 API が公開される可能性があります。カスタムのセキュリティログインロールは Astra Trident で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。

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

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、ユーザー名とパスワードが 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=admin"
```

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

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

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

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

4. ONTAP セキュリティログインロールでサポートされていることを確認する cert 認証方式。

```
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",
  "dataLIF": "1.2.3.8",
  "svm": "vserver_test",
  "clientCertificate": "Faaaakkkkeeee...Vaaalllluuueeeee",
  "clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
  "trustedCACertificate": "QNFinfO...SiqOyN",
  "storagePrefix": "myPrefix_"
}

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

認証方法を更新するか、クレデンシャルをローテーションして

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


```
cat cert-backend-updated.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "SanBackend",
  "managementLIF": "1.2.3.4",
  "dataLIF": "1.2.3.8",
  "svm": "vserver_test",
  "username": "vsadmin",
  "password": "secret",
  "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した場合、Astra 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ことを示しています。

igroup を指定します

Astra Trident は、igroup を使用して、プロビジョニングするボリューム（LUN）へのアクセスを制御します。管理者はバックエンドに igroup を指定する方法として、次の 2 つを選択できます。

- Astra Trident では、バックエンドごとに igroup を自動的に作成、管理できます。状況 `igroupName` はバックエンドの定義に含まれていないため、Astra Trident がという名前の igroup を作成します `trident-
<backend-UUID>` 指定します。これにより、各バックエンドに専用の igroup が割り当てられ、Kubernetes ノードの IQN の自動追加や削除が処理されます。

- また、事前に作成された igroup もバックエンドの定義で提供できます。これは、を使用して実行できます。igroupName パラメータを設定します。Astra Trident が、Kubernetes ノードの IQN を既存の igroup に追加または削除します。

を含むバックエンドの場合 igroupName 定義されている igroupName を使用して削除できます。tridentctl backend update Astra Tridentでigroupを自動処理すでにワークロードに接続されているボリュームへのアクセスが中断されることはありません。今後作成される igroup Astra Trident を使用して接続を処理します。



Astra Trident の一意のインスタンスごとに igroup を専用にすることを推奨します。これは、Kubernetes 管理者とストレージ管理者にとって有益です。CSI Trident は、クラスターノード IQN の igroup への追加と削除を自動化し、管理を大幅に簡易化します。Kubernetes 環境（および Astra Trident インストール）全体で同じ SVM を使用する場合、専用の igroup を使用することで、ある Kubernetes クラスターに対する変更が、別の Kubernetes クラスターに関連付けられた igroup に影響しないようにできます。また、Kubernetes クラスター内の各ノードに一意の IQN を設定することも重要です。前述のように、Astra Trident は IQN の追加と削除を自動的に処理します。ホスト間で IQN を再使用すると、ホスト間で誤って認識されて LUN にアクセスできないような、望ましくないシナリオが発生する可能性があります。

Astra Trident が CSI Provisioner として機能するように設定されている場合、Kubernetes ノード IQN は自動的に igroup に追加 / 削除されます。ノードがKubernetesクラスターに追加されると、trident-csi DemonSetによってポッドが展開されます (trident-csi-xxxxx) を追加し、ボリュームを接続できる新しいノードを登録します。ノード IQN もバックエンドの igroup に追加されます。ノードが遮断され、削除され、Kubernetes から削除された場合も、同様の手順で IQN の削除が処理されます。

Astra Trident が CSI Provisioner として実行されない場合は、Kubernetes クラスター内のすべてのワーカーノードからの iSCSI IQN を含むように、igroup を手動で更新する必要があります。Kubernetes クラスターに参加するノードの IQN を igroup に追加する必要があります。同様に、Kubernetes クラスターから削除されたノードの IQN を igroup から削除する必要があります。

双方向 **CHAP** を使用して接続を認証します

Astra Tridentは、に対して双方向CHAPを使用してiSCSIセッションを認証できます。ontap-san および ontap-san-economy ドライバ。これには、を有効にする必要があります。useCHAP バックエンド定義のオプション。に設定すると true、Astra Tridentは、SVMのデフォルトのイニシエータセキュリティを双方向CHAP に設定し、バックエンドファイルからのユーザ名とシークレットを設定します。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": "FaKePaSsWoRd",
  "igroupName": "trident",
  "chapInitiatorSecret": "cl9qxIm36DKyawxy",
  "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLSD6cNwxyz",
}
```



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

に加えて useCHAP=true、 chapInitiatorSecret、 chapTargetInitiatorSecret、 chapTargetUsername および chapUsername フィールドはバックエンド定義に含める必要があります。を実行すると、バックエンドが作成されたあとでシークレットを変更できます tridentctl update。

動作の仕組み

を設定します useCHAP trueに設定すると、ストレージ管理者は、ストレージバックエンドでCHAPを設定するようにAstra Trident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、Astra Tridentはデフォルトのセキュリティタイプをに設定します CHAP CHAPイニシエータとターゲットのユーザ名および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ることはありません。
- 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）。
- イニシエータへの追加の管理 igroupName バックエンドで提供されます。指定しない場合、デフォルトはです trident。

バックエンドが作成されると、対応するAstra Tridentによって作成されます tridentbackend CRDを実行し、CHAPシークレットとユーザ名をKubernetesシークレットとして保存します。このバックエンドのAstra 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、でCHAPパラメータを更新します backend.json ファイル。CHAPシ

ークレットを更新し、を使用する必要があります `tridentctl update` 変更を反映するためのコマンドです。



バックエンドのCHAPシークレットを更新する場合は、を使用する必要があります `tridentctl` バックエンドを更新します。Astra Trident では変更を取得できないため、CLI / ONTAP UI からストレージクラスタのクレデンシャルを更新しないでください。

```
cat backend-san.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "ontap_san_chap",
  "managementLIF": "192.168.0.135",
  "svm": "ontap_iscsi_svm",
  "useCHAP": true,
  "username": "vsadmin",
  "password": "FaKePaSsWoRd",
  "igroupName": "trident",
  "chapInitiatorSecret": "cl9qxUpDaTeD",
  "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLSD6cNwxyz",
}
```

```
./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+-----+-----+-----+-----+
+-----+-----+
|  NAME          | STORAGE DRIVER |          UUID          |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbe5c |
online |        7 |
+-----+-----+-----+-----+
+-----+-----+
```

既存の接続は影響を受けません。SVM の Astra Trident でクレデンシャルが更新されても、引き続きアクティブです。新しい接続では更新されたクレデンシャルが使用され、既存の接続は引き続きアクティブです。古い PVS を切断して再接続すると、更新されたクレデンシャルが使用されます。

ONTAP のSAN構成オプションと例

ONTAP SAN ドライバを作成して Astra Trident インストールで使用方法をご確認ください。このセクションでは、バックエンド構成の例と、バックエンドをストレージクラスにマッピングする方法を詳しく説明します。

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
version		常に 1
storageDriverName	ストレージドライバの名前	「ONTAP-NAS」、「ONTAP-NAS-エコノミー」、「ONTAP-NAS-flexgroup」、「ONTAP-SAN」、「ONTAP-SAN-エコノミー」
backendName	カスタム名またはストレージバックエンド	ドライバ名 + "_" + データ LIF
managementLIF	クラスタ管理 LIF または SVM 管理 LIF の IP アドレス MetroClusterのシームレスなスイッチオーバーを実現するには、SVM 管理LIFを指定する必要があります。	「10.0.0.1」、「[2001:1234:abcd::fefe]」
dataLIF	プロトコル LIF の IP アドレス。IPv6 には角かっこを使用します。設定後に更新することはできません	特に指定がないかぎり、SVM が派生します
useCHAP	CHAP を使用して ONTAP SAN ドライバ用の iSCSI を認証する [ブーリアン]	いいえ
chapInitiatorSecret	CHAP イニシエータシークレット。の場合は必須です 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。の場合は必須です useCHAP=true	「」
chapUsername	インバウンドユーザ名。の場合は必須です useCHAP=true	「」
chapTargetUsername	ターゲットユーザ名。の場合は必須です 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	「」

パラメータ	説明	デフォルト
trustedCACertificate	信頼された CA 証明書の Base64 エンコード値。任意。証明書ベースの認証に使用されます	「」
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	「」
svm	使用する Storage Virtual Machine	SVMの場合に生成されます managementLIF を指定します
igroupName	SAN ボリュームで使用する igroup の名前	"trident-<backend-UUID> "
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。設定後に更新することはできません	Trident
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）
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）
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：{"API" : false、"method" : true}	null
useREST	ONTAP REST API を使用するためのブーリアンパラメータ。* テクニカルレビュー * MetroClusterではサポートされません。	いいえ

<code>useREST</code> 考慮事項



- useREST は、テクニカルレビューとして提供されています。テスト環境では、本番環境のワークロードでは推奨されません。に設定すると true`Astra Tridentは、ONTAP REST APIを使用してバックエンドと通信します。この機能を使用するには、ONTAP 9.10 以降が必要です。また、使用するONTAP ログインロールにはへのアクセス権が必要です `ontap アプリケーション：これは事前定義されたによって満たされます vsadmin および cluster-admin ロール。
- useREST は、MetroCluster ではサポートされていません。

ONTAP クラスタと通信するには、認証パラメータを指定する必要があります。これは、セキュリティログインまたはインストールされている証明書のユーザ名 / パスワードです。



ネットアップONTAP バックエンドにAmazon FSXを使用している場合は、を指定しないでください `limitAggregateUsage` パラメータ。 `fsxadmin` および `vsadmin` Amazon FSX for NetApp ONTAP のロールには、アグリゲートの使用状況を取得し、Astra Tridentを通じて制限するために必要なアクセス権限が含まれていません。



使用しないでください `debugTraceFlags` トラブルシューティングを実行していて、詳細なログダンプが必要な場合を除きます。

をクリックします `ontap-san` ドライバのデフォルトでは、SVMのすべてのデータLIF IPが使用され、iSCSI マルチパスが使用されます。のデータLIFのIPアドレスを指定します `ontap-san` ドライバは、マルチパスを無効にして、指定されたアドレスだけを使用します。



バックエンドを作成するときは、この点に注意してください `dataLIF` および `storagePrefix` 作成後に変更することはできません。これら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。

`igroupName` ONTAP クラスタですでに作成されているigroupに設定できます。指定しない場合、Trident は `trident-<backend-UUID>` という名前の igroup を自動的に作成します。事前に定義された `igroupName` を指定する場合は、各 Kubernetes クラスタで igroup を使用することを推奨します。ただし、SVM が環境間で共有される場合です。これは、Astra Trident が IQN の追加や削除を自動的に維持するために必要です。

バックエンドは、作成後に igroup を更新することもできます。

- `igroupName` は、Astra Trident の外部の SVM で作成および管理される新しい igroup を指すように更新できます。
- `igroupName` は省略できます。この場合、Astra Trident は Trident によって `trident-<backend-UUID>` igroup が自動的に作成および管理されます。

どちらの場合も、ボリュームの添付ファイルには引き続きアクセスできます。以降のボリューム接続では、更新された igroup が使用されます。この更新によって、バックエンドにあるボリュームへのアクセスが中断されることはありません。

には完全修飾ドメイン名 (FQDN) を指定できます `managementLIF` オプション

``managementLIF`` すべてのONTAP ドライバをIPv6
アドレスに設定することもできます。Tridentをに必ずインストールしてください ``---use-ipv6`` フラグ。定義には注意が必要です ``managementLIF`` 角かっこ内のIPv6アドレス。



IPv6アドレスを使用する場合は、を確認してください `managementLIF` および `dataLIF` (バックエンド定義に含まれている場合) は、`[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`などの角括弧内に定義されます。状況 `dataLIF` が指定されていない場合、Astra TridentがSVMからIPv6 データLIFを取得します。

SANドライバでCHAPを使用できるようにするには、を設定します `useCHAP` パラメータの値 `true` バックエンドの定義に含まれています。その後、Astra Trident が、バックエンドで指定された SVM のデフォルト認証

として双方向 CHAP を設定して使用します。を参照してください ["こちらをご覧ください"](#) その仕組みについては、を参照してください。

をクリックします `ontap-san-economy` ドライバ、`limitVolumeSize` オプションを使用すると、`qtree` および LUN 用に管理するボリュームの最大サイズも制限されます。



Trident から、を使用して作成したすべてのボリュームの「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 `ontap-san` ドライバ。作成された各ボリュームについて、FlexVol の [Comments] フィールドに、配置先のストレージプールにあるすべてのラベルが入力されます。ストレージ管理者は、ストレージプールごとにラベルを定義し、ストレージプール内に作成されたすべてのボリュームをグループ化できます。これにより、バックエンド構成で提供されるカスタマイズ可能な一連のラベルに基づいてボリュームを簡単に区別できます。

ボリュームのプロビジョニング用のバックエンド構成オプション

これらのオプションを使用して、構成の特別なセクションで各ボリュームをデフォルトでプロビジョニングする方法を制御できます。例については、以下の設定例を参照してください。

パラメータ	説明	デフォルト
<code>spaceAllocation</code>	<code>space-allocation</code> for LUN のコマンドを指定します	正しいです
<code>spaceReserve</code>	スペースリザーベーションモード： 「none」（シン）または「volume」（シック）	なし
<code>snapshotPolicy</code>	使用する Snapshot ポリシー	なし
<code>qosPolicy</code>	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 <code>adaptiveQosPolicy</code> のいずれかを選択します	「」
<code>adaptiveQosPolicy</code>	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 <code>adaptiveQosPolicy</code> のいずれかを選択します	「」
<code>snapshotReserve</code>	Snapshot 「0」用にリザーブされているボリュームの割合	状況 <code>snapshotPolicy</code> は「none」、それ以外は「」です。
<code>splitOnClone</code>	作成時にクローンを親からスプリットします	いいえ
<code>splitOnClone</code>	作成時にクローンを親からスプリットします	いいえ

パラメータ	説明	デフォルト
encryption	<p>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 <code>false</code>。このオプションを使用するには、クラスターで 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。</p> <p>NAEがバックエンドで有効になっている場合は、Astra TridentでプロビジョニングされたすべてのボリュームがNAEに有効になります。</p> <p>詳細については、以下を参照してください。 "Astra TridentとNVEおよびNAEの相互運用性"。</p>	いいえ
luksEncryption	LUKS暗号化を有効にします。を参照してください "Linux Unified Key Setup (LUKS；統合キーセットアップ) を使用" 。	""
securityStyle	新しいボリュームのセキュリティ形式	「UNIX」
tieringPolicy	「none」を使用する階層化ポリシー	ONTAP 9.5 よりも前の SVM-DR 構成の「スナップショットのみ」



Trident が Astra 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。共有されない QoS ポリシーグループを使用して、各コンスチテュエントに個別にポリシーグループを適用することを推奨します。共有 QoS ポリシーグループにより、すべてのワークロードの合計スループットに対して上限が適用されます。

次に、デフォルトが定義されている例を示します。

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "trident_svm",
  "username": "admin",
  "password": "password",
  "labels": {"k8scluster": "dev2", "backend": "dev2-sanbackend"},
  "storagePrefix": "alternate-trident",
  "igroupName": "custom",
  "debugTraceFlags": {"api": false, "method": true},
  "defaults": {
    "spaceReserve": "volume",
    "qosPolicy": "standard",
    "spaceAllocation": "false",
    "snapshotPolicy": "default",
    "snapshotReserve": "10"
  }
}
```



を使用して作成したすべてのボリューム ontap-san ドライバであるAstra Tridentが、FlexVol のメタデータに対応するために、さらに10%の容量を追加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。Astra Trident が FlexVol に 10% を追加（ONTAP で利用可能なサイズとして表示）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。

を定義するバックエンドの場合 `snapshotReserve` Tridentは、次のようにボリュームサイズを計算します。

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

1.1 は、Astra Trident の 10% の追加料金で、FlexVol のメタデータに対応します。の場合 snapshotReserve = 5%、PVC要求= 5GiB、ボリュームの合計サイズは5.79GiB、使用可能なサイズは5.5GiBです。。 volume show 次の例のような結果が表示されます。

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

3 entries were displayed.

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

最小限の設定例

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



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

ontap-san 証明書ベースの認証を使用するドライバ

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

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "DefaultSANBackend",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "useCHAP": true,
  "chapInitiatorSecret": "cl9qxIm36DKyawxy",
  "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLSD6cNwxyz",
  "igroupName": "trident",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz"
}
```

ontap-san 双方向CHAPを備えたドライバ

これは、バックエンドの最小限の設定例です。この基本設定では、が作成されます ontap-san バックエンドの指定 useCHAP をに設定します true。

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "labels": {"k8scluster": "test-cluster-1", "backend": "testcluster1-
sanbackend"},
  "useCHAP": true,
  "chapInitiatorSecret": "cl9qxIm36DKyawxy",
  "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLSD6cNwxyz",
  "igroupName": "trident",
  "username": "vsadmin",
  "password": "secret"
}
```

ontap-san-economy ドライバ

```
{
  "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",
  "igroupName": "trident",
  "username": "vsadmin",
  "password": "secret"
}
```

仮想ストレージプールを使用するバックエンドの例

次のバックエンド定義ファイルの例では、などのすべてのストレージプールに対して特定のデフォルトが設定されています。spaceReserve「なし」の場合は、spaceAllocation との誤り encryption 実行されます。仮想ストレージプールは、ストレージセクションで定義します。

この例では、一部のストレージプールが独自に設定されています。spaceReserve、spaceAllocation、および encryption 値を指定すると、一部のプールでは、上記のデフォルト値が上書きされます。

```

{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "useCHAP": true,
  "chapInitiatorSecret": "cl9qxIm36DKyawxy",
  "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLSd6cNwxyz",
  "igroupName": "trident",
  "username": "vsadmin",
  "password": "secret",

  "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"
    }
}
]
}

```

のiSCSIの例を次に示します ontap-san-economy ドライバ:

```

{
  "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",
  "igroupName": "trident",
  "username": "vsadmin",
  "password": "secret",

  "defaults": {
    "spaceAllocation": "false",
    "encryption": "false"
  },
  "labels": {"store": "san_economy_store"},
  "region": "us_east_1",
  "storage": [
    {
      "labels": {"app": "oracledb", "cost": "30"},
      "zone": "us_east_1a",
      "defaults": {
        "spaceAllocation": "true",
        "encryption": "true"
      }
    },
    {
      "labels": {"app": "postgresdb", "cost": "20"},
      "zone": "us_east_1b",
      "defaults": {
        "spaceAllocation": "false",
        "encryption": "true"
      }
    }
  ]
}

```

```

    },
    {
      "labels": {"app": "mysqldb", "cost": "10"},
      "zone": "us_east_1c",
      "defaults": {
        "spaceAllocation": "true",
        "encryption": "false"
      }
    }
  ]
}

```

バックエンドを **StorageClasses** にマッピングします

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

- 最初のストレージクラス (protection-gold) を指定すると、内の1番目と2番目の仮想ストレージプールにマッピングされます `ontap-nas-flexgroup` 内の最初の仮想ストレージプール `ontap-san` バックエンド：ゴールドレベルの保護を提供している唯一のプールです。
- 2つ目のStorageClass (protection-not-gold) は、の3番目、4番目の仮想ストレージプールにマッピングされます `ontap-nas-flexgroup` のバックエンドと2番目の3番目の仮想ストレージプール `ontap-san` バックエンド：金色以外の保護レベルを提供する唯一のプールです。
- 第3のストレージクラス (app-mysqldb) をクリックすると、で4番目の仮想ストレージプールにマッピングされます `ontap-nas` のバックエンドと3つ目の仮想ストレージプール `ontap-san-economy` バックエンド：mysqldb タイプのアプリケーション用のストレージプール設定を提供しているプールは、これらだけです。
- 第4のストレージクラス (protection-silver-creditpoints-20k) は、の3番目の仮想ストレージプールにマッピングされます `ontap-nas-flexgroup` のバックエンドと2つ目の仮想ストレージプール `ontap-san` バックエンド：ゴールドレベルの保護を提供している唯一のプールは、20000 の利用可能なクレジットポイントです。
- 第5のストレージクラス (creditpoints-5k) をクリックすると、で2つ目の仮想ストレージプールにマッピングされます `ontap-nas-economy` のバックエンドと3つ目の仮想ストレージプール `ontap-san` バックエンド：5000 ポイントの利用可能な唯一のプールは以下のとおりです。

Trident が、どの仮想ストレージプールを選択するかを判断し、ストレージ要件を確実に満たすようにします。

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"

```


ONTAP NASバックエンドを設定します

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。

- "準備"
- "設定と例"



を使用する必要があります `ontap-nas` データ保護、ディザスタリカバリ、モビリティを必要とする本番ワークロード向けのドライバ。Astra Controlは、で作成されたボリュームに対して、シームレスな保護、ディザスタリカバリ、モビリティを提供します `ontap-nas` ドライバ。。 `ontap-nas-economy` ドライバは、想定されるボリューム使用量がONTAPでサポートされる量よりもはるかに高く、予想されるデータ保護、ディザスタリカバリ、モビリティ（Kubernetesクラスタ間でのボリュームの移動）の要件がないと予想される限られたユースケースでのみ使用してください。

ユーザ権限

Tridentは、通常はを使用して、ONTAP 管理者またはSVM管理者のどちらかとして実行される必要があります `admin` クラスタユーザまたは `vsadmin` SVMユーザ、または同じロールを持つ別の名前のユーザ。Amazon FSX for NetApp ONTAP 環境では、Astra Tridentは、クラスタを使用して、ONTAP 管理者またはSVM管理者のどちらかとして実行されるものと想定しています `fsxadmin` ユーザまたは `vsadmin` SVMユーザ、または同じロールを持つ別の名前のユーザ。。 `fsxadmin` このユーザは、クラスタ管理者ユーザを限定的に置き換えるものです。



を使用する場合 `limitAggregateUsage` クラスタ管理者権限が必要です。Amazon FSX for NetApp ONTAP をAstra Tridentとともに使用している場合は、を参照してください `limitAggregateUsage` パラメータはでは機能しません `vsadmin` および `fsxadmin` ユーザアカウント：このパラメータを指定すると設定処理は失敗します。

ONTAP 内では、Trident ドライバが使用できるより制限的な役割を作成することもできますが、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ドライバを使用してONTAPバックエンドを構成するための準備方法について説明します。ONTAP バックエンドすべてに対して、Astra Trident が SVM に少なくとも 1 つのアグリゲートを割り当てておく必要があります。

ONTAP バックエンドすべてに対して、Astra Trident が SVM に少なくとも 1 つのアグリゲートを割り当てておく必要があります。

複数のドライバを実行し、1つまたは複数のドライバを参照するストレージクラスを作成することもできます。たとえば、を使用するGoldクラスを設定できます `ontap-nas` ドライバとを使用するBronzeクラス `ontap-nas-economy` 1つ。

すべてのKubernetesワーカーノードに適切なNFSツールをインストールしておく必要があります。を参照してください ["こちらをご覧ください"](#) 詳細：

Astra Trident には、ONTAP バックエンドを認証する 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。など、事前定義されたセキュリティログインロールを使用することを推奨します admin または vsadmin ONTAP のバージョンとの互換性を最大限に高めるため。
- **証明書ベース** : Astra Trident は、バックエンドにインストールされた証明書を使用して ONTAP クラスタ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 を対象とした管理者またはクラスタを対象とした管理者のクレデンシャルが必要です。などの標準の事前定義されたロールを使用することを推奨します admin または vsadmin。これにより、今後のリリースの ONTAP との互換性が今後のリリースの Astra Trident で使用される機能 API が公開される可能性があります。カスタムのセキュリティログインロールは Astra Trident で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。

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

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

証明書ベースの認証を有効にします

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

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

一般的なワークフローは次の手順で構成されます。

手順

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

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

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

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

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

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

4. ONTAP セキュリティログインロールでサポートされていることを確認する cert 認証方式。

```
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。

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

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

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

7. 前の手順で得た値を使用してバックエンドを作成します。

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

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
```

NAME	STORAGE DRIVER	UUID
NasBackend	ontap-nas	98e19b74-aec7-4a3d-8dcf-128e5033b214
online	9	

認証方法を更新するか、クレデンシャルをローテーションして

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

```
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": "secret",
  "storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident

+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |          UUID          |
STATE | VOLUMES |
+-----+-----+-----+
+-----+-----+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |      9 |
+-----+-----+-----+
+-----+-----+

```



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

バックエンドを更新しても、すでに作成されているボリュームへのアクセスは中断されず、その後のボリューム接続にも影響しません。バックエンドの更新が成功した場合、Astra 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ことを示しています。

NFS エクスポートポリシーを管理します

Astra 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。

Astra Trident には、エクスポートポリシーを使用する際に次の 2 つのオプションがあります。

- Astra Trident は、エクスポートポリシー自体を動的に管理できます。このモードでは、許容可能な 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。Astra Trident は、この範囲に含まれるノード IP をエクスポートポリシーに自動的に追加します。または、CIDRs が指定されていない場合は、ノード上で検出されたグローバルスコープのユニキャスト IP がエクスポートポリシーに追加されます。
- ストレージ管理者は、エクスポートポリシーを作成したり、ルールを手動で追加したりできます。構成に別のエクスポートポリシー名を指定しないと、Astra Trident はデフォルトのエクスポートポリシーを使用します。

エクスポートポリシーを動的に管理

CSI Trident の 20.04 リリースでは、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、ストレージクラスタへのアクセスを、指定した範囲の IP を持つワーカーノードだけに制限することで、きめ細かな管理と自動化をサポートします。



エクスポートポリシーの動的管理は CSI Trident でのみ使用できます。ワーカーノードが NAT 処理されていないことを確認することが重要です。

例

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

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap_nas_auto_export",
  "managementLIF": "192.168.0.135",
  "svm": "svm1",
  "username": "vsadmin",
  "password": "FaKePaSsWoRd",
  "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）を含む事前に作成され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。ネットアップが推奨する、Astra Trident 専用のベストプラクティスを常に守ってください。

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

- `autoExportPolicy` がに設定されます `true`。これは、Astra Tridentがのエクスポートポリシーを作成することを示します `svm1 SVM`で、を使用してルール追加と削除を処理します `autoExportCIDRs` アドレスブロック。たとえば、UUID `403b5326-842-40db-96d0-d83fb3f4daec`のバックエンドです `autoExportPolicy` をに設定します `true` という名前のエクスポートポリシーを作成します `trident-403b5326-8482-40db-96d0-d83fb3f4daec` 指定します。
- `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", ":::0/0"]` です。定義されていない場合は、Astra Trident が、ワーカーノードで検出されたすべてのグローバルにスコープ指定されたユニキャストアドレスを追加します。

この例では、を使用しています `192.168.0.0/24` アドレススペースが指定されています。このアドレス範囲に含まれる Kubernetes ノードの IP が、Astra 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。Astra Tridentは、実行されてい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てチェックします `autoExportCIDRs`。IP をフィルタリングすると、Trident が検出したクライアント IP のエクスポートポリシールールを作成し、特定したノードごとに 1 つのルールが設定されます。

更新できます `autoExportPolicy` および `autoExportCIDRs` バックエンドを作成したあとのバックエンドの場合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。無効にすることもできます `autoExportPolicy` をバックエンドに追加し、手動で作成したエクスポートポリシーに戻します。これにはを設定する必要があります `exportPolicy` バックエンド構成のパラメータ。

Astra Tridentがバックエンドを作成または更新したら、を使用してバックエンドを確認できます `tridentctl` または対応する `tridentbackend` CRD :

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

Kubernetesクラスタにノードを追加してAstra Tridentコントローラに登録すると、既存のバックエンドのエクスポートポリシーが更新されます（に指定されたアドレス範囲に含まれる場合） autoExportCIDRs バックエンドの場合）をクリックします。

ノードを削除すると、Astra Trident はオンラインのすべてのバックエンドをチェックして、そのノードのアクセスルールを削除します。管理対象の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、Astra Trident は、この IP がクラスタ内の新しいノードによって再利用されないかぎり、不正なマウントを防止します。

以前のバックエンドの場合は、を使用してバックエンドを更新します tridentctl update backend では、Astra Tridentがエクスポートポリシーを自動的に管理します。これにより、バックエンドの UUID のあとにという名前の新しいエクスポートポリシーが作成され、バックエンドに存在するボリュームは、新しく作成したエクスポートポリシーを使用して、再びマウントします。



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。

ライブノードの IP アドレスが更新された場合は、ノード上の Astra Trident ポッドを再起動する必要があります。Trident が管理するバックエンドのエクスポートポリシーを更新して、この IP の変更を反映させます。

ONTAP NASの設定オプションと例

Astra Tridentのインストール環境でONTAP NASドライバを作成して使用方法について説明します。このセクションでは、バックエンド構成の例と、バックエンドをストレージクラスにマッピングする方法を詳しく説明します。

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
version		常に 1
storageDriverName	ストレージドライバの名前	「ONTAP-NAS」、「ONTAP-NAS-エコノミー」、「ONTAP-NAS-flexgroup」、「ONTAP-SAN」、「ONTAP-SAN-エコノミー」
backendName	カスタム名またはストレージバックエンド	ドライバ名 + "_" + データ LIF
managementLIF	クラスタ管理 LIF または SVM 管理 LIF の IP アドレス MetroClusterのシームレスなスイッチオーバーを実現するには、SVM 管理LIFを指定する必要があります。	「10.0.0.1」、「[2001:1234:abcd::fefe]」
dataLIF	プロトコル LIF の IP アドレス。IPv6 には角っこを使用します。設定後に更新することはできません	特に指定がないかぎり、SVM が派生します

パラメータ	説明	デフォルト
autoExportPolicy	エクスポートポリシーの自動作成と更新を有効にする [ブーリアン]	いいえ
autoExportCIDRs	KubernetesのノードIPをいつからフィルタリングするかを示すCIDRsのリスト autoExportPolicy が有効になります	[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。任意。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	
svm	使用する Storage Virtual Machine	SVMの場合に生成されます managementLIF を指定します
igroupName	SAN ボリュームで使用する igroup の名前	"trident-<backend-UUID> "
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。設定後に更新することはできません	Trident
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）
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）
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：{"API" : false、"method" : true}	null

パラメータ	説明	デフォルト
nfsMountOptions	NFS マウントオプションをカンマで区切ったリスト	「」
qtreesPerFlexvol	FlexVol あたりの最大 qtree 数。有効な範囲は [50、300] です。	「200」
useREST	ONTAP REST API を使用するためのブーリアンパラメータ。* テクニカルプレビュー * MetroClusterではサポートされません。	いいえ

<code>useREST</code> 考慮事項



- useREST は、テクニカルプレビューとして提供されています。テスト環境では、本番環境のワークロードでは推奨されません。に設定すると true`Astra Tridentは、ONTAP REST APIを使用してバックエンドと通信します。この機能を使用するには、ONTAP 9.10以降が必要です。また、使用するONTAP ログインロールにはへのアクセス権が必要です `ontap アプリケーション：これは事前定義されたによって満たされます vsadmin および cluster-admin ロール。
- useREST は、MetroCluster ではサポートされていません。

ONTAP クラスタと通信するには、認証パラメータを指定する必要があります。これは、セキュリティログインまたはインストールされている証明書のユーザ名 / パスワードです。



ネットアップONTAP バックエンドにAmazon FSXを使用している場合は、を指定しないでください limitAggregateUsage パラメータ。 fsxadmin および vsadmin Amazon FSX for NetApp ONTAP のロールには、アグリゲートの使用状況を取得し、Astra Tridentを通じて制限するために必要なアクセス権限が含まれていません。



使用しないでください debugTraceFlags トラブルシューティングを実行していて、詳細なログダンプが必要な場合を除きます。



バックエンドを作成するときは、を忘れないでください dataLIF および storagePrefix 作成後に変更することはできません。これら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。

には完全修飾ドメイン名 (FQDN) を指定できます managementLIF オプションにFQDNを指定することもできます dataLIF オプション。その場合は、NFSマウント処理にFQDNが使用されます。こうすることで、ラウンドロビン DNS を作成して、複数のデータ LIF 間で負荷を分散することができます。

`managementLIF` すべてのONTAP ドライバをIPv6アドレスに設定することもできます。Astra Tridentは、必ずを使用してインストールしてください `--use-ipv6` フラグ。を定義する際は注意が必要です `managementLIF` 角かっこ内のIPv6アドレス。



IPv6アドレスを使用する場合は、を確認してください managementLIF および dataLIF（バックエンド定義に含まれている場合）は、[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]などの角括弧内に定義されます。状況 dataLIF が指定されていない場合、Astra TridentがSVMからIPv6データLIFを取得します。

を使用する autoExportPolicy および autoExportCIDRs CSI Tridentでは、エクスポートポリシーを自動的に管理できます。これはすべての ONTAP-NAS-* ドライバでサポートされています。

をクリックします ontap-nas-economy ドライバ、 limitVolumeSize オプションを使用すると、qtreeおよびLUN用に管理するボリュームの最大サイズも制限されます qtreesPerFlexvol オプションを使用すると、FlexVol あたりの最大qtree数をカスタマイズできます。

。 nfsMountOptions パラメータを使用すると、マウントオプションを指定できます。Kubernetes 永続ボリュームのマウントオプションは通常ストレージクラスで指定されますが、ストレージクラスでマウントオプションが指定されていない場合、Astra Trident はストレージバックエンドの構成ファイルで指定されているマウントオプションを使用します。ストレージクラスまたは構成ファイルにマウントオプションが指定されていない場合、Astra Trident は関連付けられた永続的ボリュームにマウントオプションを設定しません。



Tridentから、を使用して作成したすべてのボリュームの「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(ontap-nas および(ontap-nas-flexgroup。使用するドライバに基づいて、FlexVol にコメントが設定されます (ontap-nas) またはFlexGroup のいずれかです (ontap-nas-flexgroup))。Trident が、ストレージプール上にあ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、ストレージプールごとにラベルを定義し、ストレージプール内に作成されたすべてのボリュームをグループ化できます。これにより、バックエンド構成で提供されるカスタマイズ可能な一連のラベルに基づいてボリュームを簡単に区別できます。

ボリュームのプロビジョニング用のバックエンド構成オプション

これらのオプションを使用して、構成の特別なセクションで各ボリュームをデフォルトでプロビジョニングする方法を制御できます。例については、以下の設定例を参照してください。

パラメータ	説明	デフォルト
spaceAllocation	space-allocation for LUN のコマンドを指定します	正しいです
spaceReserve	スペースリザーベーションモード： 「none」（シン）または「volume」（シック）	なし
snapshotPolicy	使用する Snapshot ポリシー	なし
qosPolicy	作成したボリュームに割り当てる QoS ポリシーグループ。ストレージプール / バックエンドごとに QOSPolicy または adaptiveQosPolicy のいずれかを選択します	「」

パラメータ	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 / 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	状況 snapshotPolicy は「none」、それ以外は「」です。
splitOnClone	作成時にクローンを親からスプリットします	いいえ
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 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。 NAE がバックエンドで有効になっている場合は、Astra Trident でプロビジョニングされたすべてのボリュームが NAE に有効になります。 詳細については、以下を参照してください。 "Astra Trident と NVE および NAE の相互運用性" 。	いいえ
securityStyle	新しいボリュームのセキュリティ形式	「UNIX」
tieringPolicy	「none」を使用する階層化ポリシー	ONTAP 9.5 よりも前の SVM-DR 構成の「スナップショットのみ」
unixPermissions	新しいボリュームのモード	「777」
Snapshot ディレクトリ	の表示/非表示を制御します .snapshot ディレクトリ	いいえ
エクスポートポリシー	使用するエクスポートポリシー	デフォルト
securityStyle の追加	新しいボリュームのセキュリティ形式	「UNIX」



Trident が Astra 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。共有されない QoS ポリシーグループを使用して、各コンスチテュエントに個別にポリシーグループを適用することを推奨します。共有 QoS ポリシーグループにより、すべてのワークロードの合計スループットに対して上限が適用されます。

次に、デフォルトが定義されている例を示します。

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

の場合 ontap-nas および ontap-nas-flexgroups`Tridentが新たに計算を使用して、FlexVol のサイズがsnapshotReserveの割合とPVCで正しく設定されていることを確認するようになりました。ユーザが PVC を要求すると、Astra 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、ユーザが要求したボリューム全体とがであるためです `snapshotReserve` には、その割合を指定します。Trident 21.07では、ユーザが要求したものが書き込み可能なスペースであり、Astra Tridentが定義します snapshotReserve ボリューム全体に対する割合として示されます。には適用されません ontap-nas-economy。この機能の仕組みについては、次の例を参照してください。

計算は次のとおりです。

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

snapshotReserve = 50%、PVC 要求 = 5GiB の場合、ボリュームの合計サイズは $2/0.5 = 10\text{GiB}$ であり、使用可能なサイズは 5GiB であり、これが PVC 要求で要求されたサイズです。。 volume show 次の例のような結果が表示されます。

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

2 entries were displayed.

以前のインストールからの既存のバックエンドは、Astra Trident のアップグレード時に前述のようにボリュームをプロビジョニングします。アップグレード前に作成したボリュームについては、変更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、が搭載されている2GiB PVCなどで `snapshotReserve=50` 以前は、書き込み可能なスペースが1GiBのボリュームが作成されていました。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 証明書ベースの認証を使用するドライバ

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

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

ontap-nas ドライバと自動エクスポートポリシー

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 Astra Trident に指示する方法を示しています。これは、でも同様に機能します `ontap-nas-economy` および `ontap-nas-flexgroup` ドライバ。

```
{
  "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": "secret",
  "nfsMountOptions": "nfsvers=4",
}
```

ontap-nas-flexgroup ドライバ

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-flexgroup",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "labels": {"k8scluster": "test-cluster-east-1b", "backend": "test1-
ontap-cluster"},
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret",
}
```

ontap-nas **IPv6**対応ドライバ

```
{
  "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": "netapp123"
}
```

ontap-nas-economy ドライバ

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

仮想ストレージプールを使用するバックエンドの例

次のバックエンド定義ファイルの例では、などのすべてのストレージプールに対して特定のデフォルトが設定されています。spaceReserve 「なし」の場合は、spaceAllocation との誤り encryption 実行されます。仮想ストレージプールは、ストレージセクションで定義します。

この例では、一部のストレージプールが独自に設定されています。spaceReserve、spaceAllocation および encryption 値を指定すると、一部のプールでは、上記のデフォルト値が上書きされます。

ontap-nas ドライバ

```
{
  {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "admin",
    "password": "secret",
    "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":{"app":"wordpress", "cost":"50"},
        "zone":"us_east_1c",
        "defaults": {
            "spaceReserve": "none",
            "encryption": "true",
            "unixPermissions": "0775"
        }
    },
    {
        "labels":{"app":"mysqldb", "cost":"25"},
        "zone":"us_east_1d",
        "defaults": {
            "spaceReserve": "volume",
            "encryption": "false",
            "unixPermissions": "0775"
        }
    }
]
}

```

ontap-nas-flexgroup ドライバ

```

{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",

```

```

"password": "secret",

"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-economy ドライバ

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

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

- 最初のストレージクラス (protection-gold) を指定すると、内の1番目と2番目の仮想ストレージプールにマッピングされます `ontap-nas-flexgroup` 内の最初の仮想ストレージプール `ontap-san` バックエンド：ゴールドレベルの保護を提供している唯一のプールです。
- 2つ目のStorageClass (protection-not-gold) は、の3番目、4番目の仮想ストレージプールにマッピングされます `ontap-nas-flexgroup` のバックエンドと2番目の3番目の仮想ストレージプール `ontap-san` バックエンド：金色以外の保護レベルを提供する唯一のプールです。
- 第3のストレージクラス (app-mysqldb) をクリックすると、で4番目の仮想ストレージプールにマッピングされます `ontap-nas` のバックエンドと3つ目の仮想ストレージプール `ontap-san-economy` バックエンド：mysqldb タイプのアプリケーション用のストレージプール設定を提供しているプールは、これらだけです。
- 第4のストレージクラス (protection-silver-creditpoints-20k) は、の3番目の仮想ストレージプールにマッピングされます `ontap-nas-flexgroup` のバックエンドと2つ目の仮想ストレージプール `ontap-san` バックエンド：ゴールドレベルの保護を提供している唯一のプールは、20000 の利用可能なクレジットポイントです。
- 第5のストレージクラス (creditpoints-5k) をクリックすると、で2つ目の仮想ストレージプールにマッピングされます `ontap-nas-economy` のバックエンドと3つ目の仮想ストレージプール `ontap-san` バックエンド：5000 ポイントの利用可能な唯一のプールは以下のとおりです。

Trident が、どの仮想ストレージプールを選択するかを判断し、ストレージ要件を確実に満たすようにします。

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"

```

Amazon FSX for NetApp ONTAP で Astra Trident を使用

"NetApp ONTAP 対応の Amazon FSX"は、NetApp ONTAP ストレージ・オペレーティング・システムを搭載したファイル・システムの起動と実行を可能にする、フルマネージドの AWS サービスです。Amazon FSX for NetApp 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際の簡易性、即応性、セキュリティ、拡張性を活用できます。FSX は、ONTAP のファイルシステム機能と管理 API の多くをサポートしています。

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

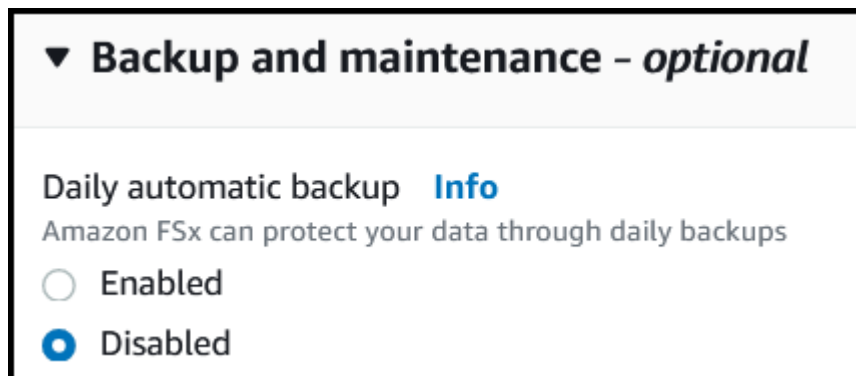
Amazon Elastic Kubernetes Service (EKS) で実行されている Astra Trident と Amazon FSX for NetApp ONTAP を使用すると、ONTAP がサポートするブロックボリュームとファイル永続ボリュームを確実にプロビジョニングできます。

ONTAP ファイルシステム用に Amazon FSX を作成します

自動バックアップが有効になっている Amazon FSX ファイルシステムで作成されたボリュームは Trident で削除できません。PVC を削除するには、PV と ONTAP ボリュームの FSX を手動で削除する必要があります。

この問題 を回避するには、次の手順

- ONTAP ファイル・システム用の FSX を作成する場合は '**Quick create**' を使用しないでください。クイック作成ワークフローでは、自動バックアップが有効になり、オプトアウトオプションはありません。
- **Standard create** を使用する場合は、自動バックアップを無効にしてください。自動バックアップを無効にすると、Trident は手動操作なしでボリュームを正常に削除できます。



Astra Trident の詳細をご確認ください

Astra Trident を初めて使用する場合は、以下のリンクを使用して確認してください。

- ["よくある質問です"](#)
- ["Astra Trident を使用するための要件"](#)
- ["Astra Trident を導入"](#)
- ["ネットアップ ONTAP 用に ONTAP、Cloud Volumes ONTAP、Amazon FSX を設定する際のベストプラクティス"](#)

ラクティス"

- "Astra Trident を統合"
- "ONTAP SAN バックエンド構成"
- "ONTAP NASバックエンド構成"

ドライバー機能の詳細をご覧ください ["こちらをご覧ください"](#)。

NetApp ONTAP 用の Amazon FSX では、を使用します **"FabricPool"** ストレージ階層を管理します。データへのアクセス頻度に基づいて階層にデータを格納することができます。

Astra Tridentは vsadmin SVMユーザまたは同じロールを持つ別の名前のユーザ。NetApp ONTAP 対応のAmazon FSXには、が搭載されています fsxadmin ONTAP を限定的に交換するユーザ admin クラスターユーザ：を使用することは推奨されません fsxadmin Tridentを使用したユーザ vsadmin SVMユーザは、より多くのAstra Trident機能にアクセスできます。

ドライバ

Astra Trident と Amazon FSX for NetApp ONTAP を統合するには、次のドライバを使用します。

- ontap-san：プロビジョニングされる各PVは、NetApp ONTAP ボリューム用に独自のAmazon FSX内にあるLUNです。
- ontap-san-economy：プロビジョニングされる各PVは、Amazon FSXあたり、NetApp ONTAP ボリューム用に構成可能なLUN数を持つLUNです。
- ontap-nas：プロビジョニングされた各PVは、NetApp ONTAP ボリュームのAmazon FSX全体です。
- ontap-nas-economy：プロビジョニングされる各PVはqtreeで、NetApp ONTAP ボリュームのAmazon FSXごとに設定可能な数のqtreeがあります。
- ontap-nas-flexgroup：プロビジョニングされた各PVは、NetApp ONTAP FlexGroup ボリュームのAmazon FSX全体です。

認証

Astra Trident には、次の 2 つの認証モードがあります。

- 証明書ベース：Astra Trident は、SVM にインストールされている証明書を使用して、FSX ファイルシステムの SVM と通信します。
- クレデンシャルベース：を使用できます fsxadmin ユーザが自身のファイルシステムまたはに割り当てられます vsadmin ユーザがSVM用に設定します。



を使用することを強く推奨します vsadmin ユーザがではなく fsxadmin バックエンドを設定します。Astra Trident は、このユーザ名とパスワードを使用して FSX ファイルシステムと通信します。

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



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

認証の詳細については、次のリンクを参照してください。

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

Amazon FSX for NetApp ONTAP を使用して、**EKS** に **Astra Trident** を導入して設定する

必要なもの

- 既存のAmazon EKSクラスタまたはを使用する自己管理型Kubernetesクラスタ `kubectl` インストール済み。
- クラスタのワーカーノードからアクセスできる、NetApp ONTAP ファイルシステムと Storage Virtual Machine (SVM) 用の既存の Amazon FSX。
- 準備されているワーカーノード ["NFS か iSCSI か"](#)。



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

Astra Trident のその他の要件については、を参照してください ["こちらをご覧ください"](#)。

手順

1. のいずれかを使用してAstra Tridentを導入 ["導入方法"](#)。
2. Trident を設定する手順は次のとおりです。
 - a. SVM の管理 LIF の DNS 名を収集します。たとえば、AWS CLIを使用してを検索します `DNSName` の下のエントリ `Endpoints` → `Management` 次のコマンドを実行した後：

```
aws fsx describe-storage-virtual-machines --region <file system region>
```

3. 認証用の証明書を作成してインストールします。を使用する場合 `ontap-san` バックエンド。を参照してください ["こちらをご覧ください"](#)。を使用する場合 `ontap-nas` バックエンド。を参照してください ["こちらをご覧ください"](#)。



ファイルシステムにアクセスできる任意の場所から SSH を使用して、ファイルシステムにログイン（証明書をインストールする場合など）できます。を使用します `fsxadmin user`、ファイルシステムの作成時に設定したパスワード、およびの管理DNS名 `aws fsx describe-file-systems`。

4. 次の例に示すように、証明書と管理 LIF の DNS 名を使用してバックエンドファイルを作成します。


```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "customBackendName",
  "managementLIF": "svm-XXXXXXXXXXXXXXXXX.fs-XXXXXXXXXXXXXXXXX.fsx.us-east-2.aws.internal",
  "svm": "svm01",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz",
}
```

バックエンドの作成については、次のリンクを参照してください。

- ["ONTAP NASドライバを使用したバックエンドの設定"](#)
- ["バックエンドに ONTAP SAN ドライバを設定します"](#)



指定しないでください dataLIF をクリックします ontap-san および ontap-san-economy Astra Tridentがマルチパスを使用できるようにするためのドライバ。



。 limitAggregateUsage パラメータはでは機能しません vsadmin および fsxadmin ユーザアカウント：このパラメータを指定すると設定処理は失敗します。

導入後、次の手順を実行してを作成します ["ストレージクラスを定義してボリュームをプロビジョニングし、ポッドでボリュームをマウント"](#)。

詳細については、こちらをご覧ください

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

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、Astra Trident とストレージシステムの関係性を定義します。Trident がストレージシステムとの通信方法を Trident から指示し、Astra Trident がボリュームをプロビジョニングする方法も解説します。Astra Trident のインストールが完了したら、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、TridentバックエンドをKubernetesインターフェイスから直接作成および管理できます。これは、を使用して実行できます kubectl または、Kubernetesディストリビューションと同等のCLIツールを使用します。

TridentBackendConfig

TridentBackendConfig (tbc、tbconfig、tbackendconfig) は、Astra Tridentをバックエンドで管理できるフロントエンドで、名前を付けたCRDです kubectl。Kubernetesやストレージ管理者は、専用のコマンドラインユーティリティを使用せずに、Kubernetes CLIを使用してバックエンドを直接作成、管理できるよ

うになりました (tridentctl)。

を作成したとき TridentBackendConfig オブジェクトの場合は次のようになります。

- バックエンドは、指定した構成に基づいて Astra Trident によって自動的に作成されます。これは、内部的にはとして表されます TridentBackend (tbe、tridentbackend) CR。
- TridentBackendConfig は一意にバインドされます TridentBackend Astra Tridentによって作成されたのです。

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



TridentBackend CRSはAstra 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場合は、を変更して更新します TridentBackendConfig オブジェクト。

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

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

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

。 spec バックエンド固有の設定パラメータを使用します。この例では、バックエンドはを使用します ontap-san storage driverおよびでは、に示す構成パラメータを使用します。使用するストレージドライバの設定オプションの一覧については、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"](#)。

。 spec セクションには、も含まれます credentials および deletionPolicy フィールドは、で新たに導入されました TridentBackendConfig CR：

- credentials：このパラメータは必須フィールドで、ストレージシステム/サービスとの認証に使用されるクレデンシャルが含まれています。ユーザが作成した Kubernetes Secret に設定されます。クレデンシャルをプレーンテキストで渡すことはできないため、エラーになります。
- deletionPolicy:このフィールドは、がどうなるかを定義します TridentBackendConfig が削除されました。次の2つの値のいずれかを指定できます。

- delete:この結果、両方が削除されます TridentBackendConfig CRとそれに関連付けられたバックエンド。これがデフォルト値です。
- retain:時 TridentBackendConfig CRが削除され、バックエンド定義は引き続き存在し、で管理できます tridentctl。削除ポリシーをに設定しています retain 以前のリリース（21.04より前）にダウングレードし、作成されたバックエンドを保持することができます。このフィールドの値は、のあとに更新できます TridentBackendConfig が作成されます。



バックエンドの名前は、を使用して設定されます spec.backendName。指定しない場合、バックエンドの名前はの名前に設定されます TridentBackendConfig オブジェクト (metadata.name)。を使用してバックエンド名を明示的に設定することを推奨します spec.backendName。



で作成されたバックエンド tridentctl が関連付けられていません TridentBackendConfig オブジェクト。このようなバックエンドの管理は、で選択できます kubectl を作成します TridentBackendConfig CR。同一の設定パラメータ（など）を指定するように注意する必要があります spec.backendName、spec.storagePrefix、spec.storageDriverName`など）。新しく作成したTridentがAstraに自動的にバインドされる `TridentBackendConfig 既存のバックエンドを使用します。

手順の概要

を使用して新しいバックエンドを作成します `kubectl`では、次の操作を実行する必要があります。

1. を作成します **"Kubernetes Secret"**。シークレットには、ストレージクラスタ / サービスと通信するために Trident から必要なクレデンシャルが含まれています。
2. を作成します TridentBackendConfig オブジェクト。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できます `kubectl get tbc <tbc-name> -n <trident-namespace>` 追加の詳細情報を収集します。

手順 1：Kubernetes Secret を作成します

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

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

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

ストレージプラットフォームシー クレットフィールドの説明	秘密	Fields概要
Azure NetApp Files の特長	ClientID	アプリケーション登録からのクラ イアント ID
Cloud Volumes Service for GCP	private_key_id です	秘密鍵の ID 。CVS 管理者ロール を持つ GCP サービスアカウントの API キーの一部
Cloud Volumes Service for GCP	private_key を使用します	秘密鍵CVS 管理者ロールを持つ GCP サービスアカウントの API キー の一部
Element （ NetApp HCI / SolidFire ）	エンドポイント	テナントのクレデンシャルを使用 する SolidFire クラスタの MVIP
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。の場合 ontap-san および ontap-san- economy
ONTAP	chapInitiatorSecret	CHAP イニシエータシークレッ ト。useCHAP = true の場合は必 須。の場合 ontap-san および ontap-san-economy
ONTAP	chapTargetUsername のコマンド	ターゲットユーザ名。useCHAP = true の場合は必須。の場合 ontap-san および ontap-san- economy
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。の場合 ontap-san お よび ontap-san-economy

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

手順2：を作成します `TridentBackendConfig` CR

これで、を作成する準備ができました `TridentBackendConfig` CR。この例では、を使用するバックエンド `ontap-san` ドライバは、を使用して作成されます `TridentBackendConfig` 以下のオブジェクト：

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

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

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

を作成しました `TridentBackendConfig` CRでは、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
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、にバインドされました `TridentBackendConfig` CR。

フェーズには次のいずれかの値を指定できます。

- `Bound`: `TridentBackendConfig` CRはバックエンドに関連付けられており、そのバックエンドにはが含まれています `configRef` をに設定します `TridentBackendConfig` CRの`uid`。
- `Unbound`:を使用して表されます `""`。。 `TridentBackendConfig` オブジェクトがバックエンドにバインドされていません。新しく作成されたすべてのファイル `TridentBackendConfig` CRSはデフォルトでこのフェーズになっています。フェーズが変更された後、再度 `Unbound` に戻すことはできません。

- **Deleting:** TridentBackendConfig CR deletionPolicy が削除対象に設定されました。をクリックします TridentBackendConfig CRが削除され、削除状態に移行します。
 - バックエンドに永続ボリューム要求 (PVC) が存在しない場合は、を削除します TridentBackendConfig その結果、Astra Tridentによってバックエンドとが削除されます TridentBackendConfig CR。
 - 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 CRはその後、削除フェーズにも入ります。バックエンドと TridentBackendConfig は、すべてのPVCが削除されたあとにのみ削除されます。
- **Lost:**に関連付けられているバックエンド TridentBackendConfig CRが誤って削除されたか、故意に削除された TridentBackendConfig CRには削除されたバックエンドへの参照があります。。 TridentBackendConfig CRは、に関係なく削除できます deletionPolicy 価値。
- **Unknown:** Astra Tridentは、に関連付けられているバックエンドの状態または存在を特定できません TridentBackendConfig CR。たとえば、APIサーバが応答していない場合や、が応答していない場合などです tridentbackends.trident.netapp.io CRDがありません。これには、ユーザの介入が必要な場合があります。

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

(オプション) 手順 4 : 詳細を確認します

バックエンドに関する詳細情報を確認するには、次のコマンドを実行します。

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

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

さらに、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 が含まれます backendName および backendUUID に応答して作成されたバックエンドの TridentBackendConfig CR。lastOperationStatus フィールドは、の最後の操作のステータスを表します TridentBackendConfig CR。ユーザーがトリガすることができます（例えば、ユーザーがで何かを変更した場合など） spec）を使用するか、Astra Tridentによってトリガーされます（Astra Tridentの再起動時など）。Success または Failed のいずれかです。phase は、間の関係のステータスを表します TridentBackendConfig CRとバックエンド。上記の例では、phase 値はバインドされています。これは、を意味します TridentBackendConfig CRはバックエンドに関連付けられています。

を実行できます `kubectl -n trident describe tbc <tbc-cr-name>` イベントログの詳細を確認するためのコマンドです。



関連付けられているが含まれているバックエンドは更新または削除できません TridentBackendConfig を使用するオブジェクト tridentctl。切り替えに関連する手順を理解する tridentctl および TridentBackendConfig、["こちらを参照してください"](#)。

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処理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 TridentBackendConfig`を使用して、Astra Tridentにバックエンドの削除と保持を指示します（ベースはです） `deletionPolicy`）。バックエンドを削除するには、を確認します deletionPolicy は削除に設定されています。のみを削除します TridentBackendConfig`を参照してください `deletionPolicy` はretainに設定されています。これにより、バックエンドがまだ存在し、を使用して管理できるようになります tridentctl。

次のコマンドを実行します。

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

Astra Tridentは、が使用していたKubernetesシークレットを削除しません 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
```

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

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場合、で使用されるKubernetes Secret TridentBackendConfig オブジェクトを更新する必要があります。Astra Trident が、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。
この場合、TridentBackendConfig オブジェクトはKubernetesを使用して直接更新できます。


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

または、既存のに変更を加えます TridentBackendConfig CRには次のコマンドを実行します。

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

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

構成ファイル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処理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](#)"を使用して、次のコマンドを実行します。

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

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

```
tridentctl logs -n trident
```

構成ファイルで問題を特定して修正したら、を実行するだけです `create` コマンドをもう一度実行します。

バックエンドを削除します

Astra Trident からバックエンドを削除するには、次の手順を実行します。

1. バックエンド名を取得します。

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

2. バックエンドを削除します。

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



Astra Trident で、まだ存在しているこのバックエンドからボリュームとスナップショットをプロビジョニングしている場合、バックエンドを削除すると、新しいボリュームをプロビジョニングできなくなります。バックエンドは「削除」状態のままになり、Trident は削除されるまでそれらのボリュームとスナップショットを管理し続けます。

既存のバックエンドを表示します

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

- 概要を取得するには、次のコマンドを実行します。

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

- すべての詳細を確認するには、次のコマンドを実行します。

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

バックエンドを更新します

新しいバックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

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

```
tridentctl logs -n trident
```

構成ファイルで問題を特定して修正したら、を実行するだけです update コマンドをもう一度実行します。

バックエンドを使用するストレージクラスを特定します

以下は、回答 でできるJSON形式の質問の例です tridentctl バックエンドオブジェクトの出力。これにはを使用します jq ユーティリティをインストールする必要があります。

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

を使用して作成されたバックエンドにも該当します TridentBackendConfig。

バックエンド管理オプション間を移動します

Astra Trident でバックエンドを管理するさまざまな方法をご確認ください。を導入しました `TridentBackendConfig` 管理者は現在、バックエンドを2つの方法で管理できるようになっています。これには、次のような質問があります。

- を使用してバックエンドを作成可能 `tridentctl` で管理できます `TridentBackendConfig`?
- を使用してバックエンドを作成可能 `TridentBackendConfig` を使用して管理します `tridentctl`?

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

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

これは、次のシナリオに該当します。

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

どちらの場合も、Trident でボリュームをスケジューリングし、処理を行っているバックエンドは引き続き存在します。管理者には次の 2 つの選択肢があります。

- の使用を続行します `tridentctl` を使用して作成されたバックエンドを管理します。
- を使用して作成したバックエンドをバインド `tridentctl` 新しい `TridentBackendConfig` オブジェクト。これにより、バックエンドはを使用して管理されます `kubectl` ではありません `tridentctl`。

を使用して、既存のバックエンドを管理します `kubectl` を作成する必要があります

`TridentBackendConfig` これは既存のバックエンドにバインドします。その仕組みの概要を以下に示します。

1. Kubernetes Secret を作成します。シークレットには、ストレージクラスタ / サービスと通信するために Trident から必要なクレデンシャルが含まれています。
2. を作成します `TridentBackendConfig` オブジェクト。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（など）を指定するように注意する必要があります `spec.backendName`、`spec.storagePrefix`、`spec.storageDriverName`（など）。 `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

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

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

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

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

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

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

```
]
}
```

手順 1 : Kubernetes Secret を作成します

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

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

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

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

手順2：を作成します TridentBackendConfig CR

次の手順では、を作成します TridentBackendConfig 既存のに自動的にバインドされるCR ontap-nas-backend（この例のように）。次の要件が満たされていることを確認します。

- 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。
- 仮想ストレージプール（存在する場合）は、元のバックエンドと同じ順序で設定する必要があります。
- 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。

この場合は、を参照してください TridentBackendConfig 次のようになります。

```

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

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

```

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

のあとに入力します TridentBackendConfig が作成されている必要があります 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 -n trident get tbc tbc-ontap-nas-backend -n trident
```

NAME	BACKEND NAME	BACKEND UUID
tbc-ontap-nas-backend	ontap-nas-backend	52f2eb10-e4c6-4160-99fc-96b3be5ab5d7

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

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

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

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

`tridentctl` を使用して、を使用して作成されたバックエンドを表示できます `TridentBackendConfig`。また、管理者は、を使用してこのようなバックエンドを完全に管理することもできます `tridentctl` 削除します `TridentBackendConfig` そして確かめなさい `spec.deletionPolicy` がに設定されます `retain`。

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ます TridentBackendConfig :

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

NAME	BACKEND NAME	BACKEND UUID
backend-tbc-ontap-san	ontap-san-backend	81abcb27-ea63-49bb-b606-0a5315ac5f82

```
tridentctl get backend ontap-san-backend -n trident
```

NAME	STORAGE DRIVER	UUID
ontap-san-backend	ontap-san	81abcb27-ea63-49bb-b606-0a5315ac5f82

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

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

では、の価値を見てみましょう deletionPolicy。これはに設定する必要があります retain。これにより、が確実に実行されます TridentBackendConfig CRが削除され、バックエンド定義は引き続き存在し、で管理できます tridentctl。

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

NAME	BACKEND NAME	BACKEND UUID
backend-tbc-ontap-san	ontap-san-backend	81abcb27-ea63-49bb-b606-0a5315ac5f82

```
# 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
backend-tbc-ontap-san	ontap-san-backend	81abcb27-ea63-49bb-b606-0a5315ac5f82



それ以外の場合は、次の手順に進まないでください `deletionPolicy` がに設定されます `retain`。

手順2：を削除します `TridentBackendConfig` CR

最後の手順は、を削除することです `TridentBackendConfig` CR。確認が完了したら `deletionPolicy` がに設定されます `retain` をクリックすると、次のように削除されます。

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

が削除されたとき `TridentBackendConfig` `Astra Trident`は、実際にバックエンド自体を削除することなく、単にオブジェクトを削除します。

ストレージクラスを管理する

ストレージクラスの作成、ストレージクラスの削除、および既存のストレージクラスの表示に関する情報を検索します。

ストレージクラスを設計する

を参照してください ["こちらをご覧ください"](#) ストレージクラスとその設定方法の詳細については、を参照してください。

ストレージクラスを作成する。

ストレージクラスファイルが作成されたら、次のコマンドを実行します。

```
kubectl create -f <storage-class-file>
```

`<storage-class-file>` は、ストレージクラスのファイル名に置き換えてください。

ストレージクラスを削除する

Kubernetes からストレージクラスを削除するには、次のコマンドを実行します。

```
kubectl delete storageclass <storage-class>
```

<storage-class> は、ストレージクラスで置き換える必要があります。

このストレージクラスで作成された永続ボリュームには変更はなく、Astra Trident によって引き続き管理されます。



Astra Tridentでは空白が強制される fsType を作成します。iSCSIバックエンドの場合は、適用することを推奨します parameters.fsType ストレージクラス。esixting StorageClassesを削除して、を使用して再作成する必要があります parameters.fsType 指定された。

既存のストレージクラスを表示します

- 既存の Kubernetes ストレージクラスを表示するには、次のコマンドを実行します。

```
kubectl get storageclass
```

- Kubernetes ストレージクラスの詳細を表示するには、次のコマンドを実行します。

```
kubectl get storageclass <storage-class> -o json
```

- Astra Trident の同期されたストレージクラスを表示するには、次のコマンドを実行します。

```
tridentctl get storageclass
```

- Astra Trident の同期されたストレージクラスの詳細を表示するには、次のコマンドを実行します。

```
tridentctl get storageclass <storage-class> -o json
```

デフォルトのストレージクラスを設定する

Kubernetes 1.6 では、デフォルトのストレージクラスを設定する機能が追加されています。永続ボリューム要求（PVC）に永続ボリュームが指定されていない場合に、永続ボリュームのプロビジョニングに使用するストレージクラスです。

- アノテーションを設定してデフォルトのストレージクラスを定義します
storageclass.kubernetes.io/is-default-class をtrueに設定してストレージクラスの定義に追加します。仕様に応じて、それ以外の値やアノテーションがない場合は false と解釈されます。

- 次のコマンドを使用して、既存のストレージクラスをデフォルトのストレージクラスとして設定できます。

```
kubectl patch storageclass <storage-class-name> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 同様に、次のコマンドを使用して、デフォルトのストレージクラスアノテーションを削除できます。

```
kubectl patch storageclass <storage-class-name> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

また、このアノテーションが含まれている Trident インストーラバンドルにも例があります。



クラスタには、常に 1 つのデフォルトストレージクラスだけを設定してください。Kubernetes では、技術的に複数のストレージを使用することはできますが、デフォルトのストレージクラスがまったくない場合と同様に動作します。

ストレージクラスのバックエンドを特定します

以下は、回答 でできる JSON 形式の質問の例です `tridentctl Astra Trident` バックエンドオブジェクトの出力これにははを使用します `jq` ユーティリティ。先にインストールする必要がある場合があります。

```
tridentctl get storageclass -o json | jq '[.items[] | {storageClass: .Config.name, backends: [.storage]|unique}]'
```

ボリューム操作を実行する

Trident がボリュームを管理するための各種機能をご紹介します。

- ["CSI トポロジを使用します"](#)
- ["スナップショットを操作します"](#)
- ["ボリュームを展開します"](#)
- ["ボリュームをインポート"](#)

CSI トポロジを使用します

Astra Trident では、を使用して、Kubernetes クラスタ内にあるノードにボリュームを選択的に作成して接続できます ["CSI トポロジ機能"](#)。CSI トポロジ機能を使用すると、領域およびアベイラビリティゾーンに基づいて、ボリュームへのアクセスをノードのサブセットに制限できます。現在、クラウドプロバイダは、Kubernetes 管理者がゾーンベースのノードを生成できるようになっています。ノードは、リージョンによって異なるアベイラビリティゾーンに配置することも、リージョンによって配置することもできます。マルチゾーンアーキテクチャでワークロード用のボリュームをプロビジョニングするために、Astra Trident は CSI トポロジを使用します。



CSI トポロジ機能の詳細については、を参照してください["こちらをご覧ください"](#)。

Kubernetes には、2 つの固有のボリュームバインドモードがあります。

- を使用 VolumeBindingMode をに設定します Immediate トポロジを認識することなくボリュームを作成できます。ボリュームバインディングと動的プロビジョニングは、PVC が作成されるときに処理されます。これがデフォルトです `VolumeBindingMode` また、トポロジの制約を適用しないクラスタにも適しています。永続ボリュームは、要求側ポッドのスケジュール要件に依存せずに作成されます。
- を使用 VolumeBindingMode をに設定します `WaitForFirstConsumer` PVCの永続的ボリュームの作成とバインディングは、PVCを使用するポッドがスケジュールされて作成されるまで遅延されます。これにより、トポロジの要件に応じたスケジュールの制約を満たすようにボリュームが作成されます。



。WaitForFirstConsumer バインディングモードでは、トポロジラベルは必要ありません。これは CSI トポロジ機能とは無関係に使用できます。

必要なもの

CSI トポロジを使用するには、次のものがが必要です。

- を実行するKubernetesクラスタ ["サポートされるKubernetesバージョン"](#)

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedaefd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedaefd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- クラスタ内のノードには、トポロジを認識するためのラベルが必要です (topology.kubernetes.io/region および topology.kubernetes.io/zone) 。このラベル * は、Astra Trident をトポロジ対応としてインストールする前に、クラスタ内のノードに存在する必要があります。

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/os":"linux","node-role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

手順 1：トポロジ対応バックエンドを作成する

Astra Trident ストレージバックエンドは、アベイラビリティゾーンに基づいてボリュームを選択的にプロビジョニングするように設計できます。各バックエンドはオプションで伝送できます `supportedTopologies` サポートする必要があるゾーンおよび領域のリストを表すブロック。ストレージクラスがそのようなバックエンドを使用する場合、ボリュームは、サポートされているリージョン/ゾーンでスケジューリングされているアプリケーションから要求された場合にのみ作成されます。

バックエンド定義の例を次に示します。

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "xxxxxxxxxxxx",
  "supportedTopologies": [
    {"topology.kubernetes.io/region": "us-east1",
     "topology.kubernetes.io/zone": "us-east1-a"},
    {"topology.kubernetes.io/region": "us-east1",
     "topology.kubernetes.io/zone": "us-east1-b"}
  ]
}
```



supportedTopologies は、バックエンドごとのリージョンとゾーンのリストを提供するために使用されます。これらのリージョンとゾーンは、StorageClass で指定できる許容値のリストを表します。バックエンドで提供されるリージョンとゾーンのサブセットを含む StorageClasses の場合、Astra Trident がバックエンドにボリュームを作成します。

を定義できます supportedTopologies ストレージプールごとに作成することもできます。次の例を参照してください。

```

{"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "nas-backend-us-central1",
"managementLIF": "172.16.238.5",
"svm": "nfs_svm",
"username": "admin",
"password": "Netapp123",
"supportedTopologies": [
  {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"},
  {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
]
"storage": [
  {
    "labels": {"workload":"production"},
    "region": "Iowa-DC",
    "zone": "Iowa-DC-A",
    "supportedTopologies": [
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"}
    ]
  },
  {
    "labels": {"workload":"dev"},
    "region": "Iowa-DC",
    "zone": "Iowa-DC-B",
    "supportedTopologies": [
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
    ]
  }
]
}

```

この例では、を使用しています region および zone ラベルはストレージプールの場所を表します。 topology.kubernetes.io/region および topology.kubernetes.io/zone ストレージプールの使用場所を指定します。

手順 2：トポロジを認識するストレージクラスを定義する

クラスタ内のノードに提供されるトポロジラベルに基づいて、トポロジ情報を含めるように StorageClasses を定義できます。これにより、作成された PVC 要求の候補となるストレージプール、および Trident によってプロビジョニングされたボリュームを使用できるノードのサブセットが決まります。

次の例を参照してください。

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"

```

上記のStorageClass定義で、volumeBindingMode がに設定されます WaitForFirstConsumer。このStorageClass で要求された PVC は、ポッドで参照されるまで処理されません。および、allowedTopologies 使用するゾーンとリージョンを提供します。。netapp-san-us-east1 StorageClassがにPVCを作成します san-backend-us-east1 上で定義したバックエンド。

ステップ 3 : PVC を作成して使用する

StorageClass を作成してバックエンドにマッピングすると、PVC を作成できるようになりました。

例を参照 spec 下記：

```

---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: netapp-san-us-east1

```

このマニフェストを使用して PVC を作成すると、次のような結果になります。


```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME      CAPACITY    ACCESS MODES    STORAGECLASS
AGE
pvc-san      Pending                                netapp-san-us-east1
2s
kubect1 describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type      Reason              Age    From
  ----      -
  Normal    WaitForFirstConsumer  6s     persistentvolume-controller
waiting
for first consumer to be created before binding

```

Trident でボリュームを作成して PVC にバインドするには、ポッド内の PVC を使用します。次の例を参照してください。

```

apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: topology.kubernetes.io/region
                operator: In
                values:
                  - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
            matchExpressions:
              - key: topology.kubernetes.io/zone
                operator: In
                values:
                  - us-east1-a
                  - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
    - name: vol1
      persistentVolumeClaim:
        claimName: pvc-san
  containers:
    - name: sec-ctx-demo
      image: busybox
      command: [ "sh", "-c", "sleep 1h" ]
      volumeMounts:
        - name: vol1
          mountPath: /data/demo
      securityContext:
        allowPrivilegeEscalation: false

```

このpodSpecにより、Kubernetesは、にあるノードにPODをスケジュールするように指示されます us-east1 リージョンを選択し、にある任意のノードから選択します us-east1-a または us-east1-b ゾーン。

次の出力を参照してください。

```
kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP              NODE
NOMINATED NODE READINESS GATES
app-pod-1     1/1     Running   0           19s   192.168.25.131  node2
<none>        <none>
kubectl get pvc -o wide
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS          AGE   VOLUMEMODE
pvc-san       Bound     pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b  300Mi
RWO           netapp-san-us-east1   48s   Filesystem
```

バックエンドを更新して追加 supportedTopologies

既存のバックエンドを更新して、のリストを追加することができます supportedTopologies を使用します tridentctl backend update。これは、すでにプロビジョニングされているボリュームには影響せず、以降の PVC にのみ使用されます。

詳細については、こちらをご覧ください

- ["コンテナのリソースを管理"](#)
- ["ノードセクタ"](#)
- ["アフィニティと非アフィニティ"](#)
- ["塗料および耐性"](#)

スナップショットを操作します

永続ボリューム（PVS）のKubernetesボリュームSnapshot（ボリュームSnapshot）を作成して、Astra Tridentボリュームのポイントインタイムコピーを保持できます。また、既存のボリュームSnapshotから、`_clone__`という名前の新しいボリュームを作成することもできます。ボリュームSnapshotは、でサポートされます ontap-nas、ontap-san、ontap-san-economy、solidfire-san、gcp-cvs`および`azure-netapp-files ドライバ。

作業を開始する前に

外部スナップショットコントローラとカスタムリソース定義（CRD）が必要です。Kubernetesオーケストレーションツール（例：Kubeadm、GKE、OpenShift）の役割を担っています。

KubernetesディストリビューションにスナップショットコントローラとCRDが含まれていない場合は、を参照してください [ボリュームSnapshotコントローラを導入する](#)。



GKE環境でオンデマンドボリュームスナップショットを作成する場合は、スナップショットコントローラを作成しないでください。GKEでは、内蔵の非表示のスナップショットコントローラを使用します。

ステップ1: VolumeSnapshotClass

次の例は、ボリュームSnapshotクラスを作成します。

```
cat snap-sc.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

。 driver Astra TridentのCSIドライバをポイントします。 deletionPolicy は、です Delete または Retain。 に設定すると Retain`を使用すると、ストレージクラスタの基盤となる物理Snapshotが、の場合でも保持されます `VolumeSnapshot オブジェクトが削除された。

詳細については、link: [./trident-reference/objects.html#Kubernetes -volumesnapshotclass-objects](#)を参照してください[VolumeSnapshotClass]。

手順 2 : 既存の PVC のスナップショットを作成します

次に、既存のPVCのスナップショットを作成する例を示します。

```
cat snap.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

この例では、という名前のPVCに対してスナップショットが作成されます pvc1 Snapshotの名前はに設定されます pvc1-snap。

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                AGE
pvc1-snap           50s
```

これでが作成されました VolumeSnapshot オブジェクト。ボリュームSnapshotはPVCに似ており、に関連付けられています VolumeSnapshotContent 実際のスナップショットを表すオブジェクト。

を識別できます VolumeSnapshotContent のオブジェクト pvc1-snap ボリュームSnapshot。ボリュームSnapshotの詳細を定義します。

```
kubectl describe volumesnapshots pvc1-snap
Name:          pvc1-snap
Namespace:     default
.
.
.
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-525400f3f660
  Source:
    API Group:
    Kind:      PersistentVolumeClaim
    Name:      pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
.
.
```

。 Snapshot Content Name このSnapshotを提供するVolumeSnapshotContentオブジェクトを特定します。。 Ready To Use パラメータは、Snapshotを使用して新しいPVCを作成できることを示します。

手順 3 : ボリューム **Snapshot** から **PVC** を作成します

次に、Snapshotを使用してPVCを作成する例を示します。

```
cat pvc-from-snap.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

dataSource に、という名前のボリュームSnapshotを使用してPVCを作成する必要があることを示します pvcl-snap データのソースとして。このコマンドを実行すると、Astra Trident が Snapshot から PVC を作成するように指示します。作成された PVC は、ポッドに接続して、他の PVC と同様に使用できます。



スナップショットが関連付けられている永続ボリュームを削除すると、対応する Trident ボリュームが「削除状態」に更新されます。Astra Trident ボリュームを削除するには、ボリュームの Snapshot を削除する必要があります。

ボリューム**Snapshot**コントローラを導入する

KubernetesディストリビューションにスナップショットコントローラとCRDが含まれていない場合は、次のように導入できます。

手順

1. ボリュームのSnapshot作成

```
cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. 目的のネームスペースにスナップショットコントローラを作成します。以下の YAML マニフェストを編集して名前空間を変更します。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

関連リンク

- ["ボリューム Snapshot"](#)
- ["ボリュームSnapshotクラス"](#)

ボリュームを展開します

Astra Trident により、Kubernetes ユーザは作成後にボリュームを拡張できます。ここでは、iSCSI ボリュームと NFS ボリュームの拡張に必要な設定について説明します。

iSCSI ボリュームを展開します

CSI プロビジョニングを使用して、iSCSI Persistent Volume（PV）を拡張できます。



iSCSIボリューム拡張は、でサポートされます ontap-san、ontap-san-economy、solidfire-san ドライバとにはKubernetes 1.16以降が必要です。

概要

iSCSI PV の拡張には、次の手順が含まれます。

- StorageClass定義を編集してを設定します allowVolumeExpansion フィールドからに移動します true。
- PVC定義を編集してを更新します spec.resources.requests.storage 新たに必要となったサイズを反映するには、元のサイズよりも大きくする必要があります。
- サイズを変更するには、PV をポッドに接続する必要があります。iSCSI PV のサイズ変更には、次の 2 つのシナリオがあります。
 - PV がポッドに接続されている場合、Astra Trident はストレージバックエンドのボリュームを拡張し、デバイスを再スキャンし、ファイルシステムのサイズを変更します。
 - 未接続の PV のサイズを変更しようとする、Astra Trident がストレージバックエンドのボリュームを拡張します。PVC がポッドにバインドされると、Trident はデバイスを再スキャンし、ファイルシステムのサイズを変更します。展開操作が正常に完了すると、Kubernetes は PVC サイズを更新します。

次の例は、iSCSI PVS の仕組みを示しています。

手順 1：ボリュームの拡張をサポートするようにストレージクラスを設定する

```
cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

既存のストレージクラスの場合は、編集してを追加します allowVolumeExpansion パラメータ

手順 2：作成した **StorageClass** を使用して **PVC** を作成します

```
cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```


Astra Trident が、永続的ボリューム（PV）を作成し、この永続的ボリューム要求（PVC）に関連付けます。

```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY
ACCESS MODES	STORAGECLASS	AGE	
san-pvc	Bound	pvc-8a814d62-bd58-4253-b0d1-82f2885db671	1Gi
RWO	ontap-san	8s	


```
kubectl get pv
```

NAME	CAPACITY	ACCESS MODES
RECLAIM POLICY	STATUS	CLAIM
	STORAGECLASS	REASON
	AGE	
pvc-8a814d62-bd58-4253-b0d1-82f2885db671	1Gi	RWO
Delete	Bound	default/san-pvc
		ontap-san
		10s

手順 3：PVC を接続するポッドを定義します

この例では、を使用するポッドが作成されます san-pvc。

```
kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
centos-pod	1/1	Running	0	65s


```
kubectl describe pvc san-pvc
```

Name: san-pvc
Namespace: default
StorageClass: ontap-san
Status: Bound
Volume: pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels: <none>
Annotations: pv.kubernetes.io/bind-completed: yes
pv.kubernetes.io/bound-by-controller: yes
volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers: [kubernetes.io/pvc-protection]
Capacity: 1Gi
Access Modes: RWO
VolumeMode: Filesystem
Mounted By: centos-pod

ステップ 4：PV を展開します

1Giから2Giに作成されたPVのサイズを変更するには、PVCの定義を編集してを更新します spec.resources.requests.storage 2Giへ。

```
kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
  ...
```

手順 5：拡張を検証する

PVC、PV、Astra Trident のボリュームのサイズを確認することで、拡張が正しく機能しているかどうかを検証できます。

```
kubectl get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound       pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO           ontap-san    11m

kubectl get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS    CLAIM          STORAGECLASS  REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi        RWO
Delete              Bound      default/san-pvc  ontap-san    12m

tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+-----+-----+
|          BACKEND UUID  | STATE | MANAGED |
+-----+-----+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san |
| block      | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true      |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

NFS ボリュームを拡張します

Astra Tridentは、でプロビジョニングしたNFS PVSのボリューム拡張をサポートしています ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、gcp-cvs`および `azure-netapp-files バックエンド

手順 1：ボリュームの拡張をサポートするようにストレージクラスを設定する

NFS PVのサイズを変更するには、管理者はまず、を設定してボリュームを拡張できるようにストレージクラスを構成する必要があります allowVolumeExpansion フィールドからに移動します true：

```
cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

このオプションを指定せずにストレージクラスを作成済みの場合は、を使用して既存のストレージクラスを編集するだけです kubectl edit storageclass ボリュームを拡張できるようにするため。

手順 2：作成した **StorageClass** を使用して **PVC** を作成します

```
cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Astra Trident が、この PVC に対して 20MiB の NFS PV を作成する必要があります。

```
kubectl get pvc
NAME                STATUS    VOLUME
CAPACITY            ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb        Bound      pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi
RWO                  ontapnas      9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                CAPACITY  ACCESS MODES
RECLAIM POLICY      STATUS    CLAIM                STORAGECLASS  REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi      RWO
Delete              Bound     default/ontapnas20mb  ontapnas
2m42s
```

ステップ3：PVを拡張する

新しく作成した20MiBのPVのサイズを1GiBに変更するには、そのPVCを編集してを設定します
spec.resources.requests.storage 1 GBに設定する場合：

```
kubectl edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  ...
```

手順4：拡張を検証する

PVC、PV、Astra Trident のボリュームのサイズを確認することで、サイズ変更が正しく機能しているかどうかを検証できます。

```
kubectl get pvc ontapnas20mb
NAME          STATUS    VOLUME
CAPACITY      ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb  Bound      pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi
RWO           ontapnas      4m44s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS    CLAIM          STORAGECLASS  REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi      RWO
Delete          Bound      default/ontapnas20mb  ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
|          NAME          | SIZE  | STORAGE CLASS |
+-----+-----+-----+-----+-----+-----+
| PROTOCOL | BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
+-----+-----+-----+-----+-----+-----+
| file      | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true     |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

ボリュームをインポート

を使用して、既存のストレージボリュームをKubernetes PVとしてインポートできます `tridentctl import`。

ボリュームインポートをサポートするドライバ

次の表は、ボリュームのインポートをサポートするドライバと、それらのアップグレードが導入されたリリースを示しています。

ドライバ	リリース。
ontap-nas	19.04
ontap-nas-flexgroup	19.04
solidfire-san	19.04
azure-netapp-files	19.04

ドライバ	リリース。
gcp-cvs	19.04
ontap-san	19.04

ボリュームをインポートする理由

Trident にボリュームをインポートするユースケースはいくつかあります。

- アプリケーションのコンテナ化と既存のデータセットの再利用
- エフェメラルアプリケーション用のデータセットのクローンを使用する
- 障害が発生した Kubernetes クラスタの再構築
- ディザスタリカバリ時にアプリケーションデータを移行する

インポートはどのように機能しますか。

Persistent Volume Claim （PVC；永続ボリューム要求）ファイルは、ボリュームインポートプロセスで PVC を作成するために使用されます。少なくとも、次の例に示すように、PVC ファイルには name、namespace、accessModes、および storageClassName フィールドが含まれている必要があります。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

。tridentctl クライアントは、既存のストレージボリュームをインポートするために使用されます。Trident は、ボリュームのメタデータを保持し、PVC と PV を作成することで、ボリュームをインポートします。

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
```

ストレージボリュームをインポートするには、ボリュームが含まれている Astra Trident バックエンドの名前と、ストレージ上のボリュームを一意に識別する名前（ONTAP FlexVol、Element Volume、CVS ボリュームパスなど）を指定します。ストレージボリュームは、読み取り / 書き込みアクセスを許可し、指定された Astra Trident バックエンドからアクセスできる必要があります。。-f string 引数は必須で、YAML または JSON PVC ファイルへのパスを指定します。

Astra Trident がインポートボリューム要求を受信すると、既存のボリュームサイズが決定され、PVC で設定されます。ストレージドライバによってボリュームがインポートされると、PV は ClaimRef を使用して PVC

に作成されます。再利用ポリシーは、最初には設定されています `retain` PVにあります。Kubernetes が PVC と PV を正常にバインドすると、再利用ポリシーがストレージクラスの再利用ポリシーに合わせて更新されます。ストレージクラスの再利用ポリシーが `delete` の場合、PVが削除されるとストレージボリュームが削除されます。

を使用してボリュームがインポートされる場合 `--no-manage` 引数として、Tridentはオブジェクトのライフサイクルに関してPVCまたはPVに対する追加の操作を実行しません。TridentはのPVイベントとPVCイベントを無視するため `--no-manage` オブジェクト。PVを削除してもストレージボリュームは削除されません。ボリュームのクローンやサイズ変更などの他の処理も無視されます。このオプションは、コンテナ化されたワークロードに Kubernetes を使用するが、Kubernetes 以外でストレージボリュームのライフサイクルを管理する場合に便利です。

PVC と PV にアノテーションが追加されます。このアノテーションは、ボリュームがインポートされたこと、および PVC と PV が管理されていることを示す二重の目的を果たします。このアノテーションは変更または削除しないでください。

Trident 19.07 以降では、PVS の添付ファイルを処理し、ボリュームをインポートの一環としてマウントします。以前のバージョンの Astra Trident を使用しているインポートの場合、データパスに処理は存在しないため、ボリュームをマウントできるかどうかボリュームインポートで検証されません。ストレージクラスが正しくない場合など、ボリュームのインポートでミスが発生した場合は、PVの再利用ポリシーを変更することでリカバリできます `retain` をクリックしてPVCとPVを削除し、`volume import` コマンドを再実行します。

ontap-nas および ontap-nas-flexgroup インポート

を使用して作成した各ボリューム `ontap-nas` driverはONTAP クラスタ上のFlexVol です。を使用してFlexVol をインポートする `ontap-nas` ドライバも同じように動作します。ONTAP クラスタにすでに存在するFlexVol は、としてインポートできます `ontap-nas` PVC。同様に、FlexGroup ボリュームはとしてインポートできます `ontap-nas-flexgroup` PVC



Trident がインポートする ONTAP のタイプは RW である必要があります。DP タイプのボリュームは SnapMirror デスティネーションボリュームです。Trident にボリュームをインポートする前に、ミラー関係を解除する必要があります。



。 `ontap-nas` ドライバで `qtree` をインポートおよび管理できない。 `ontap-nas` および `ontap-nas-flexgroup` ドライバでボリューム名の重複が許可されていません。

たとえば、という名前のボリュームをインポートします `managed_volume` という名前のバックエンドで `ontap_nas` では、次のコマンドを使用します。


```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>
```

```
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file      | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

という名前のボリュームをインポートします unmanaged_volume（上 ontap_nas backend）を使用します。Tridentは管理しません。次のコマンドを使用します。

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file>
--no-manage
```

```
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file      | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false     |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

を使用する場合 --no-manage Tridentは、ボリュームの名前を変更したり、ボリュームがマウントされたかどうかを検証したりすることはありません。ボリュームが手動でマウントされていない場合、ボリュームインポート処理は失敗します。



UnixPermissions カスタムのボリュームをインポートするという既存のバグが修正されました。PVC 定義またはバックエンド構成に unixPermissions を指定し、必要に応じて Astra Trident にボリュームをインポートするように指示できます。

ontap-san インポート

Astra Trident は、1つの LUN を含む ONTAP SAN FlexVol をインポートすることもできます。これはと同じです ontap-san ドライバ。FlexVol 内の各PVCおよびLUNにFlexVol を作成します。を使用できます tridentctl import 他の場合と同様にコマンドを実行します。

- の名前を含めます ontap-san バックエンド：

- インポートする必要がある FlexVol の名前を指定します。この FlexVol には、インポートが必要な LUN が 1 つしか含まれていないことに注意してください。
- とともに使用する必要がある PVC 定義のパスを指定します `-f` フラグ。
- PVC を管理するか、管理対象外にするかを選択します。デフォルトでは、Trident によって PVC が管理され、バックエンドの FlexVol と LUN の名前が変更されます。管理対象外のボリュームとしてインポートするには、`--no-manage` フラグ。



管理対象外のをインポートする場合 `ontap-san` ボリューム：FlexVol 内の LUN の名前がになっていることを確認します `lun0` とは、目的のイニシエータを含む `igroup` にマッピングされている。Trident が管理対象のインポートに対して自動的に処理します。

次に、Astra Trident が FlexVol をインポートし、PVC 定義に関連付けます。Astra Trident は、FlexVol の名前もに変更します `pvc-<uuid>` および FlexVol 内の LUN をからにフォーマットします `lun0`。



既存のアクティブな接続がないボリュームをインポートすることを推奨します。アクティブに使用されているボリュームをインポートする場合は、最初にボリュームをクローニングしてからインポートを実行します。

例

をインポートします `ontap-san-managed` にある FlexVol `ontap_san_default` バックエンドでを実行します `tridentctl import` コマンドの形式：

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-basic-import.yaml -n trident -d
```

```
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS |
+-----+-----+-----+-----+
| PROTOCOL |          BACKEND UUID          |  STATE  | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic          |
+-----+-----+-----+-----+
| block   | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```



ONTAP ボリュームのタイプが RW であることが Astra Trident でインポートされる必要があります。DP タイプのボリュームは SnapMirror デスティネーションボリュームです。ボリュームを Astra Trident にインポートする前に、ミラー関係を解除する必要があります。

element インポート

Trident を使用して、NetApp Element ソフトウェア / NetApp HCI ボリュームを Kubernetes クラスタにインポートできます。必要に応じて、Astra Trident バックエンドの名前、ボリュームと PVC ファイルの一意の名前をの引数として指定します `tridentctl import` コマンドを実行します

```
tridentctl import volume element_default element-managed -f pvc-basic-import.yaml -n trident -d
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+
|          BACKEND UUID  | STATE | MANAGED |
+-----+-----+-----+-----+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
+-----+-----+-----+-----+
| block      | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```



Element ドライバではボリューム名の重複がサポートされます。ボリューム名が重複している場合、Trident のボリュームインポートプロセスはエラーを返します。回避策として、ボリュームをクローニングし、一意のボリューム名を指定します。次に、クローンボリュームをインポートします。

gcp-cvs インポート



GCP の NetApp Cloud Volumes Service から作成されたボリュームをインポートするには、名前ではなくボリュームパスでボリュームを特定します。

をインポートします gcp-cvs バックエンドのボリュームの名前はです gcpcvs_YEppr を指定します `adroit-jolly-swift` では、次のコマンドを使用します。

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-file> -n trident
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+
|          BACKEND UUID  | STATE | MANAGED |
+-----+-----+-----+-----+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage | file
+-----+-----+-----+-----+
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```



ボリュームパスは、/ のあとのボリュームのエクスポートパスの部分です。たとえば、エクスポートパスがの場合などです 10.0.0.1:/adroit-jolly-swift、ボリュームのパスはです adroit-jolly-swift。

azure-netapp-files インポート

をインポートします azure-netapp-files バックエンドのボリュームの名前はです
azurenetaappfiles_40517 を指定します `importvol1` を使用して、次のコマンドを実行します。

```
tridentctl import volume azurenetaappfiles_40517 importvol1 -f <path-to-pvc-file> -n trident
```

PROTOCOL	NAME	SIZE	STORAGE CLASS	BACKEND UUID	STATE	MANAGED
file	pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab	100 GiB	anf-storage	1c01274f-d94b-44a3-98a3-04c953c9a51e	online	true



ANF ボリュームのボリュームパスは、/ のあとのマウントパスにあります。たとえば、マウントパスがの場合などです 10.0.0.2:/importvol1、ボリュームのパスはです importvol1。

ネームスペース間でNFSボリュームを共有します

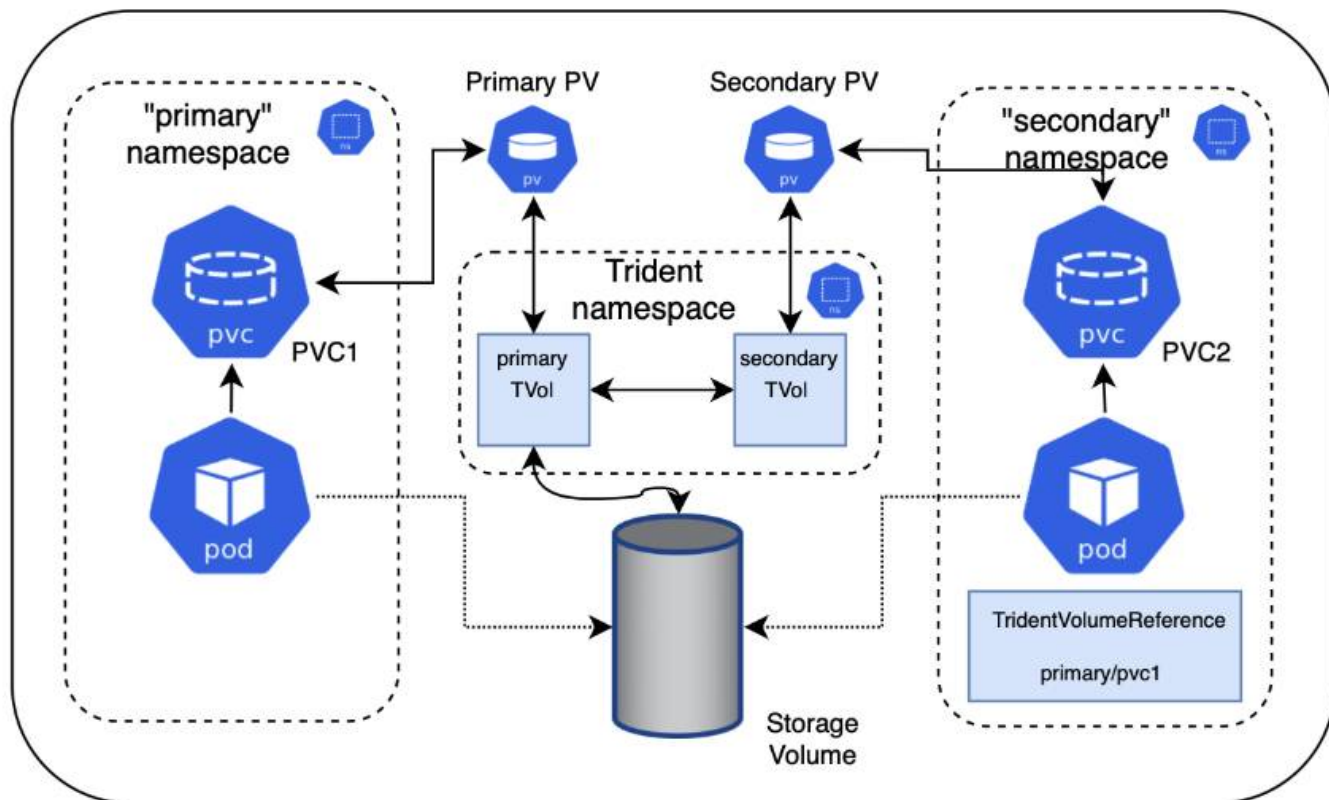
Tridentを使用すると、プライマリネームスペースにボリュームを作成し、1つ以上のセカンダリネームスペースで共有できます。

の機能

Astra TridentVolumeReference CRを使用すると、1つ以上のKubernetesネームスペース間でReadWriteMany (RWX) NFSボリュームをセキュアに共有できます。このKubernetesネイティブ解決策 には、次のようなメリットがあります。

- セキュリティを確保するために、複数のレベルのアクセス制御が可能です
- すべてのTrident NFSボリュームドライバで動作
- tridentctlやその他の非ネイティブのKubernetes機能に依存しません

この図は、2つのKubernetesネームスペース間でのNFSボリュームの共有を示しています。



クイックスタート

NFSボリューム共有はいくつかの手順で設定できます。

1

ボリュームを共有するようにソース**PVC**を設定します

ソース名前空間の所有者は、ソースPVCのデータにアクセスする権限を付与します。

2

デスティネーション名前空間に**CR**を作成する権限を付与します

クラスタ管理者が、デスティネーション名前空間の所有者にTridentVolumeReference CRを作成する権限を付与します。

3

デスティネーション名前空間に**TridentVolumeReference**を作成します

宛先名前空間の所有者は、送信元PVCを参照するためにTridentVolumeReference CRを作成します。

4

宛先名前空間に下位**PVC**を作成します

宛先名前空間の所有者は、送信元PVCからのデータソースを使用する下位PVCを作成します。

ソースネームスペースとデスティネーションネームスペースを設定します

セキュリティを確保するために、ネームスペース間共有では、ソースネームスペースの所有者、クラスタ管理者、および宛先ネームスペースの所有者によるコラボレーションとアクションが必要です。ユーザロールは各手順で指定します。

手順

1. ソース名前空間の所有者：PVCを作成します (pvc1) をソースネームスペースに追加し、デスティネーションネームスペースとの共有権限を付与します (namespace2)を使用します shareToNamespace アノテーション

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Astra TridentがPVとバックエンドのNFSストレージボリュームを作成



- カンマ区切りリストを使用して、複数の名前空間にPVCを共有できます。例：
trident.netapp.io/shareToNamespace:
namespace2, namespace3, namespace4。
- を使用して、すべてのネームスペースに共有できます *。例：
trident.netapp.io/shareToNamespace: *
- PVCを更新してを含めることができます shareToNamespace アノテーションはいつでも作成できます。

2. *クラスタ管理者：*カスタムロールとkubecfgを作成して、デスティネーションネームスペースの所有者にTridentVolumeReference CRを作成する権限を付与します。
3. *デスティネーションネームスペース所有者：*ソースネームスペースを参照するデスティネーションネームスペースにTridentVolumeReference CRを作成します pvc1。

```

apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1

```

4. 宛先名前空間の所有者：PVCを作成します (pvc2) をデスティネーションネームスペースに展開します (namespace2)を使用します shareFromPVC 送信元PVCを指定する注釈。

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi

```



宛先PVCのサイズは、送信元PVCのサイズ以下である必要があります。

結果

Astra Tridentがを読み取り shareFromPVC デスティネーションPVCにアノテーションを設定し、ソースPVを参照するストレージリソースを持たない下位のボリュームとしてデスティネーションPVを作成し、ソースPVストレージリソースを共有します。宛先PVCとPVは、通常どおりバインドされているように見えます。

共有ボリュームを削除

複数のネームスペースで共有されているボリュームは削除できます。Tridentが、ソースネームスペースのボリュームへのアクセスを削除し、ボリュームを共有する他のネームスペースへのアクセスを維持します。ボリュームを参照するすべてのネームスペースが削除されると、Astra Tridentによってボリュームが削除されます。

使用 tridentctl get 下位のボリュームを照会する

を使用する[tridentctl ユーティリティを使用すると、を実行できます get コマンドを使用して下位のボリュームを取得します。詳細については、リンク:./trident-reference/tridentctl.htmlを参照してください

[tridentctl コマンドとオプション]。

```
Usage:
  tridentctl get [option]
```

フラグ：

- `-h, --help`：ボリュームのヘルプ。
- `--parentOfSubordinate string`：クエリを下位のソースボリュームに制限します。
- `--subordinateOf string`：クエリをボリュームの下位に制限します。

制限

- Astra Tridentでは、デスティネーションネームスペースが共有ボリュームに書き込まれるのを防ぐことはできません。共有ボリュームのデータの上書きを防止するには、ファイルロックなどのプロセスを使用する必要があります。
- を削除しても、送信元PVCへのアクセスを取り消すことはできません `shareToNamespace` または `shareFromNamespace` 注釈またはを削除します `TridentVolumeReference` CR。アクセスを取り消すには、下位PVCを削除する必要があります。
- Snapshot、クローン、およびミラーリングは下位のボリュームでは実行できません。

を参照してください。

ネームスペース間のボリュームアクセスの詳細については、次の資料を参照してください。

- にアクセスします ["ネームスペース間でのボリュームの共有：ネームスペース間のボリュームアクセスを許可する場合は「Hello」と入力します"](#)。
- のデモをご覧ください ["ネットアップTV"](#)。

Astra Trident を監視

Astra Trident は、Astra Trident のパフォーマンスを監視するために使用できる一連の Prometheus 指標エンドポイントを提供します。

Astra Trident が提供する指標を使用すると、次のことが可能になります。

- Astra Trident の健全性と設定を保持処理が成功した方法と、想定どおりにバックエンドと通信できるかどうかを調べることができます。
- バックエンドの使用状況の情報を調べて、バックエンドでプロビジョニングされているボリュームの数や消費されているスペースなどを確認します。
- 利用可能なバックエンドにプロビジョニングされたボリュームの量のマッピングを維持します。
- パフォーマンスを追跡する。Astra Trident がバックエンドと通信して処理を実行するのにどれくらいの時間がかかるかを調べることができます。



デフォルトでは、Tridentの指標はターゲットポートで公開されています 8001 で /metrics エンドポイント。これらの指標は、Trident のインストール時にデフォルトで * 有効になります。

必要なもの

- Astra Trident がインストールされた Kubernetes クラスタ
- Prometheus インスタンス。これは a である場合もある "[コンテナ化された Prometheus 環境](#)" または、Prometheus をとして実行することもできます "[ネイティブアプリケーション](#)"。

手順 1 : Prometheus ターゲットを定義する

Prometheus ターゲットを定義して指標を収集し、Astra Trident が管理するバックエンド、作成するボリュームなどの情報を取得する必要があります。これ "[ブログ](#)" Prometheus と Grafana を Astra Trident とともに使用して指標を取得する方法について説明します。ブログでは、Kubernetes クラスタでオペレータとして Prometheus を実行する方法と、Astra Trident のメトリックを取得する ServiceMonitor を作成する方法について説明しています。

手順 2 : Prometheus ServiceMonitor を作成します

Tridentの指標を利用するには、を監視するPrometheus ServiceMonitorを作成する必要があります trident-csi サービスおよびリッスン metrics ポート：ServiceMonitor のサンプルは次のようになります。

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

このServiceMonitor定義は、から返されたメトリックを取得します trident-csi サービスとは、を特に探します metrics サービスのエンドポイント。その結果、PrometheusはAstra Tridentの指標：

Astra Tridentから直接取得できる指標に加えて、kubeletは多くの指標を公開しています kubelet_volume_* 独自の指標エンドポイントを使用した指標。Kubelet では、接続されているボリュームに関する情報、および

ポッドと、それが処理するその他の内部処理を確認できます。を参照してください ["こちらをご覧ください"](#)。

ステップ 3 : **PromptQL** を使用して **Trident** 指標を照会する

PromptQL は、時系列データまたは表データを返す式を作成するのに適しています。

次に、PromptQL クエリーのいくつかを示します。

Trident の健全性情報を取得

- **Astra Trident** からの **HTTP 2XX** 応答の割合

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()  
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Astra Trident** からのステータスコードによる **REST** 応答の割合

```
(sum (trident_rest_ops_seconds_total_count) by (status_code) / scalar  
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Astra Trident** によって実行された処理の平均時間 (ミリ秒)

```
sum by (operation)  
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by  
(operation)  
(trident_operation_duration_milliseconds_count{success="true"})
```

Astra Trident の使用状況に関する情報を入手

- 平均体積サイズ

```
trident_volume_allocated_bytes/trident_volume_count
```

- 各バックエンドによってプロビジョニングされた合計ボリューム容量

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

個々のボリュームの使用状況を取得する



これは、kubelet 指標も収集された場合にのみ有効になります。

- 各ボリュームの使用済みスペースの割合

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes * 100
```

Astra Trident AutoSupport の計測データ

デフォルトでは、Astra Trident は Prometheus 指標と基本バックエンド情報を毎日定期的にネットアップに送信します。

- Astra TridentからPrometheus指標や基本バックエンド情報がネットアップに送信されないようにするには、を渡します `--silence-autosupport` Astra Tridentのインストール中にフラグを付ける。
- Tridentからネットアップサポートにコンテナログをオンデマンドで送信することもできます `tridentctl send autosupport`。Astra Trident をトリガーしてログをアップロードする必要があります。ログを送信する前に、ネットアップのに同意する必要があります ["プライバシーポリシー"](#)。
- 指定しないと、Astra Trident は過去 24 時間からログを取得します。
- ログの保持期間はで指定できます `--since` フラグ。例： `tridentctl send autosupport --since=1h`。この情報は、を介して収集および送信されます `trident-autosupport` コンテナこれはAstra Tridentと一緒にインストールされます。コンテナイメージは、で取得できます ["Trident AutoSupport の略"](#)。
- Trident AutoSupport は、個人情報（PII）や個人情報を収集または送信しません。それにはが付いています ["EULA"](#) これは Trident コンテナイメージ自体には該当しません。ネットアップのデータセキュリティと信頼に対する取り組みの詳細を確認できます ["こちらをご覧ください"](#)。

Astra Trident から送信されるペイロードの例を次に示します。

```
{
  "items": [
    {
      "backendUUID": "ff3852e1-18a5-4df4-b2d3-f59f829627ed",
      "protocol": "file",
      "config": {
        "version": 1,
        "storageDriverName": "ontap-nas",
        "debug": false,
        "debugTraceFlags": null,
        "disableDelete": false,
        "serialNumbers": [
          "nwkvzfanek_SN"
        ],
        "limitVolumeSize": ""
      },
      "state": "online",
      "online": true
    }
  ]
}
```

- AutoSupport メッセージは、ネットアップの AutoSupport エンドポイントに送信されます。コンテナイメージの格納にプライベートレジストリを使用している場合は、`--image-registry` フラグ。
- インストール YAML ファイルを生成してプロキシ URL を設定することもできます。これは、`--generate-custom-yaml` を使用して実行できます `tridentctl install --generate-custom-yaml` YAML ファイルを作成し、`--proxy-url` の引数 `trident-autosupport` にコンテナがあります `trident-deployment.yaml`。

Astra Trident の指標を無効化

****メトリックがレポートされないようにするには、を使用してカスタムYAMLを生成する必要があります**
`--generate-custom-yaml` フラグを付けて編集し、`--metrics` に対する呼び出し元からのフラグ `trident-main`
 コンテナ：

著作権に関する情報

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

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

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

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

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

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

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

商標に関する情報

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