バックエンドの構成と管理

Astra Trident

NetApp
March 05, 2026

目次

バックエンドの構成と管理

バックエンドを設定

バックエンドは、 Astra Trident とストレージシステムの関係を定義します。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、ストレージクラスによって定義された要件に一致するストレージプールをバックエンドから自動的に提供します。ストレージシステムにバックエンドを設定する方法について説明します。

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

Azure NetApp Files

Azure NetApp Files バックエンドを設定します

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

Azure NetApp Filesドライバの詳細

Astra Tridentは、次のAzure NetApp Filesストレージドライバを使用してクラスタと通信します。サポートされているアクセスモードは、*ReadWriteOnce*(RWO)、*ReadOnlyMany*(ROX)、*ReadWriteMany*(RWX)、*ReadWriteOncePod*(RWOP)です。

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

考慮事項

- Azure NetApp Files サービスでは、100GB未満のボリュームはサポートされません。容量の小さいボリュームが要求されると、Astra Tridentによって自動的に100GiBのボリュームが作成されます。
- 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

AKSの管理対象ID

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

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

Trident オペレータ

Tridentオペレータを使用してAstra Tridentをインストールするには、を編集し `tridentorchestrator_cr.yaml` てに設定します `cloudProvider` `"Azure"`。例：

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

Helm

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

```
helm install trident trident-operator-100.2406.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code> tridentctl </code>

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

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

AKSのクラウドID

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

Azureでクラウドを活用するには、以下が必要です。

- AKSを使用して導入されるKubernetesクラスタ
- AKS Kubernetesクラスタに設定されたワークロードIDとoidc-issuer
- ワークロードIDを指定し `"Azure"`て `cloudIdentity`指定するを含む`Astra Trident`をインストール `cloudProvider`

Trident オペレータ

Tridentオペレータを使用してAstra Tridentをインストールするには、を編集し `tridentorchestrator_cr.yaml`て、をに `"Azure"`設定 `cloudProvider`し `cloudIdentity`azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`ます。

例：

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

Helm

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

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

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

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

<code> tridentctl </code>

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

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

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

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

Azure NetApp Files バックエンドを設定する準備をします

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

NFSボリュームと**SMB**ボリュームの前提条件

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

バックエンドを設定して使用するには "Azure NetApp Files"、次のものが必要です。

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

- 容量プール。を参照してください "Microsoft：Azure NetApp Files 用の容量プールを作成します"。
- Azure NetApp Files に委任されたサブネット。を参照してください "Microsoft：サブネットをAzure NetApp Files に委任します"。
- `subscriptionID`Azure NetApp Filesを有効にしたAzureサブスクリプションから削除します。
- `tenantID` `clientID` `clientSecret`Azure NetApp Filesサービスへの十分な権限を持つ、Azure Active Directory内のから"アプリケーション登録"。アプリケーション登録では、次のいずれかを使用します。
  - 所有者ロールまたは寄与者ロール"Azureで事前定義"。
  - "カスタム投稿者ロール"(`assignableScopes`次の権限が付与されます。権限はAstra Tridentで必要な権限のみに制限されます。カスタムロールを作成したら、"Azureポータルを使用してロールを割り当てます"を参照してください。

カスタムコントリビュータロール

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

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

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

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

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

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

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

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

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

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
                    "Microsoft.Network/virtualNetworks/read",

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

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",
```

```
"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/delete",
                    "Microsoft.Features/features/read",
                    "Microsoft.Features/operations/read",
                    "Microsoft.Features/providers/features/read",

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

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

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

- 少なくとも1つを含む "委任されたサブネット"Azure `location`。Trident 22.01では、この `location`パラメータはバックエンド構成ファイルの最上位レベルにある必須フィールドです。仮想プールで指定された場所の値は無視されます。
- を使用するに `Cloud Identity`は、から "ユーザーが割り当てた管理ID"を取得し `client ID`、でそのIDを指定します `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`。

SMBボリュームに関するその他の要件

SMBボリュームを作成するには、以下が必要です。

- Active Directoryが設定され、Azure NetApp Files に接続されています。を参照してください "Microsoft：Azure NetApp Files のActive Directory接続を作成および管理します"。
- LinuxコントローラノードとWindowsワーカーノードで少なくとも1つのWindows Server 2022を実行しているKubernetesクラスタ。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
- Azure NetApp Files がActive Directoryに対して認証できるように、Active Directoryクレデンシャルを含むAstra Tridentのシークレットが少なくとも1つ含まれています。シークレットを生成するには `smbcreds`：

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

- Windowsサービスとして設定されたCSIプロキシ。を設定するには `csi-proxy`、Windowsで実行されているKubernetesノードについて、またはを"GitHub: Windows向けCSIプロキシ"参照してください"GitHub: CSIプロキシ"。

Azure NetApp Files バックエンド構成のオプションと例

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

バックエンド構成オプション

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、要求された場所で使用可能な容量プールに、要求されたサービスレベルとサブネットに一致するAzure NetApp Filesボリュームを作成します。

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

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

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

パラメータ	説明	デフォルト
`resourceGroups`	検出されたリソースをフィルタリングするためのリソースグループのリスト	"[]" （フィルタなし）
`netappAccounts`	検出されたリソースをフィルタリングするためのネットアップアカウントのリスト	"[]" （フィルタなし）
`capacityPools`	検出されたリソースをフィルタリングする容量プールのリスト	"[]" （フィルタなし、ランダム）
`virtualNetwork`	委任されたサブネットを持つ仮想ネットワークの名前	""
`subnet`	委任先のサブネットの名前 `Microsoft.Netapp/volumes`	""
`networkFeatures`	ボリュームのVNet機能のセットは `Basic`、または `Standard`です。ネットワーク機能は一部の地域では使用できず、サブスクリプションで有効にする必要がある場合があります。この機能を有効にしないタイミングを指定する `networkFeatures`と、ボリュームのプロビジョニングが失敗します。	""
`nfsMountOptions`	NFS マウントオプションのきめ細かな制御。SMBボリュームでは無視されます。NFSバージョン4.1を使用してボリュームをマウントするには、カンマで区切ったマウントオプションのリストにを追加してNFS v4.1を `nfsvers=4`選択します。ストレージクラス定義で設定されたマウントオプションは、バックエンド構成で設定されたマウントオプションよりも優先されます。	"nfsvers=3 "
`limitVolumeSize`	要求されたボリュームサイズがこの値を超えている場合はプロビジョニングが失敗します	"" （デフォルトでは適用されません）
`debugTraceFlags`	トラブルシューティング時に使用するデバッグフラグ。例： `\{"api": false, "method": true, "discovery": true}`トラブルシューティングを行って詳細なログダンプが必要な場合を除き、このオプションは使用しないでください。	null

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

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

必要な権限とリソース

PVCの作成時に「No capacity pools found」エラーが表示される場合は、アプリケーション登録に必要な権限とリソース（サブネット、仮想ネットワーク、容量プール）が関連付けられていない可能性があります。デバッグ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。

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

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

タイプ	形式
リソースグループ	< 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。詳細については、を参照してください [構成例] 。

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

構成例

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

最小限の構成

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

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

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

AKSの管理対象**ID**

このバックエンド構成では、、 `tenantID`、 `clientID`、が `clientSecret`省略されてい `subscriptionID` ます。これらは、管理対象IDを使用する場合はオプションです。

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

AKSのクラウド**ID**

このバックエンド構成では、クラウドIDを使用する場合はオプションである、、 `clientID`、が `clientSecret`省略されて `tenantID`います。

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

容量プールフィルタを使用した特定のサービスレベル構成

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

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

高度な設定

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

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

仮想プール構成

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

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

サポートされるトポロジ構成

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

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

ストレージクラスの定義

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

フィールドヲシヨウシタテイノレイ `parameter.selector`

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

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

SMBボリュームの定義例

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

デフォルトネームスペースの基本設定

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

ネームスペースごとに異なるシークレットを使用する

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

ボリュームごとに異なるシークレットを使用する

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

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

バックエンドを作成します

バックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

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

```
tridentctl logs
```

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

Google Cloud NetAppボリューム

Google Cloud NetApp Volumeバックエンドの設定

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

```
 Google Cloud NetApp Volumes is a tech preview feature in Astra Trident
24.06.
```

Google Cloud NetApp Volumesドライバの詳細

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

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

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

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

NFSボリュームノゼンテイジョウケン

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

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

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

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

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

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

バックエンド構成オプション

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

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

パラメータ	説明	デフォルト
`location`	Astra TridentがGCNVボリュームを作成するGoogle Cloudの場所。リージョン間Kubernetesクラスタを作成する場合、で作成したボリュームは `location` 、複数のGoogle Cloudリージョンのノードでスケジュールされているワークロードで使用できます。リージョン間トラフィックは追加コストを発生させます。	
`apiKey`	ロールが割り当てられたGoogle Cloudサービスアカウントの`APIキー` `netappcloudvolumes.admin` 。このレポートには、 Google Cloud サービスアカウントの秘密鍵ファイルの JSON 形式のコンテンツが含まれています（バックエンド構成ファイルにそのままコピーされます）。には `apiKey` 、、、、、の各キーのキーと値のペアを含める必要があります。 `type` `project_id` `client_email` `client_id` `auth_uri` `token_uri` `auth_provider_x509_cert_url`、、および `client_x509_cert_url`。	
`nfsMountOptions`	NFS マウントオプションのきめ細かな制御。	"nfsvers=3 "
`limitVolumeSize`	要求されたボリュームサイズがこの値を超えている場合はプロビジョニングが失敗します。	"" （デフォルトでは適用されません）
`serviceLevel`	ストレージプールとそのボリュームのサービスレベル。値は `flex`、 `standard`、 `premium`、または `extreme`です。	
`network`	GCNVボリュームに使用されるGoogle Cloudネットワーク。	
`debugTraceFlags`	トラブルシューティング時に使用するデバッグフラグ。例： `{"api":false, "method":true}`トラブルシューティングを行って詳細なログダンプが必要な場合を除き、このオプションは使用しないでください。	null

`supportedTopologies`	このバックエンドでサポートされているリージョンとゾーンのリストを表します。詳細については、を参照してください "CSI トポロジを使用します"。例： `supportedTopologies:`
	`- topology.kubernetes.io/region: europe-west6`
	`topology.kubernetes.io/zone: europe-west6-b`	

ボリュームプロビジョニングオプション

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

パラメータ	説明	デフォルト
`exportRule`	新しいボリュームのエクスポートルール。IPv4アドレスの任意の組み合わせをカンマで区切って指定する必要があります。	"0.0.0.0/0 "
`snapshotDir`	ディレクトリへのアクセス `.snapshot`	いいえ
`snapshotReserve`	Snapshot 用にリザーブされているボリュームの割合	""（デフォルトの0を使用）
`unixPermissions`	新しいボリュームのUNIX権限（8進数の4桁）。	""

構成例

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

最小限の構成

これは、バックエンドの絶対的な最小構成です。この構成では、Astra Tridentが、設定された場所にあるGoogle Cloud NetApp Volumeに委譲されたすべてのストレージプールを検出し、それらのプールの1つに新しいボリュームをランダムに配置します。は省略されているため、 `nasType` `nfs` デフォルトが適用され、バックエンドでNFSボリュームがプロビジョニングされます。

この構成は、Google Cloud NetApp Volumeの使用を開始して試用する場合に最適ですが、実際には、プロビジョニングするボリュームに対して追加の範囲設定が必要になることがよくあります。

```
---

apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: 'f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec'
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
```

```
    -----END PRIVATE KEY-----

---

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

StoragePoolsフィルタヲシヨウシタセツテイ

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: 'f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec'
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
```

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

仮想プール構成

このバックエンド構成では、1つのファイルに複数の仮想プールが定義されます。仮想プールは、セクションで定義し storage ます。さまざまなサービスレベルをサポートする複数のストレージプールがあり、それらを表すストレージクラスをKubernetesで作成する場合に役立ちます。仮想プールラベルは、プールを区別するために使用されます。たとえば、次の例では performance 、仮想プールを区別するためにラベルと serviceLevel タイプが使用されています。

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

詳細については、を参照してください "仮想プール"。

```
---

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

```
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---

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

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

次の手順

バックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

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

```
kubectl get tridentbackendconfig

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

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

```
tridentctl logs
```

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

その他の例

ストレージクラスの定義の例

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

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

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

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

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

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

PVC定義の例PVCテイギノレイ

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

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

```
kubectl get pvc gcnv-nfs-pvc

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

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

ネットアップCloud Volumes Service for Google CloudをAstra Tridentのバックエンドとして構成する方法を、提供されている構成例を使用して説明します。

Google Cloudドライバの詳細

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

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

Cloud Volumes Service for Google Cloudに対するAstra Tridentサポートの詳細をご確認ください

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

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

必要なもの

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

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

バックエンド構成オプション

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

パラメータ	説明	デフォルト
`version`		常に 1
`storageDriverName`	ストレージドライバの名前	"GCP-cvs"
`backendName`	カスタム名またはストレージバックエンド	ドライバ名 + "_" + API キーの一部
`storageClass`	CVSサービスタイプを指定するためのオプションのパラメータ。CVSサービスタイプを選択するために使用し `software`ます。それ以外の場合、Astra Tridentはサービスタイプがcvs-Performanceとみなされ(`hardware`ます）。	
`storagePools`	CVSサービスタイプのみ。ボリューム作成用のストレージプールを指定するオプションのパラメータ。	
`projectNumber`	Google Cloud アカウントのプロジェクト番号。この値は、Google Cloudポータルのホームページにあります。	

パラメータ	説明	デフォルト
`hostProjectNumber`	共有VPCネットワークを使用する場合は必須です。このシナリオでは、 `projectNumber`はサービスプロジェクト、 `hostProjectNumber`はホストプロジェクトです。	
`apiRegion`	Astra TridentがCloud Volumes Service ボリュームを作成するGoogle Cloudリージョン。リージョン間Kubernetesクラスタを作成する場合、で作成したボリュームは `apiRegion`、複数のGoogle Cloudリージョンのノードでスケジュールされているワークロードで使用できます。リージョン間トラフィックは追加コストを発生させます。	
`apiKey`	ロールが割り当てられたGoogle Cloudサービスアカウントの`APIキー` `netappcloudvolumes.admin` 。このレポートには、 Google Cloud サービスアカウントの秘密鍵ファイルの JSON 形式のコンテンツが含まれています（バックエンド構成ファイルにそのままコピーされます）。	
`proxyURL`	CVSアカウントへの接続にプロキシサーバが必要な場合は、プロキシURLを指定します。プロキシサーバには、 HTTP プロキシまたは HTTPS プロキシを使用できます。HTTPS プロキシの場合、プロキシサーバで自己署名証明書を使用するために証明書の検証はスキップされます。認証が有効になっているプロキシサーバはサポートされていません。	
`nfsMountOptions`	NFS マウントオプションのきめ細かな制御。	"nfsvers=3 "
`limitVolumeSize`	要求されたボリュームサイズがこの値を超えている場合はプロビジョニングが失敗します。	"" （デフォルトでは適用されません）
`serviceLevel`	新しいボリュームのCVS -パフォーマンスレベルまたはCVSサービスレベル。CVS-Performanceの値は `standard`、、 `premium`または `extreme`です。CVS値は `standardsw`または `zoneredundantstandardsw`です。	CVS -パフォーマンスのデフォルトは「Standard」です。CVSのデフォルトは"standardsw"です。
`network`	Cloud Volumes Service ボリュームに使用するGoogle Cloudネットワーク。	デフォルト
`debugTraceFlags`	トラブルシューティング時に使用するデバッグフラグ。例： `\{"api":false, "method":true}`トラブルシューティングを行って詳細なログダンプが必要な場合を除き、このオプションは使用しないでください。	null

`allowedTopologies`	リージョン間アクセスを有効にするには、のStorageClass定義 `allowedTopologies`にすべてのリージョンが含まれている必要があります。例：
	`- key: topology.kubernetes.io/region`
	`values:`
	`- us-east1`
	`- europe-west1`	

ボリュームプロビジョニングオプション

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

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

CVS -パフォーマンスサービスの種類の例

次の例は、CVS -パフォーマンスサービスタイプの設定例を示しています。

例 **1** ：最小限の構成

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

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

例**2**：サービスレベルの設定

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

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

例3：仮想プールの構成

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

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

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

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

ストレージクラスの定義

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

ストレージクラスの例

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

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

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

CVSサービスタイプの例

次の例は、CVSサービスタイプの設定例を示しています。

例**1**：最小構成

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

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

例**2**：ストレージプールの構成

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

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

次の手順

バックエンド構成ファイルを作成したら、次のコマンドを実行します。

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

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

```
tridentctl logs
```

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

NetApp HCI または SolidFire バックエンドを設定します

Astra Tridentインストール環境でElementバックエンドを作成して使用する方法をご確認ください。

Elementドライバの詳細

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

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

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

開始する前に

Elementバックエンドを作成する前に、次の情報が必要になります。

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

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
`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。Astra TridentはCHAPを使用	正しい
`AccessGroups`	使用するアクセスグループ ID のリスト	「 trident 」という名前のアクセスグループの ID を検索します。
`Types`	QoS の仕様	
`limitVolumeSize`	要求されたボリュームサイズがこの値を超えている場合、プロビジョニングが失敗します	“”（デフォルトでは適用されません）
`debugTraceFlags`	トラブルシューティング時に使用するデバッグフラグ。例： {"API" ： false 、 "method" ： true}	null

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

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

次の例は、 CHAP 認証を使用するバックエンドファイルと、特定の QoS 保証を適用した 3 つのボリュームタイプのモデリングを示しています。次に、ストレージクラスパラメータを使用して各ストレージクラスを使用するように定義します `IOPS`。

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

例2：仮想プールを使用するドライバのバックエンドとストレージクラスの構成 `solidfire-san`

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

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バックエンドストレージLUNにコピーします。ストレージ管理者は、仮想プールごとにラベルを定義したり、ボリュームをラベルでグループ化したりできます。

以下に示すサンプルのバックエンド定義ファイルでは、すべてのストレージプールに特定のデフォルトが設定されており、そのデフォルトはAt Silverに設定されて `type`います。仮想プールは、セクションで定義し `storage`ます。この例では、一部のストレージプールが独自のタイプを設定し、一部のプールが上記のデフォルト値を上書きします。

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
```

```
SVIP: "<svip>:3260"
TenantName: "<tenant>"
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
- labels:
    performance: gold
    cost: '4'
  zone: us-east-1a
  type: Gold
- labels:
    performance: silver
    cost: '3'
  zone: us-east-1b
  type: Silver
- labels:
    performance: bronze
    cost: '2'
  zone: us-east-1c
  type: Bronze
- labels:
    performance: silver
    cost: '1'
  zone: us-east-1d
```

次のStorageClass定義は、上記の仮想プールを参照しています。フィールドを使用して

parameters.selector、各StorageClassはボリュームのホストに使用できる仮想プールを呼び出します。ボリュームには、選択した仮想プール内で定義された要素があります。

最初のStorageClass(solidfire-gold-four）が最初の仮想プールにマッピングされます。これは、ゴールドのパフォーマンスとゴールドのパフォーマンスを提供する唯一のプールです Volume Type QoS。最後のStorageClass(solidfire-silver）は、Silverパフォーマンスを提供するストレージプールを呼び出します。Tridentが、どの仮想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"
```

詳細情報

- "ボリュームアクセスグループ"

ONTAP SANドライバ

ONTAP SANドライバの概要

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

ONTAP SANドライバの詳細

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

保護、リカバリ、モビリティにAstra Controlを使用している場合は、を参照してください。Astra Controlドライバの互換性

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

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

Astra Controlドライバの互換性

Astra Controlは、 ontap-nas-flexgroup、、および ontap-san`ドライバ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（Kubernetesクラスタ間でのボリューム移動）を提供します `ontap-nas。詳細については、を参照してください "Astra Controlレプリケーションの前提条件" 。

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

ユーザ権限

Astra Tridentは、ONTAP管理者またはSVM管理者（通常はクラスタユーザまたは vsadmin`SVMユーザ、または同じロールの別の名前のユーザを使用）として実行することを想定しています `admin。Amazon FSx for NetApp ONTAP環境の場合、Astra Tridentは、クラスタユーザまたは `vsadmin`SVMユーザ、または同じロールの別の名前のユーザを使用して、ONTAP管理者またはSVM管理者として実行されることが想定され `fsxadmin`ます。この `fsxadmin`ユーザは、クラスタ管理者ユーザに代わる限定的なユーザです。

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

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

NVMe/TCPに関するその他の考慮事項

Astra Tridentでは、次のドライバを使用してNon-Volatile Memory Express（NVMe）プロトコルがサポートされ `ontap-san`ます。

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

Astra Tridentでは次の機能がサポートされません。

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

ONTAP SANドライバを使用してバックエンドを設定する準備をします

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

要件

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

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

ONTAPバックエンドの認証

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。ONTAPのバージョンと最大限の互換性を確保するために、や `vsadmin`などの事前定義されたセキュリティログインロールを使用することを推奨し `admin`ます。
- 証明書ベース： 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。や `vsadmin`などの事前定義された標準のロールを使用することを推奨します `admin`。これにより、今後のリリースの 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。

YAML

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

JSON

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

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

証明書ベースの認証を有効にする

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

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

一般的なワークフローは次の手順で構成されます。

手順

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

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

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

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

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

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

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

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

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

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

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

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

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

7. 前の手順で得た値を使用してバックエンドを作成します。

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

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

認証方法を更新するか、クレデンシャルをローテーションして

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

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

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

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

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

双方向 **CHAP** を使用して接続を認証します

Astra Tridentでは、ドライバと `ontap-san-economy`ドライバの双方向CHAPを使用してiSCSIセッションを認証できます `ontap-san`。これには、バックエンド定義でオプションを有効にする必要があり`useCHAP`ます。に設定する `true`と、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に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: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

`useCHAP`パラメータはブール値のオプションで、一度だけ設定できます。デフォルトでは false に設定されています。true に設定したあとで、 false に設定することはできません。

さらに `useCHAP=true`、 `chapInitiatorSecret`、 `chapTargetInitiatorSecret`、 `chapTargetUsername`、および `chapUsername`フィールドをバックエンド定義に含める必要があります。シークレットは、を実行してバックエンドを作成したあとに変更できます `tridentctl update`。

仕組み

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

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

バックエンドが作成されると、Astra Tridentは対応するCRDを作成し `tridentbackend`、CHAPシークレットとユーザ名をKubernetesシークレットとして格納します。このバックエンドの 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、ファイルのCHAPパラメータを更新し `backend.json`ます。そのためには、CHAPシークレットを更新し、コマンドを使用して変更を反映する必要があり `tridentctl update`ます。

バックエンドのCHAPシークレットを更新する場合は、を使用してバックエンドを更新する必要があります `tridentctl`。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": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

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

既存の接続は影響を受けません。 SVM 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 SANの設定オプションと例

Astra Tridentのインストール環境でONTAP SANドライバを作成して使用する方法をご紹介します。このセクションでは、バックエンドの構成例と、バックエンドをStorageClassesにマッピングするための詳細を示します。

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	説明	デフォルト
`version`		常に 1

パラメータ	説明	デフォルト
`storageDriverName`	ストレージドライバの名前	`ontap-nas` `ontap-nas-economy`、、 `ontap-nas-flexgroup`、、 `ontap-san` `ontap-san-economy`
`backendName`	カスタム名またはストレージバックエンド	ドライバ名+"_"+ dataLIF
`managementLIF`	クラスタ管理LIFまたはSVM管理LIFのIPアドレス。Fully Qualified Domain Name（FQDN；完全修飾ドメイン名）を指定できます。IPv6フラグを使用してAstra Tridentをインストールした場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`。シームレスなMetroClusterスイッチオーバーについては、を参照して[mcc-best]ください。	「 10.0.0.1 」、「 [2001:1234:abcd::fefe] 」
`dataLIF`	プロトコル LIF の IP アドレス。* iSCSIの場合は指定しないでください。**Astra Trident**は、を使用して**"ONTAP の選択的LUNマップ"**、マルチパスセッションの確立に必要な**iSCI LIF**を検出します。が明示的に定義されている場合は、警告が生成され `**dataLIF**` ます。 MetroClusterの場合は省略してください。*を参照してください[mcc-best]。	SVMの派生物です
`svm`	使用するStorage Virtual Machine * MetroClusterでは省略*を参照してください[mcc-best]。	SVMが指定されている場合に派生 `managementLIF`
`useCHAP`	CHAPを使用してONTAP SANドライバのiSCSIを認証します（ブーリアン）。Astra Tridentで、バックエンドで指定されたSVMのデフォルト認証として双方向CHAPを設定して使用する場合は、をに設定 `true` します。詳細については、を参照してください "ONTAP SANドライバを使用してバックエンドを設定する準備をします" 。	`false`
`chapInitiatorSecret`	CHAP イニシエータシークレット。必要な場合 `useCHAP=true`	""
`labels`	ボリュームに適用する任意の JSON 形式のラベルのセット	""
`chapTargetInitiatorSecret`	CHAP ターゲットイニシエータシークレット。必要な場合 `useCHAP=true`	""
`chapUsername`	インバウンドユーザ名。必要な場合 `useCHAP=true`	""
`chapTargetUsername`	ターゲットユーザ名。必要な場合 `useCHAP=true`	""
`clientCertificate`	クライアント証明書の Base64 エンコード値。証明書ベースの認証に使用されます	""
`clientPrivateKey`	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます	""

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。オプション。証明書ベースの認証に使用されます。	""
`username`	ONTAP クラスタとの通信に必要なユーザ名。クレデンシャルベースの認証に使用されます。	""
`password`	ONTAP クラスタとの通信にパスワードが必要です。クレデンシャルベースの認証に使用されます。	""
`svm`	使用する Storage Virtual Machine	SVMが指定されている場合に派生 `managementLIF`
`storagePrefix`	SVM で新しいボリュームをプロビジョニングする際に使用するプレフィックスを指定します。あとから変更することはできません。このパラメータを更新するには、新しいバックエンドを作成する必要があります。	`trident`
`limitAggregateUsage`	使用率がこの割合を超えている場合は、プロビジョニングが失敗します。Amazon FSx for NetApp ONTAP バックエンドを使用している場合は、を指定しないで `limitAggregateUsage`ください。指定されたと `vsadmin`には `fsxadmin`、アグリゲートの使用量を取得し、Astra Tridentを使用してアグリゲートを制限するために必要な権限が含まれていません。	"" （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。また、qtreeおよびLUNに対して管理するボリュームの最大サイズを制限します。	"" （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} トラブルシューティングを行って詳細なログダンプが必要な場合を除き、は使用しないでください。	`null`

パラメータ	説明	デフォルト
useREST	ONTAP REST API を使用するためのブーリアンパラメータ。 useREST に設定する true`と、Astra Trident はONTAP REST APIを使用してバックエンドと通信します。に設定する `false`と、Astra Trident はONTAP ZAPI呼び出しを使用してバックエンドと通信します。この機能にはONTAP 9.11.1以降が必要です。また、使用するONTAPログインロールには、アプリケーションへのアクセス権が必要です `ontap 。これは、事前に定義された役割と役割によって実現され vsadmin cluster-admin ます。Astra Trident 24.06リリースおよびONTAP 9.15.1以降では、が useREST デフォルトでに設定されてい true ます。ONTAP ZAPI呼び出しを使用するようににに変更してください。 useREST false useREST はNVMe/TCPに完全修飾されています。	true ONTAP 9.15.1以降の場合は、それ以外の場合は false。
sanType	iSCSIまたは `nvme`NVMe/TCPの選択に使用し `iscsi` ます。	`iscsi`空白の場合

ボリュームのプロビジョニング用のバックエンド構成オプション

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

パラメータ	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"
spaceReserve	スペースリザベーションモード：「none」（シン）または「volume」（シック）	"なし"
snapshotPolicy	使用する Snapshot ポリシー	"なし"
qosPolicy	作成したボリュームに割り当てる QoS ポリシーグループ。ストレージプール / バックエンドごとに QOSPolicy または adaptiveQosPolicy のいずれかを選択します。Trident 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。非共有のQoSポリシーグループを使用して、各コンスティチ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。	""
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	""

パラメータ	説明	デフォルト
`snapshotReserve`	Snapshot 用にリザーブされているボリュームの割合	が「none」の場合は「0」`snapshotPolicy`、それ以外の場合は「」
`splitOnClone`	作成時にクローンを親からスプリットします	いいえ
`encryption`	新しいボリュームでNetApp Volume Encryption（NVE）を有効にします。デフォルトはです。 `false`このオプションを使用するには、クラスタで NVE のライセンスが設定され、有効になっている必要があります。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に有効になります。詳細については、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
`luksEncryption`	LUKS暗号化を有効にします。を参照してください"Linux Unified Key Setup（LUKS；統合キーセットアップ）を使用"。LUKS暗号化はNVMe/TCPではサポートされません。	""
`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より前のSVM-DR設定の場合は「snapshot-only」
`nameTemplate`	カスタムボリューム名を作成するためのテンプレート。	""
`limitVolumePoolSize`	ONTAP SANエコノミーバックエンドでLUNを使用する場合の、要求可能な最大FlexVolサイズ。	""（デフォルトでは適用されません）

ボリュームプロビジョニングの例

デフォルトが定義されている例を次に示します。

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

ドライバを使用して作成されたすべてのボリュームについて、 `ontap-san`Astra TridentはLUNメタデータに対応するために10%の容量をFlexVolに追加します。LUN は、ユーザが PVC で要求したサイズとまったく同じサイズでプロビジョニングされます。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。

定義されたバックエンドについては `snapshotReserve`、Astra Tridentで次のようにボリュームのサイズが計算されます。

```
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。= 5%、PVC要求= 5GiBの場合、 `snapshotReserve`ボリュームの合計サイズは5.79GiB、使用可能なサイズは5.5GiBです。 `volume show`次の例のような結果が表示されます。

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
          _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                       online     RW          10GB     5.00GB    0%
          _pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d
                       online     RW        5.79GB     5.50GB    0%
          _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                       online     RW           1GB    511.8MB    0%
3 entries were displayed.
```

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

最小限の設定例

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

Amazon FSx on NetApp ONTAPとAstra 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 SANの例

これはドライバを使用した基本的な設定です `ontap-san`。

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

ONTAP SANの経済性の例

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

1. 例

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

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

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

証明書ベースの認証の例

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

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

双方向CHAPの例

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

ONTAP SAN CHAPの例

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

ONTAP SANエコノミーCHAPの例

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

NVMe/TCPの例

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

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

nameTemplateを使用したバックエンド構成の例

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

仮想プールを使用するバックエンドの例

これらのサンプルバックエンド定義ファイルでは、none、 `spaceAllocation`false、false `encryption`など、すべてのストレージプールに特定のデフォルトが設定されています `spaceReserve`。仮想プールは、ストレージセクションで定義します。

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

これらの例では、一部のストレージプールで独自の、、 `spaceAllocation`および `encryption`の値が設定され `spaceReserve`、一部のプールでデフォルト値が上書きされます。

ONTAP SANの例

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

ONTAP SANの経済性の例

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

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

NVMe/TCPの例

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

バックエンドを **StorageClasses** にマッピングします

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

- `protection-gold`StorageClassはバックエンドの最初の仮想プールにマッピングされます `ontap-san`。ゴールドレベルの保護を提供する唯一のプールです。

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

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

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

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

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

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

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

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

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

- `my-test-app-sc`StorageClassは、を使用してドライバの `sanType: nvme`仮想プールに `ontap-san`マッピングされます `testAPP。これは唯一のプール`testApp`です。

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

Tridentが、どの仮想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。

ONTAP NASドライバ

ONTAP NASドライバの概要

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

ドの設定について説明します。

ONTAP NASドライバの詳細

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

保護、リカバリ、モビリティにAstra Controlを使用している場合は、を参照してください。Astra Controlドライバの互換性

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

Astra Controlドライバの互換性

Astra Controlは、 `ontap-nas-flexgroup`、、および `ontap-san`ドライバ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（`Kubernetes`クラスタ間でのボリューム移動）を提供します `ontap-nas`。詳細については、を参照してください "Astra Controlレプリケーションの前提条件" 。

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

ユーザ権限

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

Amazon FSx for NetApp ONTAP環境の場合、Astra Tridentは、クラスタユーザまたは `vsadmin`SVMユーザ、または同じロールの別の名前のユーザを使用して、ONTAP管理者またはSVM管理者として実行されることが想定され `fsxadmin`ます。この `fsxadmin`ユーザは、クラスタ管理者ユーザに代わる限定的なユーザです。

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

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

ONTAP NASドライバを使用してバックエンドを設定する準備をします

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

要件

- 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、どちらか一方を参照するストレージクラスを作成できます。たとえば、ドライバを使用するGoldクラスと、ドライバを使用するBronzeクラスを `ontap-nas-economy`設定できます `ontap-nas`。
- すべてのKubernetesワーカーノードに適切なNFSツールをインストールしておく必要があります。"ここをクリック"詳細については、を参照してください。
- 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詳細については、を参照してください SMBボリュームをプロビジョニングする準備をします 。

ONTAPバックエンドの認証

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：このモードでは、ONTAPバックエンドに十分な権限が必要です。ONTAPのバージョンと最大限の互換性を確保するために、や `vsadmin`などの事前定義されたセキュリティログインロールに関連付けられたアカウントを使用することを推奨します `admin`。
- Certificate-based：Astra TridentがONTAPクラスタと通信するためには、バックエンドに証明書がインストールされている必要があります。この場合、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。や `vsadmin`などの事前定義された標準のロールを使用することを推奨しま

す `admin`。これにより、今後のリリースの 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。

YAML

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

JSON

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

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

証明書ベースの認証を有効にします

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

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

一般的なワークフローは次の手順で構成されます。

手順

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

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

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

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

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

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

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

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

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

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

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

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

7. 前の手順で得た値を使用してバックエンドを作成します。

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

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

認証方法を更新するか、クレデンシャルをローテーションして

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

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

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

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

Astra Tridentでは、ONTAPバックエンドのエクスポートポリシーを動的に管理できます。これにより、ストレージ管理者は、明示的なルールを手動で定義するのではなく、ワーカーノードの IP で許容されるアドレススペースを指定できます。エクスポートポリシーの管理が大幅に簡易化され、エクスポートポリシーを変更しても、ストレージクラスタに対する手動の操作は不要になります。さらに、この方法を使用すると、ストレージクラスタへのアクセスを指定した範囲内のIPを持つワーカーノードだけに制限できるため、きめ細かい管理が可能になります。

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

例

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

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

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

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

- `autoExportPolicy`がに設定されてい `true`ます。これは、`Astra Trident`がSVM用のエクスポートポリシーを作成し、アドレスブロックを使用してルールの追加と削除を処理する `autoExportCIDRs`ことを示します `svm1`。たとえば、UUIDが403b5326-8482-40dB-96d0-d83fb3f4daecのバックエンドをに設定する `true`と `autoExportPolicy`、という名前のエクスポートポリシーがSVMに作成されます `trident-403b5326-8482-40db-96d0-d83fb3f4daec`。

- `autoExportCIDRs`アドレスブロックのリストが含まれます。このフィールドは省略可能で、デフォルト値は ["0.0.0.0/0" 、 "::/0" です。定義されていない場合は、 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 `192.168.0.0/24`、アドレス空間が提供されています。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、実行先のノードを登録すると、ノードのIPアドレスを取得してに表示されるアドレスブロックと照合し`autoExportCIDRs`ます。IPをフィルタリングしたあと、Astra Tridentは、検出したクライアントIP用のエクスポートポリシールールを作成します。このルールには、特定したノードごとに1つのルールが含まれています。

バックエンドを作成した後で、バックエンドのおよびを `autoExportCIDRs`更新できます `autoExportPolicy`。自動的に管理されるバックエンドに新しい CIDRs を追加したり、既存の CIDRs を削除したりできます。CIDRs を削除する際は、既存の接続が切断されないように注意してください。バックエンドに対して無効にして、手動で作成したエクスポートポリシーにフォールバックすることもできます `autoExportPolicy`。この場合、バックエンド設定でパラメータを設定する必要があります `exportPolicy`。

Astra Tridentでバックエンドが作成または更新されたら、または対応する `tridentbackend`CRDを使用してバックエンドを確認でき `tridentctl`ます。

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

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（バックエンドのに指定されたアドレス範囲に該当する場合 `autoExportCIDRs`）。

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。

既存のバックエンドがある場合は、を使用してバックエンドを更新する `tridentctl update backend`と、Astra Tridentがエクスポートポリシーを自動的に管理するようになります。これにより、バックエンドのUUIDに基づいてという名前の新しいエクスポートポリシーが作成され、バックエンドにあるボリュームは再マウント時に新しく作成されたエクスポートポリシーを使用します。

> 自動管理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。

SMBボリュームをプロビジョニングする準備をします

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

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

開始する前に

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

- Linuxコントローラノードと少なくとも1つのWindowsワーカーノードでWindows Server 2022を実行しているKubernetesクラスタ。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Astra Tridentのシークレットが少なくとも1つ必要です。シークレットを生成するには `smbcreds`：

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

- Windowsサービスとして設定されたCSIプロキシ。を設定するには `csi-proxy`、Windowsで実行されているKubernetesノードについて、またはを"GitHub: Windows向けCSIプロキシ"参照してください"GitHub: CSIプロキシ"。

手順

1. オンプレミスのONTAPの場合は、必要に応じてSMB共有を作成するか、Astra TridentでSMB共有を作成できます。

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

   SMB管理共有は、共有フォルダスナップインを使用するか、ONTAP CLIを使用して作成できます"Microsoft管理コンソール"。ONTAP CLIを使用してSMB共有を作成するには、次の手順を実行します

   a. 必要に応じて、共有のディレクトリパス構造を作成します。

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

b. 指定したSVMに関連付けられているSMB共有を作成します。

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

c. 共有が作成されたことを確認します。

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

詳細については、を参照して"SMB共有を作成する"ください。

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

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

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する方法について説明します。このセクションでは、バックエンドの構成例と、バックエンドをStorageClassesにマッピングするための詳細を示します。

バックエンド構成オプション

バックエンド設定オプションについては、次の表を参照してください。

パラメータ	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-economy」、「ontap-nas-flexgroup」、「ontap-san」、「ontap-san-economy」
`backendName`	カスタム名またはストレージバックエンド	ドライバ名+"_"+ dataLIF
`managementLIF`	クラスタまたはSVM管理LIFのIPアドレス完全修飾ドメイン名（FQDN）を指定できます。IPv6フラグを使用してAstra Tridentをインストールした場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`。シームレスなMetroClusterスイッチオーバーについては、を参照して[mcc-best]ください。	「 10.0.0.1 」、「 [2001:1234:abcd::fefe] 」
`dataLIF`	プロトコル LIF の IP アドレス。指定することをお勧めします `dataLIF`。指定しない場合は、Astra TridentがSVMからデータLIFを取得します。NFSマウント処理に使用するFully Qualified Domain Name（FQDN；完全修飾ドメイン名）を指定して、ラウンドロビンDNSを作成して複数のデータLIF間で負荷を分散することができます。初期設定後に変更できます。を参照してください 。IPv6フラグを使用してAstra Tridentをインストールした場合は、IPv6アドレスを使用するように設定できます。IPv6アドレスは、のように角かっこで定義する必要があります `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`。* MetroClusterの場合は省略してください。*を参照してください[mcc-best]。	指定されたアドレス、または指定されていない場合はSVMから取得されるアドレス（非推奨）
`svm`	使用するStorage Virtual Machine * MetroClusterでは省略*を参照してください[mcc-best]。	SVMが指定されている場合に派生 `managementLIF`
`autoExportPolicy`	エクスポートポリシーの自動作成と更新を有効にします[ブーリアン]。オプションと `autoExportCIDRs`オプションを使用する `autoExportPolicy`と、Astra Tridentでエクスポートポリシーを自動的に管理できます。	正しくない
`autoExportCIDRs`	が有効な場合にKubernetesのノードIPをフィルタリングするCIDRのリスト `autoExportPolicy`。オプションと `autoExportCIDRs`オプションを使用する `autoExportPolicy`と、Astra Tridentでエクスポートポリシーを自動的に管理できます。	["0.0.0.0/0","::/0"]'
`labels`	ボリュームに適用する任意の JSON 形式のラベルのセット	""

パラメータ	説明	デフォルト
clientCertificate	クライアント証明書の Base64 エンコード値。証明書ベースの認証に使用されます	""
clientPrivateKey	クライアント秘密鍵の Base64 エンコード値。証明書ベースの認証に使用されます	""
trustedCACertificate	信頼された CA 証明書の Base64 エンコード値。オプション。証明書ベースの認証に使用されます	""
username	クラスタ / SVM に接続するためのユーザ名。クレデンシャルベースの認証に使用されます	
password	クラスタ / SVM に接続するためのパスワード。クレデンシャルベースの認証に使用されます	
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	"トライデント"
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。また、qtreeおよびLUNに対して管理するボリュームの最大サイズを制限し、オプションを使用すると、`qtreesPerFlexvol`FlexVolあたりのqtreeの最大数をカスタマイズできます。	"" （デフォルトでは適用されません）
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} トラブルシューティングを行って詳細なログダンプが必要な場合を除き、は使用しない `debugTraceFlags`でください。	null
nasType	NFSボリュームまたはSMBボリュームの作成を設定オプションは nfs、 `smb`またはnullです。nullに設定すると、デフォルトでNFSボリュームが使用されます。	nfs
nfsMountOptions	NFSマウントオプションをカンマで区切ったリスト。Kubernetes永続ボリュームのマウントオプションは通常は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。ストレージクラスや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。	""
qtreesPerFlexvol	FlexVol あたりの最大 qtree 数。有効な範囲は [50 、300] です。	"200"

パラメータ	説明	デフォルト
smbShare	Microsoft管理コンソールまたはONTAP CLIを使用して作成されたSMB共有の名前、Astra TridentでSMB共有を作成できる名前、ボリュームへの共有アクセスを禁止する場合はパラメータを空白のままにすることができます。オンプレミスのONTAPでは、このパラメータはオプションです。このパラメータはAmazon FSx for ONTAPバックエンドで必須であり、空にすることはできません。	smb-share
useREST	ONTAP REST API を使用するためのブーリアンパラメータ。 useREST に設定する true`と、Astra TridentはONTAP REST APIを使用してバックエンドと通信します。に設定する `false`と、Astra TridentはONTAP ZAPI呼び出しを使用してバックエンドと通信します。この機能にはONTAP 9.11.1以降が必要です。また、使用するONTAPログインロールには、アプリケーションへのアクセス権が必要です `ontap 。これは、事前に定義された役割と役割によって実現され vsadmin cluster-admin ます。Astra Trident 24.06リリースおよびONTAP 9.15.1以降では、が useREST デフォルトでに設定されてい true ます。ONTAP ZAPI呼び出しを使用するようににに変更してください。 useREST false	true ONTAP 9.15.1以降の場合は、それ以外の場合は false。
limitVolumePoolSize	ONTAP NASエコノミーバックエンドでqtreeを使用する場合の、要求可能なFlexVolの最大サイズ。	""（デフォルトでは適用されません）

ボリュームのプロビジョニング用のバックエンド構成オプション

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

パラメータ	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"
spaceReserve	スペースリザベーションモード：「none」（シン）または「volume」（シック）	"なし"
snapshotPolicy	使用する Snapshot ポリシー	"なし"
qosPolicy	作成したボリュームに割り当てる QoS ポリシーグループ。ストレージプール / バックエンドごとに QOSPolicy または adaptiveQosPolicy のいずれかを選択します	""
adaptiveQosPolicy	アダプティブ QoS ポリシーグループ：作成したボリュームに割り当てます。ストレージプール / バックエンドごとに QOSPolicy または adaptiveQosPolicy のいずれかを選択します。経済性に影響する ONTAP - NAS ではサポートされません。	""

パラメータ	説明	デフォルト
`snapshotReserve`	Snapshot 用にリザーブされているボリュームの割合	が「none」の場合は「0」`snapshotPolicy`、それ以外の場合は「」
`splitOnClone`	作成時にクローンを親からスプリットします	いいえ
`encryption`	新しいボリュームでNetApp Volume Encryption（NVE）を有効にします。デフォルトはです。 `false`このオプションを使用するには、クラスタで NVE のライセンスが設定され、有効になっている必要があります。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に有効になります。詳細については、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
`tieringPolicy`	「none」を使用する階層化ポリシー	ONTAP 9.5より前のSVM-DR設定の場合は「snapshot-only」
`unixPermissions`	新しいボリュームのモード	NFSボリュームの場合は「777」、SMBボリュームの場合は空（該当なし）
`snapshotDir`	ディレクトリへのアクセスを管理します。`.snapshot`	いいえ
`exportPolicy`	使用するエクスポートポリシー	デフォルト
`securityStyle`	新しいボリュームのセキュリティ形式。NFSのサポート `mixed`と `unix`セキュリティ形式SMBのサポート `mixed`と `ntfs`セキュリティ形式。	NFSのデフォルトはです `unix`。SMBのデフォルトはです `ntfs`。
`nameTemplate`	カスタムボリューム名を作成するためのテンプレート。	""

Trident 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、各コンスティチ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。

ボリュームプロビジョニングの例

デフォルトが定義されている例を次に示します。

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

と `ontap-nas-flexgroups`については、 `ontap-nas`Trident新しい計算式を使用して、FlexVol がsnapshotReserveの割合とPVCで正しくサイジングされるようになりました。ユーザが PVC を要求すると、 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、ユーザが要求したのはボリューム全体であり、その割合であるため `snapshotReserve`です。Trident 21.07では、ユーザが要求するのは書き込み可能なスペースで、Astra Tridentではボリューム全体に対する割合が定義され `snapshotReserve`ます。これはには適用されませ `ontap-nas-economy`ん。この機能の仕組みについては、次の例を参照してください。

計算は次のとおりです。

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

snapshotReserve = 50% 、 PVC 要求 = 5GiB の場合、ボリュームの合計サイズは 2/0.5 = 10GiB 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。

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、以前のと2GiBのPVCで `snapshotReserve=50` は、1GiBの書き込み可能なスペースを提供するボリュームが作成されました。たとえば、ボリュームのサイズを 3GiB に変更すると、アプリケーションの書き込み可能なスペースが 6GiB のボリュームで 3GiB になります。

最小限の設定例

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

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

ONTAP NASの経済性の例

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

ONTAP NAS FlexGroupの例

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

MetroClusterの例

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

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

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

SMBボリュームの例

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

証明書ベースの認証の例

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

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

自動エクスポートポリシーの例

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

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

IPv6アドレスの例

次に、IPv6アドレスの使用例を示し `managementLIF`ます。

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

SMBボリュームを使用した**Amazon FSx for ONTAP**の例

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

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

nameTemplateを使用したバックエンド構成の例

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

仮想プールを使用するバックエンドの例

以下に示すサンプルのバックエンド定義ファイルでは、すべてのストレージプールに特定のデフォルトが設定されています（at none、at `spaceAllocation` false、at false `encryption`など） `spaceReserve`。仮想プールは、ストレージセクションで定義します。

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

これらの例では、一部のストレージプールで独自の、、 `spaceAllocation`および `encryption`の値が設定され `spaceReserve`、一部のプールでデフォルト値が上書きされます。

ONTAP NASの例

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

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

ONTAP NAS FlexGroupの例

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

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

ONTAP NASの経済性の例

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

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

バックエンドを **StorageClasses** にマッピングします

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

- `protection-gold`StorageClassは、バックエンドの最初と2番目の仮想プールにマッピングされます `ontap-nas-flexgroup`。ゴールドレベルの保護を提供する唯一のプールです。

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

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

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

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

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

- `protection-silver-creditpoints-20k`StorageClassはバックエンドの3番目の仮想プールにマッピングされます `ontap-nas-flexgroup`。シルバーレベルの保護と20000クレジットポイントを提供する唯一のプールです。

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

- `creditpoints-5k`StorageClassは、バックエンドの3番目の仮想プールとバックエンドの2番目の仮想プール `ontap-nas-economy`にマッピングされます `ontap-nas`。これらは、5000クレジットポイントを持つ唯一のプールオファリングです。

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

Tridentが、どの仮想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。

初期設定後に更新 `dataLIF`

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

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

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

NetApp ONTAP 対応の Amazon FSX

Amazon FSX for NetApp ONTAP で 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基盤とするファイルシステムを起動して実行できる、フルマネージドのAWSサービスです。FSX for ONTAP を使用すると、使い慣れたネットアップの機能、パフォーマンス、管理機能を活用しながら、AWSにデータを格納するためのシンプルさ、即応性、セキュリティ、拡張性を活用できます。FSX for ONTAP は、ONTAP ファイルシステムの機能と管理APIをサポートしています。

Amazon Elastic Kubernetes Service（EKS）で実行されているKubernetesクラスタが、ONTAP によってサポートされるブロックおよびファイルの永続ボリュームをプロビジョニングできるように、Amazon ONTAP ファイルシステム用のAmazon FSXをAstra Tridentに統合することができます。

ファイルシステムは、オンプレミスの ONTAP クラスタに似た、 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。

要件

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

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

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

考慮事項

- SMBボリューム：

- SMBボリュームはドライバのみを使用してサポートされ `ontap-nas`ます。
- SMBボリュームはAstra Trident EKSアドオンではサポートされません。
- 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詳細については、を参照してください "SMBボリュームをプロビジョニングする準備をします"。

- Astra Trident 24.02より前のバージョンでは、自動バックアップが有効になっているAmazon FSxファイルシステム上に作成されたボリュームはTridentで削除できませんでした。Astra Trident 24.02以降でこの問題を回避するには、AWS FSx for ONTAPのバックエンド構成ファイルで、AWS `apiRegion`、AWS、およびAWS `apikey`を `secretKey`指定し `fsxFilesystemID`ます。

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

認証

Astra Tridentは、2種類の認証モードを提供します。

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

> Astra Tridentは、SVMユーザ、または別の名前で同じロールのユーザとして実行する必要があります `vsadmin`。Amazon FSx for NetApp ONTAPには、ONTAPクラスタユーザに代わる限定的なユーザが `admin`い `fsxadmin`ます。`Astra Trident`での使用を強く推奨します `vsadmin`。

- 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。

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

- "ONTAP NAS認証"
- "ONTAP SAN認証"

詳細情報

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

IAMロールとAWS Secretを作成する

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

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

AWS Secret Managerシークレットの作成

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

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

IAMポリシーの作成

次の例は、AWS CLIを使用してIAMポリシーを作成します。

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

ポリシー**JSON**ファイル：

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

サービスアカウントの作成とIAMロール

次の例では、EKSでサービスアカウント用のIAMロールを作成します。

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

Astra Trident をインストール

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

Astra Tridentは次のいずれかの方法でインストールできます。

- Helm
- EKSアドオン

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

Helmを使用して**Astra Trident**をインストール

1. Astra Tridentインストーラパッケージをダウンロード

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

   ```
   wget https://github.com/NetApp/trident/releases/download/v24.06.0/trident-
   installer-24.06.0.tar.gz
   tar -xf trident-installer-24.06.0.tar.gz
   cd trident-installer
   ```

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

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

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

   ```
   helm install trident trident-operator-100.2406.0.tgz --set
   cloudProvider=$CP --set cloudIdentity=$CI --namespace trident
   ```

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

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                           APP VERSION

trident-operator    trident     1           2024-10-14 14:31:22.463122
+0300 IDT    deployed     trident-operator-100.2406.1    24.06.1
```

EKSアドオンを使用してAstra Tridentをインストール

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

前提条件

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

- アドオンサブスクリプションがあるAmazon EKSクラスタアカウント
- AWS MarketplaceへのAWS権限：
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMIタイプ：Amazon Linux 2（AL2_x86_64）またはAmazon Linux 2 ARM（AL2_Linux_64 ARM）
- ノードタイプ：AMDまたはARM
- 既存のAmazon FSx for NetApp ONTAPファイルシステム

AWS向けのAstra Tridentアドオンを有効にする

EKSクラスタ

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

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v24.6.1-eksbuild
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v24.6.1-eksbuild.1 （専用バージョンを使用）
```

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

管理コンソール

1. でAmazon EKSコンソールを開きます https://console.aws.amazon.com/eks/home#/clusters。
2. 左側のナビゲーションペインで、*[クラスタ]*をクリックします。
3. NetApp Trident CSIアドオンを設定するクラスタの名前をクリックします。
4. をクリックし、[その他のアドオンの入手]*をクリックします。
5. [* S * elect add-ons *]ページで、次の手順を実行します。
   a. [AWS Marketplace EKS-addons]セクションで、* Astra Trident by NetApp *チェックボックスを選択します。
   b. 「 * 次へ * 」をクリックします。
6. [Configure selected add-ons* settings]ページで、次の手順を実行します。
   a. 使用する*バージョン*を選択します。
   b. では、[Not set]*のままにします。
   c. *オプションの構成設定*を展開し、*アドオン構成スキーマ*に従って、*構成値*セクションのconfigurationValuesパラメーターを前の手順で作成したrole-arnに設定します（値は次の形式にする必要があります `eks.amazonaws.com/role-arn: arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole`）。[Conflict resolution method]で[Override]を選択すると、既存のアドオンの1つ以上の設定をAmazon EKSアドオン設定で上書きできます。このオプションを有効にしない場合、既存の設定と競合すると、操作は失敗します。表示されたエラーメッセージを使用して、競合のトラブルシューティングを行うことができます。このオプションを選択する前に、Amazon EKSアドオンが自己管理に必要な設定を管理していないことを確認してください。

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

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

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

AWS CLI

1. ファイルを作成し `add-on.json` ます。

```
add-on.json
{

   "clusterName": "<eks-cluster>",
   "addonName": "netapp_trident-operator",
   "addonVersion": "v24.6.1-eksbuild.1",
   "serviceAccountRoleArn": "arn:aws:iam::123456:role/astratrident-
role",
   "configurationValues": "{"cloudIdentity":
"'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/astratrident-
role'",
   "cloudProvider": "AWS"}"
}
```

オプションのパラメータを設定する場合 `cloudIdentity`は `cloudProvider`、EKSアドオンを使用したTridentのインストール時にとしてを指定して`AWS`ください。

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

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

Astra Trident EKSアドオンの更新

EKSクラスタ

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

  出力例：

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v24.6.1-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- 前の手順の出力でupdate availableで返されたバージョンにアドオンを更新します。
  ```
  eksctl update addon --name netapp_trident-operator --version v24.6.1-
  eksbuild.1 --cluster my-cluster --force
  ```

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

管理コンソール

1. Amazon EKSコンソールを開き https://console.aws.amazon.com/eks/home#/clustersます。
2. 左側のナビゲーションペインで、*[クラスタ]*をクリックします。
3. NetApp Trident CSIアドオンを更新するクラスタの名前をクリックします。
4. [アドオン]タブをクリックします。
5. をクリックし、[Edit]*をクリックします。
6. [Configure Astra Trident by NetApp *]ページで、次の手順を実行します。
   a. 使用する*バージョン*を選択します。
   b. （オプション）* Optional configuration settings *を展開し、必要に応じて変更できます。
   c. [変更の保存 *] をクリックします。

AWS CLI

次の例では、EKSアドオンを更新します。

```
aws eks update-addon --cluster-name my-cluster netapp_trident-operator vpc-cni
--addon-version v24.6.1-eksbuild.1 \
--service-account-role-arn arn:aws:iam::111122223333:role/role-name
--configuration-values '{}' --resolve-conflicts --preserve
```

Astra Trident EKSアドオンのアンインストールと削除

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

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

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

EKSクラスタ

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

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

管理コンソール

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

AWS CLI

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

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

ストレージバックエンドの設定

ONTAP SANと**NAS**ドライバの統合

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

ル（ユーザ名とパスワード）を使用して作成できます。

YAML

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

JSON

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

バックエンドの作成については、次のページを参照してください。

- "ONTAP NASドライバを使用したバックエンドの設定"
- "ONTAP SANドライバを使用したバックエンドの設定"

FSx for ONTAPドライバの詳細

次のドライバを使用して、Astra TridentをAmazon FSX for NetApp ONTAP と統合できます。

- `ontap-san`：プロビジョニングされる各PVは、それぞれのAmazon FSx for NetApp ONTAPボリューム内のLUNです。ブロックストレージに推奨されます。
- `ontap-nas`：プロビジョニングされる各PVは、完全なAmazon FSx for NetApp ONTAPボリュームです。NFSとSMBで推奨されます。
- `ontap-san-economy`：プロビジョニングされた各PVは、Amazon FSx for NetApp ONTAPボリュームごとに設定可能なLUN数を持つLUNです。
- `ontap-nas-economy`：プロビジョニングされる各PVはqtreeであり、Amazon FSx for NetApp ONTAPボリュームごとにqtree数を設定できます。
- `ontap-nas-flexgroup`：プロビジョニングされる各PVは、完全なAmazon FSx for NetApp ONTAP FlexGroupボリュームです。

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

構成例

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

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

SMBホリユウムノストレエシクラスノセツテイ

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

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

バックエンドの高度な設定と例

バックエンド設定オプションについては、次の表を参照してください。

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

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

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

パラメータ	説明	例
limitAggregateUsage	* Amazon FSx for NetApp ONTAP には指定しないでください。*指定されたと vsadmin`には `fsxadmin、アグリゲートの使用量を取得し、Astra Tridentを使用してアグリゲートを制限するために必要な権限が含まれていません。	使用しないでください。
limitVolumeSize	要求されたボリュームサイズがこの値を超えている場合、プロビジョニングが失敗します。また、qtreeおよびLUNに対して管理するボリュームの最大サイズを制限し、オプションを使用すると、`qtreesPerFlexvol`FlexVolあたりのqtreeの最大数をカスタマイズできます。	“”（デフォルトでは適用されません）
lunsPerFlexvol	FlexVol あたりの最大LUN数。有効な範囲は50、200です。SANのみ。	“100”
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以外は使用しない `debugTraceFlags`でください。	null
nfsMountOptions	NFSマウントオプションをカンマで区切ったリスト。Kubernetes永続ボリュームのマウントオプションは通常は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。ストレージクラスや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。	""
nasType	NFSボリュームまたはSMBボリュームの作成を設定オプションは nfs、、 smb`またはnullです。*SMBボリュームの場合はに設定する必要があります `smb。*nullに設定すると、デフォルトでNFSボリュームが使用されます。	nfs
qtreesPerFlexvol	FlexVol あたりの最大 qtree 数。有効な範囲は [50 、 300] です。	"200"

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

ボリュームのプロビジョニング用のバックエンド構成オプション

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

パラメータ	説明	デフォルト
`spaceAllocation`	space-allocation for LUN のコマンドを指定します	`true`
`spaceReserve`	スペースリザベーションモード：「 none 」（シン）または「 volume 」（シック）	`none`
`snapshotPolicy`	使用する Snapshot ポリシー	`none`
`qosPolicy`	作成したボリュームに割り当てる QoS ポリシーグループ。ストレージプールまたはバックエンドごとに、QOSPolicyまたはadaptiveQosPolicyのいずれかを選択します。Trident 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。非共有のQoSポリシーグループを使用して、各コンスティチ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。	「」
`adaptiveQosPolicy`	アダプティブ QoS ポリシーグループ：作成したボリュームに割り当てます。ストレージプールまたはバックエンドごとに、QOSPolicyまたはadaptiveQosPolicyのいずれかを選択します。経済性に影響する ONTAP - NAS ではサポートされません。	「」
`snapshotReserve`	スナップショット "0" 用に予約されたボリュームの割合	がの `none`場合 `snapshotPolicy else`
`splitOnClone`	作成時にクローンを親からスプリットします	`false`
`encryption`	新しいボリュームでNetApp Volume Encryption（NVE）を有効にします。デフォルトはです。`false`このオプションを使用するには、クラスタで NVE のライセンスが設定され、有効になっている必要があります。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、を参照してください"Astra TridentとNVEおよびNAEの相互運用性"。	`false`

パラメータ	説明	デフォルト
`luksEncryption`	LUKS暗号化を有効にします。を参照してください "Linux Unified Key Setup（LUKS；統合キーセットアップ）を使用"。SANのみ。	""
`tieringPolicy`	使用する階層化ポリシー `none`	`snapshot-only`ONTAP 9 .5より前のSVM-DR構成
`unixPermissions`	新しいボリュームのモード。* SMBボリュームは空にしておきます。*	「」
`securityStyle`	新しいボリュームのセキュリティ形式。NFSのサポート `mixed`と`unix`セキュリティ形式SMBのサポート `mixed`と `ntfs`セキュリティ形式。	NFSのデフォルトはです `unix`。SMBのデフォルトはです `ntfs`。

SMBボリュームをプロビジョニングする準備をします

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

開始する前に

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

- LinuxコントローラノードとWindowsワーカーノードでWindows Server 2019を実行しているKubernetesクラスタ。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Astra Tridentのシークレットが少なくとも1つ必要です。シークレットを生成するには `smbcreds`：

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

- Windowsサービスとして設定されたCSIプロキシ。を設定するには `csi-proxy`、Windowsで実行されているKubernetesノードについて、またはを"GitHub: Windows向けCSIプロキシ"参照してください"GitHub: CSIプロキシ"。

手順

1. SMB共有を作成SMB管理共有は、共有フォルダスナップインを使用するか、ONTAP CLIを使用して作成できます"Microsoft管理コンソール"。ONTAP CLIを使用してSMB共有を作成するには、次の手順を実行します
   a. 必要に応じて、共有のディレクトリパス構造を作成します。

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

   b. 指定したSVMに関連付けられているSMB共有を作成します。

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

c. 共有が作成されたことを確認します。

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

詳細については、を参照して"SMB共有を作成する"ください。

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

パラメータ	説明	例
`smbShare`	次のいずれかを指定できます。Microsoft管理コンソールまたはONTAP CLIを使用して作成されたSMB共有の名前、またはAstra TridentでSMB共有を作成できるようにする名前。このパラメータは、Amazon FSx for ONTAPバックエンドに必要です。	`smb-share`
`nasType`	*に設定する必要があります `smb`。*nullの場合、デフォルトは `nfs`になります。	`smb`
`securityStyle`	新しいボリュームのセキュリティ形式。* SMBボリュームの場合は `ntfs`または `mixed`に設定する必要があります。*	`ntfs`SMBボリュームの場合はまたは `mixed`
`unixPermissions`	新しいボリュームのモード。* SMBボリュームは空にしておく必要があります。*	""

ストレージクラスとPVCを設定する

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

ストレージクラスを作成する。

Kubernetes StorageClassオブジェクトの設定

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

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

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

ストレージクラスを作成する。

手順

1. これはKubernetesオブジェクトなので、を使用して `kubectl`Kubernetesで作成します。

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

2. Kubernetes と Astra Trident の両方で、 * basic-csi * ストレージクラスが表示され、 Astra Trident がバックエンドのプールを検出しました。

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

PVおよびPVCの作成

"永続ボリューム_"（PV）は、Kubernetesクラスタ上のクラスタ管理者によってプロビジョニングされる物理ストレージリソースです。 "PersistentVolumeClaim_"（PVC）は、クラスタ上のPersistentVolumeへのアクセス要求です。

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

PVとPVCを作成したら、ポッドにボリュームをマウントできます。

マニフェストの例

PersistentVolumeサンプルマニフェスト

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

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

PersistentVolumeClaimサンプルマニフェスト

次に、基本的なPVC設定オプションの例を示します。

RWOアクセスを備えたPVC

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

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

NVMe / TCP対応PVC

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

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

PVおよびPVCの作成

手順

1. PVを作成

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

2. PVステータスを確認します。

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

3. PVCを作成

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

4. PVCステータスを確認します。

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

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

Astra Tridentの属性

これらのパラメータによって、特定のタイプのボリュームのプロビジョニングに使用するAstra Tridentで管理されるストレージプールが決まります。

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

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

^1 ^ ： ONTAP Select システムではサポートされていません

サンプルアプリケーションのデプロイ

サンプルアプリケーションをデプロイします。

手順

1. ボリュームをポッドにマウントします。

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

次に、PVCをポッドに接続するための基本的な設定例を示します。基本設定：

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

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

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

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

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

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

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

EKSクラスタでのAstra Trident EKSアドオンの設定

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

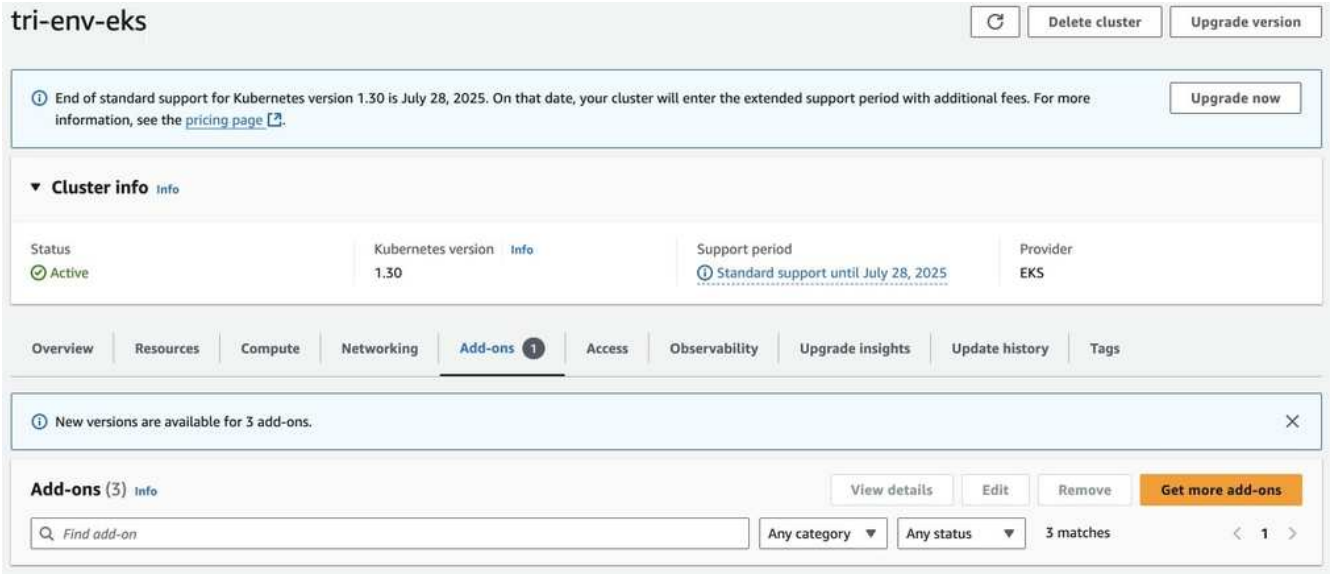
前提条件

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

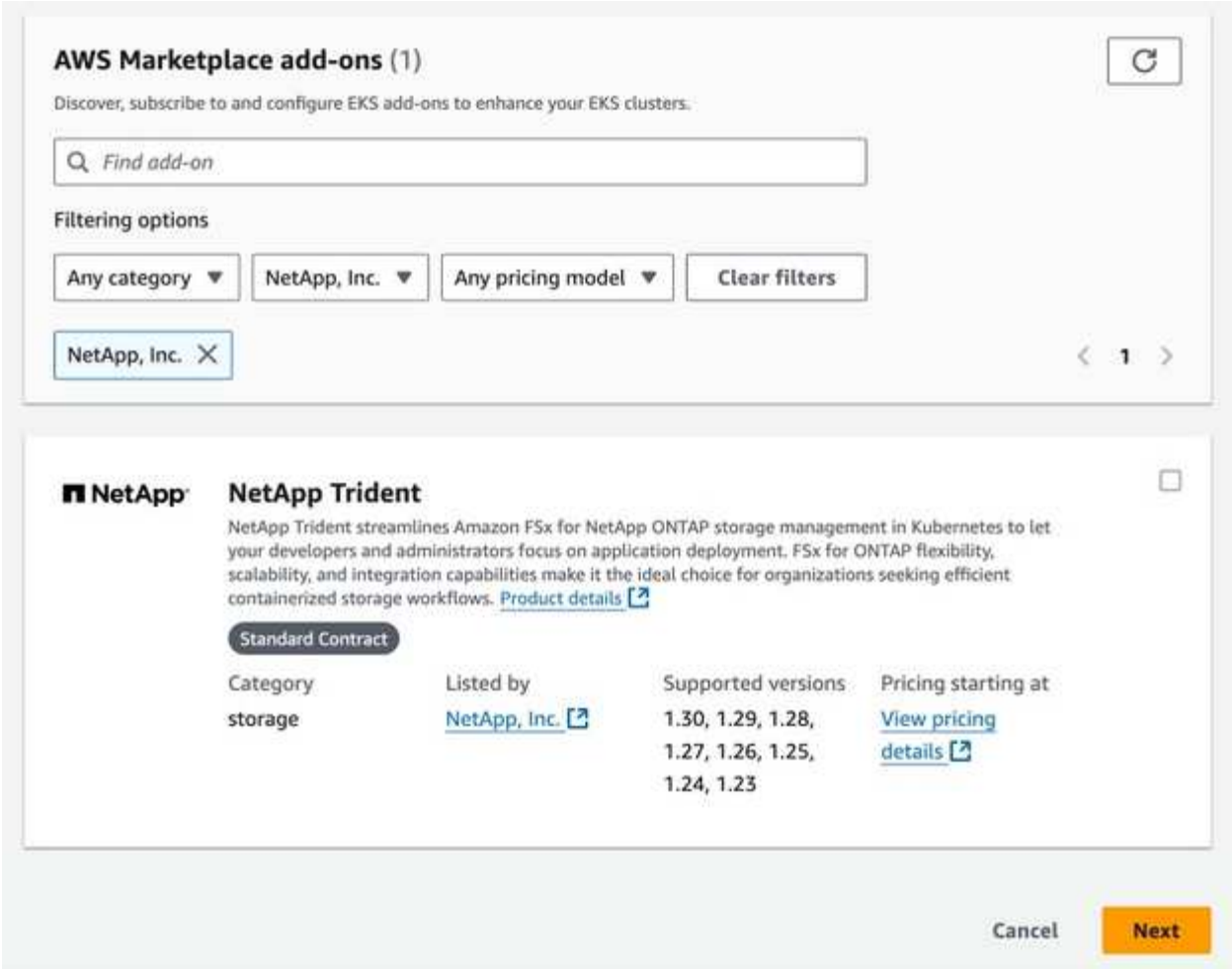
- アドオンサブスクリプションがあるAmazon EKSクラスタアカウント
- AWS MarketplaceへのAWS権限：
  ```
  "aws-marketplace:ViewSubscriptions",
  "aws-marketplace:Subscribe",
  "aws-marketplace:Unsubscribe
  ```
- AMIタイプ：Amazon Linux 2（AL2_x86_64）またはAmazon Linux 2 ARM（AL2_Linux_64 ARM）
- ノードタイプ：AMDまたはARM
- 既存のAmazon FSx for NetApp ONTAPファイルシステム

手順

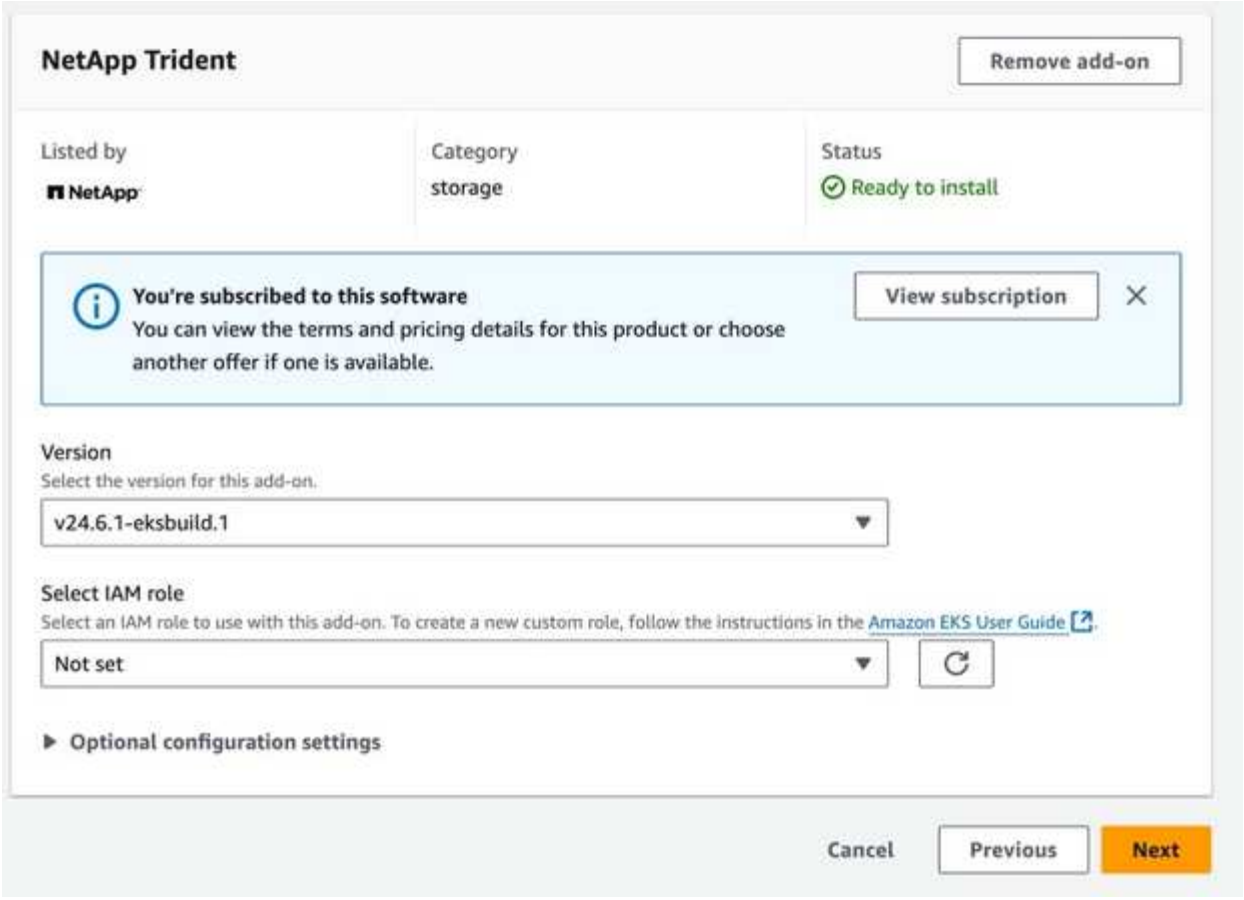
1. EKS Kubernetesクラスタで、*アドオン*タブに移動します。



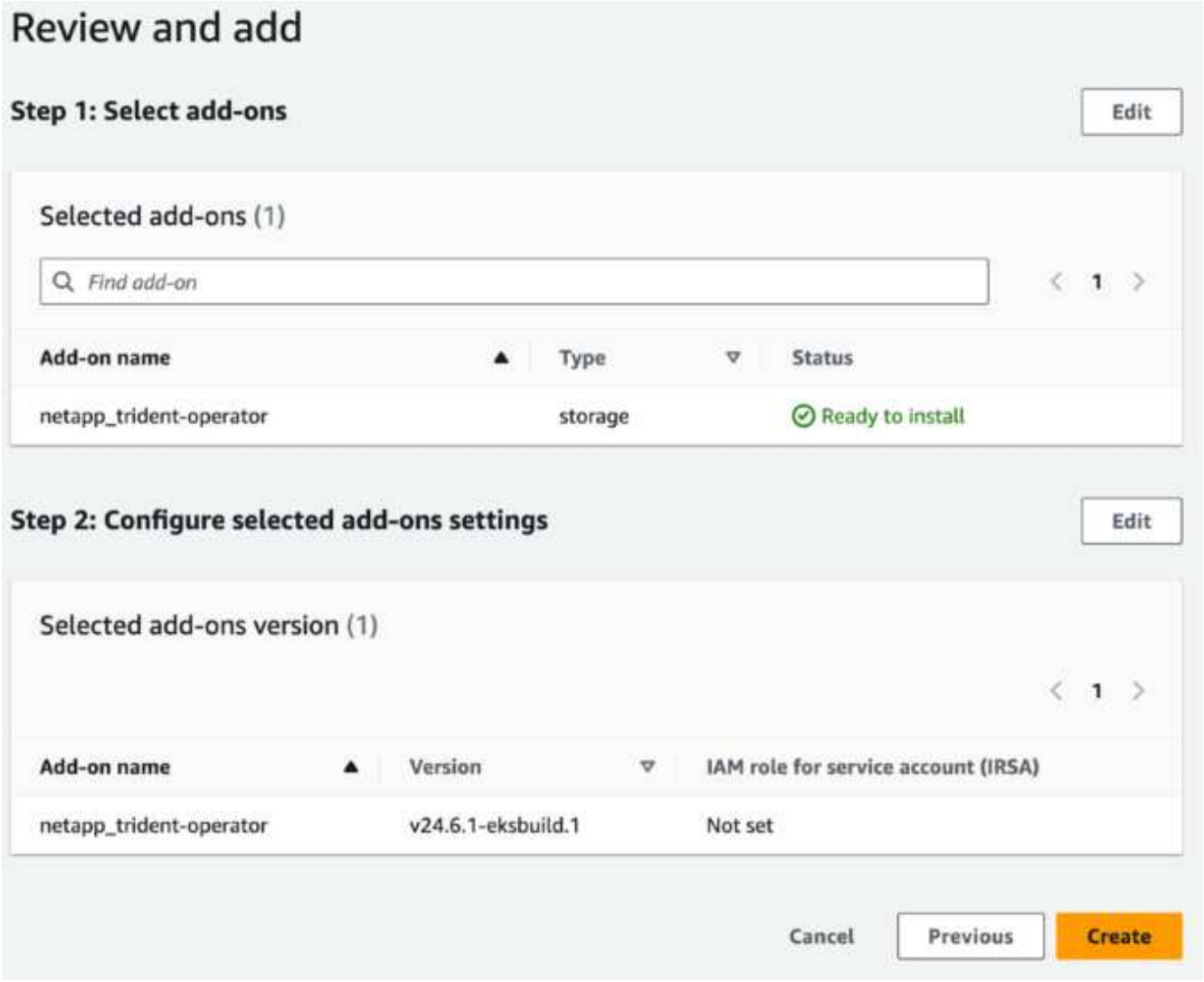
2. [AWS Marketplace add-ons]*にアクセスし、_storage_categoryを選択します。



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



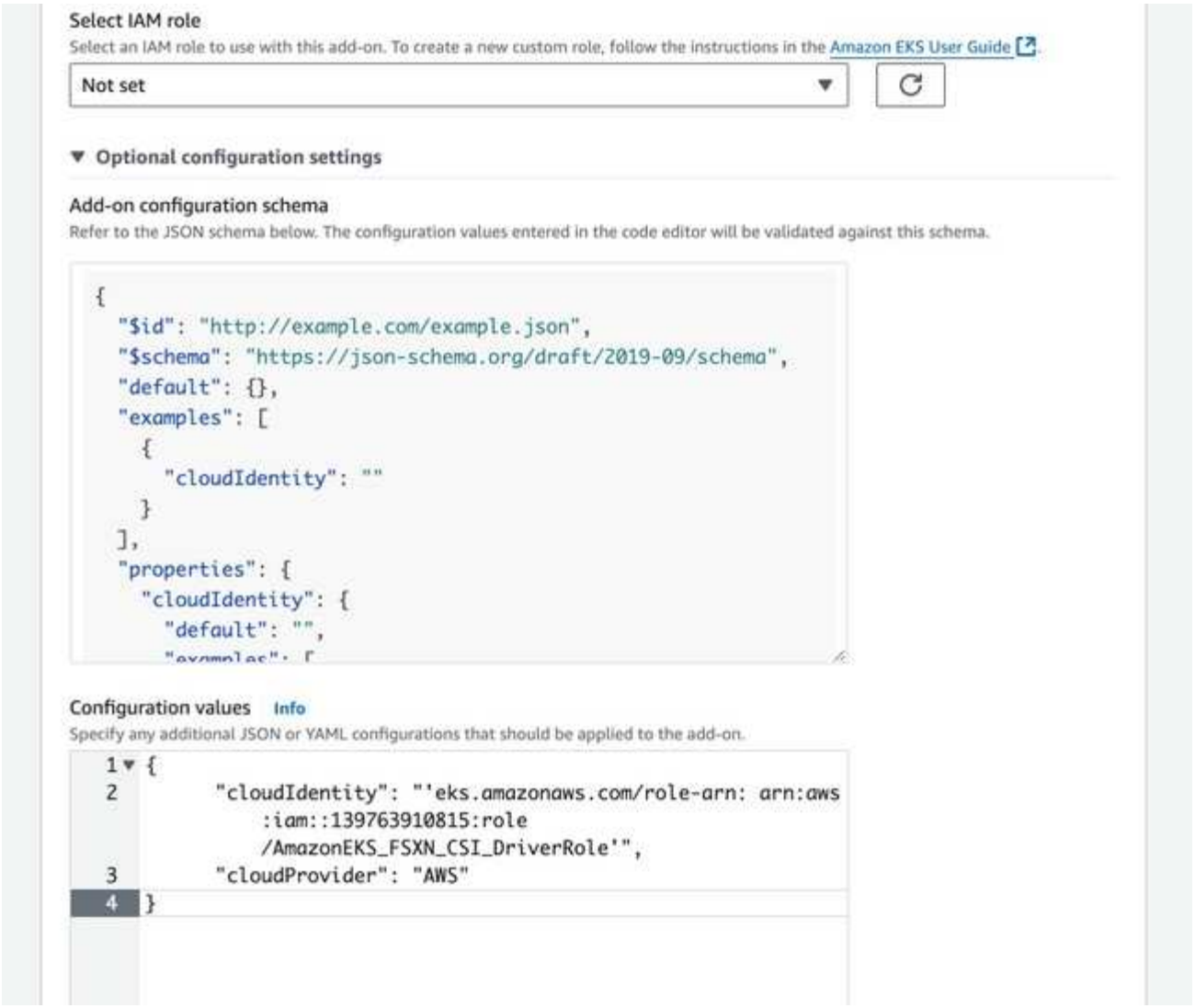
5. ノードから継承するIAMロールオプションを選択します。



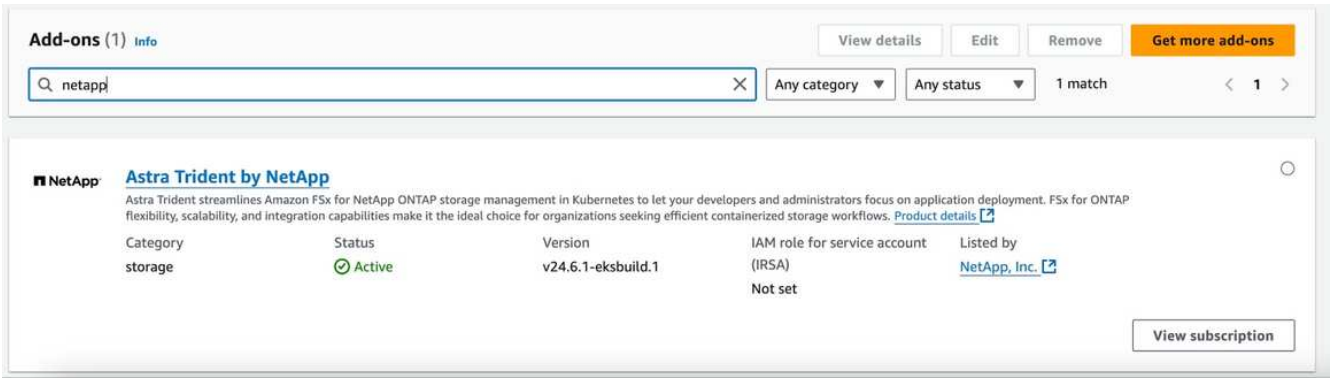
6. （オプション）必要に応じてオプションの設定を行い、* Next *を選択します。

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

オプションのパラメータを設定する場合 `cloudIdentity`は `cloudProvider`、EKSアドオンを使用したTridentのインストール時にとしてを指定して `AWS`ください。



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



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

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

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

```
eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version
```

```
v24.6.1-eksbuild
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v24.6.1-eksbuild.1
```

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

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

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

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

kubectl を使用してバックエンドを作成します

バックエンドは、 Astra Trident とストレージシステムの関係を定義します。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、Kubernetesインターフェイスから直接Tridentバックエンドを作成および管理できます。これは、またはKubernetesディストリビューション用の同等のCLIツールを使用して実行できます `kubectl`。

TridentBackendConfig

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

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

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

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

`TridentBackend`CRSは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には、オブジェクトを変更し `TridentBackendConfig`ます。

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

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

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

は spec、バックエンド固有の設定パラメータを取得します。この例では、バックエンドでストレージドライバを使用し ontap-san、次の表に示す設定パラメータを使用しています。ご使用のストレージドライバの設定オプションのリストについては、を参照してください"ストレージドライバのバックエンド設定情報"。

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

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

> バックエンドの名前はを使用して設定され `spec.backendName`ます。指定しない場合、バックエンドの名前はオブジェクトの名前（metadata.name）に設定され `TridentBackendConfig`ます。を使用してバックエンド名を明示的に設定することをお勧めし `spec.backendName`ます。

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

手順の概要

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

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

バックエンドを作成したら、を使用してそのステータスを確認し、追加の詳細情報を収集できます `kubectl get tbc <tbc-name> -n <trident-namespace>`。

手順 1 ： Kubernetes Secret を作成します

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

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

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

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

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

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

ステップ2：CRを作成する `TridentBackendConfig`

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

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

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

手順3：CRのステータスを確認する TridentBackendConfig

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

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

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

フェーズには次のいずれかの値を指定できます。

- Bound: TridentBackendConfig CRはバックエンドに関連付けられており、そのバックエンドにはCRのuidがセットされ `TridentBackendConfig`てい `configRef`ます。
- Unbound:を使用して表されます ""。 `TridentBackendConfig`オブジェクトはバックエンドにバインドされていません。デフォルトでは、新しく作成されたすべての `TridentBackendConfig`CRSがこのフェーズになります。フェーズが変更された後、再度 Unbound に戻すことはできません。
- Deleting：CR deletionPolicy`は `TridentBackendConfig`削除するように設定されています。CRが削除されると `TridentBackendConfig`、CRは削除ステートに移行します。
  - バックエンドに永続的ボリューム要求（PVC）が存在しない場合は、を削除する `TridentBackendConfig`とAstra TridentでバックエンドとCRが削除され `TridentBackendConfig`ます。
  - バックエンドに 1 つ以上の PVC が存在する場合は、削除状態になります。 TridentBackendConfig`その後、CRは削除フェーズに入ります。バックエンドとは `TridentBackendConfig`、すべてのPVCが削除された後にのみ削除されます。
- Lost：CRに関連付けられているバックエンドが TridentBackendConfig`誤ってまたは故意に削除され、 `TridentBackendConfig`CRには削除されたバックエンドへの参照が残っています。 `TridentBackendConfig`CRは、値に関係なく削除できます `deletionPolicy。

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

この段階では、バックエンドが正常に作成されます。など、追加で処理できる処理がいくつかあります"バックエンドの更新とバックエンドの削除"。

（オプション）手順 4 ：詳細を確認します

バックエンドに関する詳細情報を確認するには、次のコマンドを実行します。

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

```
NAME                      BACKEND NAME         BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   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`CRに応答して作成されたバックエンドの `TridentBackendConfig`とが `backendUUID`格納され `backendName`ます。 `lastOperationStatus`フィールドはCRの最後の処理のステータスを表し `TridentBackendConfig`ます。ユーザトリガー（でユーザが変更した場合など）、またはAstra Tridentによってトリガーされた（Astra Tridentの再起動時など）ことができます `spec`。成功または失敗のいずれかです。 `phase`CRとバックエンド間の関係のステータスを表します `TridentBackendConfig`。上の例では、の `phase`値がバインドされています。つまり、CRがバックエンドに関連付けられていることを意味します `TridentBackendConfig`。

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

> を使用して、関連付けられたオブジェクトを `tridentctl`含むバックエンドを更新または削除することはできません `TridentBackendConfig`。とを `TridentBackendConfig`切り替える手順について説明します `tridentctl` "こちらを参照してください"。

バックエンドの管理

kubectl を使用してバックエンド管理を実行します

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

バックエンドを削除します

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

次のコマンドを実行します。

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

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

既存のバックエンドを表示します

次のコマンドを実行します。

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

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

バックエンドを更新します

バックエンドを更新する理由はいくつかあります。

- ストレージシステムのクレデンシャルが変更されている。クレデンシャルを更新するには、オブジェクトで使用されるKubernetes Secretを `TridentBackendConfig`更新する必要があります。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。
  - 次のコマンドを使用して、Kubernetesから直接オブジェクトを更新できます

TridentBackendConfig。

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

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

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

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

tridentctl を使用してバックエンド管理を実行します

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

バックエンドを作成します

を作成したら"バックエンド構成ファイル"、次のコマンドを実行します。

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

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

```
tridentctl logs -n trident
```

構成ファイルの問題を特定して修正したら、コマンドをもう一度実行できます create。

バックエンドを削除します

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
```

構成ファイルの問題を特定して修正したら、コマンドをもう一度実行できます `update`。

バックエンドを使用するストレージクラスを特定します

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

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

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

バックエンド管理オプション間を移動します

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。

バックエンドを管理するためのオプション

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

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

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

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

これは、次のシナリオに該当します。

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

どちらの場合も、 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。

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

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

1. Kubernetes Secret を作成します。シークレットには、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. オブジェクトを作成し `TridentBackendConfig`ます。ストレージクラスタ / サービスの詳細を指定し、前の手順で作成したシークレットを参照します。同一の設定パラメータ（、、 `spec.storagePrefix` `spec.storageDriverName`など）を指定するように注意する必要があります `spec.backendName`。 `spec.backendName`既存のバックエンドの名前に設定する必要があります。

手順 **0** ：バックエンドを特定します

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

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

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

cat ontap-nas-backend.json

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

    "defaults": {
        "spaceReserve": "none",
        "encryption": "false"
    },
    "labels":{"store":"nas_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"msoffice", "cost":"100"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"app":"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**：**CR**を作成する `TridentBackendConfig`

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

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

この場合、は `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**：**CR**のステータスを確認する `TridentBackendConfig`

が作成されたら `TridentBackendConfig`、そのフェーズはにする必要があります `Bound`。また、既存のバックエンドと同じバックエンド名と UUID が反映されている必要があります。

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

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

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

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

`tridentctl`を使用して作成されたバックエンドの一覧表示に使用でき `TridentBackendConfig`ます。さらに、管理者は、を削除して、がに設定されている `retain`ことを確認する `spec.deletionPolicy`ことで、 `TridentBackendConfig`このようなバックエンドを完全に管理することもできます `tridentctl`。

手順 **0** ：バックエンドを特定します

たとえば、次のバックエンドがを使用して作成されたとし `TridentBackendConfig`ます。

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

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

出力からは、が正常に作成され、バックエンドにバインドされていることがわかり `TridentBackendConfig`ます（[Observe the backend's UUUID]）。

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

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

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

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

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

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

手順2：CRを削除する `TridentBackendConfig`

最後のステップはCRを削除することです `TridentBackendConfig`。がに設定されている `retain`ことを確認したら `deletionPolicy`、削除を続行できます。

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

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

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