参考文献
Trident

NetApp
January 14, 2026

目次

参考文献

Tridentポート

Tridentが通信に使用するポートの詳細については、こちらを参照してください。

Tridentポート

Tridentは次のポートを介して通信します。

ポート	目的
8443	バックチャネル HTTPS
8001	Prometheus 指標エンドポイント
8000	Trident REST サーバ
17546	Trident デミ作用 / レディネスプローブポートは、 Trident デミ作用ポッドで使用されます

Liveness/Readinessプローブポートは、フラグを使用してインストール中に変更できます `--probe-port`。このポートがワーカーノード上の別のプロセスで使用されていないことを確認することが重要です。

Trident REST API

"tridentctl コマンドとオプション"Trident REST APIを操作する最も簡単な方法ですが、必要に応じてRESTエンドポイントを直接使用することもできます。

REST APIを使用する状況

REST APIは、Kubernetes以外の環境でTridentをスタンドアロンバイナリとして使用する高度なインストールに役立ちます。

セキュリティを強化するため、ポッド内で実行する場合、Tridentは `REST API`デフォルトでlocalhostに制限されています。この動作を変更するには、ポッド構成でTridentの引数を設定する必要があり `-address`ます。

REST APIを使用する

これらのAPIの呼び出し方法の例については、debug)(`-d`フラグを渡します。詳細については、を参照してください "tridentctlを使用したTridentの管理"。

API は次のように機能します。

取得

`GET <trident-address>/trident/v1/<object-type>`

そのタイプのすべてのオブジェクトを一覧表示します。

GET <trident-address>/trident/v1/<object-type>/<object-name>

指定したオブジェクトの詳細を取得します。

投稿

POST <trident-address>/trident/v1/<object-type>

指定したタイプのオブジェクトを作成します。

- オブジェクトを作成するには JSON 構成が必要です。各オブジェクトタイプの仕様については、を参照してください"tridentctlを使用したTridentの管理"。
- オブジェクトがすでに存在する場合、動作は一定ではありません。バックエンドが既存のオブジェクトを更新しますが、それ以外のすべてのオブジェクトタイプで処理が失敗します。

削除

DELETE <trident-address>/trident/v1/<object-type>/<object-name>

指定したリソースを削除します。

バックエンドまたはストレージクラスに関連付けられているボリュームは削除されず、削除されません。詳細については、を参照してください "tridentctlを使用したTridentの管理"。

コマンドラインオプション

Tridentでは、Tridentオーケストレーションツールのコマンドラインオプションがいくつか公開されています。これらのオプションを使用して、導入環境を変更できます。

ロギング

-debug

デバッグ出力を有効にします。

-loglevel <level>

ロギングレベル（debug、info、warn、error、fatal）を設定します。デフォルトは info です。

Kubernetes

-k8s_pod

このオプションまたはを使用し `-k8s_api_server`て、Kubernetesのサポートを有効にします。これを設定すると、 Trident はポッドの Kubernetes サービスアカウントのクレデンシャルを使用して API サーバに接続します。これは、サービスアカウントが有効になっている Kubernetes クラスタで Trident がポッドとして実行されている場合にのみ機能します。

-k8s_api_server <insecure-address:insecure-port>

このオプションまたはを使用し `-k8s_pod`て、Kubernetesのサポートを有効にします。Trident を指定すると、セキュアでないアドレスとポートを使用して Kubernetes API サーバに接続されます。これにより、Tridentをポッドの外部に導入できますが、サポートされるのはAPIサーバへの安全でない接続のみです。安全に接続するには、オプションを使用してポッドにTridentを導入し `-k8s_pod`ます。

Docker

-volume_driver <name>

Dockerプラグインの登録時に使用するドライバ名。デフォルトはです `netapp`。

-driver_port <port-number>

UNIXドメインソケットではなく、このポートでリッスンします。

-config <file>

必須。バックエンド構成ファイルへのパスを指定する必要があります。

REST

-address <ip-or-host>

TridentのRESTサーバがリスンするアドレスを指定します。デフォルトは localhost です。localhost で聞いて Kubernetes ポッド内で実行しているときに、 REST インターフェイスにポッド外から直接アクセスすることはできません。ポッドのIPアドレスからRESTインターフェイスにアクセスできるようにするために使用し `-address ""`ます。

> Trident REST インターフェイスは、 127.0.0.1 （ IPv4 の場合）または [::1] （ IPv6 の場合）のみをリスンして処理するように設定できます。

-port <port-number>

TridentのRESTサーバがリスンするポートを指定します。デフォルトは8000です。

-rest

RESTインターフェイスを有効にします。デフォルトは true です。

Kubernetes オブジェクトと Trident オブジェクト

リソースオブジェクトの読み取りと書き込みを行うことで、 REST API を使用して Kubernetes や Trident を操作できます。Kubernetes と Trident 、 Trident とストレージ、 Kubernetes とストレージの関係を決定するリソースオブジェクトがいくつかあります。これらのオブジェクトの中には Kubernetes で管理されるものと Trident で管理されるものがあります。

オブジェクトは相互にどのように相互作用しますか。

おそらく、オブジェクト、その目的、操作方法を理解する最も簡単な方法は、 Kubernetes ユーザからのストレージ要求を 1 回だけ処理することです。

1. ユーザは、管理者が以前に設定したKubernetesから、特定のサイズの `StorageClass`新しい要求を `PersistentVolume`作成します `PersistentVolumeClaim`。
2. Kubernetesは `StorageClass`Tridentをプロビジョニングツールとして識別し、要求されたクラスのボリュームのプロビジョニング方法をTridentに指示するパラメータを備えています。
3. Tridentは、同じ名前を使用して一致するものを識別し `Backends`、 `StoragePools`クラス用のボリュームのプロビジョニングに使用できるかどうかを確認し `StorageClass`ます。

4. Tridentは、対応するバックエンドにストレージをプロビジョニングし、2つのオブジェクトを作成します。1つは `PersistentVolume`Kubernetesでボリュームの検索、マウント、処理方法を指示するもので、もう1つはTridentで、もう1つはと実際のストレージの関係を保持するもの `PersistentVolume`です。
5. Kubernetesは、を新しいに `PersistentVolume`バインドし `PersistentVolumeClaim`ます。実行されるホスト上の`PersistentVolume`のマウントを含むポッド `PersistentVolumeClaim`。
6. ユーザは、Tridentをポイントするを使用して、既存のPVCの `VolumeSnapshotClass`を作成します `VolumeSnapshot`。
7. Trident が PVC に関連付けられているボリュームを特定し、バックエンドにボリュームの Snapshot を作成します。また、Snapshotを識別する方法をKubernetesに指示するを作成し `VolumeSnapshotContent`ます。
8. ユーザーは、をソースとして使用して `VolumeSnapshot`を作成できます `PersistentVolumeClaim`。
9. Tridentは必要なSnapshotを特定し、およびの `Volume`作成と同じ手順を実行します `PersistentVolume`。

Kubernetesオブジェクトの詳細については、Kubernetesドキュメントのセクションを読むことを強くお勧めします "永続ボリューム"。

`PersistentVolumeClaim`Kubernetesオブジェクト

Kubernetes `PersistentVolumeClaim`オブジェクトは、Kubernetesクラスタユーザによって行われたストレージへの要求です。

Trident では、標準仕様に加えて、バックエンド構成で設定したデフォルト設定を上書きする場合に、ボリューム固有の次のアノテーションを指定できます。

アノテーション	ボリュームオプション	サポートされているドライバ
trident.netapp.io/fileSystem	ファイルシステム	ONTAP-SAN、solidfire-san-エコノミー 構成、solidfire-san-SAN間にあるSolidFireを実現します
trident.netapp.io/cloneFromPVC	cloneSourceVolume の実行中です	ontap - NAS 、 ontap - san 、 solidfire-san-files 、 gcvs 、 ONTAP - SAN - 経済性
trident.netapp.io/splitOnClone	splitOnClone	ONTAP - NAS 、 ONTAP - SAN
trident.netapp.io/protocol	プロトコル	任意
trident.netapp.io/exportPolicy	エクスポートポリシー	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup
trident.netapp.io/snapshotPolicy	Snapshot ポリシー	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup 、 ONTAP-SAN
trident.netapp.io/snapshotReserve	Snapshot リザーブ	ONTAP-NAS 、 ONTAP-NAS-flexgroup 、 ONTAP-SAN 、 GCP-cvs

アノテーション	ボリュームオプション	サポートされているドライバ
trident.netapp.io/snapshotDirectory	snapshotDirectory の略	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup
trident.netapp.io/blockSize	ブロックサイズ	solidfire - SAN

作成されたPVに再要求ポリシーが設定されている場合 `Delete`、PVが解放されると（つまり、ユーザがPVCを削除すると）、TridentはPVと元のボリュームの両方を削除します。削除操作が失敗した場合、 Trident は PV をマークします。そのような状態で操作が成功するか、 PV が手動で削除されるまで、定期的に再試行します。PVがポリシーを使用している場合 `Retain`、Tridentはポリシーを無視し、管理者がKubernetesとバックエンドからポリシーをクリーンアップすると想定します。これにより、削除前にボリュームをバックアップまたは検査できるようになります。PV を削除しても、原因 Trident で元のボリュームが削除されないことに注意してください。REST APIを使用して削除する必要があり(`tridentctl`ます）。

Trident では CSI 仕様を使用したボリュームスナップショットの作成がサポートされています。ボリュームスナップショットを作成し、それをデータソースとして使用して既存の PVC のクローンを作成できます。これにより、 PVS のポイントインタイムコピーを Kubernetes にスナップショットの形で公開できます。作成した Snapshot を使用して新しい PVS を作成できます。これがどのように機能するかを見て `On-Demand Volume Snapshots`ください。

Tridentには、クローンを作成するためのアノテーションとが `splitOnClone`用意されています `cloneFromPVC`。これらの注釈を使用して、CSI実装を使用せずにPVCのクローンを作成できます。

次に例を示します。ユーザがすでにというPVCを持っている場合、 `mysql`ユーザはなどの注釈を使用して `trident.netapp.io/cloneFromPVC: mysql`という新しいPVCを作成できます `mysqlclone`。このアノテーションセットを使用すると、 Trident はボリュームをゼロからプロビジョニングするのではなく、MySQL PVC に対応するボリュームのクローンを作成します。

次の点を考慮してください。

- NetAppでは、アイドル状態のボリュームをクローニングすることを推奨
- PVC とそのクローンは、同じ Kubernetes ネームスペースに存在し、同じストレージクラスを持つ必要があります。
- ドライバと `ontap-san`ドライバを使用している `ontap-nas`場合は、と組み合わせて `trident.netapp.io/cloneFromPVC`PVCアノテーションを設定することをお勧めし `trident.netapp.io/splitOnClone`ます。 `trident.netapp.io/splitOnClone`をに設定する `true`と、Tridentはクローンボリュームを親ボリュームからスプリットするため、クローンボリュームのライフサイクルが親から完全に切り離されますが、ストレージ効率が低下することはありません。に設定または設定し `false`ない `trident.netapp.io/splitOnClone`とバックエンドのスペース消費が削減されますが、親ボリュームとクローンボリュームの間に依存関係が作成されるので、最初にクローンを削除しないかぎり親ボリュームを削除できません。クローンをスプリットするシナリオでは、空のデータベースボリュームをクローニングする方法が効果的です。このシナリオでは、ボリュームとそのクローンで使用するデータベースボリュームのサイズが大きく異なっており、 ONTAP ではストレージ効率化のメリットはありません。

この `sample-input`ディレクトリには、Tridentで使用するPVC定義の例が含まれています。Tridentボリュームに関連するパラメータと設定の詳細については、を参照してください。

`PersistentVolume`Kubernetesオブジェクト

Kubernetes `PersistentVolume`オブジェクトは、Kubernetesクラスタで使用可能になるストレージの一部を表します。ポッドに依存しないライフサイクルがあります。

> Tridentは、プロビジョニングするボリュームに基づいて自動的にオブジェクトを作成し `PersistentVolume`、Kubernetesクラスタに登録します。自分で管理することは想定されていません。

Tridentベースを参照するPVCを作成すると `StorageClass`、Tridentは対応するストレージクラスを使用して新しいボリュームをプロビジョニングし、そのボリュームの新しいPVを登録します。プロビジョニングされたボリュームと対応する PV の構成では、 Trident は次のルールに従います。

- Trident は、 Kubernetes に PV 名を生成し、ストレージのプロビジョニングに使用する内部名を生成します。どちらの場合も、名前がスコープ内で一意であることが保証されます。
- ボリュームのサイズは、 PVC で要求されたサイズにできるだけ近いサイズに一致しますが、プラットフォームによっては、最も近い割り当て可能な数量に切り上げられる場合があります。

`StorageClass`Kubernetesオブジェクト

Kubernetes `StorageClass`オブジェクトは、の名前で指定し `PersistentVolumeClaims`、一連のプロパティを使用してストレージをプロビジョニングします。ストレージクラス自体が、使用するプロビジョニングツールを特定し、プロビジョニングツールが理解できる一連のプロパティを定義します。

管理者が作成および管理する必要がある 2 つの基本オブジェクトのうちの 1 つです。もう 1 つは Trident バックエンドオブジェクトです。

Tridentを使用するKubernetes `StorageClass`オブジェクトは次のようになります。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

これらのパラメータは Trident 固有で、クラスのボリュームのプロビジョニング方法を Trident に指示します。

ストレージクラスのパラメータは次のとおりです。

属性	タイプ	必須	製品説明
属性	[string] 文字列をマップします	いいえ	後述の「属性」セクションを参照してください

属性	タイプ	必須	製品説明
ストレージプール	[string] StringList をマップします	いいえ	内のストレージプールのリストへのバックエンド名のマッピング
AdditionalStoragePools	[string] StringList をマップします	いいえ	内のストレージプールのリストへのバックエンド名のマッピング
excludeStoragePools	[string] StringList をマップします	いいえ	内のストレージプールのリストへのバックエンド名のマッピング

ストレージ属性とその有効な値は、ストレージプールの選択属性と Kubernetes 属性に分類できます。

ストレージプールの選択の属性

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

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

属性	タイプ	値	提供	リクエスト	でサポートされます
Snapshot	ブール値	true false	プールは、Snapshot を含むボリュームをサポートします	Snapshot が有効なボリューム	ONTAP-NAS, ONTAP-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

ほとんどの場合、要求された値はプロビジョニングに直接影響します。たとえば、シックプロビジョニングを要求した場合、シックプロビジョニングボリュームが使用されます。ただし、 Element ストレージプールでは、提供されている IOPS の最小値と最大値を使用して、要求された値ではなく QoS 値を設定します。この場合、要求された値はストレージプールの選択のみに使用されます。

理想的には、単独でを使用して、特定のクラスのニーズを満たすために必要なストレージの品質をモデル化できます `attributes`。Tridentは、指定したの_all_に一致するストレージプールを自動的に検出して選択します `attributes`。

を使用してクラスに適したプールを自動的に選択できない場合 `attributes`は、パラメータと `additionalStoragePools`パラメータを使用してプールをさらに絞り込んだり、特定のプールセットを選択したりできます `storagePools`。

パラメータを使用すると、指定したいずれかに一致するプールのセットをさらに制限 `attributes`でき `storagePools`ます。つまり、Tridentでは、パラメータと `storagePools`パラメータで識別されたプールの共通部分がプロビジョニングに使用され `attributes`ます。どちらか一方のパラメータを単独で使用することも、両方を同時に使用することも

パラメータを使用すると、パラメータと `storagePools`パラメータで選択したプールに関係なく、Tridentがプロビジョニングに使用するプールのセットを拡張 `attributes`でき `additionalStoragePools`ます。

パラメータを使用すると、Tridentがプロビジョニングに使用する一連のプールをフィルタリングできます `excludeStoragePools`。このパラメータを使用すると、一致するプールがすべて削除されます。

`storagePools`パラメータおよび`additionalStoragePools`パラメータでは、各エントリはの形式になり`<backend>:<storagePoolList>`ます。`<storagePoolList>`は、指定したバックエンドのストレージプールをカンマで区切ったリストです。たとえば、の値 `additionalStoragePools`はのようになります`ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`。これらのリストでは、バックエンド値とリスト値の両方に正規表現値を使用できます。を使用すると、バックエンドとそのプールのリストを取得できます `tridentctl get backend`。

Kubernetes の属性

これらの属性は、動的プロビジョニングの際に Trident が選択するストレージプール / バックエンドには影響しません。代わりに、 Kubernetes Persistent Volume でサポートされるパラメータを提供するだけです。ワーカーノードはファイルシステムの作成操作を担当し、 xfsprogs などのファイルシステムユーティリティを必要とする場合があります。

属性	タイプ	値	製品説明	関連するドライバ	**Kubernetes** のバージョン
FSstype （英語）	文字列	ext4、ext3、xfs	ブロックボリュームのファイルシステムのタイプ	solidfire-san-group、ontap/nas、ontap -nas-エコノミー、ontap -nas-flexgroup 、ontap -san 、ONTAP - SAN -経済性	すべて
allowVolumeExpansion の略	ブーリアン	true false	PVC サイズの拡張のサポートをイネーブルまたはディセーブルにします	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup 、 ONTAPSAN 、 ONTAP-SAN-エコノミー 、 solidfire-san-, gcvs, azure-netapp-files	1.11以上
volumeBindingMode のようになりました	文字列	即時、 WaitForFirstConsumer	ボリュームバインドと動的プロビジョニングを実行するタイミングを選択します	すべて	1.19 - 1.26

- パラメータは、 `fsType`SAN LUNに必要なファイルシステムタイプを制御するために使用します。さらに、Kubernetesはストレージクラスにが含まれていることを使用して `fsType`、ファイルシステムが存在することを示します。ボリューム所有権は、が設定されている場合にのみ、ポッドのセキュリティコンテキストを `fsType`使用して制御でき `fsGroup`ます。コンテキストを使用したボリューム所有権の設定の概要については `fsGroup`、を参照してください"Kubernetes ：ポッドまたはコンテナのセキュリティコンテキストを設定します"。Kubernetesがこの値を適用する `fsGroup`のは、次の場合のみです。

  - `fsType`はストレージクラスに設定されます。
  - PVC アクセスモードは RWO です。

  NFS ストレージドライバの場合、 NFS エクスポートにはファイルシステムがすでに存在します。ストレージクラスを使用する `fsGroup`には、を指定する必要があり `fsType`ます。またはnull以外の任意の値に設定できます。 `nfs`

- ボリューム拡張の詳細については、を参照してください"ボリュームを展開します"。
- Tridentインストーラバンドルには、のTridentで使用するストレージクラスの定義例がいくつか用意されていますsample-input/storage-class-*.yaml。Kubernetes ストレージクラスを削除すると、対応する Trident ストレージクラスも削除されます。

`VolumeSnapshotClass`Kubernetesオブジェクト

Kubernetes `VolumeSnapshotClass`オブジェクトはに似てい `StorageClasses`ます。この Snapshot コピーは、複数のストレージクラスの定義に役立ちます。また、ボリューム Snapshot によって参照され、Snapshot を必要な Snapshot クラスに関連付けます。各ボリューム Snapshot は、単一のボリューム Snapshot クラスに関連付けられます。

スナップショットを作成するには、管理者がを `VolumeSnapshotClass`定義する必要があります。ボリューム Snapshot クラスは、次の定義で作成されます。

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

は driver、Kubernetesに対して、クラスのボリュームSnapshotの要求がTridentで処理されるように指定します csi-snapclass。は、 `deletionPolicy`Snapshotを削除する必要がある場合に実行する処理を指定します。 `deletionPolicy`をに設定する `Delete`と、Snapshotを削除すると、ボリュームSnapshotオブジェクトとストレージクラスタ上の基盤となるSnapshotが削除されます。または、に設定する `Retain`と、`VolumeSnapshotContent`物理Snapshotが保持されます。

`VolumeSnapshot`Kubernetesオブジェクト

Kubernetes `VolumeSnapshot`オブジェクトは、ボリュームのSnapshotの作成要求です。PVC がボリュームに対するユーザからの要求を表すのと同様に、ボリュームスナップショットは、ユーザが既存の PVC のスナップショットを作成する要求です。

ボリュームSnapshot要求を受信すると、Tridentはバックエンドでのボリュームの Snapshotの作成を自動的に管理し、一意のオブジェクトを作成してそのSnapshotを公開します。
`VolumeSnapshotContent`既存の PVC からスナップショットを作成し、新しい PVC を作成するときにスナップショットを DataSource として使用できます。

> ⓘ VolumeSnapshot のライフサイクルはソース PVC とは無関係です。ソース PVC が削除されても、スナップショットは維持されます。スナップショットが関連付けられている PVC を削除すると、 Trident はその PVC のバッキングボリュームを **Deleting** 状態でマークしますが、完全には削除しません。関連付けられている Snapshot がすべて削除されると、ボリュームは削除されます。

`VolumeSnapshotContent`Kubernetesオブジェクト

Kubernetes `VolumeSnapshotContent`オブジェクトは、プロビジョニング済みのボリュームから取得されたSnapshotを表します。これは、に似て `PersistentVolume`おり、ストレージクラスタにプロビジョニングされたSnapshotを示します。オブジェクトと `PersistentVolume`オブジェクトと同様に、`PersistentVolumeClaim`Snapshotが作成されると、 `VolumeSnapshotContent`オブジェクトはSnapshotの作成を要求したオブジェクトへの1対1のマッピングを保持し `VolumeSnapshot`ます。

> `VolumeSnapshotContent`オブジェクトには、Snapshotを一意に識別する詳細（など）が含まれます `snapshotHandle`。これは `snapshotHandle`、PVの名前とオブジェクトの名前の一意の組み合わせ `VolumeSnapshotContent`です。

Trident では、スナップショット要求を受信すると、バックエンドにスナップショットが作成されます。スナップショットが作成されると、Tridentはオブジェクトを構成し VolumeSnapshotContent、スナップショットをKubernetes APIに公開します。

> ⓘ 通常、オブジェクトを管理する必要はありませ `VolumeSnapshotContent`ん。ただし、Trident の外部でを作成する場合は例外です"ボリュームSnapshotのインポート"。

`CustomResourceDefinition`Kubernetesオブジェクト

Kubernetes カスタムリソースは、管理者が定義した Kubernetes API 内のエンドポイントであり、類似するオブジェクトのグループ化に使用されます。Kubernetes では、オブジェクトのコレクションを格納するためのカスタムリソースの作成をサポートしています。これらのリソース定義は、を実行して取得できます `kubectl get crds`。

カスタムリソース定義（ CRD ）と関連するオブジェクトメタデータは、 Kubernetes によってメタデータストアに格納されます。これにより、 Trident の独立したストアが不要になります。

Tridentは、オブジェクトを使用し `CustomResourceDefinition`て、Tridentバックエンド、Tridentストレージクラス、TridentボリュームなどのTridentオブジェクトのIDを保持します。これらのオブジェクトは Trident によって管理されます。また、 CSI のボリュームスナップショットフレームワークには、ボリュームスナップショットの定義に必要ないくつかの SSD が導入されています。

CRD は Kubernetes の構成要素です。上記で定義したリソースのオブジェクトは Trident によって作成されます。簡単な例として、を使用してバックエンドを作成する `tridentctl`と、Kubernetesで使用するために対応する `tridentbackends`CRDオブジェクトが作成されます。

Trident の CRD については、次の点に注意してください。

- Trident をインストールすると、一連の CRD が作成され、他のリソースタイプと同様に使用できるようになります。
- コマンドを使用してTridentをアンインストールする `tridentctl uninstall`と、Tridentポッドは削除されますが、作成されたCRDはクリーンアップされません。Tridentを完全に削除してゼロから再構成する方法については、を参照してください"Trident をアンインストールします"。

Trident `StorageClass`オブジェクト

Tridentは、プロビジョニングツールフィールドで指定されたKubernetesオブジェクト `csi.trident.netapp.io`に一致するストレージクラスを作成します `StorageClass`。ストレージクラス名は、そのストレージクラスが表すKubernetesオブジェクトの名前と一致し `StorageClass`ます。

> Kubernetesでは、Tridentをプロビジョニングツールとして使用するKubernetesを登録すると、これらのオブジェクトが自動的に作成され `StorageClass`ます。

ストレージクラスは、ボリュームの一連の要件で構成されます。Trident は、これらの要件と各ストレージプール内の属性を照合し、一致する場合は、そのストレージプールが、そのストレージクラスを使用するボリュームのプロビジョニングの有効なターゲットになります。

REST API を使用して、ストレージクラスを直接定義するストレージクラス設定を作成できます。ただし、Kubernetesデプロイメントの場合は、新しいKubernetesオブジェクトを登録するときに作成されることを想定してい `StorageClass`ます。

Trident バックエンドオブジェクト

バックエンドとは、 Trident がボリュームをプロビジョニングする際にストレージプロバイダを表します。 1 つの Trident インスタンスであらゆる数のバックエンドを管理できます。

> これは、自分で作成および管理する 2 つのオブジェクトタイプのうちの 1 つです。もう1つはKubernetes `StorageClass`オブジェクトです。

これらのオブジェクトの作成方法の詳細については、を参照してください"バックエンドの設定"。

Trident `StoragePool`オブジェクト

ストレージプールは、各バックエンドでのプロビジョニングに使用できる個別の場所を表します。ONTAP の場合、これらは SVM 内のアグリゲートに対応します。NetApp HCI / SolidFire では、管理者が指定した QoS 帯域に対応します。Cloud Volumes Service の場合、これらはクラウドプロバイダのリージョンに対応します。各ストレージプールには、パフォーマンス特性とデータ保護特性を定義するストレージ属性があります。

他のオブジェクトとは異なり、ストレージプールの候補は常に自動的に検出されて管理されます。

Trident `Volume`オブジェクト

ボリュームはプロビジョニングの基本単位であり、NFS共有、iSCSI LUN、FC LUNなどのバックエンドエンドエンドポイントで構成されます。Kubernetesでは、これらはに直接対応し `PersistentVolumes`ます。ボリュームを作成するときは、そのボリュームにストレージクラスが含まれていることを確認します。このクラスによって、ボリュームをプロビジョニングできる場所とサイズが決まります。

- Kubernetes では、これらのオブジェクトが自動的に管理されます。Trident がプロビジョニングしたものを表示できます。
- 関連付けられた Snapshot がある PV を削除すると、対応する Trident ボリュームが * Deleting * 状態に更新されます。Trident ボリュームを削除するには、ボリュームの Snapshot を削除する必要があります。

ボリューム構成は、プロビジョニングされたボリュームに必要なプロパティを定義します。

属性	タイプ	必須	製品説明
バージョン	文字列	いいえ	Trident API のバージョン（「 1 」）
名前	文字列	はい	作成するボリュームの名前
ストレージクラス	文字列	はい	ボリュームのプロビジョニング時に使用するストレージクラス
サイズ	文字列	はい	プロビジョニングするボリュームのサイズ（バイト単位）
プロトコル	文字列	いいえ	使用するプロトコルの種類：「 file 」または「 block 」
インターン名	文字列	いいえ	Trident が生成した、ストレージシステム上のオブジェクトの名前
cloneSourceVolume の実行中です	文字列	いいえ	ONTAP （ NAS 、 SAN ） & SolidFire - * ：クローン元のボリュームの名前
splitOnClone	文字列	いいえ	ONTAP （ NAS 、 SAN ）：クローンを親からスプリットします
Snapshot ポリシー	文字列	いいえ	ONTAP - * ：使用する Snapshot ポリシー
Snapshot リザーブ	文字列	いいえ	ONTAP - * ： Snapshot 用にリザーブされているボリュームの割合
エクスポートポリシー	文字列	いいえ	ONTAP-NAS* ：使用するエクスポートポリシー
snapshotDirectory の略	ブール値	いいえ	ONTAP-NAS* ： Snapshot ディレクトリが表示されているかどうか
unixPermissions	文字列	いいえ	ONTAP-NAS* ：最初の UNIX 権限

属性	タイプ	必須	製品説明
ブロックサイズ	文字列	いいえ	SolidFire - * ：ブロック / セクターサイズ
ファイルシステム	文字列	いいえ	ファイルシステムタイプ

ボリュームの作成時にTridentで生成され `internalName`ます。この構成は 2 つのステップで構成されます。最初に、ボリューム名の先頭にストレージプレフィックス（デフォルトまたはバックエンド構成のプレフィックス）が付加さ `trident`れ、形式の名前が生成されます。 `<prefix>-<volume-name>`その後、名前の完全消去が行われ、バックエンドで許可されていない文字が置き換えられます。ONTAPバックエンドの場合、ハイフンはアンダースコアに置き換えられます（内部名はになります `<prefix>_<volume-name>` ）。Element バックエンドの場合、アンダースコアはハイフンに置き換えられます。

ボリューム構成を使用して、REST APIを使用してボリュームを直接プロビジョニングできますが、Kubernetes環境では、ほとんどのユーザが標準のKubernetesメソッドを使用することを想定してい`PersistentVolumeClaim`ます。Trident は、プロビジョニングプロセスの一環として、このボリュームオブジェクトを自動的に作成します。

Trident `Snapshot`オブジェクト

Snapshot はボリュームのポイントインタイムコピーで、新しいボリュームのプロビジョニングやリストア状態に使用できます。Kubernetesでは、これらはオブジェクトに直接対応し`VolumeSnapshotContent`ます。各 Snapshot には、 Snapshot のデータのソースであるボリュームが関連付けられます。

各 `Snapshot`オブジェクトには、次のプロパティが含まれています。

属性	タイプ	必須	製品説明
バージョン	文字列	はい	Trident API のバージョン（「 1 」）
名前	文字列	はい	Trident Snapshot オブジェクトの名前
インターン名	文字列	はい	ストレージシステム上の Trident Snapshot オブジェクトの名前
ボリューム名	文字列	はい	Snapshot を作成する永続的ボリュームの名前
ボリュームの内部名	文字列	はい	ストレージシステムに関連付けられている Trident ボリュームオブジェクトの名前

Kubernetes では、これらのオブジェクトが自動的に管理されます。Trident がプロビジョニングしたものを表示できます。

Kubernetesオブジェクト要求が作成されると、 `VolumeSnapshot`Tridentは元のストレージシステムにSnapshotオブジェクトを作成することで機能します。このSnapshotオブジェクトのは`internalName`、プレフィックスと `VolumeSnapshot`オブジェクトのを `UID`組み合わせることで生成され `snapshot-`ます（例： `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`）。`volumeName`および `volumeInternalName`は、バッキングボリュームの詳細を取得することによって読み込

まれます。

Trident `ResourceQuota`オブジェクト

Tridentデーモンセットは、Kubernetesで利用可能な最高の優先度クラスであるプライオリティクラスを使用して `system-node-critical`、ノードの正常なシャットダウン時にTridentがボリュームを識別してクリーンアップできるようにし、リソースへの負荷が高いクラスタでは、Tridentデーモンセットポッドが優先度の低いワークロードをプリエンプトできるようにします。

これを実現するために、Tridentはオブジェクトを使用し `ResourceQuota`て、`Trident`デーモンセットの「`system-node-critical`」優先クラスを確実に満たします。デプロイメントおよびデーモンセットの作成の前に、`Trident`はオブジェクトを検索し `ResourceQuota`、検出されていない場合は適用します。

デフォルトのリソースクォータと優先クラスを詳細に制御する必要がある場合は、Helmチャートを使用してオブジェクトを生成または設定 `ResourceQuota`できます `custom.yaml`。

次に示すのは'ResourceQuota'オブジェクトがTridentのデマ作用を優先する例です

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

リソースクォータの詳細については、を参照してください"Kubernetes：リソースクォータ"。

インストールに失敗した場合のクリーンアップ `ResourceQuota`

まれに、オブジェクトの作成後にインストールが失敗する `ResourceQuota`場合は、最初に再インストールを試行してから再インストールして"アンインストール"ください。

それでも問題が解決しない場合は、オブジェクトを手動で削除し `ResourceQuota`ます。

取り外す `ResourceQuota`

独自のリソース割り当てを制御する場合は、次のコマンドを使用してTridentオブジェクトを削除できます `ResourceQuota`。

```
kubectl delete quota trident-csi -n trident
```

PODセキュリティ標準（PSS）およびセキュリティコンテキストの制約（SCC）

Kubernetesポッドのセキュリティ標準（PSS）とポッドのセキュリティポリシー（PSP）によって、権限レベルが定義され、ポッドの動作が制限されます。また、OpenShift Security Context Constraints（SCC）でも、OpenShift Kubernetes Engine固有のポッド制限を定義します。このカスタマイズを行うために、Tridentはインストール時に特定の権限を有効にします。次のセクションでは、Tridentによって設定される権限について詳しく説明します。

PSSは、Podセキュリティポリシー（PSP）に代わるものです。PSPはKubernetes v1.21で廃止され、v1.25で削除されます。詳細については、を参照してください"Kubernetes：セキュリティ"。

必須の**Kubernetes Security Context**と関連フィールド

権限	製品説明
権限があります	CSIでは、マウントポイントが双方向である必要があります。つまり、Tridentノードポッドで特権コンテナを実行する必要があります。詳細については、を参照してください "Kubernetes：マウントの伝播"。
ホストネットワーク	iSCSIデーモンに必要です。 `iscsiadm`iSCSIマウントを管理し、ホストネットワークを使用してiSCSIデーモンと通信します。
ホストIPC	NFSはIPC（プロセス間通信）を使用して'nfsdと通信します
ホストPID	NFSで開始する必要があり `rpc-statd`ます。Tridentは、NFSボリュームをマウントする前にが実行されているかどうかをホストプロセスに照会して判断し `rpc-statd`ます。

機能	この `SYS_ADMIN`機能は、特権コンテナのデフォルト機能の一部として提供されています。たとえば、`Docker`は特権コンテナに次の機能を設定します。
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccom	Seccompプロファイルは特権コンテナでは常に「制限されていない」ため、Tridentでは有効にできません。

権限	製品説明
SELinux	OpenShiftでは、特権コンテナは（「Super Privileged Container」）ドメインで実行され、非特権コンテナはドメインで実行さ `spc_t`れ `container_t`ます。では `containerd`、がインストールされていると `container-selinux`、すべてのコンテナがドメイン内で実行され `spc_t`、SELinuxが実質的に無効になります。そのため、Tridentはコンテナに追加しません `seLinuxOptions`。
DAC	特権コンテナは、ルートとして実行する必要があります。CSIに必要なUNIXソケットにアクセスするために、非特権コンテナはrootとして実行されます。

PODセキュリティ標準（PSS）

ラベル	製品説明	デフォルト
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Tridentコントローラとノードをインストールネームスペースに登録できるようにします。ネームスペースラベルは変更しないでください。	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

名前空間ラベルを変更すると、ポッドがスケジュールされず、「Error creating：…」または「Warning：trident-csi-…」が表示される場合があります。この場合は、のネームスペースラベルが変更されていないかどうかを確認してください `privileged`。その場合は、Tridentを再インストールします。

PoDセキュリティポリシー（PSP）

フィールド	製品説明	デフォルト
`allowPrivilegeEscalation`	特権コンテナは、特権昇格を許可する必要があります。	`true`
`allowedCSIDrivers`	TridentはインラインCSIエフェメラルボリュームを使用しません。	空
`allowedCapabilities`	権限のないTridentコンテナにはデフォルトよりも多くの機能が必要ないため、特権コンテナには可能なすべての機能が付与されます。	空
`allowedFlexVolumes`	Tridentではを使用できない"FlexVolドライバ"ため、これらのボリュームは許可されるボリュームのリストに含まれていません。	空
`allowedHostPaths`	Tridentノードポッドでノードのルートファイルシステムがマウントされるため、このリストを設定してもメリットはありません。	空

フィールド	製品説明	デフォルト
allowedProcMountTypes	Tridentはいずれも使用しません ProcMountTypes。	空
allowedUnsafeSysctls	Tridentには安全でないものは必要あり `sysctls`ません。	空
defaultAddCapabilities	特権コンテナに追加する機能は必要ありません。	空
defaultAllowPrivilegeEscalation	権限の昇格は、各Tridentポッドで処理されます。	false
forbiddenSysctls	いいえは `sysctls`許可されません。	空
fsGroup	Tridentコンテナはrootとして実行されます。	RunAsAny
hostIPC	NFSボリュームをマウントするにはホストIPCが通信する必要があります nfsd	true
hostNetwork	iscsiadmには、iSCSIデーモンと通信するためのホストネットワークが必要です。	true
hostPID	ホストPIDは、がノードで実行されているかどうかを確認するために必要 `rpc-statd`です。	true
hostPorts	Tridentはホストポートを使用しません。	空
privileged	Tridentノードのポッドでは、ボリュームをマウントするために特権コンテナを実行する必要があります。	true
readOnlyRootFilesystem	Tridentノードのポッドは、ノードのファイルシステムに書き込む必要があります。	false
requiredDropCapabilities	Tridentノードのポッドは特権コンテナを実行するため、機能をドロップすることはできません。	none
runAsGroup	Tridentコンテナはrootとして実行されます。	RunAsAny
runAsUser	Tridentコンテナはrootとして実行されます。	runAsAny
runtimeClass	Tridentはを使用しません RuntimeClasses。	空

フィールド	製品説明	デフォルト
`seLinux`	現在、コンテナランタイムとKubernetesディストリビューションでSELinuxを処理する方法が異なるため、Tridentは設定されて`seLinuxOptions`いません。	空
`supplementalGroups`	Tridentコンテナはrootとして実行されます。	`RunAsAny`
`volumes`	Tridentポッドには、このボリュームプラグインが必要です。	`hostPath, projected, emptyDir`

セキュリティコンテキストの制約（SCC）

ラベル	製品説明	デフォルト
`allowHostDirVolumePlugin`	Tridentノードのポッドは、ノードのルートファイルシステムをマウントします。	`true`
`allowHostIPC`	NFSボリュームをマウントするには、ホストIPCと通信する必要があり`nfsd`ます。	`true`
`allowHostNetwork`	iscsiadmには、iSCSIデーモンと通信するためのホストネットワークが必要です。	`true`
`allowHostPID`	ホストPIDは、がノードで実行されているかどうかを確認するために必要`rpc-statd`です。	`true`
`allowHostPorts`	Tridentはホストポートを使用しません。	`false`
`allowPrivilegeEscalation`	特権コンテナは、特権昇格を許可する必要があります。	`true`
`allowPrivilegedContainer`	Tridentノードのポッドでは、ボリュームをマウントするために特権コンテナを実行する必要があります。	`true`
`allowedUnsafeSysctls`	Tridentには安全でないものは必要あり`sysctls`ません。	`none`
`allowedCapabilities`	権限のないTridentコンテナにはデフォルトよりも多くの機能が必要ないため、特権コンテナには可能なすべての機能が付与されます。	空
`defaultAddCapabilities`	特権コンテナに追加する機能は必要ありません。	空
`fsGroup`	Tridentコンテナはrootとして実行されます。	`RunAsAny`

ラベル	製品説明	デフォルト
`groups`	このSCCはTridentに固有で、ユーザにバインドされています。	空
`readOnlyRootFilesystem`	Tridentノードのポッドは、ノードのファイルシステムに書き込む必要があります。	`false`
`requiredDropCapabilities`	Tridentノードのポッドは特権コンテナを実行するため、機能をドロップすることはできません。	`none`
`runAsUser`	Tridentコンテナはrootとして実行されます。	`RunAsAny`
`seLinuxContext`	現在、コンテナランタイムとKubernetesディストリビューションでSELinuxを処理する方法が異なるため、Tridentは設定されて`seLinuxOptions`いません。	空
`seccompProfiles`	特権のあるコンテナは常に「閉鎖的」な状態で実行されます。	空
`supplementalGroups`	Tridentコンテナはrootとして実行されます。	`RunAsAny`
`users`	このSCCをTridentネームスペースのTridentユーザにバインドするエントリが1つあります。	N/A
`volumes`	Tridentポッドには、このボリュームプラグインが必要です。	`hostPath, downwardAPI, projected, emptyDir`