参照
Astra Trident

NetApp
January 14, 2026

目次

参照

Astra Trident ポート

Tridentが通信に使用するポートの詳細をご確認ください。

Astra Trident ポート

Astra Trident は次のポート経由で通信：

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
8、000	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

活性/レディネスプローブポートは、を使用して設置するときに変更できます `--probe-port` フラグ。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。

Astra Trident REST API

間 "tridentctl コマンドとオプション" Trident REST APIを使用するには、RESTエンドポイントを直接使用する方法が最も簡単です。

REST APIを使用する状況

REST APIは、Kubernetes以外の環境でAstra 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。

セキュリティ強化のため、Astra Tridentをぜひご利用ください `REST API` ポッド内で実行されている場合は、デフォルトでlocalhostに制限されます。この動作を変更するには、Astra Tridentを設定する必要があります `-address` 引数をポッド構成で指定します。

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、デバッグを渡してください (`-d`）フラグ。詳細については、を参照してください "Tridentctlを使用したAstra 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** ）

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Astra 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Astra Tridentの管理"。

コマンドラインオプション

Trident 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をいくつか公開しています。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 `-k8s_api_server` をクリックしてKubernetesのサポートを有効にしこれを設定すると、 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 `-k8s_pod` をクリックしてKubernetesのサポートを有効にし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することができますが、サポートされるのは API サーバへのセキュアでない接続だけです。セキュアに接続するには、Tridentをポッドに搭載し、を使用して導入します `-k8s_pod` オプション

Docker です

-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。使用 `-address ""` RESTインターフェイスにポッドのIPアドレスからアクセスできるようにするため。

> 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. ユーザがを作成します `PersistentVolumeClaim` 新しいを要求しています `PersistentVolume` 特定のサイズのものをKubernetesから取得します `StorageClass` 以前に管理者によって設定されていたもの。
2. Kubernetes `StorageClass` Tridentをプロビジョニングツールとして特定し、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設定します。
3. Tridentはその外観を独自にしています `StorageClass` 一致するものと同じ名前を使用します `Backends` および `StoragePools` を使用して、クラスのボリュームをプロビジョニングできます。
4. Tridentは、一致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。A `PersistentVolume` Kubernetesで、ボリュームとTrident内のボリュームを検出、マウント、処理し、間の関係を保持する方法を指示します `PersistentVolume` 実際のストレージをサポートします。
5. Kubernetesがをバインド `PersistentVolumeClaim` を新しいに変更します `PersistentVolume`。を含むポッド `PersistentVolumeClaim` このPersistentVolumeを、実行されている任意のホストにマウントします。
6. ユーザがを作成します `VolumeSnapshot` を使用した既存のPVCの `VolumeSnapshotClass` Tridentを指しています。
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、を作成します `VolumeSnapshotContent` これにより、Snapshotの識別方法をKubernetesに指示します。
8. ユーザはを作成できます `PersistentVolumeClaim` を使用します `VolumeSnapshot` をソースとして使用します。
9. Tridentが必要なSnapshotを特定し、の作成と同じ手順を実行します `PersistentVolume` および `Volume`。

Kubernetes オブジェクトの詳細については、を参照することを強く推奨します "永続ボリューム" Kubernetes のドキュメントのセクション。

Kubernetes `PersistentVolumeClaim` オブジェクト

Kubernetesを `PersistentVolumeClaim` オブジェクトは、Kubernetesクラスタユーザが作成するストレージの要求です。

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、azure-netapp-files、gcp-cvs、 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

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/exportPolicy	エクスポートポリシー	ONTAP-NAS、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	Snapshot ポリシー	ONTAP-NAS、ontap-nas-economy、ontap-nas-flexgroup、ontap-SAN
trident.netapp.io/snapshotReserve	Snapshot リザーブ	ONTAP-NAS、ontap-nas-flexgroup、ontap-san 、gcp-cvs
trident.netapp.io/snapshotDirectory	snapshotDirectory の略	ONTAP-NAS、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ONTAP-NAS、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/blockSize	ブロックサイズ	solidfire - SAN

作成されたPVにがある場合 `Delete` ポリシーを再利用すると、PVが解放されたとき（つまり、ユーザがPVCを削除したとき）に、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クリーンアップすることを前提としているため、ボリュームを削除する前にバックアップや検査を実行できます。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ます (`tridentctl`）。

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。を参照してください `On-Demand Volume Snapshots` これがどのように機能するかを確認します。

Tridentが提供するのも `cloneFromPVC` および `splitOnClone` クローンを作成するためのアノテーション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`を使用すると、ユーザはという新しいPVCを作成できます `mysqlclone` などのアノテーションを使用する `trident.netapp.io/cloneFromPVC: mysql`。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。

- アイドルボリュームのクローンを作成することを推奨します。
- 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。
- を使用 `ontap-nas` および `ontap-san` ドライバが必要な場合は、PVC注釈を設定することをお勧めします `trident.netapp.io/splitOnClone` と組み合わせて使用します

`trident.netapp.io/cloneFromPVC`。を使用 `trident.netapp.io/splitOnClone` をに設定します `true`Tridentでは、クローニングされたボリュームを親ボリュームからスプリットするため、ストレージ効率を維持しないまま、クローニングされたボリュームのライフサイクルを完全に分離します。設定されていません `trident.netapp.io/splitOnClone` またはに設定します `false` 親ボリュームとクローンボリューム間の依存関係を作成するのではなく、バックエンドのスペース消費が削減されます。そのため、クローンを先に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。

。 `sample-input` Directoryには、Tridentで使用するPVC定義の例が含まれています。を参照してください をクリックして、Tridentボリュームに関連付けられているパラメータと設定の完全な概要を確認します。

Kubernetes `PersistentVolume` オブジェクト

Kubernetesを `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が実現 `PersistentVolume` オブジェクトを作成し、プロビジョニングするボリュームに基づいてKubernetesクラスタに自動的に登録します。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を作成する場合 `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、そのボリュームに新しい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。

Kubernetes `StorageClass` オブジェクト

Kubernetes `StorageClass` オブジェクトは、の名前で指定します `PersistentVolumeClaims` 一連のプロパティを指定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。

Kubernetesを `StorageClass` Tridentを使用する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	を入力します	必須	説明
属性（ Attributes ）	[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
ストレージプール	[string] StringList をマップします	いいえ	バックエンド名とリストのマッピング ストレージプール
AdditionalStoragePools	[string] StringList をマップします	いいえ	バックエンド名のマップ ストレージプールノリスト
excludeStoragePools	[string] StringList をマップします	いいえ	ハツクエントメイノマツヒンク ストレージプールノリスト

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

属性	を入力します	値	提供	リクエスト	でサポートされます
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、を使用できます `attributes` 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することだけを目的としています。Tridentは、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` を指定します。

自分が使用できない場合は `attributes` クラスに適したプールを自動的に選択するには、を使用します `storagePools` および `additionalStoragePools` プールをさらに細かく指定するためのパラメータ、または特定のプールセットを選択するためのパラメータ。

を使用できます `storagePools` 指定したパラメータに一致するプールをさらに制限します `attributes`。つまり、Tridentはによって識別されたプールの交点を使用します `attributes` および `storagePools` プロビジョニングのパラメータ。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

を使用できます `additionalStoragePools` Tridentがプロビジョニングに使用する一連のプールを、で選択されているプールに関係なく拡張するためのパラメータ `attributes` および `storagePools` パラメータ

を使用できます `excludeStoragePools` 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するためのパラメータ。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。

を参照してください `storagePools` および `additionalStoragePools` パラメータを指定すると、各エントリの形式がになります `<backend>:<storagePoolList>`、ここで `<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、の値などです `additionalStoragePools` 次のように表示されます `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。を使用できます `tridentctl get backend` バックエンドとそのプールのリストを取得します。

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	を入力します	値	説明	関連するドライバ	Kubernetes バージョン

FSstype （英語）	文字列	ext4 、 ext3 、 xfs など	ブロックのファイルシステムタイプ 個のボリューム	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	ブール値	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

- 。 `fsType` パラメータは、SAN LUNに必要なファイルシステムタイプを制御する場合に使用します。また、Kubernetesでは、の機能も使用されます `fsType` ファイルシステムが存在することを示すために、ストレージクラスに格納します。ボリューム所有権は、を使用して制御できます `fsGroup` ポッドのセキュリティコンテキスト（使用する場合のみ） `fsType` が設定されます。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" を使用したボリューム所有権の設定の概要については、を参照してください `fsGroup` コンテキスト（Context）。Kubernetesでが適用されます `fsGroup` 次の場合のみ値を指定します

  - `fsType` はストレージクラスで設定されます。
  - 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。を使用します `fsGroup` ストレージクラスでは、引き続きを指定する必要があります `fsType`。に設定できます `nfs` またはnull以外の値。

- を参照してください "ボリュームを展開します" ボリューム拡張の詳細については、を参照してください。
- Tridentのインストーラバンドルには、でTridentで使用するストレージクラス定義の例がいくつか含まれています `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。

Kubernetes VolumeSnapshotClass オブジェクト

Kubernetes VolumeSnapshotClass オブジェクトはに似ています StorageClasses。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。

A VolumeSnapshotClass Snapshotを作成するには、管理者によって定義されている必要があります。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
```

。 driver のボリュームSnapshotを要求するKubernetesに指定します csi-snapclass クラスはTridentによって処理されます。。 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。いつ deletionPolicy がに設定されます Delete`を指定すると、Snapshotが削除されたときに、ボリュームSnapshotオブジェクトおよび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、に設定します `Retain はそのことを示します VolumeSnapshotContent 物理スナップショットが保持されます。

Kubernetes VolumeSnapshot オブジェクト

Kubernetesを VolumeSnapshot objectは、ボリュームのSnapshotを作成する要求です。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要求が開始されると、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、一意のを作成してSnapshotを公開します VolumeSnapshotContent オブジェクト。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。

Kubernetes VolumeSnapshotContent オブジェクト

Kubernetesを VolumeSnapshotContent オブジェクトは、すでにプロビジョニングされているボリュームから作成されたSnapshotを表します。これはに似ています PersistentVolume とは、ストレージクラスタにプロビジョニングされたSnapshotを表します。に似ています PersistentVolumeClaim および PersistentVolume オブジェクト。スナップショットが作成されると、が表示されます VolumeSnapshotContent オブジェクトは、への1対1のマッピングを保持します VolumeSnapshot オブジェクト。オブジェクトはSnapshotの作成を要求しました。

。 `VolumeSnapshotContent` Objectには、など、Snapshotを一意に識別する詳細が含まれます `snapshotHandle`。これ `snapshotHandle` は、PVの名前との名前を一意に組み合わせたものです `VolumeSnapshotContent` オブジェクト。

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によってが設定されます `VolumeSnapshotContent` オブジェクトを作成することで、SnapshotをKubernetes APIに公開します。

通常は、 `VolumeSnapshotContent` オブジェクト。ただし、次の場合は例外です。 "ボリュームSnapshotのインポート" Astra Trident以外で作成

Kubernetes `CustomResourceDefinition` 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。を実行すると、これらのリソース定義を取得できます `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。

Astra Tridentが使用 `CustomResourceDefinition` Tridentバックエンド、Tridentストレージクラス、Tridentボリュームなど、TridentオブジェクトのIDを保持するオブジェクト。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場合を示します `tridentctl`に対応します `tridentbackends` CRDオブジェクトは、Kubernetesによって消費されるために作成されます。

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。
- Tridentをアンインストールするには、を使用します `tridentctl uninstall` コマンドであるTridentポッドが削除されましたが、作成されたSSDはクリーンアップされません。を参照してください "Trident をアンインストールします" Trident を完全に削除して再構成する方法を理解する。

Astra Trident `StorageClass` オブジェクト

TridentではKubernetesに対応するストレージクラスが作成されます `StorageClass` を指定するオブジェクト `csi.trident.netapp.io` プロビジョニング担当者のフィールドに入力します。ストレージクラス名がKubernetesの名前と一致していること `StorageClass` 表すオブジェクト。

Kubernetesでは、これらのオブジェクトはKubernetesのときに自動的に作成されます `StorageClass` Tridentをプロビジョニングツールとして使用していることが登録されます。

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環境では、新しいKubernetesを登録するときにKubernetes環境が作成されることを想定しています `StorageClass` オブジェクト。

Astra 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です `StorageClass` オブジェクト。

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"。

Astra 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。

Astra Trident `Volume` オブジェクト

ボリュームは、 NFS 共有や iSCSI LUN などのバックエンドエンドエンドエンドポイントで構成される、プロビジョニングの基本単位です。Kubernetesでは、これらはに直接対応します `PersistentVolumes`。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	を入力します	必須	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

属性	を入力します	必須	説明
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
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	ファイルシステムのタイプ

Tridentが生成 `internalName` ボリュームを作成する場合。この構成は 2 つのステップで構成されます。最初に、ストレージプレフィックス（デフォルトのプレフィックス）を先頭に追加します `trident` またはバックエンド構成内のプレフィックス）をボリューム名に変更して、形式の名前を指定します `<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、ハイフンをアンダースコアに置き換えます（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を使用することを想定しています `PersistentVolumeClaim` メソッドプロビジョニングの一環としてTridentがこのボリュームオブジェクトを自動的に作成
プロセス：

Astra 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、これらはに直接対応します `VolumeSnapshotContent` オブジェクト。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ます。

属性	を入力します	必須	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を導入したとき `VolumeSnapshot` オブジェクト要求が作成されると、Tridentはバッキング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。。 `internalName` このSnapshotオブジェクトのプレフィックスを組み合わせると、が生成されます `snapshot-` を使用 `UID` の `VolumeSnapshot` オブジェクト（例： `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`）。 `volumeName` および `volumeInternalName` 裏付けの詳細を取得することで入力されます。
ボリューム：

Astra Trident `ResourceQuota` オブジェクト

Tridentのデーモンは、を消費します `system-node-critical` 優先度クラス：Kubernetesで最も高い優先度クラスです。Astra Tridentは、ノードの正常なシャットダウン中にボリュームを識別してクリーンアップし、Tridentのデミスタポッドがリソースの負荷が高いクラスタでより低い優先度でワークロードをプリエンプトできるようにします。

そのために、Astra Tridentはを採用しています `ResourceQuota` Tridentのデミスタに対する「システムノードクリティカル」の優先クラスを満たすことを保証するオブジェクト。導入とデマ作用の開始前に、Astra Tridentがを探します `ResourceQuota` オブジェクトを検出し、検出されない場合は適用します。

デフォルトのリソースクォータおよび優先クラスをより詳細に制御する必要がある場合は、を生成できます `custom.yaml` またはを設定します `ResourceQuota` Helmチャートを使用するオブジェクト。

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"。

クリーンアップ `ResourceQuota` インストールが失敗した場合

まれに、のあとにインストールが失敗する場合があります `ResourceQuota` オブジェクトが作成されました。最初に実行してください "アンインストール中です" を再インストールします。

うまくいかない場合は、を手動で削除します `ResourceQuota` オブジェクト。

取り外します `ResourceQuota`

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

```
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、Astra 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、Astra Tridentによって設定された権限の詳細を説明します。

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

アクセス権	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	開始する必要があります `rpc-statd` NFSの場合：Astra Tridentがホストプロセスを照会して、状況を特定 `rpc-statd` を実行してからNFSボリュームをマウントしてください。

機能	。 `SYS_ADMIN` この機能は、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、権限のあるコンテナでは常に「制限なし」なので、Astra Tridentでは有効にできません。
SELinux	OpenShiftでは、特権のあるコンテナがで実行されます `spc_t` （「スーパー特権コンテナ」）ドメインおよび非特権コンテナは、で実行されます `container_t` ドメイン：オン `containerd`を使用 `container-selinux` インストールすると、すべてのコンテナがで実行されます `spc_t` domain。SELinuxは無効になります。そのため、Astra Tridentは機能しません `seLinuxOptions` コンテナへ。
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	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。その場合は、のネームスペースラベルを確認してください `privileged` が変更されました。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	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。	空です
`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。	空です
`allowedFlexVolumes`	Tridentはを利用しません "FlexVolドライバ"そのため、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。	空です
`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。	空です
`allowedProcMountTypes`	Tridentでは使用していません `ProcMountTypes`。	空です
`allowedUnsafeSysctls`	Tridentでは安全でないリソースは不要です `sysctls`。	空です
`defaultAddCapabilities`	特権コンテナに追加する機能は必要ありません。	空です
`defaultAllowPrivilegeEscalation`	権限の昇格は、各Tridentポッドで処理されます。	`false`
`forbiddenSysctls`	いいえ `sysctls` 許可されています。	空です
`fsGroup`	Tridentコンテナはrootとして実行されます。	`RunAsAny`

フィールド	説明	デフォルト
hostIPC	NFSボリュームをマウントするには、ホストIPCがと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が必要かどうかを確認します rpc-statd ノードで実行されている。	true
hostPorts	Tridentはホストポートを使用しません。	空です
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は使用しません RuntimeClasses。	空です
seLinux	Tridentが設定されていません seLinuxOptions 現在のところ、コンテナの実行時間とKubernetesのディストリビューションでのSELinuxの処理に違いがあるためです。	空です
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	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

ラベル	説明	デフォルト
`allowHostIPC`	NFSボリュームをマウントするには、ホストIPCがと通信する必要があります `nfsd`。	`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が必要かどうかを確認します `rpc-statd` ノードで実行されている。	`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では安全でないリソースは不要です `sysctls`。	`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。	空です
`defaultAddCapabilities`	特権コンテナに追加する機能は必要ありません。	空です
`fsGroup`	Tridentコンテナはrootとして実行されます。	`RunAsAny`
`groups`	このSCCはTridentに固有で、ユーザにバインドされています。	空です
`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`	Tridentが設定されていません `seLinuxOptions` 現在のところ、コンテナの実行時間とKubernetesのディストリビューションでのSELinuxの処理に違いがあるためです。	空です

ラベル	説明	デフォルト
`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。	空です
`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。	該当なし
`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`

著作権に関する情報

商標に関する情報