参照
Astra Trident

NetApp
July 31, 2024

目次

参照

Astra Trident ポート

Trident が通信するポートの詳細をご確認ください。

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
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 '--probe-port' フラグを使用してインストール時に変更できます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 の REST API を操作するには簡単です。REST エンドポイントは必要に応じて直接使用できます。

これは、 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。

セキュリティを強化するために、 Astra Trident の「 RE ST API 」は、ポッド内で実行される場合のデフォルトで 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 Astra Trident の「 -address 」引数を設定する必要があります。

API は次のように機能します。

`GET`

- 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/< オブジェクトタイプ >/< オブジェクト名 >: 名前付きオブジェクトの詳細を取得します。

`POST`

'POST <trident-address>/trident/v1/<object-type>` ：指定した型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 "ce87eb03803d5633c163541464e9e7f2"。

- 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。

`DELETE`

`D eleet<trident-address>/trident/v1/<` オブジェクトタイプ `>/<` オブジェクト名 `>` ：名前付き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 "ce87eb03803d5633c163541464e9e7f2"。

これらの API の呼び出し方法の例については 'debug(d`) フラグを渡してください詳細については、を参照してください "ce87eb03803d5633c163541464e9e7f2"。

コマンドラインオプション

Astra Trident には、いくつかのコマンドラインオプションが用意されています。これらの値は環境で変更できます。

ロギング

- --debug: デバッグ出力をイネーブルにします。
- -loglevel <level>` ：ロギングレベルを設定します（ debug 、 info 、 warn 、 error 、 fatal ）。デフォルトは info です。

Kubernetes

- `-k8s_pod `: Kubernetes サポートを有効にするには ' このオプションまたは -k8s_api_server を使用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 -k8s_pod を使用して Kubernetes 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、「 -k8s_pod 」オプションを使用してポッドに Trident を導入します。
- `-k8s_config_path<file>`: 必須。このパスを KubeConfig ファイルに指定する必要があります。

Docker です

- `-volume_driver <name>` ： Docker プラグインを登録するときに使用するドライバ名。デフォルトは 'NetApp' です
- `-ddriver_port <port-number>`: このポートでは 'UNIX ドメインソケットではなくリッスンします
- `-config <file>`: 必須。このパスをバックエンド構成ファイルに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には、「 -address 」を使用します。
- `-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 インタフェースを有効にしますデフォルトは true です。

ネットアップの製品が Kubernetes と統合されます

ネットアップのストレージ製品ポートフォリオは、 Kubernetes クラスタのさまざまな要素と統合され、高度なデータ管理機能を提供して、 Kubernetes 環境の機能、機能、パフォーマンス、可用性を強化します。

アストラ

"アストラ" Kubernetes 上で実行される大量のデータコンテナ化ワークロードを、パブリッククラウドとオンプレミスの間で簡単に管理、保護、移動できます。Astra は、ネットアップの実績のある拡張可能なストレージポートフォリオから、パブリッククラウドとオンプレミスに提供される Trident を使用して、永続的なコンテナストレージをプロビジョニングし、提供します。また、 Snapshot 、バックアップとリストア、アクティビティログ、アクティブクローニングによるデータ保護、ディザスタ / データリカバリ、データ監査、Kubernetes ワークロードの移行のユースケースなど、アプリケーションに対応した高度なデータ管理機能も豊富に搭載されています。

ONTAP

ONTAP は、あらゆるアプリケーションに高度なデータ管理機能を提供する、ネットアップのマルチプロトコルユニファイドストレージオペレーティングシステムです。ONTAP システムには、オールフラッシュ、ハイブリッド、オール HDD のいずれかの構成が採用されており、自社開発のハードウェア（ FAS と AFF ）、ノーブランド製品（ ONTAP Select ）、クラウドのみ（ Cloud Volumes ONTAP ）など、さまざまな導入モデルが用意されています。

> Trident は、上記の ONTAP 導入モデルをすべてサポートしています。

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" は、クラウドで ONTAP データ管理ソフトウェアを実行するソフトウェア型ストレージアプライアンスです。Cloud Volumes ONTAP は、本番ワークロード、ディザスタリカバリ、 DevOps 、ファイル共有、データベース管理に使用できます。ストレージ効率、高可用性、データレプリケーション、データ階層化、アプリケーションの整合性を提供することで、エンタープライズストレージをクラウドに拡張します。

NetApp ONTAP 対応の Amazon FSX

"NetApp ONTAP 対応の Amazon FSX" は、 NetApp ONTAP ストレージ・オペレーティング・システムを搭載したファイル・システムの起動と実行を可能にする、フルマネージドの AWS サービスです。FSX for ONTAP を使用すると、お客様は使い慣れたネットアップの機能、パフォーマンス、管理機能を活用しながら、 AWS にデータを格納する際のシンプルさ、即応性、セキュリティ、拡張性を活用できます。FSX for ONTAP は、ONTAP のファイルシステム機能と管理 API の多くをサポートしています。

Element ソフトウェア

"要素（ Element ）" ストレージ管理者は、パフォーマンスを保証し、ストレージの設置面積を合理化することで、ワークロードを統合できます。Element と API を組み合わせることでストレージ管理のあらゆる要素を自動化できるため、ストレージ管理者は少ない労力で多くの作業を行うことができます。

NetApp HCI

"NetApp HCI" 日常業務を自動化し、インフラ管理者がより重要な業務に集中できるようにすることで、データセンターの管理と拡張を簡易化します。

NetApp HCI は Trident によって完全にサポートされています。Trident では、コンテナ化されたアプリケーション用のストレージデバイスを、基盤となる NetApp HCI ストレージプラットフォームに直接プロビジョニングして管理できます。

SANtricity

ネットアップの E シリーズと EF シリーズのストレージプラットフォームでは、を使用します "SANtricity" 可用性とパフォーマンスに優れた堅牢なストレージを提供し、あらゆる規模のアプリケーションにストレージサービスを提供できるようにするためのオペレーティングシステム。

Trident では、製品ポートフォリオ全体で SANtricity ボリュームを作成、管理できます。

Azure NetApp Files の特長

"Azure NetApp Files の特長" は、ネットアップが提供するエンタープライズクラスの Azure ファイル共有サービスです。要件がきわめて厳しいファイルベースのワークロードも、ネットアップが提供するパフォーマンスと充実のデータ管理機能を使用して、 Azure でネイティブに実行できます。

Cloud Volumes Service for AWS

"NetApp Cloud Volumes Service for AWS" は、 NFS や SMB 経由で NAS ボリュームにオールフラッシュのパフォーマンスを提供する、クラウドネイティブのファイルサービスです。このサービスを使用すると、従来型アプリケーションを含むあらゆるワークロードを AWS クラウドで実行できます。フルマネージドサービスを提供し、ハイパフォーマンス、瞬時のクローニング、データ保護、 Elastic Container Service （ ECS ）インスタンスへのセキュアなアクセスを提供します。

Cloud Volumes Service for Google Cloud

"NetApp Cloud Volumes Service for Google Cloud" は、 NFS や SMB 経由で NAS ボリュームにオールフラッシュのパフォーマンスを提供する、クラウドネイティブのファイルサービスです。このサービスを使用すると、従来型アプリケーションを含むあらゆるワークロードを GCP クラウドで実行できます。フルマネージドサービスを提供し、一貫したハイパフォーマンス、瞬時のクローニング、データ保護、 Google Compute Engine （ GCE ）インスタンスへのセキュアなアクセスを実現します。

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. ユーザーは、「 PersistentVolumeClaim 」を作成して、特定のサイズの新しい「 PersistentVolume 」を、管理者が以前に設定した Kubernetes の「 torageClass 」から要求します。
2. Kubernetes の「 torageClass 」は、 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含んでいます。
3. Trident は、対応する「 Backends 」と「 toragePools 」を識別する同じ名前の「 S torageClass 」を参照します。この名前は、このクラスのボリュームのプロビジョニングに使用できます。
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。 Kubernetes では、「 PersistentVolume 」とは、ボリュームを検索、マウント、処理する方法を Kubernetes に伝える「 PersistentVolume 」と、「 PersistentVolume 」と実際のストレージの関係を保持する Trident 内のボリュームです。
5. Kubernetes は 'PersistentVolumeClaim を新しい 'PersistentVolume' にバインドしますPersistentVolume が実行される任意のホストに PersistentVolume をマウントする 'PersistentVolumeClaim を含むポッド。
6. ユーザーは、 Trident を指す「 VolumeSnapshotClass 」を使用して、既存の PVC の「 VolumeSnapshot 」を作成します。
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 ' スナップショットの識別方法を Kubernetes に指示する 'VolumeSnapshotContent' も作成します
8. ユーザーは 'VolumeSnapshot' をソースとして使用して 'PersistentVolumeClaim を作成できます
9. Trident は必要なスナップショットを識別し、「 PersistentVolume 」と「 Volume 」の作成に関連する一連のステップを実行します。

> 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 の「 PersistentVolumeClaim 」オブジェクトは、 Kubernetes クラスタユーザが作成した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-SAN 、 eseries-iscsi 、 ONTAP-SAN-エコノミー の 2 つのシステムがあります
trident.netapp.io/cloneFromPVC	cloneSourceVolume の実行中です	ONTAPNAS 、 ONTAPSAN 、 solidfire-san-SAN 、 aws -cvs 、 azure-netapp-files 、 GCP-cvs 、 ONTAP-SAN エコノミー

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/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 、 AWS-CVS 、 GCP-cvs
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「削除」再利用ポリシーがある場合、 Trident は PV が解放されると（つまり、ユーザーが PVC を削除したときに）、 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「 + Retain + 」ポリシーを使用している場合、 Trident はそのポリシーを無視し、管理者が Kubernetes とバックエンドからクリーンアップすると想定します。これにより、ボリュームを削除する前にバックアップまたは検査を行うことができます。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 (tridentctl') を使用して削除してください。

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。「 +On-Demand Volume Snapshots + 」を見て、これがどのように機能するかを確認してください。

Trident には ' クローン作成用の cloneFromPVC' および 'plitOnClone' 注釈も用意されていますCSI 実装（ Kubernetes 1.13 以前）を使用しなくても、または Kubernetes リリースがベータ版のボリュームスナップショット（ Kubernetes 1.16 以前）をサポートしていない場合は、これらのアノテーションを使用して PVC のクローンを作成できます。Trident 19.10 は、 PVC からのクローニングの CSI ワークフローをサポートしていることに注意してください。

> CSI Trident では 'cloneFromPVC' および 'plitOnClone' 注釈を使用できますまた ' 従来の CSI 以外のフロントエンドも使用できます

例 : ユーザがすでに「 m ysql 」という PVC を持っている場合、ユーザは「 trident.netapp.io/cloneFromPVC: mysql 」などの注釈を使用して「 m ysqlclone 」という新しい PVC を作成できます。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。
- また 'ONTAP-NAS' および 'ONTAP-SAN' ドライバを使用すると 'pvc 注釈 trident.netapp.io/splitOnClone` を trident.netapp.io/cloneFromPVC` と組み合わせて設定することが望ましい場合がありますTrident は 'trident.netapp.io/splitOnClone` を true に設定した場合 ' 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 ' 一部のストレージ効率を失うことなく ' クローン・ボリュームのライフサイクルを親ボリュームから完全に分離しますtrident.netapp.io/splitOnClone` を設定したり 'false に設定したりしないと ' 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代わりに ' バックエンドでのスペース消費が削減されますこれにより ' クローンを最初に削除しない限り ' 親ボリュームを削除できなくなります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' 」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完全な概要については、 Trident ボリュームオブジェクトを参照してください。

Kubernetes `PersistentVolume` オブジェクト

Kubernetes の 'PersistentVolume' オブジェクトは 'Kubernetes クラスタで利用できるようになったストレージの一部ですポッドに依存しないライフサイクルがあります。

> Trident は '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自動的に 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 をベースとする「 torageClass 」を参照する PVC を作成すると、 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 の「 torageClass 」オブジェクトは、「 PersistentVolumeClaims 」内の名前によって指定され、一連のプロパティを持つ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 の「 torageClass 」オブジェクトは次のようになります。

```yaml
apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
```

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	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	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

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
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 & eseries-iscsi ；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 、 E シリーズ - iSCSI 、 AWS- CVS 、GCP-cvs 、azure-NetApp-files 、 ONTAP-SAN-E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-san 、 vss-cvs 、gcp-cvs
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-san 、 vss-cvs 、gcp-cvs
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 ' 属性だけを使用して ' 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特性をモデル化できますTrident は ' 指定した属性の *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切なプールを自動的に選択できない場合は、「 toragePools 」および「 additionalStoragePools 」パラメータを使用してプールをさらに改良したり、特定のプールセットを選択したりできます。

'toragePools' パラメータを使用すると ' 指定した属性に一致するプールのセットをさらに制限できますつまり 'attributes' パラメータと 'toragePools' パラメータで指定されたプールの交点をプロビジョニングに使用し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

「 additionalStoragePools 」パラメータを使用すると、「 attributes 」パラメータと「 toragePools 」パラメータで選択され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できます。

excludeStoragePools' パラメータを使用して、 Trident がプロビジョニングに使用するプールのセットをフィルタリングできます。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。

'toragePools' パラメータと 'additionalStoragePools' パラメータでは ' 各エントリは '<backend>:<storagePoolList>' の形式で指定したバックエンドのストレージプールのカンマ区切りリストですたとえば、「 additionalStoragePools 」の値は「 ontapnas_192.168.1.100 ： aggr1 、 aggr2 ； solidfire_192.168.1.101 ： bronze 」のようになります。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。tridentctl get backend を使用してバックエンドとそのプールのリストを取得できます

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** のバージョン
FSstype （英語）	文字列	ext4 、 ext3 、 xfs など	ブロックボリュームのファイルシステムのタイプ	solidfire-san-エコノミー 、 eseries-iscsi	すべて

Trident インストーラバンドルには、「 `sample -input/storageclass-*.yaml 」で Trident で使用するストレージクラス定義の例がいくつか用意されています。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 'S torageClasses' に似ていますこの 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 'VolumeSnapshotClass' を管理者が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/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

「 driver 」は、「 csi-snapclass 」クラスのボリュームスナップショットの要求が Trident によって処理される Kubernetes を指定します。「要素ポリシー」は、スナップショットを削除する必要がある場合に実行されるアクションを指定します。「削除ポリシー」が「削除」に設定されている場合、 Snapshot を削除すると、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 は削除されます。または、「 Retain 」に設定すると、「 VolumeSnapshotContent 」と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 の VolumeSnapshot オブジェクトは ' ボリュームのスナップショット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。

ボリュームスナップショット要求が受信されると、 Trident はバックエンドでのボリュームのスナップショット作成を自動的に管理し、ユニークな「 VolumeSnapshotContent 」オブジェクトを作成することによってスナップショットを公開します。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 の「 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取得されたスナップショットを表します。これは「 PersistentVolume 」と似ており、ストレージ・クラスタ上でプロビジョニングされた Snapshot を表します。「 PersistentVolumeClaim 」および「 PersistentVolume 」オブジェクトと同様に、スナップショットが作成されると、「 VolumeContent Snapshot」 オブジェクトは「 VolumeSnapshot 」オブジェクトへの 1 対 1 のマッピングを保持します。これは、スナップショットの作成を要求しました。

> Trident は「 VolumeSnapshotContent 」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自動的に 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。

「 VolumeSnapshotContent 」オブジェクトには、スナップショットを一意に識別する詳細（「

napshotHandle 」など）が含まれています。この「 napshotHandle 」は、 PV の名前と「 VolumeSnapshotContent 」オブジェクトの名前を組み合わせた一意のものです。

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「 VolumeSnapshotContent 」オブジェクトを構成し、そのスナップショットを Kubernetes API に公開します。

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。これらのリソース定義を取得するには 'kubectl get CRDs' を実行します

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。

19.07 リリース以降、 Trident は多数の「 CustomResourceDefinition 」オブジェクトを使用して、 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 'tridentctl' を使用してバックエンドを作成すると ' 対応する tridentBackendsCRD オブジェクトが Kubernetes によって消費されるように作成されます

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 の以前のバージョンからアップグレードする場合（ステートを維持するために「 etcd」 を使用したもの）、 Trident インストーラは「 etcd' キーバリューデータストア」からデータを移行し、対応する CRD オブジェクトを作成します。
- tridentctl uninstall コマンドを使用して Trident をアンインストールすると、 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。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。

Trident `StorageClass` オブジェクト

Trident は 'Kubernetes のプロビジョニング・フィールドに csi.trident.netapp.io`/`netapp.io/trident` を指定する 'S torageClass' オブジェクトに一致するストレージ・クラスを作成しますストレージクラス名は、そのストレージクラスが表す Kubernetes の「 torageClass 」オブジェクトの名前と一致します。

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

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 の導入では、新しい Kubernetes の「 torageClass 」オブジェクトを登録するときに、これらのオブジェクトが作成されることを期待しています。

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 の「 torageClass 」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、バックエンド構成を参照してください。

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、 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 'PersistentVolumes' に直接対応し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	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 - * & AWS-cvs * ：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 は ' ボリュームの作成時に internalName を生成しますこの構成は 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「 trident 」またはバックエンド構成のプレフィックス）をボリューム名の前に付加し、「 <prefix> - <volume-name> 」という形式の名前を付けます。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 バックエンドでは、ハイフンをアンダースコアで置き換えます（つまり、内部名は「 <prefix>_<volume-name>` 」になります）。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。E シリーズでは、すべてのオブジェクト名に 30 文字の制限が適用されているため、 Trident は各ボリュームの内部名に対してランダムな文字列を生成します。CVS （ AWS ）では、一意のボリューム作成トークンに 16 ~ 36 文字の制限があり、 Trident は各ボリュームの内部名に対してランダムな文字列を生成します。

ボリューム設定を使用して、 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 の「 PersistentVolumeClaim 」メソッドを使用することを想定しています。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 'VolumeSnapshotContent' オブジェクトに直接対応します各 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。

個々の「スナップショット」オブジェクトには、以下のプロパティが含まれています。

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 の「 VolumeSnapshot 」オブジェクト要求が作成されると、 Trident は元のストレージシステム上にスナップショットオブジェクトを作成することによって動作します。このスナップショットオブジェクトの「 internalName 」は、プレフィックス「 snapshot-` と「 VolumeSnapshot 」オブジェクトの「 UID 」を組み合わせることによって生成されます（例：「 snapshot-e8d8d8a0ca-9826-11e9-9807-525400f3f660 」）。「 volumeName 」と「 volumeInternalName 」には、バッキングボリュームの詳細を取得して値を設定します。

tridentctl コマンドとオプション

。 "Trident インストーラバンドル" Astra Trident へのシンプルなアクセスを提供するコマンドラインユーティリティ「 tridentctl 」が含まれています。十分な権限を持つ Kubernetes ユーザは、このロールを使用して Astra Trident をインストールしたり、Astra Trident ポッドが含まれるネームスペースを直接管理したりできます。

使用方法については 'tridentctl --help' を実行してください

使用可能なコマンドとグローバルオプションは次のとおりです。

```
Usage:
  tridentctl [command]
```

使用可能なコマンド：

- `create` ： Astra Trident にリソースを追加します。
- 「削除」 : Astra Trident から 1 つ以上のリソースを削除します。
- 「 GET 」： Astra Trident から 1 つ以上のリソースを取得します。
- 「 help 」：任意のコマンドに関するヘルプ。
- `imag`: Astra Trident が必要とするコンテナイメージのテーブルを印刷しなさい。
- import` ：既存のリソースを Astra Trident にインポートします。
- install` ： Astra Trident をインストールします。
- logs` ： Astra Trident からログを印刷します。
- 'send': Astra Trident からリソースを送信します
- uninstall: Astra Trident をアンインストールします。
- `update` ： Astra Trident のリソースを変更します。
- アップグレード： Astra Trident のリソースをアップグレードします。
- 've rsion ： Astra Trident のバージョンを印刷します。

フラグ：

- `-d, --debug`: デバッグ出力。
- `-h, --help` ： tridentctl のヘルプ。
- `n,--namespace string`: Astra Trident 配備の名前空間。
- `-o ,--output string`: 出力形式。JSON の 1 つ | yaml | name | wide | ps （デフォルト）。
- `s'--server string`: Astra Trident REST インタフェースのアドレス / ポート。

create

create コマンドを実行して 'Astra Trident にリソースを追加できます

```
Usage:
  tridentctl create [option]
```

利用可能なオプション : backend`: Astra Trident にバックエンドを追加します。

delete

「削除」コマンドを実行すると、 Astra Trident から 1 つ以上のリソースを削除できます。

```
Usage:
  tridentctl delete [option]
```

使用可能なオプション：

- backend`: Astra Trident から 1 つ以上のストレージバックエンドを削除します。
- 'node`: Astra Trident から 1 つ以上の CSI ノードを削除します。
- 'Snapshot` ： Astra Trident から 1 つ以上のボリュームスナップショットを削除します。
- 'torageclas' ： Astra Trident から 1 つ以上のストレージクラスを削除します
- 'volume` ： Astra Trident から 1 つ以上のストレージボリュームを削除します

`get`

get コマンドを実行すると 'Astra Trident から 1 つ以上のリソースを取得できます

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

使用可能なオプション：

- backend`: Astra Trident から 1 つ以上のストレージバックエンドを取得します。
- 'Snapshot` ： Astra Trident から 1 つ以上のスナップショットを取得します。
- 「 torageclass 」： Astra Trident から 1 つ以上のストレージクラスを取得します。
- 'volume`: Astra Trident から 1 つ以上のボリュームを取得します

`images`

「 images 」フラグを実行すると、 Astra Trident が必要とするコンテナイメージのテーブルを印刷できます。

```
Usage:
  tridentctl images [flags]
```

flags:*`-h, --help`: イメージのヘルプ。*`-v ,--k8s-version string`: Kubernetes クラスタのセマンティックバージョン。

`import volume`

import volume コマンドを実行して、既存のボリュームを Astra Trident にインポートできます。

```
Usage:
  tridentctl import volume <backendName> <volumeName> [flags]
```

エイリアス : volume 'v

フラグ：

- `-f ,--filename string`: YAML または JSON PVC ファイルへのパス。
- `-h, --help` ：ボリュームのヘルプ。
- ``-- 管理なし `: PV/PVC のみを作成しますボリュームのライフサイクル管理を想定しないでください。

`install`

"install" フラグを実行して、 Astra Trident をインストールできます。

```
Usage:
  tridentctl install [flags]
```

フラグ：

- `--autosupport-image string` ： AutoSupport Telemetry のコンテナイメージ（デフォルトは「 NetApp/trident autosupport ： 20.07.0 」）。
- `-- autosupport - proxy string` : AutoSupport Telemetry を送信するためのプロキシのアドレス / ポート。
- `--csI`: CSI Trident をインストールします (Kubernetes 1.13 のみをオーバーライドしますが、機能ゲートが必要です) 。
- `--enable-node-prep : 必要なパッケージをノードにインストールしようとします
- ``--generate-custom-yaml `: インストールしないで YAML ファイルを生成します。
- `-h, --help`: インストールのヘルプ。
- `-- image-registry string : 内部イメージレジストリのアドレス / ポート。
- `--k8s-timeout duration`: すべての Kubernetes 操作のタイムアウト (デフォルトは 3m0s)
- `-- kubbelet-dir string`: kubelet の内部状態のホストの場所 (デフォルトは /var/lib/kubbelet") 。
- `--log-format string`: Astra Trident のログ形式 (text,JSON] (デフォルトは "text") 。
- `--pv string`: Astra Trident が使用するレガシー PV の名前は、存在しないことを確認します (デフォルトは "trident") 。
- `--pvc string`: Astra Trident が使用する従来の PVC の名前は、存在しないことを確認します (デフォルトは "trident") 。
- `--silence -autosupport` ： AutoSupport バンドルを自動的にネットアップに送信しないでください (デフォルトは true) 。
- `-- silent` : インストール中のほとんどの出力を無効にしなさい。

- `--trident-image string`: インストールする Astra Trident イメージ。
- `--use-custom-yaml `: setup ディレクトリに存在する既存の YAML ファイルを使用します。
- `--use-ipv6` ： Astra Trident のコミュニケーションに IPv6 を使用します。

logs

"logs" フラグを実行して、 Astra Trident からログを印刷することができます。

```
Usage:
  tridentctl logs [flags]
```

フラグ：

- `-a, --archive: 特に指定がない限り ' すべてのログを含むサポート・アーカイブを作成します
- `-h, --help`: ログのヘルプ。
- `-l, --log string`: アストラトライデントログを表示します。trident | auto | trident-operator | all （デフォルトは「 auto 」）のいずれかです。
- `-- ノード文字列 : ノードポッドログの収集元となる Kubernetes ノード名
- `-p, --previous `: 以前のコンテナインスタンスのログが存在する場合は、そのログを取得します。
- `-- sidecars` : sidecar コンテナのログを取得します

send

'end' コマンドを実行して 'Astra Trident からリソースを送信できます

```
Usage:
  tridentctl send [option]
```

利用可能なオプション： AutoSupport ： AutoSupport アーカイブをネットアップに送信します。

uninstall

uninstall フラグを実行して 'Astra Trident をアンインストールできます

```
Usage:
  tridentctl uninstall [flags]
```

flags:*`-h, --help`: アンインストールのヘルプ。*`--silent`: アンインストール中のほとんどの出力を無効にします。

update

「 update 」コマンドを実行して、 Astra Trident のリソースを変更できます。

```
Usage:
  tridentctl update [option]
```

利用可能なオプション : backend`: Astra Trident のバックエンドを更新します。

upgrade

'upgrade' コマンドを実行して 'Astra Trident のリソースをアップグレードできます

```
Usage:
tridentctl upgrade [option]
```

使用可能なオプション : volume 'NFS/iSCSI から CSI に 1 つ以上の永続ボリュームをアップグレードします

「バージョン」

「 rsion 」フラグを実行して、「 tridentctl 」のバージョンと Trident サービスを実行して印刷できます。

```
Usage:
  tridentctl version [flags]
```

flags:*`--client`: クライアントバージョンのみ (サーバは不要)*`-h, --help`: バージョンのヘルプ。