Tridentの管理と監視

Trident

NetApp
January 15, 2026

目次

Tridentの管理と監視

Tridentをアップグレード

Tridentをアップグレード

24.02 リリース以降、 Trident は4 か月ごとにリリースされ、毎年 3 つのメジャー リリースが提供されます。新しいリリースはそれぞれ以前のリリースに基づいて構築され、新しい機能、パフォーマンスの強化、バグ修正、改善が提供されます。 Tridentの新機能を活用できるよう、少なくとも年に 1 回はアップグレードすることをお勧めします。

アップグレード前の考慮事項

Tridentの最新リリースにアップグレードする場合は、次の点を考慮してください。

- 特定の Kubernetes クラスター内のすべての名前空間にわたって、 Tridentインスタンスが 1 つだけインストールされている必要があります。
- Trident 23.07 以降では、v1 ボリューム スナップショットが必須となり、アルファ スナップショットやベータ スナップショットはサポートされなくなりました。
- Google CloudのCloud Volumes Serviceを"CVSサービスタイプ"バックエンド構成を更新して、`standardsw`または `zoneredundantstandardsw`Trident 23.01 からアップグレードする場合のサービス レベル。更新に失敗した `serviceLevel`バックエンドでボリュームが失敗する可能性があります。参照 "CVS サービスタイプのサンプル" 詳細については。
- アップグレードする際には、 `parameter.fsType`で `StorageClasses`Tridentによって使用されます。削除して再作成できます `StorageClasses`既存のボリュームを中断することなく。
  - これは、強制するための要件です "セキュリティコンテキスト"SAN ボリューム用。
  - sample inputディレクトリには、https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-basic.yaml.templなどの例が含まれています。[`storage-class-basic.yaml.templ` ^]とリンク:https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-bronze-default.yaml[`storage-class-bronze-default.yaml` ^]。
  - 詳細については、"既知の問題" 。

ステップ**1:** バージョンを選択する

Tridentのバージョンは日付ベース `YY.MM`命名規則。「YY」は年の最後の 2 桁、「MM」は月です。ドットリリースは `YY.MM.X`規則では、「X」はパッチ レベルです。アップグレード元のバージョンに基づいて、アップグレード後のバージョンを選択します。

- インストールされているバージョンから 4 リリース以内の任意のターゲット リリースへの直接アップグレードを実行できます。たとえば、24.06 (または任意の 24.06 ドット リリース) から 25.06 に直接アップグレードできます。
- 4 リリース期間外のリリースからアップグレードする場合は、複数段階のアップグレードを実行します。アップグレード手順については、"以前のバージョン"アップグレード元のバージョンを、4 つのリリースの期間に収まる最新のリリースにアップグレードします。たとえば、23.07 を実行していて、25.06 にアップグレードする場合:

a. 23.07 から 24.06 への最初のアップグレード。

b. 次に、24.06 から 25.06 にアップグレードします。

> OpenShift Container Platform でTridentオペレーターを使用してアップグレードする場合は、Trident 21.01.1 以降にアップグレードする必要があります。 21.01.0 でリリースされたTridentオペレーターには既知の問題が含まれていますが、21.01.1 で修正されています。詳細については、 "GitHub上の問題の詳細" 。

ステップ**2:** 元のインストール方法を決定する

最初にTrident をインストールする際に使用したバージョンを確認するには:

1. 使用 `kubectl get pods -n trident`ポッドを検査します。
   - オペレーターポッドがない場合、Tridentは次のようにインストールされます。 `tridentctl` 。
   - オペレーター ポッドがある場合、 Trident は手動または Helm を使用してTridentオペレーターを使用してインストールされました。
2. オペレーターポッドがある場合は、 `kubectl describe torc` Trident がHelm を使用してインストールされているかどうかを判断します。
   - Helm ラベルがある場合、 Trident はHelm を使用してインストールされました。
   - Helm ラベルがない場合、 Trident はTridentオペレーターを使用して手動でインストールされています。

ステップ**3:** アップグレード方法を選択する

通常は、最初のインストールと同じ方法でアップグレードする必要がありますが、"インストール方法を切り替える" 。 Trident をアップグレードするには 2 つのオプションがあります。

- "Tridentオペレーターを使用してアップグレードする"

> 確認することをお勧めします"オペレーターのアップグレードワークフローを理解する"オペレータにアップグレードする前に。

*

オペレーターによるアップグレード

オペレーターのアップグレードワークフローを理解する

Tridentオペレーターを使用してTrident をアップグレードする前に、アップグレード中に発生するバックグラウンド プロセスを理解しておく必要があります。これには、ローリング アップデートを有効にするTridentコントローラー、コントローラー Pod とノード Pod、およびノード DaemonSet への変更が含まれます。

Tridentオペレーターのアップグレード処理

数ある中の一つ"Tridentオペレータを使用する利点"Trident をインストールおよびアップグレードすると、既存のマウントされたボリュームを中断することなく、 Tridentおよび Kubernetes オブジェクトが自動的に処理

されます。このようにして、Tridentはダウンタイムなしでアップグレードをサポートできます。"ローリングアップデート"。特に、 Tridentオペレーターは Kubernetes クラスターと通信して次の操作を行います。

- Tridentコントローラーのデプロイメントとノード DaemonSet を削除して再作成します。
- Tridentコントローラー ポッドとTridentノード ポッドを新しいバージョンに置き換えます。
  - 1 つのノードが更新されない場合でも、残りのノードの更新は妨げられません。
  - 実行中のTrident Node Pod を持つノードのみがボリュームをマウントできます。

Kubernetesクラスタ上のTridentアーキテクチャの詳細については、以下を参照してください。"Tridentアーキテクチャ"。

オペレーターのアップグレードワークフロー

Tridentオペレータを使用してアップグレードを開始すると、次のようになります。

1. トライデント**Trident**:
   a. 現在インストールされているTridentのバージョン (バージョン *n*) を検出します。
   b. CRD、RBAC、 Trident SVC を含むすべての Kubernetes オブジェクトを更新します。
   c. バージョン *n* のTridentコントローラーのデプロイメントを削除します。
   d. バージョン *n+1* のTridentコントローラーのデプロイメントを作成します。
2. **Kubernetes** は *n+1* 用のTridentコントローラー ポッドを作成します。
3. トライデント**Trident**:
   a. *n* のTrident Node DaemonSet を削除します。オペレーターはノード ポッドの終了を待機しません。
   b. *n+1* のTrident Node Daemonset を作成します。
4. **Kubernetes** は、 Trident Node Pod *n* を実行していないノードにTrident Node Pod を作成します。これにより、ノード上に任意のバージョンのTrident Node Pod が 1 つしか存在しないことが保証されます。

Trident Operator または **Helm** を使用して**Trident**インストールをアップグレードする

Tridentオペレーターを使用して、手動でも Helm を使用してもTrident をアップグレードできます。 Tridentオペレータのインストールから別のTridentオペレータのインストールにアップグレードしたり、 `tridentctl` Tridentオペレータ バージョンへのインストール。レビュー"アップグレード方法を選択してください"Tridentオペレータ インストールをアップグレードする前に。

手動インストールのアップグレード

クラスター スコープのTridentオペレーター インストールから別のクラスター スコープのTridentオペレーターインストールにアップグレードできます。すべてのTridentバージョンは、クラスター スコープの演算子を使用します。

名前空間スコープのオペレータを使用してインストールされたTrident （バージョン20.07から20.10）からアップグレードするには、"インストールされているバージョン"Tridentの。

タスク概要

Trident は、オペレーターをインストールし、Kubernetes バージョンに関連付けられたオブジェクトを作成するために使用できるバンドル ファイルを提供します。

- Kubernetes 1.24を実行しているクラスターの場合は、"バンドル_pre_1_25.yaml" 。
- Kubernetes 1.25以降を実行しているクラスターの場合は、"バンドルポスト1_25.yaml" 。

開始する前に

Kubernetesクラスタを実行していることを確認してください"サポートされているKubernetesバージョン"。

手順

1. Tridentのバージョンを確認してください:

```
./tridentctl -n trident version
```

2. 更新する `operator.yaml`、 `tridentorchestrator_cr.yaml` 、 そして `post_1_25_bundle.yaml`アップグレードするバージョン (例: 25.06) のレジストリとイメージ パス、および正しいシークレットを入力します。
3. 現在のTridentインスタンスのインストールに使用されたTridentオペレーターを削除します。たとえば、25.02 からアップグレードする場合は、次のコマンドを実行します。

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. 初期インストールをカスタマイズした場合 `TridentOrchestrator`属性を編集できます `TridentOrchestrator`インストールパラメータを変更するオブジェクト。これには、オフライン モード用のミラー化されたTridentおよび CSI イメージ レジストリを指定したり、デバッグ ログを有効にしたり、イメージ プル シークレットを指定したりするための変更が含まれる場合があります。
5. 環境に適したバンドルYAMLファイルを使用してTridentをインストールします。_<bundle.yaml>_は `bundle_pre_1_25.yaml`または `bundle_post_1_25.yaml`Kubernetes のバージョンに基づきます。たとえば、 Trident 25.06.0 をインストールする場合は、次のコマンドを実行します。

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. トライデント トルクを編集して、イメージ 25.06.0 を含めます。

Helmインストールのアップグレード

Trident Helm のインストールをアップグレードできます。

> TridentがインストールされているKubernetesクラスタを1.24から1.25以降にアップグレードする場合は、values.yamlを更新して設定する必要があります。 `excludePodSecurityPolicy`に`true`または追加 `--set excludePodSecurityPolicy=true`に `helm upgrade`クラスターをアップグレードする前にコマンドを実行する必要があります。

Trident helm をアップグレードせずに Kubernetes クラスターを 1.24 から 1.25 にアップグレード済みの場合、helm のアップグレードは失敗します。 helm のアップグレードを実行するには、前提条件として次の手順を実行してください。

1. helm-mapkubeapisプラグインをインストールする https://github.com/helm/helm-mapkubeapis。
2. Tridentがインストールされている名前空間で、 Tridentリリースのドライ ランを実行します。クリーンアップされるリソースがリストされます。

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. helm を使用して完全実行を実行し、クリーンアップを実行します。

```
helm mapkubeapis trident  --namespace trident
```

手順

1. もしあなたが"Helmを使用してTridentをインストールしました"、使用することができます `helm upgrade trident netapp-trident/trident-operator --version 100.2506.0`ワンステップでアップグレードできます。 Helm リポジトリを追加していない場合、またはアップグレードに使用できない場合は、次の手順を実行します。
   a. 最新のTridentリリースをダウンロードするには、"GitHubの_Assets_セクション" 。
   b. 使用 `helm upgrade`コマンドの場所 `trident-operator-25.06.0.tgz`アップグレードするバージョンを反映します。

```
helm upgrade <name> trident-operator-25.06.0.tgz
```

> 初期インストール時にカスタムオプションを設定する場合（ TridentおよびCSIイメージのプライベートミラーレジストリを指定するなど）、 `helm upgrade`コマンド使用 `--set`これらのオプションがアップグレード コマンドに含まれていることを確認してください。含まれていない場合、値はデフォルトにリセットされます。

2. 走る `helm list`チャートとアプリのバージョンが両方ともアップグレードされたことを確認します。走る `tridentctl logs`デバッグ メッセージを確認します。

アップグレード `tridentctl`Tridentオペレーターへのインストール

Tridentオペレーターの最新リリースにアップグレードするには、 `tridentctl`インストール。既存のバックエンドと PVC は自動的に利用できるようになります。

> インストール方法を切り替える前に、"インストール方法の変更" 。

手順

1. 最新のTridentリリースをダウンロードしてください。

```
# Download the release required [25.06.0]
mkdir 25.06.0
cd 25.06.0
wget
https://github.com/NetApp/trident/releases/download/v25.06.0/trident-
installer-25.06.0.tar.gz
tar -xf trident-installer-25.06.0.tar.gz
cd trident-installer
```

2. 作成する `tridentorchestrator`マニフェストからの CRD。

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. クラスタースコープのオペレーターを同じ名前空間にデプロイします。

```
kubectl create -f deploy/<bundle-name.yaml>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. 作成する `TridentOrchestrator`Trident をインストールするための CR。

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. Trident が意図したバージョンにアップグレードされたことを確認します。

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v25.06.0
```

tridentctl によるアップグレード

既存のTridentインストールを簡単にアップグレードできます。 `tridentctl` 。

タスク概要

Trident をアンインストールして再インストールすると、アップグレードとして機能します。 Trident をアンインストールしても、 Tridentデプロイメントで使用される永続ボリューム要求 (PVC) と永続ボリューム (PV) は削除されません。すでにプロビジョニングされている PV は、 Tridentがオフラインの間も引き続き使用可能であり、 Trident はオンラインに戻った後、その間に作成された PVC のボリュームをプロビジョニングします。

開始する前に

レビュー"アップグレード方法を選択してください"アップグレードする前に `tridentctl`。

手順

1. アンインストールコマンドを実行する `tridentctl`CRD と関連オブジェクトを除く、 Tridentに関連付けられたすべてのリソースを削除します。

```
./tridentctl uninstall -n <namespace>
```

2. Trident を再インストールします。参照"tridentctlを使用してTridentをインストールする"。

> アップグレード プロセスを中断しないでください。インストーラが完了するまで実行されたことを確認します。

tridentctl を使用してTrident を管理する

その "Tridentインストーラーバンドル"含まれるもの `tridentctl`Tridentへの簡単なアクセスを提供するコマンドライン ユーティリティ。十分な権限を持つ Kubernetes ユーザーは、これを使用してTrident をインストールしたり、 Tridentポッドを含む名前空間を管理したりできます。

コマンドとグローバルフラグ

走れる `tridentctl help`利用可能なコマンドのリストを取得するには `tridentctl`または追加する `--help`任意のコマンドにフラグを付けると、その特定のコマンドのオプションとフラグのリストが取得されます。

```
tridentctl [command] [--optional-flag]
```

Trident `tridentctl`このユーティリティは、次のコマンドとグローバル フラグをサポートしています。

コマンド

create
Tridentにリソースを追加します。

delete
Tridentから 1 つ以上のリソースを削除します。

get
Tridentから 1 つ以上のリソースを取得します。

help
あらゆるコマンドに関するヘルプ。

images
Tridentに必要なコンテナ イメージの表を印刷します。

import
既存のリソースをTridentにインポートします。

install
Tridentをインストールします。

logs
Tridentからログを印刷します。

send
Tridentからリソースを送信します。

uninstall
Tridentをアンインストールします。

update
Trident内のリソースを変更します。

update backend state
バックエンド操作を一時的に停止します。

upgrade
Trident内のリソースをアップグレードします。

version
Tridentのバージョンを出力します。

グローバルフラグ

-d、 --debug

デバッグ出力。

-h、 --help

ヘルプ `tridentctl`。

-k、 --kubeconfig string

指定する `KUBECONFIG`コマンドをローカルで実行したり、ある Kubernetes クラスターから別のクラスターに実行したりするためのパス。

あるいは、 `KUBECONFIG`特定のKubernetesクラスタと問題を指す変数 `tridentctl`そのクラスターにコマンドを送信します。

-n、 --namespace string

Tridentデプロイメントの名前空間。

-o、 --output string

出力形式。 json|yaml|name|wide|ps のいずれか (デフォルト)。

-s、 --server string

Trident REST インターフェースのアドレス/ポート。

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) でのみリッスンおよびサービスを提供するように構成できます。

コマンドオプションとフラグ

作成する

使用 `create`Tridentにリソースを追加するコマンド。

```
tridentctl create [option]
```

オプション

`backend`: Tridentにバックエンドを追加します。

消去

使用 `delete`Tridentから 1 つ以上のリソースを削除するコマンド。

```
tridentctl delete [option]
```

オプション

`backend`: 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。
`snapshot` : 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。

`storageclass` : Tridentから 1 つ以上のストレージ クラスを削除します。
`volume` : Tridentから 1 つ以上のストレージ ボリュームを削除します。

得る

使用 `get`Tridentから 1 つ以上のリソースを取得するコマンド。

```
tridentctl get [option]
```

オプション

`backend`: 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` : Tridentから 1 つ以上のスナップショットを取得します。
`storageclass` : Tridentから 1 つ以上のストレージ クラスを取得します。
`volume` : Tridentから 1 つ以上のボリュームを取得します。

フラグ

`-h、 --help` : ボリュームのヘルプ。
`--parentOfSubordinate string` : クエリを従属ソース ボリュームに制限します。
`--subordinateOf string` : ボリュームの下位にクエリを制限します。

画像

使用 `images`Tridentに必要なコンテナ イメージのテーブルを印刷するためのフラグ。

```
tridentctl images [flags]
```

フラグ

`-h、 --help` : 画像のヘルプ。
`-v 、 --k8s-version string` : Kubernetes クラスターのセマンティック バージョン。

輸入量

使用 `import volume`既存のボリュームをTridentにインポートするコマンド。

```
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` : 音量のヘルプ。
`--no-manage` : 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。

インストール

使用 `install`Trident をインストールするためのフラグ。

```
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:<current-version>」)。
--autosupport-proxy string : Autosupport テレメトリを送信するためのプロキシのアドレス/ポート。
--enable-node-prep : ノードに必要なパッケージをインストールしようとします。
--generate-custom-yaml : 何もインストールせずに YAML ファイルを生成します。
-h 、 --help : インストールのヘルプ。
--http-request-timeout : Tridentコントローラーの REST API の HTTP リクエスト タイムアウトをオーバーライドします (デフォルトは 1 分 30 秒)。
--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 3 分 0 秒)。
--kubelet-dir string : kubelet の内部状態のホストの場所 (デフォルトは "/var/lib/kubelet")。
--log-format string : Trident のログ形式 (テキスト、json) (デフォルトは「text」)。
--node-prep : Trident が指定されたデータ ストレージ プロトコルを使用してボリュームを管理するために Kubernetes クラスターのノードを準備できるようにします。 現在、 **iscsi**`サポートされている唯一の値です。 **OpenShift 4.19** 以降、この機能でサポートされる**Trident** の最小バージョンは **25.06.1** です。
`**--pv string** : Tridentによって使用されるレガシー PV の名前。これが存在しないことを確認します (デフォルトは「trident」)。
--pvc string : Tridentによって使用されるレガシー PVC の名前。これが存在しないことを確認します (デフォルトは「trident」)。
--silence-autosupport : 自動サポート バンドルをNetAppに自動的に送信しません (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ます。
--trident-image string : インストールするTridentイメージ。
--k8s-api-qps : Kubernetes API リクエストの 1 秒あたりのクエリ数 (QPS) の制限 (デフォルト 100、オプション)。
--use-custom-yaml : セットアップ 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 : Trident の通信には IPv6 を使用します。

ログ

使用 `logs`Tridentからのログを印刷するためのフラグ。

```
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 : 表示するTridentログ。 trident|auto|trident-operator|all のいずれか (デフォルトは "auto")。
--node string : ノード ポッド ログを収集する 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取得します。
--sidecars : サイドカー コンテナのログを取得します。

送信

使用 `send`Tridentからリソースを送信するコマンド。

```
tridentctl send [option]
```

オプション

`autosupport`: Autosupport アーカイブをNetAppに送信します。

uninstall

使用 `uninstall`Trident をアンインストールするためのフラグ。

```
tridentctl uninstall [flags]
```

フラグ

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

更新

使用 `update`Trident内のリソースを変更するコマンド。

```
tridentctl update [option]
```

オプション

`backend`: Tridentのバックエンドを更新します。

バックエンドの状態を更新する

使用 `update backend state`バックエンド操作を一時停止または再開するコマンド。

```
tridentctl update backend state <backend-name> [flag]
```

考慮すべき点

- バックエンドがTridentBackendConfig（tbc）を使用して作成された場合、バックエンドは `backend.json` ファイル。
- もし `userState`tbcに設定されている場合、 `tridentctl update backend state <backend-name> --user-state suspended/normal`指示。
- 設定する能力を取り戻すには `userState`tbc経由で設定した後、tridentctl経由で `userState`フィールドを tbc から削除する必要があります。これは、 `kubectl edit tbc`指示。その後 `userState`フィールドが削除された場合は、 `tridentctl update backend state`変更するコマンド `userState`バックエンドの。
- 使用 `tridentctl update backend state`変更するには `userState`。更新することもできます `userState`使用して `TridentBackendConfig`または `backend.json`ファイル; これにより、バックエンドの完全な再初期化がトリガーされ、時間がかかる場合があります。

  フラグ

  `-h、 --help` : バックエンドの状態に関するヘルプ。
  `--user-state` : に設定 `suspended`バックエンド操作を一時停止します。設定 `normal`バックエンド操作を再開します。に設定すると `suspended`:

- `AddVolume`そして `Import Volume`一時停止されます。
- `CloneVolume`、 `ResizeVolume` 、 `PublishVolume` 、 `UnPublishVolume` 、 `CreateSnapshot` 、 `GetSnapshot` 、 `RestoreSnapshot` 、 `DeleteSnapshot` 、 `RemoveVolume` 、

`GetVolumeExternal`、`ReconcileNodeAccess`引き続きご利用いただけます。

バックエンドの状態を更新することもできます。`userState`バックエンド構成ファイルのフィールド`TridentBackendConfig`または`backend.json`。詳細については、"バックエンドを管理するためのオプション"そして"kubectl を使用してバックエンド管理を実行する"。

例：

JSON

以下の手順に従って更新してください `userState`使用して `backend.json`ファイル：

1. 編集する `backend.json`ファイルに `userState`値が「suspended」に設定されたフィールド。
2. バックエンドを更新するには、 `tridentctl update backend`コマンドと更新されたパス `backend.json`ファイル。

   例：`tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

ヤムル

適用後にTBCを編集するには、 `kubectl edit <tbc-name> -n <namespace>`指示。次の例では、バックエンドの状態をsuspendに更新します。 `userState: suspended`オプション：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

version

使用 `version`バージョンを印刷するためのフラグ `tridentctl`そして実行中のTridentサービス。

```
tridentctl version [flags]
```

フラグ

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

プラグインのサポート

Tridentctl は kubectl と同様のプラグインをサポートしています。 Tridentctl は、プラグインのバイナリ ファイル名が「tridentctl-<plugin>」というスキームに従っており、バイナリが PATH 環境変数にリストされているフォルダーにある場合に、プラグインを検出します。検出されたすべてのプラグインは、tridentctl ヘルプのプラグイン セクションにリストされます。オプションとして、環境変数 TRIDENTCTL_PLUGIN_PATH でプラグインフォルダを指定して検索範囲を制限することもできます (例: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/` ）。変数を使用すると、tridenctl は指定されたフォルダー内でのみ検索します。

モニターTrident

Trident は、 Trident のパフォーマンスを監視するために使用できる Prometheus メトリック エンドポイントのセットを提供します。

概要

Tridentが提供するメトリックを使用すると、次のことが可能になります。

- Trident の状態と構成を監視します。操作がどの程度成功したか、期待どおりにバックエンドと通信できるかどうかを調べることができます。
- バックエンドの使用状況情報を調べて、バックエンドにプロビジョニングされているボリュームの数や消費されているスペースの量などを把握します。
- 利用可能なバックエンドにプロビジョニングされたボリュームの量のマッピングを維持します。
- パフォーマンスを追跡します。 Tridentがバックエンドと通信して操作を実行するのにかかる時間を確認できます。

> デフォルトでは、Tridentのメトリクスはターゲットポートに公開されます。 `8001`で`/metrics`終点。これらのメトリックは、 Tridentがインストールされると デフォルトで有効 になります。

要件

- Tridentがインストールされた Kubernetes クラスター。
- Prometheus インスタンス。これは、 "コンテナ化されたPrometheusのデプロイメント"またはPrometheusを "ネイティブアプリケーション"。

ステップ1: Prometheusターゲットを定義する

メトリックを収集し、 Trident が管理するバックエンドや作成するボリュームなどの情報を取得するには、Prometheus ターゲットを定義する必要があります。これ "ブログ"Prometheus と Grafana をTridentと組み合わせて使用してメトリックを取得する方法について説明します。このブログでは、Kubernetes クラスタ

ーで Prometheus をオペレーターとして実行する方法と、 Tridentメトリックを取得するための ServiceMonitor を作成する方法について説明します。

ステップ2: Prometheus ServiceMonitorを作成する

Tridentのメトリクスを利用するには、監視するPrometheus ServiceMonitorを作成する必要があります。 `trident-csi`サービスと `metrics`ポート。サンプルの ServiceMonitor は次のようになります。

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

このServiceMonitor定義は、 `trident-csi`サービスと特に `metrics`サービスのエンドポイント。その結果、Prometheus は Trident のメトリックを理解するように設定されるようになりました。

Tridentから直接入手できるメトリクスに加えて、kubeletは多くの `kubelet_volume_*`独自のメトリック エンドポイントを介してメトリックを取得します。 Kubelet は、接続されているボリューム、およびそれが処理するポッドやその他の内部操作に関する情報を提供できます。参照 "ここをクリックしてください。"。

ステップ3: PromQLでTridentメトリクスをクエリする

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

使用できる PromQL クエリをいくつか示します。

Tridentの健康情報を入手する

- **Trident**からの **HTTP 2XX** 応答の割合

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

- ステータス コード経由の**Trident**からの **REST** 応答の割合

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

- **Trident**によって実行された操作の平均所要時間（ミリ秒）

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

Tridentの使用情報を取得する

- 平均ボリュームサイズ

```
trident_volume_allocated_bytes/trident_volume_count
```

- 各バックエンドによってプロビジョニングされたボリュームスペースの合計

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

個々のボリューム使用量を取得する

これは、kubelet メトリックも収集される場合にのみ有効になります。

- 各ボリュームの使用済みスペースの割合

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Trident AutoSupportテレメトリについて学ぶ

デフォルトでは、 Trident はPrometheus メトリックと基本的なバックエンド情報を毎日NetAppに送信します。

- TridentがPrometheusメトリックと基本的なバックエンド情報をNetAppに送信するのを止めるには、 `--silence-autosupport` Trident のインストール中にフラグを設定します。
- TridentはコンテナログをNetAppサポートにオンデマンドで送信することもできます。 `tridentctl send autosupport` 。ログをアップロードするには、 Tridentをトリガーする必要があります。ログを提出する前に、NetAppのhttps://www.netapp.com/company/legal/privacy-policy/["プライバシーポリシー"^]。

- 指定されない限り、 Trident は過去 24 時間のログを取得します。
- ログの保存期間を指定するには、 `--since`フラグ。例えば： `tridentctl send autosupport --since=1h`。この情報は、 `trident-autosupport` Tridentと一緒にインストールされるコンテナ。コンテナイメージは次の場所から入手できます。 "TridentAutoSupport" 。
- Trident AutoSupport は、個人を特定できる情報 (PII) または個人情報を収集または送信しません。付属の "EULA"これはTridentコンテナ イメージ自体には適用されません。 NetAppのデータセキュリティと信頼性への取り組みについて詳しくは、 "ここをクリックしてください。" 。

Tridentによって送信されるペイロードの例は次のようになります。

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

- AutoSupportメッセージは、NetApp のAutoSupportエンドポイントに送信されます。コンテナイメージを保存するためにプライベートレジストリを使用している場合は、 `--image-registry`フラグ。
- インストール YAML ファイルを生成してプロキシ URL を構成することもできます。これは、 `tridentctl install --generate-custom-yaml` YAMLファイルを作成し、 `--proxy-url`の議論 `trident-autosupport`コンテナ内 `trident-deployment.yaml`。

Tridentメトリックを無効にする

メトリクスの報告を無効にするには、カスタムYAMLを生成する必要があります（ `--generate-custom-yaml`フラグ）を編集して削除します `--metrics`フラグが呼び出されないようにする `trident-main`容器。

Tridentをアンインストールする

Trident をアンインストールするには、 Tridentをインストールしたときと同じ方法を使用する必要があります。

タスク概要

- アップグレード後に発見されたバグ、依存関係の問題、またはアップグレードの失敗や不完全なアップグレードの修正が必要な場合は、 Tridentをアンインストールし、そのバージョンの特定の手順に従って以前

のバージョンを再インストールする必要があります。"version" 。これは、以前のバージョンにダウングレードする場合に推奨される唯一の方法です。

- アップグレードと再インストールを容易にするために、 Tridentをアンインストールしても、 Tridentによって作成された CRD または関連オブジェクトは削除されません。 Tridentとそのすべてのデータを完全に削除する必要がある場合は、"TridentとCRDを完全に削除する" 。

開始する前に

Kubernetes クラスターを廃止する場合は、アンインストールする前に、 Tridentによって作成されたボリュームを使用するすべてのアプリケーションを削除する必要があります。これにより、PVC が削除される前に Kubernetes ノードで未公開になることが保証されます。

元のインストール方法を決定する

Trident をアンインストールするには、インストールに使用したのと同じ方法を使用する必要があります。アンインストールする前に、最初にTrident をインストールしたときに使用したバージョンを確認してください。

1. 使用 `kubectl get pods -n trident`ポッドを検査します。
   - オペレーターポッドがない場合、Tridentは次のようにインストールされます。 `tridentctl` 。
   - オペレーター ポッドがある場合、 Trident は手動または Helm を使用してTridentオペレーターを使用してインストールされました。
2. オペレーターポッドがある場合は、 `kubectl describe tproc trident` Trident がHelm を使用してインストールされているかどうかを判断します。
   - Helm ラベルがある場合、 Trident はHelm を使用してインストールされました。
   - Helm ラベルがない場合、 Trident はTridentオペレーターを使用して手動でインストールされています。

Tridentオペレータのインストールをアンインストールする

Trident Operator のインストールは手動でアンインストールすることも、Helm を使用してアンインストールすることもできます。

手動インストールをアンインストールする

オペレータを使用してTrident をインストールした場合は、次のいずれかの方法でアンインストールできます。

1. 編集 **`TridentOrchestrator`CR** してアンインストール フラグを設定します:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"uninstall":true}}'
```

   いつ `uninstall`フラグが設定されている `true`、 TridentオペレーターはTridentをアンインストールしますが、TridentOrchestrator 自体は削除しません。 Trident を再度インストールする場合は、TridentOrchestrator をクリーンアップして新しいものを作成する必要があります。

2. 消去 **`TridentOrchestrator`**: 削除することにより `TridentOrchestrator`Trident を展開するために使用さ

れた CR の場合、オペレーターにTrident をアンインストールするように指示します。オペレーターは、`TridentOrchestrator`そして、 Tridentデプロイメントとデーモンセットを削除し、インストールの一環として作成されたTridentポッドを削除します。

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

Helmのインストールをアンインストールする

Helmを使用してTridentをインストールした場合は、以下を使用してアンインストールできます。 `helm uninstall` 。

```
#List the Helm release corresponding to the Trident install.
helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

アンインストール `tridentctl`インストール

使用 `uninstall`コマンドイン `tridentctl`CRD と関連オブジェクトを除く、 Tridentに関連付けられたすべてのリソースを削除します。

```
./tridentctl uninstall -n <namespace>
```

著作権に関する情報

商標に関する情報