Tridentの管理と監視

Trident

NetApp
July 01, 2026

目次

Tridentの管理と監視 1
- Tridentのアップグレード 1
  - Tridentのアップグレード 1
  - オペレータを使用したアップグレード 2
  - tridentctl によるアップグレード 7
- tridentctlを使用してTridentを管理する 8
  - コマンドとグローバルフラグ 8
  - コマンドオプションとフラグ 10
  - プラグインのサポート 16
- Tridentを監視 16
  - 概要 16
  - ステップ1：Prometheusターゲットを定義する 17
  - ステップ2：Prometheus ServiceMonitorを作成する 17
  - ステップ3：PromQLを使用したTridentメトリクスのクエリ 19
  - Trident AutoSupportテレメトリーの詳細 20
  - Tridentメトリクスを無効にする 21
- Tridentのアンインストール 21
  - 元のインストール方法を決定する 21
  - Tridentオペレータのインストールをアンインストールする 21
  - `tridentctl`インストールをアンインストールします 22

Tridentの管理と監視

Tridentのアップグレード

Tridentのアップグレード

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 の新機能をご利用いただくために、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。
- アップグレード時には、Tridentで使用されている `parameter.fsType`に `StorageClasses`を必ず指定することが重要です。既存のボリュームに影響を与えることなく、 `StorageClasses`を削除して再作成できます。
  - これは、SAN ボリュームに "セキュリティコンテキスト"を適用するための要件です。
  - 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[`storage-class-basic.yaml.templ`]や`storage-class-bronze-default.yaml`などの例が含まれています。
  - 詳細については、"既知の問題"を参照してください。

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の `YY.MM`命名規則に従います。「YY」は年の下2桁、「MM」は月を表します。ドットリリースは `YY.MM.X`命名規則に従い、「X」はパッチレベルを表します。アップグレード元のバージョンに基づいて、アップグレード先のバージョンを選択してください。

- 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つの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、複数ステップのアップグレードを実行してください。"以前のバージョン"からアップグレードする場合は、そのバージョンのアップグレード手順を使用して、4リリース期間に収まる最新のリリースにアップグレードします。例えば、23.07 を実行していて、25.06 にアップグレードしたい場合（：）
  a. 最初に 23.07 から 24.06 にアップグレードします。
  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の問題の詳細"を参照してください。

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. `kubectl get pods -n trident`を使用して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 `tridentctl`を使用してインストールされました。
   - オペレーターポッドがある場合、Trident は Trident オペレーターを使用して手動または Helm を使用してインストールされました。
2. オペレータポッドがある場合は、 `kubectl describe torc`を使用して、TridentがHelmを使用してインストールされたかどうかを確認します。
   - 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ました。

ステップ**3**：アップグレード方法を選択する

一般的には、最初のインストールに使用したの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オペレータを使用してアップグレードする"

オペレータでアップグレードする前に"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"を確認することをお勧めします。

*

オペレータを使用したアップグレード

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 プロセスを理解する必要があります。これには、Tridentコントローラー、コントローラーポッドとノードポッド、およびローリングアップデートを可能にするノードDaemonSetへの変更が含まれます。

Tridentオペレータのアップグレード処理

Trident のインストールとアップグレードの多くの"Tridentオペレーターを使用する利点"の 1 つは、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 と Kubernetes オブジェクトを自動的に処理することです。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。または"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以下を実行します：

- Trident コントローラの導入とノード DaemonSet を削除して再作成します。

- Trident Controller PodとTrident Node Pod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、"Tridentアーキテクチャ"を参照してください。

オペレータ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：

1. **Trident** オペレータ：
   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 Controller デプロイメントを削除します。
   d. バージョン *n+1* の Trident Controller デプロイメントを作成します。
2. **Kubernetes** は *n+1* 用の Trident Controller Pod を作成します。
3. **Trident** オペレータ：
   a. *n* の Trident ノード DaemonSet を削除します。オペレーターはノード Pod 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 が複数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、手動または Helm を使用して Trident をアップグレードできます。Trident オペレーターのインストールから別の Trident オペレーターのインストールにアップグレードすることも、 `tridentctl`のインストールから Trident オペレーターバージョンにアップグレードすることもできます。Trident オペレーターのインストールをアップグレードする前に"アップグレード方法を選択"を確認してください。

手動インストールのアップグレード

クラスタを対象とした Trident オペレータのインストールから、別のクラスタを対象とした Trident オペレータのインストールにアップグレードできます。すべての Trident バージョンでは、クラスタを対象としたオペレータを使用します。

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"のTridentのアップグレード手順を使用してください。

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 1.25以降を実行しているクラスターの場合は、"bundle_post_1_25.yaml"を使用します。

開始する前に

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

手順

1. Trident のバージョンを確認してください：

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

2. `operator.yaml`、`tridentorchestrator_cr.yaml`、`post_1_25_bundle.yaml`を、アップグレード先のバージョン（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で更新します。
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 `TridentOrchestrator`属性を使用してカスタマイズした場合は、`TridentOrchestrator`オブジェクトを編集してインストールパラメータを変更できます。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の指定、デバッグログの有効化、またはイメージプルシークレットの指定など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. Trident torc 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、クラスタをアップグレードする前に、values.yamlを更新して`excludePodSecurityPolicy`を `true`に設定するか、 `--set excludePodSecurityPolicy=true`を`helm upgrade`コマンドに追加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. https://github.com/helm/helm-mapkubeapis からhelm-mapkubeapisプラグインをインストールします。
2. Trident がインストールされているネームスペースで、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一覧表示されます。

```
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. "Helmを使用してTridentをインストールしました"の場合は、 `helm upgrade trident netapp-trident/trident-operator --version 100.2602.0`を使用して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：
   a. 最新のTridentリリースを"GitHub の *Assets* セクション"からダウンロードします。
   b. `helm upgrade`コマンドを使用します。 `trident-operator-26.02.0.tgz`にはアップグレード先のバージョンを指定します。

```
helm upgrade <name> trident-operator-26.02.0.tgz
```

> ⓘ 初期インストール時にカスタムオプション（TridentおよびCSIイメージ用のプライベートミラーリポジトリの指定など）を設定した場合は、それらのオプションがアップグレードコマンドに含まれるように、 `helm upgrade`コマンドに `--set`を追加してください。そうしないと、値がデフォルトにリセットされます。

2. `helm list`を実行して、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。 `tridentctl logs`を実行して、デバッグメッセージを確認します。

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

Tridentオペレーターの最新リリースには、 `tridentctl`インストールからアップグレードできます。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、"インストール方法間の移行"を確認してください。

手順

1. 最新の Trident リリースをダウンロードします。

```
# Download the release required [26.02.0]
mkdir 26.02.0
cd 26.02.0
wget
https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

2. マニフェストから `tridentorchestrator` CRD を作成します。

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

3. クラス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. Trident をインストールするための `TridentOrchestrator` CR を作成します。

```
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:                v26.02.0
```

tridentctl によるアップグレード

既存のTridentインストールは `tridentctl`を使用して簡単にアップグレードできます。

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すべての 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

"アップグレード方法を選択"を使用してアップグレードする前に `tridentctl`を確認してください。

手順

1. `tridentctl`でアンインストールコマンドを実行して、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。"tridentctl を使用して Trident をインストールする"を参照してください。

> 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まで実行されることを確認してください。

tridentctlを使用してTridentを管理する

`https://github.com/NetApp/trident/releases["Trident`インストーラバンドル"^]には、`Trident`への簡単なアクセスを提供する `tridentctl`コマンドラインユーティリティが含まれています。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`ポッドを含むネームスペースを管理したりできます。

コマンドとグローバルフラグ

`tridentctl help`を実行すると、 `tridentctl`で使用可能なコマンドのリストを取得できます。または、任意のコマンドに `--help`フラグを追加すると、その特定のコマンドのオプションとフラグのリストを取得できます。

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

Trident `tridentctl`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
1つ以上のリソースをTridentから削除します。

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で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でリソースをアップグレードします。

version
Tridentのバージョンを印刷します。

グローバルフラグ

-d, --debug

デバッグ出力。

-h, --help

`tridentctl`のヘルプ。

-k, --kubeconfig string

`KUBECONFIG`パスを指定して、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します。

あるいは、 `KUBECONFIG`変数をエクスポートして特定のKubernetesクラスタを指定し、そのクラスタに `tridentctl`コマンドを発行できます。

-n, --namespace string

Trident 導入の名前空間。

-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

`create`コマンドを使用して、Tridentにリソースを追加します。

```
tridentctl create [option]
```

オプション

backend：Trident にバックエンドを追加します。

削除

`delete`コマンドを使用して、Tridentから1つ以上のリソースを削除します。

```
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Snapshot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。

取得

`get`コマンドを使用して、Tridentから1つ以上のリソースを取得します。

```
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

`images`フラグを使用して、Tridentに必要なコンテナイメージのテーブルを出力します。

```
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。

ボリュームのインポート

`import volume`コマンドを使用して、既存のボリュームをTridentにインポートします。

```
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。

`--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

`install`フラグを使用してTridentをインストールします。

```
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 Telemetry 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 リクエストのタイムアウトをオーバーライドします（デフォルトは 1m30s）。
`--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）。
`--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（text、json）（デフォルトは "text"）。
`--node-prep`：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 クラスタのノードを準備できるように Trident を有効にします。現在、 **`iscsi`**が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ている `Trident` の最小バージョンは `25.06.1` です。**
`--pv string`：Trident が使用するレガシー 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 が使用するレガシー 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`：autosupport 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（デフォルトは 100、オプション）。
`--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`：Trident の通信に IPv6 を使用します。

ログ

`logs`フラグを使用して、Tridentからログを出力します。

```
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

`send`コマンドを使用して、Tridentからリソースを送信します。

```
tridentctl send [option]
```

オプション

`autosupport`：AutosupportアーカイブをNetAppに送信する。

uninstall

`uninstall`フラグを使用してTridentをアンインストールします。

```
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

`update`コマンドを使用して、Tridentのリソースを変更します。

```
tridentctl update [option]
```

オプション

`backend`：Trident で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

`update backend state`コマンドを使用して、バックエンド操作を一時停止または再開します。

```
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ている場合、そのバックエンドは `backend.json`ファイルを使用して更新できません。

- `userState`がtbcに設定されている場合、 `tridentctl update backend state <backend-name> --user-state suspended/normal`コマンドを使用して変更することはできません。
- tbc 経由で設定した後に tridentctl 経由で `userState`を設定する機能を取り戻すには、 `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`コマンドを使用して実行できます。 `userState`フィールドが削除されたら、 `tridentctl update backend state`コマンドを使用してバックエンドの `userState`を変更できます。
- `tridentctl update backend state`を使用して `userState`を変更します。また、 `userState`は `TridentBackendConfig`または `backend.json`ファイルを使って更新することもできます。この操作はバックエンドの完全な再初期化を引き起こし、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`： `suspended`に設定すると、バックエンド操作が一時停止されます。 `normal`に設定すると、バックエンド操作が再開されます。 `suspended`に設定した場合：

- `AddVolume`と `Import Volume`は一時停止されます。
- `CloneVolume`、 `ResizeVolume`、 `PublishVolume`、 `UnPublishVolume`、 `CreateSnapshot`、 `GetSnapshot`、 `RestoreSnapshot`、 `DeleteSnapshot`、 `RemoveVolume`、 `GetVolumeExternal`、 `ReconcileNodeAccess`は引き続き使用できます。

> `userState`フィールドを使用して、バックエンド構成ファイル `TridentBackendConfig`または `backend.json`でバックエンドの状態を更新することもできます。詳細については、link:../trident-use/backend_options.html["バックエンドを管理するオプション"]およびlink:../trident-use/backend_ops_kubectl.html["kubectlを使用してバックエンド管理を実行する"]を参照してください。

例：

JSON

次の手順に従って、 `userState`を `backend.json`ファイルを使用して更新します：

1. `backend.json`ファイルを編集して、 `userState`フィールドを値「suspended」に設定します。
2. `tridentctl update backend`コマンドと更新された `backend.json`ファイルへのパスを使用してバックエンドを更新します。

   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"
}
```

YAML

`kubectl edit <tbc-name> -n <namespace>`コマンドを使用して、適用後にtbcを編集できます。次の例では、 `userState: suspended`オプションを使用して、バックエンドの状態をsuspendに更新します：

```
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

`version`フラグを使用して、 `tridentctl`のバージョンと実行中のTridentサービスを出力します。

```
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い、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配置されている場合、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ーを指定して検索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。

Tridentを監視

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メトリクスエンドポイントのセットを提供します。

概要

Trident が提供する指標により、次のことが可能になります。

- Tridentの健全性と構成を監視します。処理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処理を実行するのにどれくらい時間がかかるかを確認できます。

> デフォルトでは、Tridentのメトリックはターゲットポートでを介して公開されます`8001`の `/metrics`エンドポイントで公開されます。これらのメトリックは、Tridentのインストール時に*デフォルトで有効*になります。ポート `8444`でHTTPS経由でTridentメトリックを使用するように設定することもできます。

要件

- TridentがインストールされたKubernetesクラスタ。
- Prometheus インスタンス。これは "コンテナ化されたPrometheusの導入"にすることも、Prometheus を "

ネイティブアプリケーション"として実行することもできます。

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、作成するボリュームなどに関する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。"Prometheus Operator ドキュメント"を参照してください。

ステップ2：Prometheus ServiceMonitorを作成する

Tridentメトリクスを使用するには、 `trident-csi`サービスを監視し、 `metrics`ポートでリッスンするPrometheus ServiceMonitorを作成する必要があります。サンプル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`サービスによって返されるメトリックを取得し、特にサービスの`metrics`エンドポイントを検索します。その結果、PrometheusはTridentのメトリックを理解できるように設定されました。

Trident から直接利用可能なメトリックに加えて、kubelet は独自のメトリック エンドポイントを介して多くの `kubelet_volume_*`メトリックを公開します。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。"。

HTTPS経由でTridentメトリクスを使用する

HTTPS（ポート8444）経由でTridentメトリクスを利用するには、ServiceMonitor定義を修正してTLS構成を含める必要があります。また、 `trident-csi`シークレットを `trident`ネームスペースからPrometheusが実行されているネームスペースにコピーする必要があります。これには次のコマンドを使用できます：

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

HTTPS メトリックのサンプル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:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Tridentは、すべてのインストール方法（tridentctl、Helm chart、Operator）でHTTPSメトリクスをサポートします。

- `tridentctl install`コマンドを使用している場合は、 `--https-metrics`フラグを渡してHTTPSメトリックを有効にすることができます。
- Helmチャートを使用している場合は、 `httpsMetrics`パラメータを設定してHTTPSメトリックを有効にできます。
- YAMLファイルを使用している場合、 `--https_metrics`フラグを `trident-main`コンテナに `trident-deployment.yaml`ファイルで追加できます。

ステップ3：PromQLを使用したTridentメトリクスのクエリ

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の健康情報を取得する

- **Trident**からの **HTTP 2XX** レスポンスの割合

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

- ステータスコード別の**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テレメトリーの詳細

デフォルトでは、Tridentは、Prometheusメトリクスと基本的なバックエンド情報をNetAppに毎日送信します。

- TridentがNetAppにPrometheusメトリクスと基本的なバックエンド情報を送信しないようにするには、Tridentのインストール時に `--silence-autosupport`フラグを渡します。
- Tridentは、 `tridentctl send autosupport`を介して、コンテナログをオンデマンドでNetAppサポートに送信することもできます。Tridentがログをアップロードするようにトリガーする必要があります。ログを送信する前に、NetAppのhttps://www.netapp.com/company/legal/privacy-policy/["プライバシー ポリシー"^]に同意する必要があります。
- 特に指定がない限り、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`フラグを使用します。例えば： `tridentctl send autosupport --since=1h`この情報は、Tridentと一緒にインストールされる `trident-autosupport`コンテナを介して収集および送信されます。コンテナイメージは "Trident AutoSupport"から入手できます。
- 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"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への取り組みの詳細については、 "ここをクリックしてください。"を参照してください。

Trident から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`フラグを使用できます。
- インストール用のYAMLファイルを生成して、プロキシURLを設定することもできます。これは、 `tridentctl install --generate-custom-yaml`を使用してYAMLファイルを作成し、 `--proxy-url`引数を `trident-autosupport`コンテナの `trident-deployment.yaml`に追加することで実行できます。

Tridentメトリクスを無効にする

メトリクスの報告を無効化するには、カスタムYAMLを生成し（ `--generate-custom-yaml` フラグを使用）、それらを編集して `--metrics` フラグが `trident-main` コンテナで呼び出されないように削除してください。

Tridentのアンインストール

Tridentのアンインストールには、Tridentのインストールに使用したのと同じ方法を使用してください。

タスク概要

- アップグレード後に発生したバグ、依存関係の問題、またはアップグレードの失敗や不完全さを修正する必要がある場合は、Tridentをアンインストールし、その"version"専用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。これは、以前のバージョンに_ダウングレード_するための唯一推奨される方法です。
- アップグレードや再インストールを簡単にするため、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および CRD を完全に削除"を参照してください。

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で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ノードで非公開になります。

元のインストール方法を決定する

Tridentのインストールに使用したのと同じ方法でアンインストール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のインストールに使用したバージョンを確認してください。

1. `kubectl get pods -n trident`を使用して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`tridentctl`を使用してインストールされました。
    - オペレーターポッドがある場合、Trident は Trident オペレーターを使用して手動または Helm を使用してインストールされました。
2. オペレータポッドがある場合は、 `kubectl describe tproc trident`を使用して、TridentがHelmを使用してインストールされたかどうかを確認します。
    - 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ました。

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

Tridentをオペレータを使用してインストールした場合は、次のいずれかを実行してアンインストールできます

：

1. 編集 **`TridentOrchestrator` CR** を実行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}}'
```

> `uninstall`フラグが `true`に設定されている場合、`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. 削除 **`TridentOrchestrator`**（：）Trident の導入に使用された `TridentOrchestrator` CR を削除することで、オペレータに Trident のアンインストールを指示します。オペレータは `TridentOrchestrator`の削除を処理し、Trident の導入とデーモンセットを削除して、インストールの一部として作成された 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、 `helm uninstall`を使用してアンインストールできます。

```
#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
```

`tridentctl`インストールをアンインストールします

Tridentに関連するCRDおよび関連オブジェクトを除くすべてのリソースを削除するには、 `uninstall`コマンドを `tridentctl`で使用してください：

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

著作権に関する情報

商標に関する情報