Tridentの管理と監視

Trident

NetApp
January 17, 2025

目次

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はリリースサイクルを4カ月に短縮し、毎年3つのメジャーリリースを提供しています。新しいリリースは、以前のリリースに基づいて構築され、新機能、パフォーマンスの強化、バグの修正、および改善が提供されます。Tridentの新機能を利用するには、少なくとも年に1回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クラスタ内のすべてのネームスペースには、Tridentインスタンスを1つだけインストールする必要があります。
- Trident 23.07以降では、v1ボリュームスナップショットが必要です。alphaまたはbetaスナップショットはサポートされません。
- でCloud Volumes Service for Google Cloudを作成した場合"CVS サービスタイプ"は、Trident 23.01からのアップグレード時にまたは `zoneredundantstandardsw`サービスレベルを使用するようにバックエンド構成を更新する必要があります `standardsw`。バックエンドでを更新しない `serviceLevel`と、ボリュームで障害が発生する可能性があります。詳細については、を参照してください "CVSサービスタイプのサンプル" 。
- をアップグレードする場合は、を `StorageClasses`Tridentで使用するために指定することが重要です `parameter.fsType`。既存のボリュームを停止することなく、削除や再作成を実行できます `StorageClasses`。
  - これは、強制の 要件 です "セキュリティコンテキスト" SAN ボリュームの場合。
  - 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`^]をクリックします。
  - 詳細については、を参照してください "既知の問題"。

ステップ**1**：バージョンを選択します

Tridentバージョンは、日付ベースの命名規則に従い `YY.MM`ます。「YY」は年の最後の2桁、「mm」は月です。ドットリリースは規則に従い `YY.MM.X`ます。「X」はパッチレベルです。アップグレード前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の4リリースウィンドウ内にある任意のターゲットリリースに直接アップグレードできます。たとえば、23.04（または任意の23.04 DOTリリース）から24.06に直接アップグレードできます。
- 4つのリリースウィンドウ以外のリリースからアップグレードする場合は、複数の手順でアップグレードを実行します。4リリースのウィンドウに適合する最新リリースにアップグレードするには、アップグレ

ード元ののののアップグレード手順を使用し "以前のバージョン" ます。たとえば、22.01を実行していて、24.06にアップグレードする場合は、次の手順を実行します。

a. 22.07から23.04への最初のアップグレード。

b. その後、23.04から24.06にアップグレードします。

> OpenShift Container PlatformでTridentオペレータ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 オペレータには、 21.01.1 で修正された既知の問題が含まれています。詳細については、 "GitHub の問題の詳細"。

ステップ**2:**元のインストール方法を決定します

Tridentを最初にインストールしたバージョンを確認するには、次の手順を実行します。

1. 使用 `kubectl get pods -n trident` ポッドを検査するために。
   - オペレータポッドがない場合は、を使用してTridentがインストールされています `tridentctl`。
   - オペレータポッドがある場合、Tridentは手動またはHelmを使用してTridentオペレータを使用してインストールされています。
2. オペレータポッドがある場合は、を使用して、 `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ています。
   - Helmラベルがない場合、TridentはTridentオペレータ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します

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、可能です。"インストール方法を切り替えます"Tridentをアップグレードする方法は2つあります。

- "Tridentオペレータを使用してアップグレード"

> レビューすることをお勧めします "オペレータのアップグレードワークフローについて理解する" オペレータでアップグレードする前に。

*

オペレータにアップグレードしてください

オペレータのアップグレードワークフローについて理解する

Tridentオペレータを使用して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について理解しておく必要があります。これには、Tridentコントローラ、コントローラポッドとノードポッド、およびローリング更新を可能にするノードデーモンセットに対する変更が含まれます。

Tridentオペレータのアップグレード処理

Tridentをインストールしてアップグレードするには"Tridentオペレータを使用するメリット"、既存のマウントボリュームを中断することなく、TridentオブジェクトとKubernetesオブジェクトを自動的に処理する必要があります。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。"ローリング更新"TridentオペレータはKubernetesクラスタと通信して次のことを行います。

- Trident Controller環境とノードデーモンセットを削除して再作成します。
- TridentコントローラポッドとTridentノードポッドを新しいバージョンに置き換えます。
  - 更新されていないノードは、残りのノードの更新を妨げません。
  - ボリュームをマウントできるのは、Trident Node Podを実行しているノードだけです。

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

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、次の処理が実行されます。

1. Trident演算子*：
   a. 現在インストールされているTridentのバージョン（version_n_）を検出します。
   b. CRD、RBAC、Trident SVCなど、すべてのKubernetesオブジェクトを更新
   c. version_n_用のTrident Controller環境を削除します。
   d. version_n+1_用のTrident Controller環境を作成します。
2. * Kubernetes *は、_n+1_用にTridentコントローラポッドを作成します。
3. Trident演算子*：
   a. _n_のTridentノードデーモンセットを削除します。オペレータは、Node Podが終了するのを待たない。
   b. _n+1_のTridentノードデーモンセットを作成します。
4. * Kubernetes * Trident Node Pod_n_を実行していないノードにTridentノードポッドを作成します。これにより、1つのノードに複数のTrident Node Pod（バージョンに関係なく）が存在することがなくなります。

Trident operatorまたはHelmを使用したTridentインストールのアップグレード

Tridentは、Tridentオペレータを使用して手動でアップグレードすることも、Helmを使用してアップグレードすることもできます。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することも、インストールからTridentオペレータのバージョンにアップグレードすることもできます `tridentctl` 。Trident Operatorのインストールをアップグレードする前にを参照してください"アップグレード方法を選択します"。

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オペレータインストールから、クラスタを対象とした別のTridentオペレータインストールにアップグレードできます。バージョン21.01以降のTridentでは、すべてクラスタを対象とした演

算子が使用されます。

> 名前空間を対象とした演算子（バージョン20.07～20.10）を使用してインストールされたTridentからアップグレードするには、Tridentののアップグレード手順を使用します"インストールされているバージョン"。

このタスクについて

Tridentにはバンドルファイルが用意されています。このファイルを使用して、オペレータをインストールしたり、Kubernetesバージョンに対応する関連オブジェクトを作成したりできます。

- クラスタでKubernetes 1.24を実行している場合は、を使用し "Bundle_pre_1_25.yaml"ます。
- クラスタでKubernetes 1.25以降を実行している場合は、を使用します "bundle_post_1_25.yaml"。

作業を開始する前に

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ます "サポートされるKubernetesバージョン"。

手順

1. Tridentのバージョンを確認します。

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

2. 現在のTridentインスタンスのインストールに使用したTridentオペレータを削除します。たとえば、23.07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 23.07.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

3. を使用して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、ミラーリングされたTridentおよびCSIイメージレジストリをオフラインモードに指定したり、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ます。
4. ご使用の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（__は、または `bundle_post_1_25.yaml`使用している<bundle.yaml> `bundle_pre_1_25.yaml`のバージョンに基づいています）。たとえば、Trident 24.10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 24.10.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

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`helm upgrade`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、クラスタをアップグレードする前に、values.yamlをに`true`設定するかコマンドに追加する`--set excludePodSecurityPolicy=true`ように更新する必要があります。`excludePodSecurityPolicy`

Trident Helmをアップグレードせずにクラスタを1.24から1.25にアップグレードした場合、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、次の手順を前提条件として実行します。

1. からhelm-mapkubeapisプラグインをインストールします https://github.com/helm/helm-mapkubeapis。
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。リソースが一覧表示され、クリーンアップされます。

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

3. クリーンアップを実行するには、helmを使用してフルランを実行します。

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

手順

1. を使用する "Helmを使用してTridentをインストール"と、を使用してワンステップでアップグレードできます`helm upgrade trident netapp-trident/trident-operator --version 100.2410.0`。Helmリポジトリを追加しなかった場合、またはHelmリポジトリを使用してアップグレードできない場合は、次の手順を実行します。
   a. から最新のTridentリリースをダウンロードし"GitHubの_Assets_sectionを参照してください"ます。
   b. コマンドを使用し`helm upgrade`ます。は、`trident-operator-24.10.0.tgz`アップグレード先のバージョンを反映しています。

```
helm upgrade <name> trident-operator-24.10.0.tgz
```

   初期インストール時にカスタムオプションを設定した場合（TridentイメージとCSIイメージのプライベートなミラーレジストリの指定など）は、`helm upgrade`コマンド`--set`これらのオプションがupgradeコマンドに含まれるようにするため、それらのオプションの値をdefaultにリセットします。

2. を実行します`helm list`グラフとアプリのバージョンが両方ともアップグレードされていることを確認します。を実行します`tridentctl logs`デバッグメッセージを確認します。

からのアップグレード`tridentctl` **Trident**オペレータへのインストール

からTridentの最新リリースにアップグレードできます`tridentctl`インストール：既存のバックエンドとPVCは自動的に使用可能になります。

インストール方法を切り替える前に、 "インストール方法を切り替える"。

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 [24.10.0]
mkdir 24.10.0
cd 24.10.0
wget
https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

2. マニフェストから「 tridentオーケストラ 」 CRD を作成します。

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

3. クラスタを対象としたオペレータを同じネームスペースに導入します。

```
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CRを作成し `TridentOrchestrator`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が目的のバージョンにアップグレードされたことを確認

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

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v24.10.0
```

tridentctl を使用してアップグレードします

を使用して、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`。

このタスクについて

Tridentのアンインストールと再インストールは、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環境で使用されている永続的ボリューム要求（PVC）と永続的ボリューム（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PVCは、Tridentがオフラインの間も使用できます。また、その間に作成されたPVCがオンラインに戻ったあとも、Tridentはボリュームをプロビジョニングします。

作業を開始する前に

レビュー "アップグレード方法を選択します" を使用してアップグレードする前に `tridentctl`。

手順

1. のuninstallコマンドを実行し `tridentctl`て、CRDと関連オブジェクトを除くTridentに関連付けられている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。を参照してください "tridentctlを使用したTridentのインストール"。

> 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ることを確認します。

tridentctlを使用したTridentの管理

には、 "Trident インストーラバンドル"Tridentに簡単にアクセスできるコマンドラインユーティリティが含まれてい `tridentctl`ます。十分なPrivilegesを持つKubernetesユーザは、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

走れ `tridentctl help` 使用可能なコマンドのリストを取得するには `tridentctl` または、 `--help` 特定のコマンドのオプションとフラグのリストを取得するには、任意のコマンドにフラグを付けます。

```
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
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
任意のコマンドに関するヘルプ。

images
Tridentが必要とする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

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
バックエンド処理を一時的に中断します。

upgrade
Tridentでリソースをアップグレードします。

「バージョン」
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
ヘルプ `tridentctl`。

-k、 --kubeconfig string
を指定します。 `KUBECONFIG` コマンドをローカルまたはKubernetesクラスタ間で実行するパス。

または、 `KUBECONFIG` 特定のKubernetesクラスタと問題をポイントする変数 `tridentctl` そのクラスタにコマンドを送信します。

-n、 --namespace string
Trident環境のネームスペース。

-o、 --output string
出力形式。JSON の 1 つ | yaml | name | wide | ps （デフォルト）。

-s、 --server string
Trident RESTインターフェイスのアドレス/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）のみをリスンして処理するように設定できます。

コマンドオプションとフラグ

作成

コマンドを使用し `create`て、Tridentにリソースを追加します。

```
tridentctl create [option]
```

オプション（ **Options** ）
`backend`：Tridentにバックエンドを追加します。

削除

コマンドを使用し `delete`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（ **Options** ）
`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

コマンドを使用し `get`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（ **Options** ）

`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`：クエリを下位のソースボリュームに制限します。
`--subordinateOf string`:クエリをボリュームの下位に制限します。

イメージ

フラグを使用して `images`、Tridentが必要とするコンテナイメージのテーブルを印刷します。

```
tridentctl images [flags]
```

フラグ

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

ボリュームをインポートします

コマンドを使用し `import volume`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。

をインストールします

フラグを使用し `install`てTridentをインストールします。

```
tridentctl install [flags]
```

フラグ

`--autosupport-image string`：AutoSupportテレメトリ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`：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`: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コントローラのREST APIのHTTP要求タイムアウトを上書きします（デフォルトは1m30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処理のタイムアウト（デフォルトは3m0）。
`--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ログ形式（text、json）（デフォルトは「text」）。
`--node-prep`：指定し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できるように、TridentでKubernetesクラスタ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値は、のみです。
`--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に自動的に送信しないでください(デフォルトはTRUE)。
`--silent`:インストール中にMOST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。
`--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IPv6を使用します。

ログ

フラグを使用して `logs`Tridentからログを出力します。

```
tridentctl logs [flags]
```

フラグ

`-a、 --archive`：特に指定がないかぎり、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が存在する場合は、そのインスタンスのログを取得します。
`--sidecars`:サイドカーコンテナのログを取得します。

送信

Tridentからリソースを送信するには、コマンドを使用し `send`ます。

```
tridentctl send [option]
```

オプション（ **Options** ）

`autosupport`：ネットアップにAutoSupport アーカイブを送信します。

をアンインストールします

フラグを使用して `uninstall`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のほとんどの出力を無効にします。

更新

Tridentのリソースを変更するには、コマンドを使用し `update`ます。

```
tridentctl update [option]
```

オプション（ **Options** ）

`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

を使用します `update backend state` バックエンド処理を一時停止または再開するコマンド。

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

考慮すべきポイント

- TridentBackendConfig（tbc）を使用してバックエンドを作成した場合、ファイルを使用してバックエンドを更新することはできません `backend.json` 。
- がtbcに設定されている場合 `userState` は、コマンドを使用して変更することはできません `tridentctl update backend state <backend-name> --user-state suspended/normal` 。
- tbcで設定した後にvia tridentctlを設定できるようにするには `userState`、 `userState`tbcからフィールドを削除する必要があります。これは、コマンドを使用して実行でき `kubectl edit tbc` ます。フィールドを削除したら `userState`、コマンドを使用してバックエンドのを変更 `userState`できます `tridentctl update backend state`。
- を使用して `tridentctl update backend state` を変更し `userState`ます。またはファイルを使用して更新することもでき `userState` `TridentBackendConfig` `backend.json` ます。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`:バックエンド状態のヘルプ。
  `--user-state`:に設定 `suspended` バックエンド処理を一時停止します。をに設定します `normal` バックエンド処理を再開します。に設定すると `suspended`：

- `AddVolume` `Import Volume` 一時停止しています。
- `CloneVolume`、 `ResizeVolume`、 `PublishVolume`、 `UnPublishVolume`、、 `CreateSnapshot`

GetSnapshot RestoreSnapshot、、 DeleteSnapshot、 RemoveVolume、、
GetVolumeExternal ReconcileNodeAccess 引き続き使用できます。

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

- 例： *

JSON

ファイルを使用してを更新するには、次の手順を実行し `userState` `backend.json` ます。

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

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

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

YAML

tbcが適用されたら、コマンドを使用して編集できます `kubectl edit <tbc-name> -n <namespace>` 。次に、オプションを使用してバックエンド状態をsuspendに更新する例を示し `userState: suspended` 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
```

バージョン

使用 `version` のバージョンを印刷するためのフラグ `tridentctl` 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に似たプラグインをサポートしています。Tridentctlは、プラグインバイナリファイル名が"tridentctl -<plugin>"というスキームに沿っている場合にプラグインを検出し、そのバイナリがPATH環境変数のリストにあるフォルダにあることを示します。検出されたすべてのプラグインは、tridentctlヘルプのpluginセクションに表示されます。オプションで、環境変数TRIDENTCTL_PLUGIN_PATHにプラグインフォルダ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/`)。変数が使用されている場合、tridenctlは指定されたフォルダのみを検索します。

Tridentの監視

Tridentには、Tridentのパフォーマンスの監視に使用できるPrometheus指標エンドポイントのセットが用意されています。

概要

Tridentの指標を使用すると、次のことを実行できます。

- Tridentの健全性と設定を常に確認しておきます。処理が成功した方法と、想定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の情報を調べて、バックエンドで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などを確認します。
- 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する。Tridentがバックエンドと通信して処理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 のメトリックは '/metrics エンドポイントのターゲットポート 8001' に公開されていますこれらの指標は、 Trident のインストール時にデフォルトで * 有効になります。

必要なもの

- 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 a である場合もある "コンテナ化された Prometheus 環境" または、Prometheus をとして実行することもできます "ネイティブアプリケーション"。

手順 1 ： Prometheus ターゲットを定義する

Prometheusターゲットを定義して、指標を収集し、Tridentが管理するバックエンドや作成するボリュームなどに関する情報を取得する必要があります。ここで "ブログ"は、PrometheusとGrafanaをTridentで使用して指標を取得する方法について説明します。このブログでは、KubernetesクラスタでPrometheusをオペレータとして実行する方法と、Trident指標を取得するためのServiceMonitorの作成について説明しています。

手順 2 ： Prometheus ServiceMonitor を作成します

Trident のメトリックを使用するには、「 trident-csi 」サービスを監視し、「 m etrics 」ポートを監視する Prometheus ServiceMonitor を作成する必要があります。ServiceMonitor のサンプル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、サービスから返されたメトリックを取得し `trident-csi`、特にサービスのエンドポイントを検索し `metrics`ます。その結果、PrometheusはTridentの指標を認識するように設定されました。

Kubeletは、Tridentから直接利用できるメトリクスに加えて、独自のメトリクスエンドポイントを介して多くのメトリクスを公開して `kubelet_volume_*`います。Kubelet では、接続されているボリュームに関する情報、およびポッドと、それが処理するその他の内部処理を確認できます。を参照してください "こちらをご覧ください"。

ステップ 3 ： PrompQL を使用して Trident 指標を照会する

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

次に、 PrompQL クエリーのいくつかを示します。

Trident の健常性情報を取得

- **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
```

- ステータスコードによる**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の使用状況情報の取得

- 平均体積サイズ

```
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合計ボリューム容量

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

個々のボリュームの使用状況を取得する

これは、 kubelet 指標も収集された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テレメトリの詳細

デフォルトでは、TridentはPrometheus指標と基本的なバックエンド情報を1日おきにNetAppに送信します。

- TridentからNetAppへのPrometheus指標と基本的なバックエンド情報の送信を停止するには、Tridentのインストール時にフラグを渡し `--silence-autosupport`ます。
- Tridentは、経由でコンテナログを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。ログを送信する前に、NetAppを承認する必要がありますhttps://www.netapp.com/company/legal/privacy-policy/["プライバシーポリシー"^]。
- 指定されていない場合、Tridentは過去24時間のログをフェッチします。
- フラグを使用してログの保持期間を指定できます `--since`。例： `tridentctl send autosupport --since=1h`。この情報は、Tridentと一緒にインストールされたコンテナを介して収集および送信され `trident-autosupport`ます。コンテナイメージはから入手できます "Trident AutoSupport の略"。
- Trident AutoSupport は、個人情報（ PII ）や個人情報を収集または送信しません。Tridentコンテナイメージ自体には適用されないが付属して "EULA" います。データのセキュリティと信頼に対するネットアップの取り組みについて詳しくは、こちらをご覧 "こちらをご覧ください"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:
    disableDelete: false
    serialNumbers:
    - nwkvzfanek_SN
    limitVolumeSize: ''
  state: online
  online: true
```

- AutoSupport メッセージは、ネットアップの AutoSupport エンドポイントに送信されます。プライベートレジストリを使用してコンテナイメージを格納している場合は '--image_registry' フラグを使用できます
- インストール 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 'trident-deployment.yaml の trident-autosupport コンテナに --proxy-url 引数を追加することによって実行できます

Trident指標を無効にする

- メトリックがレポートされないようにするには '--generate-custom-yaml' フラグを使用してカスタム YAML を生成し ' これらを編集して 'trident-main' コンテナに対して --metrics フラグが呼び出されないよう

にします

Trident をアンインストールします

Tridentのアンインストールには、Tridentのインストールと同じ方法を使用する必要があります。

このタスクについて

- アップグレード、依存関係の問題、またはアップグレードの失敗または不完全な実行後に見つかったバグの修正が必要な場合は、Tridentをアンインストールし、該当する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"バージョン"。これは、以前のバージョンに_downgrade_toを実行するための唯一の推奨方法です。
- アップグレードと再インストールを簡単に行うために、Tridentをアンインストールしても、Tridentによって作成されたCRDや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、を参照してください"TridentとCRDを完全に取り外します。"。

作業を開始する前に

Kubernetesクラスタの運用を停止する場合は、を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のアンインストールには、インストール時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、Tridentの最初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. 使用 `kubectl get pods -n trident` ポッドを検査するために。
   - オペレータポッドがない場合は、を使用してTridentがインストールされています `tridentctl`。
   - オペレータポッドがある場合、Tridentは手動またはHelmを使用してTridentオペレータを使用してインストールされています。
2. オペレータポッドがある場合は、を使用して、 `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ています。
   - Helmラベルがない場合、TridentはTridentオペレータ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オペレータ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。

手動インストールのアンインストール

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**を実行し、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}}'
```

をクリックします `uninstall` フラグはに設定されています `true`は、TridentオペレータがTridentをアンインストールしますが、TridentOrchestrator自体は削除されません。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 Trident を作成する必要があります。

2. 削除 **`TridentOrchestrator`**：Tridentの展開に使用したCRを削除する `TridentOrchestrator` と、`Trident`をアンインストールするようにオペレータに指示します。オペレータがの削除を処理し `TridentOrchestrator`、インストール時に作成したTridentポッドを削除して、Tridentデプロイメントとデーモンセットの削除に進みます。

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

Helmインストールのアンインストール

Helmを使用してTridentをインストールした場合は、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
```

のアンインストール `tridentctl` インストール

Tridentに関連付けられているすべてのリソース（CRDおよび関連オブジェクトを除く）を削除するには、のコマンドを `tridentctl`使用し `uninstall`ます。

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

著作権に関する情報

商標に関する情報