



Astra Tridentの管理と監視

Astra Trident

NetApp
April 03, 2024

目次

Astra Tridentの管理と監視	1
Astra Trident をアップグレード	1
Tridentctlを使用したAstra Tridentの管理	8
Astra Trident を監視	13
Astra Trident をアンインストール	17

Astra Tridentの管理と監視

Astra Trident をアップグレード

Astra Trident をアップグレード

Astra Trident は四半期ごとにリリースサイクルを実施し、毎年 4 つのメジャーリリースをリリースしています。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。ネットアップでは、Astra Tridentの新機能を活用するために、1年に1回以上アップグレードすることを推奨しています。

アップグレード前の考慮事項

最新リリースの Astra Trident にアップグレードする際は、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、Astra 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。
- Astra Trident 23.07以降では、v1ボリュームSnapshotが必要です。アルファSnapshotまたはベータSnapshotはサポートされなくなりました。
- Cloud Volumes Service for Google Cloudを "[CVS サービスタイプ](#)"を使用するには、バックエンド構成を更新する必要があります。 `standardsw` または `zoneredundantstandardsw` Astra Trident 23.01からアップグレードする場合のサービスレベル。の更新に失敗しました `serviceLevel` バックエンドでは、原因ボリュームで障害が発生する可能性があります。を参照してください "[CVSサービスタイプのサンプル](#)"を参照してください。
- アップグレードするときは、この作業を行うことが重要です `parameter.fsType` インチ `StorageClasses` Astra Tridentが使用。削除して再作成することができます `StorageClasses` 実行前のボリュームの中断はなし。
 - 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.template>などの例が含まれています[`storage-class-basic.yaml.template`] 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

Astra Tridentバージョンは日付ベースです `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、の後に続きます `YY.MM.X` 条約。ここで、「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、22.10（または任意の22.10 DOTリリース）から23.10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。のアップグレード手順を使用します。"[以前のバージョン](#)"から、4つのリリースウィンドウに適合する最新のリリースにアップグレードします。たとえば、21.10を実行していて、23.10にアップグレードする場合は、次の手順を実行します。
 - a. 21.10から22.10への最初のアップグレード。
 - b. その後、22.10から23.10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、[を参照してください "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

Astra Tridentの最初のインストールに使用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、`tridentctl`を使用してAstra Tridentがインストールされています。
 - オペレータポッドがある場合、Astra Tridentは手動またはHelm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、`kubectl describe tproc trident` をクリックし、Helmを使用してAstra Tridentがインストールされたかどうかを確認します。
 - Helmラベルがある場合は、Helmを使用してAstra Tridentがインストールされています。
 - Helmラベルがない場合は、Astra TridentをTridentオペレータを使用して手動でインストールしています。

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Astra 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、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

多数のうちの一つ "[Tridentオペレータを使用するメリット](#)" Astra Tridentのインストールとアップグレードは、既存のマウントボリュームを停止することなく、Astra TridentとKubernetesのオブジェクトを自動的に処理します。これにより、Astra 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Astra Tridentアーキテクチャの詳細については、"[Astra 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Astra 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オペレータのインストールをアップグレード

Astra Tridentは、Tridentオペレータを使用して手動またはHelmを使用してアップグレードできます。Tridentオペレータのインストール環境から別のTridentオペレータのインストール環境へのアップグレード、または 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ます](#)" Tridentオペレータの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。すべてのAstra Tridentバージョン21.01以降では、クラスタを対象とした演算子を使用します。



ネームスペースを対象としたオペレータ（バージョン20.07~20.10）を使用してインストールされたAstra Tridentからアップグレードするには、次のアップグレード手順を使用してください：
：“インストールされているバージョン”実績があります。

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以前を実行している場合は、を使用します "Bundle_pre_1_25.yaml"。
- 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. Astra Tridentのバージョンを確認します。

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

2. 現在のAstra 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. 環境に応じた適切なバンドルYAMLファイルを使用してAstra Tridentをインストールします（_YAML <bundle.yaml>_は bundle_pre_1_25.yaml または bundle_post_1_25.yaml 使用しているKubernetesのバージョンに基づきます。たとえば、Astra Trident 23.10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 23.10.0/trident-installer/deploy/<bundle.yaml> -n trident
```

Helmインストールのアップグレード

Astra Trident Helmのインストールをアップグレードできます。



Astra TridentがインストールされているKubernetesクラスターを1.24から1.25以降にアップグレードする場合は、value.yamlを更新して設定する必要があります
excludePodSecurityPolicy 終了: true または、を追加します --set
excludePodSecurityPolicy=true に移動します helm upgrade コマンドを実行してから
クラスターをアップグレードしてください。

手順

1. あなたの場合同様 **"Helmを使用したAstra Tridentのインストール"**を使用できます。helm upgrade trident netapp-trident/trident-operator --version 23.10.0 1つの手順でアップグレードできます。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. 次のサイトからAstra Tridentの最新リリースをダウンロードしてください: ["GitHubの_Assets_sectionを参照してください"](#)。
 - b. を使用します helm upgrade コマンドを入力します trident-operator-23.10.0.tgz アップグレード後のバージョンが反映されます。

```
helm upgrade <name> trident-operator-23.10.0.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 Astra Trident リリースをダウンロード

```
# Download the release required [23.10.0]
mkdir 23.10.0
cd 23.10.0
wget
https://github.com/NetApp/trident/releases/download/v23.10.0/trident-
installer-23.10.0.tar.gz
tar -xf trident-installer-23.10.0.tar.gz
cd trident-installer
```

2. を作成します 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. を作成します TridentOrchestrator Astra Tridentのインストール用に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:          v23.10.0
```

tridentctl を使用してアップグレードします

を使用すると、既存のAstra 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

Astra 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、Astra Trident 環境で使用されている Persistent Volume Claim (PVC ; 永続的ボリューム要求) と Persistent Volume (PV ; 永続的ボリューム) は削除されません。Astra Trident がオフラインの間は、すでにプロビジョニング済みの PVS を引き続き使用でき、Astra Trident は、オンラインに戻った時点で作成された PVC に対して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. のアンインストールコマンドを実行します tridentctl CRDと関連オブジェクトを除くAstra Tridentに関連付けられているすべてのリソースを削除する。

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

2. Astra Tridentを再インストールします。を参照してください ["tridentctl を使用して Astra Trident をインストールします"](#)。



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Astra Tridentの管理

。 ["Trident インストーラバンドル"](#) には、 `tridentctl` 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。十分な権限を持つKubernetesユーザは、この権限を使用してAstra Tridentをインストールしたり、Astra 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]
```

Astra 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

Astra Tridentにリソースを追加

delete

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入手します。

help

任意のコマンドに関するヘルプ。

images

Astra Tridentが必要とするコンテナイメージの表を出力します。

import

既存のリソースをAstra Tridentにインポート

install

Astra Trident をインストール

logs

Astra Tridentからログを出力

send

Astra Tridentからリソースを送信

uninstall

Astra Tridentをアンインストールします。

update

Astra Tridentでリソースを変更

update backend state

バックエンド処理を一時的に中断します。

upgrade

Astra Tridentでリソースをアップグレード

version

Astra 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

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

Astra Trident導入のネームスペース。

-o、 --output string

出力形式。 JSON の 1 つ | yaml | name | wide | ps (デフォルト)。

-s、 --server string

Astra 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

を使用します `create` Astra Tridentにリソースを追加するコマンド。

```
tridentctl create [option]
```

オプション (Options)

`backend` : Astra Tridentにバックエンドを追加

削除

を使用します `delete` コマンドを使用して、 Astra 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

`snapshot` : Astra Tridentから1つ以上のボリュームSnapshotを削除

storageclass : Astra Tridentから1つ以上のストレージクラスを削除
volume : Astra Tridentから1つ以上のストレージボリュームを削除

取得

を使用します get Astra Tridentから1つ以上のリソースを取得するためのコマンドです。

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

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

使用 images Astra Tridentが必要とするコンテナイメージの表を出力するためのフラグ。

```
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

を使用します import volume コマンドを使用して、既存のボリュームをAstra 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

を使用します install Astra Tridentのインストールにフラグを付けます。

```
tridentctl install [flags]
```

フラグ

- autosupport-image string: AutoSupportテレメトリ用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ます。
- generate-custom-yaml: インストールを行わずに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3分0)。
- 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: Astra Tridentのログ形式(テキスト、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で使用されている従来のPVCの名前。このPVCが存在しないことを確認します (デフォルトは「trident」)。
- silence-autosupport: AutoSupport バンドルを自動的にネットアップに送信しない (デフォルトはtrue)。
- silent: インストール中は、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Astra Tridentのイメージ
- 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の通信にIPv6を使用

ログ

使用 logs Astra Tridentから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: Astra Tridentのログが表示されます。trident | auto | trident-operator | all (デフォルトは「auto」) のいずれかです。
- node string: ノードポッドログの収集元の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、それを取得します。
- sidecars: サイドカーコンテナのログを取得します。

送信

を使用します send Astra Tridentからリソースを送信するコマンド。

```
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

使用 uninstall Astra 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のほとんどの出力を無効にします。

更新

を使用します update Astra Tridentでリソースを変更するコマンド。

```
tridentctl update [option]
```

オプション (Options)

- backend: Astra Tridentのバックエンドを更新。

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]
```

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、CloneVolume、Import Volume、ResizeVolume は一時停止しています。
- PublishVolume、UnPublishVolume、CreateSnapshot、GetSnapshot、RestoreSnapshot、DeleteSnapshot、RemoveVolume、GetVolumeExternal、ReconcileNodeAccess 引き続き使用できます。

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。

Astra Trident を監視

Astra Tridentは、Astra Tridentのパフォーマンス監視に使用できるPrometheus指標エンドポイントのセットを提供します。

概要

Astra Trident が提供する指標を使用すると、次のことが可能になります。

- Astra Trident の健全性と設定を保持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。Astra 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の時間がかかるかを調べることができます。



デフォルトでは、Tridentの指標はターゲットポートで公開されています 8001 で /metrics エンドポイント。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

- Astra 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 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、Astra Trident が管理するバックエンド、作成するボリュームなどの情報を取得する必要があります。これ "[ブログ](#)" Prometheus と Grafana を Astra Trident とともに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 クラスターのオペレーターとして Prometheus を実行する方法と、Astra Trident の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 の指標を利用するには、を監視する Prometheus ServiceMonitor を作成する必要があります `trident-csi` サービスおよびリッスン `metrics` ポート : 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、から返されたメトリックを取得します trident-csi サービスとは、を特に探します metrics サービスのエンドポイント。その結果、PrometheusはAstra Tridentの指標：

Astra Tridentから直接取得できる指標に加えて、kubeletは多くの指標を公開しています kubelet_volume_* 独自の指標エンドポイントを使用した指標。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 の健全性情報を取得

- **Astra 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

```

- **Astra Trident** からのステータスコードによる **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

```

- **Astra Trident** によって実行された処理の平均時間（ミリ秒）

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

Astra Trident の使用状況に関する情報を入手

- 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
```

Astra Trident AutoSupport の計測データ

デフォルトでは、Astra Trident は Prometheus 指標と基本バックエンド情報を毎日定期的にネットアップに送信します。

- Astra TridentからPrometheus指標や基本バックエンド情報がネットアップに送信されないようにするには、`--silence-autosupport` Astra Tridentのインストール中にフラグを付ける。
- Tridentからネットアップサポートにコンテナログをオンデマンドで送信することもできます `tridentctl send autosupport`。Astra Trident をトリガーしてログをアップロードする必要があります。ログを送信する前に、ネットアップの同意する必要があります ["プライバシーポリシー"](#)。
- 指定しないと、Astra Trident は過去 24 時間からログを取得します。
- ログの保持期間は、で指定できます `--since` フラグ。例：`tridentctl send autosupport --since=1h`。この情報は、を介して収集および送信されます `trident-autosupport` コンテナこれはAstra Tridentと一緒にインストールされます。コンテナイメージは、で取得できます ["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。それにはが付いていません **"EULA"** これは Trident コンテナイメージ自体には該当しません。ネットアップのデータセキュリティと信頼に対する取り組みの詳細を確認できます ["こちらをご覧ください"](#)。

Astra Trident から送信されるペイロードの例を次に示します。

```
---
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。コンテナイメージの格納にプライベートレジストリを使用している場合は、を使用できません `--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 を設定することもできます。これは、を使用して実行できます `tridentctl install --generate-custom-yaml` YAML ファイルを作成し、を追加します `--proxy-url` の引数 `trident-autosupport` にコンテナがあります `trident-deployment.yaml`。

Astra Trident の指標を無効化

**メトリックがレポートされないようにするには、を使用してカスタムYAMLを生成する必要があります `--generate-custom-yaml` フラグを付けて編集し、を削除します `--metrics` に対する呼び出し元からのフラグ `trident-main`
コンテナ：

Astra Trident をアンインストール

Astra Tridentのアンインストールには、Astra 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、アップグレードの失敗や不完全な完了後に観察されたバグの修正が必要な場合は、Astra Tridentをアンインストールし、該当する手順を使用し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、Astra Tridentをアンインストールしても、Astra Tridentで作成されたCRDや関連オブジェクトは削除されません。Astra 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。 "[Astra TridentとCRDを完全に削除](#)".

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、Astra 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アンインストールする前に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

Astra Tridentは、インストール時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、Astra 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Astra 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、Astra Tridentは手動またはHelm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、を使用します `kubectl describe tproc trident` をクリックし、Helmを使用してAstra Tridentがインストールされたかどうかを確認します。
 - Helmラベルがある場合は、Helmを使用してAstra Tridentがインストールされています。
 - Helmラベルがない場合は、Astra TridentをTridentオペレータを使用して手動でインストールしています。

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Astra 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**：TridentOrchestrator Astra Tridentの導入に使用したCRでは、Tridentをアンインストールするようオペレータに指示します。オペレータがの削除を処理します TridentOrchestrator さらに、Astra Tridentの導入とデプロイを削除し、インストールの一部として作

成したTridentポッドを削除します。

```
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Astra 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 Astra 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

を使用します `uninstall` のコマンド `tridentctl` CRDと関連オブジェクトを除くAstra Tridentに関連付けられているすべてのリソースを削除するには、次の手順を実行します。

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

著作権に関する情報

Copyright © 2024 NetApp, Inc. All Rights Reserved. Printed in the U.S.このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。