Trident オペレータとともに導入

Astra Trident

NetApp
April 16, 2024

目次

Trident オペレータとともに導入

Tridentのオペレータが、Astra Tridentを導入できます。

Astra Tridentに関する重要な情報22.10

- Astra Trident 22.10.*にアップグレードする前に、次の重要な情報をお読みください

> **Astra Tridentに関する重要な情報22.10**
>
> - TridentでKubernetes 1.25がサポートされるようになりました。Kubernetes 1.25にアップグレードする前に、Astra Trident 22.10にアップグレードする必要があります。
> - Astra Tridentは、SAN環境でマルチパス構成を厳密に使用するよう強制し、推奨値をに設定するようになりました `find_multipaths: no` multipath.confファイル内。
>
> 非マルチパス構成またはを使用 `find_multipaths: yes` または `find_multipaths: smart` multipath.confファイルの値が原因でマウントが失敗します。Tridentはの使用を推奨しています `find_multipaths: no` 21.07リリース以降

Tridentのオペレータ導入オプション

Tridentオペレータは、次のいずれかの方法で導入できます。

- Tridentの使用 "Helmチャート"：Helm ChartがTridentオペレータを導入し、Tridentをワンステップでインストールします。
- 手動：Tridentは、オペレータのインストールや関連オブジェクトの作成に使用できるファイルを提供します。
  - Kubernetes 1.24以前を実行しているクラスタの場合は、を使用します "Bundle_pre_1_25.yaml"。
  - Kubernetes 1.25以上を実行するクラスタの場合は、を使用します "bundle_post_1_25.yaml"。

> をまだ理解していない場合は、を参照してください "基本概念"今こそ、そのための絶好の機会です。

前提条件を確認する

Astra Trident を導入するには、次の前提条件を満たしている必要があります。

- サポートされているバージョンのKubernetesを実行している、サポートされているKubernetesクラスタに対するすべての権限が必要です。を確認します "要件"。
- サポートされているネットアップストレージシステムを利用できるようにしておきます。
- すべての Kubernetes ワーカーノードからボリュームをマウントできます。
- を搭載したLinuxホストがある `kubectl` （または `oc`OpenShiftを使用している場合）Kubernetesクラスタを管理するようにインストールおよび設定します。
- を設定しておきます `KUBECONFIG` Kubernetesクラスタ構成を参照する環境変数。

- を有効にしておきます "Astra Trident に必要な機能ゲート"。
- Kubernetes と Docker Enterprise を併用する場合は、 "CLI へのアクセスを有効にする手順は、ユーザが行ってください"。

それはすべてですか？最高！それでは始めましょう。

Tridentオペレータを導入し、Helmを使用してAstra Tridentをインストール

Helm を使用して Trident オペレータを導入するには、以下の手順を実行します。

必要なもの

上記の前提条件に加え、 Helm を使用して Trident Operator を導入するには、次のものが必要です。

- A "サポートされるKubernetesバージョン"
- Helm バージョン 3

手順

1. Trident の Helm リポジトリを追加：

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. を使用します `helm install` コマンドを使用し、導入環境の名前を指定します。
次の例を参照してください。

```
helm install <name> netapp-trident/trident-operator --version 22.10.0
--create-namespace --namespace <trident-namespace>
```

> Tridentのネームスペースを作成済みの場合は、を参照してください `--create -namespace` パラメータでネームスペースが追加で作成されることはありません。

インストール中に設定データを渡すには、次の 2 つの方法があります。

- `--values` （または `-f`):オーバーライドを使用してYAMLファイルを指定します。これは複数回指定でき、右端のファイルが優先されます。
- `--set`:コマンドラインでオーバーライドを指定します

たとえば、のデフォルト値を変更するには、のように指定します `debug`をクリックし、次のコマンドを実行します `--set` コマンドを実行します

```
helm install <name> netapp-trident/trident-operator --version 22.10.0
--create-namespace --namespace --set tridentDebug=true
```

。 `values.yaml` File。Helmチャートの一部で、キーのリストとデフォルト値が表示されます。

`helm list` 名前、ネームスペース、グラフ、ステータス、 アプリケーションのバージョン、リビジョン番号など。

Tridentオペレータを手動で導入し、Tridentをインストール

Trident のオペレータを手動で導入するには、以下の手順を実行します。

ステップ 1 ： Kubernetes クラスタを確認する

まず、 Linux ホストにログインして、 _working _ 、 "サポートされる Kubernetes クラスタ" に必要な権限があることを確認します。

> OpenShiftでは、を使用します `oc` ではなく `kubectl` 以降のすべての例では、を実行して、最初に* system:admin *としてログインします `oc login -u system:admin` または `oc login -u kube-admin`。

Kubernetesのバージョンを確認するには、次のコマンドを実行します。

```
kubectl version
```

Kubernetes クラスタ管理者の権限があるかどうかを確認するには、次のコマンドを実行します。

```
kubectl auth can-i '*' '*' --all-namespaces
```

Docker Hub のイメージを使用してポッドを起動し、ポッドネットワーク経由でストレージシステムにアクセスできるかどうかを確認するには、次のコマンドを実行します。

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

手順 2 ：オペレータをダウンロードして設定します

> 21.01 以降、 Trident Operator はクラスタを対象とします。TridentのオペレータがTridentをインストールするには、を作成する必要があります `TridentOrchestrator` カスタムリソース定義（CRD）およびその他のリソースの定義。Astra Trident をインストールする前に、次の手順を実行してオペレータをセットアップする必要があります。

1. からTridentインストーラバンドルの最新バージョンをダウンロードして展開します "GitHub の_Assets_sectionを参照してください"。

```
wget
https://github.com/NetApp/trident/releases/download/v22.10.0/trident-
installer-22.10.0.tar.gz
tar -xf trident-installer-22.10.0.tar.gz
cd trident-installer
```

2. 適切なCRDマニフェストを使用して、を作成します `TridentOrchestrator` CRD。次に、を作成します `TridentOrchestrator` 後でカスタムリソース（Custom Resource）をクリックして、演算子によってインストールをインスタンス化する。

   次のコマンドを実行します。

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

3. のあとに入力します `TridentOrchestrator` CRDが作成され、オペレータの展開に必要な次のリソースを作成します。

   - オペレータのサービスアカウント
   - ClusterRole および ClusterRoleBinding をサービスアカウントにバインドする
   - 専用の PodSecurityPolicy
   - 演算子自体

     Trident インストーラには、これらのリソースを定義するマニフェストが含まれています。デフォルトでは、オペレータはに配置されます `trident` ネームスペース：状況に応じて `trident` ネームスペースが存在しません。次のマニフェストを使用してネームスペースを作成してください。

```
kubectl apply -f deploy/namespace.yaml
```

4. デフォルト以外の名前空間に演算子を配置します `trident` ネームスペースの場合はを更新する必要があります `serviceaccount.yaml`、 `clusterrolebinding.yaml` および `operator.yaml` マニフェストを作成し、を生成します `bundle.yaml`。

   次のコマンドを実行してYAMLマニフェストを更新し、を生成します `bundle.yaml` を使用する `kustomization.yaml`：

```
kubectl kustomize deploy/ > deploy/bundle.yaml
```

   次のコマンドを実行してリソースを作成し、オペレータを配置します。

```
kubectl create -f deploy/bundle.yaml
```

5. 展開後にオペレータのステータスを確認するには、次の手順を実行します。

```
kubectl get deployment -n <operator-namespace>

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

```
kubectl get pods -n <operator-namespace>

NAME                                READY   STATUS             RESTARTS
AGE
trident-operator-54cb664d-lnjxh     1/1     Running            0
3m
```

オペレータによる導入で、クラスタ内のいずれかのワーカーノードで実行されるポッドが正常に作成されます。

> Kubernetes クラスタには、オペレータのインスタンスが * 1 つしか存在しないようにしてください。Trident のオペレータが複数の環境を構築することは避けてください。

手順3：作成 `TridentOrchestrator` Tridentをインストール

これで、オペレータを使って Astra Trident をインストールする準備ができました。これには作成が必要です `TridentOrchestrator`。Tridentのインストーラには、作成用の定義例が付属しています `TridentOrchestrator`。これがの設置作業から始まります `trident` ネームスペース：

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident
Name:         trident
Namespace:
Labels:       <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:22.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Tridentオペレータは、の属性を使用して、Astra Tridentのインストール方法をカスタマイズできます `TridentOrchestrator` 仕様を参照してください "Trident の導入をカスタマイズ"。

のステータス `TridentOrchestrator` インストールが正常に完了したかどうかを示し、インストールされているTridentのバージョンが表示されます。

ステータス	説明
インストール中です	このツールを使用してAstra Tridentをインストールしている `TridentOrchestrator` CR。
インストール済み	Astra Trident のインストールが完了しました。
アンインストール中です	OperatorはAstra Tridentをアンインストールしています。理由はです `spec.uninstall=true`。
アンインストール済み	Astra Trident がアンインストールされました。
失敗しました	オペレータがインストール、パッチ適用、アップデート、またはアンインストールできませんでした Astra Trident。オペレータはこの状態からのリカバリを自動的に試行します。この状態が解消されない場合は、トラブルシューティングが必要です。
更新中です	オペレータが既存のインストールを更新しています。
エラー	。 `TridentOrchestrator` は使用されません。もう一つ 存在します。

インストール中、のステータス `TridentOrchestrator` からの変更 `Installing` 終了： `Installed`。を確認した場合は `Failed` ステータスとオペレータは単独で回復できません。オペレータのログを確認する必要があります。を参照してください "トラブルシューティング" セクション。

Astra Trident のインストールが完了しているかどうかを確認するには、作成したポッドを確認します。

```
kubectl get pod -n trident

NAME                                  READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw          5/5     Running   0           1m
trident-csi-mr6zc                     2/2     Running   0           1m
trident-csi-xrp7w                     2/2     Running   0           1m
trident-csi-zh2jt                     2/2     Running   0           1m
trident-operator-766f7b8658-ldzsv     1/1     Running   0           3m
```

を使用することもできます `tridentctl` インストールされているAstra Tridentのバージョンを確認します。

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

これで、バックエンドを作成できます。を参照してください "導入後のタスク"。

> 導入時の問題のトラブルシューティングについては、を参照してください "トラブルシューティング" セクション。

Trident オペレータの環境をカスタマイズ

Tridentオペレータは、の属性を使用してAstra Tridentのインストールをカスタマイズできます `TridentOrchestrator` 仕様

インストールをカスタマイズする場合は、それ以上のカスタマイズが必要です `TridentOrchestrator` arguments allow、の使用を検討する必要があります `tridentctl` 必要に応じて変更できるカスタムYAMLマニフェストを生成します。

> `spec.namespace` は、で指定します `TridentOrchestrator` Astra Tridentがインストールされているネームスペースを示します。このパラメータ * は、 Astra Trident のインストール後に更新できません * 。これを実行すると、が実行されます `TridentOrchestrator` ステータスをに変更します `Failed`。Astra Tridentは、ネームスペース間での移行を意図していません。

設定オプション

このテーブルの詳細 `TridentOrchestrator` 属性：

パラメータ	説明	デフォルト
`namespace`	Astra Trident をインストールするネームスペース	デフォルト
`debug`	Astra Trident のデバッグを有効にします	いいえ
`windows`	をに設定します `true` Windowsワーカーノードへのインストールを有効にします。	いいえ
`IPv6`	IPv6 経由の Astra Trident をインストール	いいえ
`k8sTimeout`	Kubernetes 処理のタイムアウト	30 秒
`silenceAutosupport`	AutoSupportバンドルをNetAppに送信しない 自動	いいえ
`enableNodePrep`	ワーカーノードの依存関係を自動的に管理（ * beta * ）	いいえ
`autosupportImage`	AutoSupport テレメトリのコンテナイメージ	「NetApp/trident-autosupport：22.10.0」

パラメータ	説明	デフォルト
`autosupportProxy`	AutoSupportを送信するためのプロキシのアドレス/ポート テレメータ	"http://proxy.example.com:8888"
`uninstall`	Astra Trident のアンインストールに使用するフラグ	いいえ
`logFormat`	Astra Trident のログ形式が使用[text 、 JSON]	テキスト（ Text ）
`tridentImage`	インストールする Astra Trident イメージ	「 NetApp / Trident ： 21.04 」
`imageRegistry`	形式の内部レジストリへのパス `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage（Kubernetes 1.19以降）" またはQuay.io/k8scsi
`kubeletDir`	ホスト上の kubelet ディレクトリへのパス	「 /var/lib/kubelet 」
`wipeout`	完全な削除を実行するために削除するリソースのリスト Astra Trident	
`imagePullSecrets`	内部レジストリからイメージをプルするシークレット	
`controllerPluginNodeSelector`	Trident Controller CSI プラグインを実行しているポッドの追加ノードセレクタ。pod.spec.nodeSelector と同じ形式を使用します。	デフォルトはありません。オプションです
`controllerPluginTolerations`	Trident Controller CSI プラグインを実行しているポッドに対する許容値を上書きします。POD .spec.Toleratations と同じ形式を使用します。	デフォルトはありません。オプションです
`nodePluginNodeSelector`	Trident ノード CSI プラグインを実行しているポッドの追加ノードセレクタ。pod.spec.nodeSelector と同じ形式を使用します。	デフォルトはありません。オプションです
`nodePluginTolerations`	Trident Node CSI プラグインを実行しているポッドに対する許容値を上書きします。POD .spec.Toleratations と同じ形式を使用します。	デフォルトはありません。オプションです

ポッドパラメータの書式設定の詳細については、を参照してください "ポッドをノードに割り当てます"。

構成例

上記の属性は、を定義するときに使用できます `TridentOrchestrator` をクリックして、インストールをカスタマイズします。

例**1**：基本的なカスタム構成

次に、基本的なカスタム構成の例を示します。

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

例**2**：ノードセレクタを使用して導入します

次の例では、ノードセレクタを使用してTridentを導入する方法を示します。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

例**3**：**Windows**ワーカーノードに導入する

この例は、Windowsワーカーノードへの導入を示しています。

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