Trident オペレータとともに導入

Astra Trident

NetApp
April 16, 2024

This PDF was generated from https://docs.netapp.com/ja-jp/trident-2207/trident-get-started/kubernetes-customize-deploy.html on April 16, 2024. Always check docs.netapp.com for the latest.

目次

Trident オペレータとともに導入 1
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 1
Trident オペレータを手動で導入 2
Trident オペレータの環境をカスタマイズ 7

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。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がを提供 "バンドル.yaml" オペレータのインストールおよび関連オブジェクトの作成に使用できるファイル。

> 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。

必要なもの

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。

- Kubernetes 1.19~1.24を実行するサポート対象のKubernetesクラスタに対するすべての権限が必要です。
- 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。

- Kubernetes 1.19 ~ 1.24
- 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 <release-name> netapp-trident/trident-operator --version
22.4.0 --create-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 22.7.0 --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

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/v22.04.0/trident-
   installer-22.04.0.tar.gz
   tar -xf trident-installer-22.04.0.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:21.04
    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	オペレータは Astra Trident をインストール、パッチ適用、更新、またはアンインストールできませんでした。オペレータはこの状態からのリカバリを自動的に試みます。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。別のファイルがすでに存在します。

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が単独でリカバリできない場合は、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のインストール方法をカスタマイズできます `TridentOrchestrator` 仕様

属性のリストについては、次の表を参照してください。

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
`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 バンドルをネットアップに自動的に送信しない	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 ： 21.04.0 」
`autosupportProxy`	AutoSupport テレメトリを送信するプロキシのアドレス / ポート	"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]`	"k83.gcr.io/sig-storage (k8s 1.19+) またはQua.io/k8scsi"
`kubeletDir`	ホスト上の kubelet ディレクトリへのパス	「 /var/lib/kubelet 」
`wipeout`	Astra Trident を完全に削除するために削除するリソースのリスト	

パラメータ	説明	デフォルト
`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

`spec.namespace` は、で指定します `TridentOrchestrator` どのネームスペースAstra Tridentがにインストールされているかを示します。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、のステータスがになります `TridentOrchestrator` 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たものではありません。

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"。

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。次に例を示します。

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

次に、ノードセレクタを使用して Trident を導入する例を示します。

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

インストールのカスタマイズを検討している場合は、それ以上にカスタマイズする必要があります `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。

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権を有し、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権については、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。