はじめに
Astra Trident

NetApp
April 16, 2024

This PDF was generated from https://docs.netapp.com/ja-jp/trident-2201/trident-get-started/quickstart.html on April 16, 2024. Always check docs.netapp.com for the latest.

目次

はじめに . . . 1
ぜひお試しください . . . 1
要件 . . . 1
導入の概要 . . . 6
Trident オペレータとともに導入 . . . 9
tridentctl を使用して導入します . . . 17
次の手順 . . . 20

はじめに

ぜひお試しください

ネットアップでは、リクエストに応じてすぐに使用できるラボイメージを提供しています "ネットアップのテスト用ドライブ"。テストドライブは、 3 ノードの Kubernetes クラスタと Astra Trident がインストールおよび設定されたサンドボックス環境を提供します。Astra Trident をよく理解し、機能を調べるのに最適な方法です。

もう 1 つのオプションは、を参照することです "kubeadm インストールガイド" Kubernetes が提供します。

> 本番環境では、この手順で構築した Kubernetes クラスタを使用しないでください。本番環境向けのクラスタを作成するには、ディストリビューションに付属の本番環境導入ガイドを使用します。

Kubernetes を初めて使用する場合は、概念とツールについて理解しておいてください "こちらをご覧ください"。

要件

サポートされるフロントエンド、バックエンド、およびホスト構成を確認することから始めましょう。

> Trident が使用するポートについては、を参照してください "こちらをご覧ください"。

サポートされるフロントエンド（オーケストレーションツール）

Trident Astra は、次のような複数のコンテナエンジンとオーケストレーションツールをサポート

- Anthos オンプレミス（ VMware ）と Anthos は、ベアメタル 1.8 、 1.9 、 1.10 上に実装されています
- Kubernetes 1.17 以降（最新： 1.23 ）
- Mirantis Kubernetes Engine 3.4
- OpenShift 4.7 、 4.8 、 4.9

Trident オペレータは、次のリリースでサポートされています。

- Anthos オンプレミス（ VMware ）と Anthos は、ベアメタル 1.8 、 1.9 、 1.10 上に実装されています
- Kubernetes 1.17 以降（最新： 1.23 ）
- OpenShift 4.7 、 4.8 、 4.9

> Red Hat OpenShift Container Platform のユーザは、 4.1.8 よりも前のバージョンを使用している場合、 initiatorname.iscsi ファイルが空白になる可能性があります。これは RedHat によって識別されているバグで、 OpenShift 4.1.8 で修正されています。を参照してください "バグ修正のお知らせ"。ネットアップでは、 OpenShift 4.6.8 以降で Astra Trident を使用することを推奨しています。

Astra Trident は、 Google Kubernetes Engine （ GKE ）、 Amazon Elastic Kubernetes Services （ EKS ）、

Azure Kubernetes Service （ AKS ）、 Rancher 、 VMware Tanzu Portfolio など、フルマネージドで自己管理型の Kubernetes サービスが数多く提供されています。

サポートされるバックエンド（ストレージ）

Astra Trident を使用するには、次のバックエンドを 1 つ以上サポートする必要があります。

- NetApp ONTAP 対応の Amazon FSX
- Azure NetApp Files の特長
- Astra データストア
- Cloud Volumes ONTAP
- Cloud Volumes Service for GCP
- FAS/AFF / Select 9.3 以降
- ネットアップオール SAN アレイ（ ASA ）
- NetApp HCI / Element ソフトウェア 11 以降

機能の要件

次の表は、このリリースの Astra Trident で利用できる機能と、サポートする Kubernetes のバージョンをまとめたものです。

フィーチャー（ **Feature** ）	**Kubernetes** のバージョン	フィーチャーゲートが必要ですか？
CSI Trident	1.17 以降	いいえ
ボリューム Snapshot	1.17 以降	いいえ
ボリューム Snapshot からの PVC	1.17 以降	いいえ
iSCSI PV のサイズ変更	1.17 以降	いいえ
ONTAP 双方向 CHAP	1.17 以降	いいえ
動的エクスポートポリシー	1.17 以降	いいえ
Trident のオペレータ	1.17 以降	いいえ
自動ワーカーノード準備（ベータ版）	1.17 以降	いいえ
CSI トポロジ	1.17 以降	いいえ

テスト済みのホストオペレーティングシステム

デフォルトでは、 Astra Trident はコンテナで実行されるため、任意の Linux ワーカーで実行されます。ただし、その場合、使用するバックエンドに応じて、標準の NFS クライアントまたは iSCSI イニシエータを使用して Astra Trident が提供するボリュームをマウントできる必要があります。

Astra Trident は、特定のオペレーティングシステムを正式にサポートしているわけではありませんが、次の Linux ディストリビューションが動作することはわかっています。

- OpenShift Container Platform でサポートされている Red Hat CoreOS （ RHCOS ）バージョン
- RHEL または CentOS 7.4 以降
- Ubuntu 18.04 以降

tridentctl ユーティリティーは ' これらの Linux ディストリビューションでも動作します

ホストの設定

使用しているバックエンドによっては、 NFS や iSCSI のユーティリティをクラスタ内のすべてのワーカーにインストールする必要があります。を参照してください "こちらをご覧ください" を参照してください。

ストレージシステムの構成：

Trident を使用するには、バックエンド構成でストレージシステムを使用する前に、一部の変更が必要になることがあります。を参照してください "こちらをご覧ください" を参照してください。

コンテナイメージと対応する **Kubernetes** バージョン

エアギャップのある環境では、 Astra Trident のインストールに必要なコンテナイメージを次の表に示します。tridentctl images コマンドを使用して ' 必要なコンテナイメージのリストを確認します

Kubernetes のバージョン	コンテナイメージ
v1.17.0	• NetApp / Trident ： 22.01.1 • NetApp / Trident -autosupport ： 22.01. • k81.gcr.io/sig-storage/csi-Provisioner ： v2.2.2 • k83.GCR.IO/sig-storage/csi-attacher: v3.4.0 • k81.gcr.io/sig-storage/csi-resizer ： v1.3.0 • k83.gcr.io/sig-storage/csi-snapshotter ： v3.0.3 • k8s 。 gcr.io/sig-storage/csi-node-driver-registrar: v2.4.0 • NetApp/trident-operator ： 22.01.1 （オプション）

Kubernetes のバージョン	コンテナイメージ
v1.18.0	• NetApp / Trident ： 22.01.1 • NetApp / Trident -autosupport ： 22.01. • k81.gcr.io/sig-storage/csi-Provisioner ： v2.2.2 • k83.GCR.IO/sig-storage/csi-attacher: v3.4.0 • k81.gcr.io/sig-storage/csi-resizer ： v1.3.0 • k83.gcr.io/sig-storage/csi-snapshotter ： v3.0.3 • k8s 。 gcr.io/sig-storage/csi-node-driver-registrar: v2.4.0 • NetApp/trident-operator ： 22.01.1 （オプション）
v1.19.0	• NetApp / Trident ： 22.01.1 • NetApp / Trident -autosupport ： 22.01. • k81.gcr.io/sig-storage/csi-Provisioner ： v2.2.2 • k83.GCR.IO/sig-storage/csi-attacher: v3.4.0 • k81.gcr.io/sig-storage/csi-resizer ： v1.3.0 • k83.gcr.io/sig-storage/csi-snapshotter ： v3.0.3 • k8s 。 gcr.io/sig-storage/csi-node-driver-registrar: v2.4.0 • NetApp/trident-operator ： 22.01.1 （オプション）
v1.20.0	• NetApp / Trident ： 22.01.1 • NetApp / Trident -autosupport ： 22.01. • K83.GCR.IO/sig-storage/csi-Provisioner ： v3.1.0 • k83.GCR.IO/sig-storage/csi-attacher: v3.4.0 • k81.gcr.io/sig-storage/csi-resizer ： v1.3.0 • k83.gcr.io/sig-storage/csi-snapshotter ： v3.0.3 • k8s 。 gcr.io/sig-storage/csi-node-driver-registrar: v2.4.0 • NetApp/trident-operator ： 22.01.1 （オプション）

Kubernetes のバージョン	コンテナイメージ
v1.21.0	• NetApp / Trident ： 22.01.1 • NetApp / Trident -autosupport ： 22.01. • K83.GCR.IO/sig-storage/csi-Provisioner ： v3.1.0 • k83.GCR.IO/sig-storage/csi-attacher: v3.4.0 • k81.gcr.io/sig-storage/csi-resizer ： v1.3.0 • k83.gcr.io/sig-storage/csi-snapshotter ： v3.0.3 • k8s 。 gcr.io/sig-storage/csi-node-driver-registrar: v2.4.0 • NetApp/trident-operator ： 22.01.1 （オプション）
v1.22.0	• NetApp / Trident ： 22.01.1 • NetApp / Trident -autosupport ： 22.01. • K83.GCR.IO/sig-storage/csi-Provisioner ： v3.1.0 • k83.GCR.IO/sig-storage/csi-attacher: v3.4.0 • k81.gcr.io/sig-storage/csi-resizer ： v1.3.0 • k83.gcr.io/sig-storage/csi-snapshotter ： v3.0.3 • k8s 。 gcr.io/sig-storage/csi-node-driver-registrar: v2.4.0 • NetApp/trident-operator ： 22.01.1 （オプション）
v1.3.0	• NetApp / Trident ： 22.01.1 • NetApp / Trident -autosupport ： 22.01. • K83.GCR.IO/sig-storage/csi-Provisioner ： v3.1.0 • k83.GCR.IO/sig-storage/csi-attacher: v3.4.0 • k81.gcr.io/sig-storage/csi-resizer ： v1.3.0 • k83.gcr.io/sig-storage/csi-snapshotter ： v3.0.3 • k8s 。 gcr.io/sig-storage/csi-node-driver-registrar: v2.4.0 • NetApp/trident-operator ： 22.01.1 （オプション）

Kubernetes バージョン 1.20 以降では、検証済みの「 k8es.cr.io/sig-storage/csi-snapshotter ： v4.x 」イメージを使用します。これは、「 v1' バージョンが「 volumesnapshotes.snapshot.storage.k88.io`CRD 」に対応している場合にのみ使用します。v1beta` のバージョンが v1beta` のバージョンの有無に関わらず CRD にサービスを提供している場合は、検証済みの「 k83.gcr.io/sig-storage/csi-snapshotter: v3.x' 」イメージを使用します。

導入の概要

Trident の導入には、 Trident のオペレータまたは tridentctl を使用できます。

導入方法を選択します

使用する導入方法を決定するには、次の点を考慮してください。

Trident のオペレータが必要な理由

。 "Trident オペレータ" は、 Astra Trident のリソースを動的に管理し、セットアップフェーズを自動化する優れた方法です。いくつかの前提条件を満たす必要があります。を参照してください "要件"。

Trident オペレータには、以下に示すような利点があります。

自己回復機能

Trident の Astra インストールを監視し、導入が削除されたときや誤って変更された場合など、問題に対処する手段を積極的に講じることができます。オペレータが配備として設定されると 'trident-operator-<generated-id>'POD が作成されますこのポッドは 'TridentOrchestrator CR と Astra Trident のインストールを関連付けて ' 常にアクティブな TridentOrchestrator が 1 つだけになるようにしますつまり、オペレータは、 Astra Trident のインスタンスがクラスタ内に 1 つしかないことを確認し、セットアップを制御して、インストールがべきでないことを確認します。インストールに変更が加えられると（展開またはノードのデミスタなど）、オペレータはそれらを識別し、個別に修正します。

既存のインストール環境を簡単に更新できます

既存の展開をオペレータと簡単に更新できます。「 TridentOrchestrator 」 CR を編集するだけで、インストールを更新できます。たとえば、 Astra Trident を有効にしてデバッグログを生成する必要があるシナリオを考えてみましょう。

これを行うには、「 TridentOrchestrator 」に「 pec.debug` 」を「 true 」に設定するパッチを適用します。

```
kubectl patch torc <trident-orchestrator-name> -n trident --type=merge -p
'{"spec":{"debug":true}}'
```

TridentOrchestrator が更新されると、オペレータは既存のインストールの更新とパッチを処理します。これにより、新しいポッドの作成がトリガーされ、それに応じてインストールが変更される場合があります。

Kubernetes のアップグレードを自動的に処理

Kubernetes バージョンのクラスタをサポート対象バージョンにアップグレードすると、オペレータが既存の

Astra Trident インストールを自動的に更新し、 Kubernetes バージョンの要件を確実に満たすように変更します。

> クラスタがサポート対象外のバージョンにアップグレードされた場合、オペレータによって Astra Trident はインストールされません。Astra Trident がすでにオペレータとともにインストールされている場合、サポート対象外の Kubernetes バージョンに Astra Trident がインストールされていることを示す警告が表示されます。

Helm を使用する必要があるのはなぜですか？

Helm を使用して管理している他のアプリケーションが Astra Trident 21.01 以降である場合は、 Helm を使用して導入を管理することもできます。

いつを使用すればよいか `tridenctl`？

既存の導入環境をにアップグレードする必要がある場合や、高度にカスタマイズする場合は、のを参照してください "Tridentctl"。これは、従来の方法であった Astra Trident を導入する方法です。

導入方法間での移動に関する考慮事項

導入方法を切り替える必要があるシナリオを想像するのは難しいことではありません。tridentctl 配備からオペレータベースの配備に移行する前に ' またはその逆を行う前に ' 次の点を考慮してください

- Astra Trident のアンインストールには、常に同じ方法を使用します。tridentctl を使用して導入した場合は 'tridentctl' バイナリの適切なバージョンを使用して Astra Trident をアンインストールする必要があります同様に、オペレータとともに展開する場合は、「 TridentOrchestrator 」 CR を編集し、「 PEC.uninstall=true 」を設定して Astra Trident をアンインストールする必要があります。
- Astra Trident を削除して tridentctl を使用して Astra Trident を導入するオペレータベースの導入環境の場合は、最初に「 TridentOrchestrator 」を編集し、「 PEC.uninstall=true 」を設定して Astra Trident をアンインストールする必要があります。次に 'TridentOrchestrator とオペレータの配備を削除します次に 'tridentctl を使用してインストールできます
- オペレータベースの手動導入環境で、 Helm ベースの Trident オペレータ環境を使用する場合は、最初に手動でオペレータをアンインストールしてから Helm インストールを実行する必要があります。これにより、 Helm は必要なラベルとアノテーションを使用して Trident オペレータを導入できます。これを行わないと、 Helm ベースの Trident オペレータの導入が失敗し、ラベル検証エラーとアノテーション検証エラーが表示されます。tridentctl ベースの配備を使用している場合は 'Helm ベースの配備を問題なく使用できます

導入モードを理解する

Trident を導入する方法は 3 種類あります。

標準的な導入

Trident を Kubernetes クラスタに導入すると、 Astra Trident インストーラで次の 2 つの作業を実行できます。

- インターネット経由でコンテナイメージを取得しています
- 導入環境とノードのデプロイを作成し、 Kubernetes クラスタ内のすべての対象ノードで Astra Trident ポッドがスピンアップする。

このような標準的な導入は、次の 2 つの方法で実行できます。

- tridentctl install を使用
- Trident 演算子を使用する。Trident オペレータは、手動で導入することも、 Helm を使用して導入することもできます。

このインストールモードは、 Astra Trident をインストールする最も簡単な方法であり、ネットワークの制限を課すことのないほとんどの環境で機能します。

オフラインでの導入

エアギャップ導入を実行するには 'tridentctl install を呼び出すときに --image-registry フラグを使用して ' プライベートイメージレジストリを参照しますTrident 演算子を使用して展開する場合は、「 TridentOrchestrator 」で「 PEC.imageRegistry 」を指定することもできます。このレジストリにはが含まれている必要があります "Trident の画像"、 "Trident AutoSupport の画像"および CSI のサイドカーイメージ（ Kubernetes バージョンで必要な場合）

導入環境をカスタマイズするには 'tridentctl' を使用して Trident のリソースのマニフェストを生成します導入、開始、サービスアカウント、 Astra Trident がインストールの一部として作成するクラスタロールが含まれます。

導入環境のカスタマイズの詳細については、次のリンクを参照してください。

- "オペレータベースの展開をカスタマイズします"
- "85cda507040c129ee606d4d8df583b90"

> プライベート・イメージ・リポジトリを使用している場合は '1.17 より前のバージョンの Kubernetes には '/k8scsi' を ' プライベート・レジストリ URL の末尾には '1.17 より後のバージョンの Kubernetes には '/si' を追加する必要がありますtridentctl 配備にプライベートレジストリを使用する場合は '--trident-image' と --autosupport.image' を --image-registry' とともに使用する必要がありますTrident 演算子を使用して Astra Trident を導入する場合は、 Orchestrator CR にインストールパラメータに「 tridentImage 」と「 autosupportImage 」が含まれていることを確認してください。

リモート導入

次に、リモート導入プロセスの概要を示します。

- Astra Trident を導入するリモートマシンに適切なバージョンの kubectl を導入します。
- Kubernetes クラスタから構成ファイルをコピーし、リモートマシンで「 KUBECONFIG 」環境変数を設定します。
- 「 kubectl get nodes 」コマンドを開始して、必要な Kubernetes クラスタに接続できることを確認します。
- 標準のインストール手順を使用して、リモートマシンからの導入を完了します。

その他の既知の設定オプション

VMware Tanzu Portfolio 製品に Astra Trident をインストールする場合：

- クラスタが特権ワークロードをサポートしている必要があります。
- --kubbelet-dir` フラグは kubelet ディレクトリの場所に設定する必要があります。デフォルトでは、これは /var/vcap/data/kubelet` です。

  --kubbelet-dir' を使用して kubelet の場所を指定することは、 Trident Operator 、 Helm 、および tridentctl の展開で動作することが知られています。

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、手動または Helm を使用して導入できます。

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 1.17 以降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。
- 使用する Kubernetes クラスタを管理するために 'kubectl(OpenShift を使用している場合は 'OC) をインストールして構成した Linux ホストがあります
- 「 KUBECONFIG 」環境変数を、 Kubernetes クラスタ構成を指すように設定しておきます。
- 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。

Helm を使用して Trident オペレータを導入します

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 1.17 以降
- 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. 「 helm install 」コマンドを使用して、展開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
22.1.0 --namespace <trident-namespace>
```

> Trident の名前空間をまだ作成していない場合は '--create-namespace パラメータを helm install コマンドに追加できますHelm によってネームスペースが自動的に作成されます。

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`）: オーバーライドを含む 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 ' デフォルト値の `debug` を変更するには ' 次のように --set` コマンドを実行します

```
$ helm install <name> netapp-trident/trident-operator --version 22.1.0
--set tridentDebug=true
```

Helm チャートの一部である `values] の .yaml ファイルには、キーのリストとデフォルト値が表示されます。

「 helm list 」は、名前、名前空間、グラフ、ステータスなど、インストールに関する詳細を表示します。 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、以降のすべての例で「 kubectl 」ではなく「 OC 」を使用し、「 OC login-u SYSTEM ： admin 」または「 OC login-u kube-admin 」を実行して最初に「 *system:admin 」としてログインします。

Kubernetes のバージョンが 1.17 よりも新しいかどうかを確認するには、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 オペレータを使用して Trident をインストールするには、「 TridentOrchestrator 」カスタムリソース定義（ CRD ）を作成し、その他のリソースを定義する必要があります。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. の最新バージョンをダウンロードします "Trident インストーラバンドル" _Downloads_section から抽出します

   ```
   wget https://github.com/NetApp/trident/releases/download/v21.04/trident-
   installer-21.04.tar.gz
   tar -xf trident-installer-21.04.tar.gz
   cd trident-installer
   ```

2. 適切な CRD マニフェストを使用して、「 TridentOrchestrator 」 CRD を作成します。次に、後で「 TridentOrchestrator 」カスタムリソースを作成して、オペレータによってインストールをインスタンス化します。

   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. 「 TridentOrchestrator 」 CRD が作成されたら、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 ' 演算子は trident' 名前空間に配置されます'trident' 名前空間が存在しない場合は ' 次のマニフェストを使用して名前空間を作成します

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

4. デフォルトの 'trident` 名前空間以外の名前空間に演算子を配備するには ``erviceaccount.yaml 'clusterrolebinding.yaml ' および `operator.yML' マニフェストを更新し 'bundle.yaml を生成する必要があります

   次のコマンドを実行して YAML マニフェストを更新し、「 customizization.yaml 」を使用して「 bundle.yaml 」を生成します。

```
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 'TridentOrchestrator を作成する必要がありますTrident インストーラには 'TridentOrchestrator を作成するための定義例が付属していますこれは 'trident' 名前空間にインストールされます

```
$ 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
    Enable Node Prep:          false
    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 'TridentOrchestrator 仕様の属性を使用して 'Astra Trident のインストール方法をカスタマイズできますを参照してください "Trident の導入をカスタマイズ"。

「 TridentOrchestrator 」のステータスは、インストールが成功したかどうかを示し、インストールされている Trident のバージョンを表示します。

ステータス	説明
インストール中です	オペレータは、この「 TridentOrchestrator 」 CR を使用して Astra Trident をインストールしています。
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	オペレータは 'stra Trident をアンインストールしていますこれは 'pec.uninstall=true だからです
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。

インストール中に 'TridentOrchestrator のステータスが Installing から Installed に変わります「失敗」ステータスが表示され、オペレータが自身で回復できない場合は、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
```

また 'tridentctl を使用して 'Astra Trident のバージョンを確認することもできます

```
$ ./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 'TridentOrchestrator 仕様の属性を使用して 'Astra Trident のインストール方法をカスタマイズできます

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 」を参照してください	Astra Trident をインストールするネームスペース	デフォルト
「バグ」	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
'ilenceAutosupport	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 」を参照してください	AutoSupport テレメトリのコンテナイメージ	「 NetApp/trident-autosupport ： 21.04.0 」
「 autosupportProxy 」と入力します	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 」のように入力します	インストールする Astra Trident イメージ	「 NetApp / Trident ： 21.04 」
「 imageRegistry 」と入力します	内部レジストリへのパス。形式は「 <registry fqdn>[:port][/ssubpath]] 」です	"k83.gcr.io/sig-storage (k8s 1.17+) または Qua.io/k8scsi"
「 kubeletDir 」を参照してください	ホスト上の kubelet ディレクトリへのパス	「 /var/lib/kubelet 」
「 wippeout 」	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	
「 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	

パラメータ	説明	デフォルト
controller'PluginNodeSelector	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

「 TridentOrchestrator 」では、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すために pec.namespace` が指定されています。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、「 TridentOrchestrator 」のステータスが「 Failed 」に変わります。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。

自動ワーカーノードの前処理は、非本番環境でのみ使用することを目的とした * ベータ機能です。

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"。

「 TridentOrchestrator 」を定義するときに上記の属性を使用して、インストールをカスタマイズできます。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
```

「 TridentOrchestrator 」引数で許可される範囲を超えてインストールをカスタマイズする場合は、「 tridentctl 」を使用して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することを検討してください。

tridentctl を使用して導入します

tridentctl を使用して Astra Trident を導入できます。

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。

展開をカスタマイズするには、を参照してください "こちらをご覧ください"。

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。

- 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。
- 使用する Kubernetes クラスタを管理するために 'kubectl(OpenShift を使用している場合は 'OC) をインストールして構成した Linux ホストがあります
- 「 KUBECONFIG 」環境変数を、 Kubernetes クラスタ構成を指すように設定しておきます。
- 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。

導入環境のカスタマイズについては、を参照してください "こちらをご覧ください"。

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、以降のすべての例で「 kubectl 」の代わりに「 OC 」を使用します。まず、「 OC login-u SYSTEM ： admin 」または「 OC login-u kube-admin 」を実行して、「 admin 」としてログインする必要があります。

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>
```

Kubernetes サーバのバージョンを確認します。このポートは、 Astra Trident のインストール時に使用します。

手順 2 ：インストーラをダウンロードして展開します

> Trident インストーラは Trident ポッドを作成し、そのステートを維持するために使用される CRD オブジェクトを構成し、プロビジョニングやクラスタホストへのボリュームの接続などのアクションを実行する CSI サイドカーを初期化します。

の最新バージョンをダウンロードできます "Trident インストーラバンドル" _Downloads_section から ' 抽出します

たとえば、最新バージョンが 21.07.1 の場合は、次のようになります。

```
wget https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
tar -xf trident-installer-21.07.1.tar.gz
cd trident-installer
```

手順 3 ： Astra Trident をインストールする

tridentctl install コマンドを実行して 'Astra Trident を目的の名前空間にインストールします

```
$ ./tridentctl install -n trident
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-csi-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=21.07.1
INFO Trident installation succeeded.
....
```

インストーラが完了すると、次のように表示されます。Kubernetes クラスタ内のノードの数によっては、ポッドをさらに確認することもできます。

```
$ kubectl get pod -n trident
NAME                            READY   STATUS    RESTARTS   AGE
trident-csi-679648bd45-cv2mx    4/4     Running   0          5m29s
trident-csi-vgc8n               2/2     Running   0          5m29s

$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.1        | 21.07.1        |
+----------------+----------------+
```

上記の例のような出力が表示された場合、この手順は完了していますが、 Astra Trident の設定はまだ完了していません。次の手順に進みます。を参照してください "導入後のタスク"。

ただし、インストーラが正常に完了しなかった場合、または * running * trident-csi-< generated id> が表示されない場合は、プラットフォームがインストールされていません。

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。

tridentctl 展開をカスタマイズします

Trident インストーラを使用して属性をカスタマイズできます。たとえば、 Trident イメージをプライベートリポジトリにコピーした場合、 -trident-image' を使用してイメージ名を指定できます。Trident イメージと必要な CSI サイドカーイメージをプライベートリポジトリにコピーした場合は、 --image_registry' スイッチを使用してリポジトリの場所を指定することをお勧めします。この場合、形式は「 <registry fqdn>[:port] 」です。

Astra Trident が自動的にワーカーノードを構成するには、「 --enable-node-prep 」を使用します。この機能の詳細については、を参照してください "こちらをご覧ください"。

ワーカーノードの自動準備は * ベータ機能 * で、非本番環境でのみ使用できます。

通常の「 /var/lib/kubelet 」以外のパスに「 kubelet 」がデータを保持している Kubernetes の配布を使用する場合は、「 --kubelet -dir 」を使用して代替パスを指定できます。

インストーラの引数で許可される範囲を超えてインストールをカスタマイズする必要がある場合は、配置ファイルをカスタマイズすることもできます。--generate-custom-yaml パラメータを使用して、インストーラの「 etup 」ディレクトリに次の YAML ファイルを作成します。

- trident-clusterrolebinding.yaml
- trident-deployment.yaml
- trident-CRDs .YAML
- trident-clusterrolment.yaml
- trident-demimonimon.yamml`
- trident-service.yaml
- trident-namespac.yaml
- trident-ServiceAccount.yaml

これらのファイルを生成したら、必要に応じて変更し、「 --use-custom-yaml 」を使用してカスタム展開をインストールできます。

```
./tridentctl install -n trident --use-custom-yaml
```

次の手順

Astra Trident の導入が完了したら、バックエンドの作成、ストレージクラスの作成、ボリュームのプロビジョニング、ポッドでのボリュームのマウントを実行できます。

手順 1 ：バックエンドを作成する

これで、 Astra Trident がボリュームのプロビジョニングに使用するバックエンドを作成できるようになります。そのためには、必要なパラメータを含む「 backend.json 」ファイルを作成します。さまざまなバックエンドタイプのサンプル構成ファイルは 'sample-input' ディレクトリにあります

を参照してください "こちらをご覧ください" バックエンドタイプのファイルを設定する方法の詳細については、を参照してください。

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

```
./tridentctl -n trident create backend -f backend.json
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+
```

作成に失敗した場合は、バックエンド設定に何か問題があります。次のコマンドを実行すると、ログを表示して原因を特定できます。

```
./tridentctl -n trident logs
```

問題に対処したら、この手順の最初に戻ってやり直してください。トラブルシューティングのヒントについては、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。

手順 2 ：ストレージクラスを作成する

Kubernetes ユーザは、を指定する Persistent Volume クレーム（ PVC ）を使用してボリュームをプロビジョニングします "ストレージクラス" 名前で検索できます。詳細情報はユーザには表示されませんが、ストレージクラスは、そのクラスに使用されるプロビジョニングツール（この場合は Trident ）と、そのクラスがプロビジョニングツールにもたらす意味を特定します。

ストレージクラスの Kubernetes ユーザがボリュームを必要なときに指定するストレージクラスを作成します。このクラスの構成では、前の手順で作成したバックエンドをモデリングし、 Astra Trident が新しいボリュームのプロビジョニングにこのバックエンドを使用するようにする必要があります。

最もシンプルなストレージクラスは ' インストーラに付属の 'sample -input/storage-class-csl.yamml.templ' ファイルに基づいていますこれは ' *backend_type_* をストレージドライバ名に置き換えます

```
./tridentctl -n trident get backend
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

これは Kubernetes オブジェクトなので 'kubectl' を使用して Kubernetes で作成します

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

Kubernetes と Astra Trident の両方で、 * basic-csi * ストレージクラスが表示され、 Astra Trident がバックエンドのプールを検出しました。

```
kubectl get sc basic-csi
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

手順 3 ：最初のボリュームをプロビジョニングします

これで、最初のボリュームを動的にプロビジョニングできます。これは Kubernetes を作成することで実現されます "永続的ボリュームの要求" （ PVC ）オブジェクト。

作成したストレージクラスを使用するボリュームの PVC を作成します。

例については 'sample-input/pvc-basic-csI.yaml を参照してくださいストレージクラス名が、作成した名前と一致していることを確認します。

```
kubectl create -f sample-input/pvc-basic-csi.yaml

kubectl get pvc --watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic          1s
basic     Pending   pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   0
basic          5s
basic     Bound     pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   1Gi
RWO           basic          7s
```

手順 4 ：ボリュームをポッドにマウントする

次に、ボリュームをマウントします。次に '/usr/share/nginx/html' の下に PV をマウントする nginx ポッドを起動します

```
cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

この時点でポッド（アプリケーション）は存在しなくなりますが、ボリュームはまだ存在しています。必要に応じて、別のポッドから使用できます。

ボリュームを削除するには、要求を削除します。

```
kubectl delete pvc basic
```

これで、次のような追加タスクを実行できます。

- "追加のバックエンドを設定"
- "追加のストレージクラスを作成する。"

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。