はじめに
Astra Trident

NetApp
April 16, 2024

目次

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"。

試乗について学びます

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"。

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

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：ベアメタル1.9、1.10、1.11
- Kubernetes 1.20~1.25
- Mirantis Kubernetes Engine 3.5
- OpenShift 4.8、4.9、4.10、4.11

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：ベアメタル1.9、1.10、1.11
- Kubernetes 1.20~1.25
- OpenShift 4.8、4.9、4.10、4.11

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
- 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、このリリースので使用できる機能をまとめたものです。
Astra Tridentと、それがサポートするKubernetesのバージョン

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	1.20-1.25	いいえ
ボリューム Snapshot	1.20-1.25	いいえ
ボリューム Snapshot からの PVC	1.20-1.25	いいえ
iSCSI PV のサイズ変更	1.20-1.25	いいえ
ONTAP 双方向 CHAP	1.20-1.25	いいえ

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？
動的エクスポートポリシー	1.20-1.25	いいえ
Trident のオペレータ	1.20-1.25	いいえ
自動ワーカーノード準備（ベータ版）	1.20-1.25	いいえ
CSI トポロジ	1.20-1.25	いいえ

テスト済みのホストオペレーティングシステム

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、動作確認済みの特徴は次のとおりです。

- 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.
- Ubuntu 18.04以降（最新22.04）
- Windows Server 2019

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。

。 `tridentctl` ユーティリティは、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。を使用します `tridentctl images` 必要なコンテナイメージのリストを確認するコマンド。

Kubernetes のバージョン	コンテナイメージ
v1.20.0	• docker.io/netapp/trident：22.10.0 • docker.io / netapp/trident-autosupport：22.10 • registry.k8s.io/sig-storage/csi-provisioner：v3.3.0 • registry.k8s.io/sig-storage/csi-attacher：v4.0.0 • registry.k8s.io/sig-storage/csi-resizer：v1.6.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.1.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.5.1 • docker.io/netapp/trident-operator：22.10.0（オプション）
v1.21.0	• docker.io/netapp/trident：22.10.0 • docker.io / netapp/trident-autosupport：22.10 • registry.k8s.io/sig-storage/csi-provisioner：v3.3.0 • registry.k8s.io/sig-storage/csi-attacher：v4.0.0 • registry.k8s.io/sig-storage/csi-resizer：v1.6.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.1.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.5.1 • docker.io/netapp/trident-operator：22.10.0（オプション）
v1.22.0	• docker.io/netapp/trident：22.10.0 • docker.io / netapp/trident-autosupport：22.10 • registry.k8s.io/sig-storage/csi-provisioner：v3.3.0 • registry.k8s.io/sig-storage/csi-attacher：v4.0.0 • registry.k8s.io/sig-storage/csi-resizer：v1.6.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.1.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.5.1 • docker.io/netapp/trident-operator：22.10.0（オプション）

Kubernetes のバージョン	コンテナイメージ
v1.3.0	• docker.io/netapp/trident：22.10.0 • docker.io / netapp/trident-autosupport：22.10 • registry.k8s.io/sig-storage/csi-provisioner：v3.3.0 • registry.k8s.io/sig-storage/csi-attacher：v4.0.0 • registry.k8s.io/sig-storage/csi-resizer：v1.6.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.1.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.5.1 • docker.io/netapp/trident-operator：22.10.0（オプション）
v1.24.0	• docker.io/netapp/trident：22.10.0 • docker.io / netapp/trident-autosupport：22.10 • registry.k8s.io/sig-storage/csi-provisioner：v3.3.0 • registry.k8s.io/sig-storage/csi-attacher：v4.0.0 • registry.k8s.io/sig-storage/csi-resizer：v1.6.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.1.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.5.1 • docker.io/netapp/trident-operator：22.10.0（オプション）
v1.25.0	• docker.io/netapp/trident：22.10.0 • docker.io / netapp/trident-autosupport：22.10 • registry.k8s.io/sig-storage/csi-provisioner：v3.3.0 • registry.k8s.io/sig-storage/csi-attacher：v4.0.0 • registry.k8s.io/sig-storage/csi-resizer：v1.6.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.1.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.5.1 • docker.io/netapp/trident-operator：22.10.0（オプション）

Kubernetesバージョン1.20以降では、検証済みを使用してください `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` イメージは、の場合にのみ作成します `v1` のバージョンがを処理しています `volumesnapshots.snapshot.storage.k8s.gcr.io` CRD。状況に応じて `v1beta1` バージョンは、の有無にかかわらず、CRDに対応しています `v1` バージョン：検証済みを使用します `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` イメージ（Image）：

導入の概要

Tridentのオペレータが、またはと連携してAstra Tridentを導入できます `tridentctl`。

22.04 リリース以降、 Astra Trident がインストールされるたびに AES キーが再生成されなくなりました。今回のリリースでは、 Astra Trident がインストールする新しいシークレットオブジェクトが、インストール全体で維持されます。つまり、 `tridentctl` 22.04では、以前のバージョンのTridentをアンインストールできますが、それより前のバージョンでは22.04のインストールをアンインストールできません。

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

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のオペレータが対応するタイミング

。 "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。オペレータが配置として設定されている場合は、「」を参照してください `trident-operator-<generated-id>` ポッドが作成されました。このポッドでは、を関連付けま

す `TridentOrchestrator` Astra Tridentをインストールした CRでは、常に1つのアクティブな状態が保証されます `TridentOrchestrator`。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。を編集するだけで済みます `TridentOrchestrator` CRを使用してインストールを更新します。
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、にパッチを適用します `TridentOrchestrator` をクリックして設定します `spec.debug` 終了： `true`：

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

実行後 `TridentOrchestrator` が更新され、オペレータが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。

BlueXP（旧**Cloud Manager**）を使用した**Kubernetes**クラスタの管理

を使用 "Astra TridentでBlueXPを使用"では、最新バージョンのAstra Tridentにアップグレードし、ストレージクラスを追加して管理し、作業環境に接続し、Cloud Backup Service を使用して永続的ボリュームをバックアップすることができます。BlueXPは、Tridentオペレータを使用したAstra Tridentの導入を、手動またはHelmを使用してサポートしています。

Helmを使用する状況

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。

を使用する状況 `tridenctl`

既存の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、の使用を検討してください "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。から移動する前に、次の点を考慮してください `tridentctl` オペレータベースの展開への展開、またはその逆の展開：

- 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。を使用してを導入した場合 `tridentctl`を使用する場合は、適切なバージョンのを使用する必要があります `tridentctl` Astra Tridentをアンインストールするためのバイナリ。同様に、演算子を使用してを配置する場合は、を編集する必要があります `TridentOrchestrator` CRおよびSET `spec.uninstall=true` Astra Tridentをアンインストールする方法
- オペレータベースの導入環境で、削除して使用する場合 `tridentctl` Astra Tridentを導入するには、まずを編集する必要があります `TridentOrchestrator` をクリックして設定します `spec.uninstall=true` Astra Tridentをアンインストールする方法次に、を削除します `TridentOrchestrator` オペレータによる導入も可能です。その後、を使用してをインストールできます `tridentctl`。
- 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。を使用する場合は `tridentctl`- Helmベースの展開を使用すると、問題を発生させずに導入できます。

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。

- を使用します `tridentctl install`
- 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

エアギャップ展開を実行するには、を使用します `--image-registry` 呼び出し時にフラグを設定します `tridentctl install` をクリックして、プライベートイメージレジストリを指定します。Tridentのオペレータを使用して導入する場合は、と指定することもできます `spec.imageRegistry` をクリックします `TridentOrchestrator`。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）

を使用して導入をカスタマイズできます `tridentctl` 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"
  *

> プライベートイメージリポジトリを使用する場合は、を追加する必要があります `/sig-storage` プライベートレジストリURLの末尾に移動します。のプライベートレジストリを使用する場合 `tridentctl` は、を使用する必要があります `--trident-image` および `--autosupport-image` と組み合わせて使用します `--image-registry`。Tridentオペレータを使用してAstra Tridentを導入する場合は、Orchestrator CRに含まれていることを確認します `tridentImage` および `autosupportImage` をインストールパラメータに指定します。

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。

- 適切なバージョンのを導入します `kubectl` Astra Tridentの導入元となるリモートマシン。
- Kubernetesクラスタから構成ファイルをコピーし、を設定します `KUBECONFIG` リモートマシンの環境変数。
- を開始します `kubectl get nodes` コマンドを使用して、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。
- 。 `--kubelet-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。デフォルトはです `/var/vcap/data/kubelet`。

  を使用してkubeletの場所を指定します `--kubelet-dir` は、Trident Operator、Helm、およびで動作することがわかっています `tridentctl` 導入：

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」
`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"

パラメータ	説明	デフォルト
`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
```

tridentctl を使用して導入します

を使用して、Astra Tridentを導入できます `tridentctl`。をよく理解しておくことをお勧めします "基本概念"。をカスタマイズします `tridentctl` 配置については、を参照してください "tridentctl 展開をカスタマイズします"。

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

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。

- サポート対象のKubernetesクラスタに対するすべての権限。
- サポートされているネットアップストレージシステムへのアクセス。
- Kubernetesワーカーノードすべて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 `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。

- 。 `KUBECONFIG` 環境変数は、Kubernetesクラスタの構成を指します。
- 。 "Astra Trident に必要な機能ゲート" が有効になります。
- 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"。

ステップ 1 ： Kubernetes クラスタを確認する

Linuxホストにログインし、管理が機能していることを確認します。 "サポートされる Kubernetes クラスタ" また、必要な権限があります。

> OpenShiftで、を使用できます `oc` ではなく `kubectl` 以降に示すすべての例では、を実行して、最初に* system:admin *としてログインする必要があります `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。

Tridentインストーラバンドルの最新バージョンは、からダウンロードして展開できます "GitHub の_Assets_sectionを参照してください"。

たとえば、最新バージョンが22.10.0の場合、次のようになります。

```
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
```

手順 3 ： Astra Trident をインストールする

を実行して、必要なネームスペースにAstra Tridentをインストールします `tridentctl install` コマンドを実行します

```
./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=22.10.0
INFO Trident installation succeeded.
....
```

> WindowsノードでAstra Tridentを実行できるようにするには、を追加します `--windows` インストールコマンドへのフラグ： `$ ./tridentctl install --windows -n trident`。

インストーラが完了すると、次のような出力が表示されます。Kubernetesクラスタ内のノードの数によっては、ポッドがさらに存在する場合があります。

```
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 |
+----------------+----------------+
| 22.10.0        | 22.10.0        |
+----------------+----------------+
```

Astra Tridentの設定を完了するには、に進みます "導入後のタスク"。

インストーラが正常に完了しない場合、または `trident-csi-<generated id>` ステータス* RUNNING * がなく、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"。

tridentctl 展開をカスタマイズします

Astra Tridentインストーラを使用して、導入をカスタマイズできます。

インストーラの詳細を確認してください

Astra 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は、を使用してイメージ名を指定できます `--trident-image`。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、を使用してリポジトリの場所を指定することを推奨します `--image-registry` スイッチ。の形式を指定します `<registry FQDN>[:port]`。

Kubernetesのディストリビューションを使用している場合 `kubelet` データを通常以外のパスに保持します `/var/lib/kubelet` を使用して、代替パスを指定できます `--kubelet-dir`。

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。を使用する `--generate-custom-yaml` パラメータは、インストーラのに次のYAMLファイルを作成します `setup` ディレクトリ：

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`

- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`
- `trident-resourcequota.yaml`

これらのファイルを生成したら、必要に応じて変更し、を使用できます `--use-custom-yaml` をクリックして、カスタム導入環境をインストールします。

```
./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。これを行うには、を作成します `backend.json` 必要なパラメータを含むファイル。さまざまなバックエンドタイプの設定ファイルの例については、を参照してください `sample-input` ディレクトリ。

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。

をベースにしたストレージクラスが最もシンプルになりました `sample-input/storage-class-csi.yaml.templ` インストーラに付属のファイル *`BACKEND_TYPE`* ストレージドライバの名前を指定します。

```
./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、を使用します `kubectl` をクリックして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。

を参照してください `sample-input/pvc-basic-csi.yaml` たとえば、のように指定します。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。nginxポッドを起動し、の下にPVをマウントします `/usr/share/nginx/html`。

```
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。"