はじめに
Astra Trident

NetApp
November 14, 2025

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

Astra Tridentをインストールする前に、次の一般的なシステム要件を確認してください。個 々 のバックエンドには追加の要件がある場合があり

Astra Trident 23.01に関する重要な情報

- Astra Tridentに関する次の重要な情報をお読みください。*

** ：Trident ** に関する重要な情報

- TridentでKubernetes 1.27がサポートされるようになりました。Kubernetesをアップグレードする前にTridentをアップグレード
- Astra Tridentは、SAN環境でマルチパス構成を厳密に使用し、推奨される値をに設定します `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1.12
- Kubernetes 1.21~1.27
- Mirantis Kubernetes Engine 3.5
- OpenShift 4.9 ~ 4.12

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1.12
- Kubernetes 1.21~1.27
- OpenShift 4.9 ~ 4.12

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。

> Astra Tridentがインストールされ ているKubernetesクラスタを1.24から1.25以降にアップグレードする前に、を参照してください "Helm ベースのオペレータインストレーションをアップグレードします"。

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 9.5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	1.21 ~ 1.27	いいえ
ボリューム Snapshot	1.21 ~ 1.27	いいえ
ボリューム Snapshot からの PVC	1.21 ~ 1.27	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？
iSCSI PV のサイズ変更	1.21 ~ 1.27	いいえ
ONTAP 双方向 CHAP	1.21 ~ 1.27	いいえ
動的エクスポートポリシー	1.21 ~ 1.27	いいえ
Trident のオペレータ	1.21 ~ 1.27	いいえ
CSI トポロジ	1.21 ~ 1.27	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ませんが、動作確認済みのものは次のとおりです。

- OpenShift Container Platform（AMD64およびARM64）でサポートされているRed Hat CoreOS（RHCOS）のバージョン
- RHEL 8+（AMD64およびARM64）
- Ubuntu 22.04以降（AMD64およびARM64）
- Windows Server 2019（AMD64）

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

Kubernetesクラスタ内のすべてのワーカーノードが、ポッド用にプロビジョニングしたボリュームをマウントできる必要があります。ワーカーノードを準備するには、ドライバの選択に基づいてNFSツールまたはiSCSIツールをインストールする必要があります。

"ワーカーノードを準備します"

ストレージシステムの構成：

Astra Tridentでは、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、変更が必要になる場合があります。

"バックエンドを設定"

Astra Trident ポート

Astra Tridentが通信するには、特定のポートへのアクセスが必要です。

"Astra Trident ポート"

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
v1.21.0	• Docker.io/NetApp/trident：23.04.0 • docker.io / netapp/trident-autosupport：23.04 • registry.k8s.io/sig-storage/csi-provisioner：v3.4.1 • registry.k8s.io/sig-storage/csi-attacher：v4.2.0 • registry.k8s.io/sig-storage/csi-resizer：v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.7.0 • docker.io/netapp/trident-operator：23.04.0（オプション）
v1.22.0	• Docker.io/NetApp/trident：23.04.0 • docker.io / netapp/trident-autosupport：23.04 • registry.k8s.io/sig-storage/csi-provisioner：v3.4.1 • registry.k8s.io/sig-storage/csi-attacher：v4.2.0 • registry.k8s.io/sig-storage/csi-resizer：v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.7.0 • docker.io/netapp/trident-operator：23.04.0（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：23.04.0 • docker.io / netapp/trident-autosupport：23.04 • registry.k8s.io/sig-storage/csi-provisioner：v3.4.1 • registry.k8s.io/sig-storage/csi-attacher：v4.2.0 • registry.k8s.io/sig-storage/csi-resizer：v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.7.0 • docker.io/netapp/trident-operator：23.04.0（オプション）
v1.24.0	• Docker.io/NetApp/trident：23.04.0 • docker.io / netapp/trident-autosupport：23.04 • registry.k8s.io/sig-storage/csi-provisioner：v3.4.1 • registry.k8s.io/sig-storage/csi-attacher：v4.2.0 • registry.k8s.io/sig-storage/csi-resizer：v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.7.0 • docker.io/netapp/trident-operator：23.04.0（オプション）
v1.25.0	• Docker.io/NetApp/trident：23.04.0 • docker.io / netapp/trident-autosupport：23.04 • registry.k8s.io/sig-storage/csi-provisioner：v3.4.1 • registry.k8s.io/sig-storage/csi-attacher：v4.2.0 • registry.k8s.io/sig-storage/csi-resizer：v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.7.0 • docker.io/netapp/trident-operator：23.04.0（オプション）

Kubernetes のバージョン	コンテナイメージ
v1.26.0	• Docker.io/NetApp/trident：23.04.0 • docker.io / netapp/trident-autosupport：23.04 • registry.k8s.io/sig-storage/csi-provisioner：v3.4.1 • registry.k8s.io/sig-storage/csi-attacher：v4.2.0 • registry.k8s.io/sig-storage/csi-resizer：v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.7.0 • docker.io/netapp/trident-operator：23.04.0（オプション）
v1.27.0	• Docker.io/NetApp/trident：23.04.0 • docker.io / netapp/trident-autosupport：23.04 • registry.k8s.io/sig-storage/csi-provisioner：v3.4.1 • registry.k8s.io/sig-storage/csi-attacher：v4.2.0 • registry.k8s.io/sig-storage/csi-resizer：v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter：v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar：v2.7.0 • docker.io/netapp/trident-operator：23.04.0（オプション）

Kubernetesバージョン1.21以降では、検証済みの `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）：

Astra Trident をインストール

Astra Tridentのインストール方法をご確認ください

ネットアップでは、Astra Tridentをさまざまな環境や組織に導入できるように、複数のインストールオプションを提供しています。Tridentは、Tridentオペレータ（手動またはHelmを使用）またはでインストールできます `tridentctl`。このトピックでは、適切なインストールプロセスを選択するための重要な情報を提供します。

Astra Tridentに関する重要な情報23.04

- Astra Tridentに関する次の重要な情報をお読みください。*

** ：Trident ** に関する重要な情報

- TridentでKubernetes 1.27がサポートされるようになりました。Kubernetesをアップグレードする前にTridentをアップグレード
- Astra Tridentは、SAN環境でマルチパス構成を厳密に使用し、推奨される値をに設定します `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

作業を開始する前に

インストールパスに関係なく、次のものが必要です。

- サポートされているバージョンのKubernetesと機能の要件を有効にして実行されている、サポートされるKubernetesクラスタに対するすべての権限。を確認します "要件" を参照してください。
- 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` Kubernetesクラスタ構成を参照するように設定された環境変数。
- 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"。

に慣れていない場合は "基本概念"今こそ、そのための絶好の機会です。

インストール方法を選択します

適切なインストール方法を選択します。また、に関する考慮事項についても確認しておく必要があります "メソッド間を移動しています" 決定する前に。

Trident演算子を使用する

Tridentのオペレータは、手動で導入する場合でも、Helmを使用する場合でも、Astra Tridentのリソースを動的に管理して簡単にインストールできます。それは可能である "Tridentのオペレータ環境をカスタマイズ" で属性を使用する `TridentOrchestrator` カスタムリソース（CR）。

Tridentオペレータには次のようなメリットがあります。

** Astra Tridentオブジェクト作成**

Tridentオペレータが、Kubernetesのバージョンに応じて次のオブジェクトを自動的に作成します。

- 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（Kubernetes 1.25以前用）
- 演算子自体

**** 自己回復機能****

OperatorはAstra Tridentの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の問題に対処するための手段をアクティブに講じます。A `trident-operator-<generated-id>` ポッドが作成され、が関連付けられます `TridentOrchestrator` Astra Tridentをインストールした CR。これにより、クラスタ内にAstra Tridentのインスタンスが1つだけ存在し、そのセットアップを制御することで、インストールがべき等の状態であることを確認できます。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。

**** は、インストール済みの既存の**** 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**の自動アップグレード処理****

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。

** NetApp Consoleを使用した Kubernetes クラスタ管理**

NetApp Consoleを使用するAstra Tridentでは、最新バージョンのAstra Tridentにアップグレードしたり、ストレージ クラスを追加および管理して作業環境に接続したり、 Cloud Backup Serviceを使用して永続ボリュームをバックアップしたりできます。コンソールは、手動または Helm を使用して、 Tridentオペレーターを使用したAstra Trident のデプロイメントをサポートします。

を使用します `tridentctl`

既存の環境を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ことを検討している場合は、アップグレードを検討する必要があります 。これは、従来の方法であった Astra Trident を導入する方法です。

可能です Tridentリソースのマニフェストを生成するには、次の手順を実行します導入、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。

> 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。
> 適切なインストール方法_を選択します。

インストールモードを選択します

組織で必要な_インストールモード_（標準、オフライン、またはリモート）に基づいて導入プロセスを決定します。

標準インストール

これは、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。標準インストールモードでは、必要なTridentを格納するためにデフォルトのレジストリが使用されます (`docker.io`）とCSIを参照してください (`registry.k8s.io`）イメージ。

標準モードを使用すると、Astra Tridentインストーラは次のように動作します。

- インターネット経由でコンテナイメージを取得します
- 導入環境または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します

オフラインインストール

オフラインインストールモードは、エアギャップまたは安全な場所で必要になる場合があります。このシナリオでは、必要なTridentイメージとCSIイメージを格納するために、1つのプライベートなミラーリングされたレジストリ、または2つのミラーリングされたレジストリを作成できます。

CSIイメージは、レジストリ設定に関係なく、1つのレジストリに存在する必要があります。

リモートインストール

次に、リモートインストール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。

メソッドとモードに基づいてプロセスを選択します

決定が終わったら、適切なプロセスを選択します。

メソッド	インストールモード
Tridentのオペレータ（手動）	"標準インストール" "オフラインインストール"
Tridentオペレータ（Helm）	"標準インストール" "オフラインインストール"
`tridentctl`	"標準インストールまたはオフラインインストール"

インストール方法を切り替える

インストール方法を変更することもできます。その前に、次の点を考慮してください。

- Astra Tridentのインストールと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、削除して代わりにを使用する場合は `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。

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オペレータを使用してインストール

Tridentオペレータを手動で導入（標準モード）

Tridentオペレータが手動で導入してAstra Tridentをインストールできます。このプロセスでは、環境 をインストールする際に、Astra Tridentで必要なコンテナイメージがプライベートレジストリに格納されません。プライベートイメージレジストリがある場合は、を使用します "オフライン導入のプロセス"。

Astra Tridentに関する重要な情報**23.04**

- Astra Tridentに関する次の重要な情報をお読みください。*

** ：Trident ** に関する重要な情報

- TridentでKubernetes 1.27がサポートされるようになりました。Kubernetesをアップグレードする前にTridentをアップグレード
- Astra Tridentは、SAN環境でマルチパス構成を厳密に使用し、推奨される値をに設定します `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オペレータを手動で導入し、**Trident**をインストール

レビュー "インストールの概要" インストールの前提条件を満たし、環境に適したインストールオプションを選択していることを確認します。

作業を開始する前に

インストールを開始する前に、Linuxホストにログインして、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" 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`。

1. Kubernetesのバージョンを確認します。

```
kubectl version
```

2. クラスタ管理者の権限を確認します。

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

3. 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ことを確認します。

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

手順**1**：**Trident**のインストーラパッケージをダウンロード

Astra Tridentインストーラパッケージには、Tridentオペレータの導入とAstra Tridentのインストールに必要なものがすべて含まれています。から最新バージョンのTridentインストーラ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/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

手順**2**：を作成します `TridentOrchestrator` **CRD**

を作成します `TridentOrchestrator` カスタムリソース定義（CRD）。を作成します `TridentOrchestrator` カスタムリソース。で適切なCRD YAMLバージョンを使用します `deploy/crds` を作成します `TridentOrchestrator` CRD。

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

手順**3**：**Trident**のオペレータを導入する

Astra Tridentインストーラには、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バンドルファイルが用意されています。このバンドルファイルを使用すると、オペレータを簡単に導入し、デフォルトの設定でAstra Tridentをインストールできます。

- 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

- デフォルトでは、Tridentのインストーラによって `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。次を使用して作成してください：

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

- オペレータを以外のネームスペースに配置する場合 `trident` 名前空間、更新 `serviceaccount.yaml`、 `clusterrolebinding.yaml` および `operator.yaml` を使用してバンドルファイルを生成します `kustomization.yaml`。

  a. を作成します `kustomization.yaml` 次のコマンドを使用して、*<bundle>* is `bundle_pre_1_25` または `bundle_post_1_25` 使用しているKubernetesのバージョンに基づきます。

  ```
  cp kustomization_<bundle>.yaml kustomization.yaml
  ```

  b. 次のコマンドを使用してバンドルをコンパイルします。WHERE_STORE_IS <bundle> `bundle_pre_1_25` または `bundle_post_1_25` 使用しているKubernetesのバージョンに基づきます。

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

手順

1. 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
```

2. operator、deployment、およびReplicaSetsが作成されたことを確認します。

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

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**4**： `TridentOrchestrator` **Trident**をインストール

これで、を作成できます `TridentOrchestrator` Astra Tridentを導入必要に応じて、を実行できます "Tridentのインストールをカスタマイズ" で属性を使用する `TridentOrchestrator` 仕様

```
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:23.04
    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:23.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

インストールを確認します。

インストールを確認するには、いくつかの方法があります。

を使用します `TridentOrchestrator` ステータス

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。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。 "ログをチェックしてください"。

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。

ポッドの作成ステータスを使用する

作成したポッドのステータスを確認することで、Astra Tridentのインストールが完了したかどうかを確認できます。

```
kubectl get pods -n trident

NAME                                  READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw   6/6     Running   0
1m
trident-node-linux-mr6zc              2/2     Running   0
1m
trident-node-linux-xrp7w              2/2     Running   0
1m
trident-node-linux-zh2jt              2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv     1/1     Running   0
3m
```

を使用します `tridentctl`

を使用できます `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 |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

次のステップ

できるようになりました。 "バックエンドとストレージクラスを作成し、ボリュームをプロビジョニングして、ポッドにボリュームをマウントします"。

Tridentオペレータを手動で導入（オフラインモード）

Tridentオペレータが手動で導入してAstra Tridentをインストールできます。このプロセスでは、環境 をインストールする際に、Astra Tridentで必要なコンテナイメージがプライベートレジストリに格納されます。プライベートイメージレジストリがない場合は、を使用します "標準的な導入のプロセス"。

Astra Tridentに関する重要な情報**23.04**

- Astra Tridentに関する次の重要な情報をお読みください。*

** ：Trident ** に関する重要な情報

- TridentでKubernetes 1.27がサポートされるようになりました。Kubernetesをアップグレードする前にTridentをアップグレード
- Astra Tridentは、SAN環境でマルチパス構成を厳密に使用し、推奨される値をに設定します `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オペレータを手動で導入し、**Trident**をインストール

レビュー "インストールの概要" インストールの前提条件を満たし、環境に適したインストールオプションを選択していることを確認します。

作業を開始する前に

Linuxホストにログインして、管理が機能していることとを確認します "サポートされる Kubernetes クラスタ" 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`。

1. Kubernetesのバージョンを確認します。

```
kubectl version
```

2. クラスタ管理者の権限を確認します。

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

3. 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ことを確認します。

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

手順**1**：**Trident**のインストーラパッケージをダウンロード

Astra Tridentインストーラパッケージには、Tridentオペレータの導入とAstra Tridentのインストールに必要なものがすべて含まれています。から最新バージョンのTridentインストーラ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/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

手順**2**：を作成します `TridentOrchestrator` **CRD**

を作成します `TridentOrchestrator` カスタムリソース定義（CRD）。を作成します `TridentOrchestrator` カスタムリソース。で適切なCRD YAMLバージョンを使用します `deploy/crds` を作成します `TridentOrchestrator` CRD：

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

手順**3**：オペレータのレジストリの場所を更新します

インチ `/deploy/operator.yaml`、を更新します `image: docker.io/netapp/trident-`

`operator:23.04.0` イメージレジストリの場所を反映します。。 "TridentとCSIの画像" 1つのレジストリまたは別のレジストリに配置できますが、すべてのCSIイメージは同じレジストリに配置する必要があります。例：

- `image: <your-registry>/trident-operator:23.04.0` すべての画像が1つのレジストリにある場合。
- `image: <your-registry>/netapp/trident-operator:23.04.0` TridentイメージがCSIイメージとは別のレジストリにある場合。

ステップ**4**：**Trident**オペレータを導入

Astra Tridentインストーラには、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バンドルファイルが用意されています。このバンドルファイルを使用すると、オペレータを簡単に導入し、デフォルトの設定でAstra Tridentをインストールできます。

- 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

- デフォルトでは、Tridentのインストーラによって `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。次を使用して作成してください：

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

- オペレータを以外のネームスペースに配置する場合 `trident` 名前空間、更新 `serviceaccount.yaml`、 `clusterrolebinding.yaml` および `operator.yaml` を使用してバンドルファイルを生成します `kustomization.yaml`。

  a. を作成します `kustomization.yaml` 次のコマンドを使用して、*<bundle>* is `bundle_pre_1_25` または `bundle_post_1_25` 使用しているKubernetesのバージョンに基づきます。

  ```
  cp kustomization_<bundle>.yaml kustomization.yaml
  ```

  b. 次のコマンドを使用してバンドルをコンパイルします。WHERE_STORE_IS <bundle> `bundle_pre_1_25` または `bundle_post_1_25` 使用しているKubernetesのバージョンに基づきます。

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

手順

1. リソースを作成し、オペレータを配置します。

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

2. operator、deployment、およびReplicaSetsが作成されたことを確認します。

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

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**5:**でイメージレジストリの場所を更新します `TridentOrchestrator`

。 "TridentとCSIの画像" 1つのレジストリまたは別のレジストリに配置できますが、すべてのCSIイメージは同じレジストリに配置する必要があります。更新 `deploy/crds/tridentorchestrator_cr.yaml` レジストリ設定に基づいて追加の場所の仕様を追加します。

1つのレジストリ内のイメージ

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:23.04"
tridentImage: "<your-registry>/trident:23.04.0"
```

異なるレジストリ内の画像

を追加する必要があります `sig-storage` に移動します `imageRegistry` 別のレジストリの場所を使用します。

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.04"
tridentImage: "<your-registry>/netapp/trident:23.04.0"
```

手順**6**： `TridentOrchestrator` **Trident**をインストール

これで、を作成できます `TridentOrchestrator` Astra Tridentを導入必要に応じて、さらに行うことができます "Tridentのインストールをカスタマイズ" で属性を使用する `TridentOrchestrator` 仕様次の例は、TridentイメージとCSIイメージが異なるレジストリにあるインストールを示しています。

```
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:
  Autosupport Image:  <your-registry>/netapp/trident-autosupport:23.04
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.04.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>/sig-storage
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/netapp/trident:23.04.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

インストールを確認するには、いくつかの方法があります。

を使用します `TridentOrchestrator` ステータス

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。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。 "ログをチェックしてください"。

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。

ポッドの作成ステータスを使用する

作成したポッドのステータスを確認することで、Astra Tridentのインストールが完了したかどうかを確認できます。

```
kubectl get pods -n trident

NAME                                   READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw    6/6     Running   0
1m
trident-node-linux-mr6zc               2/2     Running   0
1m
trident-node-linux-xrp7w               2/2     Running   0
1m
trident-node-linux-zh2jt               2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv      1/1     Running   0
3m
```

を使用します `tridentctl`

を使用できます `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 |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

次のステップ

できるようになりました。 "バックエンドとストレージクラスを作成し、ボリュームをプロビジョニングして、ポッドにボリュームをマウントします"。

Helm（標準モード）を使用してTridentを導入

Tridentオペレータを導入し、Helmを使用してAstra Tridentをインストールできます。このプロセスでは、環境 をインストールする際に、Astra Tridentで必要なコンテナイメージがプライベートレジストリに格納されません。プライベートイメージレジストリがある場合は、を使用します "オフライン導入のプロセス"。

Astra Tridentに関する重要な情報23.04

- Astra Tridentに関する次の重要な情報をお読みください。*

** ：Trident ** に関する重要な情報

- TridentでKubernetes 1.27がサポートされるようになりました。Kubernetesをアップグレードする前にTridentをアップグレード
- Astra Tridentは、SAN環境でマルチパス構成を厳密に使用し、推奨される値をに設定します `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オペレータを導入し、**Helm**を使用して**Astra Trident**をインストール

Tridentの使用 "Helmチャート" Tridentオペレータを導入し、Tridentを一度にインストールできます。

レビュー "インストールの概要" インストールの前提条件を満たし、環境に適したインストールオプションを選択していることを確認します。

作業を開始する前に

に加えて "導入の前提条件" 必要です "Helm バージョン 3"。

手順

1. Astra Trident Helmリポジトリを追加します。

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

2. 使用 `helm install` をクリックし、次の例に示すように、導入環境の名前を指定します `23.04.0` は、インストールするAstra Tridentのバージョンです。

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--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。

を使用できます `helm list` 名前、ネームスペース、グラフ、ステータス、アプリケーションバージョンなどのインストールの詳細を確認するには、次の手順を実行します。とリビジョン番号。

インストール中に設定データを渡す

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。

オプション	説明
`--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` コマンドを入力します `23.04.0` は、インストールするAstra Tridentのバージョンです。

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--create-namespace --namespace --set tridentDebug=true
```

設定オプション

このテーブルと `values.yaml` Helmチャートの一部であるファイルには、キーとそのデフォルト値のリストが表示されます。

オプション	説明	デフォルト
`nodeSelector`	ポッド割り当てのノードラベル	
`podAnnotations`	ポッドの注釈	
`deploymentAnnotations`	配置のアノテーション	
`tolerations`	ポッド割り当ての許容値	
`affinity`	ポッド割り当てのアフィニティ	
`tridentControllerPluginNodeSelector`	ポッド用の追加のノードセレクタ。を参照してください [コントローラポッドとノードポッドについて] を参照してください。	
`tridentControllerPluginTolerations`	ポッドに対するKubernetesの許容範囲を上書きします。を参照してください [コントローラポッドとノードポッドについて] を参照してください。	
`tridentNodePluginNodeSelector`	ポッド用の追加のノードセレクタ。を参照してください [コントローラポッドとノードポッドについて] を参照してください。	
`tridentNodePluginTolerations`	ポッドに対するKubernetesの許容範囲を上書きします。を参照してください [コントローラポッドとノードポッドについて] を参照してください。	

オプション	説明	デフォルト
imageRegistry	のレジストリを指定します trident-operator、trident、およびその他の画像。デフォルトをそのまま使用する場合は、空のままにします。	""
imagePullPolicy	のイメージプルポリシーを設定します trident-operator。	IfNotPresent
imagePullSecrets	のイメージプルシークレットを設定します trident-operator、trident、およびその他の画像。	
kubeletDir	kubeletの内部状態のホスト位置を上書きできます。	'"/var/lib/kubeelet"'
operatorLogLevel	Tridentオペレータのログレベルを次のように設定できます。trace、 debug、 info、 warn、error`または `fatal。	"info"
operatorDebug	Tridentオペレータのログレベルをdebugに設定できます。	true
operatorImage	のイメージを完全に上書きできます trident-operator。	""
operatorImageTag	のタグを上書きできます trident-operator イメージ（Image）：	""
tridentIPv6	IPv6クラスタでAstra Tridentを動作させることができます。	false
tridentK8sTimeout	ほとんどのKubernetes API処理でデフォルトの30秒タイムアウトを上書きします（0以外の場合は秒単位）。	0
tridentHttpRequestTimeout	HTTP要求のデフォルトの90秒タイムアウトをで上書きします 0s タイムアウトの期間は無限です。負の値は使用できません。	"90s"
tridentSilenceAutosupport	Astra Tridentの定期的なAutoSupport レポートを無効にできます。	false
tridentAutosupportImageTag	Astra Trident AutoSupport コンテナのイメージのタグを上書きできます。	<version>
tridentAutosupportProxy	Astra TridentのAutoSupport コンテナがHTTPプロキシ経由で自宅に通信できるようになります。	""
tridentLogFormat	Astra Tridentのログ形式を設定します (text または json）。	"text"

オプション	説明	デフォルト
`tridentDisableAuditLog`	Astra Trident監査ロガーを無効にします。	`true`
`tridentLogLevel`	Astra Tridentのログレベルを次のように設定できます。 `trace`、 `debug`、 `info`、 `warn`、 `error`` または `fatal`。	`"info"`
`tridentDebug`	Astra Tridentのログレベルをに設定できます `debug`。	`false`
`tridentLogWorkflows`	特定のAstra Tridentワークフローを有効にして、トレースロギングやログ抑制を実行できます。	""
`tridentLogLayers`	特定のAstra Tridentレイヤでトレースロギングやログ抑制を有効にできます。	""
`tridentImage`	Astra Tridentのイメージを完全に上書きできます。	""
`tridentImageTag`	Astra Tridentのイメージのタグを上書きできます。	""
`tridentProbePort`	Kubernetesの活性/準備プローブに使用されるデフォルトポートを上書きできます。	""
`windows`	WindowsワーカーノードにAstra Tridentをインストールできます。	`false`
`enableForceDetach`	強制切り離し機能を有効にできます。	`false`
`excludePodSecurityPolicy`	オペレータポッドのセキュリティポリシーを作成から除外します。	`false`

コントローラポッドとノードポッドについて

Astra Tridentは、単一のコントローラポッドと、クラスタ内の各ワーカーノード上のノードポッドとして実行されます。Astra Tridentボリュームをマウントするすべてのホストでノードポッドが実行されている必要があります。

Kubernetes "ノードセレクタ" および "寛容さと汚れ" は、特定のノードまたは優先ノードで実行されるようにポッドを制限するために使用されます。「ControllerPlugin」およびを使用します `NodePlugin`を使用すると、拘束とオーバーライドを指定できます。

- コントローラプラグインは、Snapshotやサイズ変更などのボリュームのプロビジョニングと管理を処理します。
- ノードプラグインによって、ノードへのストレージの接続が処理されます。

次のステップ

できるようになりました。 "バックエンドとストレージクラスを作成し、ボリュームをプロビジョニングし

て、ポッドにボリュームをマウントします"。

Helm（オフラインモード）を使用したTridentのオペレータの導入

Tridentオペレータを導入し、Helmを使用してAstra Tridentをインストールできます。このプロセスでは、環境 をインストールする際に、Astra Tridentで必要なコンテナイメージがプライベートレジストリに格納されます。プライベートイメージレジストリがない場合は、を使用します "標準的な導入のプロセス"。

Astra Tridentに関する重要な情報23.04

- Astra Tridentに関する次の重要な情報をお読みください。*

** ：Trident ** に関する重要な情報

- TridentでKubernetes 1.27がサポートされるようになりました。Kubernetesをアップグレードする前にTridentをアップグレード
- Astra Tridentは、SAN環境でマルチパス構成を厳密に使用し、推奨される値をに設定します `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オペレータを導入し、Helmを使用してAstra Tridentをインストール

Tridentの使用 "Helmチャート" Tridentオペレータを導入し、Tridentを一度にインストールできます。

レビュー "インストールの概要" インストールの前提条件を満たし、環境に適したインストールオプションを選択していることを確認します。

作業を開始する前に

に加えて "導入の前提条件" 必要です "Helm バージョン 3"。

手順

1. Astra Trident Helmリポジトリを追加します。

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

2. 使用 `helm install` 展開およびイメージレジストリの場所の名前を指定します。。 "TridentとCSIの画像" 1つのレジストリまたは別のレジストリに配置できますが、すべてのCSIイメージは同じレジストリに配置する必要があります。例では、 `23.04.0` は、インストールするAstra Tridentのバージョンです。

1つのレジストリ内のイメージ

```
helm install <name> netapp-trident/trident-operator --version
23.04.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace>
```

異なるレジストリ内の画像

を追加する必要があります `sig-storage` に移動します `imageRegistry` 別のレジストリの場所を使用します。

```
helm install <name> netapp-trident/trident-operator --version
23.04.0 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.04.0 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.04 --set tridentImage=<your-
registry>/netapp/trident:23.04.0 --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。

を使用できます `helm list` 名前、ネームスペース、グラフ、ステータス、アプリケーションバージョンなどのインストールの詳細を確認するには、次の手順を実行します。とリビジョン番号。

インストール中に設定データを渡す

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。

オプション	説明
`--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` コマンドを入力します `23.04.0` は、インストールするAstra Tridentのバージョンです。

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--create-namespace --namespace --set tridentDebug=true
```

このテーブルと `values.yaml` Helmチャートの一部であるファイルには、キーとそのデフォルト値のリストが表示されます。

オプション	説明	デフォルト
`nodeSelector`	ポッド割り当てのノードラベル	
`podAnnotations`	ポッドの注釈	
`deploymentAnnotations`	配置のアノテーション	
`tolerations`	ポッド割り当ての許容値	
`affinity`	ポッド割り当てのアフィニティ	
`tridentControllerPluginNodeSelector`	ポッド用の追加のノードセレクタ。を参照してください [コントローラポッドとノードポッドについて] を参照してください。	
`tridentControllerPluginTolerations`	ポッドに対するKubernetesの許容範囲を上書きします。を参照してください [コントローラポッドとノードポッドについて] を参照してください。	
`tridentNodePluginNodeSelector`	ポッド用の追加のノードセレクタ。を参照してください [コントローラポッドとノードポッドについて] を参照してください。	
`tridentNodePluginTolerations`	ポッドに対するKubernetesの許容範囲を上書きします。を参照してください [コントローラポッドとノードポッドについて] を参照してください。	
`imageRegistry`	のレジストリを指定します `trident-operator`、`trident`、およびその他の画像。デフォルトをそのまま使用する場合は、空のままにします。	""
`imagePullPolicy`	のイメージプルポリシーを設定します `trident-operator`。	`IfNotPresent`
`imagePullSecrets`	のイメージプルシークレットを設定します `trident-operator`、`trident`、およびその他の画像。	
`kubeletDir`	kubeletの内部状態のホスト位置を上書きできます。	'"/var/lib/kubeelet"'
`operatorLogLevel`	Tridentオペレータのログレベルを次のように設定できます。`trace`、`debug`、`info`、`warn`、`error`または `fatal`。	`"info"`

オプション	説明	デフォルト
operatorDebug	Tridentオペレータのログレベルをdebugに設定できます。	true
operatorImage	のイメージを完全に上書きできます trident-operator。	""
operatorImageTag	のタグを上書きできます trident-operator イメージ（Image）：	""
tridentIPv6	IPv6クラスタでAstra Tridentを動作させることができます。	false
tridentK8sTimeout	ほとんどのKubernetes API処理でデフォルトの30秒タイムアウトを上書きします（0以外の場合は秒単位）。	0
tridentHttpRequestTimeout	HTTP要求のデフォルトの90秒タイムアウトをで上書きします 0s タイムアウトの期間は無限です。負の値は使用できません。	"90s"
tridentSilenceAutosupport	Astra Tridentの定期的なAutoSupport レポートを無効にできます。	false
tridentAutosupportImageTag	Astra Trident AutoSupport コンテナのイメージのタグを上書きできます。	<version>
tridentAutosupportProxy	Astra TridentのAutoSupport コンテナがHTTPプロキシ経由で自宅に通信できるようになります。	""
tridentLogFormat	Astra Tridentのログ形式を設定します (text または json）。	"text"
tridentDisableAuditLog	Astra Trident監査ロガーを無効にします。	true
tridentLogLevel	Astra Tridentのログレベルを次のように設定できます。 trace、 debug、 info、 warn、 error` または `fatal。	"info"
tridentDebug	Astra Tridentのログレベルをに設定できます debug。	false
tridentLogWorkflows	特定のAstra Tridentワークフローを有効にして、トレースロギングやログ抑制を実行できます。	""
tridentLogLayers	特定のAstra Tridentレイヤでトレースロギングやログ抑制を有効にできます。	""

オプション	説明	デフォルト
`tridentImage`	Astra Tridentのイメージを完全に上書きできます。	""
`tridentImageTag`	Astra Tridentのイメージのタグを上書きできます。	""
`tridentProbePort`	Kubernetesの活性/準備プローブに使用されるデフォルトポートを上書きできます。	""
`windows`	WindowsワーカーノードにAstra Tridentをインストールできます。	`false`
`enableForceDetach`	強制切り離し機能を有効にできます。	`false`
`excludePodSecurityPolicy`	オペレータポッドのセキュリティポリシーを作成から除外します。	`false`

コントローラポッドとノードポッドについて

Astra Tridentは、単一のコントローラポッドと、クラスタ内の各ワーカーノード上のノードポッドとして実行されます。Astra Tridentボリュームをマウントするすべてのホストでノードポッドが実行されている必要があります。

Kubernetes "ノードセレクタ" および "寛容さと汚れ" は、特定のノードまたは優先ノードで実行されるようにポッドを制限するために使用されます。「ControllerPlugin」およびを使用します `NodePlugin`を使用すると、拘束とオーバーライドを指定できます。

- コントローラプラグインは、Snapshotやサイズ変更などのボリュームのプロビジョニングと管理を処理します。
- ノードプラグインによって、ノードへのストレージの接続が処理されます。

次のステップ

できるようになりました。 "バックエンドとストレージクラスを作成し、ボリュームをプロビジョニングして、ポッドにボリュームをマウントします"。

Tridentオペレータのインストールをカスタマイズ

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` 引数allow、使用を検討してください `tridentctl` 必要に応じて変更するカスタムYAMLマニフェストを生成します。

コントローラポッドとノードポッドについて

Astra Tridentは、単一のコントローラポッドと、クラスタ内の各ワーカーノード上のノードポッドとして実行されます。Astra Tridentボリュームをマウントするすべてのホストでノードポッドが実行されている必要があります。

Kubernetes "ノードセレクタ" および "寛容さと汚れ" は、特定のノードまたは優先ノードで実行されるように

ポッドを制限するために使用されます。「ControllerPlugin」およびを使用します `NodePlugin`を使用すると、拘束とオーバーライドを指定できます。

- コントローラプラグインは、Snapshotやサイズ変更などのボリュームのプロビジョニングと管理を処理します。
- ノードプラグインによって、ノードへのストレージの接続が処理されます。

設定オプション

`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指定します。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。

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
`enableForceDetach`	`ontap-san` および `ontap-san-economy` のみ。 KubernetesのNon-Graceful Node Shutdown（NGN）と連携して、ノードに障害が発生した場合に、マウントされたボリュームを含むワークロードを新しいノードに安全に移行する機能をクラスタ管理者に提供します。 *これは23.04の実験的な機能です。*を参照してください [フォースデタッチの詳細] を参照してください。	`false`
`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
`useIPv6`	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
`autosupportImage`	AutoSupport テレメトリのコンテナイメージ	"netapp/trident-autosupport：23.07"

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：23.07"
`imageRegistry`	形式の内部レジストリへのパス `<registry FQDN>[:port][/subpath]`	「k8s.gcr.io/sig-storage」（k8s 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/kubeelet"
`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	
`imagePullPolicy`	Tridentオペレータのイメージプルポリシーを設定します。有効な値は次のとおりです。 `Always` 常にイメージをプルする。 `IfNotPresent` ノード上にイメージが存在しない場合にのみ取得します。 `Never` 画像を絶対に引き出さないでください。	`IfNotPresent`
`controllerPluginNodeSelector`	ポッド用の追加のノードセレクタ。 の形式はと同じです `pod.spec.nodeSelector`。	デフォルトはありません。オプションです
`controllerPluginTolerations`	ポッドに対するKubernetesの許容範囲を上書きします。はと同じ形式です `pod.spec.Tolerations`。	デフォルトはありません。オプションです
`nodePluginNodeSelector`	ポッド用の追加のノードセレクタ。の形式はと同じです `pod.spec.nodeSelector`。	デフォルトはありません。オプションです

パラメータ	説明	デフォルト
nodePluginTolerations	ポッドに対するKubernetesの許容範囲を上書きします。はと同じ形式です pod.spec.Tolerations。	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"。

フォースデタッチの詳細

では、[強制切り離し]を使用できます ontap-san および ontap-san-economy のみ。強制接続解除を有効にする前に、Kubernetesクラスタで非グレースフルノードシャットダウン（NGN）を有効にする必要があります。詳細については、を参照してください "Kubernetes：正常なノードシャットダウンではありません"。

Astra TridentはKubernetes NGNに依存しているため、削除しないでください out-of-service 許容できないすべてのワークロードが再スケジュールされるまで、正常でないノードから影響を受けます。汚染を無謀に適用または削除すると、バックエンドのデータ保護が危険にさらされる可能性があります。

Kubernetesクラスタ管理者がを適用したとき node.kubernetes.io/out-of-service=nodeshutdown:NoExecute ノードおよびに影響を与えます enableForceDetach がに設定されます `true`Astra Tridentがノードのステータスを判断し、次の処理を実行します。

1. そのノードにマウントされたボリュームのバックエンドI/Oアクセスを停止します。
2. Astra Tridentノードオブジェクトをにマークします dirty (新しい出版物には安全ではありません)。

Tridentコントローラは、（とマークされたあとに）ノードが再認定されるまで、新しいパブリッシュボリューム要求を拒否します dirty`をクリックします。マウントされたPVCでスケジュールされているワークロード（クラスタノードが正常で準備が完了したあとも含む）は、Astra Tridentがノードを検証できるまで受け入れられません `clean (新しい出版物のための安全)。

ノードの健常性が回復してtaintが削除されると、Astra Tridentは次の処理を実行します。

1. ノード上の古い公開パスを特定してクリーンアップします。
2. ノードがに含まれている場合 cleanable 状態（out-of-service taintが削除され、ノードがinになっています Ready 状態）。古い公開済みパスはすべてクリーンで、Astra Tridentはノードをとして再登録します clean 新しいボリュームのノードへの公開を許可します。

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を使用してインストールします

tridentctlを使用してインストールします

を使用して、Astra Tridentをインストールできます `tridentctl`。このプロセスでは、Astra Tridentで必要なコンテナイメージがプライベートレジストリに格納されているかどうかに関係なく、環境 のインストールを実行します。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23.04

- Astra Tridentに関する次の重要な情報をお読みください。*

 ：Trident に関する重要な情報

- TridentでKubernetes 1.27がサポートされるようになりました。Kubernetesをアップグレードする前にTridentをアップグレード
- Astra Tridentは、SAN環境でマルチパス構成を厳密に使用し、推奨される値をに設定します `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

を使用してAstra Tridentをインストールします `tridentctl`

レビュー "インストールの概要" インストールの前提条件を満たし、環境に適したインストールオプションを選択していることを確認します。

作業を開始する前に

インストールを開始する前に、Linuxホストにログインして、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" 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`。

1. Kubernetesのバージョンを確認します。

```
kubectl version
```

2. クラスタ管理者の権限を確認します。

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

3. 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ことを確認します。

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

手順**1**：**Trident**のインストーラパッケージをダウンロード

Astra Tridentインストーラパッケージは、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オブジェクトを設定し、CSIサイドカーを初期化して、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します。から最新バージョンのTridentインストーラ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"。例では、選択した<trident-installer-XX.XX.X.tar.gz> Tridentバージョンを使用してupdate_Tridentを更新します。

```
wget https://github.com/NetApp/trident/releases/download/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

手順**2**：**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追加の引数を追加して、イメージのレジストリの場所を指定できます。

標準モード

```
./tridentctl install -n trident
```

1つのレジストリ内のイメージ

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:23.04 --trident
-image <your-registry>/trident:23.04.0
```

異なるレジストリ内の画像

を追加する必要があります `sig-storage` に移動します `imageRegistry` 別のレジストリの場所を使用します。

```
./tridentctl install -n trident --image-registry <your-registry>/sig-
storage --autosupport-image <your-registry>/netapp/trident-
autosupport:23.04 --trident-image <your-
registry>/netapp/trident:23.04.0
```

インストールステータスは次のようになります。

```
....
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-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=23.04.0
INFO Trident installation succeeded.
....
```

インストールを確認します。

ポッドの作成ステータスまたはを使用して、インストールを確認できます `tridentctl`。

ポッドの作成ステータスを使用する

作成したポッドのステータスを確認することで、Astra Tridentのインストールが完了したかどうかを確認できます。

```
kubectl get pods -n trident

NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-679648bd45-cv2mx   6/6     Running   0          5m29s
trident-node-linux-vgc8n              2/2     Running   0          5m29s
```

インストーラが正常に完了しない場合、または `trident-controller-<generated id>` (`trident-csi-<generated id>` 23.01より前のバージョンでは、* RUNNING *ステータスがありません。プラットフォームはインストールされませんでした。使用 `-d` 終了： "デバッグモードをオンにします" および問題 のトラブルシューティングを行います。

を使用します `tridentctl`

を使用できます `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 |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

構成例

例**1**：**Windows**ノードで**Astra Trident**の実行を有効にします

WindowsノードでAstra Tridentを実行できるようにするには、次の手順を実行します。

```
tridentctl install --windows -n trident
```

例**2**：強制切り離しを有効にします

強制切り離しの詳細については、を参照してください "Tridentオペレータのインストールをカスタマイズ"。

```
tridentctl install --enable-force-detach=true -n trident
```

次のステップ

できるようになりました。 "バックエンドとストレージクラスを作成し、ボリュームをプロビジョニングして、ポッドにボリュームをマウントします"。

tridentctlのインストールをカスタマイズします

Astra Tridentインストーラを使用して、インストール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のインストールが完了したら、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、ポッドへ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。"

著作権に関する情報

商標に関する情報