



Astra Trident をインストール

Astra Trident

NetApp
November 14, 2025

目次

Astra Trident をインストール	1
Astra Tridentのインストール方法をご確認ください	1
Astra Trident 23.01に関する重要な情報	1
作業を開始する前に	1
インストール方法を選択します	1
インストールモードを選択します	3
メソッドとモードに基づいてプロセスを選択します	4
インストール方法を切り替える	5
その他の既知の設定オプション	5
Tridentオペレータを使用してインストール	5
Tridentオペレータを手動で導入（標準モード）	5
Tridentオペレータを手動で導入（オフラインモード）	11
Helm（標準モード）を使用してTridentを導入	17
Helm（オフラインモード）を使用したTridentのオペレータの導入	22
Tridentオペレータのインストールをカスタマイズ	26
tridentctlを使用してインストールします	30
tridentctlを使用してインストールします	30
tridentctlのインストールをカスタマイズします	33

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 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 1.26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 `kubect1`（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 ["×](#)

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

組織に必要な_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"
<code>tridentctl</code>	"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せん `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 `/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 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 1.26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 クラスター](#)" 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/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.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。

- 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。

Tridentのインストーラがオペレータをに導入します 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。を使用してください kubectl apply -f deploy/namespace.yaml をクリックして作成します。

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
```



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：

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

2. オペレータが配備されたことを確認します

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

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



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：を作成します 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:23.01
    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:23.01.1
  Message:            Trident installed Namespace:
trident
  Status:              Installed
  Version:              v23.01.1
Events:
  Type Reason Age From Message ---- -
  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。

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-v4cpw1m	6/6	Running	0
trident-node-linux-mr6zc1m	2/2	Running	0
trident-node-linux-xrp7w1m	2/2	Running	0
trident-node-linux-zh2jt1m	2/2	Running	0
trident-operator-766f7b8658-ldzsv3m	1/1	Running	0

を使用します `tridentctl`

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 |
+-----+-----+
| 23.01.1        | 23.01.1        |
+-----+-----+
```

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 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 1.26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/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.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:23.01.1` 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:23.01.1` 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:23.01.1` 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

Tridentのインストーラがオペレータをに導入します `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。を使用してください `kubectl apply -f deploy/namespace.yaml` をクリックして作成します。

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` オペレータを配備する前に、

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
```



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。

- 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`。

2. オペレータが配備されたことを確認します

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

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



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:23.01"
tridentImage: "<your-registry>/trident:23.01.1"
```

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:23.01"
tridentImage: "<your-registry>/netapp/trident:23.01.1"
```

手順6:を作成します 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:23.01
  Debug:             true
  Image Registry:    <your-registry>/sig-storage
  Namespace:         trident
  Trident Image:     <your-registry>/netapp/trident:23.01.1
Status:
  Current Installation Params:
    IPv6:             false
    Autosupport Hostname:
    Autosupport Image: <your-registry>/netapp/trident-
autosupport:23.01
    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:23.01.1
  Message:            Trident installed
  Namespace:          trident
  Status:              Installed
  Version:             v23.01.1
Events:
  Type Reason Age From Message ----
  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 <code>TridentOrchestrator</code> 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 <code>spec.uninstall=true</code> 。
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	。 <code>TridentOrchestrator</code> は使用されません。別のファイルがすでに存在します。

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
trident-controller-7d466bf5c7-v4cpw 1m	6/6	Running	0
trident-node-linux-mr6zc 1m	2/2	Running	0
trident-node-linux-xrp7w 1m	2/2	Running	0
trident-node-linux-zh2jt 1m	2/2	Running	0
trident-operator-766f7b8658-ldzsv 3m	1/1	Running	0

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 |
+-----+-----+
| 23.01.1       | 23.01.1       |
+-----+-----+
```

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に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 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 1.26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](#)" が必要です "[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 23.01.1 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 23.01.1  
--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 23.01.1 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 23.01.1
--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
<code>nodeSelector</code>	ポッド割り当てのノードラベル	
<code>podAnnotations</code>	ポッドの注釈	
<code>deploymentAnnotations</code>	配置のアノテーション	
<code>tolerations</code>	ポッド割り当ての許容値	
<code>affinity</code>	ポッド割り当てのアフィニティ	
<code>tridentControllerPluginNodeSelector</code>	ポッド用の追加のノードセクタ。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。	
<code>tridentControllerPluginTolerations</code>	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。	
<code>tridentNodePluginNodeSelector</code>	ポッド用の追加のノードセクタ。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。	
<code>tridentNodePluginTolerations</code>	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/kubelet"
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クラスターで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 **"ノードセレクトラ"** 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に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 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 1.26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、23.01.1 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
23.01.1 --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
23.01.1 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.01.1 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.01 --set tridentImage=<your-
registry>/netapp/trident:23.01.1 --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
<code>--values</code> (または <code>-f</code>)	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。
<code>--set</code>	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 23.01.1 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 23.01.1
--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
<code>nodeSelector</code>	ポッド割り当てのノードラベル	
<code>podAnnotations</code>	ポッドの注釈	
<code>deploymentAnnotations</code>	配置のアノテーション	
<code>tolerations</code>	ポッド割り当ての許容値	
<code>affinity</code>	ポッド割り当てのアフィニティ	
<code>tridentControllerPluginNodeSelector</code>	ポッド用の追加のノードセクタ。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。	
<code>tridentControllerPluginTolerations</code>	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。	
<code>tridentNodePluginNodeSelector</code>	ポッド用の追加のノードセクタ。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。	
<code>tridentNodePluginTolerations</code>	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。	
<code>imageRegistry</code>	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。	""
<code>imagePullPolicy</code>	のイメージプルポリシーを設定します trident-operator。	IfNotPresent
<code>imagePullSecrets</code>	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。	
<code>kubeletDir</code>	kubeletの内部状態のホスト位置を上書きできます。	"/var/lib/kubelet"
<code>operatorLogLevel</code>	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せん 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 "ノードセレクトラ" 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名前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、名前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
<code>namespace</code>	<code>Astra Trident</code> をインストールする名前スペース	デフォルト
<code>debug</code>	<code>Astra Trident</code> のデバッグを有効にします	いいえ
<code>windows</code>	をに設定します <code>true</code> <code>Windows</code> 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
<code>IPv6</code>	<code>IPv6</code> 経由の <code>Astra Trident</code> をインストール	いいえ
<code>k8sTimeout</code>	Kubernetes 処理のタイムアウト	30 秒
<code>silenceAutosupport</code>	<code>AutoSupport</code> バンドルをネットアップに自動的に送信しない	いいえ
<code>enableNodePrep</code>	ワーカーノードの依存関係を自動的に管理 (* beta *)	いいえ
<code>autosupportImage</code>	<code>AutoSupport</code> テレメトリのコンテナイメージ	"netapp/trident-autosupport : 23.01"
<code>autosupportProxy</code>	<code>AutoSupport</code> 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"
<code>uninstall</code>	<code>Astra Trident</code> のアンインストールに使用するフラグ	いいえ
<code>logFormat</code>	<code>Astra Trident</code> のログ形式が使用 [text、JSON]	テキスト (Text)
<code>tridentImage</code>	インストールする <code>Astra Trident</code> イメージ	「 NetApp / Trident : 21.04 」

パラメータ	説明	デフォルト
imageRegistry	形式の内部レジストリへのパス <registry FQDN>[:port] [/subpath]	"k83.gcr.io/sig-storage (k8s 1.19+) またはQua.io/k8scsi"
kubeletDir	ホスト上の kubelet ディレクトリへのパス	「 /var/lib/kubelet 」
wipeout	Astra Trident を完全に削除するために削除するリソースのリスト	
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	ポッドに対するKubernetesの許容範囲を上書きします。POD.spec.Tolerations 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	ポッド用の追加のノードセクタ。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	ポッドに対するKubernetesの許容範囲を上書きします。POD.spec.Tolerations 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：ノードセクタを使用して導入します

次の例では、ノードセクタを使用して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 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 1.26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 クラスター](#)" 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. クラスター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ー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/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.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。



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。

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:23.01 --trident  
-image <your-registry>/trident:23.01.1
```

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:23.01 --trident-image <your-  
registry>/netapp/trident:23.01.1
```

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=23.01.1  
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 |
+-----+-----+
| 23.01.1       | 23.01.1       |
+-----+-----+
```

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
```

著作権に関する情報

Copyright © 2025 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权を有し、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权については、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。