Trident をインストール

Trident

NetApp
January 17, 2025

目次

Trident をインストール

Tridentのインストールについて

Tridentをさまざまな環境や組織にインストールできるように、NetAppには複数のインストールオプションが用意されています。Tridentは、Tridentオペレータ（手動またはHelmを使用）またはを使用して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。

Trident 24.06に関する重要な情報

- Tridentに関する次の重要な情報をお読みください。*

Trident に関するの重要な情報

- TridentでKubernetes 1.31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
- Tridentでは、SAN環境でのマルチパス構成の使用が厳密に適用されます。multipath.confファイルの推奨値はです `find_multipaths: no`。

  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

手動で導入する場合でも、Helmを使用する場合でも、Tridentオペレータはインストールを簡素化し、Tridentリソースを動的に管理するための優れた方法です。カスタムリソース（CR）の属性を使用する`TridentOrchestrator`こともできます"Tridentのオペレータ環境をカスタマイズ"。

Tridentオペレータには次のようなメリットがあります。

** Trident**オブジェクトの作成****

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

****リソースアカウンタビリティ****

クラスタを対象としたTridentオペレータは、Tridentインストールに関連付けられたリソースをクラスタレベルで管理します。これにより、ネームスペースを対象とした演算子を使用してクラスタを対象としたリソースを管理する際に発生する可能性のあるエラーを軽減できます。これは、自己修復とパッチ適用に不可欠です。

**** 自己回復機能****

オペレータはTridentのインストールを監視し、展開が削除された場合や誤って変更された場合などの問題に積極的に対処します。 `trident-operator-<generated-id>`CRをTridentインストールに関連付けるポッドが作成され`TridentOrchestrator`ます。これにより、クラスタ内に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して、インストールが強力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、デバッグログを生成するためにTridentを有効に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、を `TridentOrchestrator` `true`次のように設定し `spec.debug`ます。

```
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、新しいポッドが作成され、それに応じてインストールが変更される可能性があります。

****クリーン再インストール****

クラスタを対象としたTridentオペレータを使用すると、クラスタを対象としたリソースを完全に削除できます。ユーザーはTridentを完全にアンインストールして簡単に再インストールできます。

** 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、オペレータは既存の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、オペレータがTridentをインストールできません。Tridentがオペレータとともにすでにインストールされている場合は、サポートされていないKubernetesバージョンに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。

を使用します `tridentctl`

アップグレードが必要な既存の導入環境がある場合、または導入環境を高度にカスタマイズする場合は、を検討する必要があります。これは、従来のTridentの導入方法です。

Tridentリソースのマニフェストを生成できます。これには、導入、デーモンセット、サービスアカウント、Tridentのインストール時に作成されるクラスタロールが含まれます。

> 22.04リリース以降では、Tridentをインストールするたびにに`AES`キーが再生成されなくなりました。このリリースでは、Tridentは新しいシークレットオブジェクトをインストールします。このオブジェクトは複数のインストールにまたがって保持されます。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、以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、ネットワーク制限が適用され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、デフォルトのレジストリを使用して(`docker.io`、必要なTrident（およびCSI(`registry.k8s.io`）イメージを格納します。

標準モードを使用する場合、Tridentインストーラは次の処理を実行します。

- インターネット経由でコンテナイメージを取得します
- デプロイメントまたはノードデーモンセットを作成します。これにより、Kubernetesクラスタ内の対象となるすべてのノードでTridentポッドがスピンアップされます。

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。

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

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"

メソッド	インストールモード
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。

- 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てください。を使用してを展開した場合は `tridentctl`、適切なバージョンのバイナリを使用してTridentをアンインストールする必要があります `tridentctl`。同様に、オペレータを使用して展開する場合は、CRを編集し、Tridentをアンインストールするように設定する `spec.uninstall=true`必要があります `TridentOrchestrator`。
- オペレータベースの導入環境を削除してTridentの導入に使用する場合 `tridentctl`は、まず`Trident`を編集してからアンインストールするように設定する `spec.uninstall=true`必要があります `TridentOrchestrator`。次に、とオペレータの配置を削除し `TridentOrchestrator`ます。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ポートフォリオ製品にTridentをインストールする場合：

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

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

Tridentオペレータ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オペレータを手動で導入して、Tridentをインストールできます。このプロセスは、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保存されていないインストールに適用されます。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"オフライン導入のプロセス"ます。

Trident 24.10に関する重要な情報

- 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**の重要な情報

- TridentでKubernetes 1.31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
- Tridentでは、SAN環境でのマルチパス構成の使用が厳密に適用されます。multipath.confファイルの推奨値はです `find_multipaths: no`。

  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、以降のすべての例で「 kubectl 」ではなく「 OC 」を使用し、「 OC login-u SYSTEM ： admin 」または「 OC login-u kube-admin 」を実行して最初に「 *system:admin 」としてログインします。

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

Tridentインストーラパッケージには、Tridentオペレータの導入とTridentのインストールに必要なすべてのものが含まれています。からTridentインストーラの最新バージョンをダウンロードして展開し"GitHubの_Assets_sectionを参照してください"ます。

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

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

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

```
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

Tridentインストーラには、オペレータのインストールと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。バンドルファイルは、デフォルト設定を使用してオペレータを導入し、Tridentをインストールするための簡単な方法です。

- クラスタでKubernetes 1.24を実行している場合は、を使用し `bundle_pre_1_25.yaml`ます。

- 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.yaml>* is `bundle_pre_1_25.yaml` または `bundle_post_1_25.yaml` 使用しているKubernetesのバージョンに基づきます。

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/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.yaml> `bundle_pre_1_25.yaml` または `bundle_post_1_25.yaml` 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**：を作成します `TridentOrchestrator` **Trident**をインストール

これで、を作成してTridentをインストールできます `TridentOrchestrator`。必要に応じて、仕様内の属性を使用 `TridentOrchestrator`できます"Tridentのインストールをカスタマイズ"。

```
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
  nodePrep:
  - iscsi
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:24.10
    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:24.10.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v24.10.0
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	オペレータはこのCRを使用してTridentをインストールしています `TridentOrchestrator`。
インストール済み	Tridentは正常にインストールされました。
アンインストール中です	オペレータはTridentをアンインストールしています。 `spec.uninstall=true`
アンインストール済み	Tridentがアンインストールされます。
失敗しました	オペレータはTridentをインストール、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。オペレータは自動的にこの状態から回復しようとします。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。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

作成されたポッドのステータスを確認することで、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`

を使用して、インストールされているTridentのバージョンを確認できます `tridentctl`。

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.10.0        | 24.10.0        |
+----------------+----------------+
```

Tridentオペレータを手動で導入（オフラインモード）

Tridentオペレータを手動で導入して、Tridentをインストールできます。このプロセスは、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保存されているインストールに適用されます。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"標準的な導入のプロセス"ます。

Trident 24.10に関する重要な情報

- Tridentに関する次の重要な情報をお読みください。*

Trident に関するの重要な情報

- TridentでKubernetes 1.31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
- Tridentでは、SAN環境でのマルチパス構成の使用が厳密に適用されます。multipath.confファイルの推奨値はです `find_multipaths: no`。

  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、以降のすべての例で「 kubectl 」ではなく「 OC 」を使用し、「 OC login-u SYSTEM ： admin 」または「 OC login-u kube-admin 」を実行して最初に「 *system:admin 」としてログインします。

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

Tridentインストーラパッケージには、Tridentオペレータの導入とTridentのインストールに必要なすべてのものが含まれています。からTridentインストーラの最新バージョンをダウンロードして展開し"GitHubの_Assets_sectionを参照してください"ます。

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

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

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

```
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

で `/deploy/operator.yaml`、イメージレジストリの場所を反映するように更新し `image: docker.io/netapp/trident-operator:24.10.0` ます。は "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:24.10.0` イメージがすべて1つのレジストリに格納されている場合。
- `image: <your-registry>/netapp/trident-operator:24.10.0` 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

Tridentインストーラには、オペレータのインストールと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。バンドルファイルは、デフォルト設定を使用してオペレータを導入し、Tridentをインストールするための簡単な方法です。

- クラスタでKubernetes 1.24を実行している場合は、を使用し `bundle_pre_1_25.yaml`ます。
- 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.yaml>* is `bundle_pre_1_25.yaml` または `bundle_post_1_25.yaml` 使用しているKubernetesのバージョンに基づきます。

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/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.yaml> `bundle_pre_1_25.yaml` または `bundle_post_1_25.yaml` 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**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:24.10"
tridentImage: "<your-registry>/trident:24.10.0"
```

異なるレジストリ内の画像

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:24.10"
tridentImage: "<your-registry>/trident:24.10.0"
```

手順**6**：を作成します `TridentOrchestrator` **Trident**をインストール

これで、を作成してTridentをインストールできます `TridentOrchestrator`。必要に応じて、仕様内の属性をさらに使用 `TridentOrchestrator`できます"Tridentのインストールをカスタマイズ"。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>/trident-autosupport:24.10
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:24.10.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:24.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/trident:24.10.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v24.10.0
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	オペレータはこのCRを使用してTridentをインストールしています `TridentOrchestrator`。
インストール済み	Tridentは正常にインストールされました。
アンインストール中です	オペレータはTridentをアンインストールしています。 `spec.uninstall=true`
アンインストール済み	Tridentがアンインストールされます。
失敗しました	オペレータはTridentをインストール、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。オペレータは自動的にこの状態から回復しようとします。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。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

作成されたポッドのステータスを確認することで、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`

を使用して、インストールされているTridentのバージョンを確認できます `tridentctl`。

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.10.0        | 24.10.0        |
+----------------+----------------+
```

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Tridentをインストールできます。このプロセスは、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保存されていないインストールに適用されます。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"オフライン導入のプロセス"ます。

Trident 24.10に関する重要な情報

- Tridentに関する次の重要な情報をお読みください。*

Trident に関するの重要な情報

- TridentでKubernetes 1.31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
- Tridentでは、SAN環境でのマルチパス構成の使用が厳密に適用されます。multipath.confファイルの推奨値はです `find_multipaths: no`。

  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**Trident**をインストールする

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. 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. を使用し `helm install`、次の例のように導入環境の名前を指定します。 `100.2404.0`は、インストールするTridentのバージョンです。

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0
--create-namespace --namespace <trident-namespace>
```

> すでにTridentの名前空間を作成している場合'--create-namespaceパラメータは追加の名前空間を作成しません

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 `debug`、次のコマンドを実行します。は、インストールするTridentのバージョンです。 `100.2410.0`

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0
--create-namespace --namespace trident --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	

オプション	説明	デフォルト

`affinity`	ポッド割り当てのアフィニティ	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		values.yamlファイルからデフォルトのアフィニティを削除しないでください。カスタムアフィニティを提供する場合は、デフォルトのアフィニティを拡張します。

`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 」と入力します	、、 `trident`およびその他のイメージのレジストリを指定します `trident-operator`。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。重要：プライベートリポジトリにTridentをインストールする場合、スイッチを使用してリポジトリの場所を指定する場合は `imageRegistry`、リポジトリパスにはを使用しないで`/netapp/`ください。	""
`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 」を参照してください	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。	「真」
`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クラスタでのTridentの動作を有効にできます。	「偽」
`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`	Trident定期AutoSupportレポートをディセーブルにできます。	「偽」

オプション	説明	デフォルト
tridentAutosupportImageTag	Trident AutoSupportコンテナのイメージのタグを上書きできます。	<version>
tridentAutosupportProxy	Trident AutoSupportコンテナがHTTPプロキシ経由で自宅に電話できるようにします。	""
tridentLogFormat	Tridentロギング形式を設定し(text`ます。または `json)	"text"
tridentDisableAuditLog	Trident監査ロガーをディセーブルにします。	「真」
tridentLogLevel	Tridentのログレベルを、、 debug info、、 warn、 `error`または `fatal`に設定 `trace`できます。	"info"
tridentDebug	Tridentのログレベルをに設定できます debug。	「偽」
tridentLogWorkflows	特定のTridentワークフローのトレースロギングまたはログ抑制を有効にできます。	""
tridentLogLayers	トレースロギングまたはログ抑制に対して特定のTridentレイヤをイネーブルにできます。	""
「 tridentImage 」のように入力します	Tridentのイメージを完全に上書きできます。	""
tridentImageTag	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	TridentをWindowsワーカーノードにインストールできるようにします。	「偽」
enableForceDetach	強制切り離し機能を有効にできます。	「偽」
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。	「偽」
cloudProvider	をに設定します "Azure" AKSクラスタで管理対象IDまたはクラウドIDを使用する場合。EKSクラスタでクラウドIDを使用する場合は、「aws」に設定します。	""

オプション	説明	デフォルト
`cloudIdentity`	AKSクラスタでクラウドIDを使用する場合は、ワークロードID（「azure.workload.identity/client-id：xxxxxxxxxx-xxxx-xxxxxxx」）に設定します。EKSクラスタでクラウドIDを使用する場合は、AWS IAMロール（「eks.amazonaws.com/role-arn: arn：aws：iam：：123456：role / Trident -role」）に設定されます。	""
`iscsiSelfHealingInterval`	iSCSIの自己修復が実行される間隔。	`5m0s`
`iscsiSelfHealingWaitTime`	iSCSIの自己修復が、ログアウトとその後のログインを実行して古いセッションの解決を開始するまでの時間。	`7m0s`
`nodePrep`	指定したデータストレージプロトコルを使用してボリュームを管理できるように、Trident でKubernetesクラスタのノードを準備できるようにします。現在 `**iscsi**`サポートされている値は、のみです。	

コントローラポッドとノードポッドについて

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動作します。Tridentボリュームをマウントする可能性がある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。

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Tridentをインストールできます。このプロセスは、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保存されているインストールに適用されます。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"標準的な導入のプロセス"ます。

Trident 24.10に関する重要な情報

- Tridentに関する次の重要な情報をお読みください。*

Trident に関するの重要な情報

- TridentでKubernetes 1.31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
- Tridentでは、SAN環境でのマルチパス構成の使用が厳密に適用されます。multipath.confファイルの推奨値はです `find_multipaths: no`。

  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**Trident**をインストールする

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"。

> プライベートリポジトリにTridentをインストールするときに、スイッチを使用してリポジトリの場所を指定する場合は `imageRegistry`、リポジトリパスにを使用しないで `/netapp/`ください。

手順

1. 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. を使用し `helm install` 、展開とイメージレジストリの場所の名前を指定します。は "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。この例では、 `100.2410.0`はインストールする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
100.2410.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

異なるレジストリ内の画像

```
helm install <name> netapp-trident/trident-operator --version
100.2410.0 --set imageRegistry=<your-registry> --set
operatorImage=<your-registry>/trident-operator:24.10.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:24.06
--set tridentImage=<your-registry>/trident:24.10.0 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

すでにTridentの名前空間を作成している場合'--create-namespaceパラメータは追加の名前空間を作成しません

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 `debug`、次のコマンドを実行します。は、インストールするTridentのバージョンです。 `100.2410.0`

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0
--create-namespace --namespace trident --set tridentDebug=true
```

nodePrep値を追加するには、次のコマンドを実行します。

```
helm install <name> netapp-trident/trident-operator --version 100.2406.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

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。

values.yamlファイルからデフォルトのアフィニティを削除しないでください。カスタムアフィニティを提供する場合は、デフォルトのアフィニティを拡張します。

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	

オプション	説明	デフォルト

`affinity`	ポッド割り当てのアフィニティ	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		values.yamlファイルからデフォルトのアフィニティを削除しないでください。カスタムアフィニティを提供する場合は、デフォルトのアフィニティを拡張します。

`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。	

オプション	説明	デフォルト
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 」と入力します	、、 trident`およびその他のイメージのレジストリを指定します `trident-operator。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。重要：プライベートリポジトリにTridentをインストールする場合、スイッチを使用してリポジトリの場所を指定する場合は imageRegistry、リポジトリパスにはを使用しないで `/netapp/`ください。	""
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 」を参照してください	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。	「真」
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クラスタでのTridentの動作を有効にできます。	「偽」
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

オプション	説明	デフォルト
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	Trident定期AutoSupportレポートをディセーブルにできます。	「偽」
tridentAutosupportImageTag	Trident AutoSupportコンテナのイメージのタグを上書きできます。	<version>
tridentAutosupportProxy	Trident AutoSupportコンテナがHTTPプロキシ経由で自宅に電話できるようにします。	""
tridentLogFormat	Tridentロギング形式を設定し(text`ます。または `json)	"text"
tridentDisableAuditLog	Trident監査ロガーをディセーブルにします。	「真」
tridentLogLevel	Tridentのログレベルを、、 debug info、、 warn、 `error`または `fatal`に設定 `trace`できます。	"info"
tridentDebug	Tridentのログレベルをに設定できます debug。	「偽」
tridentLogWorkflows	特定のTridentワークフローのトレースロギングまたはログ抑制を有効にできます。	""
tridentLogLayers	トレースロギングまたはログ抑制に対して特定のTridentレイヤをイネーブルにできます。	""
「 tridentImage 」のように入力します	Tridentのイメージを完全に上書きできます。	""
tridentImageTag	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	TridentをWindowsワーカーノードにインストールできるようにします。	「偽」
enableForceDetach	強制切り離し機能を有効にできます。	「偽」
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。	「偽」

オプション	説明	デフォルト
`nodePrep`	指定したデータストレージプロトコルを使用してボリュームを管理できるように、Tridentでクラスタのノードを準備できるようにします。現在`**iscsi**`サポートされている値は、のみです。	

Tridentオペレータのインストールをカスタマイズ

Trident演算子を使用すると、仕様の属性を使用してTridentのインストールをカスタマイズでき `TridentOrchestrator`ます。引数で許可される範囲を超えてインストールをカスタマイズする場合 `TridentOrchestrator`は、を使用してカスタムYAMLマニフェストを生成し、必要に応じて変更することを検討して `tridentctl`ください。

コントローラポッドとノードポッドについて

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動作します。Tridentボリュームをマウントする可能性がある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、Tridentがインストールされているネームスペースを示すためににに指定されてい `TridentOrchestrator`ます。このパラメータ*は、Tridentのインストール後は更新できません*。これを実行しようとすると、 `TridentOrchestrator`ステータスがに変わり `Failed`ます。Tridentをネームスペース間で移行することは想定されていません。

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`	Tridentをインストールするネームスペース	`"default"`
「バグ」	Tridentのデバッグを有効にする	「偽」
`enableForceDetach`	`ontap-san`、 `ontap-san-economy`および `ontap-nas-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。	「偽」

パラメータ	説明	デフォルト
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。	「偽」
cloudProvider	をに設定します "Azure" AKSクラスタで管理対象IDまたはクラウドIDを使用する場合。EKSクラスタでクラウドIDを使用する場合は、「aws」に設定します。	""
cloudIdentity	AKSクラスタでクラウドIDを使用する場合は、ワークロードID（「azure.workload.identity/client-id：xxxxxxxxx-xxxx-xxxxxxx」）に設定します。EKSクラスタでクラウドIDを使用する場合は、AWS IAMロール（「eks.amazonaws.com/role-arn: arn：aws：iam：：123456：role / Trident -role」）に設定されます。	""
IPv6	IPv6経由のTridentのインストール	いいえ
k8sTimeout	Kubernetes 処理のタイムアウト	30sec
silenceAutosupport	AutoSupportバンドルをNetAppに送信しない自動	「偽」
「 autosupportImage 」を参照してください	AutoSupport テレメトリのコンテナイメージ	"netapp/trident-autosupport:24.10"
「 autosupportProxy 」と入力します	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	Tridentのアンインストールに使用するフラグ	「偽」
logFormat	使用するTridentログ形式[text、json]	"text"
「 tridentImage 」のように入力します	インストールするTridentイメージ	"netapp/trident:24.10"
「 imageRegistry 」と入力します	形式の内部レジストリへのパス <registry FQDN>[:port][/subpath]	"k8s.gcr.io"（Kubernetes 1.19以降）または "quay.io/k8scsi"
「 kubeletDir 」を参照してください	ホスト上の kubelet ディレクトリへのパス	"/var/lib/kubelet"
wipeout	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
controller’PluginNodeSelector	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
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
「 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
`nodePrep`	指定したデータストレージプロトコルを使用してボリュームを管理できるように、TridentでKubernetesクラスタのノードを準備できるようにします。現在 **`iscsi`** サポートされている値は、のみです。	

ポッドパラメータのフォーマットの詳細については、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

[強制切り離し（Force detach）]は、 `ontap-san-economy`および `onatp-nas-economy`でのみ使用でき `ontap-san`ます。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"。

ドライバを使用する場合 `ontap-nas-economy`は、管理対象のエクスポートポリシーを使用してtaintが適用されたKubernetesノードからのアクセスをTridentが制限できるように、バックエンド構成のパラメータをに `true`設定する必要 `autoExportPolicy`があります。

TridentはKubernetes NGNに依存しているため、許容できないすべてのワークロードのスケジュールを再設定するまで、正常でないノードからテイントを削除しないで `out-of-service`ください。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taintをノードに適用し、 `enableForceDetach`をに設定する `true`と `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute`、Tridentはノードのステータスを確認し、次の処理を行います。

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. Tridentノードオブジェクトを（新しいパブリケーションに対しては安全ではない）としてマークします `dirty`。

Tridentコントローラは、Tridentノードポッドによって（とマークされた後で）ノードが再修飾されるまで、新しいパブリッシュボリューム要求を拒否し `dirty`ます。マウントされたPVCを使用してスケジュールされたワークロード（クラスタノードが正常で準備が完了したあとも）は、Tridentがそのノードを検証できるようになるまで受け入れられません `clean`（新しいパブリケーションに対して安全）。

ノードの健常性が回復してtaintが削除されると、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. ノードが状態（アウトオブサービス状態が削除され、ノードが `Ready`状態）で、古い公開パスがすべてクリーンである場合、 `cleanable`Tridentはノードをとして再登録し、新しい公開ボリュームをそのノードに許可します `clean。

構成例

次の属性を使用できます： [設定オプション] テイギスルバアイ `TridentOrchestrator` をクリックして、インストールをカスタマイズします。

基本的なカスタム設定

これは、基本的なカスタムインストールの例です。

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

ノードセレクタ

この例では、ノードセレクタを使用してTridentをインストールします。

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

Windowsワーカーノード

この例では、WindowsワーカーノードにTridentをインストールします。

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

AKSクラスタ上の管理対象**ID**

この例では、AKSクラスタで管理対象IDを有効にするためにTridentをインストールします。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
```

AKSクラスタ上のクラウド**ID**

この例では、AKSクラスタ上のクラウドIDで使用するTridentをインストールします。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx'
```

EKSクラスタ上のクラウド**ID**

この例では、AKSクラスタ上のクラウドIDで使用するTridentをインストールします。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

GKEのクラウド**ID**

この例では、GKEクラスタにクラウドIDで使用するTridentをインストールします。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

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Tridentをインストールでき `tridentctl`ます。このプロセスは、`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保存されているかどうかにかかわらず、インストールに適用されます。配置をカスタマイズするには `tridentctl`、を参照してください"tridentctl 展開をカスタマイズします"。

Trident 24.10に関する重要な情報

- 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**の重要な情報

- 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
- Tridentでは、SAN環境でのマルチパス構成の使用が厳密に適用されます。multipath.confファイルの推奨値はです `find_multipaths: no`。

  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**Trident**をインストール `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、以降のすべての例で「 kubectl 」ではなく「 OC 」を使用し、「 OC login-u SYSTEM ： admin 」または「 OC login-u kube-admin 」を実行して最初に「 *system:admin 」としてログインします。

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

Tridentインストーラパッケージは、Tridentポッドを作成し、その状態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、ボリュームのプロビジョニングやクラスタホストへの接続などのアクションを実行します。からTridentインストーラの最新バージョンをダウンロードして展開し"GitHubの_Assets_sectionを参照してください"ます。この例では、選択したTridentバージョンで_< Tridentインストーラ-XX.X.X.tar.gz>_を更新します。

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

手順**2**：**Trident**をインストールする

コマンドを実行して、目的のネームスペースにTridentをインストールし `tridentctl install`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:24.10 --trident
-image <your-registry>/trident:24.10.0
```

異なるレジストリ内の画像

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:24.10 --trident
-image <your-registry>/trident:24.10.0
```

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=24.10.0
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

作成されたポッドのステータスを確認することで、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`

を使用して、インストールされているTridentのバージョンを確認できます `tridentctl`。

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.10.0        | 24.10.0        |
+----------------+----------------+
```

構成例

次の例は、を使用してTridentをインストールする場合の設定例 `tridentctl`です。

Windowsノオト

WindowsノードでTridentを実行するには、次の手順を実行します。

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

強制的に切り離し

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
```

tridentctlのインストールをカスタマイズします

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

Tridentインストーラでは、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、次の形式のスイッチを使用してリポジトリの場所を指定することをお勧めし `--image-registry`ます。 `<registry FQDN>[:port]`

> プライベートリポジトリにTridentをインストールするときに、スイッチを使用してリポジトリの場所を指定する場合は `--image-registry`、リポジトリパスにを使用しないで `/netapp/`ください。例： `./tridentctl install --image-registry <image-registry> -n <namespace>`

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

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

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

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

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

著作権に関する情報

商標に関する情報