



Tridentをインストール

Trident

NetApp
October 29, 2026

目次

Tridentをインストール	1
Tridentのインストールについて	1
Trident 26.06に関する重要な情報	1
開始する前に	1
インストール方法を選択してください	1
インストールモードを選択してください	3
方法とモードに基づいてプロセスを選択します	4
インストール方法間の移行	4
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たインストール	5
Tridentオペレータを手動で導入する（標準モード）	5
Tridentオペレーターを手動で導入する（オフラインモード）	10
Helm を使用した Trident オペレーターの導入（標準モード）	16
Helm を使用して Trident オペレーターを展開する（オフライン モード）	23
Tridentオペレータのインストールをカスタマイズ	31
tridentctlを使用してインストールする	41
tridentctl を使用して Trident をインストールする	41
Trident tridentctl インストールのカスタマイズ	45
OpenShift 認定オペレーターを使用したインストール	46
OpenShift OperatorHubを使用してTridentをインストールする	46
OpenShift認定Tridentオペレーターに切り替える	49

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導入できるようにするため、NetAppでは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オペレーター（手動またはHelmを使用）を使用するか、`tridentctl`を使用してインストールできます。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 26.06に関する重要な情報

Trident に関する以下の重要な情報を必ずお読みください。

▼ Tridentに関する重要な情報

- Kubernetes は Trident で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 をアップグレードしてください。
- Trident は SAN 環境でのマルチパス構成の使用を厳格に強制し、multipath.conf ファイル内の推奨値は `find\_multipaths: no` です。

非マルチパス構成の使用、または multipath.conf ファイルでの `find\_multipaths: yes` または `find\_multipaths: smart` 値の使用は、マウントの失敗を引き起こします。Trident は、21.07 リリース以降 `find\_multipaths: no` の使用を推奨しています。

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 を実行し、機能要件が有効になっている、サポートされている Kubernetes クラスターに対する完全な権限。詳細については、["要件"](#)を参照してください。
- サポートされているNetAppストレージシステムへのアクセス。
- すべての Kubernetes ワーカーノードからボリュームをマウントする機能。
- kubectl（または、OpenShiftを使用している場合は oc）がインストールされ、使用するKubernetesクラスタを管理するように設定されたLinuxホスト。
- `KUBECONFIG`環境変数をKubernetesクラスター構成を指すように設定します。
- Docker EnterpriseでKubernetesを使用している場合は、["手順に従ってCLIアクセスを有効にします"](#)。
- クラスターは特権ワークロードをサポートする必要があります。



"[基本的な概念](#)"にまだ慣れていない場合は、今こそそれを実行する絶好の機会です。

インストール方法を選択してください

適切なインストール方法を選択してください。決定を下す前に["方法間の移動"](#)の考慮事項も確認する必要があります。

Tridentオペレーターの使用

手動でのデプロイでもHelmを使用する場合でも、Tridentオペレーター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優れた方法です。["Tridentオペレータの導入をカスタマイズ"](#)を使用して、TridentOrchestrator カスタムリソース（CR）の属性を設定することもできます。

Trident Operatorを使用する利点は次のとおりです：

▼ **Tridentオブジェクトの作成**

Trident オペレーターは、Kubernetes 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。

- オペレーター向けのServiceAccount
- ClusterRole および ClusterRoleBinding を ServiceAccount に追加
- 専用のPodSecurityPolicy（Kubernetes 1.25以前の場合）
- オペレータ自体

▼ **リソースのアカウントビリティ**

クラスタースコープのTridentオペレーターは、クラスターレベルでのTridentインストールに関連するリソースを管理します。これにより、ネームスペーススコープのオペレーターを使用してクラスタースコープのリソースを管理するときに発生する可能性のあるエラーが軽減されます。これは自己修復とパッチ適用に不可欠です。

▼ **自己修復機能**

オペレーターはTridentのインストールを監視し、デプロイメントが削除されたり誤って変更されたりした場合などの問題に対処するための対策を積極的に講じます。trident-operator-`<generated-id>`ポッドが作成され、``TridentOrchestrator`` CRとTridentのインストールが関連付けられます。これにより、クラスター内にTridentのインスタンスが1つだけ存在することが保証され、そのセットアップが制御され、インストールがべき等であることが確認されます。インストールに変更が加えられると（デプロイメントまたはノードデーモンセットの削除など）、オペレーターはそれらを識別し、個別に修正します。

▼ **既存のインストールへの簡単なアップデート**

オペレーターを使用すると、既存のデプロイメントを簡単に更新できます。インストールを更新するには、`TridentOrchestrator`` CR を編集するだけです。

例えば、Tridentでデバッグログを生成する必要があるシナリオを考えてみましょう。これを行うには、``TridentOrchestrator`` にパッチを適用して ``spec.debug`` を ``true`` に設定します：

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

``TridentOrchestrator`` が更新されると、オペレーターは更新を処理し、既存のインストールにパッチを適用します。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可能性があります。

▼ **クリーンな再インストール**

クラスタースコープのTridentオペレータを使用すると、クラスタースコープのリソースを完全に削除できま

す。ユーザは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

クラスターのKubernetesバージョンがサポートされているバージョンにアップグレードされると、オペレーターは既存の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。



クラスターがサポートされていないバージョンにアップグレードされた場合、オペレーターはTridentのインストールを禁止します。Tridentがオペレーターですでにインストールされている場合は、Tridentがサポートされていない Kubernetes バージョンに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。

使用 `tridentctl`

既存の展開をアップグレードする必要がある場合、または展開を高度にカスタマイズしたい場合は、を検討する必要があります。これは、Tridentを導入する従来の方法です。

を使用して、Tridentリソースのマニフェストを生成できます。これには、Tridentがインストールの一部として作成するデプロイメント、デーモンセット、サービスアカウント、およびクラスターロールが含まれます。



22.04リリース以降、Tridentのインストール時にAESキーが再生成されることはなくなります。このリリースでは、Tridentはインストール間で保持される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。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_を選択してください。

インストールモードを選択してください

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）に基づいて展開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、ネットワーク制限を課さないほとんどの環境で動作します。標準インストールモードでは、必要なTrident(`docker.io`およびCSI(`registry.k8s.io`イメージを格納するためにデフォルトのレジストリを使用します。

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、Kubernetesクラスター内のすべての適格ノード上でTridentポッドを起動します

オフラインインストール

エアギャップまたは安全な場所では、オフラインインストールモードが必要になる場合があります。このシナリオでは、必要なTridentおよびCSIイメージを格納するために、単一のプライベートミラーレジストリまたは2つのミラーレジストリを作成できます。



レジストリ構成に関係なく、CSI イメージは 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

リモート インストール プロセスの概要：

- Tridentを導入するリモートマシンに、適切なバージョンの `kubectl` を導入します。
- Kubernetes クラスターから設定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` コマンドを実行して、必要な Kubernetes クラスター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。

方法とモードに基づいてプロセスを選択します

決定したら、適切なプロセスを選択します。

方法	インストールモード
Tridentオペレーター（手動）	"標準インストール"
	"オフラインインストール"
Tridentオペレーター（Helm）	"標準インストール"
	"オフラインインストール"
tridentctl	"標準またはオフラインインストール"

インストール方法間の移行

インストール方法を変更することもできます。これを実行する前に、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。`tridentctl` で導入した場合は、適切なバージョンの `tridentctl` 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。同様に、オペレーターでデプロイする場合は、`TridentOrchestrator` CRを編集し、`spec.uninstall=true` を設定してTridentをアンインストールする必要があります。
- 削除して代わりに使用したいオペレーターベースのデプロイメントがある場合 `tridentctl` Tridentを展開するには、まず `TridentOrchestrator` を編集して `spec.uninstall=true` を設定し、Tridentをアンインストールする必要があります。次に `TridentOrchestrator` とオペレータのデプロイメントを削除します。その後、`tridentctl` を使用してインストールできます。
- 手動のオペレーターベースのデプロイメントがあり、HelmベースのTridentオペレーターをデプロイする場合は、まずオペレーターを手動でアンインストールしてから、Helmの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持つTridentオペレーターをデプロイできます。これを行わないと、HelmベースのTridentオペレーターのデプロイメントは、ラベル検証エラーとアノテーション検証エラーにより失敗します。
- `tridentctl` ベースのデプロイメントを使用している場合は、Tridentをアンインストールせずに、HelmベースまたはOperatorベースのデプロイメントを実行できます。

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：

- `--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。デフォルトでは、`/var/vcap/data/kubelet` です。

`--kubelet-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` デプロイメントで機能することが知られています。

Trident オペレータを使用したインストール

Trident オペレータを手動で導入する（標準モード）

Trident オペレーターを手動で導入して Trident をインストールできます。このプロセスは、Trident 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をお持ちの場合は、"[オフライン展開のプロセス](#)"を使用してください。

Trident 26.06 に関する重要な情報

Trident に関する以下の重要な情報を必ずお読みください。

▼ Trident に関する重要な情報

- Kubernetes 1.37 が Trident で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 をアップグレードしてください。
- Trident は SAN 環境でのマルチパス構成の使用を厳格に強制し、multipath.conf ファイル内の推奨値は `find_multipaths: no` です。

非マルチパス構成の使用、または multipath.conf ファイルでの `find_multipaths: yes` または `find_multipaths: smart` 値の使用は、マウントの失敗を引き起こします。Trident は、21.07 リリース以降 `find_multipaths: no` の使用を推奨しています。

Trident オペレーターを手動で展開して Trident をインストールする

"[インストールの概要](#)"を確認して、インストールの前提条件を満たしていること、および環境に適したインストールオプションが選択されていることを確認します。

開始する前に

インストールを始める前に、Linux ホストにログインし、正常に動作している"[サポートされている Kubernetes クラスタ](#)"を管理していること、および必要な権限を持っていることを確認します。



OpenShift では、以下のすべての例で `oc` を `kubectl` の代わりに使用し、最初に `oc login -u system:admin` または `oc login -u kube-admin` を実行して **system:admin** としてログインしてください。

▼ Details

1. Kubernetes のバージョンを確認します：

```
kubectl version
```

2. クラスタ管理者権限を確認します：

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

3. Docker Hub からイメージを使用するポッドを起動し、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>
```

ステップ1：Tridentインストーラーパッケージをダウンロードする

Trident インストーラー パッケージには、Trident オペレーターを導入して Trident をインストールするために必要なものがすべて含まれています。最新バージョンのTridentインストーラーを["GitHub の Assets セクション"](#)からダウンロードして解凍します。

```
wget https://github.com/NetApp/trident/releases/download/v26.06.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.tar.gz
cd trident-installer
```

ステップ2：TridentOrchestrator CRDを作成する

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

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

ステップ3：Tridentオペレーターを導入する

Tridentインストーラーは、オペレーターをインストールし、関連オブジェクトを作成するために使用できるバンドル ファイルを提供します。バンドル ファイルは、オペレーターを導入し、デフォルト設定を使用してTridentをインストールする簡単な方法です。

- Kubernetes 1.24を実行しているクラスターの場合は、`bundle\_pre\_1\_25.yaml`を使用します。
- Kubernetes 1.25以降を実行しているクラスターの場合は、`bundle\_post\_1\_25.yaml`を使用します。

開始する前に

- デフォルトでは、Tridentインストーラは`trident`ネームスペースにオペレータを導入します。`trident`ネームスペースが存在しない場合は、次のコマンドを使用して作成します：

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

- `trident`名前空間以外の名前空間にオペレーターを導入するには、`serviceaccount.yaml`、`clusterrolebinding.yaml`、`operator.yaml`を更新し、`kustomization.yaml`を使用してバンドルファイルを作成します。
 - a. `kustomization.yaml`を次のコマンドで作成します。ここで、`<bundle.yaml>`は、Kubernetesのバージョンに応じて`bundle\_pre\_1\_25.yaml`または`bundle\_post\_1\_25.yaml`になります。

```
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。`<bundle.yaml>`は`bundle\_pre\_1\_25.yaml`または`bundle\_post\_1\_25.yaml` 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、オペレーターをデプロイします：

```
kubectl create -f deploy/<bundle.yaml>
```

2. オペレーター、デプロイメント、レプリカセットが作成されたことを確認します。

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



Kubernetes クラスターにはオペレーターのインスタンスが **1つ** のみ存在する必要があります。Trident オペレーターの複数のデプロイメントを作成しないでください。

ステップ4：`TridentOrchestrator`を作成してTridentをインストールする

これで`TridentOrchestrator`を作成し、Tridentをインストールできます。オプションで、"[Tridentインストールをカスタマイズ](#)"を`TridentOrchestrator`仕様の属性を使って行うこともできます。

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
```

```
kubectl describe torc trident
```

```
Name:          trident
Namespace:
Labels:         <none>
Annotations:    <none>
API Version:    trident.netapp.io/v1
Kind:           TridentOrchestrator
...
Spec:
  Debug:        true
  Namespace:    trident
  nodePrep:
    - iscsi
Status:
  Current Installation Params:
    IPv6:                false
    Autosupport Hostname:
    Autosupport Image:    netapp/trident-autosupport:26.06
    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:26.06.0
  Message:              Trident installed Namespace:
trident
  Status:                Installed
  Version:                v26.06.0
Events:
  Type Reason Age From Message ----
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed
```

インストールの確認

インストールを確認する方法はいくつかあります。

`TridentOrchestrator`ステータスの使用

`TridentOrchestrator`のステータスは、インストールが成功したかどうかを示し、インストー

ルされた Trident のバージョンを表示します。インストール中、
`TridentOrchestrator` のステータスは `Installing` から `Installed` に変わります。
`Failed` ステータスが表示され、オペレータが自己修復できない場合は、[link:../troubleshooting.html\["ログを確認する"\]](#)。

ステータス	概要
インストールしています	オペレーターは、この TridentOrchestrator CR を使用してTridentをインストールしています。
インストール済み	Tridentが正常にインストールされました。
アンインストール	オペレーターはTridentをアンインストール中です 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した。
失敗	オペレーターはTridentのインストール、パッチ適用、更新、またはアンインストールができませんでした。オペレーターは自動的にこの状態からの回復を試みます。この状態が続く場合は、トラブルシューティングが必要になります。
更新中	オペレーターは既存のインストールを更新しています。
エラー	その `TridentOrchestrator` は使用されません。別のものがすでに存在します。

ポッド作成ステータスの使用

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-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

``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 -n trident version
```

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 26.06.0        | 26.06.0        |
+-----+-----+
```

Tridentオペレーターを手動で導入する（オフラインモード）

Tridentオペレーターを手動で導入して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、["標準導入のプロセス"](#)を使用します。

Tridentに関する重要な情報

Trident に関する以下の重要な情報を必ずお読みください。

▼ Tridentに関する重要な情報

- Kubernetes 1.37 が Trident で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 をアップグレードしてください。
- Trident は SAN 環境でのマルチパス構成の使用を厳格に強制し、multipath.conf ファイル内の推奨値は ``find_multipaths: no``です。

非マルチパス構成の使用、または multipath.conf ファイルでの ``find_multipaths: yes``または ``find_multipaths: smart``値の使用は、マウントの失敗を引き起こします。Trident は、21.07 リリース以降 ``find_multipaths: no``の使用を推奨しています。

Tridentオペレーターを手動で展開してTridentをインストールする

["インストールの概要"](#)を確認して、インストールの前提条件を満たしていること、および環境に適したインストールオプションが選択されていることを確認します。

開始する前に

Linux ホストにログインし、正常に動作している["サポートされている Kubernetes クラスタ"](#)を管理していること、および必要な権限があることを確認します。



OpenShift では、以下のすべての例で `oc` を `kubectl` の代わりに使用し、最初に `oc login -u system:admin` または `oc login -u kube-admin` を実行して **system:admin** として口

グインしてください。

▼ Details

1. Kubernetes のバージョンを確認します：

```
kubectl version
```

2. クラスタ管理者権限を確認します：

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

3. Docker Hub からイメージを使用するポッドを起動し、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>
```

ステップ1：Tridentインストーラーパッケージをダウンロードする

Trident インストーラー パッケージには、Trident オペレーターを導入して Trident をインストールするために必要なものがすべて含まれています。最新バージョンのTridentインストーラーを["GitHub の Assets セクション"](#)からダウンロードして解凍します。

```
wget https://github.com/NetApp/trident/releases/download/v6.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.tar.gz
cd trident-installer
```

ステップ2：TridentOrchestrator CRDを作成する

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

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

ステップ3：オペレータのレジストリの場所を更新する

```
`/deploy/operator.yaml`で、`image: docker.io/netapp/trident-
operator:26.06.0`を更新して、イメージレジストリの場所を反映します。link:.../trident-
```

```
get-started/requirements.html#container-images-and-corresponding-  
kubernetes-versions["TridentとCSIのイメージ"]は  
1つのレジストリまたは異なる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。次に例を示します。
```

- `image: <your-registry>/trident-operator:26.06.0` すべてのイメージが 1 つのレジストリにある場合。
- `image: <your-registry>/netapp/trident-operator:26.06.0` Trident イメージが CSI イメージとは異なるレジストリにある場合。

ステップ4：Tridentオペレーターを導入する

Tridentインストーラーは、オペレーターをインストールし、関連オブジェクトを作成するために使用できるバンドル ファイルを提供します。バンドル ファイルは、オペレーターを導入し、デフォルト設定を使用してTridentをインストールする簡単な方法です。

- Kubernetes 1.24を実行しているクラスターの場合は、`bundle_pre_1_25.yaml`を使用します。
- Kubernetes 1.25以降を実行しているクラスターの場合は、`bundle_post_1_25.yaml`を使用します。

開始する前に

- デフォルトでは、Tridentインストーラーは `trident` ネームスペースにオペレータを導入します。`trident` ネームスペースが存在しない場合は、次のコマンドを使用して作成します：

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

- `trident` `名前空間以外の名前空間にオペレーターを導入するには、`serviceaccount.yaml`、`clusterrolebinding.yaml`、`operator.yaml`を更新し、`kustomization.yaml`を使用してバンドルファイルを作成します。
 - a. `kustomization.yaml`を次のコマンドで作成します。ここで、`<bundle.yaml>`は、Kubernetes のバージョンに応じて `bundle_pre_1_25.yaml` または `bundle_post_1_25.yaml` になります。

```
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。`<bundle.yaml>`は `bundle_pre_1_25.yaml` または `bundle_post_1_25.yaml` 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、オペレーターをデプロイします：

```
kubectl create -f deploy/<bundle.yaml>
```

2. オペレーター、デプロイメント、レプリカセットが作成されたことを確認します。

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



Kubernetes クラスターにはオペレーターのインスタンスが **1 つ** のみ存在する必要があります。Trident オペレーターの複数のデプロイメントを作成しないでください。

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

"TridentとCSIのイメージ"は1つのレジストリまたは異なる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。`deploy/crds/tridentorchestrator\_cr.yaml`を更新して、レジストリ構成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画像

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.06"
tridentImage: "<your-registry>/trident:26.06.0"
```

異なるレジストリ内のイメージ

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.06"
tridentImage: "<your-registry>/trident:26.06.0"
```

ステップ6: `TridentOrchestrator`を作成してTridentをインストールする

これで `TridentOrchestrator`を作成し、Tridentをインストールできます。オプションで、"Tridentインストールをカスタマイズ"をさらに `TridentOrchestrator`仕様の属性を使ってカスタマイズできます。次の例は、TridentとCSIイメージが異なるレジストリに配置されているインストール方法を示しています。

```
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:26.06
```

```

Debug:                true
Image Registry:       <your-registry>
Namespace:            trident
Trident Image:        <your-registry>/trident:26.06.0
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image: <your-registry>/trident-autosupport:26.06
    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:26.06.0
  Message:             Trident installed
  Namespace:           trident
  Status:              Installed
  Version:             v26.06.0
Events:
  Type Reason Age From Message ---- -
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed

```

インストールの確認

インストールを確認する方法はいくつかあります。

`TridentOrchestrator`ステータスの使用

`TridentOrchestrator`のステータスは、インストールが成功したかどうかを示し、インストールされた Trident のバージョンを表示します。インストール中、`TridentOrchestrator`のステータスは `Installing` から `Installed` に変わります。`Failed` ステータスが表示され、オペレータが自己修復できない場合は、[link:../troubleshooting.html](#)["ログを確認する"]。

ステータス	概要
インストールしています	オペレーターは、この TridentOrchestrator CR を使用してTridentをインストールしています。
インストール済み	Tridentが正常にインストールされました。
アンインストール	オペレーターはTridentをアンインストール中です 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した。
失敗	オペレーターはTridentのインストール、パッチ適用、更新、またはアンインストールができませんでした。オペレーターは自動的にこの状態からの回復を試みます。この状態が続く場合は、トラブルシューティングが必要になります。
更新中	オペレーターは既存のインストールを更新しています。
エラー	その `TridentOrchestrator` は使用されません。別のものがすでに存在します。

ポッド作成ステータスの使用

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			

使用 **tridentctl**

`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 -n trident version
```

SERVER VERSION	CLIENT VERSION
26.06.0	26.06.0

Helm を使用した Trident オペレーターの導入（標準モード）

Helm を使用して Trident オペレーターを導入し、Trident をインストールできます。このプロセスは、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をお持ちの場合は、"[オフライン展開のプロセス](#)"を使用してください。

Trident 26.06に関する重要な情報

Trident に関する以下の重要な情報を必ずお読みください。

▼ Tridentに関する重要な情報

- Kubernetes 1.37 が Trident で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 をアップグレードしてください。
- Trident は SAN 環境でのマルチパス構成の使用を厳格に強制し、multipath.conf ファイル内の推奨値は `find\_multipaths: no` です。

非マルチパス構成の使用、または multipath.conf ファイルでの `find\_multipaths: yes` または `find\_multipaths: smart` 値の使用は、マウントの失敗を引き起こします。Trident は、21.07 リリース以降 `find\_multipaths: no` の使用を推奨しています。

Helmを使用してTridentオペレーターを導入し、Tridentをインストールする

Trident "[Helmチャート](#)" を使用すると、Trident オペレーターを導入し、Trident を 1 つのステップでインストールできます。

"[インストールの概要](#)"を確認して、インストールの前提条件を満たしていること、および環境に適したインストールオプションが選択されていることを確認します。

開始する前に

"[導入の前提条件](#)"に加えて"[Helm バージョン3](#)"が必要です。

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. `helm install`を使用して、次の例のようにデプロイメントの名前を指定します。ここで、`100.2606.0`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 100.2606.0
--create-namespace --namespace <trident-namespace>
```



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

```
`helm
list`を使用して、名前、名前空間、チャート、ステータス、アプリのバージョン、リビジョン
番号などのインストールの詳細を確認できます。
```

インストール中に構成データを渡す

インストール中に構成データを渡す方法は2つあります：

オプション	概要
--values (または -f)	オーバーライドを含む YAML ファイルを指定します。これは複数回指定することができ、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、`debug`のデフォルト値を変更するには、次のコマンドを実行します。ここで、`100.2606.0`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 100.2606.0
--create-namespace --namespace trident --set tridentDebug=true
```

構成オプション

この表と、Helm チャートの一部である `values.yaml` ファイルは、キーとそのデフォルト値のリストを提供します。

オプション	概要	デフォルト
nodeSelector	Pod割り当て用のノードラベル	
podAnnotations	Podアノテーション	
deploymentAnnotations	デプロイメントアノテーション	
tolerations	pod割り当ての許容範囲	

オプション	概要	デフォルト
affinity	Pod割り当ての親和性	<pre> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDur ingExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div>  <p>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。</p> </div>
tridentContr ollerPluginN odeSelector	ポッドの追加ノードセクター。 詳細については、 [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。	
tridentContr ollerPluginT olerations	ポッドの Kubernetes 許容値をオーバーライドします。詳細については、 [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。	
tridentNodeP luginNodeSel ector	ポッドの追加ノードセクター。 詳細については、 [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。	
tridentNodeP luginTolerat ions	ポッドの Kubernetes 許容値をオーバーライドします。詳細については、 [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	概要	デフォルト
imageRegistry	trident-operator、trident、およびその他のイメージのレジストリを識別します。デフォルトを受け入れ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、リポジトリの場所を指定するために `imageRegistry` スイッチを使用するときは、リポジトリパスに `/netapp/` を使用しないでください。	""
imagePullPolicy	`trident-operator` のイメージプルポリシーを設定します。	IfNotPresent
imagePullSecrets	trident-operator、trident、およびその他のイメージのイメージプルシークレットを設定します。	
kubeletDir	kubelet の内部状態のホストの場所を上書きできます。	"/var/lib/kubelet"
operatorLogLevel	Tridentオペレータのログレベルを trace、debug、info、warn、error、または `fatal` に設定できます。	"info"
operatorDebug	Trident オペレータのログレベルをデバッグに設定できます。	true
operatorImage	`trident-operator` の画像の完全な上書きを許可します。	""
operatorImageTag	`trident-operator` イメージのタグを上書きできます。	""
tridentIPv6	Trident が IPv6 クラスターで動作できるようにします。	false
tridentK8sTimeout	ほとんどの Kubernetes API 操作のデフォルトの 30 秒のタイムアウトをオーバーライドします（ゼロ以外の場合は秒単位）。	0
tridentHttpRequestTimeout	HTTPリクエストのデフォルトの90秒のタイムアウトを上書きします。`0s` はタイムアウトの無限の期間です。負の値は許可されません。	"90s"
tridentSilenceAutosupport	Tridentの定期的なAutoSupportレポートを無効にすることができます。	false

オプション	概要	デフォルト
tridentAutoSupportImageTag	Trident AutoSupport コンテナのイメージタグを上書きできます。	<version>
tridentAutoSupportProxy	Trident AutoSupport コンテナが HTTP プロキシ経由でホームに通信できるようにします。	""
tridentLogFormat	Trident ログ形式を設定します (text または json)。	"text"
tridentDisableAuditLog	Trident 監査ログを無効にします。	true
tridentLogLevel	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	Trident のログレベルを `debug` に設定できるようにします。ノードヘルスチェック (NHC) オペレータとの連携により、強制デタッチプロセスを自動化できます。詳細については、 "Trident を使用したステートフルアプリケーションのフェイルオーバーの自動化" を参照してください。	false
tridentLogWorkflows	特定の Trident ワークフローでトレースログまたはログ抑制を有効にできます。	""
tridentLogLayers	特定の Trident レイヤーでトレースログまたはログ抑制を有効にできます。	""
tridentImage	Trident の画像の完全な上書きを許可します。	""
tridentImageTag	Trident の画像のタグを上書きできます。	""
tridentProbePort	Kubernetes の liveness/readiness プロブに使用されるデフォルトのポートをオーバーライドできます。	""
windows	Windows ワーカー ノードに Trident をインストールできるようにします。	false
enableForceDetach	強制デタッチ機能を有効にできます。	false
excludePodSecurityPolicy	オペレータ Pod セキュリティ ポリシーを作成から除外します。	false

オプション	概要	デフォルト
cloudProvider	AKS クラスターでマネージド ID またはクラウド ID を使用する場合は "Azure" に設定します。EKS クラスターでクラウド ID を使用する場合は「AWS」に設定します。	""
cloudIdentity	AKS クラスターでクラウド ID を使用する場合は、ワークロード ID ("azure.workload.identity/client-id: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX") に設定します。EKS クラスターでクラウド 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") に設定します。	""
iscsiSelfHealingInterval	iSCSI 自己修復が呼び出される間隔。	5m0s
iscsiSelfHealingWaitTime	iSCSI 自己修復がログアウトとそれに続くログインを実行することで、古いセッションを解決しようと試みるまでの期間。	7m0s
nodePrep	Trident が指定され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するために、Kubernetes クラスターのノードを準備できるようにします。*現在、`iscsi` がサポートされている唯一の値です。*注：OpenShift 4.19 以降、この機能でサポートされている Trident の最小バージョンは 25.06.1 です。	
enableConcurrency	スループット向上のために Trident コントローラの同時操作を有効にします。サポートされているドライバの詳細については、リンク先を参照してください ： https://docs.netapp.com/us-en/trident/trident-use/controller-scalability.html#controller-scalability [コントローラのスケラビリティに関するドキュメント]	false
k8sAPIQPS	Kubernetes API サーバーと通信するときにコントローラーが使用する 1 秒あたりのクエリ数 (QPS) の制限。バースト値は QPS 値に基づいて自動的に設定されます。	100 ; オプション

オプション	概要	デフォルト
resources	<p>Trident コントローラ、ノード、およびオペレータ ポッドの Kubernetes リソース制限とリクエストを設定します。Kubernetes でのリソース割り当てを管理するために、各コンテナとサイドカーの CPU とメモリを設定できます。</p> <p>リソース要求と制限の設定の詳細については、"Pod と Container のリソース管理"を参照してください。</p> <div> <div>⚠</div> <ul style="list-style-type: none"> コンテナまたはフィールドの名前を変更しないでください。 インデントを変更しないでください。YAML インデントは適切な解析に重要です。 デフォルトでは制限は適用されません。リクエストのみにデフォルト値があり、指定されていない場合は自動的に適用されます。 コンテナ名は、ポッド仕様に表示されるとおりにリストされます。 サイドカーは各メイン コンテナの下にリストされます。 TORC `status.CurrentInstallationParams` `フィールドを確認して、現在適用されている値を表示します。 </div> <div> <div>i</div> </div>	<pre>resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident-autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory: node:</pre>

オプション	概要	デフォルト
httpsMetrics	Prometheus メトリック エンドポイントに対して HTTPS を有効にします。	false
hostNetwork	Trident コントローラのホスト ネットワークを有効にします。これは、マルチホーム ネットワークでフロントエンド トラフィックとバックエンド トラフィックを分離する場合に便利です。	false

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して実行され、クラスター内の各ワーカーノード上のノードポッドも実行されます。ノードポッドは、Tridentボリュームをマウントする可能性のあるホスト上で実行されている必要があります。

Kubernetes ["ノードセクター"](#)および["tolerations と taints"](#)は、ポッドを特定のノードまたは優先ノードで実行するように制限するために使用されます。`ControllerPlugin`および`NodePlugin`を使用して、制約とオーバーライドを指定できます。

- コントローラー プラグインは、スナップショット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 プラグインは、ストレージをノードに接続する処理を行います。

Helm を使用して Trident オペレーターを展開する（オフライン モード）

Helm を使用して Trident オペレーターを導入し、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、["標準導入のプロセス"](#)を使用します。

Trident 26.06に関する重要な情報

Trident に関する以下の重要な情報を必ずお読みください。

▼ Tridentに関する重要な情報

- Kubernetes 1.37 が Trident で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 をアップグレードしてください。
- Trident は SAN 環境でのマルチパス構成の使用を厳格に強制し、multipath.conf ファイル内の推奨値は`find\_multipaths: no`です。

非マルチパス構成の使用、または multipath.conf ファイルでの`find\_multipaths: yes`または`find\_multipaths: smart`値の使用は、マウントの失敗を引き起こします。Trident は、21.07 リリース以降`find\_multipaths: no`の使用を推奨しています。

Helmを使用してTridentオペレーターを導入し、Tridentをインストールする

Trident ["Helmチャート"](#) を使用すると、Trident オペレーターを導入し、Trident を 1 つのステップでインストールできます。

["インストールの概要"](#)を確認して、インストールの前提条件を満たしていること、および環境に適したインストールオプションが選択されていることを確認します。

開始する前に

["導入の前提条件"](#)に加えて["Helm バージョン3"](#)が必要です。



プライベートリポジトリにTridentをインストールする際、`imageRegistry` 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、リポジトリパスに `/netapp/` を使用しないでください。

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. `helm install`を使用して、デプロイメントの名前とイメージレジストリの場所を指定します。["Trident とCSIのイメージ"](#)は1つのレジストリまたは異なる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、`100.2606.0`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画像

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0 --set imageRegistry=<your-registry> --create-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

異なるレジストリ内のイメージ

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0 --set imageRegistry=<your-registry> --set operatorImage=<your-registry>/trident-operator:26.06.0 --set tridentAutosupportImage=<your-registry>/trident-autosupport:26.06 --set tridentImage=<your-registry>/trident:26.06.0 --create-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```



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

```
`helm list`を使用して、名前、名前空間、チャート、ステータス、アプリのバージョン、リビジョン
```

番号などのインストールの詳細を確認できます。

インストール中に構成データを渡す

インストール中に構成データを渡す方法は2つあります：

オプション	概要
--values (または -f)	オーバーライドを含む YAML ファイルを指定します。これは複数回指定することができ、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、`debug` のデフォルト値を変更するには、次のコマンドを実行します。ここで、`100.2606.0` 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 100.2606.0
--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 100.2606.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

構成オプション

この表と、Helm チャートの一部である `values.yaml` ファイルは、キーとそのデフォルト値のリストを提供します。



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	概要	デフォルト
nodeSelector	Pod 割り当て用のノードラベル	
podAnnotations	Pod アノテーション	
deploymentAnnotations	デプロイメントアノテーション	
tolerations	pod 割り当ての許容範囲	

オプション	概要	デフォルト
affinity	Pod割り当ての親和性	<pre> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div>  <p>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。</p> </div>
tridentControllerPluginNodeSelector	ポッドの追加ノードセレクター。詳細については、 "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	ポッドの Kubernetes 許容値をオーバーライドします。詳細については、 "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	概要	デフォルト
tridentNodePluginNodeSelector	ポッドの追加ノードセクター。 詳細については、" 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	ポッドの Kubernetes 許容値をオーバーライドします。詳細については、" 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	trident-operator、trident、およびその他のイメージのレジストリを識別します。デフォルトを受け入れ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、リポジトリの場所を指定するために `imageRegistry` スイッチを使用するときは、リポジトリパスに `/netapp/` を使用しないでください。	""
imagePullPolicy	`trident-operator` のイメージプルポリシーを設定します。	IfNotPresent
imagePullSecrets	trident-operator、trident、およびその他のイメージのイメージプルシークレットを設定します。	
kubeletDir	kubelet の内部状態のホストの場所を上書きできます。	"/var/lib/kubelet"
operatorLogLevel	Tridentオペレータのログレベルを trace、debug、info、warn、error、または `fatal` に設定できます。	"info"
operatorDebug	Trident オペレータのログレベルをデバッグに設定できます。	true
operatorImage	`trident-operator` の画像の完全な上書きを許可します。	""
operatorImageTag	`trident-operator` イメージのタグを上書きできます。	""
tridentIPv6	Trident が IPv6 クラスターで動作できるようにします。	false

オプション	概要	デフォルト
tridentK8sTimeout	<p>ほとんどの Kubernetes API 操作のデフォルトの 180 秒のタイムアウトをオーバーライドします（ゼロ以外の場合は秒単位）。</p> <div>  <p><code>`tridentK8sTimeout`</code> パラメータは、Tridentのインストールにのみ適用されます。</p> </div>	180
tridentHttpRequestTimeout	HTTPリクエストのデフォルトの90秒のタイムアウトを上書きします。`0s`はタイムアウトの無限の期間です。負の値は許可されません。	"90s"
tridentSilenceAutosupport	Tridentの定期的なAutoSupportレポートを無効にすることができます。	false
tridentAutosupportImageTag	Trident AutoSupportコンテナのイメージタグを上書きできます。	<version>
tridentAutosupportProxy	Trident AutoSupport コンテナが HTTP プロキシ経由でホームに通信できるようにします。	""
tridentLogFormat	Tridentログ形式を設定します（ <code>text`または`json`</code> ）。	"text"
tridentDisableAuditLog	Trident監査ロガーを無効にします。	true
tridentLogLevel	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	Tridentのログレベルを`debug`に設定できます。	false
tridentLogWorkflows	特定のTridentワークフローでトレースログまたはログ抑制を有効にできます。	""
tridentLogLayers	特定のTridentレイヤーでトレースログまたはログ抑制を有効にできます。	""
tridentImage	Tridentの画像の完全な上書きを許可します。	""
tridentImageTag	Tridentの画像のタグを上書きできます。	""

オプション	概要	デフォルト
tridentProbePort	Kubernetes の liveness/readiness プロブに使用されるデフォルトのポートをオーバーライドできます。	""
windows	Windows ワーカー ノードに Trident をインストールできるようにします。	false
enableForceDetach	強制デタッチ機能を有効にできます。ノードヘルスチェック (NHC) オペレータと連携することで、強制デタッチプロセスを自動化できます。詳細については、" Trident を使用したステートフルアプリケーションのフェイルオーバーの自動化 "を参照してください。	false
excludePodSecurityPolicy	オペレータ Pod セキュリティ ポリシーを作成から除外します。	false
nodePrep	<p>Tridentが指定され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するために、Kubernetes クラスターのノードを準備できるようにします。現在、`iscsi` がサポートされている唯一の値です。</p> <div>  <p>OpenShift 4.19 以降、この機能でサポートされる Trident の最小バージョンは 25.06.1 です。</p> </div>	

オプション	概要	デフォルト
resources	<p>Trident コントローラ、ノード、およびオペレータ ポッドの Kubernetes リソース制限とリクエストを設定します。Kubernetes でのリソース割り当てを管理するために、各コンテナとサイドカーの CPU とメモリを設定できます。</p> <p>リソース要求と制限の設定の詳細については、"Pod と Container のリソース管理"を参照してください。</p> <div> <div>⚠</div> <ul style="list-style-type: none"> コンテナまたはフィールドの名前を変更しないでください。 インデントを変更しないでください。YAML インデントは適切な解析に重要です。 </div> <div> <div>i</div> <ul style="list-style-type: none"> デフォルトでは制限は適用されません。リクエストのみにデフォルト値が設定されます。 コンテナ名は、ポッド仕様に表示されるとおりにリストされます。 サイドカーは各メイン コンテナの下にリストされます。 TORC <code>`status.CurrentInstallationParams`</code> フィールドを確認して、現在適用されている値を表示します。 </div>	<pre>resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident-autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory:</pre>

Tridentオペレータのインストールをカスタマイズ

Trident オペレータを使用すると、TridentOrchestrator spec 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ます。TridentOrchestrator 引数で許可される範囲を超えてインストールをカスタマイズする場合は、tridentctl を使用して、必要に応じて変更できるカスタム YAML マニフェストを生成することを検討してください。

コントローラポッドとノードポッドについて

Tridentは、クラスター内の各ワーカー ノード上で単一のコントローラー ポッドとノード ポッドとして実行されます。ノードポッドは、Tridentボリュームをマウントする可能性のあるホスト上で実行されている必要があります。

Kubernetes"ノードセクター"および"[tolerations](#) と [taints](#)"は、ポッドを特定のノードまたは優先ノードで実行するように制限するために使用されます。`ControllerPlugin`および`NodePlugin`を使用して、制約とオーバーライドを指定できます。

- コントローラー プラグインは、スナップショット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 プラグインは、ストレージをノードに接続する処理を行います。

構成オプション



`spec.namespace`は`TridentOrchestrator`で指定され、Tridentがインストールされている名前空間を示します。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*。更新しようとすると、`TridentOrchestrator`のステータスが`Failed`に変更されます。Tridentは名前空間間で移行することを意図していません。

この表には`TridentOrchestrator`属性の詳細が記載されています。

パラメータ	概要	デフォルト
namespace	Tridentをインストールする名前空間	"default"
debug	Tridentのデバッグを有効にする	false
enableForceDetach	ontap-san、ontap-san-economy、ontap-nas、および`ontap-nas-economy`のみ。Kubernetes Non-Graceful Node Shutdown (NGNS) と連携して、ノードが正常でなくなった場合に、マウントされたボリュームを持つ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ー管理者に付与します。詳細については、" Tridentを使用したステートフルアプリケーションのフェイルオーバーの自動化 "を参照してください。	false
windows	`true`に設定すると、Windows ワーカー ノードへのインストールが有効になります。	false

パラメータ	概要	デフォルト
cloudProvider	AKS クラスターでマネージド ID またはクラウド ID を使用する場合は ``Azure`` に設定します。 ``AWS`` に設定します (EKS クラスターでクラウド ID を使用する場合)。 ``GCP`` に設定します (GKE クラスターでクラウド ID を使用する場合)。	""
cloudIdentity	AKS クラスターでクラウド ID を使用する場合は、ワークロード ID ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx") に設定します。 EKS クラスターでクラウド 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") に設定します。 GKE クラスターでクラウド ID を使用する場合は、クラウド ID ("iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com") に設定します。	""
IPv6	IPv6 経由で Trident をインストール	false
k8sTimeout	<div>  `k8sTimeout` パラメータは、Trident のインストールにのみ適用されます。 </div>	180sec
silenceAutosupport	AutoSupport バンドルを NetApp に自動的に送信しない	false
autosupportImage	Autosupport Telemetry のコンテナイメージ	"netapp/trident-autosupport10"
autosupportProxy	Autosupport テレメトリを送信するためのプロキシのアドレス/ポート	"http://proxy.example.com:8888"
uninstall	Trident をアンインストールするために使用するフラグ	false
logFormat	Trident で使用するログ形式 [text,json]	"text"
tridentImage	インストールする Trident イメージ	"netapp/trident:26.06"
imageRegistry	内部レジストリへのパス。形式は <registry FQDN>[:port] [/subpath]	"registry.k8s.io"
kubeletDir	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ためのシークレット	

パラメータ	概要	デフォルト
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します。 IfNotPresent ノード上にイメージがまだ存在しない場合にのみイメージをプルします。 Never イメージを決してプルしません。	IfNotPresent
controllerPluginNodeSelector	ポッドの追加ノードセクター。 `pod.spec.nodeSelector`と同じ形式に従います。	デフォルトなし；オプション
controllerPluginTolerations	ポッドの Kubernetes 許容値をオーバーライドします。 `pod.spec.Tolerations`と同じ形式に従います。	デフォルトなし；オプション
nodePluginNodeSelector	ポッドの追加ノードセクター。 `pod.spec.nodeSelector`と同じ形式に従います。	デフォルトなし；オプション
nodePluginTolerations	ポッドの Kubernetes 許容値をオーバーライドします。 `pod.spec.Tolerations`と同じ形式に従います。	デフォルトなし；オプション
nodePrep	Tridentが指定され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するために、Kubernetes クラスターのノードを準備できるようにします。現在、`iscsi`がサポートされている唯一の値です。  OpenShift 4.19 以降、この機能でサポートされる Trident の最小バージョンは 25.06.1 です。	
k8sAPIQPS	Kubernetes API サーバーと通信するときにコントローラーが使用する 1 秒あたりのクエリ数 (QPS) の制限。バースト値は QPS 値に基づいて自動的に設定されます。	100；オプション
enableConcurrency	スループット向上のために Trident コントローラの同時操作を有効にします。サポートされているドライバーの詳細については、リンク先を参照してください： https://docs.netapp.com/us-en/trident/trident-use/controller-scalability.html#controller-scalability [コントローラーのスケーラビリティに関するドキュメント]	resources

パラメータ	概要	デフォルト
<p>Trident コントローラーと ノード ポッドの Kubernetes リソース制限 とリクエストを設定しま す。Kubernetes でのリソ ース割り当てを管理する ために、各コンテナとサ イドカーの CPU とメモリ を設定できます。</p> <p>リソース要求と制限の設 定の詳細について は、"Pod と Container の リソース管理"を参照して ください。</p> <div> <div>  <ul style="list-style-type: none"> • コンテ ナまた はフィ ールド の名前 を変更 しない でくだ さい。 • インデ ントを 変更し ないで くださ い。YA ML イ ンデン トは適 切な解 析に重 要で す。 </div> <div>  <ul style="list-style-type: none"> • デフォ ルトで は制限 は適用 されま せん。リク エストの みにデ フォルト 値が あり、 指定さ </div> </div>	<pre> resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident-autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory: node: </pre>	<p>httpsMetrics</p>

パラメータ	概要	デフォルト
Prometheus メトリック エンドポイントに対して HTTPS を有効にします。	false	hostNetwork



ポッドパラメータのフォーマットの詳細については、"[Pod をノードに割り当てる](#)"を参照してください。

サンプル構成

[\[構成オプション\]](#)の属性を使用して `TridentOrchestrator` を定義し、インストールをカスタマイズできます。

▼ 基本的なカスタム設定

この例は、`cat deploy/crds/tridentorchestrator\_cr\_imagepullsecrets.yaml` コマンドの実行後に作成されたもので、基本的なカスタム インストールを表します：

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

▼ ノードセクタ

この例では、ノードセクターを使用して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

この例は、`cat deploy/crds/tridentorchestrator\_cr.yaml` コマンドの実行後に作成され、Windows ワーカーノードに 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
  windows: true
```

▼ AKS クラスター上の管理対象 ID

この例では、AKS クラスターでマネージド 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 クラスターのクラウド ID

この例では、AKS クラスター上のクラウド ID で使用するために 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"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-
xxxx-xxxxxxxxxxxx'
```

▼ EKS クラスターのクラウド ID

この例では、AKS クラスター上のクラウド ID で使用するために 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: "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 クラスタ上のクラウド ID で使用するために 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
```

▼ TridentコントローラーとTrident Linuxノードポッド用のKubernetesリソースリクエストと制限の設定

この例では、TridentコントローラーとTrident Linuxノードポッドに対するKubernetesリソースリクエストと制限を設定します。



免責事項：この例で示されている要求値と制限値は、デモンストレーションのみを目的としています。環境とワークロードの要件に基づいてこれらの値を調整してください。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
```

```

        memory: 80Mi
    limits:
        cpu: 200m
        memory: 256Mi
# sidecars
csi-provisioner:
    requests:
        cpu: 2m
        memory: 20Mi
    limits:
        cpu: 100m
        memory: 64Mi
csi-attacher:
    requests:
        cpu: 2m
        memory: 20Mi
    limits:
        cpu: 100m
        memory: 64Mi
csi-resizer:
    requests:
        cpu: 3m
        memory: 20Mi
    limits:
        cpu: 100m
        memory: 64Mi
csi-snapshotter:
    requests:
        cpu: 2m
        memory: 20Mi
    limits:
        cpu: 100m
        memory: 64Mi
trident-autosupport:
    requests:
        cpu: 1m
        memory: 30Mi
    limits:
        cpu: 50m
        memory: 128Mi
node:
    linux:
        trident-main:
            requests:
                cpu: 10m
                memory: 60Mi

```

```

    limits:
      cpu: 200m
      memory: 256Mi
# sidecars
node-driver-registrar:
  requests:
    cpu: 1m
    memory: 10Mi
  limits:
    cpu: 50m
    memory: 32Mi

```

▼ **Kubernetes** のリソースリクエストと制限の設定（**Trident** コントローラーと **Trident Windows** および **Linux** ノードポッド）

この例では、TridentコントローラーとTrident WindowsおよびLinuxノードポッドのKubernetesリソースリクエストと制限を設定します。



免責事項：この例で示されている要求値と制限値は、デモンストレーションのみを目的としています。環境とワークロードの要件に基づいてこれらの値を調整してください。

```

apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
    # sidecars
    csi-provisioner:
      requests:
        cpu: 2m
        memory: 20Mi
      limits:

```

```
    cpu: 100m
    memory: 64Mi
csi-attacher:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-resizer:
  requests:
    cpu: 3m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-snapshotter:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
trident-autosupport:
  requests:
    cpu: 1m
    memory: 30Mi
  limits:
    cpu: 50m
    memory: 128Mi
node:
  linux:
    trident-main:
      requests:
        cpu: 10m
        memory: 60Mi
      limits:
        cpu: 200m
        memory: 256Mi
    # sidecars
    node-driver-registrar:
      requests:
        cpu: 1m
        memory: 10Mi
      limits:
        cpu: 50m
```

```

        memory: 32Mi
windows:
  trident-main:
    requests:
      cpu: 6m
      memory: 40Mi
    limits:
      cpu: 200m
      memory: 128Mi
# sidecars
node-driver-registrar:
  requests:
    cpu: 6m
    memory: 40Mi
  limits:
    cpu: 100m
    memory: 128Mi
liveness-probe:
  requests:
    cpu: 2m
    memory: 40Mi
  limits:
    cpu: 50m
    memory: 64Mi

```

tridentctlを使用してインストールする

tridentctl を使用して Trident をインストールする

Tridentは`tridentctl`を使用してインストールできます。このプロセスは、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保存されているかどうかに関係なく適用されます。`tridentctl`の導入をカスタマイズするには、"[tridentctl 導入環境のカスタマイズ](#)"を参照してください。

Trident10に関する重要な情報

Trident に関する以下の重要な情報を必ずお読みください。

▼ Tridentに関する重要な情報

- Kubernetes 1.27 が Trident で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 をアップグレードしてください。
- Trident は SAN 環境でのマルチパス構成の使用を厳格に強制し、multipath.conf ファイル内の推奨値は`find\_multipaths: no`です。

非マルチパス構成の使用、または multipath.conf ファイルでの`find\_multipaths: yes`または

`find\_multipaths: smart`値の使用は、マウントの失敗を引き起こします。Trident は、 21.07 リリース以降 `find\_multipaths: no` の使用を推奨しています。

Tridentを使用してインストール `tridentctl`

"[インストールの概要](#)"を確認して、インストールの前提条件を満たしていること、および環境に適したインストールオプションが選択されていることを確認します。

開始する前に

インストールを始める前に、Linuxホストにログインし、正常に動作している"[サポートされている Kubernetes クラスタ](#)"を管理していること、および必要な権限を持っていることを確認します。



OpenShift では、以下のすべての例で `oc` を `kubectl` の代わりに使用し、最初に `oc login -u system:admin` または `oc login -u kube-admin` を実行して **system:admin** としてログインしてください。

▼ Details

1. Kubernetes のバージョンを確認します：

```
kubectl version
```

2. クラスタ管理者権限を確認します：

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

3. Docker Hub からイメージを使用するポッドを起動し、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>
```

ステップ1：Tridentインストーラーパッケージをダウンロードする

Tridentインストーラーパッケージは、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オブジェクトを構成し、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ーホストへの接続などのアクションを実行します。最新バージョンのTridentインストーラーを"[GitHub の Assets セクション](#)"からダウンロードして解凍します。例の `_<trident-installer-XX.XX.X.tar.gz>_` を選択したTridentバージョンで更新します。

```
wget https://github.com/NetApp/trident/releases/download/v26.06.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.tar.gz
cd trident-installer
```

ステップ2：Tridentをインストールする

目的のネームスペースに Trident をインストールするには、`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画像

```
./tridentctl install -n trident --image-registry <your-registry>  
--autosupport-image <your-registry>/trident-autosupport:26.06 --trident  
-image <your-registry>/trident:26.06.0
```

異なる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:26.06 --trident  
-image <your-registry>/trident:26.06.0
```

インストールの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=26.10.0  
INFO Trident installation succeeded.  
....
```

インストールの確認

ポッド作成ステータスを使用してインストールを確認するか、`tridentctl`を使用できます。

ポッド作成ステータスの使用

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) が*Running*ステータスでない場合、プラットフォームはインストールされていません。`-d`を使用して["デバッグモードをオンにする"](#)問題のトラブルシューティングを行います。

使用 tridentctl

`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 -n trident version
```

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 26.06.0       | 26.06.0       |
+-----+-----+
```

サンプル構成

以下の例は、`tridentctl`を使用してTridentをインストールするためのサンプル構成を示しています。

▼ Windows ノード

Windows ノードで Trident を実行できるようにするには：

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

▼ 強制的に切り離す

詳細については、["Tridentを使用したステートフルアプリケーションのフェイルオーバーの自動化"](#)を参照し

てください。

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

▼ 同時実行Tridentコントローラー操作を有効にする

スループットを向上させるために同時 Trident コントローラー操作を有効にするには、この例に示すように、インストール時に `--enable-concurrency` オプションを追加します。サポートされているドライバーの詳細については、リンク先を参照してください：<https://docs.netapp.com/us-en/trident/trident-use/controller-scalability.html#controller-scalability>[コントローラーのスケーラビリティに関するドキュメント]

```
tridentctl install --enable-concurrency -n trident
```

Trident tridentctl インストールのカスタマイズ

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。

インストーラについて

Tridentインストーラを使用すると、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、`--trident-image`を使用してイメージ名を指定できます。Tridentイメージと必要なCSIサイドカーイメージをプライベートリポジトリにコピーした場合は、`--image-registry`スイッチを使用してそのリポジトリの場所を指定することをお勧めします。このスイッチは `



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

Kubernetesのディストリビューションを使用している場合、`kubelet`が通常の`/var/lib/kubelet`とは異なるパスにデータを保存する場合は、`--kubelet-dir`を使用して代替パスを指定できます。

インストーラーの引数で許可されてい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、展開ファイルをカスタマイズすることもできます。`--generate-custom-yaml`パラメータを使用すると、インストーラーの`setup`ディレクトリに以下のYAMLファイルが作成されます：

- 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、`--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
```

OpenShift 認定オペレーターを使用したインストール

OpenShift OperatorHubを使用してTridentをインストールする

Red Hat OpenShiftを使用している場合は、Red Hat認定オペレーターを使用してNetApp Tridentをインストールできます。この手順を使用して、Red Hat OpenShift Container PlatformからTridentをインストールします。

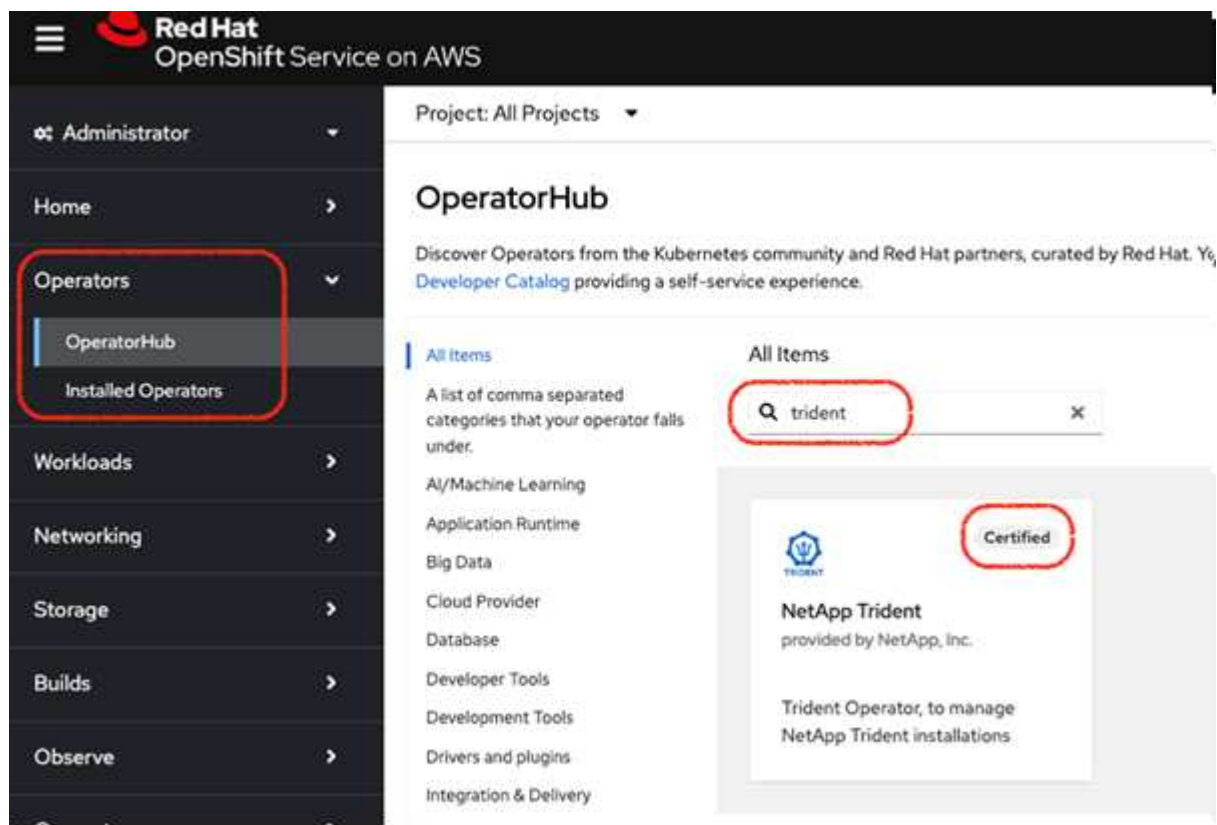
開始する前に

インストールを始める前に、"[Tridentインストールのための環境を準備する](#)"。

Tridentオペレーターを検索してインストールする

手順

1. OpenShift OperatorHubに移動し、NetApp Tridentを検索します。



2. **NetApp Trident** をクリックして、インストール設定を開きます。
3. 必要なオプションを選択し、**Install** をクリックしてOperator構成を開きます。

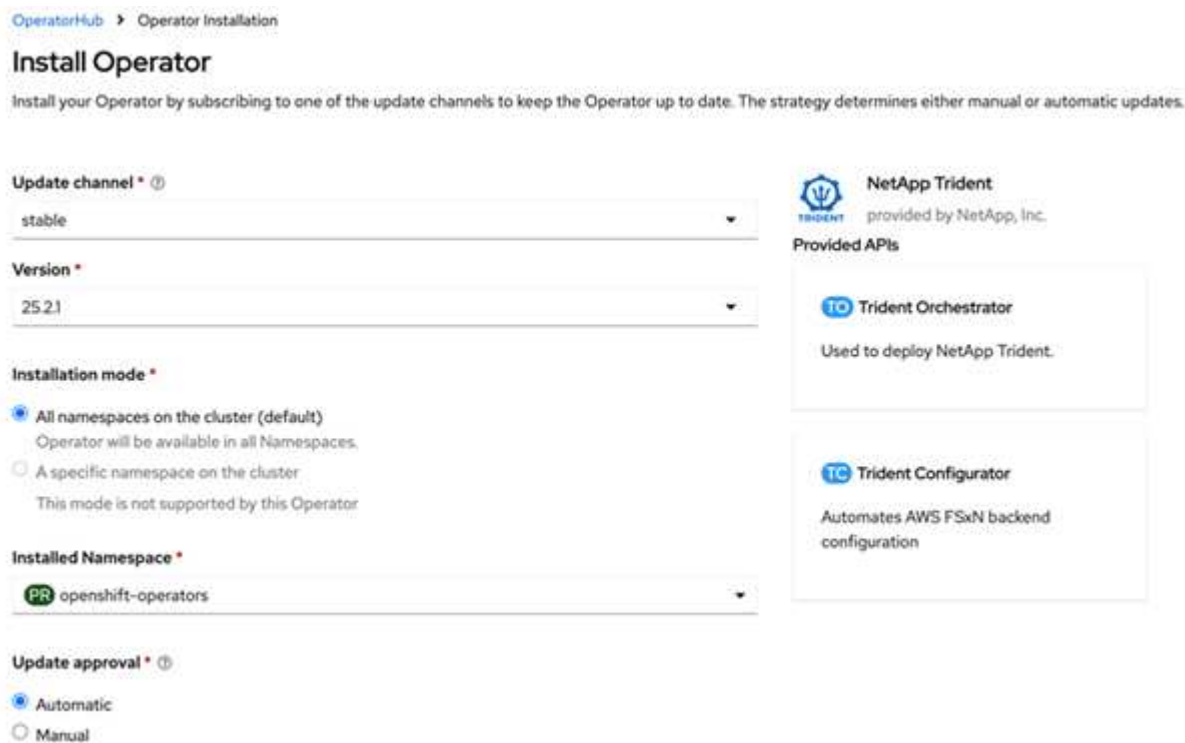


The image shows a window titled "NetApp Trident" with a close button (X) in the top right corner. Below the title bar is a blue "Install" button. The main content area is divided into two sections. On the left, there are three dropdown menus: "Channel" set to "stable", "Version" set to "25.2.1" (with a dropdown menu open showing "25.2.1" selected with a blue checkmark and "25.2.0" below it), and "Source" set to "Certified". On the right, there is a text description: "NetApp Trident is an open source storage provisioner and orchestrator maintained by NetApp. It enables you to create storage volumes for containerized applications managed by Docker and Kubernetes. For full release information, including patch release changes, see <https://docs.netapp.com/us-en/trident/trident-rn.html>."



必ず最新の Operator バージョンを選択してください。

4. すべてのパラメータをそのままにして、**Install** をクリックします。



The image shows the "OperatorHub" page for "Operator Installation". The breadcrumb trail is "OperatorHub > Operator Installation". The main heading is "Install Operator". Below the heading is a subheading: "Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates." The form contains several sections: "Update channel" with a dropdown set to "stable"; "Version" with a dropdown set to "25.2.1"; "Installation mode" with two radio buttons: "All namespaces on the cluster (default)" (selected) and "A specific namespace on the cluster" (unselected); "Installed Namespace" with a dropdown set to "openshift-operators"; and "Update approval" with two radio buttons: "Automatic" (selected) and "Manual" (unselected). On the right side, there is a section titled "NetApp Trident" with a close button (X) in the top right corner. Below the title bar is a blue "Install" button. The main content area is divided into two sections. On the left, there are three dropdown menus: "Channel" set to "stable", "Version" set to "25.2.1" (with a dropdown menu open showing "25.2.1" selected with a blue checkmark and "25.2.0" below it), and "Source" set to "Certified". On the right, there is a text description: "NetApp Trident is an open source storage provisioner and orchestrator maintained by NetApp. It enables you to create storage volumes for containerized applications managed by Docker and Kubernetes. For full release information, including patch release changes, see <https://docs.netapp.com/us-en/trident/trident-rn.html>."

5. *View Operator*をクリックして、Operatorの詳細を表示します。



Provided APIs

TO Trident Orchestrator

Used to deploy NetApp Trident.

[Create instance](#)**TC** Trident Configurator

Automates AWS FSxN backend configuration

[Create instance](#)

6. **YAML view** をクリックし、次の内容をフォームに貼り付けます：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false
```



UIにはデフォルトの例が表示されます。設定全体をコピーする代わりに、直接編集することもできます。

オプション：並行処理を有効にする



同時実行を有効にするには、次のフィールドを spec に追加します：

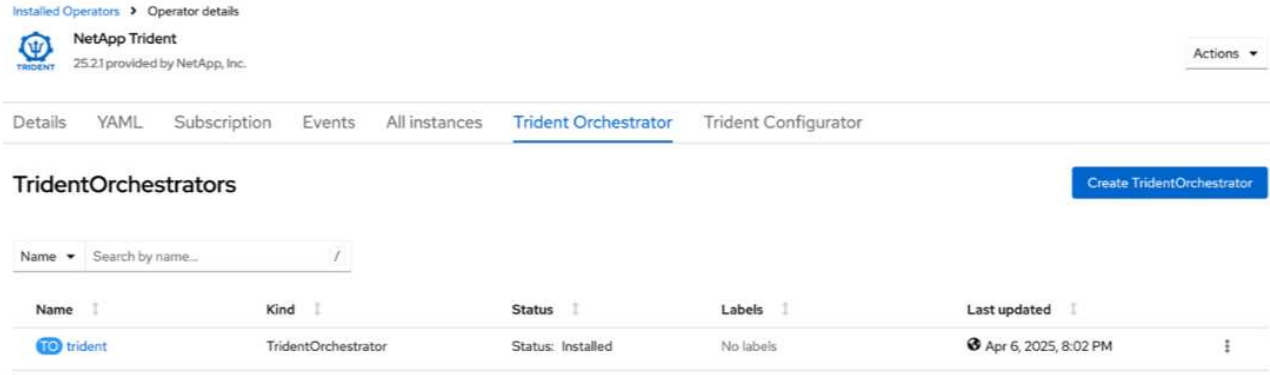
```
enableConcurrency: true
```



- Red Hat Enterprise Linux CoreOS (RHCOS) では iSCSI が有効化および構成されていません。

- `nodePrep`パラメータを追加して、すべてのOpenShiftワーカーノードでiSCSIおよびマルチパスサービスを設定および有効にすることができます。
- OpenShift 4.19 以降、この機能でサポートされる Trident の最小バージョンは 25.06.1 です。

1. *作成*をクリックします。Trident Orchestrator が完全にインストールされます。



Tridentオペレーターをアンインストールします

手順

1. インストールされているオペレータのリストから Trident オペレータを選択します。
2. 演算子からすべてのオペランドインスタンスを削除する場合に選択します。



*この演算子からすべてのオペランドインスタンスを削除する*チェックボックスを選択しない場合、Tridentはアンインストールされません。

3. **Uninstall** をクリックします。

OpenShift認定Tridentオペレーターに切り替える

コミュニティオペレーター、Helmベースのインストール、または手動でデプロイされたオペレーターから、Red Hat OpenShift認定Tridentオペレーターに切り替えることができます。各方法のプロセスでは、既存のオペレーターをアンインストールしてから、OperatorHubを使用して認定オペレーターをインストールします。

開始する前に

インストールを始める前に、"[Tridentインストールのための環境を準備する](#)"。

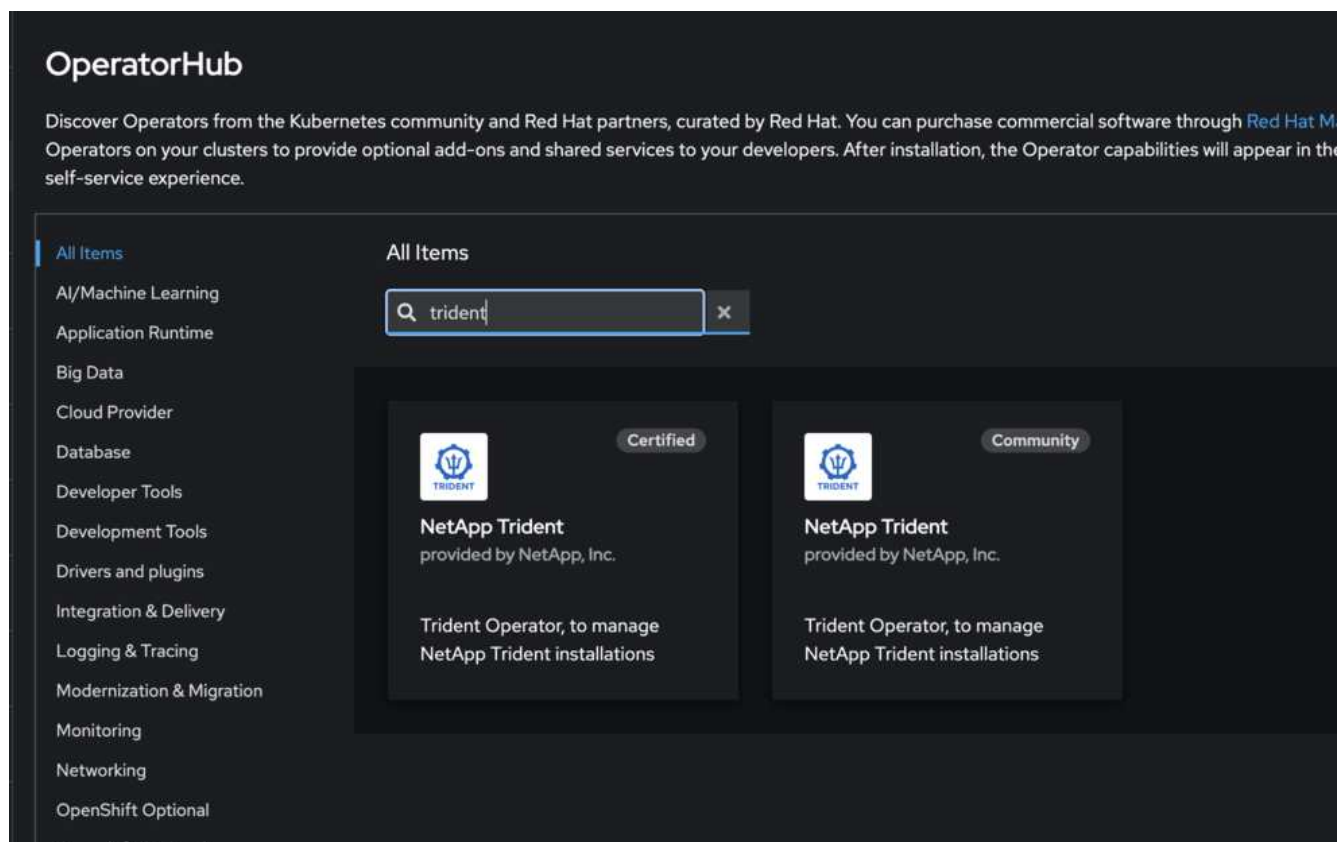


アンインストールプロセス中に `TridentOrchestrator` カスタムリソース (CR) を削除しないでください。`TridentOrchestrator` CRは、認定オペレーターをインストールした後に必要なバックエンドとストレージクラスの設定を保持します。

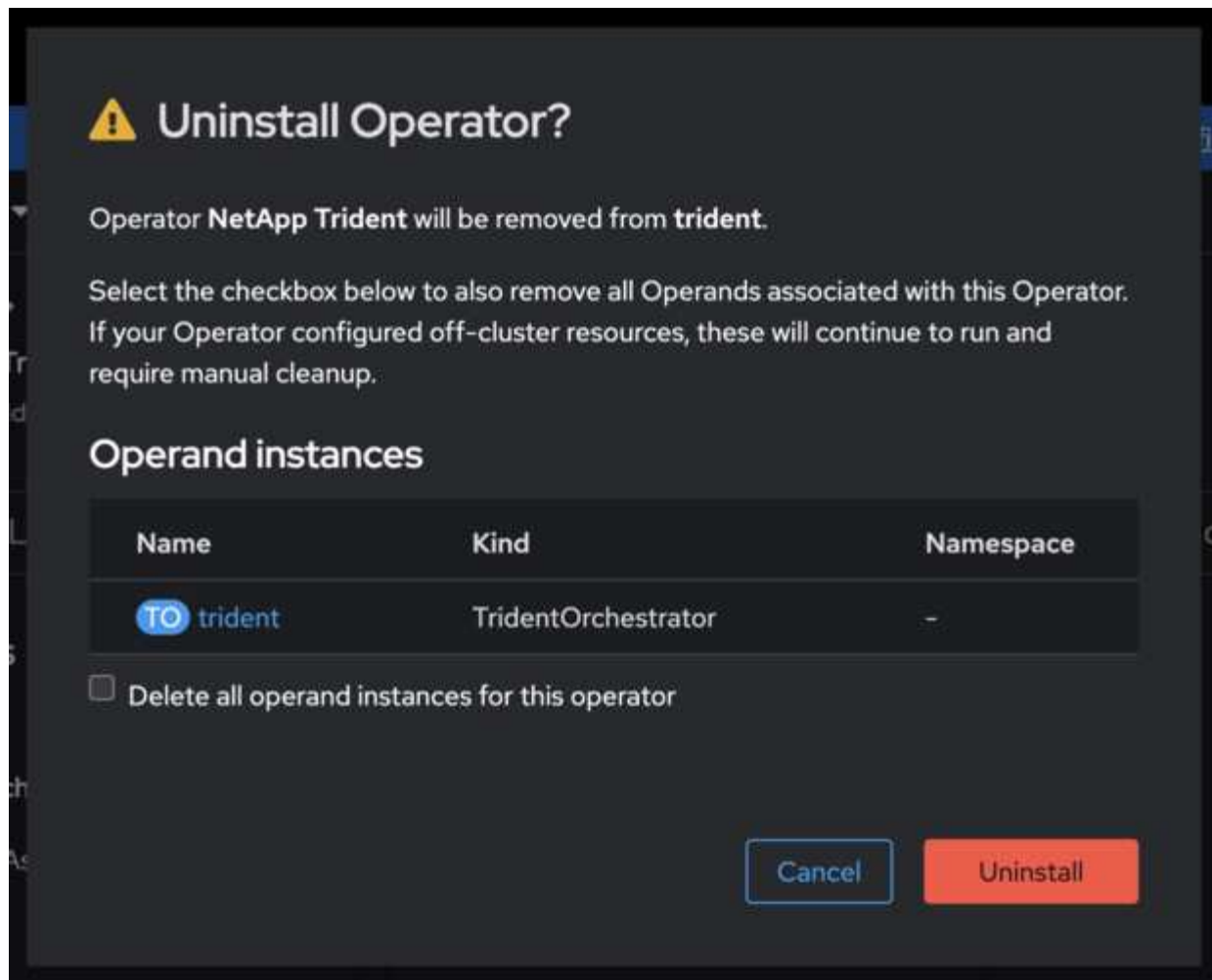
コミュニティオペレーターから切り替える

手順

1. OpenShiftコンソールを使用してOperatorHubに移動します。



2. NetApp Tridentコミュニティオペレーターを見つけます。



*この演算子からすべてのオペランドインスタンスを削除する*を選択しないでください。

3. **Uninstall** をクリックします。
4. アンインストールが完了したら、[OpenShift 認定オペレーターをインストールする](#)に進みます。

Helmベースのoperatorインストールから切り替える

手順

1. Trident インストールの Helm リリースを一覧表示します：

```
helm ls -n trident
```

2. Helm リリースをアンインストールします：

```
helm uninstall <release-name> -n trident
```

3. アンインストールが完了したら、[OpenShift 認定オペレーターをインストールする](#)に進みます。

手動で展開されたオペレーターから切り替える

インストーラー パッケージの `bundle.yaml` を使用してオペレーターを手動で展開することでTridentをインストールした場合は、同じマニフェストを削除して削除します。

手順

1. バンドル マニフェストを使用してオペレーターのデプロイメントを削除します：

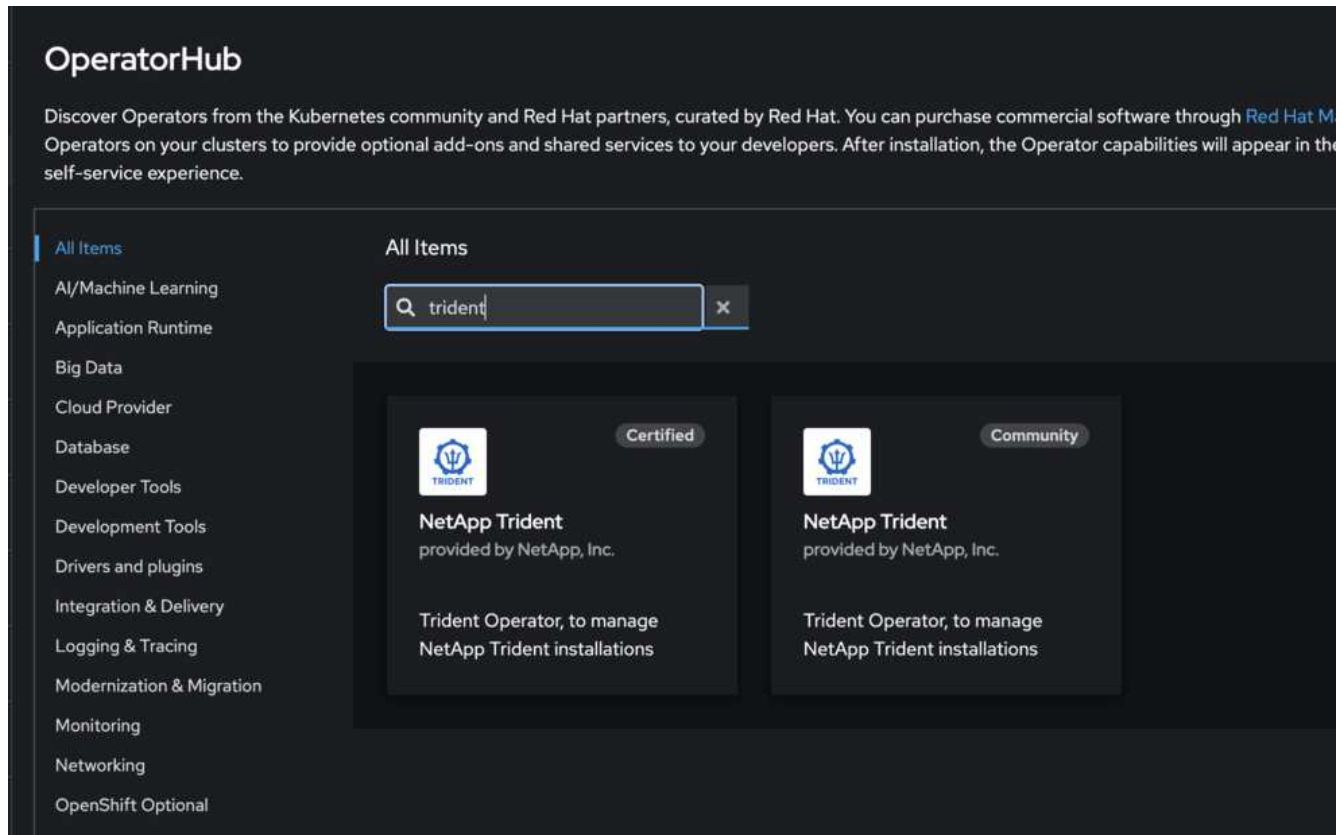
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. アンインストールが完了したら、[OpenShift 認定オペレーターをインストールする](#)に進みます。

OpenShift 認定オペレーターをインストールする

手順

1. Red Hat OperatorHubに移動します。
2. NetApp Trident Operatorを検索して選択します。



3. 画面上の指示に従ってオペレーターをインストールします。

検証

- コンソールのOperatorHubを確認して、新しい認定オペレータが正常にインストールされたことを確認します。

著作権に関する情報

Copyright © 2026 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。