Trident 운영자를 사용하여 설치합니다

Trident

NetApp
July 01, 2026

목차

Trident 운영자를 사용하여 설치합니다

Trident 운영자 수동 배포(표준 모드)

Trident 운영자를 수동으로 배포하여 Trident를 설치할 수 있습니다. 이 프로세스는 Trident에 필요한 컨테이너 이미지가 프라이빗 레지스트리에 저장되지 않은 설치에 적용됩니다. 프라이빗 이미지 레지스트리가 있는 경우 "오프라인 배포 프로세스"을(를) 사용하십시오.

Trident 26.02에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

Trident에 대한 중요 정보

- Kubernetes 1.35는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Trident는 SAN 환경에서 다중 경로 구성 사용을 엄격하게 시행하며, multipath.conf 파일에 권장 값은 `find_multipaths: no`입니다.

  다중 경로를 사용하지 않는 구성 또는 multipath.conf 파일에서 `find_multipaths: yes` 또는 `find_multipaths: smart` 값을 사용하면 마운트가 실패합니다. Trident는 21.07 릴리스부터 `find_multipaths: no` 사용을 권장해 왔습니다.

Trident 운영자를 수동으로 배포하고 Trident를 설치합니다

"설치 개요"을(를) 검토하여 설치 사전 요구 사항을 충족했는지, 그리고 사용 환경에 맞는 올바른 설치 옵션을 선택했는지 확인하십시오.

시작하기 전에

설치를 시작하기 전에 Linux 호스트에 로그인하여 작동 중인 "지원되는 Kubernetes 클러스터"를 관리하고 있는지, 필요한 권한이 있는지 확인합니다.

> OpenShift를 사용할 때, 아래 모든 예제에서 `oc`을(를) `kubectl` 대신 사용하고, 먼저 `oc login -u system:admin` 또는 `oc login -u kube-admin`을(를) 실행하여 *system:admin*으로 로그인하십시오.

1. Kubernetes 버전을 확인합니다.

```
kubectl version
```

2. 클러스터 관리자 권한을 확인하십시오.

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

3. Docker Hub의 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인하십시오.

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

1단계**: Trident** 설치 패키지를 다운로드하세요**.**

Trident 설치 프로그램 패키지에는 Trident 운영자를 배포하고 Trident를 설치하는 데 필요한 모든 것이 포함되어 있습니다. "GitHub의 *Assets* 섹션"에서 최신 버전의 Trident 설치 프로그램을 다운로드하여 압축을 풉니다.

```
wget https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

2단계**:** `TridentOrchestrator` **CRD** 생성

`TridentOrchestrator` Custom Resource Definition(CRD)을 생성합니다. 나중에 `TridentOrchestrator` Custom Resource를 생성합니다. `deploy/crds`에서 적절한 CRD YAML 버전을 사용하여 `TridentOrchestrator` CRD를 생성합니다.

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

3단계**: Trident** 운영자를 배포합니다

Trident 설치 프로그램은 오퍼레이터를 설치하고 관련 객체를 생성하는 데 사용할 수 있는 번들 파일을 제공합니다. 번들 파일은 오퍼레이터를 배포하고 기본 구성을 사용하여 Trident를 설치하는 쉬운 방법입니다.

- Kubernetes 1.24를 실행하는 클러스터의 경우 `bundle_pre_1_25.yaml`를 사용하십시오.

- Kubernetes 1.25 이상 버전을 실행하는 클러스터의 경우 `bundle_post_1_25.yaml`를 사용하십시오.

시작하기 전에

- 기본적으로 Trident 설치 프로그램은 `trident` 네임스페이스에 오퍼레이터를 배포합니다. `trident` 네임스페이스가 존재하지 않으면 다음 명령을 사용하여 생성하십시오.

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

- `trident` 네임스페이스가 아닌 다른 네임스페이스에 오퍼레이터를 배포하려면 `serviceaccount.yaml`, `clusterrolebinding.yaml` 및 `operator.yaml`를 업데이트하고 `kustomization.yaml`를 사용하여 번들 파일을 생성하십시오.

  a. `kustomization.yaml`을 다음 명령을 사용하여 생성합니다. 여기서 _<bundle.yaml>_는 Kubernetes 버전에 따라 `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml`입니다.

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

  b. 다음 명령어를 사용하여 번들을 컴파일하십시오. 여기서 _<bundle.yaml>_는 `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` 사용 중인 Kubernetes 버전에 따라 다릅니다.

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

단계

1. 리소스를 생성하고 운영자를 배포합니다:

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

2. operator, deployment 및 replicaset이 생성되었는지 확인하십시오.

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

Kubernetes 클러스터에는 오퍼레이터 인스턴스가 하나만 존재해야 합니다. Trident 오퍼레이터 배포를 여러 개 생성하지 마십시오.

4단계: `TridentOrchestrator` 생성 및 **Trident** 설치

이제 `TridentOrchestrator`를 생성하고 `Trident`를 설치할 수 있습니다. 선택적으로, "Trident 설치를 사용자 지정합니다"를 `TridentOrchestrator` 사양의 속성을 사용하여 수행할 수 있습니다.

```
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:26.02
    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:26.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v26.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

설치 확인

설치를 확인하는 방법에는 여러 가지가 있습니다.

Using `TridentOrchestrator` 상태 사용

`TridentOrchestrator`의 상태는 설치가 성공했는지 여부를 나타내며 설치된 `Trident` 버전을 표시합니다. 설치 중 `TridentOrchestrator`의 상태는 `Installing`에서 `Installed`로 변경됩니다. `Failed` 상태를 관찰하고 운영자가 자체적으로 복구할 수 없는 경우link:../troubleshooting.html["로그를 확인합니다"].

상태	설명
설치	운영자는 이 `TridentOrchestrator` CR을 사용하여 Trident를 설치하고 있습니다.
설치됨	Trident가 성공적으로 설치되었습니다.
제거하기	운영자가 Trident을 제거 중입니다 `spec.uninstall=true`.
제거됨	Trident가 제거되었습니다.
실패	운영자가 Trident를 설치, 패치, 업데이트 또는 제거할 수 없습니다. 운영자가 자동으로 이 상태에서 복구를 시도합니다. 이 상태가 지속되면 문제 해결이 필요합니다.
업데이트 중	운영자가 기존 설치를 업데이트하고 있습니다.
오류	`TridentOrchestrator`은(는) 사용되지 않습니다. 다른 하나가 이미 존재합니다.

POD 생성 상태 사용

생성된 Pod의 상태를 검토하여 Trident 설치가 완료되었는지 확인할 수 있습니다.

```
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
```

사용 `tridentctl`

`tridentctl` 을 사용하여 설치된 `Trident` 버전을 확인할 수 있습니다.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Trident 운영자를 수동으로 배포합니다(오프라인 모드)

Trident 운영자를 수동으로 배포하여 Trident를 설치할 수 있습니다. 이 프로세스는 Trident에 필요한 컨테이너 이미지가 프라이빗 레지스트리에 저장된 설치에 적용됩니다. 프라이빗 이미지 레지스트리가 없는 경우 "표준 배포 프로세스"을 사용하십시오.

Trident에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

Trident에 대한 중요 정보

- Kubernetes 1.35는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Trident는 SAN 환경에서 다중 경로 구성 사용을 엄격하게 시행하며, multipath.conf 파일에 권장 값은 `find_multipaths: no`입니다.

  다중 경로를 사용하지 않는 구성 또는 multipath.conf 파일에서 `find_multipaths: yes` 또는 `find_multipaths: smart` 값을 사용하면 마운트가 실패합니다. Trident는 21.07 릴리스부터 `find_multipaths: no` 사용을 권장해 왔습니다.

Trident 운영자를 수동으로 배포하고 Trident를 설치합니다

"설치 개요"을(를) 검토하여 설치 사전 요구 사항을 충족했는지, 그리고 사용 환경에 맞는 올바른 설치 옵션을 선택했는지 확인하십시오.

시작하기 전에

Linux 호스트에 로그인하여 정상적으로 작동하는 "지원되는 Kubernetes 클러스터"를 관리하고 있는지, 그리고 필요한 권한이 있는지 확인하십시오.

OpenShift를 사용할 때, 아래 모든 예제에서 `oc`을(를) `kubectl` 대신 사용하고, 먼저 `oc login -u system:admin` 또는 `oc login -u kube-admin`을(를) 실행하여 *system:admin*으로 로그인하십시오.

1. Kubernetes 버전을 확인합니다.

```
kubectl version
```

2. 클러스터 관리자 권한을 확인하십시오.

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

3. Docker Hub의 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인하십시오.

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

1단계: **Trident** 설치 패키지를 다운로드하세요.

Trident 설치 프로그램 패키지에는 Trident 운영자를 배포하고 Trident를 설치하는 데 필요한 모든 것이 포함되어 있습니다. "GitHub의 *Assets* 섹션"에서 최신 버전의 Trident 설치 프로그램을 다운로드하여 압축을 풉니다.

```
wget https://github.com/NetApp/trident/releases/download/v6.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

2단계: `TridentOrchestrator` **CRD** 생성

`TridentOrchestrator` Custom Resource Definition(CRD)을 생성합니다. 나중에 `TridentOrchestrator` Custom Resources를 생성하게 됩니다. `deploy/crds`에서 적절한 CRD YAML 버전을 사용하여 `TridentOrchestrator` CRD를 생성하십시오.

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

3단계: 운영자에서 레지스트리 위치 업데이트

`/deploy/operator.yaml`에서 `image: docker.io/netapp/trident-operator:26.02.0`를 업데이트하여 이미지 레지스트리의 위치를 반영하십시오. xref:{relative_path}../trident-get-started/requirements.html#container-images-and-corresponding-kubernetes-versions["Trident 및 CSI 이미지"]는 하나의 레지스트리 또는 여러 레지스트리에 위치할 수 있지만, 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다. 예를 들면 다음과 같습니다.

- `image: <your-registry>/trident-operator:26.02.0` 이미지가 모두 하나의 레지스트리에 있는 경우.
- `image: <your-registry>/netapp/trident-operator:26.02.0` Trident 이미지가 CSI 이미지와 다른 레지스트리에 있는 경우.

4단계: **Trident** 운영자를 배포합니다

Trident 설치 프로그램은 오퍼레이터를 설치하고 관련 객체를 생성하는 데 사용할 수 있는 번들 파일을 제공합니다. 번들 파일은 오퍼레이터를 배포하고 기본 구성을 사용하여 Trident를 설치하는 쉬운 방법입니다.

- Kubernetes 1.24를 실행하는 클러스터의 경우 `bundle_pre_1_25.yaml`를 사용하십시오.
- Kubernetes 1.25 이상 버전을 실행하는 클러스터의 경우 `bundle_post_1_25.yaml`를 사용하십시오.

시작하기 전에

- 기본적으로 Trident 설치 프로그램은 `trident` 네임스페이스에 오퍼레이터를 배포합니다. `trident` 네임스페이스가 존재하지 않으면 다음 명령을 사용하여 생성하십시오.

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

- `trident` 네임스페이스가 아닌 다른 네임스페이스에 오퍼레이터를 배포하려면 `serviceaccount.yaml`, `clusterrolebinding.yaml` 및 `operator.yaml`를 업데이트하고 `kustomization.yaml`를 사용하여 번들 파일을 생성하십시오.

  a. `kustomization.yaml`을 다음 명령을 사용하여 생성합니다. 여기서 _<bundle.yaml>_는 Kubernetes 버전에 따라 `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml`입니다.

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

  b. 다음 명령어를 사용하여 번들을 컴파일하십시오. 여기서 _<bundle.yaml>_는 `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` 사용 중인 Kubernetes 버전에 따라 다릅니다.

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

단계

1. 리소스를 생성하고 운영자를 배포합니다:

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

2. operator, deployment 및 replicaset이 생성되었는지 확인하십시오.

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

Kubernetes 클러스터에는 오퍼레이터 인스턴스가 하나만 존재해야 합니다. Trident 오퍼레이터 배포를 여러 개 생성하지 마십시오.

5단계: **`TridentOrchestrator`**에서 이미지 레지스트리 위치 업데이트

"Trident 및 CSI 이미지"는 하나의 레지스트리 또는 여러 레지스트리에 위치할 수 있지만, 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다. 레지스트리 구성에 따라 추가 위치 사양을 추가하려면 `deploy/crds/tridentorchestrator_cr.yaml`를 업데이트하십시오.

하나의 레지스트리에 있는 이미지

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

서로 다른 레지스트리에 있는 이미지

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

6단계: `TridentOrchestrator` 생성 및 **Trident** 설치

이제 `TridentOrchestrator`를 생성하고 `Trident`를 설치할 수 있습니다. 선택적으로, "Trident 설치를 사용자 지정합니다"를 `TridentOrchestrator` 사양의 속성을 사용하여 추가로 구성할 수 있습니다. 다음 예시는 Trident와 CSI 이미지가 서로 다른 레지스트리에 위치한 설치를 보여줍니다.

```
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:26.02
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.02
    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:26.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

설치 확인

설치를 확인하는 방법에는 여러 가지가 있습니다.

Using `TridentOrchestrator` 상태 사용

`TridentOrchestrator`의 상태는 설치가 성공했는지 여부를 나타내며 설치된 `Trident` 버전을 표시합니다. 설치 중 `TridentOrchestrator`의 상태는 `Installing`에서 `Installed`로 변경됩니다. `Failed` 상태를 관찰하고 운영자가 자체적으로 복구할 수 없는 경우link:../troubleshooting.html["로그를 확인합니다"].

상태	설명
설치	운영자는 이 `TridentOrchestrator` CR을 사용하여 Trident를 설치하고 있습니다.
설치됨	Trident가 성공적으로 설치되었습니다.
제거하기	운영자가 Trident을 제거 중입니다 `spec.uninstall=true`.
제거됨	Trident가 제거되었습니다.
실패	운영자가 Trident를 설치, 패치, 업데이트 또는 제거할 수 없습니다. 운영자가 자동으로 이 상태에서 복구를 시도합니다. 이 상태가 지속되면 문제 해결이 필요합니다.
업데이트 중	운영자가 기존 설치를 업데이트하고 있습니다.
오류	`TridentOrchestrator`은(는) 사용되지 않습니다. 다른 하나가 이미 존재합니다.

POD 생성 상태 사용

생성된 Pod의 상태를 검토하여 Trident 설치가 완료되었는지 확인할 수 있습니다.

```
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
```

사용 `tridentctl`

`tridentctl` 을 사용하여 설치된 `Trident` 버전을 확인할 수 있습니다.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Helm을 사용하여 Trident 운영자 배포(표준 모드)

Helm을 사용하여 Trident 오퍼레이터를 배포하고 Trident를 설치할 수 있습니다. 이 프로세스는 Trident에 필요한 컨테이너 이미지가 프라이빗 레지스트리에 저장되지 않은 설치에 적용됩니다. 프라이빗 이미지 레지스트리가 있는 경우 "오프라인 배포 프로세스"을(를) 사용하십시오.

Trident 25.10에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

Trident에 대한 중요 정보

- Kubernetes 1.35는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Trident는 SAN 환경에서 다중 경로 구성 사용을 엄격하게 시행하며, multipath.conf 파일에 권장 값은 `find_multipaths: no`입니다.

  다중 경로를 사용하지 않는 구성 또는 multipath.conf 파일에서 `find_multipaths: yes` 또는 `find_multipaths: smart` 값을 사용하면 마운트가 실패합니다. Trident는 21.07 릴리스부터 `find_multipaths: no` 사용을 권장해 왔습니다.

Helm을 사용하여 Trident 오퍼레이터를 배포하고 Trident를 설치하십시오

Trident "Helm Chart"를 사용하여 Trident 운영자를 배포하고 Trident를 한 번에 설치할 수 있습니다.

"설치 개요"을(를) 검토하여 설치 사전 요구 사항을 충족했는지, 그리고 사용 환경에 맞는 올바른 설치 옵션을 선택했는지 확인하십시오.

시작하기 전에

"배포 사전 요구 사항" 외에도 "Helm 버전 3"이(가) 필요합니다.

단계

1. Trident Helm 리포지토리 추가:

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

2. `helm install`를 사용하고 다음 예시와 같이 배포 이름을 지정하십시오. 여기서 `100..0`는 설치하려는 Trident 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace <trident-namespace>
```

> 이미 Trident에 대한 네임스페이스를 생성한 경우 `--create-namespace` 매개변수는 추가 네임스페이스를 생성하지 않습니다.

`helm list`을 사용하여 이름, 네임스페이스, 차트, 상태, 앱 버전 및 수정 번호와 같은 설치 세부 정보를 검토할 수 있습니다.

설치 중에 구성 데이터를 전달합니다

설치 중에 구성 데이터를 전달하는 방법에는 두 가지가 있습니다.

옵션	설명
`--values` (또는 `-f`)	재정의를 포함하는 YAML 파일을 지정합니다. 이 파일은 여러 번 지정할 수 있으며 가장 오른쪽에 있는 파일이 우선 적용됩니다.
`--set`	명령줄에서 재정의를 지정합니다.

예를 들어, `debug`의 기본값을 변경하려면 다음 명령을 실행하십시오. 여기서 `100.2602.0`는 설치하려는 Trident 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

구성 옵션

이 표와 Helm 차트의 일부인 `values.yaml` 파일은 키 목록과 해당 기본값을 제공합니다.

옵션	설명	기본값
`nodeSelector`	POD 할당을 위한 노드 레이블	
`podAnnotations`	Pod 주석	
`deploymentAnnotations`	배포 주석	
`tolerations`	POD 할당에 대한 톨러레이션	

옵션	설명	기본값

`affinity`	POD 할당에 대한 선호도	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		values.yaml 파일에서 기본 선호도를 제거하지 마십시오. 사용자 지정 선호도를 제공하려면 기본 선호도를 확장하십시오.

`tridentControllerPluginNodeSelector`	Pod에 대한 추가 노드 선택기입니다. 자세한 내용은 컨트롤러 Pod 및 노드 Pod 이해를 참조하십시오.	
`tridentControllerPluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 자세한 내용은 컨트롤러 Pod 및 노드 Pod 이해를 참조하십시오.	
`tridentNodePluginNodeSelector`	Pod에 대한 추가 노드 선택기입니다. 자세한 내용은 컨트롤러 Pod 및 노드 Pod 이해를 참조하십시오.	
`tridentNodePluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 자세한 내용은 컨트롤러 Pod 및 노드 Pod 이해를 참조하십시오.	

옵션	설명	기본값
imageRegistry	trident-operator, trident 및 기타 이미지에 대한 레지스트리를 식별합니다. 기본값을 사용하려면 비워 두십시오. 중요: 개인 리포지토리에 Trident를 설치할 때 리포지토리 위치를 지정하기 위해 imageRegistry 스위치를 사용하는 경우 리포지토리 경로에 `/netapp/`를 사용하지 마십시오.	""
imagePullPolicy	`trident-operator`에 대한 이미지 풀 정책을 설정합니다.	IfNotPresent
imagePullSecrets	trident-operator, trident 및 기타 이미지에 대한 이미지 가져오기 비밀 키를 설정합니다.	
kubeletDir	kubelet의 내부 상태의 호스트 위치를 재정의할 수 있습니다.	"/var/lib/kubelet"
operatorLogLevel	Trident 운영자의 로그 레벨을 trace, debug, info, warn, error 또는 `fatal`로 설정할 수 있습니다.	"info"
operatorDebug	Trident 운영자의 로그 수준을 디버그로 설정할 수 있습니다.	true
operatorImage	`trident-operator`에 대한 이미지의 완전한 재정의를 허용합니다.	""
operatorImageTag	trident-operator 이미지의 태그를 재정의할 수 있습니다.	""
tridentIPv6	Trident가 IPv6 클러스터에서 작동하도록 설정할 수 있습니다.	false
tridentK8sTimeout	대부분의 Kubernetes API 작업에 대한 기본 30초 시간 초과를 재정의합니다(0이 아닌 경우 초 단위).	0
tridentHttpRequestTimeout	HTTP 요청에 대한 기본 90초 타임아웃을 재정의하며, `0s`는 타임아웃의 무한 기간입니다. 음수 값은 허용되지 않습니다.	"90s"
tridentSilenceAutosupport	Trident 주기적인 AutoSupport 보고 기능을 비활성화할 수 있습니다.	false
tridentAutosupportImageTag	Trident AutoSupport 컨테이너의 이미지 태그를 재정의할 수 있습니다.	<version>
tridentAutosupportProxy	Trident AutoSupport 컨테이너가 HTTP 프록시를 통해 홈에 전화를 걸 수 있도록 합니다.	""

옵션	설명	기본값
`tridentLogFormat`	Trident 로깅 형식을 설정합니다 ((`text` 또는 `json`).	`"text"`
`tridentDisableAuditLog`	Trident 감사 로거를 비활성화합니다.	`true`
`tridentLogLevel`	Trident의 로그 레벨을 `trace`, `debug`, `info`, `warn`, `error` 또는 `fatal`로 설정할 수 있습니다.	`"info"`
`tridentDebug`	Trident의 로그 레벨을 `debug`로 설정할 수 있습니다. 노드 상태 점검(NHC) 연산자와 통합하여 강제 분리 프로세스를 자동화할 수 있습니다. 자세한 내용은 "Trident를 사용하여 상태 저장 애플리케이션의 페일오버 자동화"를 참조하십시오.	`false`
`tridentLogWorkflows`	특정 Trident 워크플로우에 대해 추적 로깅 또는 로그 억제를 활성화할 수 있습니다.	`""`
`tridentLogLayers`	특정 Trident 레이어에 대해 추적 로깅 또는 로그 억제를 활성화할 수 있습니다.	`""`
`tridentImage`	Trident의 이미지를 완전히 재정의할 수 있습니다.	`""`
`tridentImageTag`	Trident의 이미지 태그를 재정의할 수 있습니다.	`""`
`tridentProbePort`	Kubernetes 활성/준비 상태 프로브에 사용되는 기본 포트를 재정의할 수 있습니다.	`""`
`windows`	Windows 워커 노드에 Trident를 설치할 수 있도록 합니다.	`false`
`enableForceDetach`	강제 분리 기능을 활성화할 수 있습니다.	`false`
`excludePodSecurityPolicy`	운영자 포드 보안 정책을 생성에서 제외합니다.	`false`
`cloudProvider`	`"Azure"`AKS 클러스터에서 관리형 ID 또는 클라우드 ID를 사용하는 경우로 설정합니다. EKS 클러스터에서 클라우드 ID를 사용하는 경우 "AWS"로 설정합니다.	`""`

옵션	설명	기본값
`cloudIdentity`	AKS 클러스터에서 클라우드 ID를 사용하는 경우 워크로드 ID("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx")로 설정하십시오. EKS 클러스터에서 클라우드 ID를 사용하는 경우 AWS IAM 역할("eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role")로 설정하십시오.	""
`iscsiSelfHealingInterval`	iSCSI 자체 복구가 호출되는 간격입니다.	`5m0s`
`iscsiSelfHealingWaitTime`	iSCSI 자체 복구 기능이 로그아웃 후 로그인을 수행하여 오래된 세션을 해결하려고 시도하는 기간입니다.	`7m0s`
`nodePrep`	Trident가 지정된 데이터 스토리지 프로토콜을 사용하여 볼륨을 관리하도록 Kubernetes 클러스터의 노드를 준비할 수 있습니다. 현재 `**iscsi**`만 지원됩니다**.** 참고: OpenShift 4.19부터 이 기능을 지원하는 최소 Trident 버전은 25.06.1입니다.	
`enableConcurrency`	처리량 향상을 위해 동시 Trident 컨트롤러 작업을 지원합니다. **Tech Preview**: 이 기능은 실험적이며 현재 ONTAP-NAS(NFS만 해당) 및 ONTAP-SAN(통합 ONTAP 9용 NVMe) 드라이버를 사용한 제한된 병렬 워크플로를 지원하며, ONTAP-SAN 드라이버(통합 ONTAP 9의 iSCSI 및 FCP 프로토콜)에 대한 기존 Tech Preview도 지원합니다.	거짓
`k8sAPIQPS`	컨트롤러가 Kubernetes API 서버와 통신할 때 사용하는 초당 쿼리 수(QPS) 제한입니다. 버스트 값은 QPS 값을 기반으로 자동으로 설정됩니다.	`100`; 선택 사항

옵션	설명	기본값

`resources`	Trident 컨트롤러, 노드 및 오퍼레이터 파드에 대한 Kubernetes 리소스 제한 및 요청을 설정합니다. 각 컨테이너 및 사이드카에 대한 CPU 및 메모리를 구성하여 Kubernetes에서 리소스 할당을 관리할 수 있습니다.
	
	리소스 요청 및 제한 구성에 대한 자세한 내용은 "Pod 및 컨테이너에 대한 리소스 관리"을(를) 참조하십시오.
	
	• 컨테이너 또는 필드의 이름을 변경하지 마십시오.
	• 들여쓰기를 변경하지 마십시오. YAML 들여쓰기는 올바른 구문 분석에 매우 중요합니다.
	
	• 기본적으로 제한은 적용되지 않으며 요청에만 기본값이 있고 지정하지 않으면 자동으로 적용됩니다.
	• 컨테이너 이름은 Pod 사양에 나타난 대로 나열됩니다.
	• 사이드카는 각 메인 컨테이너 아래에 나열됩니다.
	• TORC `status.CurrentInstallationParams` 필드를 확인하여 현재 적용된 값을 확인하십시오.
		<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:
		    linux:
		      trident-main:</pre>

옵션	설명	기본값
`httpsMetrics`	Prometheus 메트릭 엔드포인트에 대해 HTTPS를 활성화합니다.	거짓
`hostNetwork`	Trident 컨트롤러에 호스트 네트워킹을 활성화합니다. 이는 멀티홈 네트워크에서 프런트엔드와 백엔드 트래픽을 분리하려는 경우 유용합니다.	거짓

컨트롤러 **Pod** 및 노드 **Pod** 이해

Trident는 단일 컨트롤러 Pod와 클러스터의 각 작업자 노드에 있는 노드 Pod로 실행됩니다. 노드 Pod는 Trident 볼륨을 마운트하려는 모든 호스트에서 실행되어야 합니다.

Kubernetes "노드 선택기"와 "톨러레이션 및 테인트"는 특정 또는 선호하는 노드에서 실행되도록 Pod를 제약하는 데 사용됩니다. `ControllerPlugin`와 `NodePlugin`를 사용하여 제약 및 오버라이드를 지정할 수 있습니다.

- 컨트롤러 플러그인은 스냅샷 및 크기 조정과 같은 볼륨 프로비저닝 및 관리를 처리합니다.
- 노드 플러그인은 스토리지를 노드에 연결하는 작업을 처리합니다.

```
      cpu: 1m
      memory: 10Mi
    limits:
      cpu:
      memory:
  windows:
  trident-main:
    requests:
      cpu: 6m
      memory: 40Mi
    limits:
      cpu:
      memory:
  autosupport-proxy:
    requests:
      cpu: 6m
      memory: 40Mi
    limits:
      cpu:
      memory:
  liveness-probe:
    requests:
      memory:
```

Helm을 사용하여 Trident 운영자 배포(오프라인 모드)

Helm을 사용하여 Trident 오퍼레이터를 배포하고 Trident를 설치할 수 있습니다. 이 프로세스는 Trident에 필요한 컨테이너 이미지가 프라이빗 레지스트리에 저장된 설치에 적용됩니다. 프라이빗 이미지 레지스트리가 없는 경우 "표준 배포 프로세스"을 사용하십시오.

Trident 26.02에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

Trident에 대한 중요 정보

- Kubernetes 1.35는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Trident는 SAN 환경에서 다중 경로 구성 사용을 엄격하게 시행하며, multipath.conf 파일에 권장 값은 `find_multipaths: no`입니다.

  다중 경로를 사용하지 않는 구성 또는 multipath.conf 파일에서 `find_multipaths: yes` 또는 `find_multipaths: smart` 값을 사용하면 마운트가 실패합니다. Trident는 21.07 릴리스부터 `find_multipaths: no` 사용을 권장해 왔습니다.

Helm을 사용하여 Trident 오퍼레이터를 배포하고 Trident를 설치하십시오

Trident "Helm Chart"를 사용하여 Trident 운영자를 배포하고 Trident를 한 번에 설치할 수 있습니다.

"설치 개요"을(를) 검토하여 설치 사전 요구 사항을 충족했는지, 그리고 사용 환경에 맞는 올바른 설치 옵션을 선택했는지 확인하십시오.

시작하기 전에

"배포 사전 요구 사항" 외에도 "Helm 버전 3"이(가) 필요합니다.

> 비공개 리포지토리에 Trident를 설치할 때 `imageRegistry` 스위치를 사용하여 리포지토리 위치를 지정하는 경우 리포지토리 경로에 `/netapp/`를 사용하지 마십시오.

단계

1. Trident Helm 리포지토리 추가:

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

2. `helm install`를 사용하여 배포 및 이미지 레지스트리 위치에 이름을 지정하십시오. "Trident 및 CSI 이미지"는 하나의 레지스트리 또는 여러 레지스트리에 위치할 수 있지만, 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다. 예시에서 `100.2602.0`는 설치하려는 Trident 버전입니다.

하나의 레지스트리에 있는 이미지

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

서로 다른 레지스트리에 있는 이미지

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.02.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.02
--set tridentImage=<your-registry>/trident:26.02.0 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

> 이미 Trident에 대한 네임스페이스를 생성한 경우 `--create-namespace` 매개변수는 추가 네임스페이스를 생성하지 않습니다.

`helm list`을 사용하여 이름, 네임스페이스, 차트, 상태, 앱 버전 및 수정 번호와 같은 설치 세부 정보를 검토할 수 있습니다.

설치 중에 구성 데이터를 전달합니다

설치 중에 구성 데이터를 전달하는 방법에는 두 가지가 있습니다.

옵션	설명
`--values` (또는 `-f`)	재정의를 포함하는 YAML 파일을 지정합니다. 이 파일은 여러 번 지정할 수 있으며 가장 오른쪽에 있는 파일이 우선 적용됩니다.
`--set`	명령줄에서 재정의를 지정합니다.

예를 들어, `debug`의 기본값을 변경하려면 다음 명령을 실행하십시오. 여기서 `100.2602.0`는 설치하려는 Trident 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

nodePrep 값을 추가하려면 다음 명령을 실행하십시오.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

구성 옵션

이 표와 Helm 차트의 일부인 `values.yaml` 파일은 키 목록과 해당 기본값을 제공합니다.

values.yaml 파일에서 기본 선호도를 제거하지 마십시오. 사용자 지정 선호도를 제공하려면 기본 선호도를 확장하십시오.

옵션	설명	기본값
`nodeSelector`	POD 할당을 위한 노드 레이블	
`podAnnotations`	Pod 주석	
`deploymentAnnotations`	배포 주석	
`tolerations`	POD 할당에 대한 톨러레이션	

옵션	설명	기본값

`affinity`	POD 할당에 대한 선호도	`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 파일에서 기본 선호도를 제거하지 마십시오. 사용자 지정 선호도를 제공하려면 기본 선호도를 확장하십시오.

`tridentControllerPluginNodeSelector`	Pod에 대한 추가 노드 선택기입니다. 자세한 내용은 "컨트롤러 Pod 및 노드 Pod 이해"를 참조하십시오.	
`tridentControllerPluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 자세한 내용은 "컨트롤러 Pod 및 노드 Pod 이해"를 참조하십시오.	
`tridentNodePluginNodeSelector`	Pod에 대한 추가 노드 선택기입니다. 자세한 내용은 "컨트롤러 Pod 및 노드 Pod 이해"를 참조하십시오.	

옵션	설명	기본값
`tridentNodePluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 자세한 내용은 "컨트롤러 Pod 및 노드 Pod 이해"를 참조하십시오.	
`imageRegistry`	`trident-operator`, `trident` 및 기타 이미지에 대한 레지스트리를 식별합니다. 기본값을 사용하려면 비워 두십시오. 중요: 개인 리포지토리에 Trident를 설치할 때 리포지토리 위치를 지정하기 위해 `imageRegistry` 스위치를 사용하는 경우 리포지토리 경로에 `/netapp/`를 사용하지 마십시오.	""
`imagePullPolicy`	`trident-operator`에 대한 이미지 풀 정책을 설정합니다.	`IfNotPresent`
`imagePullSecrets`	`trident-operator`, `trident` 및 기타 이미지에 대한 이미지 가져오기 비밀 키를 설정합니다.	
`kubeletDir`	kubelet의 내부 상태의 호스트 위치를 재정의할 수 있습니다.	`"/var/lib/kubelet"`
`operatorLogLevel`	Trident 운영자의 로그 레벨을 `trace`, `debug`, `info`, `warn`, `error` 또는 `fatal`로 설정할 수 있습니다.	`"info"`
`operatorDebug`	Trident 운영자의 로그 수준을 디버그로 설정할 수 있습니다.	`true`
`operatorImage`	`trident-operator`에 대한 이미지의 완전한 재정의를 허용합니다.	""
`operatorImageTag`	`trident-operator` 이미지의 태그를 재정의할 수 있습니다.	""
`tridentIPv6`	Trident가 IPv6 클러스터에서 작동하도록 설정할 수 있습니다.	`false`
`tridentK8sTimeout`	대부분의 Kubernetes API 작업에 대한 기본 180초 시간 초과를 재정의합니다(0이 아닌 경우 초 단위). `tridentK8sTimeout` 매개변수는 Trident 설치에만 적용됩니다.	`180`
`tridentHttpRequestTimeout`	HTTP 요청에 대한 기본 90초 타임아웃을 재정의하며, `0s`는 타임아웃의 무한 기간입니다. 음수 값은 허용되지 않습니다.	`"90s"`

옵션	설명	기본값
`tridentSilenceAutosupport`	Trident 주기적인 AutoSupport 보고 기능을 비활성화할 수 있습니다.	`false`
`tridentAutosupportImageTag`	Trident AutoSupport 컨테이너의 이미지 태그를 재정의할 수 있습니다.	`<version>`
`tridentAutosupportProxy`	Trident AutoSupport 컨테이너가 HTTP 프록시를 통해 홈에 전화를 걸 수 있도록 합니다.	""
`tridentLogFormat`	Trident 로깅 형식을 설정합니다 ((`text` 또는 `json`).	`"text"`
`tridentDisableAuditLog`	Trident 감사 로거를 비활성화합니다.	`true`
`tridentLogLevel`	Trident의 로그 레벨을 `trace`, `debug`, `info`, `warn`, `error` 또는 `fatal`로 설정할 수 있습니다.	`"info"`
`tridentDebug`	Trident의 로그 레벨을 `debug`로 설정할 수 있습니다.	`false`
`tridentLogWorkflows`	특정 Trident 워크플로우에 대해 추적 로깅 또는 로그 억제를 활성화할 수 있습니다.	""
`tridentLogLayers`	특정 Trident 레이어에 대해 추적 로깅 또는 로그 억제를 활성화할 수 있습니다.	""
`tridentImage`	Trident의 이미지를 완전히 재정의할 수 있습니다.	""
`tridentImageTag`	Trident의 이미지 태그를 재정의할 수 있습니다.	""
`tridentProbePort`	Kubernetes 활성/준비 상태 프로브에 사용되는 기본 포트를 재정의할 수 있습니다.	""
`windows`	Windows 워커 노드에 Trident를 설치할 수 있도록 합니다.	`false`
`enableForceDetach`	강제 분리 기능을 활성화할 수 있습니다. 노드 상태 점검(NHC) 연산자와 통합하여 강제 분리 프로세스를 자동화할 수 있습니다. 자세한 내용은 "Trident를 사용하여 상태 저장 애플리케이션의 페일오버 자동화"를 참조하십시오.	`false`
`excludePodSecurityPolicy`	운영자 포드 보안 정책을 생성에서 제외합니다.	`false`

옵션	설명	기본값
`nodePrep`	Trident가 지정된 데이터 스토리지 프로토콜을 사용하여 볼륨을 관리하도록 Kubernetes 클러스터의 노드를 준비할 수 있습니다. 현재 **`iscsi`**만 지원됩니다**.** OpenShift 4.19부터 이 기능을 지원하는 최소 Trident 버전은 25.06.1입니다.	

옵션	설명	기본값

`resources`	Trident 컨트롤러, 노드 및 오퍼레이터 파드에 대한 Kubernetes 리소스 제한 및 요청을 설정합니다. 각 컨테이너 및 사이드카에 대한 CPU 및 메모리를 구성하여 Kubernetes에서 리소스 할당을 관리할 수 있습니다.
	
	리소스 요청 및 제한 구성에 대한 자세한 내용은 "Pod 및 컨테이너에 대한 리소스 관리"을(를) 참조하십시오.
	
	• 컨테이너 또는 필드의 이름을 변경하지 마십시오.
	• 들여쓰기를 변경하지 마십시오. YAML 들여쓰기는 올바른 구문 분석에 매우 중요합니다.
	
	• 기본적으로 제한은 적용되지 않으며 요청에만 기본값이 있습니다.
	• 컨테이너 이름은 Pod 사양에 나타난 대로 나열됩니다.
	• 사이드카는 각 메인 컨테이너 아래에 나열됩니다.
	• TORC `status.CurrentInstallationParams` 필드를 확인하여 현재 적용된 값을 확인하십시오.
		`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:`
		`  linux:`

Trident 운영자 설치 사용자 지정

Trident 운영자를 사용하면 `TridentOrchestrator` 사양의 속성을 사용하여 Trident 설치를 사용자 지정할 수 있습니다. `TridentOrchestrator` 인수가 허용하는 것 이상으로 설치를 사용자 지정하려면 `tridentctl`를 사용하여 필요에 따라 수정할 사용자 지정 YAML 매니페스트를 생성하는 것을 고려하십시오.

컨트롤러 Pod 및 노드 Pod 이해

Trident는 클러스터의 각 작업자 노드에서 단일 컨트롤러 포드와 노드 포드로 실행됩니다. 노드 Pod는 Trident 볼륨을 마운트하려는 모든 호스트에서 실행되어야 합니다.

Kubernetes "노드 선택기"와 "톨러레이션 및 테인트"는 특정 또는 선호하는 노드에서 실행되도록 Pod를 제약하는 데 사용됩니다. `ControllerPlugin`와 `NodePlugin`를 사용하여 제약 및 오버라이드를 지정할 수 있습니다.

- 컨트롤러 플러그인은 스냅샷 및 크기 조정과 같은 볼륨 프로비저닝 및 관리를 처리합니다.
- 노드 플러그인은 스토리지를 노드에 연결하는 작업을 처리합니다.

구성 옵션

> `spec.namespace`는 `TridentOrchestrator`에 지정되어 Trident가 설치된 네임스페이스를 나타냅니다. 이 파라미터는 **Trident**가 설치된 후에는 업데이트할 수 없습니다. 업데이트를 시도하면 `TridentOrchestrator` 상태가 `Failed`로 변경됩니다. Trident는 네임스페이스 간 마이그레이션을 위한 것이 아닙니다.

이 표에서는 `TridentOrchestrator` 속성에 대해 자세히 설명합니다.

매개변수	설명	기본값
`namespace`	Trident를 설치할 네임스페이스	`"default"`
`debug`	Trident에 대한 디버깅 활성화	`false`
`enableForceDetach`	`ontap-san`, `ontap-san-economy`, `ontap-nas` 및 `ontap-nas-economy`에만 해당됩니다. Kubernetes Non-Graceful Node Shutdown(NGNS)과 함께 작동하여 클러스터 관리자가 노드가 비정상 상태가 될 경우 마운트된 볼륨이 있는 워크로드를 새 노드로 안전하게 마이그레이션할 수 있는 기능을 부여합니다. 자세한 내용은 "Trident를 사용하여 상태 저장 애플리케이션의 페일오버 자동화"를 참조하십시오.	`false`
`windows`	`true`로 설정하면 Windows 워커 노드에 설치할 수 있습니다.	`false`

매개변수	설명	기본값
cloudProvider	`"Azure"`AKS 클러스터에서 관리형 ID 또는 클라우드 ID를 사용하는 경우 로 설정합니다. EKS 클러스터에서 클라우드 ID를 사용하는 경우 `"AWS"`로 설정합니다. GKE 클러스터에서 클라우드 ID를 사용하는 경우 `"GCP"`로 설정합니다.	""
cloudIdentity	AKS 클러스터에서 클라우드 ID를 사용하는 경우 워크로드 ID("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx")로 설정하십시오. EKS 클러스터에서 클라우드 ID를 사용하는 경우 AWS IAM 역할("eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role")로 설정하십시오. GKE 클러스터에서 클라우드 ID를 사용하는 경우 클라우드 ID("'iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com'")로 설정합니다.	""
IPv6	IPv6를 통해 Trident 설치	거짓
k8sTimeout	Kubernetes 작업 시간 초과. k8sTimeout 매개변수는 Trident 설치에만 적용됩니다.	180sec
silenceAutosupport	자동 지원 번들을 NetApp으로 자동으로 보내지 마십시오	false
autosupportImage	Autosupport Telemetry를 위한 컨테이너 이미지	"netapp/trident-autosupport10"
autosupportProxy	AutoSupport 원격 측정을 보내기 위한 프록시의 주소/포트	"http://proxy.example.com:8888"
uninstall	Trident를 제거하는 데 사용되는 플래그	false
logFormat	사용할 Trident 로깅 형식 [text,json]	"text"
tridentImage	설치할 Trident 이미지	"netapp/trident:26.02"
imageRegistry	내부 레지스트리 경로, 형식 <registry FQDN>[:port][/subpath]	"registry.k8s.io"
kubeletDir	호스트의 kubelet 디렉터리 경로	"/var/lib/kubelet"
wipeout	Trident를 완전히 제거하기 위해 삭제할 리소스 목록	
imagePullSecrets	내부 레지스트리에서 이미지를 가져오는 시크릿	

매개변수	설명	기본값
`imagePullPolicy`	Trident 오퍼레이터의 이미지 풀링 정책을 설정합니다. 유효한 값은 다음과 같습니다. `Always` 항상 이미지를 풀링합니다. `IfNotPresent` 노드에 이미지가 이미 없는 경우에만 이미지를 풀링합니다. `Never` 이미지를 전혀 풀링하지 않습니다.	`IfNotPresent`
`controllerPluginNodeSelector`	Pod에 대한 추가 노드 선택기입니다. `pod.spec.nodeSelector`과 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
`controllerPluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. `pod.spec.Tolerations`과 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
`nodePluginNodeSelector`	Pod에 대한 추가 노드 선택기입니다. `pod.spec.nodeSelector`과 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
`nodePluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. `pod.spec.Tolerations`과 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
`nodePrep`	Trident가 지정된 데이터 스토리지 프로토콜을 사용하여 볼륨을 관리하도록 Kubernetes 클러스터의 노드를 준비할 수 있습니다. 현재 **`iscsi`**만 지원됩니다. ⓘ OpenShift 4.19부터 이 기능을 지원하는 최소 Trident 버전은 25.06.1입니다.	
`k8sAPIQPS`	컨트롤러가 Kubernetes API 서버와 통신할 때 사용하는 초당 쿼리 수(QPS) 제한입니다. 버스트 값은 QPS 값을 기반으로 자동으로 설정됩니다.	`100`; 선택 사항
`enableConcurrency`	처리량 향상을 위해 동시 Trident 컨트롤러 작업을 지원합니다. ⓘ **Tech Preview**: 이 기능은 실험적이며 현재 ONTAP-NAS(NFS만 해당) 및 ONTAP-SAN(통합 ONTAP 9용 NVMe) 드라이버를 사용한 제한된 병렬 워크플로를 지원하며, ONTAP-SAN 드라이버(통합 ONTAP 9의 iSCSI 및 FCP 프로토콜)에 대한 기존 Tech Preview도 지원합니다.	거짓

매개변수	설명	기본값
`resources`	Trident 컨트롤러 및 노드 Pod에 대한 Kubernetes 리소스 제한 및 요청을 설정합니다. 각 컨테이너 및 사이드카에 대한 CPU 및 메모리를 구성하여 Kubernetes에서 리소스 할당을 관리할 수 있습니다. 리소스 요청 및 제한 구성에 대한 자세한 내용은 "Pod 및 컨테이너에 대한 리소스 관리"을(를) 참조하십시오. • 컨테이너 또는 필드의 이름을 변경하지 마십시오. • 들여쓰기를 변경하지 마십시오. YAML 들여쓰기는 올바른 구문 분석에 매우 중요합니다. • 기본적으로 제한은 적용되지 않으며 요청에만 기본값이 있고 지정하지 않으면 자동으로 적용됩니다. • 컨테이너 이름은 Pod 사양에 나타난 대로 나열됩니다. • 사이드카는 각 메인 컨테이너 아래에 나열됩니다. • TORC `status.CurrentInstallationParams` 필드를 확인하여 현재 적용된 값을 확인하십시오.	`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:`

매개변수	설명	기본값
`httpsMetrics`	Prometheus 메트릭 엔드포인트에 대해 HTTPS를 활성화합니다.	거짓
`hostNetwork`	Trident 컨트롤러에 호스트 네트워킹을 활성화합니다. 이는 멀티홈 네트워크에서 프런트엔드와 백엔드 트래픽을 분리하려는 경우 유용합니다.	거짓

Pod 매개 변수 형식 지정에 대한 자세한 내용은 "노드에 Pod 할당"을 참조하십시오.

샘플 구성

구성 옵션를 정의할 때 `TridentOrchestrator`의 속성을 사용하여 설치를 사용자 지정할 수 있습니다.

기본 사용자 지정 구성

이 예시는 `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` 명령을 실행한 후 생성된 것으로, 기본적인 사용자 지정 설치를 나타냅니다.

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

노드 선택기

이 예제는 노드 선택기를 사용하여 Trident를 설치합니다.

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

Windows 작업자 노드

이 예제는 `cat deploy/crds/tridentorchestrator_cr.yaml` 명령을 실행한 후 생성되었으며 Windows 작업자 노드에 Trident를 설치합니다.

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

AKS 클러스터의 관리 ID

이 예제는 AKS 클러스터에서 관리형 ID를 활성화하기 위해 Trident를 설치합니다.

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

AKS 클러스터의 클라우드 ID

이 예제는 AKS 클러스터에서 클라우드 ID와 함께 사용할 수 있도록 Trident를 설치합니다.

```
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 클러스터의 클라우드 ID

이 예제는 AKS 클러스터에서 클라우드 ID와 함께 사용할 수 있도록 Trident를 설치합니다.

```
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'"
```

GKE용 클라우드 ID

이 예제는 GKE 클러스터에서 클라우드 ID와 함께 사용할 수 있도록 Trident를 설치합니다.

```
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 컨트롤러 및 **Trident Linux** 노드 **Pod**에 대한 **Kubernetes** 리소스 요청 및 제한 구성

이 예제는 Trident 컨트롤러 및 Trident Linux 노드 Pod에 대한 Kubernetes 리소스 요청 및 제한을 구성합니다.

면책 조항: 이 예에 제공된 요청 및 제한 값은 데모 목적으로만 사용됩니다. 환경 및 워크로드 요구 사항에 따라 이러한 값을 조정하십시오.

```
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
```

Trident 컨트롤러 및 **Trident Windows/Linux** 노드 **Pod**에 대한 **Kubernetes** 리소스 요청 및 제한 구성

이 예제는 Trident 컨트롤러와 Trident Windows 및 Linux 노드 Pod에 대한 Kubernetes 리소스 요청 및 제한을 구성합니다.

면책 조항: 이 예에 제공된 요청 및 제한 값은 데모 목적으로만 사용됩니다. 환경 및 워크로드 요구 사항에 따라 이러한 값을 조정하십시오.

```
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
```