



Trident 설치

Trident

NetApp
October 29, 2026

목차

Trident 설치	1
Trident 설치에 대해 알아보세요	1
Trident 26.06에 대한 중요 정보	1
시작하기 전에	1
설치 방법을 선택하십시오	1
설치 모드를 선택하십시오	3
방법 및 모드에 따라 프로세스를 선택합니다	4
설치 방법 간 이동	4
기타 알려진 구성 옵션	4
Trident 운영자를 사용하여 설치합니다	5
Trident 운영자 수동 배포(표준 모드)	5
Trident 운영자를 수동으로 배포합니다(오프라인 모드)	10
Helm을 사용하여 Trident 운영자 배포(표준 모드)	15
Helm을 사용하여 Trident 운영자 배포(오프라인 모드)	22
Trident 운영자 설치 사용자 지정	30
tridentctl을 사용하여 설치	40
tridentctl을 사용하여 Trident를 설치합니다	40
Trident tridentctl 설치 사용자 정의	44
OpenShift 인증 운영자를 사용하여 설치	45
OpenShift OperatorHub를 사용하여 Trident 설치	45
OpenShift 인증 Trident 운영자로 전환하세요	48

Trident 설치

Trident 설치에 대해 알아보세요

다양한 환경과 조직에 Trident를 설치할 수 있도록 NetApp에서는 여러 설치 옵션을 제공합니다. Trident 운영자(수동 또는 Helm 사용)를 사용하거나 `tridentctl`를 사용하여 Trident를 설치할 수 있습니다. 이 항목에서는 적합한 설치 프로세스를 선택하는 데 필요한 중요한 정보를 제공합니다.

Trident 26.06에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

▼ Trident에 대한 중요 정보

- 이제 Kubernetes가 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 먼저 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.

시작하기 전에

설치 경로와 관계없이 다음 항목이 반드시 필요합니다.

- 지원되는 버전의 Kubernetes가 실행되고 기능 요구 사항이 활성화된 지원되는 Kubernetes 클러스터에 대한 모든 권한이 필요합니다. 자세한 내용은 ["요구사항"](#)을(를) 참조하십시오.
- 지원되는 NetApp 스토리지 시스템에 대한 액세스.
- 모든 Kubernetes 작업자 노드에서 볼륨을 마운트하는 기능.
- `kubectl`(또는 OpenShift를 사용하는 경우 `oc`)가 설치 및 구성되어 사용하려는 Kubernetes 클러스터를 관리하는 Linux 호스트.
- `KUBECONFIG` 환경 변수가 Kubernetes 클러스터 구성을 가리키도록 설정되어 있습니다.
- Docker Enterprise와 함께 Kubernetes를 사용하는 경우 ["CLI 액세스를 활성화하려면 해당 단계를 따르십시오"](#).
- 클러스터는 권한 있는 워크로드를 지원해야 합니다.



"[기본 개념](#)"에 대해 아직 잘 모르신다면 지금이 바로 알아볼 좋은 기회입니다.

설치 방법을 선택하십시오

본인에게 맞는 설치 방법을 선택하세요. 결정하기 전에 ["방법 간 이동"](#)에 대한 고려 사항도 검토해야 합니다.

Trident 운영자 사용

수동으로 배포하든 Helm을 사용하든, Trident operator는 설치를 간소화하고 Trident 리소스를 동적으로 관리하는 훌륭한 방법입니다. "[Trident 운영자 배포를 사용자 지정하세요](#)" TridentOrchestrator 사용자 정의 리소스(CR)의 속성을 활용할 수도 있습니다.

Trident 운영자 사용의 이점은 다음과 같습니다.

▼ Trident 객체 생성

Trident 운영자는 Kubernetes 버전에 대해 다음 개체를 자동으로 생성합니다.

- 운영자를 위한 ServiceAccount
- ClusterRole 및 ClusterRoleBinding을 ServiceAccount에 추가합니다
- 전용 PodSecurityPolicy(Kubernetes 1.25 이하 버전용)
- 운영자 자체

▼ 리소스 책임

클러스터 범위의 Trident 오퍼레이터는 클러스터 수준에서 Trident 설치와 관련된 리소스를 관리합니다. 이를 통해 네임스페이스 범위의 오퍼레이터를 사용하여 클러스터 범위 리소스를 관리할 때 발생할 수 있는 오류를 줄일 수 있습니다. 이는 자체 복구 및 패치에 필수적입니다.

▼ 자가 치유 기능

운영자는 Trident 설치를 모니터링하고 배포가 삭제되거나 실수로 수정되는 등의 문제를 해결하기 위해 적극적으로 조치를 취합니다. trident-operator-`<generated-id>` Pod가 생성되어 TridentOrchestrator CR을 Trident 설치와 연결합니다. 이를 통해 클러스터에 Trident 인스턴스가 하나만 존재하도록 하고 설정을 제어하여 설치가 역동성을 갖도록 합니다. 설치에 변경 사항이 발생하면(예: 배포 또는 노드 daemonset 삭제) 운영자는 이를 식별하고 개별적으로 수정합니다.

▼ 기존 설치에 대한 간편한 업데이트

운영자를 이용하면 기존 배포를 간편하게 업데이트할 수 있습니다. 설치를 업데이트하려면 TridentOrchestrator CR만 편집하면 됩니다.

예를 들어, Trident가 디버그 로그를 생성하도록 설정해야 하는 시나리오를 생각해 보십시오. 이를 위해 `TridentOrchestrator`를 패치하여 `spec.debug`를 `true`로 설정하십시오:

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

`TridentOrchestrator`가 업데이트되면 operator는 업데이트를 처리하고 기존 설치에 패치를 적용합니다. 이로 인해 설치를 적절히 수정하기 위해 새 pod 생성이 트리거될 수 있습니다.

▼ 클린 재설치

클러스터 범위 Trident 운영자를 사용하면 클러스터 범위 리소스를 깔끔하게 제거할 수 있습니다. 사용자는 Trident를 완전히 제거하고 쉽게 다시 설치할 수 있습니다.

▼ 자동 Kubernetes 업그레이드 처리

클러스터의 Kubernetes 버전이 지원되는 버전으로 업그레이드되면 운영자는 기존 Trident 설치를 자동으로 업데이트하고 Kubernetes 버전의 요구 사항을 충족하도록 변경합니다.



클러스터가 지원되지 않는 버전으로 업그레이드되면 오퍼레이터는 Trident 설치를 방지합니다. 오퍼레이터를 통해 Trident가 이미 설치된 경우, 지원되지 않는 Kubernetes 버전에 Trident가 설치되었다는 경고 메시지가 표시됩니다.

사용 `tridentctl`

기존 배포 환경을 업그레이드해야 하거나 배포 환경을 고도로 맞춤 설정하려는 경우 를 고려해야 합니다. 이것은 Trident를 배포하는 기존 방법입니다.

를 사용하여 Trident 리소스에 대한 매니페스트를 생성할 수 있습니다. 여기에는 Trident가 설치의 일부로 생성하는 배포, 데몬셋, 서비스 계정 및 클러스터 역할이 포함됩니다.



22.04 릴리스부터 Trident를 설치할 때마다 AES 키를 다시 생성하지 않습니다. 이 릴리스에서는 Trident가 설치 전반에 걸쳐 유지되는 새로운 secret 객체를 설치합니다. 즉, `tridentctl` 22.04에서는 이전 버전의 Trident를 제거할 수 있지만 이전 버전에서는 22.04 설치를 제거할 수 없습니다. 적절한 설치 _방법_을 선택하십시오.

설치 모드를 선택하십시오

조직에서 요구하는 설치 모드(표준, 오프라인 또는 원격)에 따라 배포 프로세스를 결정하십시오.

표준 설치

이는 Trident를 설치하는 가장 쉬운 방법이며 네트워크 제한이 없는 대부분의 환경에서 작동합니다. 표준 설치 모드는 기본 레지스트리를 사용하여 필요한 Trident(`docker.io` 및 `CSI(registry.k8s.io)` 이미지를 저장합니다.

표준 모드를 사용하면 Trident 설치 프로그램은 다음과 같습니다.

- 인터넷을 통해 컨테이너 이미지를 가져옵니다
- Kubernetes 클러스터 내의 모든 적합한 노드에 Trident Pod를 실행하는 배포 또는 노드 데몬셋을 생성합니다

오프라인 설치

외부와 완전히 분리된 환경이나 보안이 강화된 장소에서는 오프라인 설치 모드가 필요할 수 있습니다. 이 경우, 필요한 Trident 및 CSI 이미지를 저장하기 위해 단일 개인 미러링 레지스트리 또는 두 개의 미러링 레지스트리를 생성할 수 있습니다.



레지스트리 구성에 관계없이 CSI 이미지는 하나의 레지스트리에 있어야 합니다.

원격 설치

다음은 원격 설치 프로세스에 대한 개략적인 개요입니다.

- Trident를 배포하려는 원격 시스템에 `kubectrl`의 적절한 버전을 배포합니다.
- Kubernetes 클러스터에서 구성 파일을 복사하고 원격 시스템에 `KUBECONFIG` 환경 변수를 설정하십시오.

- `kubectl get nodes` 명령을 시작하여 필요한 Kubernetes 클러스터에 연결할 수 있는지 확인합니다.
- 표준 설치 단계를 사용하여 원격 시스템에서 배포를 완료합니다.

방법 및 모드에 따라 프로세스를 선택합니다

결정을 내린 후 적절한 프로세스를 선택합니다.

방법	설치 모드
Trident 운영자(수동)	"표준 설치" "오프라인 설치"
Trident 운영자(Helm)	"표준 설치" "오프라인 설치"
<code>tridentctl</code>	"표준 또는 오프라인 설치"

설치 방법 간 이동

설치 방법을 변경할 수도 있습니다. 변경하기 전에 다음 사항을 고려하십시오.

- Trident를 설치 및 제거할 때는 항상 동일한 방법을 사용하십시오. `tridentctl`를 사용하여 배포한 경우 적절한 버전의 `tridentctl` 바이너리를 사용하여 Trident를 제거해야 합니다. 마찬가지로 오퍼레이터를 사용하여 배포하는 경우 `TridentOrchestrator` CR을 편집하고 `spec.uninstall=true`를 설정하여 Trident를 제거해야 합니다.
- 운영자 기반 배포를 제거하고 대신 `tridentctl`을 사용하여 Trident를 배포하려면 먼저 `TridentOrchestrator`를 편집하고 `spec.uninstall=true`를 설정하여 Trident를 제거해야 합니다. 그런 다음 `TridentOrchestrator`와 운영자 배포를 삭제합니다. `tridentctl`를 사용하여 설치할 수 있습니다.
- 수동 오퍼레이터 기반 배포 방식을 사용하다가 Helm 기반 Trident 오퍼레이터 배포 방식으로 전환하려면, 먼저 기존 오퍼레이터를 수동으로 제거한 후 Helm을 사용하여 설치해야 합니다. 이렇게 하면 Helm이 필요한 레이블과 어노테이션을 포함하여 Trident 오퍼레이터를 배포할 수 있습니다. 이 단계를 거치지 않으면 Helm 기반 Trident 오퍼레이터 배포 시 레이블 유효성 검사 오류와 어노테이션 유효성 검사 오류가 발생합니다.
- `tridentctl` 기반 배포 환경이 있는 경우 Trident를 제거하지 않고 Helm 기반 또는 Operator 기반 배포를 수행할 수 있습니다.

기타 알려진 구성 옵션

VMWare Tanzu Portfolio 제품에 Trident를 설치할 때:

- `--kubelet-dir` 플래그는 kubelet 디렉터리의 위치로 설정해야 합니다. 기본적으로 이것은 `/var/vcap/data/kubelet`입니다.

`--kubelet-dir`을 사용하여 kubelet 위치를 지정하는 것은 Trident Operator, Helm 및 `tridentctl` 배포에서 작동하는 것으로 알려져 있습니다.

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 26.06에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

▼ Trident에 대한 중요 정보

- Kubernetes 1.37는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 먼저 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.

▼ Details

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/v26.06.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.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:26.06
  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

```

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 경우 [link:../troubleshooting.html](#)["로그를 확인합니다"] .

상태	설명
설치	운영자는 이 <code>TridentOrchestrator</code> CR을 사용하여 Trident를 설치하고 있습니다.
설치됨	Trident가 성공적으로 설치되었습니다.
제거하기	운영자가 Trident을 제거 중입니다 <code>spec.uninstall=true</code> .

상태	설명
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 |
+-----+-----+
| 26.06.0       | 26.06.0       |
+-----+-----+
```

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](#)"를 사용하십시오.

Trident에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

▼ Trident에 대한 중요 정보

- Kubernetes 1.37는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 먼저 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.

▼ Details

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-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.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:26.06.0`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:26.06.0` 이미지가 모두 하나의 레지스트리에 있는 경우.
- `image: <your-registry>/netapp/trident-operator:26.06.0` 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:26.06"
tridentImage: "<your-registry>/trident:26.06.0"
```

서로 다른 레지스트리에 있는 이미지

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

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 `Trident`와 `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:26.06
  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

```

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 경우 [link:../troubleshooting.html](#) ["로그를 확인합니다"] .

상태	설명
설치	운영자는 이 <code>TridentOrchestrator</code> CR을 사용하여 Trident를 설치하고 있습니다.
설치됨	Trident가 성공적으로 설치되었습니다.
제거하기	운영자가 Trident을 제거 중입니다 <code>spec.uninstall=true</code> .
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	<code>TridentOrchestrator</code> 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 |
+-----+-----+
| 26.06.0       | 26.06.0       |
+-----+-----+
```

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 26.06에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

▼ Trident에 대한 중요 정보

- Kubernetes 1.37는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 먼저

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 **Trident**를 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 `100.2606.0`는 설치하려는 Trident 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0  
--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
<code>--values</code> (또는 <code>-f</code>)	재정의할 포함하는 YAML 파일을 지정합니다. 이 파일은 여러 번 지정할 수 있으며 가장 오른쪽에 있는 파일이 우선 적용됩니다.

옵션	설명
--set	명령줄에서 재정의를 지정합니다.

예를 들어, `debug`의 기본값을 변경하려면 다음 명령을 실행하십시오. 여기서 `100.2606.0`는 설치하려는 Trident 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
--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	
affinity	POD 할당에 대한 선호도	<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 파일에서 기본 선호도를 제거하지 마십시오. 사용자 지정 선호도를 제공하려면 기본 선호도를 확장하십시오.</p> </div>

옵션	설명	기본값
tridentControllerPluginNodeSelector	Pod에 대한 추가 노드 선택기입니다. 자세한 내용은 컨트롤러 Pod 및 노드 Pod 이해 을(를) 참조하십시오.	
tridentControllerPluginTolerations	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 자세한 내용은 컨트롤러 Pod 및 노드 Pod 이해 을(를) 참조하십시오.	
tridentNodePluginNodeSelector	Pod에 대한 추가 노드 선택기입니다. 자세한 내용은 컨트롤러 Pod 및 노드 Pod 이해 을(를) 참조하십시오.	
tridentNodePluginTolerations	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 자세한 내용은 컨트롤러 Pod 및 노드 Pod 이해 을(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 재정의의를 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

옵션	설명	기본값
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.	""
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 파일오버 자동화" 을 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

옵션	설명	기본값
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.	""
cloudIdentity	AKS 클러스터에서 클라우드 ID를 사용하는 경우 워크로드 ID("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx")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	Trident 컨트롤러의 동시 작동을 지원하여 처리량을 향상시킵니다. 지원되는 드라이버에 대한 자세한 내용은 링크: Controller Scalability documentation 를 참조하십시오.	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	<p>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.</p> <p>리소스 요청 및 제한 구성에 대한 자세한 내용은 "Pod 및 컨테이너에 대한 리소스 관리"을(를) 참조하십시오.</p> <div> <div>ⓘ</div> <ul style="list-style-type: none"> 컨테이너 또는 필드의 이름을 변경하지 마십시오. 들여쓰기를 변경하지 마십시오. YAML 들여쓰기는 올바른 구문 분석에 매우 중요합니다. </div> <div> <div>ⓘ</div> <ul style="list-style-type: none"> 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. </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>

옵션	설명	기본값
httpsMetrics	Prometheus 메트릭 엔드포인트에 대해 HTTPS를 활성화합니다.	거짓
hostNetwork	Trident 컨트롤러에 호스트 네트워킹을 활성화합니다. 이는 멀티홈 네트워크에서 프론트엔드와 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.

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"를 사용하십시오.

Trident 26.06에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

▼ Trident에 대한 중요 정보

- Kubernetes 1.37는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 먼저 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 `100.2606.0`는 설치하려는 Trident 버전입니다.

하나의 레지스트리에 있는 이미지

```
helm install <name> netapp-trident/trident-operator --version  
100.2606.0 --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  
100.2606.0 --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}
```



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

`helm list`를 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)	재정의의 포함하는 YAML 파일을 지정합니다. 이 파일은 여러 번 지정할 수 있으며 가장 오른쪽에 있는 파일이 우선 적용됩니다.
--set	명령줄에서 재정의의를 지정합니다.

예를 들어, `debug`의 기본값을 변경하려면 다음 명령을 실행하십시오. 여기서 `100.2606.0`는 설치하려는 Trident 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
--create-namespace --namespace trident --set tridentDebug=true
```

nodePrep 값을 추가하려면 다음 명령을 실행하십시오.

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
--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	<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 파일에서 기본 선호도를 제거하지 마십시오. 사용자 지정 선호도를 제공하려면 기본 선호도를 확장하십시오.</p> </div>
tridentControllerPluginNodeSelector	Pod에 대한 추가 노드 선택기입니다. 자세한 내용은 "컨트롤러 Pod 및 노드 Pod 이해" 을(를) 참조하십시오.	
tridentControllerPluginTolerations	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 자세한 내용은 "컨트롤러 Pod 및 노드 Pod 이해" 을(를) 참조하십시오.	
tridentNodePluginNodeSelector	Pod에 대한 추가 노드 선택기입니다. 자세한 내용은 "컨트롤러 Pod 및 노드 Pod 이해" 을(를) 참조하십시오.	

옵션	설명	기본값
tridentNodePluginTolerations	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 자세한 내용은 "컨트롤러 Pod 및 노드 Pod 이해" 을 (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 재정의 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). <div>  <div>tridentK8sTimeout 매개변수는 Trident 설치에만 적용됩니다.</div> </div>	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 파일오버 자동화" 을 참조하십시오.	false
excludePodSecurityPolicy	운영자 포드 보안 정책을 생성에서 제외합니다.	false

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

옵션	설명	기본값
resources	<p>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.</p> <p>리소스 요청 및 제한 구성에 대한 자세한 내용은 "Pod 및 컨테이너에 대한 리소스 관리"을(를) 참조하십시오.</p> <div> <div>!</div> <ul style="list-style-type: none"> 컨테이너 또는 필드의 이름을 변경하지 마십시오. 들여쓰기를 변경하지 마십시오. YAML 들여쓰기는 올바른 구문 분석에 매우 중요합니다. </div> <div> <div>i</div> <ul style="list-style-type: none"> 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. </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 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 Trident가 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 파일오버 자동화" 를 참조하십시오.	false
windows	`true`로 설정하면 Windows 워커 노드에 설치할 수 있습니다.	false

매개변수	설명	기본값
cloudProvider	<p>`"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.</p>	""
cloudIdentity	<p>AKS 클러스터에서 클라우드 ID를 사용하는 경우 워크로드 ID("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx")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.</p>	""
IPv6	IPv6를 통해 Trident 설치	거짓
k8sTimeout	<p>Kubernetes 작업 시간 초과.</p> <div>  <p>k8sTimeout 매개변수는 Trident 설치에만 적용됩니다.</p> </div>	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:26.06"
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`와 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
controllerPluginTolerations	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. `pod.spec.Tolerations`와 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
nodePluginNodeSelector	Pod에 대한 추가 노드 선택기입니다. `pod.spec.nodeSelector`와 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
nodePluginTolerations	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. `pod.spec.Tolerations`와 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	Trident 컨트롤러의 동시 작동을 지원하여 처리량을 향상시킵니다. 지원되는 드라이버에 대한 자세한 내용은 링크: Controller Scalability documentation 를 참조하십시오.	resources

매개변수	설명	기본값
<p>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.</p> <p>리소스 요청 및 제한 구성에 대한 자세한 내용은 "Pod 및 컨테이너에 대한 리소스 관리"을(를) 참조하십시오.</p> <div> <div>  <ul style="list-style-type: none"> 컨테이너 또는 필드의 이름을 변경하지 마십시오. 들여쓰기를 변경하지 마십시오. YAML 들여쓰기는 올바른 구문 분석에 매우 중요합니다. </div> <div>  <ul style="list-style-type: none"> 기본적으로 제한은 적용되지 않으며 요청에만 기본값이 있고 지정하지 않으면 자동으 </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>	httpsMetrics

매개변수	설명	기본값
Prometheus 메트릭 엔드포인트에 대해 HTTPS를 활성화합니다.	거짓	hostNetwork



Pod 매개 변수 형식 지정에 대한 자세한 내용은 "[노드에 Pod 할당](#)"을 참조하십시오.

샘플 구성

[구성 옵션](#)을 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-xxxxxxxxxxxx'

```

▼ 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 Pod에 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 Pod에 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

```

tridentctl을 사용하여 설치

tridentctl을 사용하여 Trident를 설치합니다

`tridentctl`를 사용하여 Trident를 설치할 수 있습니다. 이 프로세스는 Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장되어 있거나 저장되어 있지 않은 설치에 적용됩니다. `tridentctl` 배포를 사용자 지정하려면 `xref:{relative_path}kubernetes-customize-deploy-tridentctl.html["tridentctl 배포 사용자 지정"]`를 참조하십시오.

Trident10에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

▼ Trident에 대한 중요 정보

- Kubernetes 1.27는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 먼저 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.

`tridentctl`를 사용하여 Trident 설치

"[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.

▼ Details

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 패키지는 Trident Pod를 생성하고, 상태를 유지하는 데 사용되는 CRD 객체를 구성하며, 클러스터 호스트에 볼륨을 프로비저닝하고 연결하는 등의 작업을 수행하도록 CSI 사이드카를 초기화합니다. "[GitHub의 Assets 섹션](#)"에서 최신 버전의 Trident 설치 프로그램을 다운로드하여 압축을 풉니다. 예제의 `_<trident-installer-XX.XX.X.tar.gz>`를 선택한 Trident 버전으로 업데이트합니다.

```
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
```

2단계: Trident 설치

`tridentctl install` 명령을 실행하여 원하는 네임스페이스에 Trident를 설치합니다. 추가 인수를 추가하여 이미지 레지스트리 위치를 지정할 수 있습니다.

표준 모드

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

하나의 레지스트리에 있는 이미지

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

서로 다른 레지스트리에 있는 이미지

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

설치 상태는 다음과 같이 표시되어야 합니다.

```
....  
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.  
....
```

설치 확인

파드 생성 상태 또는 `tridentctl`를 사용하여 설치를 확인할 수 있습니다.

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-679648bd45-cv2mx	6/6	Running	0	5m29s
trident-node-linux-vgc8n	2/2	Running	0	5m29s



설치 프로그램이 성공적으로 완료되지 않거나 trident-controller-<generated id> (`trident-csi-<generated id>`에 실행 중 상태가 표시되지 않으면 플랫폼이 설치되지 않은 것입니다. `d`를 사용하여 ["디버그 모드 켜기"](#)하고 문제를 해결하세요.

사용 tridentctl

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

```
./tridentctl -n trident version
```

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

샘플 구성

다음 예는 `tridentctl`을 사용하여 Trident를 설치하는 샘플 구성을 제공합니다.

▼ Windows 노드

Windows 노드에서 Trident를 실행하려면 다음을 수행합니다.

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

▼ 강제 분리

자세한 내용은 ["Trident를 사용하여 상태 저장 애플리케이션의 페일오버 자동화"](#)를 참조하십시오.

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

▼ 동시 Trident 컨트롤러 작업 활성화

처리량 향상을 위해 Trident 컨트롤러의 동시 작업을 활성화하려면 이 예시와 같이 설치 중에 `--enable-concurrency` 옵션을 추가하십시오. 지원되는 드라이버에 대한 자세한 내용은 링크: [Controller Scalability documentation](#)를 참조하십시오.

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

Trident tridentctl 설치 사용자 정의

Trident 설치 프로그램을 사용하여 설치를 사용자 지정할 수 있습니다.

설치 프로그램에 대해 알아보십시오

Trident 설치 프로그램을 사용하면 속성을 사용자 지정할 수 있습니다. 예를 들어, Trident 이미지를 개인 저장소에 복사한 경우 `--trident-image``를 사용하여 이미지 이름을 지정할 수 있습니다. Trident 이미지와 필요한 CSI 사이드카 이미지를 모두 개인 저장소에 복사한 경우 `--image-registry` 스위치를 사용하여 해당 저장소의 위치를 지정하는 것이 좋습니다. 이 스위치는 `<registry FQDN>[:port]` 형식을 사용합니다.



비공개 리포지토리에 Trident를 설치할 때 `--image-registry` 스위치를 사용하여 리포지토리 위치를 지정하는 경우 리포지토리 경로에 `/netapp/``를 사용하지 마십시오. 예를 들면 다음과 같습니다. `./tridentctl install --image-registry <image-registry> -n <namespace>`

Kubernetes 배포판을 사용하는 경우, `'kubelet'`가 일반적인 `'/var/lib/kubelet'`가 아닌 다른 경로에 데이터를 저장하는 경우 `--kubelet-dir``를 사용하여 대체 경로를 지정할 수 있습니다.

설치 프로그램의 인수로 제공되는 범위를 넘어 설치를 사용자 지정해야 하는 경우 배포 파일을 사용자 지정할 수도 있습니다. `--generate-custom-yaml`` 매개변수를 사용하면 설치 프로그램의 `setup`` 디렉터리에 다음 YAML 파일이 생성됩니다.

- `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`` *

이 파일들을 생성한 후에는 필요에 따라 수정하고, `--use-custom-yaml``를 사용하여 사용자 지정 배포를 설치할 수

있습니다.

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

OpenShift 인증 운영자를 사용하여 설치

OpenShift OperatorHub를 사용하여 Trident 설치

Red Hat OpenShift를 사용하는 경우 Red Hat 인증 운영자를 사용하여 NetApp Trident를 설치할 수 있습니다. Red Hat OpenShift Container Platform에서 Trident를 설치하려면 다음 절차를 따르십시오.

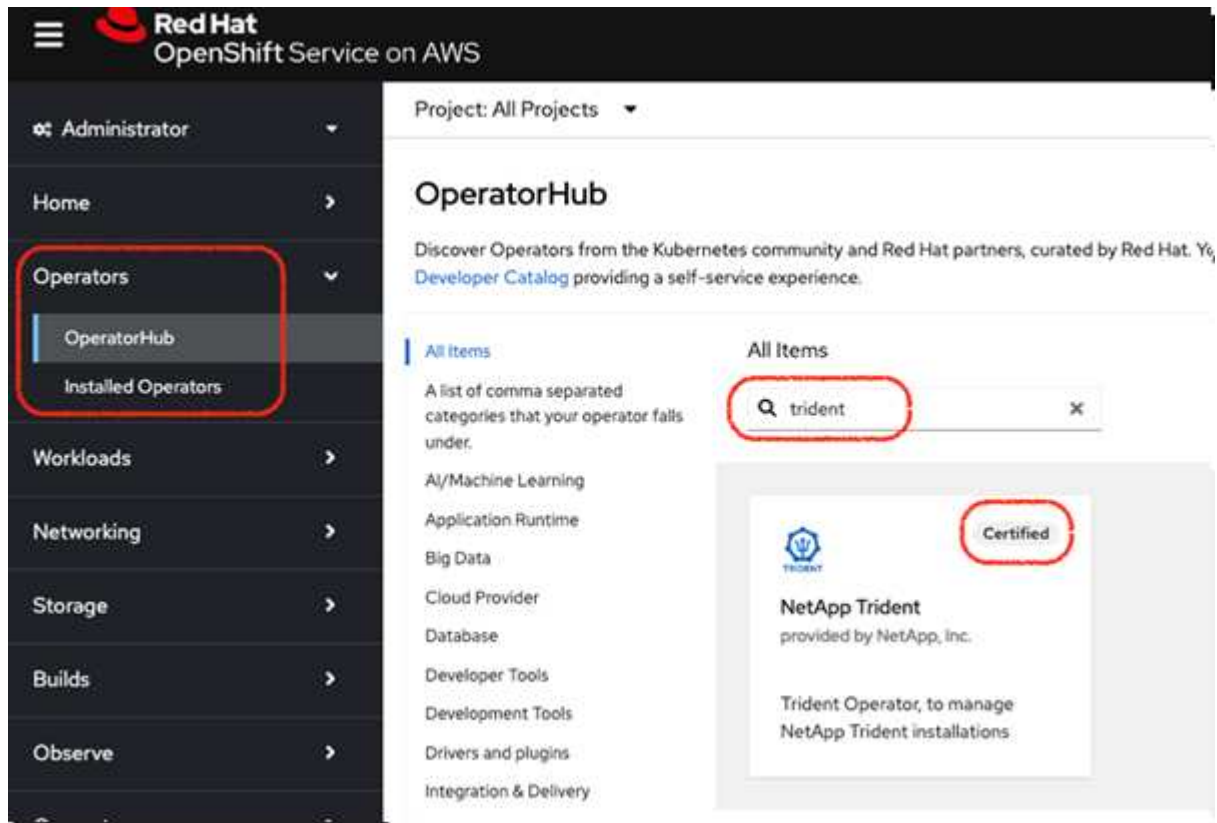
시작하기 전에

설치를 시작하기 전에 "[Trident 설치를 위한 환경을 준비하세요](#)".

Trident 운영자를 찾아서 설치하십시오

단계

1. OpenShift OperatorHub로 이동하여 NetApp Trident를 검색합니다.



2. *NetApp Trident*를 클릭하여 설치 설정을 엽니다.
3. 필요한 옵션을 선택하고 *Install*을 클릭하여 Operator 구성을 엽니다.



NetApp Trident

25.2.1 provided by NetApp, Inc.



Install

Channel

stable

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>.

Version

25.2.1

25.2.1

25.2.0

Source

Certified



최신 Operator 버전을 선택했는지 확인하십시오.

4. 모든 매개변수를 그대로 유지하고 *Install*을 클릭하십시오.

OperatorHub > Operator Installation

Install Operator

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.

Update channel *

stable

Version *

25.2.1

Installation mode *

- ☒ All namespaces on the cluster (default)
Operator will be available in all Namespaces.
- ☐ A specific namespace on the cluster
This mode is not supported by this Operator

Installed Namespace *

openshift-operators

Update approval *

- ☒ Automatic
- ☐ Manual



NetApp Trident

provided by NetApp, Inc.

Provided APIs

TO Trident Orchestrator

Used to deploy NetApp Trident.

TC Trident Configurator

Automates AWS FSxN backend configuration

5. *운영자 보기*를 클릭하여 운영자의 세부 정보를 확인합니다.



Provided APIs

Trident Orchestrator

Used to deploy NetApp Trident.

Create instance

Trident Configurator

Automates AWS FSxN backend configuration

Create instance

6. *YAML 보기*를 클릭하고 다음 내용을 양식에 붙여넣으세요.

```
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
```



사용자 인터페이스는 기본 예시를 제공합니다. 전체 구성을 복사하는 대신 직접 편집할 수 있습니다.

선택 사항: 동시 실행 활성화



동시성을 활성화하려면 spec에 다음 필드를 추가하십시오.

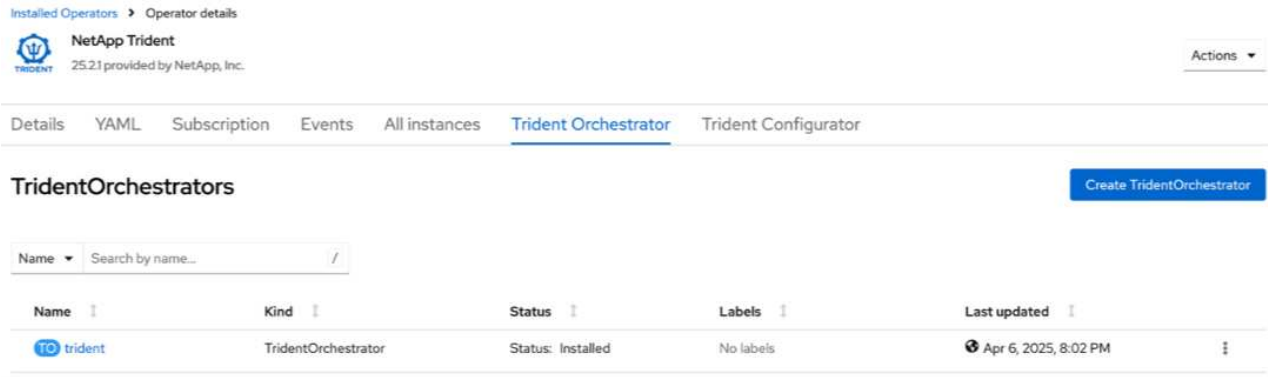
```
enableConcurrency: true
```



- Red Hat Enterprise Linux CoreOS(RHCOS)에는 iSCSI가 활성화 및 구성되어 있지 않습니다.
- `nodePrep` 매개변수를 추가하면 모든 OpenShift 워커 노드에서 iSCSI 및 멀티패스 서비스를 구성하고 활성화할 수 있습니다.

- OpenShift 4.19부터 이 기능을 지원하는 최소 Trident 버전은 25.06.1입니다.

1. *생성*을 클릭하면 Trident Orchestrator가 완전히 설치됩니다.



Trident 운영자를 제거합니다

단계

1. 설치된 Operator 목록에서 Trident Operator를 선택합니다.
2. 연산자에서 모든 피연산자 인스턴스를 삭제할지 여부를 선택하십시오.



이 연산자의 모든 피연산자 인스턴스 삭제 확인란을 선택하지 않으면 Trident가 제거되지 않습니다.

3. *제거*를 클릭합니다.

OpenShift 인증 Trident 운영자로 전환하세요

Red Hat OpenShift 인증 Trident 오퍼레이터는 커뮤니티 오퍼레이터, Helm 기반 설치 또는 수동으로 배포된 오퍼레이터에서 전환할 수 있습니다. 각 방법의 프로세스에는 기존 오퍼레이터를 제거한 다음 OperatorHub를 사용하여 인증된 오퍼레이터를 설치하는 작업이 포함됩니다.

시작하기 전에

설치를 시작하기 전에 "[Trident 설치를 위한 환경을 준비하세요](#)".

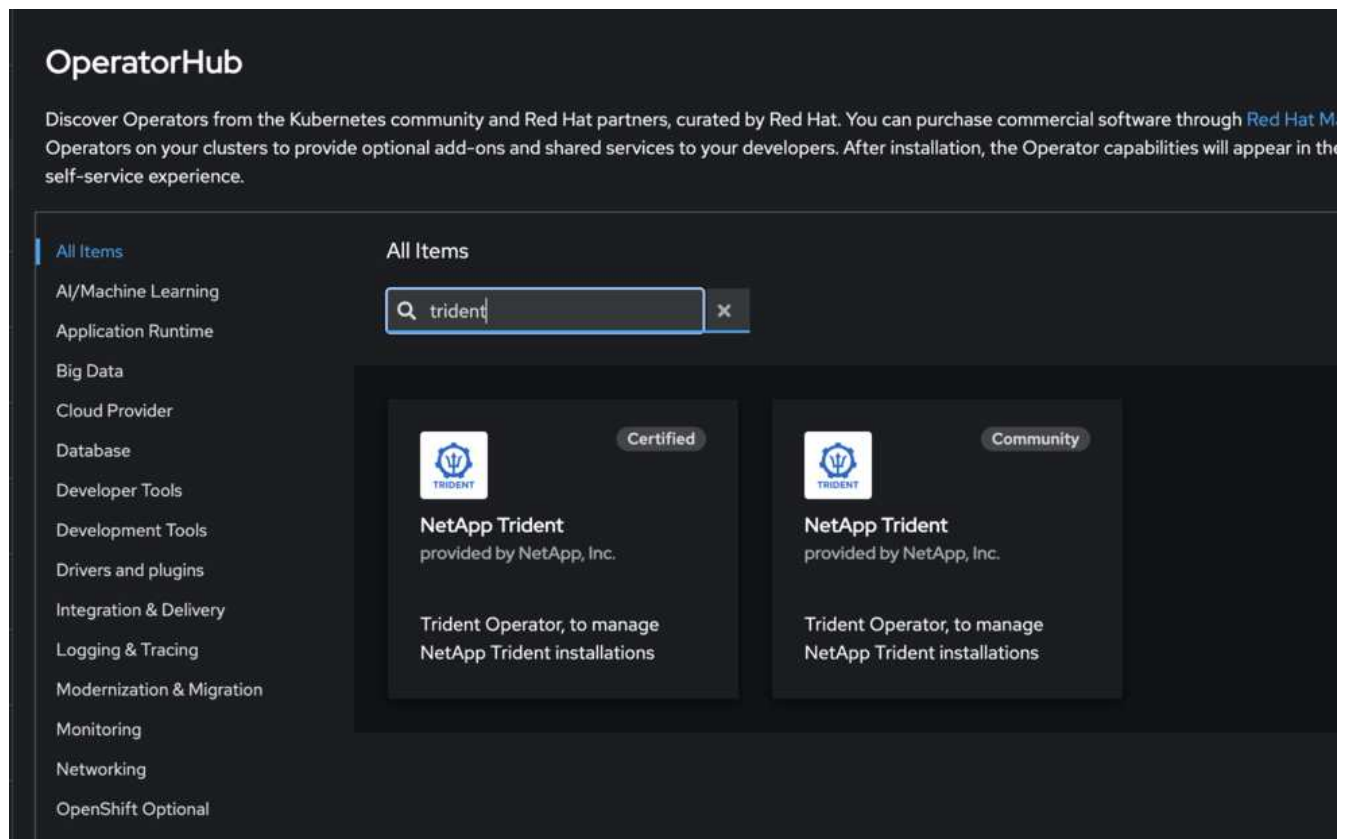


TridentOrchestrator 사용자 지정 리소스(CR)를 제거 과정 중에 삭제하지 마십시오. TridentOrchestrator CR은 인증된 운영자를 설치한 후 필요한 백엔드 및 스토리지 클래스 구성을 유지합니다.

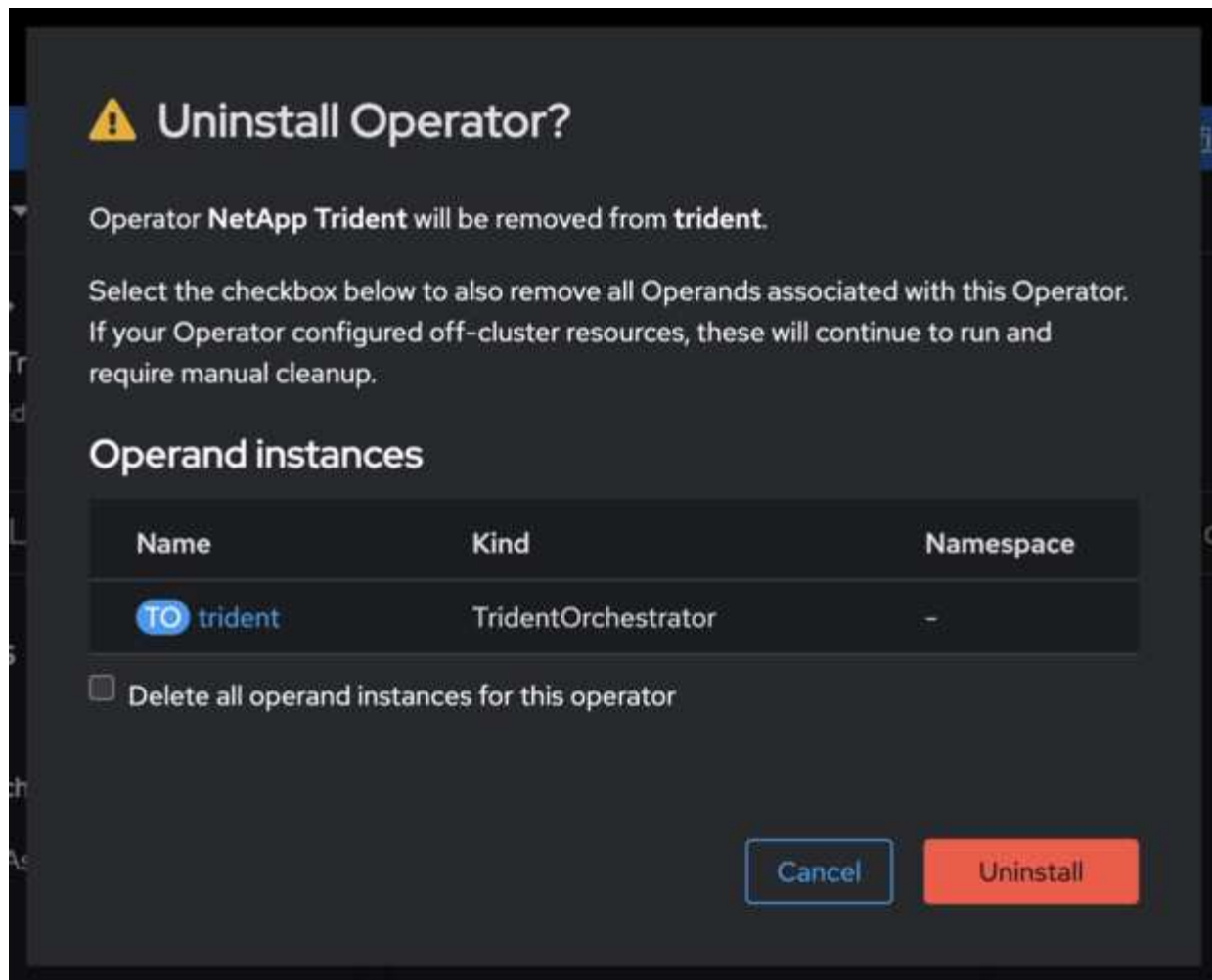
커뮤니티 운영자에서 전환

단계

1. OpenShift 콘솔을 사용하여 OperatorHub로 이동합니다.



2. NetApp Trident 커뮤니티 운영자를 찾으세요.



*이 연산자에서 모든 피연산자 인스턴스 삭제*를 선택하지 마십시오.

3. *제거*를 클릭합니다.
4. 제거가 완료되면 [OpenShift 인증 운영자를 설치하세요](#)(으)로 진행하십시오.

Helm 기반 운영자 설치에서 전환

단계

1. Trident 설치에 대한 Helm 릴리스를 나열합니다.

```
helm ls -n trident
```

2. Helm 릴리스를 제거하세요:

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

3. 제거가 완료되면 [OpenShift 인증 운영자를 설치하세요](#)(으)로 진행하십시오.

수동으로 배포된 **operator**에서 전환

설치 패키지의 `bundle.yaml`를 사용하여 수동으로 운영자를 배포하여 Trident를 설치한 경우 동일한 매니페스트를 삭제하여 제거하십시오.

단계

1. 번들 매니페스트를 사용하여 운영자 배포를 삭제합니다.

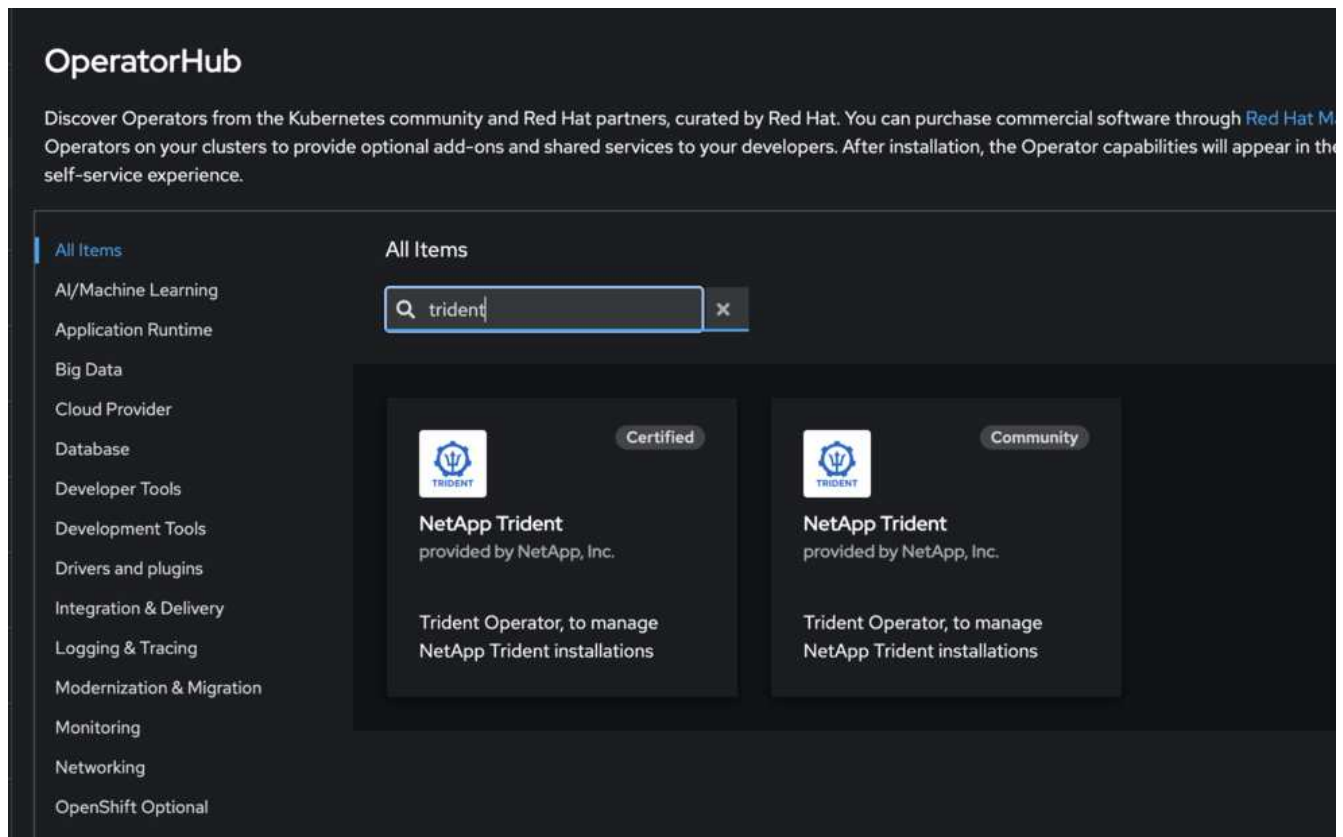
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. 제거가 완료되면 [OpenShift 인증 운영자를 설치하세요](#)(으)로 진행하십시오.

OpenShift 인증 운영자를 설치하세요

단계

1. Red Hat OperatorHub로 이동합니다.
2. NetApp Trident Operator를 검색하여 선택하십시오.



3. 화면의 지시에 따라 Operator를 설치하십시오.

확인

- 콘솔의 OperatorHub에서 새 인증 운영자가 성공적으로 설치되었는지 확인하십시오.

저작권 정보

Copyright © 2026 NetApp, Inc. All Rights Reserved. 미국에서 인쇄된 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이선스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이선스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이선스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이선스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 <http://www.netapp.com/TM>에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.