Installare Trident

Trident

NetApp
July 01, 2026

Sommario

Installare Trident

Scopri l'installazione di Trident

Per garantire che Trident possa essere installato in un'ampia varietà di ambienti e organizzazioni, NetApp offre diverse opzioni di installazione. Puoi installare Trident utilizzando l'operatore Trident (manualmente o tramite Helm) o con `tridentctl`. Questo argomento fornisce informazioni importanti per la scelta del processo di installazione più adatto a te.

Informazioni critiche su Trident 26.02

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.35 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Prima di iniziare

Indipendentemente dal percorso di installazione, è necessario disporre di:

- Privilegi completi per un cluster Kubernetes supportato che esegue una versione supportata di Kubernetes e con i requisiti delle funzionalità abilitati. Consultare "requisiti" per i dettagli.
- Accesso a un sistema storage NetApp.
- Capacità di montare volumi da tutti i nodi worker di Kubernetes.
- Un host Linux con `kubectl` (o `oc`, se si utilizza OpenShift) installato e configurato per gestire il cluster che si desidera utilizzare.
- La `KUBECONFIG`variabile di ambiente impostata per puntare alla configurazione del cluster Kubernetes.
- Se si utilizza Kubernetes con Docker Enterprise, "segui i loro passaggi per abilitare l'accesso CLI".
- Il cluster deve supportare carichi di lavoro privilegiati.

Se non hai familiarità con il "concetti di base", questo è il momento giusto per farlo.

Scegli il tuo metodo di installazione

Seleziona il metodo di installazione più adatto alle tue esigenze. Dovresti anche esaminare le considerazioni per "spostarsi tra i metodi" prima di prendere una decisione.

Utilizzo dell'operatore Trident

Che si utilizzi la distribuzione manuale o Helm, l'operatore Trident è un ottimo modo per semplificare l'installazione e gestire dinamicamente le risorse Trident. Puoi persino "personalizza la distribuzione dell'operatore Trident" usando gli attributi nella `TridentOrchestrator` custom resource (CR).

I vantaggi dell'utilizzo dell'operatore Trident includono:

Creazione oggetto Trident

L'operatore Trident crea automaticamente i seguenti oggetti per la tua versione di Kubernetes.

- ServiceAccount per l'operatore
- ClusterRole e ClusterRoleBinding al ServiceAccount
- Dedicato PodSecurityPolicy (per Kubernetes 1.25 e versioni precedenti)
- L'operatore stesso

Responsabilità delle risorse

L'operatore Trident con ambito cluster gestisce le risorse associate a un'installazione Trident a livello di cluster. Questo riduce gli errori che potrebbero verificarsi durante la gestione delle risorse con ambito cluster utilizzando un operatore con ambito namespace. Questo è essenziale per l'auto-riparazione e l'applicazione di patch.

Capacità di self-healing

L'operatore monitora l'installazione di Trident e adotta misure attive per risolvere eventuali problemi, ad esempio quando il deployment viene eliminato o modificato accidentalmente. Un `trident-operator-<generated-id>` pod viene creato che associa una `TridentOrchestrator` CR a un'installazione di Trident. Questo garantisce che ci sia una sola istanza di Trident nel cluster e ne controlla la configurazione, assicurando che l'installazione sia idempotente. Quando vengono apportate modifiche all'installazione (ad esempio, l'eliminazione del deployment o del node daemonset), l'operatore le identifica e le corregge individualmente.

Aggiornamenti facili alle installazioni esistenti

È possibile aggiornare facilmente una distribuzione esistente con l'operatore. È sufficiente modificare il `TridentOrchestrator` CR per apportare aggiornamenti a un'installazione.

Ad esempio, considera uno scenario in cui è necessario abilitare Trident per generare log di debug. Per farlo, applica una patch al tuo `TridentOrchestrator` per impostare `spec.debug` su `true`:

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

Dopo `TridentOrchestrator` l'aggiornamento, l'operatore elabora gli aggiornamenti e applica le patch all'installazione esistente. Questo potrebbe innescare la creazione di nuovi pod per modificare l'installazione di conseguenza.

Reinstallazione pulita

L'operatore Trident con ambito cluster consente la rimozione pulita delle risorse con ambito cluster. Gli utenti possono disinstallare completamente Trident e reinstallarlo facilmente.

Gestione automatica degli upgrade di Kubernetes

Quando la versione Kubernetes del cluster viene aggiornata a una versione supportata, l'operatore aggiorna automaticamente un'installazione Trident esistente e la modifica per garantire che soddisfi i requisiti della versione Kubernetes.

> ⓘ Se il cluster viene aggiornato a una versione non supportata, l'operatore impedisce l'installazione di Trident. Se Trident è già stato installato con l'operatore, viene visualizzato un avviso per indicare che Trident è installato su una versione di Kubernetes non supportata.

Utilizzando `tridentctl`

Se hai una distribuzione esistente che deve essere aggiornata o se desideri personalizzarla in modo significativo, dovresti prendere in considerazione . Questo è il metodo convenzionale per distribuire Trident.

È possibile generare i manifest per le risorse Trident. Questo include la deployment, il daemonset, l'account di servizio e il cluster role che Trident crea come parte della sua installazione.

> ⓘ A partire dalla versione 22.04, le chiavi AES non verranno più rigenerate ogni volta che si installa Trident. Con questa versione, Trident installerà un nuovo oggetto segreto che persiste tra le installazioni. Ciò significa che, `tridentctl` nella versione 22.04 è possibile disinstallare le versioni precedenti di Trident, ma le versioni precedenti non possono disinstallare le installazioni della 22.04. Selezionare il metodo di installazione appropriato.

Scegli la modalità di installazione

Determina il processo di distribuzione in base alla *modalità di installazione* (Standard, Offline o Remote) richiesta dalla tua organizzazione.

Installazione standard

Questo è il modo più semplice per installare Trident e funziona per la maggior parte degli ambienti che non impongono restrizioni di rete. La modalità di installazione standard utilizza i registri predefiniti per memorizzare le immagini Trident (`docker.io`) e CSI (`registry.k8s.io`) richieste.

Quando si utilizza la modalità standard, il programma di installazione di Trident:

- Recupera le immagini del container tramite Internet
- Crea un deployment o un daemonset di nodi, che avvia i pod Trident su tutti i nodi idonei nel cluster Kubernetes

Installazione offline

La modalità di installazione offline potrebbe essere necessaria in una posizione isolata o sicura. In questo scenario, è possibile creare un singolo registro privato con mirroring o due registri con mirroring per archiviare le immagini Trident e CSI richieste.

Indipendentemente dalla configurazione del registro, le immagini CSI devono risiedere in un unico registro.

Installazione remota

Ecco una panoramica high-level del processo di installazione remota:

- Distribuisci la versione appropriata di `kubectl` sulla macchina remota da cui desideri distribuire Trident.
- Copia i file di configurazione dal cluster Kubernetes e imposta la `KUBECONFIG` variabile di ambiente sulla macchina remota.
- Avvia un `kubectl get nodes` comando per verificare che sia possibile connettersi al cluster Kubernetes richiesto.
- Completare la distribuzione dalla macchina remota utilizzando i passaggi di installazione standard.

Seleziona il processo in base al tuo metodo e modalità

Dopo aver preso le tue decisioni, seleziona il processo appropriato.

Metodo	Modalità di installazione
Trident operator (manualmente)	"Installazione standard" "Installazione offline"
Operatore Trident (Helm)	"Installazione standard" "Installazione offline"

Metodo	Modalità di installazione
`tridentctl`	"Installazione standard o offline"

Spostarsi tra i metodi di installazione

Puoi decidere di cambiare il metodo di installazione. Prima di farlo, considera quanto segue:

- Utilizzare sempre lo stesso metodo per installare e disinstallare Trident. Se si è eseguito il deployment con `tridentctl`, è necessario utilizzare la versione appropriata del `tridentctl` binario per disinstallare Trident. Analogamente, se si esegue il deployment con l'operatore, è necessario modificare la `TridentOrchestrator` CR e impostare `spec.uninstall=true` per disinstallare Trident.
- Se hai una distribuzione basata su operatore che vuoi rimuovere e usare invece `tridentctl` per distribuire Trident, dovresti prima modificare `TridentOrchestrator` e impostare `spec.uninstall=true` per disinstallare Trident. Quindi elimina `TridentOrchestrator` e la distribuzione dell'operatore. Puoi quindi installare utilizzando `tridentctl`.
- Se si dispone di una distribuzione manuale basata sull'operatore e si desidera utilizzare la distribuzione dell'operatore Trident basata su Helm, è necessario disinstallare manualmente prima l'operatore e quindi eseguire l'installazione di Helm. Ciò consente a Helm di distribuire l'operatore Trident con le etichette e le annotazioni richieste. Se non si esegue questa operazione, la distribuzione dell'operatore Trident basata su Helm non riuscirà a causa di un errore di convalida delle etichette e delle annotazioni.
- Se si dispone di una distribuzione basata su `tridentctl`, è possibile eseguire una distribuzione basata su Helm o su Operator senza disinstallare Trident.

Altre opzioni di configurazione note

Quando si installa Trident sui prodotti VMWare Tanzu Portfolio:

- Il `--kubelet-dir` flag dovrebbe essere impostato sulla posizione della directory kubelet. Per impostazione predefinita, questa è `/var/vcap/data/kubelet`.

  La specifica della posizione del kubelet utilizzando `--kubelet-dir` è nota per funzionare con Trident Operator, Helm e `tridentctl` deployment.

Installa utilizzando l'operatore Trident

Distribuisci manualmente l'operatore Trident (modalità standard)

È possibile distribuire manualmente l'operatore Trident per installare Trident. Questa procedura si applica alle installazioni in cui le immagini dei container richieste da Trident non sono archiviate in un registro privato. Se si dispone di un registro immagini privato, utilizzare il "processo per la distribuzione offline".

Informazioni critiche su Trident 26.02

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.35 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Distribuire manualmente l'operatore Trident e installare Trident

Verificate "la panoramica dell'installazione" per assicurarvi di aver soddisfatto i prerequisiti di installazione e di aver selezionato l'opzione di installazione corretta per il vostro ambiente.

Prima di iniziare

Prima di iniziare l'installazione, accedi all'host Linux e verifica che stia gestendo un "cluster Kubernetes supportato" funzionante e che tu disponga dei privilegi necessari.

Con OpenShift, utilizzare `oc` invece di `kubectl` in tutti gli esempi che seguono e accedere prima come **system:admin** eseguendo `oc login -u system:admin` o `oc login -u kube-admin`.

1. Verifica la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verifica di poter avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il tuo sistema storage tramite la rete del pod:

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

Passaggio 1: Scarica il pacchetto di installazione di Trident

Il pacchetto di installazione Trident contiene tutto ciò che serve per distribuire l'operatore Trident e installare Trident. Scarica ed estrai la versione più recente del programma di installazione di Trident da "la sezione *Assets* su GitHub".

```
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
```

Passaggio 2: crea la `TridentOrchestrator` CRD

Crea la `TridentOrchestrator` Definizione di Risorsa Personalizzata (CRD). Creerai una `TridentOrchestrator` Risorsa Personalizzata in seguito. Utilizza la versione YAML appropriata del CRD in `deploy/crds` per creare la `TridentOrchestrator` CRD.

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

Fase 3: distribuisci l'operatore Trident

Il programma di installazione di Trident fornisce un file bundle che può essere usato per installare l'operatore e creare oggetti associati. Il file bundle è un modo semplice per distribuire l'operatore e installare Trident usando una configurazione predefinita.

- Per i cluster che eseguono Kubernetes 1.24, usa `bundle_pre_1_25.yaml`.
- Per i cluster che eseguono Kubernetes 1.25 o versioni successive, utilizzare `bundle_post_1_25.yaml`.

Prima di iniziare

- Per impostazione predefinita, il programma di installazione di Trident distribuisce l'operatore nel `trident` namespace. Se il `trident` namespace non esiste, crearlo utilizzando:

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

- Per distribuire l'operatore in uno spazio dei nomi diverso dal `trident` namespace, aggiornare `serviceaccount.yaml`, `clusterrolebinding.yaml` e `operator.yaml` e generare il file bundle usando il `kustomization.yaml`.

  a. Crea `kustomization.yaml` utilizzando il seguente comando dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` in base alla tua versione di Kubernetes.

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

  b. Compila il bundle usando il seguente comando, dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` in base alla tua versione di Kubernetes.

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

Passaggi

1. Crea le risorse e distribuisci l'operatore:

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

2. Verificare che l'operatore, il deployment e i replicasets siano stati creati.

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

> Dovrebbe esserci solo **un'istanza** dell'operatore in un cluster Kubernetes. Non creare distribuzioni multiple dell'operatore Trident.

Passaggio 4: crea `TridentOrchestrator` e installa Trident

Ora puoi creare il `TridentOrchestrator` e installare Trident. Facoltativamente, puoi "personalizza la tua installazione Trident" utilizzando gli attributi nella `TridentOrchestrator` spec.

```
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
```

Verificare l'installazione

Esistono diversi modi per verificare l'installazione.

Utilizzo `TridentOrchestrator` status

Lo stato di `TridentOrchestrator` indica se l'installazione è riuscita e visualizza la versione di Trident installata. Durante l'installazione, lo stato di `TridentOrchestrator` cambia da `Installing` a `Installed`. Se si osserva lo stato `Failed` e l'operatore non è in grado di ripristinarsi autonomamente, "controlla i registri".

Stato	Descrizione
Installazione	L'operatore sta installando Trident utilizzando questo `TridentOrchestrator` CR.
Installato	Trident è stato installato correttamente.
Disinstallazione	L'operatore sta disinstallando Trident, perché `spec.uninstall=true`.
Disinstallato	Trident è disinstallato.
Fallito	L'operatore non è riuscito a installare, applicare patch, aggiornare o disinstallare Trident; l'operatore tenterà automaticamente di recuperare da questo stato. Se questo stato persiste, sarà necessaria la risoluzione dei problemi.
Aggiornamento	L'operatore sta aggiornando un'installazione esistente.
Errore	The `TridentOrchestrator` non è utilizzato. Ne esiste già un altro.

Utilizzo dello stato di creazione del pod

Puoi confermare se l'installazione di Trident è stata completata esaminando lo stato dei pod creati:

```
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
```

Utilizzando `tridentctl`

Puoi usare `tridentctl` per verificare la versione di Trident installata.

```
./tridentctl -n trident version

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

Distribuisci manualmente l'operatore Trident (modalità offline)

È possibile distribuire manualmente l'operatore Trident per installare Trident. Questa procedura si applica alle installazioni in cui le immagini dei container richieste da Trident sono archiviate in un registro privato. Se non si dispone di un registro immagini privato, utilizzare il "processo per la distribuzione standard".

Informazioni critiche su Trident

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.35 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Distribuire manualmente l'operatore Trident e installare Trident

Verificate "la panoramica dell'installazione" per assicurarvi di aver soddisfatto i prerequisiti di installazione e di aver selezionato l'opzione di installazione corretta per il vostro ambiente.

Prima di iniziare

Accedi all'host Linux e verifica che stia gestendo un sistema operativo funzionante "cluster Kubernetes supportato" e che tu disponga dei privilegi necessari.

> Con OpenShift, utilizzare `oc` invece di `kubectl` in tutti gli esempi che seguono e accedere prima come **system:admin** eseguendo `oc login -u system:admin` o `oc login -u kube-admin`.

1. Verifica la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verifica di poter avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il tuo sistema storage tramite la rete del pod:

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

Passaggio 1: Scarica il pacchetto di installazione di Trident

Il pacchetto di installazione Trident contiene tutto ciò che serve per distribuire l'operatore Trident e installare Trident. Scarica ed estrai la versione più recente del programma di installazione di Trident da "la sezione *Assets* su GitHub".

```
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
```

Passaggio 2: crea la `TridentOrchestrator` CRD

Crea la `TridentOrchestrator` Definizione di Risorsa Personalizzata (CRD). Creerai una `TridentOrchestrator` Custom Resources più tardi. Usa la versione YAML appropriata della CRD in `deploy/crds` per creare la `TridentOrchestrator` CRD:

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

Passaggio 3: Aggiorna la posizione del registro nell'operatore

In `/deploy/operator.yaml`, aggiorna `image: docker.io/netapp/trident-operator:26.02.0` per riflettere la posizione del tuo registro delle immagini. Le tue "Trident e immagini CSI" possono essere localizzate in un unico registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Ad esempio:

- `image: <your-registry>/trident-operator:26.02.0` se tutte le immagini si trovano in un unico registro.

- `image: <your-registry>/netapp/trident-operator:26.02.0` se la tua immagine Trident si trova in un registro diverso rispetto alle tue immagini CSI.

Fase 4: distribuisci l'operatore Trident

Il programma di installazione di Trident fornisce un file bundle che può essere usato per installare l'operatore e creare oggetti associati. Il file bundle è un modo semplice per distribuire l'operatore e installare Trident usando una configurazione predefinita.

- Per i cluster che eseguono Kubernetes 1.24, usa `bundle_pre_1_25.yaml`.
- Per i cluster che eseguono Kubernetes 1.25 o versioni successive, utilizzare `bundle_post_1_25.yaml`.

Prima di iniziare

- Per impostazione predefinita, il programma di installazione di Trident distribuisce l'operatore nel `trident` namespace. Se il `trident` namespace non esiste, crearlo utilizzando:

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

- Per distribuire l'operatore in uno spazio dei nomi diverso dal `trident` namespace, aggiornare `serviceaccount.yaml`, `clusterrolebinding.yaml` e `operator.yaml` e generare il file bundle usando il `kustomization.yaml`.

  a. Crea `kustomization.yaml` utilizzando il seguente comando dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` in base alla tua versione di Kubernetes.

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

  b. Compila il bundle usando il seguente comando, dove *<bundle.yaml>* è `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` in base alla tua versione di Kubernetes.

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

Passaggi

1. Crea le risorse e distribuisci l'operatore:

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

2. Verificare che l'operatore, il deployment e i replicasets siano stati creati.

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

Dovrebbe esserci solo **un'istanza** dell'operatore in un cluster Kubernetes. Non creare distribuzioni multiple dell'operatore Trident.

Fase 5: Aggiorna la posizione del registro delle immagini nel `TridentOrchestrator`

Le tue "Trident e immagini CSI" possono essere localizzate in un unico registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Aggiorna `deploy/crds/tridentorchestrator_cr.yaml` per aggiungere le specifiche di posizione aggiuntive in base alla configurazione del tuo registro.

Immagini in un registry

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

Immagini in diversi registri

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

Passaggio 6: crea `TridentOrchestrator` e installa Trident

Ora puoi creare il `TridentOrchestrator` e installare Trident. Facoltativamente, puoi "personalizza la tua installazione Trident"ulteriormente utilizzando gli attributi nella `TridentOrchestrator` specifica. Il seguente esempio mostra un'installazione in cui le immagini Trident e CSI si trovano in registri diversi.

```
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
```

Verificare l'installazione

Esistono diversi modi per verificare l'installazione.

Utilizzo `TridentOrchestrator` status

Lo stato di `TridentOrchestrator` indica se l'installazione è riuscita e visualizza la versione di Trident installata. Durante l'installazione, lo stato di `TridentOrchestrator` cambia da `Installing` a `Installed`. Se si osserva lo stato `Failed` e l'operatore non è in grado di ripristinarsi autonomamente, "controlla i registri".

Stato	Descrizione
Installazione	L'operatore sta installando Trident utilizzando questo `TridentOrchestrator` CR.
Installato	Trident è stato installato correttamente.
Disinstallazione	L'operatore sta disinstallando Trident, perché `spec.uninstall=true`.
Disinstallato	Trident è disinstallato.
Fallito	L'operatore non è riuscito a installare, applicare patch, aggiornare o disinstallare Trident; l'operatore tenterà automaticamente di recuperare da questo stato. Se questo stato persiste, sarà necessaria la risoluzione dei problemi.
Aggiornamento	L'operatore sta aggiornando un'installazione esistente.
Errore	The `TridentOrchestrator` non è utilizzato. Ne esiste già un altro.

Utilizzo dello stato di creazione del pod

Puoi confermare se l'installazione di Trident è stata completata esaminando lo stato dei pod creati:

```
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
```

Utilizzando `tridentctl`

Puoi usare `tridentctl` per verificare la versione di Trident installata.

```
./tridentctl -n trident version

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

Distribuisci l'operatore Trident tramite Helm (modalità standard)

È possibile distribuire l'operatore Trident e installare Trident utilizzando Helm. Questa procedura si applica alle installazioni in cui le immagini dei container richieste da Trident non sono archiviate in un registro privato. Se si dispone di un registro immagini privato, utilizzare il "processo per la distribuzione offline".

Informazioni critiche su Trident 25.10

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.35 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Distribuire l'operatore Trident e installare Trident utilizzando Helm

Utilizzando il Trident "Helm Chart" puoi distribuire l'operatore Trident e installare Trident in un unico passaggio.

Verificate "la panoramica dell'installazione" per assicurarvi di aver soddisfatto i prerequisiti di installazione e di aver selezionato l'opzione di installazione corretta per il vostro ambiente.

Prima di iniziare

Oltre al "prerequisiti di deployment" è necessario "Helm versione 3".

Passaggi

1. Aggiungi il repository Trident Helm:

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

2. Usa `helm install` e specifica un nome per la distribuzione come nel seguente esempio in cui `100..0` è la versione di Trident che stai installando.

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

> Se hai già creato uno spazio dei nomi per Trident, il parametro `--create-namespace` non creerà uno spazio dei nomi aggiuntivo.

È possibile utilizzare `helm list` per esaminare i dettagli dell'installazione, come nome, namespace, chart, stato, versione dell'app e numero di revisione.

Passare i dati di configurazione durante l'installazione

Esistono due modi per passare i dati di configurazione durante l'installazione:

Opzione	Descrizione
`--values` (o `-f`)	Specificare un file YAML con le sovrascritture. Questo può essere specificato più volte e il file più a destra avrà la precedenza.
`--set`	Specificare le sovrascritture sulla riga di comando.

Ad esempio, per modificare il valore predefinito di `debug`, eseguire il seguente comando dove `100.2602.0` è la versione di Trident che si sta installando:

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

Opzioni di configurazione

Questa tabella e il file `values.yaml`, che fa parte dell'Helm chart, forniscono l'elenco delle chiavi e i loro valori predefiniti.

Opzione	Descrizione	Predefinito
`nodeSelector`	Etichette dei nodi per l'assegnazione dei pod	
`podAnnotations`	Annotazioni del pod	

Opzione	Descrizione	Predefinito
`deploymentAnnotations`	Annotazioni di distribuzione	
`tolerations`	Tolleranze per l'assegnazione dei pod	

`affinity`	Affinità per l'assegnazione dei pod	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		Non rimuovere l'affinità predefinita dal file values.yaml. Quando vuoi fornire un'affinità personalizzata, estendi l'affinità predefinita.

`tridentControllerPluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Consultare Comprendere i pod controller e i pod nodo per i dettagli.	
`tridentControllerPluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Consultare Comprendere i pod controller e i pod nodo per i dettagli.	
`tridentNodePluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Consultare Comprendere i pod controller e i pod nodo per i dettagli.	

Opzione	Descrizione	Predefinito
`tridentNodePluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Consultare Comprendere i pod controller e i pod nodo per i dettagli.	
`imageRegistry`	Identifica il registro per le `trident-operator`, `trident` e altre immagini. Lascia vuoto per accettare l'impostazione predefinita. IMPORTANTE: Quando si installa Trident in un repository privato, se si utilizza l'opzione `imageRegistry` per specificare la posizione del repository, non usare `/netapp/` nel percorso del repository.	`""`
`imagePullPolicy`	Imposta il criterio di pull dell'immagine per il `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Imposta i segreti di estrazione delle immagini per le `trident-operator`, `trident` e altre immagini.	
`kubeletDir`	Consente di sovrascrivere la posizione host dello stato interno di kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Consente di impostare il livello di log dell'operatore Trident su: `trace`, `debug`, `info`, `warn`, `error` o `fatal`.	`"info"`
`operatorDebug`	Consente di impostare il livello di log dell'operatore Trident su debug.	`true`
`operatorImage`	Consente la completa sostituzione dell'immagine per `trident-operator`.	`""`
`operatorImageTag`	Permette di sovrascrivere il tag dell' `trident-operator` immagine.	`""`
`tridentIPv6`	Consente di abilitare Trident per funzionare in cluster IPv6.	`false`
`tridentK8sTimeout`	Sostituisce il timeout predefinito di 30 secondi per la maggior parte delle operazioni API di Kubernetes (se diverso da zero, in secondi).	`0`

Opzione	Descrizione	Predefinito
`tridentHttpRequestTimeout`	Sovrascrive il timeout predefinito di 90 secondi per le richieste HTTP, con `0s` che rappresenta una durata infinita per il timeout. Non sono ammessi valori negativi.	`"90s"`
`tridentSilenceAutosupport`	Consente di disabilitare la segnalazione periodica AutoSupport di Trident.	`false`
`tridentAutosupportImageTag`	Consente di sovrascrivere il tag dell'immagine per il container Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Consente al container Trident AutoSupport di telefonare a casa tramite un proxy HTTP.	`""`
`tridentLogFormat`	Imposta il formato di logging Trident (`text` o `json`).	`"text"`
`tridentDisableAuditLog`	Disattiva il logger di audit di Trident.	`true`
`tridentLogLevel`	Consente di impostare il livello di log di Trident su: `trace`, `debug`, `info`, `warn`, `error`, o `fatal`.	`"info"`
`tridentDebug`	Consente di impostare il livello di log di Trident su `debug`. È possibile automatizzare il processo di distacco forzato tramite integrazione con node health check (NHC) operator. Per informazioni, vedi "Automatizzare il failover delle applicazioni stateful con Trident".	`false`
`tridentLogWorkflows`	Consente di abilitare flussi di lavoro Trident specifici per la registrazione delle tracce o la soppressione dei log.	`""`
`tridentLogLayers`	Consente di abilitare specifici livelli Trident per la registrazione delle tracce o la soppressione dei log.	`""`
`tridentImage`	Consente la completa sostituzione dell'immagine per Trident.	`""`
`tridentImageTag`	Consente di sovrascrivere il tag dell'immagine per Trident.	`""`
`tridentProbePort`	Consente di sovrascrivere la porta predefinita utilizzata per le sonde liveness/readiness di Kubernetes.	`""`

Opzione	Descrizione	Predefinito
`windows`	Consente l'installazione di Trident sul nodo worker di Windows.	`false`
`enableForceDetach`	Consente di abilitare la funzione force detach.	`false`
`excludePodSecurityPolicy`	Esclude la pod security policy dell'operatore dalla creazione.	`false`
`cloudProvider`	Impostare su `"Azure"` quando si utilizzano identità gestite o un'identità cloud su un cluster AKS. Impostare su "AWS" quando si utilizza un'identità cloud su un cluster EKS.	`""`
`cloudIdentity`	Impostare su workload identity ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") quando si utilizza l'identità cloud su un cluster AKS. Impostare su AWS IAM role ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'") quando si utilizza l'identità cloud su un cluster EKS.	`""`
`iscsiSelfHealingInterval`	L'intervallo al quale viene invocata la funzione di auto-riparazione iSCSI.	`5m0s`
`iscsiSelfHealingWaitTime`	La durata dopo la quale la funzione di auto-riparazione iSCSI avvia un tentativo di risolvere una sessione non aggiornata eseguendo un logout e un successivo login.	`7m0s`
`nodePrep`	Consente a Trident di preparare i nodi del cluster Kubernetes per gestire i volumi utilizzando il protocollo storage specificato. **Attualmente, `iscsi` è l'unico valore supportato.** NOTA: a partire da OpenShift 4.19, la versione minima di Trident supportata per questa funzionalità è la 25.06.1.	

Opzione	Descrizione	Predefinito
`enableConcurrency`	Consente operazioni simultanee del controller Trident per un throughput migliorato. **Anteprima tecnica**: questa funzionalità è sperimentale e attualmente supporta flussi di lavoro paralleli limitati con i driver ONTAP-NAS (solo NFS) e ONTAP-SAN (NVMe per unified ONTAP 9), oltre all'anteprima tecnica esistente per il driver ONTAP-SAN (protocolli iSCSI e FCP in unified ONTAP 9).	falso
`k8sAPIQPS`	Limite di query al secondo (QPS) utilizzato dal controller durante la comunicazione con il server API Kubernetes. Il valore Burst viene impostato automaticamente in base al valore QPS.	`100`; facoltativo

Opzione	Descrizione	Predefinito

`resources`	Imposta i limiti delle risorse Kubernetes e le richieste per i pod del controller, del nodo e dell'operatore Trident. Puoi configurare CPU e memoria per ogni container e sidecar per gestire l'allocazione delle risorse in Kubernetes.
	
	Per ulteriori informazioni sulla configurazione delle richieste e dei limiti delle risorse, fare riferimento a "Gestione delle risorse per pod e container".
	
	• NON modificare i nomi di alcun container o campo.
	• NON modificare l'indentazione - l'indentazione YAML è fondamentale per una corretta analisi.
		<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>

Opzione	Descrizione	Predefinito
`httpsMetrics`	Abilita HTTPS per l'endpoint delle metriche Prometheus.	falso
`hostNetwork`	Abilita la rete host per il controller Trident. Questa funzionalità è utile quando si desidera separare il traffico frontend e backend in una rete multi-home.	falso

Comprendere i pod controller e i pod nodo

Trident viene eseguito come un singolo pod controller, più un pod nodo su ciascun nodo worker del cluster. Il pod nodo deve essere in esecuzione su qualsiasi host in cui si desidera potenzialmente montare un volume Trident.

Kubernetes "selettori di nodi" e "tolleranze e taint" sono utilizzati per vincolare un pod a essere eseguito su un nodo specifico o preferito. Utilizzando `ControllerPlugin` e `NodePlugin`, puoi specificare vincoli e override.

- Il plugin del controller gestisce il provisioning e la gestione dei volumi, come snapshot e ridimensionamento.
- Il plugin del nodo gestisce il collegamento dello storage al nodo.

Distribuire l'operatore Trident utilizzando Helm (Offline mode)

È possibile distribuire l'operatore Trident e installare Trident utilizzando Helm. Questa procedura si applica alle installazioni in cui le immagini dei container richieste da Trident sono archiviate in un registro privato. Se non si dispone di un registro immagini privato, utilizzare il "processo per la distribuzione standard".

Informazioni critiche su Trident 26.02

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.35 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Distribuire l'operatore Trident e installare Trident utilizzando Helm

Utilizzando il Trident "Helm Chart" puoi distribuire l'operatore Trident e installare Trident in un unico passaggio.

Verificate "la panoramica dell'installazione" per assicurarvi di aver soddisfatto i prerequisiti di installazione e di aver selezionato l'opzione di installazione corretta per il vostro ambiente.

Prima di iniziare

Oltre al "prerequisiti di deployment" è necessario "Helm versione 3".

> Quando si installa Trident in un repository privato, se si utilizza l'opzione `imageRegistry` per specificare la posizione del repository, non utilizzare `/netapp/` nel percorso del repository.

Passaggi

1. Aggiungi il repository Trident Helm:

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

2. Usa `helm install` e specifica un nome per il deployment e la posizione del registro delle immagini. Le tue "Trident e immagini CSI" possono essere localizzate in un unico registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Negli esempi, `100.2602.0` è la versione di Trident che stai installando.

Immagini in un registry

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

Immagini in diversi registri

```
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}
```

> Se hai già creato uno spazio dei nomi per Trident, il parametro `--create-namespace` non creerà uno spazio dei nomi aggiuntivo.

È possibile utilizzare `helm list` per esaminare i dettagli dell'installazione, come nome, namespace, chart, stato, versione dell'app e numero di revisione.

Passare i dati di configurazione durante l'installazione

Esistono due modi per passare i dati di configurazione durante l'installazione:

Opzione	Descrizione
`--values` (o `-f`)	Specificare un file YAML con le sovrascritture. Questo può essere specificato più volte e il file più a destra avrà la precedenza.
`--set`	Specificare le sovrascritture sulla riga di comando.

Ad esempio, per modificare il valore predefinito di `debug`, eseguire il seguente comando dove `100.2602.0` è la versione di Trident che si sta installando:

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

Per aggiungere il valore nodePrep, eseguire il seguente comando:

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

Opzioni di configurazione

Questa tabella e il file `values.yaml`, che fa parte dell'Helm chart, forniscono l'elenco delle chiavi e i loro valori predefiniti.

> Non rimuovere l'affinità predefinita dal file values.yaml. Quando vuoi fornire un'affinità personalizzata, estendi l'affinità predefinita.

Opzione	Descrizione	Predefinito
`nodeSelector`	Etichette dei nodi per l'assegnazione dei pod	
`podAnnotations`	Annotazioni del pod	
`deploymentAnnotations`	Annotazioni di distribuzione	
`tolerations`	Tolleranze per l'assegnazione dei pod	

Opzione	Descrizione	Predefinito

`affinity`	Affinità per l'assegnazione dei pod	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		Non rimuovere l'affinità predefinita dal file values.yaml. Quando vuoi fornire un'affinità personalizzata, estendi l'affinità predefinita.

`tridentControllerPluginNod` `eSelector`	Selettori di nodo aggiuntivi per i pod. Consultare "Comprendere i pod controller e i pod nodo" per i dettagli.	
`tridentControllerPluginTol` `erations`	Sostituisce le tolleranze di Kubernetes per i pod. Consultare "Comprendere i pod controller e i pod nodo" per i dettagli.	

Opzione	Descrizione	Predefinito
`tridentNodePluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Consultare "Comprendere i pod controller e i pod nodo" per i dettagli.	
`tridentNodePluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Consultare "Comprendere i pod controller e i pod nodo" per i dettagli.	
`imageRegistry`	Identifica il registro per le `trident-operator`, `trident` e altre immagini. Lascia vuoto per accettare l'impostazione predefinita. IMPORTANTE: Quando si installa Trident in un repository privato, se si utilizza l'opzione `imageRegistry` per specificare la posizione del repository, non usare `/netapp/` nel percorso del repository.	""
`imagePullPolicy`	Imposta il criterio di pull dell'immagine per il `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Imposta i segreti di estrazione delle immagini per le `trident-operator`, `trident` e altre immagini.	
`kubeletDir`	Consente di sovrascrivere la posizione host dello stato interno di kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Consente di impostare il livello di log dell'operatore Trident su: `trace`, `debug`, `info`, `warn`, `error` o `fatal`.	`"info"`
`operatorDebug`	Consente di impostare il livello di log dell'operatore Trident su debug.	`true`
`operatorImage`	Consente la completa sostituzione dell'immagine per `trident-operator`.	""
`operatorImageTag`	Permette di sovrascrivere il tag dell' `trident-operator` immagine.	""
`tridentIPv6`	Consente di abilitare Trident per funzionare in cluster IPv6.	`false`

Opzione	Descrizione	Predefinito
`tridentK8sTimeout`	Sostituisce il timeout predefinito di 180 secondi per la maggior parte delle operazioni API di Kubernetes (se diverso da zero, in secondi). Il `tridentK8sTimeout` parametro è applicabile solo per l'installazione Trident.	`180`
`tridentHttpRequestTimeout`	Sovrascrive il timeout predefinito di 90 secondi per le richieste HTTP, con `0s` che rappresenta una durata infinita per il timeout. Non sono ammessi valori negativi.	`"90s"`
`tridentSilenceAutosupport`	Consente di disabilitare la segnalazione periodica AutoSupport di Trident.	`false`
`tridentAutosupportImageTag`	Consente di sovrascrivere il tag dell'immagine per il container Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Consente al container Trident AutoSupport di telefonare a casa tramite un proxy HTTP.	""
`tridentLogFormat`	Imposta il formato di logging Trident (`text` o `json`).	`"text"`
`tridentDisableAuditLog`	Disattiva il logger di audit di Trident.	`true`
`tridentLogLevel`	Consente di impostare il livello di log di Trident su: `trace`, `debug`, `info`, `warn`, `error`, o `fatal`.	`"info"`
`tridentDebug`	Consente di impostare il livello di log di Trident su `debug`.	`false`
`tridentLogWorkflows`	Consente di abilitare flussi di lavoro Trident specifici per la registrazione delle tracce o la soppressione dei log.	""
`tridentLogLayers`	Consente di abilitare specifici livelli Trident per la registrazione delle tracce o la soppressione dei log.	""
`tridentImage`	Consente la completa sostituzione dell'immagine per Trident.	""
`tridentImageTag`	Consente di sovrascrivere il tag dell'immagine per Trident.	""

Opzione	Descrizione	Predefinito
`tridentProbePort`	Consente di sovrascrivere la porta predefinita utilizzata per le sonde liveness/readiness di Kubernetes.	""
`windows`	Consente l'installazione di Trident sul nodo worker di Windows.	`false`
`enableForceDetach`	Consente di abilitare la funzione force detach. È possibile automatizzare il processo di distacco forzato tramite integrazione con node health check (NHC) operator. Per informazioni, vedi "Automatizzare il failover delle applicazioni stateful con Trident".	`false`
`excludePodSecurityPolicy`	Esclude la pod security policy dell'operatore dalla creazione.	`false`
`nodePrep`	Consente a Trident di preparare i nodi del cluster Kubernetes per gestire i volumi utilizzando il protocollo storage specificato. **Attualmente, `iscsi` è l'unico valore supportato.** A partire da OpenShift 4.19, la versione minima di Trident supportata per questa funzionalità è la 25.06.1.	

Opzione	Descrizione	Predefinito

`resources`	Imposta i limiti delle risorse Kubernetes e le richieste per i pod del controller, del nodo e dell'operatore Trident. Puoi configurare CPU e memoria per ogni container e sidecar per gestire l'allocazione delle risorse in Kubernetes.
	
	Per ulteriori informazioni sulla configurazione delle richieste e dei limiti delle risorse, fare riferimento a "Gestione delle risorse per pod e container".
	
	• NON modificare i nomi di alcun container o campo.
	• NON modificare l'indentazione - l'indentazione YAML è fondamentale per una corretta analisi.
		<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:</pre>

Personalizzare l'installazione dell'operatore Trident

L'operatore Trident permette di personalizzare l'installazione Trident usando gli attributi nella `TridentOrchestrator` spec. Se vuoi personalizzare l'installazione oltre ciò che gli argomenti di `TridentOrchestrator` consentono, considera l'utilizzo di `tridentctl` per generare manifesti YAML personalizzati da modificare secondo necessità.

Comprendere i pod controller e i pod nodo

Trident viene eseguito come un singolo controller pod e un node pod su ogni nodo worker nel cluster. Il pod nodo deve essere in esecuzione su qualsiasi host in cui si desidera potenzialmente montare un volume Trident.

Kubernetes "selettori di nodi" e "tolleranze e taint" sono utilizzati per vincolare un pod a essere eseguito su un nodo specifico o preferito. Utilizzando `ControllerPlugin` e `NodePlugin`, puoi specificare vincoli e override.

- Il plugin del controller gestisce il provisioning e la gestione dei volumi, come snapshot e ridimensionamento.
- Il plugin del nodo gestisce il collegamento dello storage al nodo.

Opzioni di configurazione

> **!** `spec.namespace` è specificato in `TridentOrchestrator` per indicare il namespace in cui Trident è installato. Questo parametro **non può essere aggiornato dopo che Trident è installato**. Il tentativo di farlo fa sì che lo stato `TridentOrchestrator` cambi in `Failed`. Trident non è destinato a essere migrato tra namespace.

Questa tabella illustra `TridentOrchestrator` gli attributi.

Parametro	Descrizione	Predefinito
`namespace`	Spazio dei nomi in cui installare Trident	`"default"`
`debug`	Abilita il debug per Trident	`false`
`enableForceDetach`	`ontap-san`, `ontap-san-economy`, `ontap-nas` e `ontap-nas-economy` solo. Funziona con Kubernetes Non-Graceful Node Shutdown (NGNS) per garantire agli amministratori cluster la possibilità di migrare in modo sicuro i carichi di lavoro con volumi montati su nuovi nodi nel caso in cui un nodo diventi non sano. Per informazioni, vedi "Automatizzare il failover delle applicazioni stateful con Trident".	`false`
`windows`	L'impostazione su `true` abilita l'installazione sui nodi worker Windows.	`false`

Parametro	Descrizione	Predefinito
`cloudProvider`	Impostare su `"Azure"` quando si utilizzano identità gestite o un'identità cloud su un cluster AKS. Impostare su `"AWS"` quando si utilizza una cloud identity su un cluster EKS. Impostare su `"GCP"` quando si utilizza una cloud identity su un cluster GKE.	`""`
`cloudIdentity`	Impostare su workload identity ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") quando si utilizza l'identità cloud su un cluster AKS. Impostare su AWS IAM role ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'") quando si utilizza l'identità cloud su un cluster EKS. Impostare su cloud identity ("'iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com'") quando si utilizza cloud identity su un cluster GKE.	`""`
`IPv6`	Install Trident su IPv6	falso
`k8sTimeout`	Timeout per le operazioni Kubernetes. Il `k8sTimeout` parametro è applicabile solo per l'installazione Trident.	`180sec`
`silenceAutosupport`	Non inviare automaticamente i bundle di autosupport a NetApp	`false`
`autosupportImage`	L'immagine container per Autosupport Telemetry	`"netapp/trident-autosupport10"`
`autosupportProxy`	L'indirizzo/porta di un proxy per l'invio della Telemetria di Autosupport	`"http://proxy.example.com:8888"`
`uninstall`	Un flag utilizzato per disinstallare Trident	`false`
`logFormat`	Formato di registrazione Trident da utilizzare [text,json]	`"text"`
`tridentImage`	Trident image da installare	`"netapp/trident:26.02"`
`imageRegistry`	Percorso al registro interno, del formato `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`
`kubeletDir`	Percorso della directory kubelet sull'host	`"/var/lib/kubelet"`
`wipeout`	Un elenco di risorse da eliminare per eseguire una rimozione completa di Trident	
`imagePullSecrets`	Segreti per prelevare immagini da un registro interno	

Parametro	Descrizione	Predefinito
`imagePullPolicy`	Imposta la policy di pull dell'immagine per il Trident operator. I valori validi sono: `Always` per estrarre sempre l'immagine. `IfNotPresent` per estrarre l'immagine solo se non esiste già sul nodo. `Never` per non estrarre mai l'immagine.	`IfNotPresent`
`controllerPluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Segue lo stesso formato come `pod.spec.nodeSelector`.	Nessun valore predefinito; facoltativo
`controllerPluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Segue lo stesso formato di `pod.spec.Tolerations`.	Nessun valore predefinito; facoltativo
`nodePluginNodeSelector`	Selettori di nodo aggiuntivi per i pod. Segue lo stesso formato come `pod.spec.nodeSelector`.	Nessun valore predefinito; facoltativo
`nodePluginTolerations`	Sostituisce le tolleranze di Kubernetes per i pod. Segue lo stesso formato di `pod.spec.Tolerations`.	Nessun valore predefinito; facoltativo
`nodePrep`	Consente a Trident di preparare i nodi del cluster Kubernetes per gestire i volumi utilizzando il protocollo storage specificato. **Attualmente, `iscsi` è l'unico valore supportato.** A partire da OpenShift 4.19, la versione minima di Trident supportata per questa funzionalità è la 25.06.1.	
`k8sAPIQPS`	Limite di query al secondo (QPS) utilizzato dal controller durante la comunicazione con il server API Kubernetes. Il valore Burst viene impostato automaticamente in base al valore QPS.	`100`; facoltativo
`enableConcurrency`	Consente operazioni simultanee del controller Trident per un throughput migliorato. **Anteprima tecnica**: questa funzionalità è sperimentale e attualmente supporta flussi di lavoro paralleli limitati con i driver ONTAP-NAS (solo NFS) e ONTAP-SAN (NVMe per unified ONTAP 9), oltre all'anteprima tecnica esistente per il driver ONTAP-SAN (protocolli iSCSI e FCP in unified ONTAP 9).	falso

Parametro	Descrizione	Predefinito
`resources`	Imposta i limiti delle risorse Kubernetes e le richieste per il controller Trident e i pod dei nodi. Puoi configurare CPU e memoria per ogni container e sidecar per gestire l'allocazione delle risorse in Kubernetes. Per ulteriori informazioni sulla configurazione delle richieste e dei limiti delle risorse, fare riferimento a "Gestione delle risorse per pod e container". • NON modificare i nomi di alcun container o campo. • NON modificare l'indentazione - l'indentazione YAML è fondamentale per una corretta analisi. • Per impostazione predefinita non vengono applicati limiti: solo le richieste hanno valori predefiniti e sono applicate automaticamente se non specificate. • I nomi dei container sono elencati così come appaiono nelle specifiche del pod. • I sidecar sono elencati sotto ogni container principale. • Controllare il campo `status.CurrentInstallationParams` TORC per visualizzare i valori attualmente applicati.	`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:`

Parametro	Descrizione	Predefinito
`httpsMetrics`	Abilita HTTPS per l'endpoint delle metriche Prometheus.	falso
`hostNetwork`	Abilita la rete host per il controller Trident. Questa funzionalità è utile quando si desidera separare il traffico frontend e backend in una rete multi-home.	falso

Per ulteriori informazioni sulla formattazione dei parametri del pod, consulta "Assegnazione dei pod ai nodi".

Configurazioni di esempio

È possibile utilizzare gli attributi in Opzioni di configurazione quando si definisce `TridentOrchestrator` per personalizzare l'installazione.

Configurazione personalizzata di base

Questo esempio, creato dopo l'esecuzione del comando `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml`, rappresenta un'installazione personalizzata di base:

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

Selettori di nodo

Questo esempio installa Trident con node selectors.

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

memory:
liveness-

Nodi worker Windows

Questo esempio, creato dopo aver eseguito il `cat deploy/crds/tridentorchestrator_cr.yaml` comando, installa Trident su un nodo worker Windows.

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

Identità gestite su un cluster AKS

Questo esempio installa Trident per abilitare le identità gestite su un cluster AKS.

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

Identità cloud su un cluster AKS

Questo esempio installa Trident per l'uso con un'identità cloud su un cluster AKS.

```
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'
```

Identità cloud su un cluster EKS

Questo esempio installa Trident per l'uso con un'identità cloud su un cluster AKS.

```
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'"
```

Identità cloud per GKE

Questo esempio installa Trident per l'uso con un'identità cloud su un cluster GKE.

```
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
```

Configurazione delle richieste di risorse Kubernetes e dei limiti per il controller Trident e i pod dei nodi Linux Trident

Questo esempio configura le richieste e i limiti di risorse Kubernetes per il controller Trident e i pod Trident del nodo Linux.

Disclaimer: I valori di richiesta e limite forniti in questo esempio sono solo a scopo dimostrativo. Adatta questi valori in base al tuo ambiente e ai requisiti del carico di lavoro.

```
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
```

Configurazione delle richieste di risorse Kubernetes e dei limiti per il controller Trident e per i pod dei nodi Trident Windows e Linux

Questo esempio configura le richieste e i limiti di risorse Kubernetes per il controller Trident e i pod Trident dei nodi Windows e Linux.

Disclaimer: I valori di richiesta e limite forniti in questo esempio sono solo a scopo dimostrativo. Adatta questi valori in base al tuo ambiente e ai requisiti del carico di lavoro.

```
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
```

Installare utilizzando tridentctl

Installare utilizzando tridentctl

È possibile installare Trident usando `tridentctl`. Questa procedura si applica alle installazioni in cui le immagini dei container richieste da Trident sono archiviate sia in un registro privato che no. Per personalizzare la tua `tridentctl` distribuzione, fare riferimento a "Personalizza la distribuzione di tridentctl".

Informazioni critiche su Trident10

È necessario leggere le seguenti informazioni critiche su Trident.

Informazioni critiche su Trident

- Kubernetes 1.27 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Trident impone rigorosamente l'uso della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  L'utilizzo di una configurazione non multipath o l'utilizzo di `find_multipaths: yes` o `find_multipaths: smart` nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'utilizzo di `find_multipaths: no` dalla release 21.07.

Installa Trident usando `tridentctl`

Verificate "la panoramica dell'installazione" per assicurarvi di aver soddisfatto i prerequisiti di installazione e di aver selezionato l'opzione di installazione corretta per il vostro ambiente.

Prima di iniziare

Prima di iniziare l'installazione, accedi all'host Linux e verifica che stia gestendo un "cluster Kubernetes supportato" funzionante e che tu disponga dei privilegi necessari.

Con OpenShift, utilizzare `oc` invece di `kubectl` in tutti gli esempi che seguono e accedere prima come **system:admin** eseguendo `oc login -u system:admin` o `oc login -u kube-admin`.

1. Verifica la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verifica di poter avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il tuo sistema storage tramite la rete del pod:

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

Passaggio 1: Scarica il pacchetto di installazione di Trident

Il pacchetto di installazione di Trident crea un pod Trident, configura gli oggetti CRD utilizzati per mantenerne lo stato e inizializza i sidecar CSI per eseguire azioni come il provisioning e il collegamento di volumi agli host del cluster. Scarica ed estrai la versione più recente del programma di installazione di Trident da "la sezione *Assets* su GitHub". Aggiorna *<trident-installer-XX.XX.X.tar.gz>* nell'esempio con la versione di Trident selezionata.

```
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
```

Passaggio 2: installa Trident

Installa Trident nello spazio dei nomi desiderato eseguendo il `tridentctl install` comando. Puoi aggiungere argomenti aggiuntivi per specificare la posizione del registro delle immagini.

Modalità standard

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

Immagini in un registry

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident
-image <your-registry>/trident:26.02.0
```

Immagini in diversi registri

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident
-image <your-registry>/trident:26.02.0
```

Lo stato dell'installazione dovrebbe essere simile a questo.

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

Verificare l'installazione

Puoi verificare la tua installazione utilizzando lo stato di creazione del pod o `tridentctl`.

Utilizzo dello stato di creazione del pod

Puoi confermare se l'installazione di Trident è stata completata esaminando lo stato dei pod creati:

```
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
```

Se il programma di installazione non viene completato correttamente o `trident-controller-<generated id>` (`trident-csi-<generated id>` nelle versioni precedenti alla 23.01) non presenta lo stato **Running**, la piattaforma non è stata installata. Usa `-d` per "attiva la modalità debug" e risolvere il problema.

Utilizzando `tridentctl`

Puoi usare `tridentctl` per verificare la versione di Trident installata.

```
./tridentctl -n trident version

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

Configurazioni di esempio

Gli esempi seguenti forniscono configurazioni di esempio per l'installazione di Trident usando `tridentctl`.

Nodi Windows

Per abilitare Trident sui nodi Windows:

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

Forza distacco

Per informazioni, vedi "Automatizzare il failover delle applicazioni stateful con Trident".

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

Abilita le operazioni simultanee del controller Trident

Per abilitare operazioni simultanee del controller Trident per migliorare il throughput, aggiungere l'opzione `--enable-concurrency` durante l'installazione come mostrato in questo esempio.

> **Anteprima tecnica**: questa funzionalità è sperimentale e attualmente supporta flussi di lavoro paralleli limitati con i driver ONTAP-NAS (solo NFS) e ONTAP-SAN (NVMe per unified ONTAP 9), oltre all'anteprima tecnica esistente per il driver ONTAP-SAN (protocolli iSCSI e FCP in unified ONTAP 9).

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

Personalizza l'installazione di tridentctl

È possibile utilizzare il programma di installazione Trident per personalizzare l'installazione.

Informazioni sul programma di installazione

Il programma di installazione Trident consente di personalizzare gli attributi. Ad esempio, se hai copiato l'immagine Trident in un repository privato, puoi specificare il nome dell'immagine usando `--trident-image`. Se hai copiato l'immagine Trident e anche le immagini CSI sidecar necessarie in un repository privato, potrebbe essere preferibile specificare la posizione di quel repository usando lo switch `--image-registry`, che assume la forma `<registry FQDN>[:port]`.

> Quando si installa Trident in un repository privato, se si utilizza l'opzione `--image-registry` per specificare la posizione del repository, non utilizzare `/netapp/` nel percorso del repository. Ad esempio: `./tridentctl install --image-registry <image-registry> -n <namespace>`

Se si utilizza una distribuzione di Kubernetes, dove `kubelet` conserva i dati su un percorso diverso dal solito `/var/lib/kubelet`, è possibile specificare il percorso alternativo utilizzando `--kubelet-dir`.

Se si desidera personalizzare l'installazione al di là di quanto consentito dagli argomenti del programma di installazione, è possibile personalizzare anche i file di deployment. L'utilizzo del parametro `--generate-custom-yaml` crea i seguenti file YAML nella directory del programma di installazione `setup`:

- `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` *

Dopo aver generato questi file, puoi modificarli secondo le tue esigenze e poi usare `--use-custom-yaml` per installare la tua distribuzione personalizzata.

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

Installa utilizzando l'operatore certificato OpenShift

Installare Trident utilizzando OpenShift OperatorHub

Se si utilizza Red Hat OpenShift, è possibile installare NetApp Trident utilizzando l'operatore certificato Red Hat. Utilizzare questa procedura per installare Trident dalla piattaforma Red Hat OpenShift Container Platform.

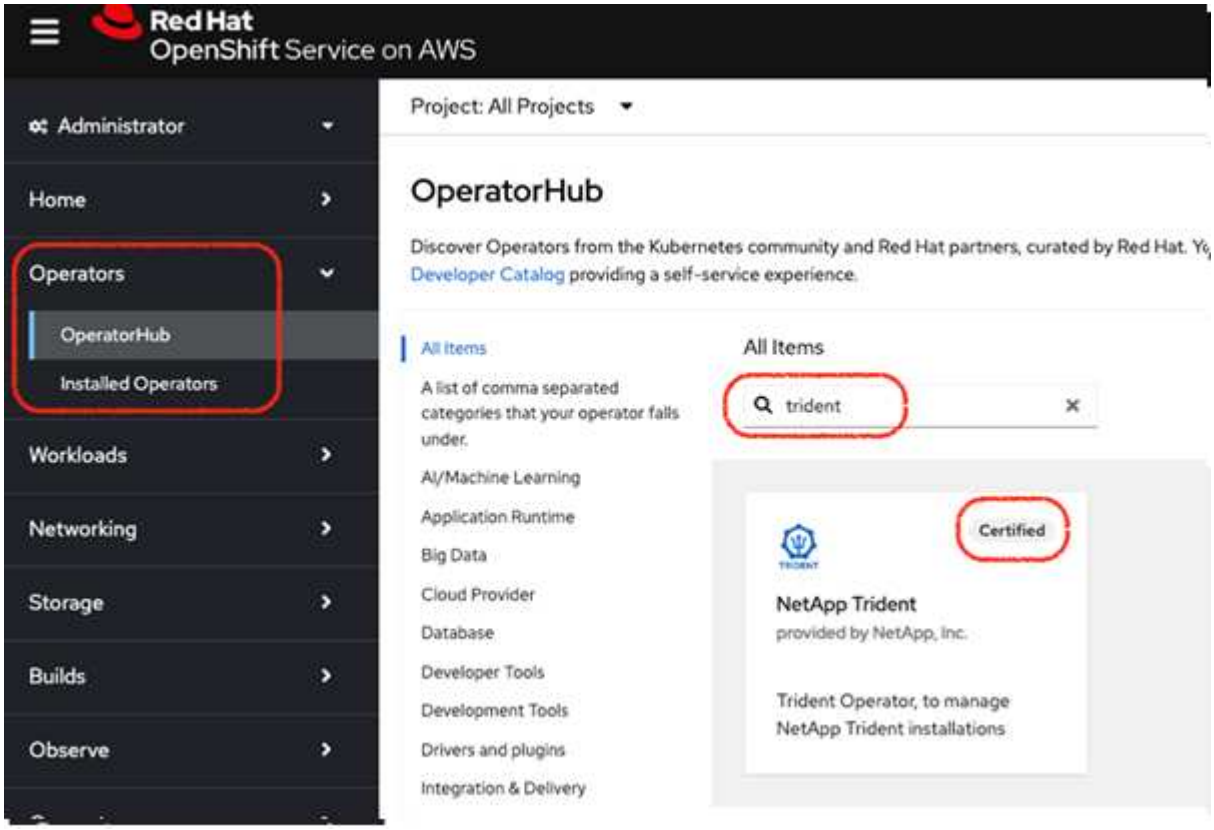
Prima di iniziare

Prima di iniziare l'installazione, "prepara il tuo ambiente per l'installazione di Trident".

Trova e installa l'operatore Trident

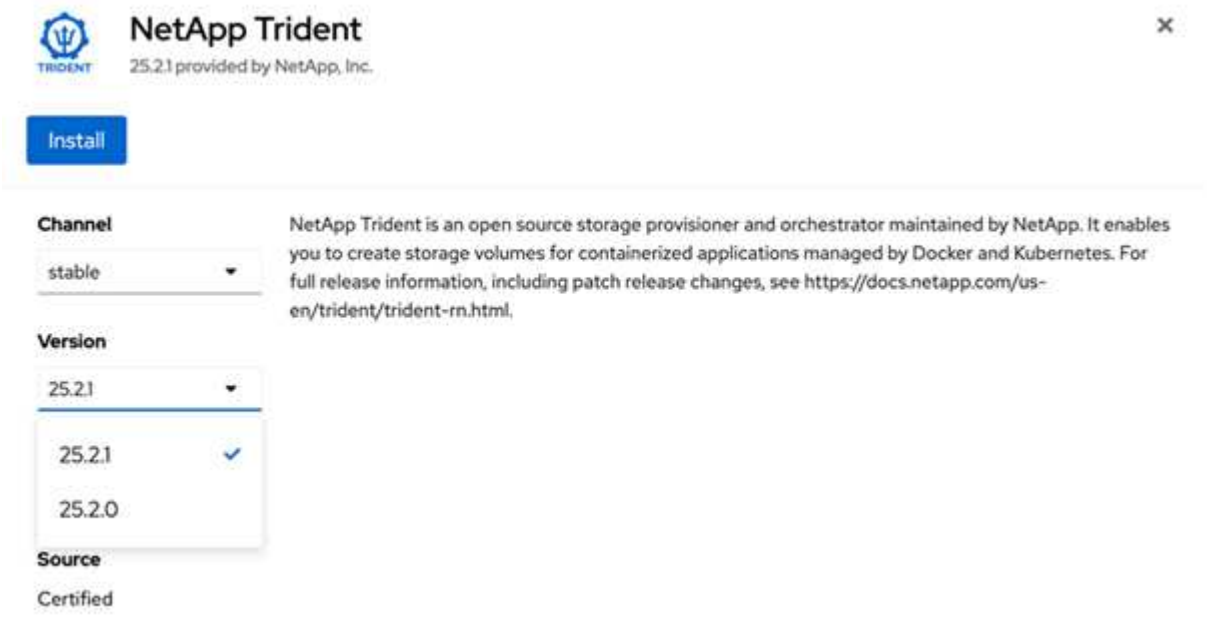
Passaggi

1. Naviga su OpenShift OperatorHub e cerca NetApp Trident.



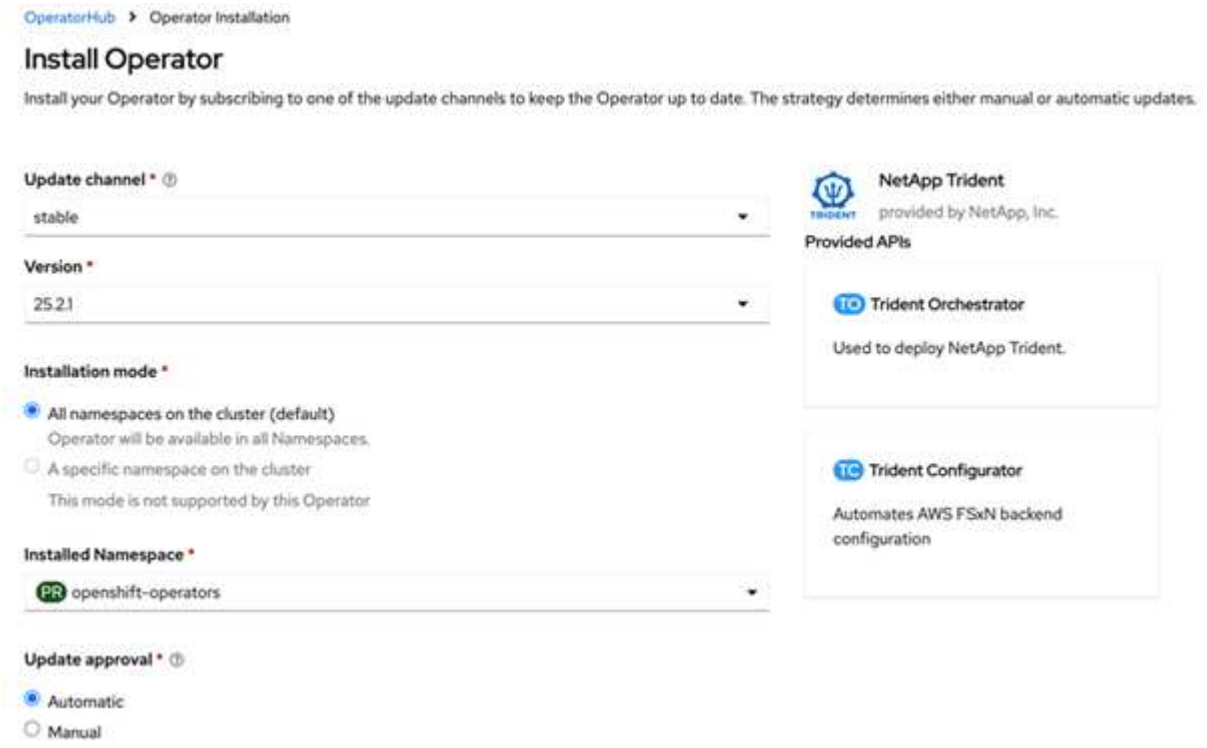
2. Fare clic su **NetApp Trident** per aprire le impostazioni di installazione.

3. Seleziona le opzioni richieste e fai clic su **Install** per aprire la configurazione dell'Operator.

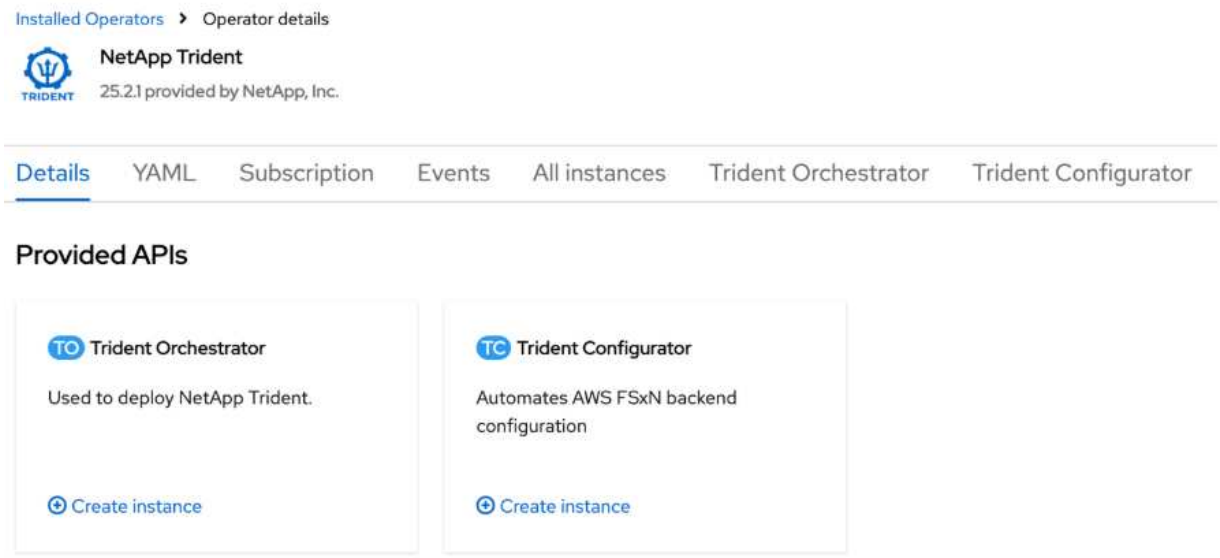


Assicurati di selezionare la versione più recente di Operator.

4. Mantieni tutti i parametri così come sono e fai clic su **Install**.



5. Fare clic su **View Operator** per visualizzare i dettagli dell'Operator.



6. Fare clic su **Vista YAML** e incollare quanto segue nel modulo:

```
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
```

L'interfaccia utente fornisce un esempio predefinito. È possibile modificarlo direttamente anziché copiare una configurazione completa.

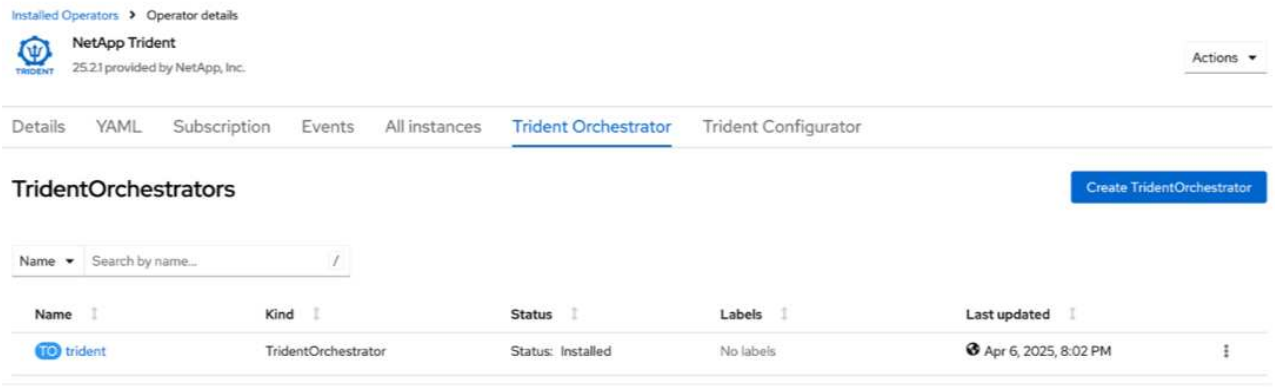
Opzionale: Abilita la concorrenza

Per abilitare la concorrenza, aggiungere il seguente campo alla specifica:

```
enableConcurrency: true
```

- Red Hat Enterprise Linux CoreOS (RHCOS) non ha iSCSI abilitato e configurato.
- È possibile aggiungere il `nodePrep` parametro per configurare e abilitare entrambi i servizi iSCSI e Multipath su tutti i nodi worker OpenShift.
- A partire da OpenShift 4.19, la versione minima di Trident supportata per questa funzionalità è la 25.06.1.

1. Fare clic su **Create**; il Trident Orchestrator sarà completamente installato.



Disinstallare l'operatore Trident

Passaggi

1. Seleziona l'operatore Trident dall'elenco degli operator installati.
2. Seleziona se vuoi eliminare tutte le istanze dell'operando dall'operatore.

Se non si seleziona la casella di controllo **Elimina tutte le istanze dell'operando da questo operatore**, Trident non verrà disinstallato.

3. Fare clic su **Disinstalla**.

Passa all'operatore Trident certificato OpenShift

È possibile passare all'operatore Trident certificato Red Hat OpenShift dall'operatore della community, da un'installazione basata su Helm o da un operatore distribuito manualmente. La procedura per ciascun metodo prevede la disinstallazione dell'operatore esistente e la successiva installazione dell'operatore certificato tramite OperatorHub.

Prima di iniziare

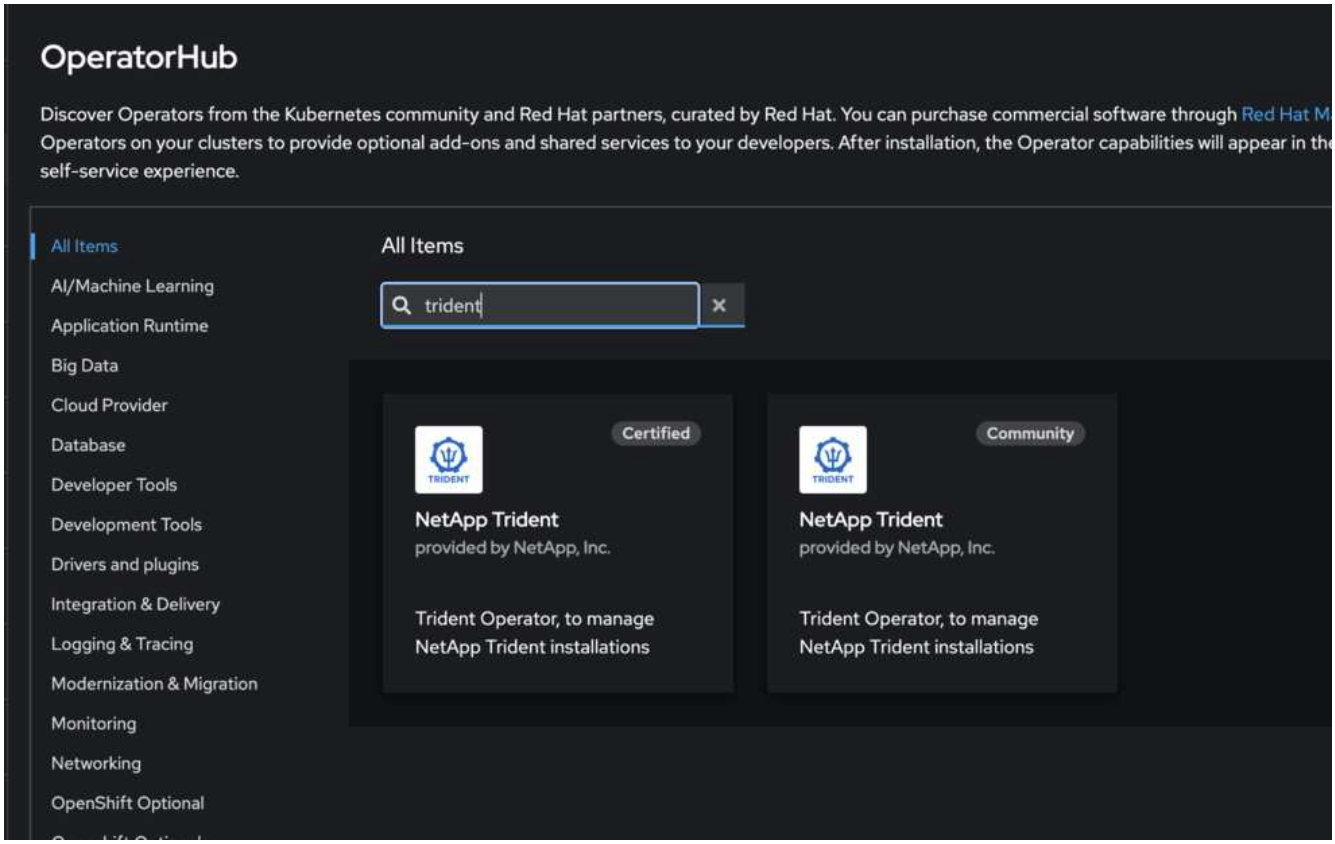
Prima di iniziare l'installazione, "prepara il tuo ambiente per l'installazione di Trident".

Non eliminare la `TridentOrchestrator` risorsa personalizzata (CR) durante il processo di disinstallazione. La `TridentOrchestrator` CR conserva la configurazione del backend e della classe di archiviazione, necessaria dopo l'installazione dell'operatore certificato.

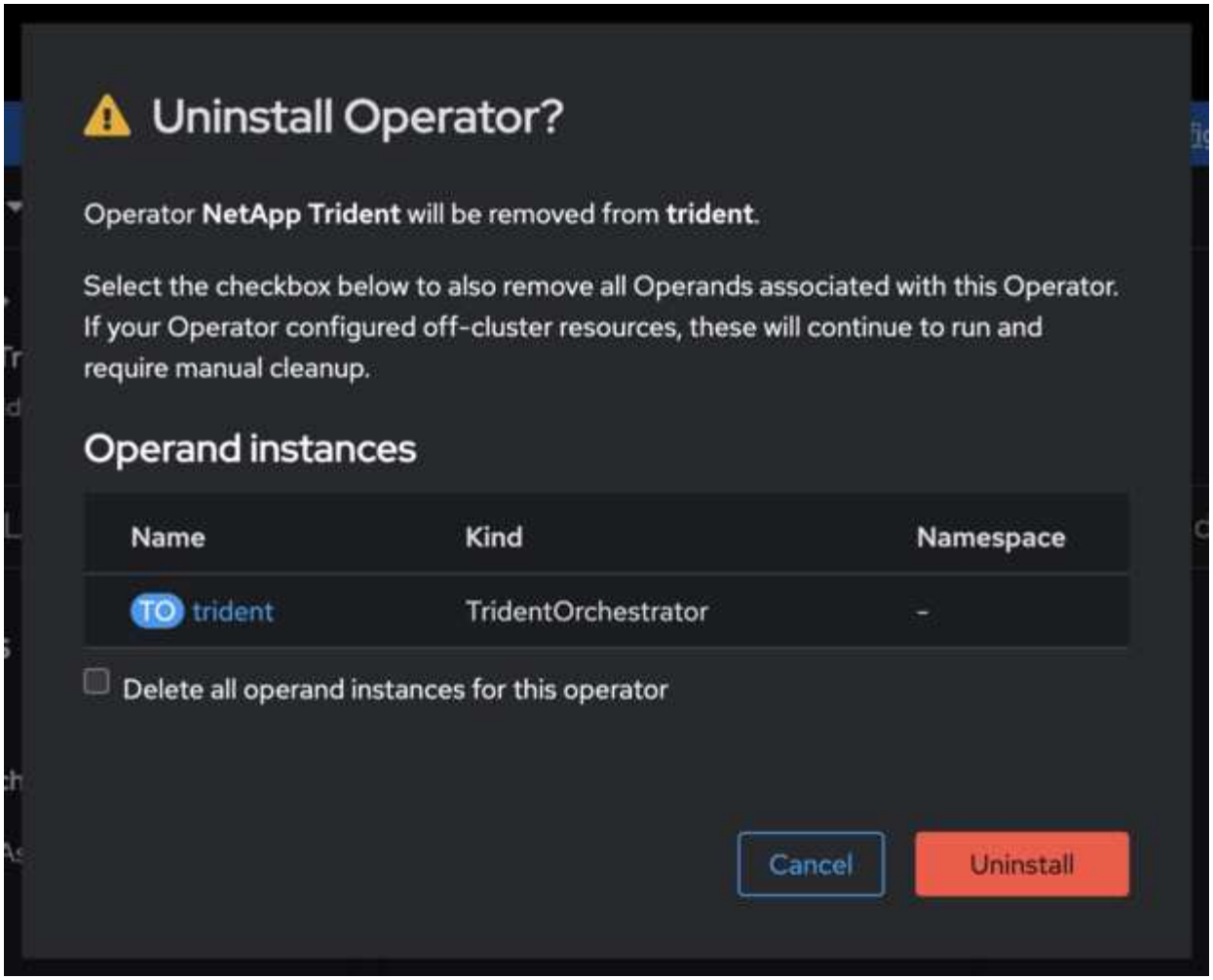
Passa dall'operatore della community

Passaggi

1. Utilizzare la console OpenShift per navigare verso la OperatorHub.



2. Trova l'operatore della comunità NetApp Trident.



Non selezionare **Elimina tutte le istanze dell'operando da questo operatore**.

3. Fare clic su **Disinstalla**.
4. Dopo il completamento della disinstallazione, procedere a Installa l'operatore certificato OpenShift.

Passa da un'installazione dell'operatore basata su Helm

Passaggi

1. Elenca la release di Helm per la tua installazione di Trident:

```
helm ls -n trident
```

2. Disinstalla la release Helm:

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

3. Dopo il completamento della disinstallazione, procedere a Installa l'operatore certificato OpenShift.

Passa da un operatore distribuito manualmente

Se hai installato Trident distribuendo manualmente l'operatore utilizzando un `bundle.yaml` dal pacchetto di installazione, rimuovilo eliminando lo stesso manifesto.

Passaggi

1. Elimina la distribuzione dell'operatore utilizzando il manifesto del bundle:

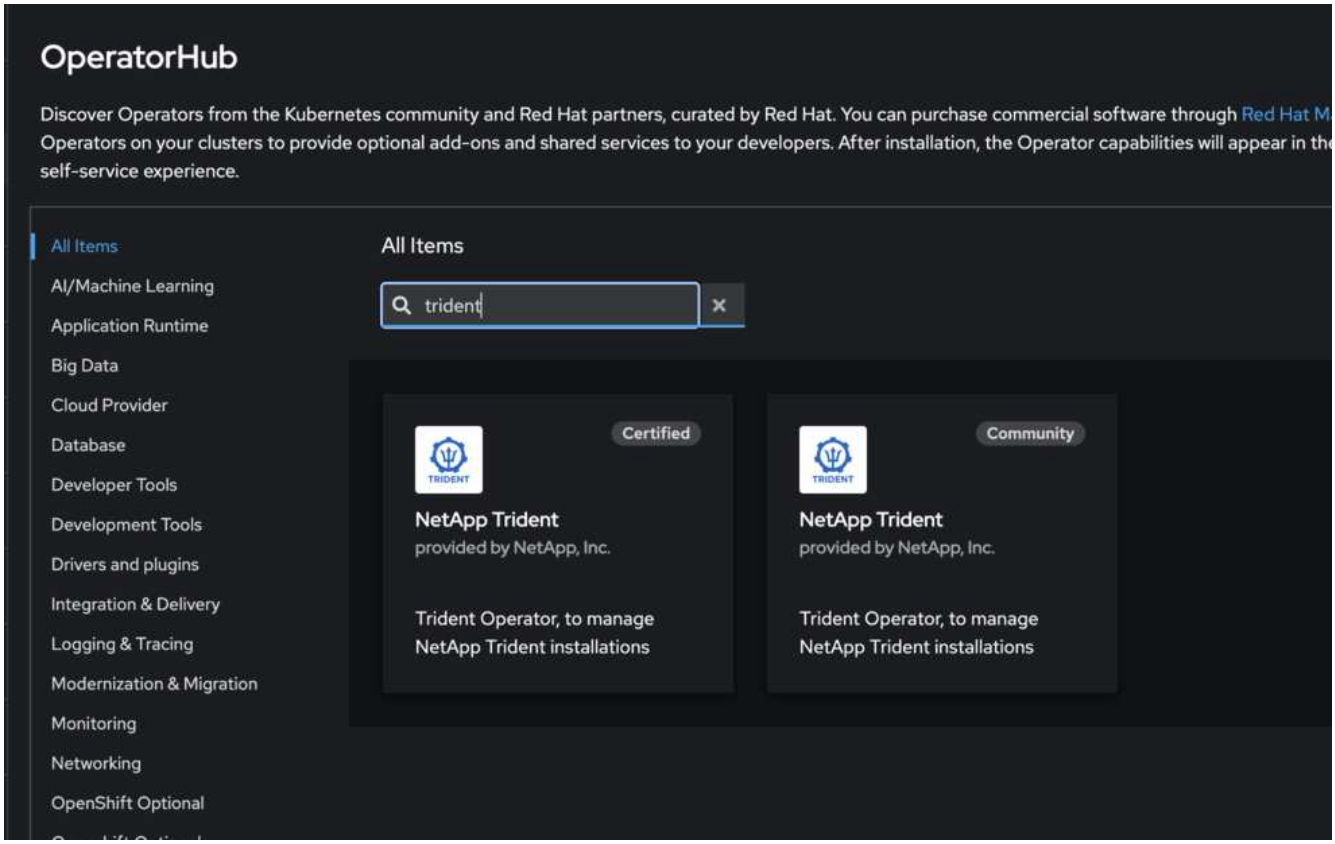
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. Dopo il completamento della disinstallazione, procedere a Installa l'operatore certificato OpenShift.

Installa l'operatore certificato OpenShift

Passaggi

1. Accedere a Red Hat OperatorHub.
2. Cerca e seleziona l'operatore NetApp Trident.



3. Seguire le istruzioni sullo schermo per installare l'operatore.

Verifica

- Controlla OperatorHub nella console per assicurarti che il nuovo operatore certificato sia stato installato correttamente.

Informazioni sul copyright

Informazioni sul marchio commerciale