Installare utilizzando l'operatore Trident

Astra Trident

NetApp
April 16, 2024

Sommario

Installare utilizzando l'operatore Trident

Implementare manualmente l'operatore Trident (modalità standard)

È possibile implementare manualmente l'operatore Trident per installare Astra Trident. Questo processo si applica alle installazioni in cui le immagini container richieste da Astra Trident non sono memorizzate in un registro privato. Se si dispone di un registro di immagini privato, utilizzare "processo per l'implementazione offline".

Informazioni critiche su Astra Trident 23.01

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1.26 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Astra Trident impone rigorosamente l'utilizzo della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  Utilizzo di configurazioni o utilizzo non multipathing di `find_multipaths: yes` oppure `find_multipaths: smart` il valore nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'uso di `find_multipaths: no` dalla release 21.07.

Implementare manualmente l'operatore Trident e installare Trident

Revisione "panoramica dell'installazione" per assicurarsi di aver soddisfatto i prerequisiti di installazione e selezionato l'opzione di installazione corretta per il proprio ambiente.

Prima di iniziare

Prima di iniziare l'installazione, accedere all'host Linux e verificare che stia gestendo un "Cluster Kubernetes supportato" e che si dispone dei privilegi necessari.

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

1. Verificare la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verificare che sia possibile avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il sistema di storage tramite la rete pod:

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

Fase 1: Scaricare il pacchetto di installazione di Trident

Il pacchetto di installazione di Astra Trident contiene tutto il necessario per implementare l'operatore Trident e installare Astra Trident. Scaricare ed estrarre la versione più recente del programma di installazione Trident da "La sezione *Assets* su GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

Fase 2: Creare `TridentOrchestrator` CRD

Creare il `TridentOrchestrator` Definizione personalizzata delle risorse (CRD). Verrà creato un `TridentOrchestrator` Risorse personalizzate in un secondo momento. Utilizzare la versione CRD YAML appropriata in `deploy/crds` per creare `TridentOrchestrator` CRD.

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

Fase 3: Implementare l'operatore Trident

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

- Per i cluster che eseguono Kubernetes 1.24 o versione precedente, utilizzare `bundle_pre_1_25.yaml`.

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

Il programma di installazione di Trident implementa l'operatore in `trident` namespace. Se il `trident` spazio dei nomi inesistente, utilizzare `kubectl apply -f deploy/namespace.yaml` per crearlo.

Fasi

1. Creare le risorse e implementare l'operatore:

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

> Per implementare l'operatore in uno spazio dei nomi diverso da `trident` namespace, aggiornamento `serviceaccount.yaml`, `clusterrolebinding.yaml` e. `operator.yaml` e generare il file bundle utilizzando `kustomization.yaml`:
>
> ```
> kubectl kustomize deploy/ > deploy/<bundle>.yaml
> ```

2. Verificare che l'operatore sia stato implementato.

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

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

> In un cluster Kubernetes dovrebbe esserci solo **un'istanza** dell'operatore. Non creare implementazioni multiple dell'operatore Trident.

Fase 4: Creare `TridentOrchestrator` E installare Trident

Ora è possibile creare `TridentOrchestrator` E installare Astra Trident. Se lo si desidera, è possibile "Personalizzare l'installazione di Trident" utilizzando gli attributi in `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
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:23.01
    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:23.01.1
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.01.1
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 di `TridentOrchestrator` stato

Lo stato di `TridentOrchestrator` Indica se l'installazione ha avuto esito positivo e visualizza la versione di Trident installata. Durante l'installazione, lo stato di `TridentOrchestrator` modifiche da `Installing` a. `Installed`. Se si osserva `Failed` e l'operatore non è in grado di ripristinarsi da solo, "controllare i registri".

Stato	Descrizione
Installazione in corso	L'operatore sta installando Astra Trident `TridentOrchestrator` CR.
Installato	Astra Trident è stato installato correttamente.
Disinstallazione in corso	L'operatore sta disinstallando Astra Trident, perché `spec.uninstall=true`.
Disinstallato	Astra Trident disinstallato.
Non riuscito	L'operatore non ha potuto installare, applicare patch, aggiornare o disinstallare Astra Trident; l'operatore tenterà automaticamente di eseguire il ripristino da questo stato. Se lo stato persiste, è necessario eseguire la risoluzione dei problemi.
Aggiornamento in corso	L'operatore sta aggiornando un'installazione esistente.
Errore	Il `TridentOrchestrator` non viene utilizzato. Un'altra esiste già.

Utilizzo dello stato di creazione del pod

È possibile verificare se l'installazione di Astra 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
```

Utilizzo di `tridentctl`

È possibile utilizzare `tridentctl` Per verificare la versione di Astra Trident installata.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

Cosa succederà

Ora puoi "creare un backend e una classe di storage, eseguire il provisioning di un volume e montare il volume in un pod".

Implementare manualmente l'operatore Trident (modalità offline)

È possibile implementare manualmente l'operatore Trident per installare Astra Trident. Questo processo si applica alle installazioni in cui le immagini container richieste da Astra Trident sono memorizzate in un registro privato. Se non si dispone di un registro di immagini privato, utilizzare "processo per l'implementazione standard".

Informazioni critiche su Astra Trident 23.01

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1.26 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Astra Trident impone rigorosamente l'utilizzo della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  Utilizzo di configurazioni o utilizzo non multipathing di `find_multipaths: yes` oppure `find_multipaths: smart` il valore nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'uso di `find_multipaths: no` dalla release 21.07.

Implementare manualmente l'operatore Trident e installare Trident

Revisione "panoramica dell'installazione" per assicurarsi di aver soddisfatto i prerequisiti di installazione e selezionato l'opzione di installazione corretta per il proprio ambiente.

Prima di iniziare

Accedere all'host Linux e verificare che stia gestendo un'applicazione e. "Cluster Kubernetes supportato" e che si dispone dei privilegi necessari.

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

1. Verificare la versione di Kubernetes:

```
kubectl version
```

2. Verificare i privilegi di amministratore del cluster:

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

3. Verificare che sia possibile avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il sistema di storage tramite la rete pod:

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

Fase 1: Scaricare il pacchetto di installazione di Trident

Il pacchetto di installazione di Astra Trident contiene tutto il necessario per implementare l'operatore Trident e installare Astra Trident. Scaricare ed estrarre la versione più recente del programma di installazione Trident da "La sezione *Assets* su GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

Fase 2: Creare `TridentOrchestrator` CRD

Creare il `TridentOrchestrator` Definizione personalizzata delle risorse (CRD). Verrà creato un `TridentOrchestrator` Risorse personalizzate in un secondo momento. Utilizzare la versione CRD YAML appropriata in `deploy/crds` per creare `TridentOrchestrator` CRD:

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

Fase 3: Aggiornare la posizione del registro nell'operatore

Poll `/deploy/operator.yaml`, aggiornare `image: docker.io/netapp/trident-operator:23.01.1` per riflettere la posizione del registro delle immagini. Il tuo "Immagini Trident e CSI" Può trovarsi in un registro

o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Ad esempio:

- `image: <your-registry>/trident-operator:23.01.1` se tutte le immagini si trovano in un unico registro.
- `image: <your-registry>/netapp/trident-operator:23.01.1` Se l'immagine Trident si trova in un registro diverso dalle immagini CSI.

Fase 4: Implementare l'operatore Trident

Il programma di installazione di Trident implementa l'operatore in `trident` namespace. Se il `trident` spazio dei nomi inesistente, utilizzare `kubectl apply -f deploy/namespace.yaml` per crearlo.

Per implementare l'operatore in uno spazio dei nomi diverso da `trident` namespace, aggiornamento `serviceaccount.yaml`, `clusterrolebinding.yaml` e. `operator.yaml` prima di implementare l'operatore.

1. Creare le risorse e implementare l'operatore:

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

> Il programma di installazione di Astra Trident fornisce un file bundle che può essere utilizzato per installare l'operatore e creare oggetti associati. Il file bundle è un modo semplice per implementare l'operatore e installare Astra Trident utilizzando una configurazione predefinita.
>
> - Per i cluster che eseguono Kubernetes 1.24 o versione precedente, utilizzare `bundle_pre_1_25.yaml`.
> - Per i cluster che eseguono Kubernetes 1.25 o versioni successive, utilizzare `bundle_post_1_25.yaml`.

2. Verificare che l'operatore sia stato implementato.

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

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

> In un cluster Kubernetes dovrebbe esserci solo **un'istanza** dell'operatore. Non creare implementazioni multiple dell'operatore Trident.

Fase 5: Aggiornare la posizione del registro delle immagini in `TridentOrchestrator`

Il tuo "Immagini Trident e CSI" Può trovarsi in un registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Aggiornare `deploy/crds/tridentorchestrator_cr.yaml` per aggiungere le specifiche di posizione aggiuntive in base alla configurazione del registro di sistema.

Immagini in un registro

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:23.01"
tridentImage: "<your-registry>/trident:23.01.1"
```

Immagini in diversi registri

È necessario aggiungere `sig-storage` al `imageRegistry` per utilizzare diverse posizioni del registro di sistema.

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.01"
tridentImage: "<your-registry>/netapp/trident:23.01.1"
```

Fase 6: Creare `TridentOrchestrator` E installare Trident

Ora è possibile creare `TridentOrchestrator` E installare Astra Trident. Se lo si desidera, è possibile fare di più "Personalizzare l'installazione di Trident" utilizzando gli attributi in `TridentOrchestrator` spec. L'esempio seguente mostra un'installazione in cui le immagini Trident e CSI si trovano in diversi registri.

```
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>/netapp/trident-autosupport:23.01
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.01.1
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>/sig-storage
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/netapp/trident:23.01.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.01.1
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 di `TridentOrchestrator` stato

Lo stato di `TridentOrchestrator` Indica se l'installazione ha avuto esito positivo e visualizza la versione di Trident installata. Durante l'installazione, lo stato di `TridentOrchestrator` modifiche da `Installing` a. `Installed`. Se si osserva `Failed` e l'operatore non è in grado di ripristinarsi da solo, "controllare i registri".

Stato	Descrizione
Installazione in corso	L'operatore sta installando Astra Trident `TridentOrchestrator` CR.
Installato	Astra Trident è stato installato correttamente.
Disinstallazione in corso	L'operatore sta disinstallando Astra Trident, perché `spec.uninstall=true`.
Disinstallato	Astra Trident disinstallato.
Non riuscito	L'operatore non ha potuto installare, applicare patch, aggiornare o disinstallare Astra Trident; l'operatore tenterà automaticamente di eseguire il ripristino da questo stato. Se lo stato persiste, è necessario eseguire la risoluzione dei problemi.
Aggiornamento in corso	L'operatore sta aggiornando un'installazione esistente.
Errore	Il `TridentOrchestrator` non viene utilizzato. Un'altra esiste già.

Utilizzo dello stato di creazione del pod

È possibile verificare se l'installazione di Astra 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
```

Utilizzo di `tridentctl`

È possibile utilizzare `tridentctl` Per verificare la versione di Astra Trident installata.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

Cosa succederà

Ora puoi "creare un backend e una classe di storage, eseguire il provisioning di un volume e montare il volume in un pod".

Implementare l'operatore Trident utilizzando Helm (modalità standard)

È possibile implementare l'operatore Trident e installare Astra Trident utilizzando Helm. Questo processo si applica alle installazioni in cui le immagini container richieste da Astra Trident non sono memorizzate in un registro privato. Se si dispone di un registro di immagini privato, utilizzare "processo per l'implementazione offline".

Informazioni critiche su Astra Trident 23.01

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1.26 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Astra Trident impone rigorosamente l'utilizzo della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  Utilizzo di configurazioni o utilizzo non multipathing di `find_multipaths: yes` oppure `find_multipaths: smart` il valore nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'uso di `find_multipaths: no` dalla release 21.07.

Implementare l'operatore Trident e installare Astra Trident utilizzando Helm

Utilizzo di Trident "Grafico di comando" È possibile implementare l'operatore Trident e installare Trident in un'unica fase.

Revisione "panoramica dell'installazione" per assicurarsi di aver soddisfatto i prerequisiti di installazione e selezionato l'opzione di installazione corretta per il proprio ambiente.

Prima di iniziare

Oltre a "prerequisiti per l'implementazione" di cui hai bisogno "Helm versione 3".

Fasi

1. Aggiungere il repository Astra Trident Helm:

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

2. Utilizzare `helm install` e specificare un nome per la distribuzione come nell'esempio seguente dove `23.01.1` È la versione di Astra Trident che si sta installando.

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace <trident-namespace>
```

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

È possibile utilizzare `helm list` per rivedere i dettagli dell'installazione, ad esempio nome, spazio dei nomi, grafico, stato, versione dell'applicazione, 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 override. Questo valore può essere specificato più volte e il file più a destra avrà la precedenza.
`--set`	Specificare le sostituzioni sulla riga di comando.

Ad esempio, per modificare il valore predefinito di `debug`, eseguire quanto segue `--set` comando dove `23.01.1` È la versione di Astra Trident che si sta installando:

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

Opzioni di configurazione

Questa tabella e il `values.yaml` Il file, che fa parte del grafico Helm, fornisce l'elenco delle chiavi e i relativi valori predefiniti.

Opzione	Descrizione	Predefinito
`nodeSelector`	Etichette dei nodi per l'assegnazione dei pod	
`podAnnotations`	Annotazioni Pod	
`deploymentAnnotations`	Annotazioni di implementazione	
`tolerations`	Pedaggi per l'assegnazione del pod	
`affinity`	Affinità per l'assegnazione del pod	
`tridentControllerPluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`tridentControllerPluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`tridentNodePluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`tridentNodePluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`imageRegistry`	Identifica il registro di sistema per `trident-operator`, `trident`e altre immagini. Lasciare vuoto per accettare l'impostazione predefinita.	""
`imagePullPolicy`	Imposta il criterio di pull dell'immagine per `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Imposta i segreti di pull dell'immagine per `trident-operator`, `trident`e altre immagini.	
`kubeletDir`	Consente di ignorare la posizione host dello stato interno del 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`

Opzione	Descrizione	Predefinito
`operatorImage`	Consente di eseguire l'override completo dell'immagine per `trident-operator`.	""
`operatorImageTag`	Consente di sovrascrivere il tag di `trident-operator` immagine.	""
`tridentIPv6`	Consente ad Astra Trident di funzionare nei cluster IPv6.	`false`
`tridentK8sTimeout`	Esegue l'override del timeout predefinito di 30 secondi per la maggior parte delle operazioni API di Kubernetes (se diverso da zero, in secondi).	`0`
`tridentHttpRequestTimeout`	Esegue l'override del timeout predefinito di 90 secondi per le richieste HTTP, con `0s` è una durata infinita per il timeout. Non sono consentiti valori negativi.	`"90s"`
`tridentSilenceAutosupport`	Consente di disattivare il reporting periodico AutoSupport di Astra Trident.	`false`
`tridentAutosupportImageTag`	Consente di ignorare il tag dell'immagine per il contenitore Astra Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Consente al container Astra Trident AutoSupport di telefonare a casa tramite un proxy HTTP.	""
`tridentLogFormat`	Imposta il formato di registrazione di Astra Trident (`text` oppure `json`).	`"text"`
`tridentDisableAuditLog`	Disattiva l'audit logger Astra Trident.	`true`
`tridentLogLevel`	Consente di impostare il livello di log di Astra Trident su: `trace`, `debug`, `info`, `warn`, `error`, o. `fatal`.	`"info"`
`tridentDebug`	Consente di impostare il livello di log di Astra Trident su `debug`.	`false`
`tridentLogWorkflows`	Consente di attivare specifici flussi di lavoro di Astra Trident per la registrazione delle tracce o la soppressione dei log.	""

Opzione	Descrizione	Predefinito
`tridentLogLayers`	Consente di attivare specifici livelli Astra Trident per la registrazione delle tracce o la soppressione dei log.	""
`tridentImage`	Consente l'override completo dell'immagine per Astra Trident.	""
`tridentImageTag`	Consente di ignorare il tag dell'immagine per Astra Trident.	""
`tridentProbePort`	Consente di ignorare la porta predefinita utilizzata per le sonde liveness/readiness Kubernetes.	""
`windows`	Consente di installare Astra Trident sul nodo di lavoro Windows.	`false`
`enableForceDetach`	Consente di attivare la funzione di distacco forzato.	`false`
`excludePodSecurityPolicy`	Esclude la creazione della policy di sicurezza del pod operatore.	`false`

Comprensione dei pod controller e dei pod di nodi

Astra Trident viene eseguito come singolo pod controller, più un pod di nodi su ciascun nodo di lavoro nel cluster. Il pod nodo deve essere in esecuzione su qualsiasi host in cui si desidera montare un volume Astra Trident.

Kubernetes "selettori di nodi" e. "tollerazioni e contamini" vengono utilizzati per vincolare l'esecuzione di un pod su un nodo specifico o preferito. Utilizzo di`ControllerPlugin` e. `NodePlugin`, è possibile specificare vincoli e override.

- Il plug-in del controller gestisce il provisioning e la gestione dei volumi, ad esempio snapshot e ridimensionamento.
- Il plug-in del nodo gestisce il collegamento dello storage al nodo.

Cosa succederà

Ora puoi "creare un backend e una classe di storage, eseguire il provisioning di un volume e montare il volume in un pod".

Implementare l'operatore Trident utilizzando Helm (modalità offline)

È possibile implementare l'operatore Trident e installare Astra Trident utilizzando Helm. Questo processo si applica alle installazioni in cui le immagini container richieste da Astra Trident sono memorizzate in un registro privato. Se non si dispone di un registro di immagini privato, utilizzare "processo per l'implementazione standard".

Informazioni critiche su Astra Trident 23.01

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1.26 è ora supportato in Trident. Aggiornare Trident prima di aggiornare Kubernetes.
- Astra Trident impone rigorosamente l'utilizzo della configurazione multipathing negli ambienti SAN, con un valore consigliato di `find_multipaths: no` nel file multipath.conf.

  Utilizzo di configurazioni o utilizzo non multipathing di `find_multipaths: yes` oppure `find_multipaths: smart` il valore nel file multipath.conf causerà errori di montaggio. Trident ha raccomandato l'uso di `find_multipaths: no` dalla release 21.07.

Implementare l'operatore Trident e installare Astra Trident utilizzando Helm

Utilizzo di Trident "Grafico di comando" È possibile implementare l'operatore Trident e installare Trident in un'unica fase.

Revisione "panoramica dell'installazione" per assicurarsi di aver soddisfatto i prerequisiti di installazione e selezionato l'opzione di installazione corretta per il proprio ambiente.

Prima di iniziare

Oltre a "prerequisiti per l'implementazione" di cui hai bisogno "Helm versione 3".

Fasi

1. Aggiungere il repository Astra Trident Helm:

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

2. Utilizzare `helm install` e specificare un nome per la distribuzione e la posizione del registro delle immagini. Il tuo "Immagini Trident e CSI" Può trovarsi in un registro o in registri diversi, ma tutte le immagini CSI devono trovarsi nello stesso registro. Negli esempi, `23.01.1` È la versione di Astra Trident che si sta installando.

Immagini in un registro

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace>
```

Immagini in diversi registri

È necessario aggiungere `sig-storage` al `imageRegistry` per utilizzare diverse posizioni del registro di sistema.

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.01.1 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.01 --set tridentImage=<your-
registry>/netapp/trident:23.01.1 --create-namespace --namespace
<trident-namespace>
```

Se è già stato creato uno spazio dei nomi per Trident, il `--create-namespace` il parametro non crea uno spazio dei nomi aggiuntivo.

È possibile utilizzare `helm list` per rivedere i dettagli dell'installazione, ad esempio nome, spazio dei nomi, grafico, stato, versione dell'applicazione, 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 override. Questo valore può essere specificato più volte e il file più a destra avrà la precedenza.
`--set`	Specificare le sostituzioni sulla riga di comando.

Ad esempio, per modificare il valore predefinito di `debug`, eseguire quanto segue `--set` comando dove `23.01.1` È la versione di Astra Trident che si sta installando:

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

Opzioni di configurazione

Questa tabella e il `values.yaml` Il file, che fa parte del grafico Helm, fornisce l'elenco delle chiavi e i relativi valori predefiniti.

Opzione	Descrizione	Predefinito
`nodeSelector`	Etichette dei nodi per l'assegnazione dei pod	
`podAnnotations`	Annotazioni Pod	
`deploymentAnnotations`	Annotazioni di implementazione	
`tolerations`	Pedaggi per l'assegnazione del pod	
`affinity`	Affinità per l'assegnazione del pod	
`tridentControllerPluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`tridentControllerPluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`tridentNodePluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`tridentNodePluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Fare riferimento a. Comprensione dei pod controller e dei pod di nodi per ulteriori informazioni.	
`imageRegistry`	Identifica il registro di sistema per `trident-operator`, `trident`e altre immagini. Lasciare vuoto per accettare l'impostazione predefinita.	""
`imagePullPolicy`	Imposta il criterio di pull dell'immagine per `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Imposta i segreti di pull dell'immagine per `trident-operator`, `trident`e altre immagini.	
`kubeletDir`	Consente di ignorare la posizione host dello stato interno del kubelet.	"'/var/lib/kubelet"`

Opzione	Descrizione	Predefinito
`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 di eseguire l'override completo dell'immagine per `trident-operator`.	""
`operatorImageTag`	Consente di sovrascrivere il tag di `trident-operator` immagine.	""
`tridentIPv6`	Consente ad Astra Trident di funzionare nei cluster IPv6.	`false`
`tridentK8sTimeout`	Esegue l'override del timeout predefinito di 30 secondi per la maggior parte delle operazioni API di Kubernetes (se diverso da zero, in secondi).	`0`
`tridentHttpRequestTimeout`	Esegue l'override del timeout predefinito di 90 secondi per le richieste HTTP, con `0s` è una durata infinita per il timeout. Non sono consentiti valori negativi.	`"90s"`
`tridentSilenceAutosupport`	Consente di disattivare il reporting periodico AutoSupport di Astra Trident.	`false`
`tridentAutosupportImageTag`	Consente di ignorare il tag dell'immagine per il contenitore Astra Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Consente al container Astra Trident AutoSupport di telefonare a casa tramite un proxy HTTP.	""
`tridentLogFormat`	Imposta il formato di registrazione di Astra Trident (`text` oppure `json`).	`"text"`
`tridentDisableAuditLog`	Disattiva l'audit logger Astra Trident.	`true`
`tridentLogLevel`	Consente di impostare il livello di log di Astra Trident su: `trace`, `debug`, `info`, `warn`, `error`, o. `fatal`.	`"info"`
`tridentDebug`	Consente di impostare il livello di log di Astra Trident su `debug`.	`false`

Opzione	Descrizione	Predefinito
`tridentLogWorkflows`	Consente di attivare specifici flussi di lavoro di Astra Trident per la registrazione delle tracce o la soppressione dei log.	""
`tridentLogLayers`	Consente di attivare specifici livelli Astra Trident per la registrazione delle tracce o la soppressione dei log.	""
`tridentImage`	Consente l'override completo dell'immagine per Astra Trident.	""
`tridentImageTag`	Consente di ignorare il tag dell'immagine per Astra Trident.	""
`tridentProbePort`	Consente di ignorare la porta predefinita utilizzata per le sonde liveness/readiness Kubernetes.	""
`windows`	Consente di installare Astra Trident sul nodo di lavoro Windows.	`false`
`enableForceDetach`	Consente di attivare la funzione di distacco forzato.	`false`
`excludePodSecurityPolicy`	Esclude la creazione della policy di sicurezza del pod operatore.	`false`

Comprensione dei pod controller e dei pod di nodi

Astra Trident viene eseguito come singolo pod controller, più un pod di nodi su ciascun nodo di lavoro nel cluster. Il pod nodo deve essere in esecuzione su qualsiasi host in cui si desidera montare un volume Astra Trident.

Kubernetes "selettori di nodi" e. "tollerazioni e contamini" vengono utilizzati per vincolare l'esecuzione di un pod su un nodo specifico o preferito. Utilizzo di`ControllerPlugin` e. `NodePlugin`, è possibile specificare vincoli e override.

- Il plug-in del controller gestisce il provisioning e la gestione dei volumi, ad esempio snapshot e ridimensionamento.
- Il plug-in del nodo gestisce il collegamento dello storage al nodo.

Cosa succederà

Ora puoi "creare un backend e una classe di storage, eseguire il provisioning di un volume e montare il volume in un pod".

Personalizzare l'installazione dell'operatore Trident

L'operatore Trident consente di personalizzare l'installazione di Astra Trident utilizzando gli attributi in `TridentOrchestrator` spec. Se si desidera personalizzare l'installazione oltre ciò che si desidera `TridentOrchestrator` gli argomenti lo consentono, valutare

l'utilizzo `tridentctl` Per generare manifesti YAML personalizzati da modificare in base alle necessità.

Comprensione dei pod controller e dei pod di nodi

Astra Trident viene eseguito come singolo pod controller, più un pod di nodi su ciascun nodo di lavoro nel cluster. Il pod nodo deve essere in esecuzione su qualsiasi host in cui si desidera montare un volume Astra Trident.

Kubernetes "selettori di nodi" e. "tollerazioni e contamini" vengono utilizzati per vincolare l'esecuzione di un pod su un nodo specifico o preferito. Utilizzo di`ControllerPlugin` e. `NodePlugin`, è possibile specificare vincoli e override.

- Il plug-in del controller gestisce il provisioning e la gestione dei volumi, ad esempio snapshot e ridimensionamento.
- Il plug-in del nodo gestisce il collegamento dello storage al nodo.

Opzioni di configurazione

> `spec.namespace` è specificato in `TridentOrchestrator` Per indicare lo spazio dei nomi in cui è installato Astra Trident. Questo parametro **non può essere aggiornato dopo l'installazione di Astra Trident**. Il tentativo di eseguire questa operazione causa il `TridentOrchestrator` stato in cui passare `Failed`. Astra Trident non deve essere migrato tra spazi dei nomi.

Questa tabella è dettagliata `TridentOrchestrator` attributi.

Parametro	Descrizione	Predefinito
`namespace`	Spazio dei nomi in cui installare Astra Trident	"predefinito"
`debug`	Attiva il debug per Astra Trident	falso
`windows`	Impostazione su `true` Attiva l'installazione su nodi di lavoro Windows.	falso
`IPv6`	Installare Astra Trident su IPv6	falso
`k8sTimeout`	Timeout per le operazioni Kubernetes	30 sec
`silenceAutosupport`	Non inviare pacchetti AutoSupport automaticamente a NetApp	falso
`enableNodePrep`	Gestire automaticamente le dipendenze dei nodi di lavoro (**BETA**)	falso
`autosupportImage`	L'immagine del contenitore per la telemetria AutoSupport	"netapp/trident-autosupport:23.01"

Parametro	Descrizione	Predefinito
`autosupportProxy`	Indirizzo/porta di un proxy per l'invio di telemetria AutoSupport	"http://proxy.example.com:8888"
`uninstall`	Flag utilizzato per disinstallare Astra Trident	falso
`logFormat`	Formato di registrazione Astra Trident da utilizzare [text,json]	"testo"
`tridentImage`	Immagine Astra Trident da installare	"netapp/trident:21.04"
`imageRegistry`	Percorso al registro interno, del formato `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage (k8s 1.19+) o quay.io/k8scsi"
`kubeletDir`	Percorso della directory del kubelet sull'host	"/var/lib/kubelet"
`wipeout`	Un elenco di risorse da eliminare per eseguire una rimozione completa di Astra Trident	
`imagePullSecrets`	Secrets (segreti) per estrarre immagini da un registro interno	
`imagePullPolicy`	Imposta il criterio di pull dell'immagine per l'operatore Trident. I valori validi sono: `Always` per estrarre sempre l'immagine. `IfNotPresent` per estrarre l'immagine solo se non esiste già nel nodo. `Never` per non tirare mai l'immagine.	`IfNotPresent`
`controllerPluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Segue lo stesso formato di pod.spec.nodeSelector.	Nessuna impostazione predefinita; opzionale
`controllerPluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Segue lo stesso formato di pod.spec.Tolerations.	Nessuna impostazione predefinita; opzionale
`nodePluginNodeSelector`	Selettori di nodi aggiuntivi per i pod. Segue lo stesso formato di pod.spec.nodeSelector.	Nessuna impostazione predefinita; opzionale
`nodePluginTolerations`	Ignora le tolleranze Kubernetes per i pod. Segue lo stesso formato di pod.spec.Tolerations.	Nessuna impostazione predefinita; opzionale

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

Configurazioni di esempio

È possibile utilizzare gli attributi menzionati in precedenza per la definizione `TridentOrchestrator` per personalizzare l'installazione.

Esempio 1: Configurazione personalizzata di base

Questo è un esempio per una configurazione personalizzata di base.

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

Esempio 2: Implementazione con selettori di nodo

Questo esempio illustra come può essere implementato Trident con i selettori di nodo:

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

Esempio 3: Implementazione su nodi di lavoro Windows

In questo esempio viene illustrata la distribuzione su un nodo di lavoro Windows.

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

Informazioni sul copyright

Informazioni sul marchio commerciale