Inizia subito

Astra Trident

NetApp
April 04, 2024

Sommario

Inizia subito

Provalo

NetApp fornisce un'immagine di laboratorio pronta all'uso che è possibile richiedere tramite "Test drive di NetApp".

Scopri di più su Test Drive

Il Test Drive offre un ambiente sandbox dotato di un cluster Kubernetes a tre nodi e Astra Trident installato e configurato. È un ottimo modo per familiarizzare con Astra Trident ed esplorarne le funzionalità.

Un'altra opzione consiste nel vedere "Guida all'installazione di kubeadm" Fornito da Kubernetes.

> Non utilizzare il cluster Kubernetes creato utilizzando queste istruzioni in produzione. Utilizza le guide all'implementazione in produzione fornite dalla tua distribuzione per creare cluster pronti per la produzione.

Se questa è la prima volta che utilizzi Kubernetes, familiarizza con i concetti e gli strumenti "qui".

Requisiti

Prima di installare Astra Trident, è necessario esaminare questi requisiti generali di sistema. I backend specifici potrebbero avere requisiti aggiuntivi.

Informazioni critiche su Astra Trident 23.01

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1.27 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.

Frontend supportati (orchestratori)

Astra Trident supporta diversi motori e orchestratori di container, tra cui:

- Anthos on-Prem (VMware) e anthos on Bare Metal 1.12
- Kubernetes 1.21 - 1.27
- Motore di Mirantis Kubernetes 3.5

- OpenShift 4.9 - 4.12

L'operatore Trident è supportato con le seguenti versioni:

- Anthos on-Prem (VMware) e anthos on Bare Metal 1.12
- Kubernetes 1.21 - 1.27
- OpenShift 4.9 - 4.12

Astra Trident lavora anche con una serie di altre offerte Kubernetes completamente gestite e autogestite, tra cui Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Rancher e VMware Tanzu Portfolio.

> Prima di aggiornare un cluster Kubernetes dalla versione 1.24 alla 1.25 o successiva su cui è installato Astra Trident, vedere "Aggiornare un'installazione basata su Helm".

Back-end supportati (storage)

Per utilizzare Astra Trident, sono necessari uno o più dei seguenti backend supportati:

- Amazon FSX per NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Cloud Volumes Service per GCP
- FAS/AFF/Select 9.5 o versione successiva
- Array All SAN (ASA) NetApp
- Software NetApp HCI/Element 11 o superiore

Requisiti delle funzionalità

La tabella seguente riassume le funzionalità disponibili con questa release di Astra Trident e le versioni di Kubernetes supportate.

Funzione	Versione di Kubernetes	Sono richiesti i gate delle funzionalità?
Trident CSI	1.21 - 1.27	No
Snapshot dei volumi	1.21 - 1.27	No
PVC dalle istantanee dei volumi	1.21 - 1.27	No
Ridimensionamento di iSCSI PV	1.21 - 1.27	No
CHAP bidirezionale ONTAP	1.21 - 1.27	No
Policy di esportazione dinamiche	1.21 - 1.27	No

Funzione	Versione di Kubernetes	Sono richiesti i gate delle funzionalità?
Operatore Trident	1.21 - 1.27	No
Topologia CSI	1.21 - 1.27	No

Sistemi operativi host testati

Sebbene Astra Trident non supporti ufficialmente sistemi operativi specifici, sono noti i seguenti elementi:

- Versioni di RedHat CoreOS (RHCOS) supportate da OpenShift Container Platform (AMD64 e ARM64)
- RHEL 8+ (AMD64 E ARM64)
- Ubuntu 22.04 o versione successiva (AMD64 e ARM64)
- Windows Server 2019 (AMD64)

Per impostazione predefinita, Astra Trident viene eseguito in un container e, di conseguenza, viene eseguito su qualsiasi worker Linux. Tuttavia, questi lavoratori devono essere in grado di montare i volumi forniti da Astra Trident utilizzando il client NFS standard o iSCSI Initiator, a seconda dei backend utilizzati.

Il `tridentctl` Utility può essere eseguita anche su una qualsiasi di queste distribuzioni di Linux.

Configurazione dell'host

Tutti i nodi di lavoro nel cluster Kubernetes devono essere in grado di montare i volumi forniti per i pod. Per preparare i nodi di lavoro, è necessario installare gli strumenti NFS o iSCSI in base alla selezione del driver.

"Preparare il nodo di lavoro"

Configurazione del sistema storage

Astra Trident potrebbe richiedere modifiche a un sistema storage prima che possa essere utilizzato da una configurazione di back-end.

"Configurare i backend"

Porte Astra Trident

Astra Trident richiede l'accesso a porte specifiche per la comunicazione.

"Porte Astra Trident"

Immagini container e corrispondenti versioni di Kubernetes

Per le installazioni a gapping d'aria, l'elenco seguente è un riferimento alle immagini dei container necessarie per installare Astra Trident. Utilizzare `tridentctl images` per verificare l'elenco delle immagini container necessarie.

Versione di Kubernetes	Immagine container
v1.21.1.0	• docker.io/netapp/trident:23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisioner:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (opzionale)
v1.22.0	• docker.io/netapp/trident:23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisioner:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (opzionale)
v1.23.0	• docker.io/netapp/trident:23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisioner:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (opzionale)

Versione di Kubernetes	Immagine container
v1.24.0	• docker.io/netapp/trident:23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisioner:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (opzionale)
v1.25.0	• docker.io/netapp/trident:23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisioner:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (opzionale)
v1.26.0	• docker.io/netapp/trident:23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisioner:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (opzionale)

Versione di Kubernetes	Immagine container
v1.27.0	• docker.io/netapp/trident:23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisioner:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (opzionale)

Su Kubernetes versione 1.21 e successive, utilizzare il validato `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` immagine solo se `v1` la versione di sta servendo `volumesnapshots.snapshot.storage.k8s.gcr.io` CRD. Se il `v1beta1` La versione sta servendo il CRD con/senza `v1` versione, utilizzare il validato `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` immagine.

Installare Astra Trident

Scopri di più sull'installazione di Astra Trident

Per garantire che Astra Trident possa essere installato in una vasta gamma di ambienti e organizzazioni, NetApp offre diverse opzioni di installazione. Puoi installare Astra Trident usando l'operatore Trident (manualmente o usando Helm) o con `tridentctl`. In questo argomento vengono fornite informazioni importanti per la scelta del processo di installazione appropriato.

Informazioni critiche su Astra Trident 23.04

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1.27 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.

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 requisiti di funzionalità attivati. Esaminare "requisiti" per ulteriori informazioni.
- Accesso a un sistema storage NetApp supportato.
- Possibilità di montare volumi da tutti i nodi di lavoro Kubernetes.
- Un host Linux con `kubectl` (o. `oc`, Se si utilizza OpenShift) installato e configurato per gestire il cluster Kubernetes che si desidera utilizzare.
- Il `KUBECONFIG` Variabile d'ambiente impostata per puntare alla configurazione del cluster Kubernetes.
- Se utilizzi Kubernetes con Docker Enterprise, "Seguire la procedura per abilitare l'accesso CLI".

Se non si è ancora familiarizzato con il "concetti di base", è il momento ideale per farlo.

Scegliere il metodo di installazione desiderato

Seleziona il metodo di installazione più adatto alle tue esigenze. È inoltre necessario esaminare le considerazioni per "passaggio da un metodo all'altro" prima di prendere la decisione.

Utilizzando l'operatore Trident

Sia che si tratti di implementare manualmente o utilizzare Helm, l'operatore Trident è un ottimo modo per semplificare l'installazione e gestire dinamicamente le risorse di Astra Trident. Puoi anche farlo "Personalizzare l'implementazione dell'operatore Trident" utilizzando gli attributi in `TridentOrchestrator` Risorsa personalizzata (CR).

I vantaggi derivanti dall'utilizzo dell'operatore Trident includono:

** Astra Trident Object crefoot **

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

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

** capabilitàdi **

L'operatore monitora l'installazione di Astra Trident e prende attivamente le misure necessarie per risolvere i problemi, ad esempio quando l'implementazione viene eliminata o se viene accidentalmente modificata. R `trident-operator-<generated-id>` viene creato un pod che associa a. `TridentOrchestrator` CR con installazione Astra Trident. In questo modo si garantisce la presenza di una sola istanza di Astra Trident nel cluster e ne controlla la configurazione, assicurandosi che l'installazione sia idempotent. Quando vengono apportate modifiche all'installazione (ad esempio, l'eliminazione dell'implementazione o del demonset di nodi), l'operatore li identifica e li corregge singolarmente.

** Easy aggiorna l'installazione esistente **

È possibile aggiornare facilmente un'implementazione esistente con l'operatore. È sufficiente modificare `TridentOrchestrator` CR per aggiornare un'installazione.

Ad esempio, si consideri uno scenario in cui è necessario abilitare Astra Trident per generare i log di debug. A tale scopo, applicare una patch al `TridentOrchestrator` da impostare `spec.debug` a. `true`:

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

Dopo `TridentOrchestrator` viene aggiornato, l'operatore elabora gli aggiornamenti e le patch dell'installazione esistente. Questo potrebbe attivare la creazione di nuovi pod per modificare l'installazione di conseguenza.

** aggiornamento handlate **

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

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

** consente di gestire i cluster utilizzando BlueXP (in precedenza Cloud Manager) **

Con "Astra Trident con BlueXP", È possibile eseguire l'aggiornamento alla versione più recente di Astra Trident, aggiungere e gestire classi di storage e connetterle agli ambienti di lavoro, nonché eseguire il backup di volumi persistenti utilizzando Cloud Backup Service. BlueXP supporta l'implementazione di Astra Trident utilizzando l'operatore Trident, manualmente o utilizzando Helm.

Utilizzo di `tridentctl`

Se si dispone di un'implementazione esistente che deve essere aggiornata o se si desidera personalizzare in modo efficace l'implementazione, è necessario prendere in considerazione . Questo è il metodo convenzionale per implementare Astra Trident.

È possibile Per generare i manifesti per le risorse Trident. Ciò include la distribuzione, il demonset, l'account del servizio e il ruolo del cluster creato da Astra Trident durante l'installazione.

A partire dalla versione 22.04, le chiavi AES non verranno più rigenerate ogni volta che Astra Trident viene installato. Con questa release, Astra Trident installerà un nuovo oggetto segreto che persiste tra le installazioni. Questo significa, `tridentctl` In 22.04 è possibile disinstallare le versioni precedenti di Trident, ma le versioni precedenti non possono disinstallare le installazioni 22.04.
Selezionare il *metodo* di installazione appropriato.

Scegliere la modalità di installazione

Determinare il processo di implementazione in base alla *modalità di installazione* (Standard, Offline o Remote) richiesta dall'organizzazione.

Installazione standard

Questo è il modo più semplice per installare Astra 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 Trident richiesto (`docker.io`) E CSI (`registry.k8s.io`).

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

- Recupera le immagini container su Internet
- Crea una distribuzione o un demonset di nodi, che consente di attivare i pod Astra Trident su tutti i nodi idonei nel cluster Kubernetes

Installazione offline

La modalità di installazione offline potrebbe essere richiesta in un luogo sicuro o con aria compressa. In questo scenario, è possibile creare un singolo registro privato mirrorato o due registri mirrorati per memorizzare le immagini Trident e CSI richieste.

Indipendentemente dalla configurazione del Registro di sistema, le immagini CSI devono risiedere in un unico Registro di sistema.

Installazione remota

Di seguito viene riportata una panoramica generale del processo di installazione remota:

- Implementare la versione appropriata di `kubectl` Sul computer remoto da cui si desidera implementare Astra Trident.
- Copiare i file di configurazione dal cluster Kubernetes e impostare `KUBECONFIG` variabile di ambiente sul computer remoto.
- Avviare un `kubectl get nodes` Per verificare che sia possibile connettersi al cluster Kubernetes richiesto.
- Completare l'implementazione dal computer remoto utilizzando i passaggi di installazione standard.

Selezionare il processo in base al metodo e alla modalità

Dopo aver preso le decisioni, selezionare il processo appropriato.

Metodo	Modalità di installazione
Operatore Trident (manualmente)	"Installazione standard" "Installazione offline"
Operatore Trident (Helm)	"Installazione standard" "Installazione offline"
`tridentctl`	"Installazione standard o offline"

Passaggio da un metodo di installazione all'altro

È possibile modificare il metodo di installazione. Prima di procedere, considerare quanto segue:

- Utilizzare sempre lo stesso metodo per installare e disinstallare Astra Trident. Se hai implementato con `tridentctl`, utilizzare la versione appropriata di `tridentctl` Binario per disinstallare Astra Trident. Allo stesso modo, se si esegue la distribuzione con l'operatore, è necessario modificare `TridentOrchestrator` CR e set `spec.uninstall=true` Per disinstallare Astra Trident.
- Se si dispone di un'implementazione basata su operatore che si desidera rimuovere e utilizzare `tridentctl` Per implementare Astra Trident, devi prima modificarlo `TridentOrchestrator` e impostare `spec.uninstall=true` Per disinstallare Astra Trident. Quindi eliminare `TridentOrchestrator` e l'implementazione dell'operatore. È quindi possibile installare utilizzando `tridentctl`.
- Se si dispone di un'implementazione manuale basata su operatore e si desidera utilizzare l'implementazione dell'operatore Trident basata su Helm, è necessario prima disinstallare manualmente l'operatore ed eseguire l'installazione di Helm. Ciò consente a Helm di implementare l'operatore Trident con le etichette e le annotazioni richieste. In caso contrario, l'implementazione dell'operatore Trident basata su Helm avrà esito negativo, con un errore di convalida dell'etichetta e un errore di convalida dell'annotazione. Se si dispone di un `tridentctl`L'implementazione basata su consente di utilizzare l'implementazione basata su Helm senza problemi.

Altre opzioni di configurazione note

Quando si installa Astra Trident sui prodotti del portfolio VMware Tanzu:

- Il cluster deve supportare workload con privilegi.
- Il `--kubelet-dir` flag deve essere impostato sulla posizione della directory di kubelet. Per impostazione predefinita, questo è `/var/vcap/data/kubelet`.

  Specificare la posizione del kubelet utilizzando `--kubelet-dir` È noto per lavorare con Trident Operator, Helm e. `tridentctl` implementazioni.

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 23.04

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1.27 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/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.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 versioni precedenti, utilizzare `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 implementa l'operatore in `trident` namespace. Se il `trident` namespace inesistente, crearlo utilizzando:

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

  a. Creare il `kustomization.yaml` usando il seguente comando dove si trova *<bundle>* `bundle_pre_1_25` oppure `bundle_post_1_25` In base alla versione di Kubernetes.

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

  b. Compilare il bundle usando il seguente comando dove *<bundle>* è `bundle_pre_1_25` oppure

`bundle_post_1_25` In base alla versione di Kubernetes.

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

Fasi

1. Creare le risorse e implementare l'operatore:

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

2. Verificare che l'operatore, l'implementazione e i replicaset siano stati creati.

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

> 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:23.04
    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:23.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.04.0
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 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 altro già esiste.

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 |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

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 23.04

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1.27 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/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.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:23.04.0` 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:23.04.0` se tutte le immagini si trovano in un unico registro.

- `image: <your-registry>/netapp/trident-operator:23.04.0` Se l'immagine Trident si trova in un registro diverso dalle immagini CSI.

Fase 4: 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 versioni precedenti, utilizzare `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 implementa l'operatore in `trident` namespace. Se il `trident` namespace inesistente, crearlo utilizzando:

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

  a. Creare il `kustomization.yaml` usando il seguente comando dove si trova *<bundle>* `bundle_pre_1_25` oppure `bundle_post_1_25` In base alla versione di Kubernetes.

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

  b. Compilare il bundle usando il seguente comando dove *<bundle>* è `bundle_pre_1_25` oppure `bundle_post_1_25` In base alla versione di Kubernetes.

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

Fasi

1. Creare le risorse e implementare l'operatore:

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

2. Verificare che l'operatore, l'implementazione e i replicaset siano stati creati.

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

> 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:23.04"
tridentImage: "<your-registry>/trident:23.04.0"
```

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:23.04"
tridentImage: "<your-registry>/netapp/trident:23.04.0"
```

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:23.04
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.04.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.04
    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:23.04.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.04.0
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 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 altro già esiste.

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 |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

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 23.04

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1.27 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 `23.04.0` È la versione di Astra Trident che si sta installando.

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--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 `23.04.0` È la versione di Astra Trident che si sta installando:

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--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.	

Opzione	Descrizione	Predefinito
`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`
`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>`

Opzione	Descrizione	Predefinito
`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.	""
`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 23.04

È necessario leggere le seguenti informazioni critiche su Astra Trident.

informazioni su Astra

- Kubernetes 1.27 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, `23.04.0` È la versione di Astra Trident che si sta

installando.

Immagini in un registro

```
helm install <name> netapp-trident/trident-operator --version
23.04.0 --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
23.04.0 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.04.0 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.04 --set tridentImage=<your-
registry>/netapp/trident:23.04.0 --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 `23.04.0` È la versione di Astra Trident che si sta installando:

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--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
`enableForceDetach`	`ontap-san` e. `ontap-san-economy` solo. Funziona con Kubernetes non-Graged Node Shutdown (NGNS) per consentire agli amministratori del cluster di migrare in sicurezza i carichi di lavoro con volumi montati su nuovi nodi in caso di problemi di integrità di un nodo. **Questa è una funzionalità sperimentale di 23.04.** fare riferimento a. Dettagli sulla forza di distacco per dettagli importanti.	`false`
`windows`	Impostazione su `true` Attiva l'installazione su nodi di lavoro Windows.	falso
`useIPv6`	Installare Astra Trident su IPv6	falso

Parametro	Descrizione	Predefinito
`k8sTimeout`	Timeout per le operazioni Kubernetes	30 sec
`silenceAutosupport`	Non inviare pacchetti AutoSupport a NetApp in maniera automatica	falso
`autosupportImage`	L'immagine del contenitore per la telemetria AutoSupport	"netapp/trident-autosupport:23,07"
`autosupportProxy`	Indirizzo/porta di un proxy per l'invio della AutoSupport Telemetria	"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:23,07"
`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`

Parametro	Descrizione	Predefinito
`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".

Dettagli sulla forza di distacco

Forza distacco disponibile per `ontap-san` e. `ontap-san-economy` solo. Prima di attivare la funzione di force stach, è necessario attivare la funzione NGNS (non-aggraziate node shutdown) sul cluster Kubernetes. Per ulteriori informazioni, fare riferimento a. "Kubernetes: Shutdown del nodo non aggraziato".

Poiché Astra Trident si affida a Kubernetes NGNS, non rimuovere `out-of-service` esegue il tinding da un nodo non integro fino a quando tutti i carichi di lavoro non tollerabili non vengono ripianificati. L'applicazione o la rimozione sconsiderata della contaminazione può compromettere la protezione dei dati back-end.

Quando l'amministratore del cluster Kubernetes ha applicato il `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` al nodo e. `enableForceDetach` è impostato su `true`, Astra Trident determinerà lo stato del nodo e:

1. Interrompere l'accesso i/o back-end per i volumi montati su quel nodo.
2. Contrassegna l'oggetto nodo Astra Trident come `dirty` (non sicuro per le nuove pubblicazioni).

Il controller Trident rifiuterà le nuove richieste di volumi di pubblicazione fino a quando il nodo non viene riqualificato (dopo essere stato contrassegnato come) `dirty`) Dal pod di nodi Trident. Tutti i carichi di lavoro pianificati con un PVC montato (anche dopo che il nodo del cluster è integro e pronto) non saranno accettati fino a quando Astra Trident non sarà in grado di verificare il nodo `clean` (sicuro per le nuove pubblicazioni).

Quando lo stato del nodo viene ripristinato e la contaminazione viene rimossa, Astra Trident:

1. Identificare e pulire i percorsi pubblicati obsoleti sul nodo.
2. Se il nodo si trova in una `cleanable` stato (la manutenzione fuori servizio è stata rimossa e il nodo si trova in `Ready` state) e tutti i percorsi pubblicati e obsoleti sono puliti, Astra Trident riporterà il nodo come `clean` e consentire nuovi volumi pubblicati al nodo.

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
```

Installare usando tridentctl

Installare usando tridentctl

Puoi installare Astra Trident usando `tridentctl`. Questo processo si applica alle installazioni in cui le immagini container richieste da Astra Trident sono memorizzate o meno in un registro privato. Per personalizzare il `tridentctl` implementazione, fare riferimento a. "Personalizzare l'implementazione tridentctl".

Informazioni critiche su Astra Trident 23.04

È necessario leggere le seguenti informazioni critiche su Astra Trident.

** informazioni su Astra **

- Kubernetes 1.27 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.

Installare Astra Trident utilizzando `tridentctl`

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

```
wget https://github.com/NetApp/trident/releases/download/v23.04.0/trident-installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

Fase 2: Installare Astra Trident

Installare Astra Trident nello spazio dei nomi desiderato eseguendo `tridentctl install` comando. È possibile aggiungere ulteriori argomenti per specificare la posizione del Registro di sistema dell'immagine.

Modalità standard

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

Immagini in un registro

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:23.04 --trident
-image <your-registry>/trident:23.04.0
```

Immagini in diversi registri

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

```
./tridentctl install -n trident --image-registry <your-registry>/sig-
storage --autosupport-image <your-registry>/netapp/trident-
autosupport:23.04 --trident-image <your-
registry>/netapp/trident:23.04.0
```

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=23.04.0
INFO Trident installation succeeded.
....
```

Verificare l'installazione

È possibile verificare l'installazione utilizzando lo stato di creazione del pod o. `tridentctl`.

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-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 oppure `trident-controller-<generated id>` (`trident-csi-<generated id>` Nelle versioni precedenti alla 23.01) non ha lo stato **running**, la piattaforma non è stata installata. Utilizzare `-d` a. "attivare la modalità di debug" e risolvere il problema.

Utilizzo di `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

Configurazioni di esempio

Esempio 1: Abilitare Astra Trident per l'esecuzione sui nodi Windows

Per consentire l'esecuzione di Astra Trident su nodi Windows:

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

Esempio 2: Abilitare la forza di distacco

Per ulteriori informazioni sulla forza di distacco, fare riferimento a. "Personalizzare l'installazione dell'operatore Trident".

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

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 di tridentctl

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

Informazioni sul programma di installazione

Il programma di installazione di Astra Trident consente di personalizzare gli attributi. Ad esempio, se l'immagine Trident è stata copiata in un repository privato, è possibile specificare il nome dell'immagine utilizzando `--trident-image`. Se l'immagine Trident e le immagini sidecar CSI necessarie sono state copiate in un repository privato, potrebbe essere preferibile specificare la posizione di tale repository utilizzando `--image-registry` switch, che assume la forma `<registry FQDN>[:port]`.

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

Se è necessario personalizzare l'installazione oltre a quanto consentito dall'argomento del programma di installazione, è possibile personalizzare i file di distribuzione. Utilizzando il `--generate-custom-yaml` Il parametro crea i seguenti file YAML nel programma di installazione `setup` directory:

- `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, è possibile modificarli in base alle proprie esigenze e utilizzarli `--use-custom-yaml` per installare l'implementazione personalizzata.

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

Cosa succederà?

Dopo aver installato Astra Trident, è possibile procedere con la creazione di un backend, la creazione di una classe di storage, il provisioning di un volume e il montaggio del volume in un pod.

Fase 1: Creazione di un backend

È ora possibile creare un backend che verrà utilizzato da Astra Trident per il provisioning dei volumi. A tale scopo, creare un `backend.json` che contiene i parametri necessari. I file di configurazione di esempio per diversi tipi di backend sono disponibili in `sample-input` directory.

Vedere "qui" per ulteriori informazioni su come configurare il file per il tipo di backend.

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

```
./tridentctl -n trident create backend -f backend.json
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+
```

Se la creazione non riesce, si è verificato un errore nella configurazione del back-end. È possibile visualizzare i log per determinare la causa eseguendo il seguente comando:

```
./tridentctl -n trident logs
```

Dopo aver risolto il problema, tornare all'inizio di questo passaggio e riprovare. Per ulteriori suggerimenti sulla risoluzione dei problemi, vedere "la risoluzione dei problemi" sezione.

Fase 2: Creazione di una classe di storage

Kubernetes consente agli utenti di eseguire il provisioning dei volumi utilizzando le dichiarazioni di volumi persistenti (PVC) che specificano a. "classe di storage" per nome. I dettagli sono nascosti agli utenti, ma una classe di storage identifica il provisioning utilizzato per tale classe (in questo caso Trident) e il significato di tale classe per il provisioning.

Creare una classe di storage Kubernetes gli utenti specificheranno quando desiderano un volume. La configurazione della classe deve modellare il backend creato nel passaggio precedente, in modo che Astra Trident lo utilizzi per il provisioning di nuovi volumi.

La classe di storage più semplice da utilizzare è basata su `sample-input/storage-class-csi.yaml.templ` file fornito con il programma di installazione, in sostituzione *`BACKEND_TYPE`* con il nome del driver di storage.

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

Si tratta di un oggetto Kubernetes, quindi si utilizza `kubectl` Per crearlo in Kubernetes.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

Ora dovrebbe essere visualizzata una classe di storage **Basic-csi** in Kubernetes e Astra Trident, mentre Astra Trident avrebbe scoperto i pool sul backend.

```
kubectl get sc basic-csi
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Fase 3: Eseguire il provisioning del primo volume

Ora sei pronto per eseguire il provisioning dinamico del tuo primo volume. Per eseguire questa operazione, creare un Kubernetes "richiesta di volume persistente" (PVC).

Creare un PVC per un volume che utilizzi la classe di storage appena creata.

Vedere `sample-input/pvc-basic-csi.yaml` ad esempio. Assicurarsi che il nome della classe di storage corrisponda a quello creato.

```
kubectl create -f sample-input/pvc-basic-csi.yaml

kubectl get pvc --watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic          1s
basic     Pending   pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   0
basic          5s
basic     Bound     pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   1Gi
RWO           basic          7s
```

Fase 4: Montare i volumi in un pod

Ora montiamo il volume. Lanceremo un pod nginx che monta il PV sotto `/usr/share/nginx/html`.

```
cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

A questo punto, il pod (applicazione) non esiste più, ma il volume è ancora presente. Se lo si desidera, è possibile utilizzarlo da un altro pod.

Per eliminare il volume, eliminare la richiesta di rimborso:

```
kubectl delete pvc basic
```

È ora possibile eseguire attività aggiuntive, come ad esempio:

- "Configurare backend aggiuntivi."
- "Creare ulteriori classi di storage."

Informazioni sul copyright

Informazioni sul marchio commerciale