Inizia subito

Astra Trident

NetApp
April 16, 2024

Sommario

Inizia subito

Provalo

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

Inizia esaminando i frontend, i backend e la configurazione host supportati.

> Per ulteriori informazioni sulle porte utilizzate da Astra Trident, vedere "qui".

Frontend supportati (orchestratori)

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

- Anthos on-Prem (VMware) e anthos on Bare Metal 1.8, 1.9, 1.10
- Kubernetes 1.17 o versione successiva (ultimo: 1.23)
- Motore di Mirantis Kubernetes 3.4
- OpenShift 4.7, 4.8, 4.9

L'operatore Trident è supportato con le seguenti versioni:

- Anthos on-Prem (VMware) e anthos on Bare Metal 1.8, 1.9, 1.10
- Kubernetes 1.17 o versione successiva (ultimo: 1.23)
- OpenShift 4.7, 4.8, 4.9

> Gli utenti di Red Hat OpenShift Container Platform potrebbero notare che il file initiator name.iscsi è vuoto se si utilizza una versione inferiore a 4.6.8. Questo è un bug identificato da RedHat per essere corretto con OpenShift 4.6.8. Vedi questo "annuncio di risoluzione dei bug". NetApp consiglia di utilizzare Astra Trident su OpenShift 4.6.8 e versioni successive.

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.

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
- Archivio dati Astra
- Cloud Volumes ONTAP
- Cloud Volumes Service per GCP
- FAS/AFF/Select 9.3 o versione successiva
- Array All SAN (ASA) NetApp
- Software NetApp HCI/Element 11 o versione successiva

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	1.17 e versioni successive	No
Snapshot dei volumi	1.17 e versioni successive	No
PVC dalle istantanee dei volumi	1.17 e versioni successive	No
Ridimensionamento di iSCSI PV	1.17 e versioni successive	No
CHAP bidirezionale ONTAP	1.17 e versioni successive	No
Policy di esportazione dinamiche	1.17 e versioni successive	No
Operatore Trident	1.17 e versioni successive	No
Preparazione automatica dei nodi di lavoro (beta)	1.17 e versioni successive	No
Topologia CSI	1.17 e versioni successive	No

Sistemi operativi host testati

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.

Sebbene Astra Trident non supporti ufficialmente sistemi operativi specifici, le seguenti distribuzioni Linux sono note per funzionare:

- Versioni di RedHat CoreOS (RHCOS) supportate da OpenShift Container Platform
- RHEL o CentOS 7.4 o versione successiva
- Ubuntu 18.04 o versione successiva

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

Configurazione dell'host

A seconda del backend in uso, le utility NFS e/o iSCSI devono essere installate su tutti gli utenti del cluster. Vedere "qui" per ulteriori informazioni.

Configurazione del sistema storage

Astra Trident potrebbe richiedere alcune modifiche a un sistema storage prima che possa essere utilizzato da una configurazione di back-end. Vedere "qui" per ulteriori informazioni.

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
v1.17.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v2.2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)
v1.18.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v2.2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)

Versione di Kubernetes	Immagine container
v1.19.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v2.2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)
v1.20.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v3.1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)
v1.21.1.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v3.1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)

Versione di Kubernetes	Immagine container
v1.22.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v3.1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)
v1.23.0	• netapp/tridente:22.01.1 • netapp/trident-autosupport:22.01 • k8s.gcr.io/sig-storage/csi-provisioner:v3.1.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.4.0 • k8s.gcr.io/sig-storage/csi-resizer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0 • netapp/trident-operator:22.01.1 (opzionale)

Su Kubernetes versione 1.20 e successive, utilizzare il validato `k8s.gcr.io/sig-storage/csi-snapshotter:v4.x` immagine solo se `v1` la versione di sta servendo `volumesnapshots.snapshot.storage.k8s.io` CRD. Se il `v1beta1` La versione sta servendo il CRD con/senza `v1` versione, utilizzare il validato `k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` immagine.

Panoramica dell'implementazione

Puoi implementare Astra Trident usando l'operatore Trident o con `tridentctl`.

Scegliere il metodo di implementazione

Per determinare quale metodo di implementazione utilizzare, considerare quanto segue:

Perché dovrei utilizzare l'operatore Trident?

Il "Operatore Trident" È un ottimo modo per gestire dinamicamente le risorse di Astra Trident e automatizzare la fase di setup. Alcuni prerequisiti devono essere soddisfatti. Vedere "i requisiti".

L'operatore Trident offre diversi vantaggi, come indicato di seguito.

Funzionalità di riparazione automatica

È possibile monitorare un'installazione di Astra Trident e prendere attivamente misure per risolvere problemi, ad esempio quando l'implementazione viene eliminata o se viene modificata accidentalmente. Quando l'operatore è impostato come implementazione, un `trident-operator-<generated-id>` pod creato. Questo pod associa un `TridentOrchestrator` CR con un'installazione Astra Trident e garantisce sempre che sia attivo un solo prodotto `TridentOrchestrator`. In altre parole, l'operatore garantisce che vi sia una sola istanza di Astra Trident nel cluster e ne controlla la configurazione, assicurandosi che l'installazione sia idempotente. 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.

Semplici aggiornamenti alle installazioni esistenti

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.

Gestisce automaticamente gli aggiornamenti di Kubernetes

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.

Perché dovrei usare Helm?

Se si utilizzano altre applicazioni che vengono gestite utilizzando Helm, a partire da Astra Trident 21.01, è possibile gestire la distribuzione anche utilizzando Helm.

Quando dovrei usare `tridenctl`?

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

Considerazioni sul passaggio da un metodo di implementazione all'altro

Non è difficile immaginare uno scenario in cui si desidera passare da un metodo di implementazione all'altro. Prima di tentare di spostarsi da un, tenere in considerazione quanto segue `tridentctl` implementazione su

un'implementazione basata su operatore o viceversa:

- Utilizzare sempre lo stesso metodo per 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 desidera rimuovere e utilizzare un'implementazione basata su operatore `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, quindi 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.

Comprendere le modalità di implementazione

Ci sono tre modi per implementare Astra Trident.

Implementazione standard

L'implementazione di Trident su un cluster Kubernetes comporta due operazioni da parte del programma di installazione di Astra Trident:

- Recupero delle immagini container tramite Internet
- Creazione di un demonset di implementazione e/o nodi, che consente di attivare i pod Astra Trident su tutti i nodi idonei nel cluster Kubernetes.

Un'implementazione standard come questa può essere eseguita in due modi diversi:

- Utilizzo di `tridentctl install`
- Utilizzando l'operatore Trident. È possibile implementare l'operatore Trident manualmente o utilizzando Helm.

Questa modalità di installazione è il modo più semplice per installare Astra Trident e funziona per la maggior parte degli ambienti che non impongono restrizioni di rete.

Implementazione offline

Per eseguire un'implementazione con aria compressa, è possibile utilizzare `--image-registry` contrassegno durante l'invocazione `tridentctl install` per puntare a un registro di immagini privato. Se si esegue l'implementazione con l'operatore Trident, è possibile specificare in alternativa `spec.imageRegistry` nel `TridentOrchestrator`. Questo registro deve contenere "Immagine di Trident", il "Immagine Trident AutoSupport"E le immagini sidecar CSI come richiesto dalla versione di Kubernetes.

Per personalizzare l'implementazione, è possibile utilizzare `tridentctl` Generare i manifesti per le risorse di Trident. Ciò include la distribuzione, il demonset, l'account del servizio e il ruolo del cluster creato da Astra Trident durante l'installazione.

Per ulteriori informazioni sulla personalizzazione della distribuzione, consultare i seguenti collegamenti:

- "Personalizza la tua implementazione basata su operatore"
  *

> Se si utilizza un repository di immagini privato, è necessario aggiungere `/k8scsi` Per le versioni di Kubernetes precedenti alla 1.17 o. `/sig-storage` Per le versioni di Kubernetes successive alla 1.17 fino alla fine dell'URL privato del Registro di sistema. Quando si utilizza un registro di sistema privato per `tridentctl` implementazione, è necessario utilizzare `--trident-image` e. `--autosupport-image` in combinazione con `--image-registry`. Se stai implementando Astra Trident utilizzando l'operatore Trident, assicurati che orchestrator CR includa `tridentImage` e. `autosupportImage` nei parametri di installazione.

Implementazione remota

Di seguito viene riportata una panoramica generale del processo di implementazione 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.

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.

Implementazione con l'operatore Trident

Puoi implementare Astra Trident con l'operatore Trident. È possibile implementare l'operatore Trident manualmente o utilizzando Helm.

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

Di cosa hai bisogno

Per implementare Astra Trident, devono essere soddisfatti i seguenti prerequisiti:

- Si dispone dei privilegi completi per un cluster Kubernetes supportato che esegue Kubernetes 1.17 e versioni successive.
- Hai accesso a un sistema storage NetApp supportato.

- È possibile montare volumi da tutti i nodi di lavoro Kubernetes.
- Hai un host Linux con `kubectl` (o. `oc`, Se si utilizza OpenShift) installato e configurato per gestire il cluster Kubernetes che si desidera utilizzare.
- È stato impostato il `KUBECONFIG` Variabile di ambiente che punta alla configurazione del cluster Kubernetes.
- È stata attivata la "Porte caratteristiche richieste da Astra Trident".
- Se utilizzi Kubernetes con Docker Enterprise, "Seguire la procedura per abilitare l'accesso CLI".

Hai tutto questo? Fantastico! Iniziamo.

Implementare l'operatore Trident utilizzando Helm

Eseguire i passaggi elencati per implementare l'operatore Trident utilizzando Helm.

Di cosa hai bisogno

Oltre ai prerequisiti elencati in precedenza, per implementare l'operatore Trident utilizzando Helm, è necessario disporre di quanto segue:

- Kubernetes 1.17 e versioni successive
- Helm versione 3

Fasi

1. Aggiungere il repository Helm di Trident:

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

2. Utilizzare `helm install` e specificare un nome per la distribuzione. Vedere il seguente esempio:

```
helm install <release-name> netapp-trident/trident-operator --version
22.1.0 --namespace <trident-namespace>
```

> Se non è stato ancora creato uno spazio dei nomi per Trident, è possibile aggiungere `--create-namespace` al `helm install` comando. Helm crea automaticamente lo spazio dei nomi.

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

- `--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:

```
$ helm install <name> netapp-trident/trident-operator --version 22.1.0
--set tridentDebug=true
```

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

`helm list` mostra i dettagli dell'installazione, ad esempio nome, spazio dei nomi, grafico, stato, versione dell'applicazione, numero di revisione e così via.

Implementare l'operatore Trident manualmente

Eseguire i passaggi elencati per implementare manualmente l'operatore Trident.

Fase 1: Qualificare il cluster Kubernetes

La prima cosa da fare è accedere all'host Linux e verificare che stia gestendo un "Cluster Kubernetes supportato" disporre dei privilegi necessari per.

> 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`.

Per verificare se la versione di Kubernetes è successiva alla 1.17, eseguire il seguente comando:

```
kubectl version
```

Per verificare se si dispone dei privilegi di amministratore del cluster Kubernetes, eseguire il seguente comando:

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

Per verificare se è possibile avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il sistema di storage sulla rete pod, eseguire il seguente comando:

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

Fase 2: Scaricare e configurare l'operatore

> A partire da 21.01, l'operatore Trident ha un ambito cluster. L'utilizzo dell'operatore Trident per installare Trident richiede la creazione di `TridentOrchestrator` Definizione personalizzata delle risorse (CRD) e definizione di altre risorse. Prima di installare Astra Trident, eseguire questa procedura per configurare l'operatore.

1. Scaricare l'ultima versione di "Pacchetto di installazione Trident" Dalla sezione *Downloads* ed estrarla.

```
wget https://github.com/NetApp/trident/releases/download/v21.04/trident-
installer-21.04.tar.gz
tar -xf trident-installer-21.04.tar.gz
cd trident-installer
```

2. Utilizzare il manifesto CRD appropriato per creare `TridentOrchestrator` CRD. Quindi, creare un `TridentOrchestrator` Custom Resource in seguito per creare un'installazione da parte dell'operatore.

   Eseguire il seguente comando:

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

3. Dopo il `TridentOrchestrator` Viene creato un CRD, creare le seguenti risorse necessarie per l'implementazione dell'operatore:

   - Un account di servizio per l'operatore
   - Un ClusterRole e ClusterRoleBinding al ServiceAccount
   - Una policy PodSecurityPolicy dedicata
   - L'operatore stesso

     Il programma di installazione di Trident contiene i manifesti per la definizione di queste risorse. Per impostazione predefinita, l'operatore viene implementato in `trident` namespace. Se il `trident` namespace non esiste, utilizzare il seguente manifesto per crearne uno.

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

4. Per implementare l'operatore in uno spazio dei nomi diverso da quello predefinito `trident` namespace, è necessario aggiornare `serviceaccount.yaml`, `clusterrolebinding.yaml` e. `operator.yaml` manifesta e genera il tuo `bundle.yaml`.

   Eseguire il comando seguente per aggiornare i manifesti YAML e generare il `bundle.yaml` utilizzando il `kustomization.yaml`:

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

   Eseguire il seguente comando per creare le risorse e implementare l'operatore:

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

5. Per verificare lo stato dell'operatore dopo l'implementazione, procedere come segue:

```
$ kubectl get deployment -n <operator-namespace>
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m

$ kubectl get pods -n <operator-namespace>
NAME                                READY   STATUS             RESTARTS
AGE
trident-operator-54cb664d-lnjxh     1/1     Running            0
3m
```

L'implementazione dell'operatore crea correttamente un pod in esecuzione su uno dei nodi di lavoro nel cluster.

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

Fase 3: Creazione `TridentOrchestrator` E installare Trident

Ora sei pronto per installare Astra Trident usando l'operatore! Per questo è necessario creare `TridentOrchestrator`. Il programma di installazione di Trident include definizioni di esempio per la creazione `TridentOrchestrator`. In questo modo viene eseguita un'installazione in `trident` namespace.

```
$ 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:21.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

L'operatore Trident consente di personalizzare il modo in cui Astra Trident viene installato utilizzando gli attributi in `TridentOrchestrator` spec. Vedere "Personalizza la tua implementazione Trident".

Lo Stato di `TridentOrchestrator` Indica se l'installazione ha avuto esito positivo e visualizza la versione di Trident installata.

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.

Durante l'installazione, lo stato di `TridentOrchestrator` modifiche da `Installing` a. `Installed`. Se si osserva `Failed` e l'operatore non è in grado di eseguire il ripristino da solo, è necessario controllare i registri dell'operatore. Vedere "risoluzione dei problemi" sezione.

Puoi verificare se l'installazione di Astra Trident è stata completata dando un'occhiata ai pod creati:

```
$ kubectl get pod -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw        5/5     Running   0           1m
trident-csi-mr6zc                   2/2     Running   0           1m
trident-csi-xrp7w                   2/2     Running   0           1m
trident-csi-zh2jt                   2/2     Running   0           1m
trident-operator-766f7b8658-ldzsv   1/1     Running   0           3m
```

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

```
$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

Ora puoi continuare a creare un back-end. Vedere "attività post-implementazione".

Per la risoluzione dei problemi durante l'implementazione, consultare "risoluzione dei problemi" sezione.

Personalizzare l'implementazione dell'operatore Trident

L'operatore Trident consente di personalizzare il modo in cui Astra Trident viene installato utilizzando gli attributi in `TridentOrchestrator` spec.

Per un elenco degli attributi, consultare la tabella seguente:

Parametro	Descrizione	Predefinito
`namespace`	Spazio dei nomi in cui installare Astra Trident	"predefinito"
`debug`	Attiva il debug per Astra Trident	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:21.04.0"
`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 1.17+) 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	

Parametro	Descrizione	Predefinito
`imagePullSecrets`	Secrets (segreti) per estrarre immagini da un registro interno	
`controllerPluginNodeSelector`	Selettori di nodi aggiuntivi per i pod che eseguono il Plugin CSI del controller Trident. Segue lo stesso formato di pod.spec.nodeSelector.	Nessuna impostazione predefinita; opzionale
`controllerPluginTolerations`	Ignora le tolleranze per i pod che eseguono il Plugin CSI del controller Trident. Segue lo stesso formato di pod.spec.Tolerations.	Nessuna impostazione predefinita; opzionale
`nodePluginNodeSelector`	Selettori di nodi aggiuntivi per i pod che eseguono il Plugin CSI di Trident Node. Segue lo stesso formato di pod.spec.nodeSelector.	Nessuna impostazione predefinita; opzionale
`nodePluginTolerations`	Ignora le tolleranze per i pod che eseguono il Plugin CSI di Trident Node. Segue lo stesso formato di pod.spec.Tolerations.	Nessuna impostazione predefinita; opzionale

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

La preparazione automatica dei nodi di lavoro è una funzionalità * beta* che deve essere utilizzata solo in ambienti non di produzione.

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

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

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

Ecco un altro esempio che mostra come Trident può essere implementato 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
```

Se si desidera personalizzare l'installazione oltre ciò che si desidera `TridentOrchestrator` gli argomenti lo consentono, dovresti considerare di utilizzare `tridentctl` Per generare manifesti YAML personalizzati che è possibile modificare in base alle esigenze.

Implementare con tridentctl

Puoi implementare Astra Trident utilizzando `tridentctl`.

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

Per personalizzare l'implementazione, vedere "qui".

Di cosa hai bisogno

Per implementare Astra Trident, devono essere soddisfatti i seguenti prerequisiti:

- Si dispone dei privilegi completi per un cluster Kubernetes supportato.
- Hai accesso a un sistema storage NetApp supportato.
- È possibile montare volumi da tutti i nodi di lavoro Kubernetes.
- Hai un host Linux con `kubectl` (o. `oc`, Se si utilizza OpenShift) installato e configurato per gestire il cluster Kubernetes che si desidera utilizzare.
- È stato impostato il `KUBECONFIG` Variabile di ambiente che punta alla configurazione del cluster Kubernetes.
- È stata attivata la "Porte caratteristiche richieste da Astra Trident".
- Se utilizzi Kubernetes con Docker Enterprise, "Seguire la procedura per abilitare l'accesso CLI".

Hai tutto questo? Fantastico! Iniziamo.

Per informazioni sulla personalizzazione della distribuzione, vedere "qui".

Fase 1: Qualificare il cluster Kubernetes

La prima cosa da fare è accedere all'host Linux e verificare che stia gestendo un "Cluster Kubernetes supportato" disporre dei privilegi necessari per.

> Con OpenShift, si utilizza `oc` invece di `kubectl` in tutti gli esempi riportati di seguito, eseguire prima l'accesso come **system:admin** `oc login -u system:admin` oppure `oc login -u kube-admin`.

Per controllare la versione di Kubernetes, eseguire il seguente comando:

```
kubectl version
```

Per verificare se si dispone dei privilegi di amministratore del cluster Kubernetes, eseguire il seguente comando:

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

Per verificare se è possibile avviare un pod che utilizza un'immagine da Docker Hub e raggiungere il sistema di storage sulla rete pod, eseguire il seguente comando:

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

Identificare la versione del server Kubernetes. Lo userai quando installi Astra Trident.

Fase 2: Scaricare ed estrarre il programma di installazione

> Il programma di installazione di Trident crea un pod Trident, configura gli oggetti CRD utilizzati per mantenere il proprio stato e inizializza i sidecar CSI che eseguono azioni, come il provisioning e il collegamento di volumi agli host del cluster.

È possibile scaricare l'ultima versione di "Pacchetto di installazione Trident" Dalla sezione *Downloads* ed estrarla.

Ad esempio, se la versione più recente è 21.07.1:

```
wget https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
tar -xf trident-installer-21.07.1.tar.gz
cd trident-installer
```

Fase 3: Installare Astra Trident

Installare Astra Trident nello spazio dei nomi desiderato eseguendo `tridentctl install` comando.

```
$ ./tridentctl install -n trident
....
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-csi-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=21.07.1
INFO Trident installation succeeded.
....
```

Al termine del programma di installazione, il suo aspetto sarà simile a questo. A seconda del numero di nodi nel cluster Kubernetes, è possibile osservare più pod:

```
$ kubectl get pod -n trident
NAME                           READY   STATUS    RESTARTS   AGE
trident-csi-679648bd45-cv2mx   4/4     Running   0          5m29s
trident-csi-vgc8n              2/2     Running   0          5m29s

$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.1        | 21.07.1        |
+----------------+----------------+
```

Se l'output è simile all'esempio precedente, questo passaggio è stato completato, ma Astra Trident non è ancora completamente configurato. Andare avanti e passare alla fase successiva. Vedere "attività post-implementazione".

Tuttavia, se il programma di installazione non viene completato correttamente o non viene visualizzato il valore **in esecuzione** `trident-csi-<generated id>`, la piattaforma non è stata installata.

Per la risoluzione dei problemi durante l'implementazione, consultare "risoluzione dei problemi" sezione.

Personalizzare l'implementazione tridentctl

Il programma di installazione di 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]`.

Per fare in modo che Astra Trident configuri automaticamente i nodi di lavoro, utilizzare `--enable-node-prep`. Per ulteriori informazioni sul funzionamento, vedere "qui".

> La preparazione automatica del nodo di lavoro è una funzionalità * beta* destinata all'utilizzo solo in ambienti non di produzione.

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`

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 implementato 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."