Installare Astra Control Center

Astra Control Center

NetApp
June 06, 2024

Sommario

Installare Astra Control Center utilizzando il processo standard

Per installare Astra Control Center, scaricare il pacchetto di installazione dal NetApp Support Site ed eseguire la procedura seguente per installare Astra Control Center Operator e Astra Control Center nel proprio ambiente. È possibile utilizzare questa procedura per installare Astra Control Center in ambienti connessi a Internet o con connessione ad aria.

Per gli ambienti Red Hat OpenShift, è possibile utilizzare un "procedura alternativa" Per installare Astra Control Center utilizzando OpenShift OperatorHub.

Di cosa hai bisogno

- "Prima di iniziare l'installazione, preparare l'ambiente per l'implementazione di Astra Control Center".
- Se hai configurato o vuoi configurare le policy di sicurezza dei pod nel tuo ambiente, familiarizza con le policy di sicurezza dei pod e con il modo in cui influiscono sull'installazione di Astra Control Center. Vedere "Comprendere le restrizioni delle policy di sicurezza del pod".
- Assicurarsi che tutti gli operatori del cluster siano in buono stato e disponibili.

```
kubectl get clusteroperators
```

- Assicurarsi che tutti i servizi API siano in buono stato e disponibili:

```
kubectl get apiservices
```

- Assicurarsi che l'FQDN Astra che si intende utilizzare sia instradabile a questo cluster. Ciò significa che si dispone di una voce DNS nel server DNS interno o si sta utilizzando un percorso URL principale già registrato.
- Se nel cluster esiste già un cert-manager, è necessario eseguirne alcune "fasi preliminari" In modo che Astra Control Center non installi il proprio cert-manager.

A proposito di questa attività

Il processo di installazione di Astra Control Center esegue le seguenti operazioni:

- Installa i componenti Astra in `netapp-acc` namespace (o personalizzato).
- Crea un account predefinito.
- Stabilisce un indirizzo e-mail amministrativo predefinito per l'utente e una password monouso predefinita. A questo utente viene assegnato il ruolo Owner (Proprietario) nel sistema necessario per il primo accesso all'interfaccia utente.
- Consente di determinare se tutti i pod Astra Control Center sono in esecuzione.
- Installa l'interfaccia utente Astra.

(Valido solo per la release Astra Data Store Early Access Program (EAP)). Se si intende gestire Astra Data Store utilizzando Astra Control Center e abilitare i flussi di lavoro VMware, implementare Astra Control Center solo su `pcloud` namespace e non su `netapp-acc` namespace o uno spazio dei nomi personalizzato descritto nei passaggi di questa procedura.

Non eseguire il seguente comando durante l'intero processo di installazione per evitare di eliminare tutti i pod di Astra Control Center: `kubectl delete -f astra_control_center_operator_deploy.yaml`

Se si utilizza Podman di Red Hat invece di Docker Engine, è possibile utilizzare i comandi Podman al posto dei comandi Docker.

Fasi

Per installare Astra Control Center, procedere come segue:

- Scarica e disimballa il bundle Astra Control Center
- Installare il plug-in NetApp Astra kubectl
- Aggiungere le immagini al registro locale
- Impostare namespace e secret per i registri con requisiti di autenticazione
- Installare l'operatore del centro di controllo Astra
- Configurare Astra Control Center
- Completare l'installazione dell'Astra Control Center e dell'operatore
- Verificare lo stato del sistema
- Impostare l'ingresso per il bilanciamento del carico
- Accedere all'interfaccia utente di Astra Control Center

Scarica e disimballa il bundle Astra Control Center

1. Scarica il bundle Astra Control Center (`astra-control-center-[version].tar.gz`) da "Sito di supporto NetApp".
2. Scarica la zip dei certificati e delle chiavi di Astra Control Center dal "Sito di supporto NetApp".
3. (Facoltativo) utilizzare il seguente comando per verificare la firma del bundle:

```
openssl dgst -sha256 -verify AstraControlCenter-public.pub -signature
astra-control-center-[version].tar.gz.sig astra-control-center-
[version].tar.gz
```

4. Estrarre le immagini:

```
tar -vxzf astra-control-center-[version].tar.gz
```

Installare il plug-in NetApp Astra kubectl

NetApp Astra `kubectl` Il plug-in della riga di comando consente di risparmiare tempo durante l'esecuzione di attività comuni associate all'implementazione e all'aggiornamento di Astra Control Center.

Di cosa hai bisogno

NetApp fornisce binari per il plug-in per diverse architetture CPU e sistemi operativi. Prima di eseguire questa attività, è necessario conoscere la CPU e il sistema operativo in uso. Sui sistemi operativi Linux e Mac, è possibile utilizzare `uname -a` per raccogliere queste informazioni.

Fasi

1. Elencare NetApp Astra disponibile `kubectl` Binari del plug-in e annotare il nome del file necessario per il sistema operativo e l'architettura della CPU:

```
ls kubectl-astra/
```

2. Copiare il file nella stessa posizione dello standard `kubectl` utility. In questo esempio, il `kubectl` l'utility si trova in `/usr/local/bin` directory. Sostituire `<binary-name>` con il nome del file desiderato:

```
cp kubectl-astra/<binary-name> /usr/local/bin/kubectl-astra
```

Aggiungere le immagini al registro locale

1. Completare la sequenza di passaggi appropriata per il motore dei container:

Docker

1. Passare alla directory Astra:

```
cd acc
```

2. inserire le immagini del pacchetto nella directory delle immagini di Astra Control Center nel registro locale. Eseguire le seguenti sostituzioni prima di eseguire il comando:
   - Sostituire BUNDLE_FILE con il nome del file bundle Astra Control (ad esempio, `acc.manifest.yaml`).
   - Sostituire MY_REGISTRY con l'URL del repository Docker.
   - Sostituire MY_REGISTRY_USER con il nome utente.
   - Sostituire MY_REGISTRY_TOKEN con un token autorizzato per il Registro di sistema.

```
kubectl astra packages push-images -m BUNDLE_FILE -r MY_REGISTRY
-u MY_REGISTRY_USER -p MY_REGISTRY_TOKEN
```

Podman

1. Accedere al Registro di sistema:

```
podman login [your_registry_path]
```

2. Eseguire il seguente script, eseguendo la sostituzione <YOUR_REGISTRY> come indicato nei commenti:

```
# You need to be at the root of the tarball.
# You should see these files to confirm correct location:
#   acc.manifest.yaml
#   acc/

# Replace <YOUR_REGISTRY> with your own registry (e.g
registry.customer.com or registry.customer.com/testing, etc..)
export REGISTRY=<YOUR_REGISTRY>
export PACKAGENAME=acc
export PACKAGEVERSION=22.08.1-26
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
  # Load to local cache
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //')

  # Remove path and keep imageName.
  astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')

  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/netapp/astra/${PACKAGENAME}
/${PACKAGEVERSION}/${astraImageNoPath}

  # Push to the local repo.
  podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

Impostare namespace e secret per i registri con requisiti di autenticazione

1. Esportare il KUBECONFIG per il cluster host Astra Control Center:

```
export KUBECONFIG=[file path]
```

2. Se si utilizza un registro che richiede l'autenticazione, è necessario effettuare le seguenti operazioni:

   a. Creare il `netapp-acc-operator` spazio dei nomi:

```
kubectl create ns netapp-acc-operator
```

   Risposta:

```
namespace/netapp-acc-operator created
```

b. Creare un segreto per `netapp-acc-operator` namespace. Aggiungere informazioni su Docker ed eseguire il seguente comando:

> Il segnaposto `your_registry_path` deve corrispondere alla posizione delle immagini caricate in precedenza (ad esempio, `[Registry_URL]/netapp/astra/astracc/22.08.1-26`).

```
kubectl create secret docker-registry astra-registry-cred -n netapp-
acc-operator --docker-server=[your_registry_path] --docker-username
=[username] --docker-password=[token]
```

Esempio di risposta:

```
secret/astra-registry-cred created
```

> Se si elimina lo spazio dei nomi dopo la generazione del segreto, è necessario rigenerare il segreto per lo spazio dei nomi dopo la ricostruzione dello spazio dei nomi.

c. Creare il `netapp-acc` namespace (o personalizzato).

```
kubectl create ns [netapp-acc or custom namespace]
```

Esempio di risposta:

```
namespace/netapp-acc created
```

d. Creare un segreto per `netapp-acc` namespace (o personalizzato). Aggiungere informazioni su Docker ed eseguire il seguente comando:

```
kubectl create secret docker-registry astra-registry-cred -n [netapp-
acc or custom namespace] --docker-server=[your_registry_path]
--docker-username=[username] --docker-password=[token]
```

Risposta

```
secret/astra-registry-cred created
```

a. (opzionale) se si desidera che il cluster venga gestito automaticamente da Astra Control Center

dopo l'installazione, assicurarsi di fornire il kubeconfig come segreto all'interno dello spazio dei nomi di Astra Control Center in cui si intende eseguire la distribuzione utilizzando questo comando:

```
kubectl create secret generic [acc-kubeconfig-cred or custom secret name] --from-file=<path-to-your-kubeconfig> -n [netapp-acc or custom namespace]
```

Installare l'operatore del centro di controllo Astra

1. Modificare la directory:

```
cd manifests
```

2. Modificare l'YAML di implementazione dell'operatore di Astra Control Center (`astra_control_center_operator_deploy.yaml`) per fare riferimento al registro locale e al segreto.

```
vim astra_control_center_operator_deploy.yaml
```

Un YAML di esempio annotato segue questi passaggi.

a. Se si utilizza un registro che richiede l'autenticazione, sostituire la riga predefinita di `imagePullSecrets: []` con i seguenti elementi:

```
imagePullSecrets:
- name: <astra-registry-cred>
```

b. Cambiare `[your_registry_path]` per `kube-rbac-proxy` al percorso del registro in cui sono state inviate le immagini in a. passaggio precedente.

c. Cambiare `[your_registry_path]` per `acc-operator-controller-manager` al percorso del registro in cui sono state inviate le immagini in a. passaggio precedente.

d. (Per le installazioni che utilizzano l'anteprima di Astra Data Store) vedere questo problema noto relativo a. "Provisioning delle classi di storage e modifiche aggiuntive da apportare al programma YAML".

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    control-plane: controller-manager
  name: acc-operator-controller-manager
  namespace: netapp-acc-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      control-plane: controller-manager
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: [your_registry_path]/kube-rbac-proxy:v4.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        command:
        - /manager
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        image: [your_registry_path]/acc-operator:[version x.y.z]
        imagePullPolicy: IfNotPresent
      imagePullSecrets: []
```

3. Installare l'operatore del centro di controllo Astra:

```
kubectl apply -f astra_control_center_operator_deploy.yaml
```

Esempio di risposta:

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.astra.
netapp.io created
role.rbac.authorization.k8s.io/acc-operator-leader-election-role created
clusterrole.rbac.authorization.k8s.io/acc-operator-manager-role created
clusterrole.rbac.authorization.k8s.io/acc-operator-metrics-reader
created
clusterrole.rbac.authorization.k8s.io/acc-operator-proxy-role created
rolebinding.rbac.authorization.k8s.io/acc-operator-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-manager-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-proxy-
rolebinding created
configmap/acc-operator-manager-config created
service/acc-operator-controller-manager-metrics-service created
deployment.apps/acc-operator-controller-manager created
```

4. Verificare che i pod siano in esecuzione:

```
kubectl get pods -n netapp-acc-operator
```

Configurare Astra Control Center

1. Modificare il file delle risorse personalizzate (CR) di Astra Control Center (`astra_control_center_min.yaml`) Per creare account, AutoSupport, Registro di sistema e altre configurazioni necessarie:

> `astra_control_center_min.yaml` È il CR predefinito ed è adatto per la maggior parte delle installazioni. Familiarizzare con tutti "Opzioni CR e relativi valori potenziali" Per garantire la corretta implementazione di Astra Control Center per il proprio ambiente. Se sono necessarie personalizzazioni aggiuntive per il proprio ambiente, è possibile utilizzare `astra_control_center.yaml` Come CR alternativa.

```
vim astra_control_center_min.yaml
```

> Se si utilizza un registro che non richiede autorizzazione, è necessario eliminare `secret` linea entro `imageRegistry` in caso negativo, l'installazione non riesce.

a. Cambiare `[your_registry_path]` al percorso del registro di sistema in cui sono state inviate le immagini nel passaggio precedente.

b. Modificare il `accountName` stringa al nome che si desidera associare all'account.

c. Modificare il `astraAddress` Stringa all'FQDN che si desidera utilizzare nel browser per accedere ad Astra. Non utilizzare `http://` oppure `https://` nell'indirizzo. Copiare questo FQDN per utilizzarlo in un passo successivo.

d. Modificare il `email` stringa all'indirizzo iniziale predefinito dell'amministratore. Copiare questo indirizzo e-mail per utilizzarlo in passo successivo.

e. Cambiare `enrolled` Per AutoSupport a. `false` per i siti senza connettività internet o senza retain `true` per i siti connessi.

f. Se si utilizza un cert-manager esterno, aggiungere le seguenti righe a. `spec`:

```
spec:
  crds:
    externalCertManager: true
```

g. (Facoltativo) aggiungere un nome `firstName` e cognome `lastName` dell'utente associato all'account. È possibile eseguire questo passaggio ora o in un secondo momento all'interno dell'interfaccia utente.

h. (Facoltativo) modificare `storageClass` Valore per un'altra risorsa Trident storageClass, se richiesto dall'installazione.

i. (Facoltativo) se si desidera che il cluster venga gestito automaticamente da Astra Control Center dopo l'installazione e si è già provveduto creato il segreto contenente il kubeconfig per questo cluster, Fornire il nome del segreto aggiungendo un nuovo campo a questo file YAML chiamato `astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"`

j. Completare una delle seguenti operazioni:

- **Other ingress controller (ingressType:Generic)**: Questa è l'azione predefinita con Astra Control Center. Dopo l'implementazione di Astra Control Center, è necessario configurare il controller di ingresso per esporre Astra Control Center con un URL.

  L'installazione predefinita di Astra Control Center imposta il gateway (`service/traefik`) per essere del tipo `ClusterIP`. Questa installazione predefinita richiede l'impostazione di Kubernetes IngressController/Ingress per instradare il traffico verso di essa. Se si desidera utilizzare un ingresso, vedere "Impostare l'ingresso per il bilanciamento del carico".

- **Service load balancer (ingressType:AccTraefik)**: Se non si desidera installare un IngressController o creare una risorsa Ingress, impostare `ingressType` a. `AccTraefik`.

  In questo modo viene implementato l'Astra Control Center `traefik` Gateway come servizio di tipo Kubernetes LoadBalancer.

  Astra Control Center utilizza un servizio del tipo "LoadBalancer" (`svc/traefik` Nello spazio dei nomi di Astra Control Center) e richiede l'assegnazione di un indirizzo IP esterno accessibile. Se nel proprio ambiente sono consentiti i bilanciatori di carico e non ne è già configurato uno, è possibile utilizzare MetalLB o un altro servizio di bilanciamento del carico esterno per assegnare un indirizzo IP esterno al servizio. Nella configurazione del server DNS interno, puntare il nome DNS scelto per Astra Control Center sull'indirizzo IP con bilanciamento del carico.

Per ulteriori informazioni sul tipo di servizio "LoadBalancer" e sull'ingresso, vedere "Requisiti".

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"
  ingressType: "Generic"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
```

Completare l'installazione dell'Astra Control Center e dell'operatore

1. Se non lo si è già fatto in un passaggio precedente, creare il `netapp-acc` namespace (o personalizzato):

   ```
   kubectl create ns [netapp-acc or custom namespace]
   ```

   Esempio di risposta:

   ```
   namespace/netapp-acc created
   ```

2. Installare Astra Control Center in `netapp-acc` spazio dei nomi (o personalizzato):

   ```
   kubectl apply -f astra_control_center_min.yaml -n [netapp-acc or custom namespace]
   ```

   Esempio di risposta:

```
astracontrolcenter.astra.netapp.io/astra created
```

Verificare lo stato del sistema

Se preferisci utilizzare OpenShift, puoi utilizzare comandi oc paragonabili per le fasi di verifica.

1. Verificare che tutti i componenti del sistema siano installati correttamente.

```
kubectl get pods -n [netapp-acc or custom namespace]
```

Ogni pod deve avere uno stato di `Running`. L'implementazione dei pod di sistema potrebbe richiedere alcuni minuti.

Esempio di risposta

```
NAME                                        READY   STATUS    RESTARTS
AGE
acc-helm-repo-6b44d68d94-d8m55              1/1     Running   0
13m
activity-78f99ddf8-hltct                    1/1     Running   0
10m
api-token-authentication-457nl              1/1     Running   0
9m28s
api-token-authentication-dgwsz              1/1     Running   0
9m28s
api-token-authentication-hmqqc              1/1     Running   0
9m28s
asup-75fd554dc6-m6qzh                       1/1     Running   0
9m38s
authentication-6779b4c85d-92gds             1/1     Running   0
8m11s
bucketservice-7cc767f8f8-lqwr8              1/1     Running   0
9m31s
certificates-549fd5d6cb-5kmd6               1/1     Running   0
9m56s
certificates-549fd5d6cb-bkjh9               1/1     Running   0
9m56s
cloud-extension-7bcb7948b-hn8h2             1/1     Running   0
10m
cloud-insights-service-56ccf86647-fgg69     1/1     Running   0
9m46s
composite-compute-677685b9bb-7vgsf          1/1     Running   0
10m
composite-volume-657d6c5585-dnq79           1/1     Running   0
9m49s
credentials-755fd867c8-vrlmt                1/1     Running   0
11m
entitlement-86495cdf5b-nwhh2                1/1     Running   2
10m
features-5684fb8b56-8d6s8                   1/1     Running   0
10m
fluent-bit-ds-rhx7v                         1/1     Running   0
7m48s
fluent-bit-ds-rjms4                         1/1     Running   0
7m48s
fluent-bit-ds-zf5ph                         1/1     Running   0
7m48s
graphql-server-66d895f544-w6hjd             1/1     Running   0
3m29s
```

```
identity-744df448d5-rlcmm                1/1    Running  0
10m
influxdb2-0                              1/1    Running  0
13m
keycloak-operator-75c965cc54-z7csw       1/1    Running  0
8m16s
krakend-798d6df96f-9z2sk                 1/1    Running  0
3m26s
license-5fb7d75765-f8mjg                 1/1    Running  0
9m50s
login-ui-7d5b7df85d-l2s7s                1/1    Running  0
3m20s
loki-0                                   1/1    Running  0
13m
metrics-facade-599b9d7fcc-gtmgl          1/1    Running  0
9m40s
monitoring-operator-67cc74f844-cdplp     2/2    Running  0
8m11s
nats-0                                   1/1    Running  0
13m
nats-1                                   1/1    Running  0
13m
nats-2                                   1/1    Running  0
12m
nautilus-769f5b74cd-k5jxm                1/1    Running  0
9m42s
nautilus-769f5b74cd-kd9gd                1/1    Running  0
8m59s
openapi-84f6ccd8ff-76kvp                 1/1    Running  0
9m34s
packages-6f59fc67dc-4g2f5                1/1    Running  0
9m52s
polaris-consul-consul-server-0           1/1    Running  0
13m
polaris-consul-consul-server-1           1/1    Running  0
13m
polaris-consul-consul-server-2           1/1    Running  0
13m
polaris-keycloak-0                       1/1    Running  0
8m7s
polaris-keycloak-1                       1/1    Running  0
5m49s
polaris-keycloak-2                       1/1    Running  0
5m15s
polaris-keycloak-db-0                    1/1    Running  0
8m6s
```

```
polaris-keycloak-db-1                     1/1    Running  0
5m49s
polaris-keycloak-db-2                     1/1    Running  0
4m57s
polaris-mongodb-0                         2/2    Running  0
13m
polaris-mongodb-1                         2/2    Running  0
12m
polaris-mongodb-2                         2/2    Running  0
12m
polaris-ui-565f56bf7b-zwr8b               1/1    Running  0
3m19s
polaris-vault-0                           1/1    Running  0
13m
polaris-vault-1                           1/1    Running  0
13m
polaris-vault-2                           1/1    Running  0
13m
public-metrics-6d86d66444-2wbzl           1/1    Running  0
9m30s
storage-backend-metrics-77c5d98dcd-dbhg5 1/1    Running  0
9m44s
storage-provider-78c885f57c-6zcv4         1/1    Running  0
9m36s
telegraf-ds-2l2m9                         1/1    Running  0
7m48s
telegraf-ds-qfzgh                         1/1    Running  0
7m48s
telegraf-ds-shrms                         1/1    Running  0
7m48s
telegraf-rs-bjpkt                         1/1    Running  0
7m48s
telemetry-service-6684696c64-qzfdf        1/1    Running  0
10m
tenancy-6596b6c54d-vmpsm                  1/1    Running  0
10m
traefik-7489dc59f9-6mnst                  1/1    Running  0
3m19s
traefik-7489dc59f9-xrkgg                  1/1    Running  0
3m4s
trident-svc-6c8dc458f5-jswcl              1/1    Running  0
10m
vault-controller-6b954f9b76-gz9nm         1/1    Running  0
11m
```

2. (Facoltativo) per assicurarsi che l'installazione sia completata, è possibile guardare `acc-operator` registra usando il seguente comando.

```
kubectl logs deploy/acc-operator-controller-manager -n netapp-acc-
operator -c manager -f
```

> `accHost` la registrazione del cluster è una delle ultime operazioni e, in caso di errore, la distribuzione non avrà esito negativo. In caso di errore di registrazione del cluster indicato nei registri, è possibile tentare di nuovo la registrazione attraverso il flusso di lavoro add cluster "Nell'interfaccia utente" O API.

3. Una volta eseguiti tutti i pod, verificare che l'installazione sia stata eseguita correttamente (`READY` è `True`) E ottenere la password monouso da utilizzare per l'accesso ad Astra Control Center:

```
kubectl get AstraControlCenter -n netapp-acc
```

Risposta:

```
NAME    UUID                                    VERSION     ADDRESS
READY
astra   ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f  22.08.1-26
10.111.111.111  True
```

> Copiare il valore UUID. La password è `ACC-` Seguito dal valore UUID (`ACC-[UUID]` oppure, in questo esempio, `ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f`).

Impostare l'ingresso per il bilanciamento del carico

È possibile configurare un controller di ingresso Kubernetes che gestisce l'accesso esterno ai servizi, come il bilanciamento del carico in un cluster.

Questa procedura spiega come configurare un controller di ingresso (`ingressType:Generic`). Questa è l'azione predefinita con Astra Control Center. Dopo l'implementazione di Astra Control Center, è necessario configurare il controller di ingresso per esporre Astra Control Center con un URL.

> Se non si desidera configurare un controller di ingresso, è possibile impostarlo `ingressType:AccTraefik`). Astra Control Center utilizza un servizio del tipo "LoadBalancer" (`svc/traefik` Nello spazio dei nomi di Astra Control Center) e richiede l'assegnazione di un indirizzo IP esterno accessibile. Se nel proprio ambiente sono consentiti i bilanciatori di carico e non ne è già configurato uno, è possibile utilizzare MetalLB o un altro servizio di bilanciamento del carico esterno per assegnare un indirizzo IP esterno al servizio. Nella configurazione del server DNS interno, puntare il nome DNS scelto per Astra Control Center sull'indirizzo IP con bilanciamento del carico. Per ulteriori informazioni sul tipo di servizio "LoadBalancer" e sull'ingresso, vedere "Requisiti".

I passaggi variano a seconda del tipo di controller di ingresso utilizzato:

- Ingresso Istio
- Controller di ingresso nginx
- Controller di ingresso OpenShift

Di cosa hai bisogno

- Il necessario "controller di ingresso" dovrebbe essere già implementato.
- Il "classe di ingresso" corrispondente al controller di ingresso dovrebbe già essere creato.
- Si stanno utilizzando versioni di Kubernetes comprese tra v1.19 e v1.22.

Passaggi per l'ingresso di Istio

1. Configurare l'ingresso Istio.

   Questa procedura presuppone che Istio venga distribuito utilizzando il profilo di configurazione "predefinito".

2. Raccogliere o creare il certificato e il file della chiave privata desiderati per Ingress Gateway.

   È possibile utilizzare un certificato CA o autofirmato. Il nome comune deve essere l'indirizzo Astra (FQDN).

   Esempio di comando:

   ```
   openssl req -x509 -nodes -days 365 -newkey rsa:2048
   -keyout tls.key -out tls.crt
   ```

3. Crea un segreto `tls secret name` di tipo `kubernetes.io/tls` Per una chiave privata TLS e un certificato in `istio-system namespace` Come descritto in TLS secrets (segreti TLS).

   Esempio di comando:

   ```
   kubectl create secret tls [tls secret name]
   --key="tls.key"
   --cert="tls.crt" -n istio-system
   ```

   Il nome del segreto deve corrispondere a. `spec.tls.secretName` fornito in `istio-ingress.yaml` file.

4. Implementare una risorsa income in `netapp-acc` (O con nome personalizzato) che utilizza lo spazio dei nomi v1beta1 (deprecato in Kubernetes versione inferiore a o 1.22) o il tipo di risorsa v1 per uno schema obsoleto o per uno schema nuovo:

   Uscita:

```
apiVersion: networking.k8s.io/v1beta1
kind: IngressClass
metadata:
  name: istio
spec:
  controller: istio.io/ingress-controller
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress
  namespace: istio-system
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC addess>
    secretName: [tls secret name]
  rules:
  - host: [ACC addess]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          serviceName: traefik
          servicePort: 80
```

Per il nuovo schema v1, seguire questo esempio:

```
kubectl apply -f istio-Ingress.yaml
```

Uscita:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: istio
spec:
  controller: istio.io/ingress-controller
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress
  namespace: istio-system
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC addess>
    secretName: [tls secret name]
  rules:
  - host: [ACC addess]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: traefik
            port:
              number: 80
```

5. Implementare Astra Control Center come di consueto.
6. Controllare lo stato dell'ingresso:

```
kubectl get ingress -n netapp-acc
```

Risposta:

```
NAME    CLASS HOSTS             ADDRESS         PORTS   AGE
ingress istio astra.example.com 172.16.103.248  80, 443 1h
```

Procedura per il controller di ingresso Nginx

1. Creare un segreto di tipo[`kubernetes.io/tls`] Per una chiave privata TLS e un certificato in `netapp-acc` (o con nome personalizzato) come descritto in "Segreti TLS".

2. Implementare una risorsa income in `netapp-acc` (o con nome personalizzato) namespace utilizzando `v1beta1` (Obsoleto in Kubernetes versione inferiore a o 1.22) o. `v1` tipo di risorsa per uno schema obsoleto o nuovo:

   a. Per a. `v1beta1` schema obsoleto, seguire questo esempio:

```
apiVersion: extensions/v1beta1
Kind: IngressClass
metadata:
  name: ingress-acc
  namespace: [netapp-acc or custom namespace]
  annotations:
    kubernetes.io/ingress.class: [class name for nginx controller]
spec:
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    http:
      paths:
      - backend:
        serviceName: traefik
        servicePort: 80
        pathType: ImplementationSpecific
```

   b. Per `v1` nuovo schema, seguire questo esempio:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netapp-acc-ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: [class name for nginx controller]
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: <ACC addess>
    http:
      paths:
        - path:
          backend:
            service:
              name: traefik
              port:
                number: 80
          pathType: ImplementationSpecific
```

Procedura per il controller di ingresso OpenShift

1. Procurarsi il certificato e ottenere la chiave, il certificato e i file CA pronti per l'uso con il percorso OpenShift.
2. Creare il percorso OpenShift:

```
oc create route edge --service=traefik
--port=web -n [netapp-acc or custom namespace]
--insecure-policy=Redirect --hostname=<ACC address>
--cert=cert.pem --key=key.pem
```

Accedere all'interfaccia utente di Astra Control Center

Dopo aver installato Astra Control Center, si modifica la password dell'amministratore predefinito e si accede alla dashboard dell'interfaccia utente di Astra Control Center.

Fasi

1. In un browser, immettere l'FQDN utilizzato in `astraAddress` in `astra_control_center_min.yaml` CR quando Astra Control Center è stato installato.
2. Accettare i certificati autofirmati quando richiesto.

È possibile creare un certificato personalizzato dopo l'accesso.

3. Nella pagina di accesso di Astra Control Center, inserire il valore utilizzato per `email` poll `astra_control_center_min.yaml` CR quando Astra Control Center è stato installato, seguito dalla password monouso (`ACC-[UUID]`).

Se si immette una password errata per tre volte, l'account admin viene bloccato per 15 minuti.

4. Selezionare **Login**.
5. Modificare la password quando richiesto.

Se si tratta del primo accesso e si dimentica la password e non sono ancora stati creati altri account utente amministrativi, contattare il supporto NetApp per assistenza per il recupero della password.

6. (Facoltativo) rimuovere il certificato TLS autofirmato esistente e sostituirlo con un "Certificato TLS personalizzato firmato da un'autorità di certificazione (CA)".

Risolvere i problemi di installazione

Se uno dei servizi è in `Error` stato, è possibile esaminare i registri. Cercare i codici di risposta API nell'intervallo da 400 a 500. Questi indicano il luogo in cui si è verificato un guasto.

Fasi

1. Per esaminare i registri dell'operatore di Astra Control Center, immettere quanto segue:

```
kubectl logs --follow -n netapp-acc-operator $(kubectl get pods -n
netapp-acc-operator -o name) -c manager
```

Cosa succederà

Completare l'implementazione eseguendo "attività di installazione".

=
:allow-uri-read:

Comprendere le restrizioni delle policy di sicurezza del pod

Astra Control Center supporta la limitazione dei privilegi tramite PSP (Pod Security policy). Le policy di sicurezza Pod consentono di limitare gli utenti o i gruppi in grado di eseguire i container e i privilegi che questi possono avere.

Alcune distribuzioni di Kubernetes, come RKE2, dispongono di un criterio di protezione pod predefinito troppo restrittivo e causano problemi durante l'installazione di Astra Control Center.

È possibile utilizzare le informazioni e gli esempi inclusi qui per comprendere le policy di sicurezza dei pod create da Astra Control Center e configurare le policy di sicurezza dei pod che forniscono la protezione

necessaria senza interferire con le funzioni di Astra Control Center.

PSP installati da Astra Control Center

Astra Control Center crea diverse policy di sicurezza del pod durante l'installazione. Alcune di queste sono permanenti, alcune vengono create durante determinate operazioni e vengono rimosse una volta completata l'operazione.

PSP creati durante l'installazione

Durante l'installazione di Astra Control Center, l'operatore di Astra Control Center installa un criterio di protezione pod personalizzato, un oggetto ruolo e un oggetto RoleBinding per supportare la distribuzione dei servizi Astra Control Center nello spazio dei nomi Astra Control Center.

I nuovi criteri e oggetti hanno i seguenti attributi:

```
kubectl get psp

NAME                            PRIV    CAPS          SELINUX    RUNASUSER
FSGROUP     SUPGROUP    READONLYROOTFS   VOLUMES
avp-psp                         false                 RunAsAny   RunAsAny
RunAsAny    RunAsAny    false            *
netapp-astra-deployment-psp     false                 RunAsAny   RunAsAny
RunAsAny    RunAsAny    false            *

kubectl get role

NAME                                    CREATED AT
netapp-astra-deployment-role            2022-06-27T19:34:58Z

kubectl get rolebinding

NAME                                    ROLE
AGE
netapp-astra-deployment-rb              Role/netapp-astra-deployment-role
32m
```

PSP creati durante le operazioni di backup

Durante le operazioni di backup, Astra Control Center crea un criterio di protezione Pod dinamico, un oggetto ClusterRole e un oggetto RoleBinding. Questi supportano il processo di backup, che avviene in uno spazio dei nomi separato.

I nuovi criteri e oggetti hanno i seguenti attributi:

```
kubectl get psp

NAME                             PRIV    CAPS
SELINUX    RUNASUSER          FSGROUP     SUPGROUP    READONLYROOTFS
VOLUMES
netapp-astra-backup             false   DAC_READ_SEARCH
RunAsAny   RunAsAny           RunAsAny    RunAsAny    false           *

kubectl get role

NAME                   CREATED AT
netapp-astra-backup   2022-07-21T00:00:00Z

kubectl get rolebinding

NAME                   ROLE                          AGE
netapp-astra-backup   Role/netapp-astra-backup   62s
```

PSP creati durante la gestione del cluster

Quando gestisci un cluster, Astra Control Center installa l'operatore di monitoraggio netapp nel cluster gestito. Questo operatore crea un criterio di protezione pod, un oggetto ClusterRole e un oggetto RoleBinding per implementare i servizi di telemetria nello spazio dei nomi Astra Control Center.

I nuovi criteri e oggetti hanno i seguenti attributi:

```
kubectl get psp

NAME                             PRIV    CAPS
SELINUX    RUNASUSER          FSGROUP     SUPGROUP    READONLYROOTFS
VOLUMES
netapp-monitoring-psp-nkmo     true    AUDIT_WRITE,NET_ADMIN,NET_RAW
RunAsAny   RunAsAny           RunAsAny    RunAsAny    false           *

kubectl get role

NAME                                          CREATED AT
netapp-monitoring-role-privileged             2022-07-21T00:00:00Z

kubectl get rolebinding

NAME                                                ROLE
AGE
netapp-monitoring-role-binding-privileged           Role/netapp-
monitoring-role-privileged             2m5s
```

Abilitare la comunicazione di rete tra spazi dei nomi

Alcuni ambienti utilizzano costrutti NetworkPolicy per limitare il traffico tra gli spazi dei nomi. L'operatore di Astra Control Center, Astra Control Center e Astra Plugin per VMware vSphere sono tutti in spazi dei nomi diversi. I servizi in questi diversi spazi dei nomi devono essere in grado di comunicare tra loro. Per attivare questa comunicazione, attenersi alla seguente procedura.

Fasi

1. Eliminare le risorse NetworkPolicy presenti nello spazio dei nomi di Astra Control Center:

```
kubectl get networkpolicy -n netapp-acc
```

2. Per ogni oggetto NetworkPolicy restituito dal comando precedente, utilizzare il seguente comando per eliminarlo. Sostituire <OBJECT_NAME> con il nome dell'oggetto restituito:

```
kubectl delete networkpolicy <OBJECT_NAME> -n netapp-acc
```

3. Applicare il seguente file di risorse per configurare l'oggetto acc-avp-network-policy per consentire ai servizi Astra Plugin per VMware vSphere di effettuare richieste ai servizi Astra Control Center. Sostituire le informazioni tra parentesi <> con quelle dell'ambiente:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: acc-avp-network-policy
  namespace: <ACC_NAMESPACE_NAME> # REPLACE THIS WITH THE ASTRA CONTROL
CENTER NAMESPACE NAME
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: <PLUGIN_NAMESPACE_NAME> #
REPLACE THIS WITH THE ASTRA PLUGIN FOR VMWARE VSPHERE NAMESPACE NAME
```

4. Applicare il seguente file di risorse per configurare l'oggetto acc-operator-network-policy per consentire all'operatore Astra Control Center di comunicare con i servizi Astra Control Center. Sostituire le informazioni tra parentesi <> con quelle dell'ambiente:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: acc-operator-network-policy
  namespace: <ACC_NAMESPACE_NAME> # REPLACE THIS WITH THE ASTRA CONTROL
CENTER NAMESPACE NAME
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: <NETAPP-ACC-OPERATOR> #
REPLACE THIS WITH THE OPERATOR NAMESPACE NAME
```

Rimuovere le limitazioni delle risorse

Alcuni ambienti utilizzano gli oggetti ResourceQuotas e LimitRanges per impedire alle risorse di uno spazio dei nomi di consumare tutta la CPU e la memoria disponibili nel cluster. Astra Control Center non imposta limiti massimi, pertanto non sarà conforme a tali risorse. È necessario rimuoverli dagli spazi dei nomi in cui si intende installare Astra Control Center.

Per recuperare e rimuovere le quote e i limiti, procedere come segue. In questi esempi, l'output del comando viene visualizzato immediatamente dopo il comando.

Fasi

1. Ottieni le quote delle risorse nello spazio dei nomi netapp-acc:

```
kubectl get quota -n netapp-acc
```

   Risposta:

```
NAME          AGE   REQUEST                                          LIMIT
pods-high     16s   requests.cpu: 0/20, requests.memory: 0/100Gi
limits.cpu: 0/200, limits.memory: 0/1000Gi
pods-low      15s   requests.cpu: 0/1, requests.memory: 0/1Gi
limits.cpu: 0/2, limits.memory: 0/2Gi
pods-medium   16s   requests.cpu: 0/10, requests.memory: 0/20Gi
limits.cpu: 0/20, limits.memory: 0/200Gi
```

2. Eliminare tutte le quote delle risorse in base al nome:

```
kubectl delete resourcequota  pods-high -n netapp-acc
```

```
kubectl delete resourcequota  pods-low -n netapp-acc
```

```
kubectl delete resourcequota  pods-medium -n netapp-acc
```

3. Ottieni gli intervalli limite nello spazio dei nomi netapp-acc:

```
kubectl get limits -n netapp-acc
```

Risposta:

```
NAME              CREATED AT
cpu-limit-range   2022-06-27T19:01:23Z
```

4. Eliminare gli intervalli di limiti in base al nome:

```
kubectl delete limitrange cpu-limit-range -n netapp-acc
```

=
:allow-uri-read:

Informazioni sul copyright

Informazioni sul marchio commerciale