Aggiungere un cluster a gestione automatica

Astra Control Service

NetApp
June 04, 2024

Sommario

Aggiungere un cluster a gestione automatica

Aggiungere un cluster pubblico autogestato ad Astra Control Service

Dopo aver configurato l'ambiente, è possibile creare un cluster Kubernetes e aggiungerlo ad Astra Control Service.

Un cluster a gestione automatica è un cluster che viene direttamente provisioning e gestione. Astra Control Service supporta cluster autogestiti eseguiti in un ambiente di cloud pubblico. È possibile aggiungere un cluster a gestione automatica ad Astra Control Service caricando un `kubeconfig.yaml` file. È necessario assicurarsi che il cluster soddisfi i requisiti descritti di seguito.

Distribuzioni Kubernetes supportate

È possibile utilizzare Astra Control Service per gestire i seguenti tipi di cluster pubblici e autogestati:

Distribuzione Kubernetes	Versioni supportate
Kubernetes (upstream)	da 1,27 a 1,29
Rancher Kubernetes Engine (RKE)	RKE 1: Versioni 1.24.17, 1.25.13, 1.26.8 con Rancher Manager 2.7.9 RKE 2: Versioni 1.23.16 e 1.24.13 con Rancher Manager 2.6.13 RKE 2: Versioni 1.24.17, 1.25.14, 1.26.9 con Rancher Manager 2.7.9
Red Hat OpenShift Container Platform	da 4,12 a 4,14

Queste istruzioni presuppongono che sia già stato creato un cluster a gestione automatica.

- Aggiungere il cluster ad Astra Control Service
- Modificare la classe di storage predefinita

Aggiungere il cluster ad Astra Control Service

Dopo aver effettuato l'accesso ad Astra Control Service, il primo passo è iniziare a gestire i cluster. Prima di aggiungere un cluster ad Astra Control Service, è necessario eseguire attività specifiche e assicurarsi che il cluster soddisfi determinati requisiti.

Prima di iniziare

Un cluster a gestione automatica è un cluster che viene direttamente provisioning e gestione. Astra Control Service supporta cluster autogestiti eseguiti in un ambiente di cloud pubblico. I tuoi cluster a gestione autonoma possono utilizzare Astra Control Provivisioner per interfacciarsi con i servizi storage NetApp, oppure possono utilizzare driver Container Storage Interface (CSI) per interfacciarsi con Amazon Elastic Block Store (EBS), Azure Managed Disks e Google Persistent Disk.

Astra Control Service supporta cluster autogestiti che utilizzano le seguenti distribuzioni Kubernetes:

- Red Hat OpenShift Container Platform
- Motore di rancher Kubernetes
- Kubernetes upstream

Il cluster a gestione automatica deve soddisfare i seguenti requisiti:

- Il cluster deve essere accessibile via Internet.
- Se si utilizza o si prevede di utilizzare lo storage abilitato con i driver CSI, i driver CSI appropriati devono essere installati sul cluster. Per ulteriori informazioni sull'utilizzo dei driver CSI per integrare lo storage, consultare la documentazione del servizio di storage.
- Si ha accesso al file cluster kubeconfig che include un solo elemento di contesto. Segui "queste istruzioni" per generare un file kubeconfig.
- Se si aggiunge il cluster utilizzando un file kubeconfig che fa riferimento a un'autorità di certificazione privata, aggiungere la riga seguente al `cluster` sezione del file kubeconfig. In questo modo si permette ad Astra Control di aggiungere il cluster:

  ```
  insecure-skip-tls-verify: true
  ```

- **Solo Rancher**: Quando si gestiscono i cluster di applicazioni in un ambiente Rancher, modificare il contesto predefinito del cluster di applicazioni nel file kubeconfig fornito da Rancher per utilizzare un contesto del piano di controllo invece del contesto del server API Rancher. In questo modo si riduce il carico sul server API Rancher e si migliorano le performance.
- **Requisiti di Astra Control Provisioner**: Dovresti avere un Astra Control Provisioner configurato correttamente, inclusi i suoi componenti Astra Trident, per gestire i cluster.
  - **Rivedi i requisiti dell'ambiente Astra Trident**: Prima di installare o aggiornare Astra Control provisioner, consulta "frontend, backend e configurazioni host supportati".
  - **Abilitare la funzionalità Astra Control Provisioner**: Si consiglia vivamente di installare Astra Trident 23,10 o versione successiva e di abilitare "Astra Control Provivisioner funzionalità di storage avanzate". Nelle prossime release, Astra Control non supporterà Astra Trident se anche Astra Control Provisioner non è abilitato.
  - **Configurare un backend di archiviazione**: Deve essere presente almeno un backend di archiviazione "Configurato in Astra Trident" sul cluster.
  - **Configurare una classe di archiviazione**: Deve essere presente almeno una classe di archiviazione "Configurato in Astra Trident" sul cluster. Se è configurata una classe di archiviazione predefinita, assicurarsi che sia la classe di archiviazione **only** con l'annotazione predefinita.
  - **Configurare un controller snapshot volume e installare una classe snapshot volume**: "Installare un controller per lo snapshot del volume" In modo che le snapshot possano essere

create in Astra Control. "Creare" almeno uno `VolumeSnapshotClass` Utilizzando Astra Trident.

Fasi

1. Nella dashboard, selezionare **Manage Kubernetes cluster** (Gestisci cluster Kubernetes).

   Seguire le istruzioni per aggiungere il cluster.

2. **Provider**: Selezionare la scheda **Other** per aggiungere dettagli sul cluster a gestione automatica.
   a. **Altro**: Fornisci dettagli sul tuo cluster autogestito caricando un `kubeconfig.yaml` o incollando il contenuto di `kubeconfig.yaml` file dagli appunti.

   > Se crei il tuo `kubeconfig` file, è necessario definire solo **un** elemento di contesto al suo interno. Fare riferimento a. "Documentazione Kubernetes" per informazioni sulla creazione `kubeconfig` file.

3. **Nome credenziale**: Fornire un nome per la credenziale del cluster a gestione automatica che si sta caricando su Astra Control. Per impostazione predefinita, il nome della credenziale viene compilato automaticamente come nome del cluster.
4. **Private route identifier**: Questo campo può essere utilizzato solo con cluster privati.
5. Selezionare **Avanti**.
6. (Facoltativo) **Storage**: Facoltativamente, selezionare la classe di storage che si desidera utilizzare per impostazione predefinita per le applicazioni Kubernetes distribuite in questo cluster.
   a. Per selezionare una nuova classe di storage predefinita per il cluster, attivare la casella di controllo **Assegna una nuova classe di storage predefinita**.
   b. Selezionare una nuova classe di storage predefinita dall'elenco.

   > Ogni servizio di storage del cloud provider visualizza le seguenti informazioni su prezzo, performance e resilienza:
   >
   > - Cloud Volumes Service per Google Cloud: Informazioni su prezzi, performance e resilienza
   > - Google Persistent Disk: Non sono disponibili informazioni su prezzi, performance o resilienza
   > - Azure NetApp Files: Informazioni su performance e resilienza
   > - Dischi gestiti Azure: Non sono disponibili informazioni su prezzi, performance o resilienza
   > - Amazon Elastic Block Store: Nessuna informazione su prezzi, performance o resilienza disponibile
   > - Amazon FSX per NetApp ONTAP: Nessuna informazione su prezzi, performance o resilienza disponibile
   > - NetApp Cloud Volumes ONTAP: Non sono disponibili informazioni su prezzi, performance o resilienza

   Ogni classe di storage può utilizzare uno dei seguenti servizi:

   - "Cloud Volumes Service per Google Cloud"

- "Disco persistente di Google"
  - "Azure NetApp Files"
  - "Dischi gestiti da Azure"
  - "Amazon Elastic Block Store"
  - "Amazon FSX per NetApp ONTAP"
  - "NetApp Cloud Volumes ONTAP"

    Scopri di più "Classi di storage per cluster Amazon Web Services". Scopri di più "Classi di storage per cluster AKS". Scopri di più "Classi di storage per cluster GKE".

c. Selezionare **Avanti**.
d. **Review & Approve** (Rivedi e approva): Verifica dei dettagli della configurazione.
e. Selezionare **Add** per aggiungere il cluster ad Astra Control Service.

Modificare la classe di storage predefinita

È possibile modificare la classe di storage predefinita per un cluster.

Modificare la classe di storage predefinita utilizzando Astra Control

È possibile modificare la classe di storage predefinita per un cluster da Astra Control. Se il cluster utilizza un servizio backend di storage precedentemente installato, potrebbe non essere possibile utilizzare questo metodo per modificare la classe di storage predefinita (l'azione **Set as default** non è selezionabile). In questo caso, è possibile Modificare la classe di storage predefinita utilizzando la riga di comando.

Fasi

1. Nell'interfaccia utente di Astra Control Service, selezionare **Clusters**.
2. Nella pagina **Clusters**, selezionare il cluster che si desidera modificare.
3. Selezionare la scheda **Storage**.
4. Selezionare la categoria **classi di storage**.
5. Selezionare il menu **azioni** per la classe di storage che si desidera impostare come predefinita.
6. Selezionare **Imposta come predefinito**.

Modificare la classe di storage predefinita utilizzando la riga di comando

È possibile modificare la classe di storage predefinita per un cluster utilizzando i comandi Kubernetes. Questo metodo funziona indipendentemente dalla configurazione del cluster.

Fasi

1. Accedere al cluster Kubernetes.
2. Elencare le classi di storage nel cluster:

```
kubectl get storageclass
```

3. Rimuovere la designazione predefinita dalla classe di storage predefinita. Sostituire <SC_NAME> con il nome della classe di storage:

```
kubectl patch storageclass <SC_NAME> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-
class":"false"}}}'
```

4. Contrassegna una classe di storage diversa come predefinita. Sostituire <SC_NAME> con il nome della classe di storage:

```
kubectl patch storageclass <SC_NAME> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

5. Confermare la nuova classe di storage predefinita:

```
kubectl get storageclass
```

Aggiungere un cluster privato autogestato ad Astra Control Service

Dopo aver configurato l'ambiente, è possibile creare un cluster Kubernetes e aggiungerlo ad Astra Control Service.

Un cluster a gestione automatica è un cluster che viene direttamente provisioning e gestione. Astra Control Service supporta cluster autogestiti eseguiti in un ambiente di cloud pubblico. È possibile aggiungere un cluster a gestione automatica ad Astra Control Service caricando un `kubeconfig.yaml` file. È necessario assicurarsi che il cluster soddisfi i requisiti descritti di seguito.

Distribuzioni Kubernetes supportate

È possibile utilizzare Astra Control Service per gestire i seguenti tipi di cluster privati a gestione automatica:

Distribuzione Kubernetes	Versioni supportate
Kubernetes (upstream)	da 1,27 a 1,29
Rancher Kubernetes Engine (RKE)	RKE 1: Versioni 1.24.17, 1.25.13, 1.26.8 con Rancher Manager 2.7.9 RKE 2: Versioni 1.23.16 e 1.24.13 con Rancher Manager 2.6.13 RKE 2: Versioni 1.24.17, 1.25.14, 1.26.9 con Rancher Manager 2.7.9
Red Hat OpenShift Container Platform	da 4,12 a 4,14

Queste istruzioni presuppongono che sia già stato creato un cluster privato e che sia stato preparato un metodo sicuro per accedervi in remoto.

Per aggiungere il cluster privato ad Astra Control Service, è necessario eseguire le seguenti operazioni:

1. Installare il connettore Astra
2. Configurare lo storage persistente
3. Aggiungere il cluster privato autogestato ad Astra Control Service

Installare il connettore Astra

Prima di aggiungere un cluster privato, devi installare Astra Connector sul cluster in modo che Astra Control possa comunicare con esso. Fare riferimento a. "Installa la versione precedente di Astra Connector per cluster privati gestiti con flussi di lavoro non nativi di Kubernetes" per istruzioni.

Configurare lo storage persistente

Configurare lo storage persistente per il cluster. Fare riferimento alla documentazione introduttiva per ulteriori informazioni sulla configurazione dello storage persistente:

- "Configurare Microsoft Azure con Azure NetApp Files"
- "Configurare Microsoft Azure con dischi gestiti Azure"
- "Configurare Amazon Web Services"
- "Configurare Google Cloud"

Aggiungere il cluster privato autogestato ad Astra Control Service

È ora possibile aggiungere il cluster privato ad Astra Control Service.

Prima di iniziare

Un cluster a gestione automatica è un cluster che viene direttamente provisioning e gestione. Astra Control Service supporta cluster autogestiti eseguiti in un ambiente di cloud pubblico. I tuoi cluster a gestione autonoma possono utilizzare Astra Control Provivisioner per interfacciarsi con i servizi storage NetApp, oppure possono utilizzare driver Container Storage Interface (CSI) per interfacciarsi con Amazon Elastic Block Store (EBS), Azure Managed Disks e Google Persistent Disk.

Astra Control Service supporta cluster autogestiti che utilizzano le seguenti distribuzioni Kubernetes:

- Red Hat OpenShift Container Platform
- Motore di rancher Kubernetes
- Kubernetes upstream

Il cluster a gestione automatica deve soddisfare i seguenti requisiti:

- Il cluster deve essere accessibile via Internet.
- Se si utilizza o si prevede di utilizzare lo storage abilitato con i driver CSI, i driver CSI appropriati devono essere installati sul cluster. Per ulteriori informazioni sull'utilizzo dei driver CSI per integrare lo storage, consultare la documentazione del servizio di storage.
- Si ha accesso al file cluster kubeconfig che include un solo elemento di contesto. Segui "queste istruzioni" per generare un file kubeconfig.
- Se si aggiunge il cluster utilizzando un file kubeconfig che fa riferimento a un'autorità di certificazione privata, aggiungere la riga seguente al `cluster` sezione del file kubeconfig. In questo modo si permette ad Astra Control di aggiungere il cluster:

  ```
  insecure-skip-tls-verify: true
  ```

- **Solo Rancher**: Quando si gestiscono i cluster di applicazioni in un ambiente Rancher, modificare il contesto predefinito del cluster di applicazioni nel file kubeconfig fornito da Rancher per utilizzare un contesto del piano di controllo invece del contesto del server API Rancher. In questo modo si riduce il carico sul server API Rancher e si migliorano le performance.
- **Requisiti di Astra Control Provisioner**: Dovresti avere un Astra Control Provisioner configurato correttamente, inclusi i suoi componenti Astra Trident, per gestire i cluster.
  - **Rivedi i requisiti dell'ambiente Astra Trident**: Prima di installare o aggiornare Astra Control provisioner, consulta "frontend, backend e configurazioni host supportati".
  - **Abilitare la funzionalità Astra Control Provisioner**: Si consiglia vivamente di installare Astra Trident 23,10 o versione successiva e di abilitare "Astra Control Provivisioner funzionalità di storage avanzate". Nelle prossime release, Astra Control non supporterà Astra Trident se anche Astra Control Provisioner non è abilitato.
  - **Configurare un backend di archiviazione**: Deve essere presente almeno un backend di archiviazione "Configurato in Astra Trident" sul cluster.
  - **Configurare una classe di archiviazione**: Deve essere presente almeno una classe di archiviazione "Configurato in Astra Trident" sul cluster. Se è configurata una classe di archiviazione predefinita, assicurarsi che sia la classe di archiviazione **only** con l'annotazione predefinita.
  - **Configurare un controller snapshot volume e installare una classe snapshot volume**: "Installare un controller per lo snapshot del volume" In modo che le snapshot possano essere

create in Astra Control. "Creare" almeno uno `VolumeSnapshotClass` Utilizzando Astra Trident.

Fasi

1. Nella dashboard, selezionare **Manage Kubernetes cluster** (Gestisci cluster Kubernetes).

   Seguire le istruzioni per aggiungere il cluster.

2. **Provider**: Selezionare la scheda **Other** per aggiungere dettagli sul cluster a gestione automatica.
3. **Altro**: Fornisci dettagli sul tuo cluster autogestito caricando un `kubeconfig.yaml` o incollando il contenuto di `kubeconfig.yaml` file dagli appunti.

   Se crei il tuo `kubeconfig` file, è necessario definire solo **un** elemento di contesto al suo interno. Fare riferimento a. "queste istruzioni" per informazioni sulla creazione `kubeconfig` file.

4. **Nome credenziale**: Fornire un nome per la credenziale del cluster a gestione automatica che si sta caricando su Astra Control. Per impostazione predefinita, il nome della credenziale viene compilato automaticamente come nome del cluster.
5. **Private route identifier**: Immettere l'identificativo di percorso privato, che è possibile ottenere da Astra Connector. Se si esegue una query su Astra Connector tramite `kubectl get astraconnector -n astra-connector` l'identificatore di route privato viene definito `ASTRACONNECTORID`.

   L'identificatore di route privato è il nome associato al connettore Astra che consente la gestione di un cluster Kubernetes privato da parte di Astra. In questo contesto, un cluster privato è un cluster Kubernetes che non espone il proprio server API a Internet.

6. Selezionare **Avanti**.
7. (Facoltativo) **Storage**: Facoltativamente, selezionare la classe di storage che si desidera utilizzare per impostazione predefinita per le applicazioni Kubernetes distribuite in questo cluster.
   a. Per selezionare una nuova classe di storage predefinita per il cluster, attivare la casella di controllo **Assegna una nuova classe di storage predefinita**.
   b. Selezionare una nuova classe di storage predefinita dall'elenco.

Ogni servizio di storage del cloud provider visualizza le seguenti informazioni su prezzo, performance e resilienza:

- Cloud Volumes Service per Google Cloud: Informazioni su prezzi, performance e resilienza
- Google Persistent Disk: Non sono disponibili informazioni su prezzi, performance o resilienza
- Azure NetApp Files: Informazioni su performance e resilienza
- Dischi gestiti Azure: Non sono disponibili informazioni su prezzi, performance o resilienza
- Amazon Elastic Block Store: Nessuna informazione su prezzi, performance o resilienza disponibile
- Amazon FSX per NetApp ONTAP: Nessuna informazione su prezzi, performance o resilienza disponibile
- NetApp Cloud Volumes ONTAP: Non sono disponibili informazioni su prezzi, performance o resilienza

Ogni classe di storage può utilizzare uno dei seguenti servizi:

- "Cloud Volumes Service per Google Cloud"
- "Disco persistente di Google"
- "Azure NetApp Files"
- "Dischi gestiti da Azure"
- "Amazon Elastic Block Store"
- "Amazon FSX per NetApp ONTAP"
- "NetApp Cloud Volumes ONTAP"

Scopri di più "Classi di storage per cluster Amazon Web Services". Scopri di più "Classi di storage per cluster AKS". Scopri di più "Classi di storage per cluster GKE".

c. Selezionare **Avanti**.
d. **Review & Approve** (Rivedi e approva): Verifica dei dettagli della configurazione.
e. Selezionare **Add** per aggiungere il cluster ad Astra Control Service.

Modificare la classe di storage predefinita

È possibile modificare la classe di storage predefinita per un cluster.

Modificare la classe di storage predefinita utilizzando Astra Control

È possibile modificare la classe di storage predefinita per un cluster da Astra Control. Se il cluster utilizza un servizio backend di storage precedentemente installato, potrebbe non essere possibile utilizzare questo metodo per modificare la classe di storage predefinita (l'azione **Set as default** non è selezionabile). In questo caso, è possibile Modificare la classe di storage predefinita utilizzando la riga di comando.

Fasi

1. Nell'interfaccia utente di Astra Control Service, selezionare **Clusters**.

2. Nella pagina **Clusters**, selezionare il cluster che si desidera modificare.
3. Selezionare la scheda **Storage**.
4. Selezionare la categoria **classi di storage**.
5. Selezionare il menu **azioni** per la classe di storage che si desidera impostare come predefinita.
6. Selezionare **Imposta come predefinito**.

Modificare la classe di storage predefinita utilizzando la riga di comando

È possibile modificare la classe di storage predefinita per un cluster utilizzando i comandi Kubernetes. Questo metodo funziona indipendentemente dalla configurazione del cluster.

Fasi

1. Accedere al cluster Kubernetes.
2. Elencare le classi di storage nel cluster:

```
kubectl get storageclass
```

3. Rimuovere la designazione predefinita dalla classe di storage predefinita. Sostituire <SC_NAME> con il nome della classe di storage:

```
kubectl patch storageclass <SC_NAME> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-
class":"false"}}}'
```

4. Contrassegna una classe di storage diversa come predefinita. Sostituire <SC_NAME> con il nome della classe di storage:

```
kubectl patch storageclass <SC_NAME> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

5. Confermare la nuova classe di storage predefinita:

```
kubectl get storageclass
```

Controllare la versione di Astra Trident

Per aggiungere un cluster a gestione autonoma che utilizzi Astra Control Provivisioner o Astra Trident per i servizi di storage, assicurati che la versione installata di Astra Trident sia la 23,10 o più recente.

Fasi

1. Determina la versione di Astra Trident che stai utilizzando:

```
kubectl get tridentversions -n trident
```

Se Astra Trident è installato, viene visualizzato un output simile a quanto segue:

```
NAME      VERSION
trident   24.02.0
```

Se Astra Trident non è installato, viene visualizzato un output simile a quanto segue:

```
error: the server doesn't have a resource type "tridentversions"
```

2. Effettuare una delle seguenti operazioni:
   - Se utilizzi Astra Trident 23,01 o versione precedente, utilizza questi elementi "istruzioni" Per effettuare l'aggiornamento a una versione più recente di Astra Trident prima di effettuare l'aggiornamento a Astra Control Provivisioner. È possibile "eseguire un aggiornamento diretto" A Astra Control Provivisioner 24,02 se il tuo Astra Trident si trova all'interno di una finestra a quattro release della versione 24,02. Ad esempio, puoi eseguire l'upgrade direttamente da Astra Trident 23,04 a Astra Control Provisioner 24,02.
   - Se stai eseguendo Astra Trident 23,10 o versione successiva, verifica che Astra Control provisioner sia stato "attivato". Astra Control Provisioner non funzionerà con le versioni di Astra Control Center precedenti alla 23,10. "Aggiorna Astra Control provisioner" In modo che abbia la stessa versione di Astra Control Center che stai effettuando l'aggiornamento per accedere alle funzionalità più recenti.
3. Assicurarsi che i pod siano in funzione:

```
kubectl get pods -n trident
```

4. Controllare se le classi di storage utilizzano i driver Astra Trident supportati. Il nome del provider deve essere `csi.trident.netapp.io`. Fare riferimento al seguente esempio:

```
kubectl get sc
```

Esempio di risposta:

```
NAME                    PROVISIONER                    RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)    csi.trident.netapp.io          Delete
Immediate           true                   5d23h
```

Creare un file kubeconfig

È possibile aggiungere un cluster ad Astra Control Service utilizzando un file kubeconfig. A seconda del tipo di cluster che si desidera aggiungere, potrebbe essere necessario creare manualmente un file kubeconfig per il cluster utilizzando passaggi specifici.

- Creare un file kubeconfig per i cluster Amazon EKS
- Creare un file kubeconfig per Red Hat OpenShift Service su cluster AWS (ROSA)
- Creare un file kubeconfig per altri tipi di cluster

Creare un file kubeconfig per i cluster Amazon EKS

Segui queste istruzioni per creare un file kubeconfig e un token secret permanente per i cluster Amazon EKS. Per i cluster ospitati in EKS è necessario un token secret permanente.

Fasi

1. Seguire le istruzioni nella documentazione di Amazon per generare un file kubeconfig:

   "Creazione o aggiornamento di un file kubeconfig per un cluster Amazon EKS"

2. Creare un account di servizio come segue:

   a. Creare un file di account del servizio denominato `astracontrol-service-account.yaml`.

      Modificare il nome dell'account di servizio in base alle necessità. Lo spazio dei nomi `kube-system` è necessario per questi passaggi. Se si modifica il nome dell'account di servizio, è necessario apportare le stesse modifiche nei seguenti passaggi.

```
<strong>astracontrol-service-account.yaml</strong>
```

+

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: astra-admin-account
  namespace: kube-system
```

3. Applicare l'account del servizio:

```
kubectl apply -f astracontrol-service-account.yaml
```

4. Creare un `ClusterRoleBinding` file chiamato `astracontrol-clusterrolebinding.yaml`.

```
<strong>astracontrol-clusterrolebinding.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astra-admin-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: astra-admin-account
  namespace: kube-system
```

5. Applicare l'associazione del ruolo del cluster:

```
kubectl apply -f astracontrol-clusterrolebinding.yaml
```

6. Creare un file token secret dell'account di servizio chiamato `astracontrol-secret.yaml`.

```
<strong>astracontrol-secret.yaml</strong>
```

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: astra-admin-account
  name: astra-admin-account
  namespace: kube-system
type: kubernetes.io/service-account-token
```

7. Applicare il token secret:

```
kubectl apply -f astracontrol-secret.yaml
```

8. Recuperare il token secret:

```
kubectl get secret astra-admin-account -n kube-system -o
jsonpath='{.data.token}' | base64 -d
```

9. Sostituire `user` Sezione del file kubeconfig AWS EKS con il token, come mostrato nell'esempio seguente:

```
user:
    token: k8s-aws-
v1.aHR0cHM6Ly9zdHMudXMtd2VzdC0yLmFtYXpvbmF3cy5jb20vP0FjdGlvbj1HZXRDYWxsZ
XJJZGVudGl0eSZWZXJzaW9uPTIwMTEtMDYtMTUmWC1BbXotQWxnb3JpdGhtPUFXUzQtSE1BQ
y1TSEEyNTYmWC1BbXotQ3JlZGVudGlhbD1BS0lBM1JEWDdKU0haWU9LSEQ2SyUyRjIwMjMwN
DAzJTJGdXMtd2VzdC0yJTJGc3RzJTJGYXdzNF9yZXF1ZXN0JlgtQW16LURhdGU9MjAyMzA0M
DNUMjA0MzQwWiZYLUFtei1FeHBpcmVzPTYwJlgtQW16LVNpZ25lZEhlYWRlcnM9aG9zdCUzQ
ngtazhzLWF3cy1pZCZYLUFtei1TaWduYXR1cmU9YjU4ZWM0NzdiM2NkZGYxNGRhNzU4MGI2Z
WQ2zY2NzI2YWIwM2UyNThjMjRhNTJjNmVhNjc4MTRlNjJkOTg2Mg
```

Creare un file kubeconfig per Red Hat OpenShift Service su cluster AWS (ROSA)

Segui queste istruzioni per creare un file kubeconfig per Red Hat OpenShift Service su cluster AWS (ROSA).

Fasi

1. Accedere al cluster ROSA.
2. Creare un account di servizio:

```
oc create sa astracontrol-service-account
```

3. Aggiungere un ruolo cluster:

```
oc adm policy add-cluster-role-to-user cluster-admin -z astracontrol-
service-account
```

4. Utilizzando l'esempio seguente, creare un file di configurazione segreto dell'account di servizio:

```
<strong>secret-astra-sa.yaml</strong>
```

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-astracontrol-service-account
  annotations:
    kubernetes.io/service-account.name: "astracontrol-service-account"
type: kubernetes.io/service-account-token
```

5. Creare il segreto:

```
oc create -f secret-astra-sa.yaml
```

6. Modificare l'account di servizio creato e aggiungere il nome segreto dell'account del servizio Astra Control a. `secrets` sezione:

```
oc edit sa astracontrol-service-account
```

```
apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-dvfcd
kind: ServiceAccount
metadata:
  creationTimestamp: "2023-08-04T04:18:30Z"
  name: astracontrol-service-account
  namespace: default
  resourceVersion: "169770"
  uid: 965fa151-923f-4fbd-9289-30cad15998ac
secrets:
- name: astracontrol-service-account-dockercfg-dvfcd
- name: secret-astracontrol-service-account ####ADD THIS ONLY####
```

7. Elencare i segreti dell'account di servizio, sostituendo `<CONTEXT>` con il contesto corretto per l'installazione:

```
kubectl get serviceaccount astracontrol-service-account --context
<CONTEXT> --namespace default -o json
```

La fine dell'output dovrebbe essere simile a quanto segue:

```
"secrets": [
{ "name": "astracontrol-service-account-dockercfg-dvfcd"},
{ "name": "secret-astracontrol-service-account"}
]
```

Gli indici di ciascun elemento in `secrets` l'array inizia con 0. Nell'esempio precedente, l'indice per `astracontrol-service-account-dockercfg-dvfcd` sarebbe 0 e l'indice per `secret-astracontrol-service-account` sarebbe 1. Nell'output, annotare il numero dell'indice per il segreto dell'account del servizio. Questo numero di indice sarà necessario nella fase successiva.

8. Generare il kubeconfig come segue:
   a. Creare un `create-kubeconfig.sh` file. Sostituire `TOKEN_INDEX` all'inizio del seguente script con il valore corretto.

```
<strong>create-kubeconfig.sh</strong>
```

```
# Update these to match your environment.
# Replace TOKEN_INDEX with the correct value
# from the output in the previous step. If you
# didn't change anything else above, don't change
# anything else here.

SERVICE_ACCOUNT_NAME=astracontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astracontrol
KUBECONFIG_FILE='kubeconfig-sa'

CONTEXT=$(kubectl config current-context)

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.secrets[TOKEN_INDEX].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.data.token}')

TOKEN=$(echo ${TOKEN_DATA} | base64 -d)

# Create dedicated kubeconfig
# Create a full copy
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp
```

```
# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp

# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token
-user

# Set context to correct namespace
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}

# Flatten/minify kubeconfig
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  view --flatten --minify > ${KUBECONFIG_FILE}

# Remove tmp
rm ${KUBECONFIG_FILE}.full.tmp
rm ${KUBECONFIG_FILE}.tmp
```

b. Eseguire la sorgente dei comandi per applicarli al cluster Kubernetes.

```
source create-kubeconfig.sh
```

9. (Facoltativo) rinominare il kubeconfig con un nome significativo per il cluster.

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

Creare un file kubeconfig per altri tipi di cluster

Segui queste istruzioni per creare un file kubeconfig con ruolo limitato o esteso per i cluster Rancher, Upstream Kubernetes e Red Hat OpenShift.

Per i cluster gestiti utilizzando kubeconfig, è possibile creare un'autorizzazione limitata o un ruolo di amministratore di autorizzazioni esteso per Astra Control Service.

Questa procedura consente di creare una configurazione separata se uno dei seguenti scenari si applica al proprio ambiente:

- Si desidera limitare le autorizzazioni di Astra Control sui cluster gestiti
- Si utilizzano più contesti e non è possibile utilizzare il kubeconfig di Astra Control predefinito configurato durante l'installazione oppure un ruolo limitato con un singolo contesto non funziona nell'ambiente

Prima di iniziare

Prima di completare la procedura, assicurarsi di disporre dei seguenti elementi per il cluster che si desidera gestire:

- R "versione supportata" di kubectl è installato.
- Kubectl accesso al cluster che si intende aggiungere e gestire con Astra Control Service

> Per questa procedura, non è necessario l'accesso kubectl al cluster che esegue Astra Control Service.

- Un kubeconfig attivo per il cluster che si intende gestire con i diritti di amministratore del cluster per il contesto attivo

Fasi

1. Creare un account di servizio:
   a. Creare un file di account del servizio denominato `astracontrol-service-account.yaml`.

```
<strong>astracontrol-service-account.yaml</strong>
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: astracontrol-service-account
  namespace: default
```

   b. Applicare l'account del servizio:

```
kubectl apply -f astracontrol-service-account.yaml
```

2. Creare uno dei seguenti ruoli del cluster con autorizzazioni sufficienti per la gestione di un cluster da parte di Astra Control:

Ruolo cluster limitato

Questo ruolo contiene le autorizzazioni minime necessarie per gestire un cluster da Astra Control:

a. Creare un `ClusterRole` file chiamato, ad esempio, `astra-admin-account.yaml`.

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:

# Get, List, Create, and Update all resources
# Necessary to backup and restore all resources in an app
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - get
  - list
  - create
  - patch

# Delete Resources
# Necessary for in-place restore and AppMirror failover
- apiGroups:
  - ""
  - apps
  - autoscaling
  - batch
  - crd.projectcalico.org
  - extensions
  - networking.k8s.io
  - policy
  - rbac.authorization.k8s.io
  - snapshot.storage.k8s.io
  - trident.netapp.io
  resources:
  - configmaps
  - cronjobs
  - daemonsets
  - deployments
```

```
    - horizontalpodautoscalers
    - ingresses
    - jobs
    - namespaces
    - networkpolicies
    - persistentvolumeclaims
    - poddisruptionbudgets
    - pods
    - podtemplates
    - replicasets
    - replicationcontrollers
    - replicationcontrollers/scale
    - rolebindings
    - roles
    - secrets
    - serviceaccounts
    - services
    - statefulsets
    - tridentmirrorrelationships
    - tridentsnapshotinfos
    - volumesnapshots
    - volumesnapshotcontents
    verbs:
    - delete

  # Watch resources
  # Necessary to monitor progress
  - apiGroups:
    - ""
    resources:
    - pods
    - replicationcontrollers
    - replicationcontrollers/scale
    verbs:
    - watch

  # Update resources
  - apiGroups:
    - ""
    - build.openshift.io
    - image.openshift.io
    resources:
    - builds/details
    - replicationcontrollers
    - replicationcontrollers/scale
    - imagestreams/layers
```

```
  - imagestreamtags
  - imagetags
  verbs:
  - update
```

b. (Solo per i cluster OpenShift) aggiungere quanto segue alla fine di `astra-admin-account.yaml` file:

```
# OpenShift security
- apiGroups:
  - security.openshift.io
  resources:
  - securitycontextconstraints
  verbs:
  - use
  - update
```

c. Applicare il ruolo del cluster:

```
kubectl apply -f astra-admin-account.yaml
```

Ruolo cluster esteso

Questo ruolo contiene autorizzazioni estese per un cluster da gestire con Astra Control. È possibile utilizzare questo ruolo se si utilizzano più contesti e non è possibile utilizzare il kubeconfig di Astra Control predefinito configurato durante l'installazione oppure se un ruolo limitato con un singolo contesto non funziona nell'ambiente:

Quanto segue `ClusterRole` I passaggi sono un esempio generale di Kubernetes. Consultare la documentazione della distribuzione Kubernetes per istruzioni specifiche sull'ambiente in uso.

a. Creare un `ClusterRole` file chiamato, ad esempio, `astra-admin-account.yaml`.

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

b. Applicare il ruolo del cluster:

```
kubectl apply -f astra-admin-account.yaml
```

3. Creare l'associazione del ruolo del cluster all'account del servizio per il ruolo del cluster:

a. Creare un `ClusterRoleBinding` file chiamato `astracontrol-clusterrolebinding.yaml`.

```
<strong>astracontrol-clusterrolebinding.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astracontrol-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: astra-admin-account
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

b. Applicare l'associazione del ruolo del cluster:

```
kubectl apply -f astracontrol-clusterrolebinding.yaml
```

4. Creare e applicare il token secret:
   a. Creare un file token secret chiamato `secret-astracontrol-service-account.yaml`.

```
<strong>secret-astracontrol-service-account.yaml</strong>
```

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-astracontrol-service-account
  namespace: default
  annotations:
    kubernetes.io/service-account.name: "astracontrol-service-
account"
type: kubernetes.io/service-account-token
```

   b. Applicare il token secret:

```
kubectl apply -f secret-astracontrol-service-account.yaml
```

5. Aggiungere il token secret all'account del servizio aggiungendo il nome a `secrets` array (l'ultima riga dell'esempio seguente):

```
kubectl edit sa astracontrol-service-account
```

```
apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-48xhx
kind: ServiceAccount
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},
"name":"astracontrol-service-account","namespace":"default"}}
  creationTimestamp: "2023-06-14T15:25:45Z"
  name: astracontrol-service-account
  namespace: default
  resourceVersion: "2767069"
  uid: 2ce068c4-810e-4a96-ada3-49cbf9ec3f89
secrets:
- name: astracontrol-service-account-dockercfg-48xhx
<strong>- name: secret-astracontrol-service-account</strong>
```

6. Elencare i segreti dell'account di servizio, sostituendo `<context>` con il contesto corretto per l'installazione:

```
kubectl get serviceaccount astracontrol-service-account --context
<context> --namespace default -o json
```

La fine dell'output dovrebbe essere simile a quanto segue:

```
"secrets": [
{ "name": "astracontrol-service-account-dockercfg-48xhx"},
{ "name": "secret-astracontrol-service-account"}
]
```

Gli indici di ciascun elemento in `secrets` l'array inizia con 0. Nell'esempio precedente, l'indice per `astracontrol-service-account-dockercfg-48xhx` sarebbe 0 e l'indice per `secret-astracontrol-service-account` sarebbe 1. Nell'output, annotare il numero dell'indice per il segreto dell'account del servizio. Questo numero di indice è necessario nel passaggio successivo.

7. Generare il kubeconfig come segue:
   a. Creare un `create-kubeconfig.sh` file.
   b. Sostituire `TOKEN_INDEX` all'inizio del seguente script con il valore corretto.

```
<strong>create-kubeconfig.sh</strong>
```

```
# Update these to match your environment.
# Replace TOKEN_INDEX with the correct value
# from the output in the previous step. If you
# didn't change anything else above, don't change
# anything else here.

SERVICE_ACCOUNT_NAME=astracontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astracontrol
KUBECONFIG_FILE='kubeconfig-sa'

CONTEXT=$(kubectl config current-context)

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.secrets[TOKEN_INDEX].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.data.token}')

TOKEN=$(echo ${TOKEN_DATA} | base64 -d)

# Create dedicated kubeconfig
# Create a full copy
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp

# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp

# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user
```

```
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token-
user

# Set context to correct namespace
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}

# Flatten/minify kubeconfig
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  view --flatten --minify > ${KUBECONFIG_FILE}

# Remove tmp
rm ${KUBECONFIG_FILE}.full.tmp
rm ${KUBECONFIG_FILE}.tmp
```

c. Eseguire la sorgente dei comandi per applicarli al cluster Kubernetes.

```
source create-kubeconfig.sh
```

8. (Facoltativo) rinominare il kubeconfig con un nome significativo per il cluster.

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

Informazioni sul copyright

Informazioni sul marchio commerciale