Ajoutez un cluster autogéré

Astra Control Service

NetApp
April 24, 2024

Sommaire

Ajoutez un cluster autogéré

Ajoutez un cluster public autogéré à Astra Control Service

Une fois votre environnement configuré, vous êtes prêt à créer un cluster Kubernetes, puis à l'ajouter à Astra Control Service.

Un cluster autogéré est un cluster que vous provisionnez et gérez directement. ASTRA Control Service prend en charge les clusters autogérés qui s'exécutent dans un environnement de cloud public. Vous pouvez ajouter un cluster auto-géré au service Astra Control en téléchargeant un `kubeconfig.yaml` fichier. Vous devez vous assurer que le cluster répond aux exigences décrites ici.

Distributions Kubernetes prises en charge

Vous pouvez utiliser Astra Control Service pour gérer les types suivants de clusters publics et autogérés :

Distribution Kubernetes	Versions prises en charge
Kubernetes (en amont)	1.27 à 1.29
Rancher Kubernetes Engine (RKE)	RKE 1 : versions 1.24.17, 1.25.13, 1.26.8 avec Rancher Manager 2.7.9 RKE 2 : versions 1.23.16 et 1.24.13 avec Rancher Manager 2.6.13 RKE 2 : versions 1.24.17, 1.25.14, 1.26.9 avec Rancher Manager 2.7.9
Plateforme de conteneurs Red Hat OpenShift	4.12 à 4.14

Ces instructions supposent que vous avez déjà créé un cluster autogéré.

- Ajoutez le cluster à Astra Control Service
- Modifiez la classe de stockage par défaut

Ajoutez le cluster à Astra Control Service

Une fois connecté au service Astra Control, la première étape consiste à commencer à gérer vos clusters. Avant d'ajouter un cluster à Astra Control Service, vous devez effectuer des tâches spécifiques et vous assurer qu'il répond à certaines exigences.

Avant de commencer

Un cluster autogéré est un cluster que vous provisionnez et gérez directement. ASTRA Control Service prend en charge les clusters autogérés qui s'exécutent dans un environnement de cloud public. Vos clusters autogérés peuvent utiliser Astra Control Provisioner pour s'interfacer avec les services de stockage NetApp ou des pilotes Container Storage interface (CSI) pour s'interfacer avec Amazon Elastic Block Store (EBS), les disques gérés Azure et le service Google persistent Disk.

ASTRA Control Service prend en charge les clusters autogérés qui utilisent les distributions Kubernetes suivantes :

- Plateforme de conteneurs Red Hat OpenShift
- Moteur rancher Kubernetes
- Kubernetes en amont

Votre cluster autogéré doit répondre aux exigences suivantes :

- Le cluster doit être accessible via Internet.
- Si vous utilisez ou prévoyez d'utiliser le stockage activé avec des pilotes CSI, les pilotes CSI appropriés doivent être installés sur le cluster. Pour plus d'informations sur l'utilisation des pilotes CSI pour intégrer le stockage, reportez-vous à la documentation de votre service de stockage.
- Vous avez accès au fichier kubeconfig du cluster qui ne contient qu'un seul élément de contexte. Suivre "ces instructions" pour générer un fichier kubeconfig.
- Si vous ajoutez le cluster à l'aide d'un fichier kubeconfig qui fait référence à une autorité de certification privée (CA), ajoutez la ligne suivante au `cluster` section du fichier kubeconfig. Cela permet à Astra Control d'ajouter le cluster :

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

- **Rancher uniquement**: Lorsque vous gérez des clusters d'applications dans un environnement Rancher, modifiez le contexte par défaut du cluster d'applications dans le fichier kubeconfig fourni par Rancher pour utiliser un contexte de plan de contrôle au lieu du contexte du serveur d'API Rancher. La charge est réduite sur le serveur API Rancher et les performances sont améliorées.
- **Exigences du mécanisme de provisionnement Astra Control** : vous devez avoir un mécanisme de provisionnement Astra Control correctement configuré, y compris ses composants Astra Trident, pour gérer les clusters.
  - **Revoir les exigences de l'environnement Astra Trident** : avant d'installer ou de mettre à niveau Astra Control Provisioner, consultez le "systèmes front-end, systèmes back-end et configurations hôte pris en charge".
  - **Activer la fonctionnalité Astra Control Provisioner** : il est fortement recommandé d'installer Astra Trident 23.10 ou version ultérieure et de l'activer "Fonctionnalité de stockage avancée Astra Control Provisioner". Dans les prochaines versions, Astra Control ne prendra pas en charge Astra Trident si le mécanisme de provisionnement Astra Control n'est pas également activé.
  - **Configurer un back-end de stockage** : au moins un back-end de stockage doit l'être "Configuré dans Astra Trident" sur le cluster.
  - **Configurer une classe de stockage** : au moins une classe de stockage doit être "Configuré dans Astra Trident" sur le cluster. Si une classe de stockage par défaut est configurée, assurez-vous qu'il s'agit de la classe de stockage **Only** qui possède l'annotation par défaut.

- **Configurer un contrôleur de snapshot de volume et installer une classe de snapshot de volume** : "Installez un contrôleur de snapshot de volume" Il est ainsi possible de créer des snapshots dans Astra Control. "Création" au moins un `VolumeSnapshotClass` Avec Astra Trident.

Étapes

1. Dans le Tableau de bord, sélectionnez **Manage Kubernetes cluster**.

   Suivez les invites pour ajouter le cluster.

2. **Fournisseur** : sélectionnez l'onglet **autre** pour ajouter des détails sur votre cluster auto-géré.
   a. **Autre**: Fournir des détails sur votre cluster auto-géré en téléchargeant un `kubeconfig.yaml` ou en collant le contenu du `kubeconfig.yaml` fichier à partir du presse-papiers.

   > Si vous créez la vôtre `kubeconfig` fichier, vous ne devez définir que **un** élément de contexte dans celui-ci. Reportez-vous à la section "Documentation Kubernetes" pour plus d'informations sur la création `kubeconfig` fichiers.

3. **Nom d'identification** : indiquez un nom pour les informations d'identification de cluster autogérées que vous téléchargez sur Astra Control. Par défaut, le nom des identifiants est automatiquement renseigné comme nom du cluster.
4. **ID de route privée** : ce champ est destiné uniquement aux clusters privés.
5. Sélectionnez **Suivant**.
6. (Facultatif) **Storage** : si vous le souhaitez, sélectionnez la classe de stockage que les applications Kubernetes déployées sur ce cluster doivent utiliser par défaut.
   a. Pour sélectionner une nouvelle classe de stockage par défaut pour le cluster, cochez la case **affecter une nouvelle classe de stockage par défaut**.
   b. Sélectionnez une nouvelle classe de stockage par défaut dans la liste.

   > Chaque fournisseur de service de stockage cloud affiche les informations suivantes en matière de prix, de performance et de résilience :
   >
   > - Cloud Volumes Service pour Google Cloud : informations sur le prix, la performance et la résilience
   > - Google persistent Disk : pas d'informations sur le prix, la performance ou la résilience disponibles
   > - Azure NetApp Files : informations sur les performances et la résilience
   > - Azure Managed Disks : aucun prix, performances ou résilience disponibles
   > - Amazon Elastic Block Store : pas d'informations disponibles sur le prix, la performance ou la résilience
   > - Amazon FSX pour NetApp ONTAP : aucune information disponible concernant le prix, les performances ou la résilience
   > - NetApp Cloud Volumes ONTAP : aucune information disponible sur le prix, les performances ou la résilience

   Chaque classe de stockage peut utiliser l'un des services suivants :

- "Cloud Volumes Service pour Google Cloud"
- "Disque persistant Google"
  - "Azure NetApp Files"
  - "Disques gérés Azure"
  - "Amazon Elastic Block Store"
  - "Amazon FSX pour NetApp ONTAP"
  - "NetApp Cloud Volumes ONTAP"

    En savoir plus sur "Classes de stockage pour les clusters Amazon Web Services". En savoir plus sur "Classes de stockage pour les clusters AKS". En savoir plus sur "Classes de stockage pour clusters GKE".

c. Sélectionnez **Suivant**.

d. **Revoir et approuver** : consultez les détails de la configuration.

e. Sélectionnez **Ajouter** pour ajouter le cluster à Astra Control Service.

Modifiez la classe de stockage par défaut

Vous pouvez modifier la classe de stockage par défaut d'un cluster.

Modifiez la classe de stockage par défaut avec Astra Control

Vous pouvez modifier la classe de stockage par défaut d'un cluster depuis Astra Control. Si votre cluster utilise un service back-end de stockage installé précédemment, il se peut que vous ne puissiez pas utiliser cette méthode pour modifier la classe de stockage par défaut (l'action **Set as default** n'est pas sélectionnable). Dans ce cas, vous pouvez Modifiez la classe de stockage par défaut à l'aide de la ligne de commande.

Étapes

1. Dans l'interface utilisateur du service de contrôle Astra, sélectionnez **clusters**.
2. Sur la page **clusters**, sélectionnez le cluster que vous souhaitez modifier.
3. Sélectionnez l'onglet **stockage**.
4. Sélectionnez la catégorie **classes de stockage**.
5. Sélectionnez le menu **actions** pour la classe de stockage que vous souhaitez définir par défaut.
6. Sélectionnez **définir comme valeur par défaut**.

Modifiez la classe de stockage par défaut à l'aide de la ligne de commande

Vous pouvez modifier la classe de stockage par défaut d'un cluster à l'aide des commandes Kubernetes. Cette méthode fonctionne quelle que soit la configuration du cluster.

Étapes

1. Connectez-vous à votre cluster Kubernetes.
2. Lister les classes de stockage de votre cluster :

```
kubectl get storageclass
```

3. Supprimez la désignation par défaut de la classe de stockage par défaut. Remplacez <SC_NAME> par le nom de la classe de stockage :

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

4. Sélectionnez par défaut une classe de stockage différente. Remplacez <SC_NAME> par le nom de la classe de stockage :

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

5. Confirmez la nouvelle classe de stockage par défaut :

```
kubectl get storageclass
```

Ajoutez un cluster privé autogéré à Astra Control Service

Une fois votre environnement configuré, vous êtes prêt à créer un cluster Kubernetes, puis à l'ajouter à Astra Control Service.

Un cluster autogéré est un cluster que vous provisionnez et gérez directement. ASTRA Control Service prend en charge les clusters autogérés qui s'exécutent dans un environnement de cloud public. Vous pouvez ajouter un cluster auto-géré au service Astra Control en téléchargeant un `kubeconfig.yaml` fichier. Vous devez vous assurer que le cluster répond aux exigences décrites ici.

Distributions Kubernetes prises en charge

Vous pouvez utiliser Astra Control Service pour gérer les types suivants de clusters privés et autogérés :

Distribution Kubernetes	Versions prises en charge
Kubernetes (en amont)	1.27 à 1.29
Rancher Kubernetes Engine (RKE)	RKE 1 : versions 1.24.17, 1.25.13, 1.26.8 avec Rancher Manager 2.7.9 RKE 2 : versions 1.23.16 et 1.24.13 avec Rancher Manager 2.6.13 RKE 2 : versions 1.24.17, 1.25.14, 1.26.9 avec Rancher Manager 2.7.9
Plateforme de conteneurs Red Hat OpenShift	4.12 à 4.14

Ces instructions supposent que vous avez déjà créé un cluster privé et préparé une méthode sécurisée pour y accéder à distance.

Pour ajouter votre cluster privé à Astra Control Service, vous devez effectuer les tâches suivantes :

1. Poser le connecteur Astra
2. Configuration du stockage persistant
3. Ajoutez le cluster privé autogéré à Astra Control Service

Poser le connecteur Astra

Avant d'ajouter un cluster privé, vous devez installer Astra Connector sur le cluster afin qu'Astra Control puisse communiquer avec lui. Reportez-vous à la section "Installez la version précédente d'Astra Connector pour les clusters privés gérés avec des workflows non natifs Kubernetes" pour obtenir des instructions.

Configuration du stockage persistant

Configurer le stockage persistant pour le cluster. Pour plus d'informations sur la configuration du stockage persistant, reportez-vous à la documentation de mise en route :

- "Configuration de Microsoft Azure avec Azure NetApp Files"
- "Configuration de Microsoft Azure avec des disques gérés Azure"
- "Configurer Amazon Web Services"
- "Configurez Google Cloud"

Ajoutez le cluster privé autogéré à Astra Control Service

Vous pouvez maintenant ajouter le cluster privé à Astra Control Service.

Avant de commencer

Un cluster autogéré est un cluster que vous provisionnez et gérez directement. ASTRA Control Service prend en charge les clusters autogérés qui s'exécutent dans un environnement de cloud public. Vos clusters autogérés peuvent utiliser Astra Control Provisioner pour s'interfacer avec les services de stockage NetApp ou des pilotes Container Storage interface (CSI) pour s'interfacer avec Amazon Elastic Block Store (EBS), les disques gérés Azure et le service Google persistent Disk.

ASTRA Control Service prend en charge les clusters autogérés qui utilisent les distributions Kubernetes suivantes :

- Plateforme de conteneurs Red Hat OpenShift
- Moteur rancher Kubernetes
- Kubernetes en amont

Votre cluster autogéré doit répondre aux exigences suivantes :

- Le cluster doit être accessible via Internet.
- Si vous utilisez ou prévoyez d'utiliser le stockage activé avec des pilotes CSI, les pilotes CSI appropriés doivent être installés sur le cluster. Pour plus d'informations sur l'utilisation des pilotes CSI pour intégrer le stockage, reportez-vous à la documentation de votre service de stockage.
- Vous avez accès au fichier kubeconfig du cluster qui ne contient qu'un seul élément de contexte. Suivre "ces instructions" pour générer un fichier kubeconfig.
- Si vous ajoutez le cluster à l'aide d'un fichier kubeconfig qui fait référence à une autorité de certification privée (CA), ajoutez la ligne suivante au `cluster` section du fichier kubeconfig. Cela permet à Astra Control d'ajouter le cluster :

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

- **Rancher uniquement**: Lorsque vous gérez des clusters d'applications dans un environnement Rancher, modifiez le contexte par défaut du cluster d'applications dans le fichier kubeconfig fourni par Rancher pour utiliser un contexte de plan de contrôle au lieu du contexte du serveur d'API Rancher. La charge est réduite sur le serveur API Rancher et les performances sont améliorées.
- **Exigences du mécanisme de provisionnement Astra Control** : vous devez avoir un mécanisme de provisionnement Astra Control correctement configuré, y compris ses composants Astra Trident, pour gérer les clusters.
  - **Revoir les exigences de l'environnement Astra Trident** : avant d'installer ou de mettre à niveau Astra Control Provisioner, consultez le "systèmes front-end, systèmes back-end et configurations hôte pris en charge".
  - **Activer la fonctionnalité Astra Control Provisioner** : il est fortement recommandé d'installer Astra Trident 23.10 ou version ultérieure et de l'activer "Fonctionnalité de stockage avancée Astra Control Provisioner". Dans les prochaines versions, Astra Control ne prendra pas en charge Astra Trident si le mécanisme de provisionnement Astra Control n'est pas également activé.
  - **Configurer un back-end de stockage** : au moins un back-end de stockage doit l'être "Configuré dans Astra Trident" sur le cluster.
  - **Configurer une classe de stockage** : au moins une classe de stockage doit être "Configuré dans Astra Trident" sur le cluster. Si une classe de stockage par défaut est configurée, assurez-vous qu'il s'agit de la classe de stockage **Only** qui possède l'annotation par défaut.

- **Configurer un contrôleur de snapshot de volume et installer une classe de snapshot de volume** : "Installez un contrôleur de snapshot de volume" Il est ainsi possible de créer des snapshots dans Astra Control. "Création" au moins un `VolumeSnapshotClass` Avec Astra Trident.

Étapes

1. Dans le Tableau de bord, sélectionnez **Manage Kubernetes cluster**.

   Suivez les invites pour ajouter le cluster.

2. **Fournisseur** : sélectionnez l'onglet **autre** pour ajouter des détails sur votre cluster auto-géré.
3. **Autre**: Fournir des détails sur votre cluster auto-géré en téléchargeant un `kubeconfig.yaml` ou en collant le contenu du `kubeconfig.yaml` fichier à partir du presse-papiers.

   > Si vous créez la vôtre `kubeconfig` fichier, vous ne devez définir que **un** élément de contexte dans celui-ci. Reportez-vous à la section "ces instructions" pour plus d'informations sur la création `kubeconfig` fichiers.

4. **Nom d'identification** : indiquez un nom pour les informations d'identification de cluster autogérées que vous téléchargez sur Astra Control. Par défaut, le nom des identifiants est automatiquement renseigné comme nom du cluster.
5. **Identificateur de route privée** : saisissez l'identificateur de route privée que vous pouvez obtenir à partir du connecteur Astra. Si vous interrogez le connecteur Astra via le `kubectl get astraconnector -n astra-connector` l'identificateur de route privée est appelé `ASTRACONNECTORID`.

   > L'identifiant de la route privée est le nom associé à Astra Connector qui permet de gérer un cluster Kubernetes privé par Astra. Dans ce contexte, un cluster privé est un cluster Kubernetes qui n'expose pas son serveur d'API à Internet.

6. Sélectionnez **Suivant**.
7. (Facultatif) **Storage** : si vous le souhaitez, sélectionnez la classe de stockage que les applications Kubernetes déployées sur ce cluster doivent utiliser par défaut.
   a. Pour sélectionner une nouvelle classe de stockage par défaut pour le cluster, cochez la case **affecter une nouvelle classe de stockage par défaut**.
   b. Sélectionnez une nouvelle classe de stockage par défaut dans la liste.

Chaque fournisseur de service de stockage cloud affiche les informations suivantes en matière de prix, de performance et de résilience :

- Cloud Volumes Service pour Google Cloud : informations sur le prix, la performance et la résilience
- Google persistent Disk : pas d'informations sur le prix, la performance ou la résilience disponibles
- Azure NetApp Files : informations sur les performances et la résilience
- Azure Managed Disks : aucun prix, performances ou résilience disponibles
- Amazon Elastic Block Store : pas d'informations disponibles sur le prix, la performance ou la résilience
- Amazon FSX pour NetApp ONTAP : aucune information disponible concernant le prix, les performances ou la résilience
- NetApp Cloud Volumes ONTAP : aucune information disponible sur le prix, les performances ou la résilience

Chaque classe de stockage peut utiliser l'un des services suivants :

- "Cloud Volumes Service pour Google Cloud"
- "Disque persistant Google"
- "Azure NetApp Files"
- "Disques gérés Azure"
- "Amazon Elastic Block Store"
- "Amazon FSX pour NetApp ONTAP"
- "NetApp Cloud Volumes ONTAP"

  En savoir plus sur "Classes de stockage pour les clusters Amazon Web Services". En savoir plus sur "Classes de stockage pour les clusters AKS". En savoir plus sur "Classes de stockage pour clusters GKE".

c. Sélectionnez **Suivant**.

d. **Revoir et approuver** : consultez les détails de la configuration.

e. Sélectionnez **Ajouter** pour ajouter le cluster à Astra Control Service.

Modifiez la classe de stockage par défaut

Vous pouvez modifier la classe de stockage par défaut d'un cluster.

Modifiez la classe de stockage par défaut avec Astra Control

Vous pouvez modifier la classe de stockage par défaut d'un cluster depuis Astra Control. Si votre cluster utilise un service back-end de stockage installé précédemment, il se peut que vous ne puissiez pas utiliser cette méthode pour modifier la classe de stockage par défaut (l'action **Set as default** n'est pas sélectionnable). Dans ce cas, vous pouvez Modifiez la classe de stockage par défaut à l'aide de la ligne de commande.

Étapes

1. Dans l'interface utilisateur du service de contrôle Astra, sélectionnez **clusters**.

2. Sur la page **clusters**, sélectionnez le cluster que vous souhaitez modifier.
3. Sélectionnez l'onglet **stockage**.
4. Sélectionnez la catégorie **classes de stockage**.
5. Sélectionnez le menu **actions** pour la classe de stockage que vous souhaitez définir par défaut.
6. Sélectionnez **définir comme valeur par défaut**.

Modifiez la classe de stockage par défaut à l'aide de la ligne de commande

Vous pouvez modifier la classe de stockage par défaut d'un cluster à l'aide des commandes Kubernetes. Cette méthode fonctionne quelle que soit la configuration du cluster.

Étapes

1. Connectez-vous à votre cluster Kubernetes.
2. Lister les classes de stockage de votre cluster :

```
kubectl get storageclass
```

3. Supprimez la désignation par défaut de la classe de stockage par défaut. Remplacez <SC_NAME> par le nom de la classe de stockage :

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

4. Sélectionnez par défaut une classe de stockage différente. Remplacez <SC_NAME> par le nom de la classe de stockage :

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

5. Confirmez la nouvelle classe de stockage par défaut :

```
kubectl get storageclass
```

Vérifiez la version d'Astra Trident

Pour ajouter un cluster autogéré qui utilise Astra Control Provisioner ou Astra Trident pour les services de stockage, assurez-vous que la version installée d'Astra Trident est la version 23.10 ou la plus récente.

Étapes

1. Déterminez la version d'Astra Trident que vous exécutez :

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

Si Astra Trident est installé, le résultat est similaire à ce qui suit :

```
NAME      VERSION
trident   24.02.0
```

Si Astra Trident n'est pas installé, le résultat est similaire à ce qui suit :

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

2. Effectuez l'une des opérations suivantes :
   - Si vous exécutez Astra Trident 23.01 ou une version antérieure, utilisez-les "instructions" Pour effectuer une mise à niveau vers une version plus récente d'Astra Trident avant de passer à Astra Control Provisioner. C'est possible "effectuer une mise à niveau directe" Vers Astra Control Provisioner 24.02 si votre Astra Trident se trouve dans une fenêtre à quatre versions de la version 24.02. Par exemple, vous pouvez effectuer une mise à niveau directe d'Astra Trident 23.04 vers Astra Control Provisioner 24.02.
   - Si vous exécutez Astra Trident 23.10 ou version ultérieure, vérifiez que le mécanisme de provisionnement Astra Control a été utilisé "activé". ASTRA Control Provisioner ne fonctionnera pas avec les versions d'Astra Control Center antérieures à 23.10. "Mettez à niveau votre mécanisme de provisionnement Astra Control" De sorte qu'il dispose de la même version que l'Astra Control Center que vous mettez à niveau pour accéder aux dernières fonctionnalités.
3. Assurez-vous que les pods fonctionnent :

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

4. Vérifiez si les classes de stockage utilisent les pilotes Trident Astra pris en charge. Le nom de provisionnement doit être `csi.trident.netapp.io`. Reportez-vous à l'exemple suivant :

```
kubectl get sc
```

Exemple de réponse :

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

Créez un fichier kubeconfig

Vous pouvez ajouter un cluster à Astra Control Service à l'aide d'un fichier kubeconfig. Selon le type de cluster à ajouter, vous devrez peut-être créer manuellement un fichier kubeconfig pour votre cluster en suivant des étapes spécifiques.

- Créez un fichier kubeconfig pour les clusters Amazon EKS
- Créez un fichier kubeconfig pour les clusters Red Hat OpenShift Service sur AWS (ROSA)
- Créez un fichier kubeconfig pour d'autres types de clusters

Créez un fichier kubeconfig pour les clusters Amazon EKS

Suivez ces instructions pour créer un fichier kubeconfig et un code secret de jeton permanent pour les clusters Amazon EKS. Un code secret de jeton permanent est requis pour les clusters hébergés dans EKS.

Étapes

1. Suivez les instructions de la documentation Amazon pour générer un fichier kubeconfig :

   "Création ou mise à jour d'un fichier kubeconfig pour un cluster Amazon EKS"

2. Créer un compte de service comme suit :

   a. Créez un fichier de compte de service appelé `astracontrol-service-account.yaml`.

      Ajustez le nom du compte de service si nécessaire. L'espace de noms `kube-system` est nécessaire pour ces étapes. Si vous modifiez le nom du compte de service ici, vous devez appliquer les mêmes modifications dans les étapes suivantes.

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

+

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

3. Appliquer le compte de service :

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

4. Créer un `ClusterRoleBinding` fichier appelé `astracontrol-clusterrolebinding.yaml`.

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

```yaml
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. Appliquer la liaison de rôle de cluster :

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

6. Créez un fichier secret de token de compte de service appelé `astracontrol-secret.yaml`.

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

```yaml
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. Appliquer le secret de jeton :

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

8. Récupérer le secret de jeton :

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

9. Remplacer l' `user` Section du fichier kubeconfig AWS EKS avec le jeton, comme illustré ci-dessous :

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

Créez un fichier kubeconfig pour les clusters Red Hat OpenShift Service sur AWS (ROSA)

Suivez ces instructions pour créer un fichier kubeconfig pour les clusters Red Hat OpenShift Service sur AWS (ROSA).

Étapes

1. Connectez-vous au cluster ROSA.
2. Créer un compte de service :

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

3. Ajouter un rôle de cluster :

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

4. À l'aide de l'exemple suivant, créez un fichier de configuration secret de compte de service :

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

```yaml
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. Créez le secret :

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

6. Modifiez le compte de service que vous avez créé et ajoutez le nom secret du compte de service Astra Control au `secrets` section :

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

```yaml
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. Indiquez les secrets du compte de service, en les remplaçant `<CONTEXT>` avec le contexte approprié pour votre installation :

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

La fin de la sortie doit ressembler à ce qui suit :

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

Les indices pour chaque élément dans `secrets` la matrice commence par 0. Dans l'exemple ci-dessus, l'index de `astracontrol-service-account-dockercfg-dvfcd` serait 0 et l'index pour `secret-astracontrol-service-account` serait 1. Dans votre sortie, notez le numéro d'index du compte de service secret. Vous aurez besoin de ce numéro d'index à l'étape suivante.

8. Générez le kubeconfig comme suit :

   a. Créer un `create-kubeconfig.sh` fichier. Remplacement `TOKEN_INDEX` au début du script suivant avec la valeur correcte.

```
<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. Source des commandes à appliquer à votre cluster Kubernetes.

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

9. (Facultatif) Renommer le kubeconfig pour nommer votre cluster.

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

Créez un fichier kubeconfig pour d'autres types de clusters

Suivez ces instructions pour créer un fichier kubeconfig de rôle limité ou étendu pour les clusters Rancher, Kubernetes en amont et Red Hat OpenShift.

Pour les clusters gérés à l'aide de kubeconfig, vous pouvez éventuellement créer une autorisation limitée ou un rôle d'administrateur d'autorisations étendues pour Astra Control Service.

Cette procédure vous aide à créer un kubeconfig distinct si l'un des scénarios suivants s'applique à votre environnement :

- Vous souhaitez limiter les autorisations Astra Control sur les clusters qu'il gère
- Vous utilisez plusieurs contextes et ne pouvez pas utiliser le kubeconfig Astra Control par défaut configuré lors de l'installation, sinon un rôle limité avec un seul contexte ne fonctionnera pas dans votre environnement

Avant de commencer

Assurez-vous que vous disposez des éléments suivants pour le cluster que vous souhaitez gérer avant d'effectuer la procédure suivante :

- A "version prise en charge" de kubectl est installé.
- Kubectl accès au cluster que vous envisagez d'ajouter et de gérer avec Astra Control Service

> Pour cette procédure, vous n'avez pas besoin d'un accès kubectl au cluster exécutant Astra Control Service.

- Un kubeconfig actif pour le cluster que vous avez l'intention de gérer avec des droits d'administrateur de cluster pour le contexte actif

Étapes

1. Créer un compte de service :
   a. Créez un fichier de compte de service appelé `astracontrol-service-account.yaml`.

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

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

   b. Appliquer le compte de service :

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

2. Créez l'un des rôles de cluster suivants avec des autorisations suffisantes pour qu'un cluster soit géré par

Astra Control :

Rôle limité du cluster

Ce rôle contient les autorisations minimales nécessaires à la gestion d'un cluster par Astra Control :

a. Créer un `ClusterRole` fichier appelé, par exemple, `astra-admin-account.yaml`.

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

```yaml
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. (Pour les clusters OpenShift uniquement) Ajouter les éléments suivants à la fin du `astra-admin-account.yaml` fichier :

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

c. Appliquer le rôle de cluster :

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

Rôle de cluster étendu

Ce rôle contient des autorisations étendues pour qu'un cluster soit géré par Astra Control. Vous pouvez utiliser ce rôle si vous utilisez plusieurs contextes et que vous ne pouvez pas utiliser le kubeconfig Astra Control par défaut configuré lors de l'installation, ou si un rôle limité avec un seul contexte ne fonctionnera pas dans votre environnement :

> Les éléments suivants `ClusterRole` Les étapes constituent un exemple Kubernetes général. Pour des instructions spécifiques à votre environnement, reportez-vous à la documentation de votre distribution Kubernetes.

a. Créer un `ClusterRole` fichier appelé, par exemple, `astra-admin-account.yaml`.

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

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

b. Appliquer le rôle de cluster :

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

3. Créer la liaison de rôle cluster pour le rôle cluster vers le compte de service :

a. Créer un `ClusterRoleBinding` fichier appelé `astracontrol-clusterrolebinding.yaml`.

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

```yaml
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. Appliquer la liaison de rôle de cluster :

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

4. Créez et appliquez le secret de jeton :
   a. Créez un fichier secret de jeton appelé `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. Appliquer le secret de jeton :

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

5. Ajoutez le secret de jeton au compte de service en ajoutant son nom au `secrets` tableau (dernière ligne de l'exemple suivant) :

```
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. Indiquez les secrets du compte de service, en les remplaçant `<context>` avec le contexte approprié pour votre installation :

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

La fin de la sortie doit ressembler à ce qui suit :

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

Les indices pour chaque élément dans `secrets` la matrice commence par 0. Dans l'exemple ci-dessus, l'index de `astracontrol-service-account-dockercfg-48xhx` serait 0 et l'index pour `secret-astracontrol-service-account` serait 1. Dans votre sortie, notez le numéro d'index du compte de service secret. Vous aurez besoin de ce numéro d'index à l'étape suivante.

7. Générez le kubeconfig comme suit :
   a. Créer un `create-kubeconfig.sh` fichier.
   b. Remplacement `TOKEN_INDEX` au début du script suivant avec la valeur correcte.

```
<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. Source des commandes à appliquer à votre cluster Kubernetes.

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

8. (Facultatif) Renommer le kubeconfig pour nommer votre cluster.

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```