添加自管理集群

Astra Control Service

NetApp
April 24, 2024

This PDF was generated from https://docs.netapp.com/zh-cn/astra-control-service/get-started/add-public-self-managed-cluster.html on April 24, 2024. Always check docs.netapp.com for the latest.

目录

添加自管理集群

将公共自管理集群添加到Astra Control Service

设置环境后，您可以创建 Kubernetes 集群，然后将其添加到 Astra Control Service 。

自管理集群是指直接配置和管理的集群。Astra Control Service支持在公共云环境中运行的自管理集群。您可以通过上传将自管理集群添加到Astra Control Service `kubeconfig.yaml` 文件您需要确保集群满足此处所述的要求。

支持的Kubnetes分发版

您可以使用Astra Control Service管理以下类型的公共自管理集群：

Kubbernetes分发	支持的版本
Kubnetes (上游)	1.27至1.29
Rancher Kubernetes Engine （ RKE）	RKE 1：版本1.24.17、1.25.13、1.26.8、带RANcher Manager 2.7.9 RKE 2：版本1.23.16和1.24.13、带Randcher Manager 2.6.13 RKE 2：版本1.24.17、1.25.14、1.26.9、带RANcher Manager 2.7.9
Red Hat OpenShift 容器平台	4.12至4.14

以下说明假定您已创建一个自行管理的集群。

- 将集群添加到Asta Control Service
- [更改默认存储类]

将集群添加到Asta Control Service

登录到 Astra Control Service 后，第一步是开始管理集群。在将集群添加到Astra Control Service之前、您需要执行特定任务并确保集群满足特定要求。

开始之前

自管理集群是指直接配置和管理的集群。Astra Control Service支持在公共云环境中运行的自管理集群。您的自行管理集群可以使用Astra控件配置程序与NetApp存储服务连接、也可以使用容器存储接口(CSI)驱动程序与Amazon Elastic Block Store (EBS)、Azure托管磁盘和Google持久磁盘连接。

Astra控制服务支持使用以下Kubernetes分发版的自管理集群：

- Red Hat OpenShift 容器平台
- Rancher Kubernetes引擎
- 上游Kubernetes

您的自管理集群需要满足以下要求：

- 集群必须可通过Internet访问。
- 如果您正在使用或计划使用已启用CSI驱动程序的存储、则必须在集群上安装相应的CSI驱动程序。有关使用CSI驱动程序集成存储的详细信息、请参阅存储服务的文档。
- 您可以访问仅包含一个上下文元素的集群kubeconfigfile文件。请遵循 "这些说明" 生成kubeconfig.文件。
- 如果要使用引用私有证书颁发机构(CA)的kubeconfigfile文件添加集群、请将以下行添加到 `cluster` kubeconfig"文件的部分。这样、Astra Control便可添加集群：

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

- *仅Rancher *：在Rancher环境中管理应用程序集群时、请修改Rancher提供的kubeconfig文件中的应用程序集群默认上下文、以使用控制平面上下文、而不是Rancher API服务器上下文。这样可以减少Rancher API 服务器上的负载并提高性能。
- **Astra Control**配置程序要求：要管理集群、您应正确配置Astra Control配置程序(包括其Astra三项功能组件)。
  - 查看**Astra**三端环境要求：在安装或升级Astra Control配置程序之前、请查看 "支持的前端、后端和主机配置"。
  - 启用**Astra Control**配置程序功能：强烈建议您安装Astra Trident 23.10或更高版本并启用 "Asta Control配置程序高级存储功能"。在未来版本中、如果Asta Control配置程序未启用、则Asta Control将不支持Asta Trent。
  - 配置存储后端：必须至少有一个存储后端 "已在Asta Trdent中配置" 在集群上。
  - 配置存储类：必须至少有一个存储类 "已在Asta Trdent中配置" 在集群上。如果配置了默认存储类，请确保该存储类是具有默认标注的*Only"存储类。
  - 配置卷快照控制器并安装卷快照类： "安装卷快照控制器" 以便可以在Astra Control中创建快照。 "创建" 至少一个 `VolumeSnapshotClass` 使用Asta三端到功能。

步骤

1. 在信息板上，选择 * 管理 Kubernetes 集群 * 。

   按照提示添加集群。

2. 提供程序：选择*其他*选项卡以添加有关自行管理的集群的详细信息。
   a. 其他：通过上传提供有关自管理集群的详细信息 `kubeconfig.yaml` 文件或粘贴的内容 `kubeconfig.yaml` 文件。

      创建自己的 `kubeconfig` file中、您只能定义*一*上下文元素。请参见 "Kubernetes 文档" 有关创建的信息 `kubeconfig` 文件。

3. 凭据名称：提供要上传到Astra Control的自管理集群凭据的名称。默认情况下，凭据名称会自动填充为集群的名称。
4. 专用路由标识符：此字段仅适用于专用集群。
5. 选择 * 下一步 * 。
6. (可选)存储：(可选)选择默认情况下希望部署到此集群中的Kubernetes应用程序使用的存储类。
   a. 要为集群选择新的默认存储类，请启用*Assign a new default storage class*复选框。
   b. 从列表中选择新的默认存储类。

      每个云提供商存储服务都会显示以下价格、性能和弹性信息：

      - Cloud Volumes Service for Google Cloud：价格、性能和弹性信息
      - Google Persistent Disk：没有价格、性能或弹性信息
      - Azure NetApp Files ：性能和弹性信息
      - Azure受管磁盘：无可用的价格、性能或弹性信息
      - Amazon Elastic Block Store：没有价格、性能或弹性信息
      - 适用于NetApp ONTAP 的Amazon FSX：没有价格、性能或弹性信息
      - NetApp Cloud Volumes ONTAP ：没有价格、性能或弹性信息

      每个存储类均可使用以下服务之一：

      - "适用于 Google Cloud 的 Cloud Volumes Service"
      - "Google 持久磁盘"
        - "Azure NetApp Files"
        - "Azure 受管磁盘"
        - "Amazon Elastic Block Store"
        - "适用于 NetApp ONTAP 的 Amazon FSX"
        - "NetApp Cloud Volumes ONTAP"

          了解更多信息 "Amazon Web Services集群的存储类"。了解更多信息 "AKS 集群的存储类"。了解更多信息 "GKE 集群的存储类"。

   c. 选择 * 下一步 * 。
   d. 审核和批准：审核配置详细信息。
   e. 选择*Add*将集群添加到Astra Control Service。

更改默认存储类

您可以更改集群的默认存储类。

使用**Astra Control**更改默认存储类

您可以在Astra Control中更改集群的默认存储类。如果集群使用先前安装的存储后端服务、则可能无法使用此方法更改默认存储类(不能选择*设置为默认值*操作)。在这种情况下、您可以 [使用命令行更改默认存储类]。

步骤

1. 在 Astra 控制服务 UI 中，选择 * 集群 * 。
2. 在*集群*页面上、选择要更改的集群。
3. 选择 * 存储 * 选项卡。
4. 选择*存储类*类别。
5. 选择要设置为默认值的存储类的*操作*菜单。
6. 选择*设置为默认值*。

使用命令行更改默认存储类

您可以使用Kubernetes命令更改集群的默认存储类。无论集群的配置如何、此方法都有效。

步骤

1. 登录到Kubernetes集群。
2. 列出集群中的存储类：

```
kubectl get storageclass
```

3. 从默认存储类中删除默认指定。将<SC_NAME> 替换为存储类的名称：

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

4. 将其他存储类标记为默认值。将<SC_NAME> 替换为存储类的名称：

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

5. 确认新的默认存储类：

```
kubectl get storageclass
```

将私有自管理集群添加到Asta Control Service

设置环境后，您可以创建 Kubernetes 集群，然后将其添加到 Astra Control Service 。

自管理集群是指直接配置和管理的集群。Astra Control Service支持在公共云环境中运行的自管理集群。您可以通过上传将自管理集群添加到Astra Control Service `kubeconfig.yaml` 文件您需要确保集群满足此处所述的要求。

支持的Kubnetes分发版

您可以使用Astra Control Service管理以下类型的专用自管理集群：

Kubbernetes分发	支持的版本
Kubnetes (上游)	1.27至1.29
Rancher Kubernetes Engine （ RKE）	RKE 1：版本1.24.17、1.25.13、1.26.8、带RANcher Manager 2.7.9 RKE 2：版本1.23.16和1.24.13、带Randcher Manager 2.6.13 RKE 2：版本1.24.17、1.25.14、1.26.9、带RANcher Manager 2.7.9
Red Hat OpenShift 容器平台	4.12至4.14

以下说明假定您已创建私有集群并准备好了远程访问该集群的安全方法。

要将专用集群添加到Astra Control Service、您需要执行以下任务：

1. 安装A作用 连接器
2. [设置永久性存储]
3. 将专用自管理集群添加到Asta Control Service

安装A作用 连接器

在添加专用集群之前、您需要在此集群上安装Astra Connector、以便Astra Control可以与其通信。请参见 "为使用非Kubnetes本机工作流管理的专用集群安装以前版本的Astra Connector" 有关说明，请参见。

设置永久性存储

为集群配置永久性存储。有关配置永久性存储的详细信息、请参见入门文档：

- "使用 Azure NetApp Files 设置 Microsoft Azure"
- "使用 Azure 受管磁盘设置 Microsoft Azure"
- "设置Amazon Web Services"
- "设置 Google Cloud"

将专用自管理集群添加到Asta Control Service

现在、您可以将专用集群添加到Astra Control Service。

开始之前

自管理集群是指直接配置和管理的集群。Astra Control Service支持在公共云环境中运行的自管理集群。您的自行管理集群可以使用Astra控件配置程序与NetApp存储服务连接、也可以使用容器存储接口(CSI)驱动程序与Amazon Elastic Block Store (EBS)、Azure托管磁盘和Google持久磁盘连接。

Astra控制服务支持使用以下Kubernetes分发版的自管理集群：

- Red Hat OpenShift 容器平台
- Rancher Kubernetes引擎
- 上游Kubernetes

您的自管理集群需要满足以下要求：

- 集群必须可通过Internet访问。
- 如果您正在使用或计划使用已启用CSI驱动程序的存储、则必须在集群上安装相应的CSI驱动程序。有关使用CSI驱动程序集成存储的详细信息、请参阅存储服务的文档。
- 您可以访问仅包含一个上下文元素的集群kubeconfigfile文件。请遵循 "这些说明" 生成kubeconfig.文件。
- 如果要使用引用私有证书颁发机构(CA)的kubeconfigfile文件添加集群、请将以下行添加到 `cluster` kubeconfig"文件的部分。这样、Astra Control便可添加集群：

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

- *仅Rancher *：在Rancher环境中管理应用程序集群时、请修改Rancher提供的kubeconfig文件中的应用程序集群默认上下文、以使用控制平面上下文、而不是Rancher API服务器上下文。这样可以减少Rancher API 服务器上的负载并提高性能。
- **Astra Control**配置程序要求：要管理集群、您应正确配置Astra Control配置程序(包括其Astra三项功能组件)。
  - 查看**Astra**三端环境要求：在安装或升级Astra Control配置程序之前、请查看 "支持的前端、后端和主机配置"。
  - 启用**Astra Control**配置程序功能：强烈建议您安装Astra Trident 23.10或更高版本并启用 "Asta Control配置程序高级存储功能"。在未来版本中、如果Asta Control配置程序未启用、则Asta Control将不支持Asta Trent。
  - 配置存储后端：必须至少有一个存储后端 "已在Asta Trdent中配置" 在集群上。
  - 配置存储类：必须至少有一个存储类 "已在Asta Trdent中配置" 在集群上。如果配置了默认存储类，请确保该存储类是具有默认标注的*Only"存储类。
  - 配置卷快照控制器并安装卷快照类： "安装卷快照控制器" 以便可以在Astra Control中创建快照。 "创建" 至少一个 `VolumeSnapshotClass` 使用Asta三端到功能。

步骤

1. 在信息板上，选择 * 管理 Kubernetes 集群 * 。

   按照提示添加集群。

2. 提供程序：选择*其他*选项卡以添加有关自行管理的集群的详细信息。
3. 其他：通过上传提供有关自管理集群的详细信息 `kubeconfig.yaml` 文件或粘贴的内容 `kubeconfig.yaml` 文件。

   > 创建自己的 `kubeconfig` file中、您只能定义*一*上下文元素。请参见 "这些说明" 有关创建的信息 `kubeconfig` 文件。

4. 凭据名称：提供要上传到Astra Control的自管理集群凭据的名称。默认情况下，凭据名称会自动填充为集群的名称。
5. 专用路由标识符：输入专用路由标识符，您可以从Astra Connector获取该标识符。如果您通过查询Astra Connector `kubectl get astraconnector -n astra-connector` 命令中、专用路由标识符称为 `ASTRACONNECTORID`。

   > 专用路由标识符是与Asta Connector关联的名称、Asta可通过Asta管理专用Kubnetes集群。在这种情况下、专用集群是指不向Internet公开其API服务器的Kubbernetes集群。

6. 选择 * 下一步 * 。
7. (可选)存储：(可选)选择默认情况下希望部署到此集群中的Kubernetes应用程序使用的存储类。
   a. 要为集群选择新的默认存储类，请启用*Assign a new default storage class*复选框。
   b. 从列表中选择新的默认存储类。

   > 每个云提供商存储服务都会显示以下价格、性能和弹性信息：
   >
   > - Cloud Volumes Service for Google Cloud：价格、性能和弹性信息
   > - Google Persistent Disk：没有价格、性能或弹性信息
   > - Azure NetApp Files ：性能和弹性信息
   > - Azure受管磁盘：无可用的价格、性能或弹性信息
   > - Amazon Elastic Block Store：没有价格、性能或弹性信息
   > - 适用于NetApp ONTAP 的Amazon FSX：没有价格、性能或弹性信息
   > - NetApp Cloud Volumes ONTAP ：没有价格、性能或弹性信息

   每个存储类均可使用以下服务之一：

   - "适用于 Google Cloud 的 Cloud Volumes Service"
   - "Google 持久磁盘"
   - "Azure NetApp Files"
   - "Azure 受管磁盘"

- "Amazon Elastic Block Store"
- "适用于 NetApp ONTAP 的 Amazon FSX"
- "NetApp Cloud Volumes ONTAP"

  了解更多信息 "Amazon Web Services集群的存储类"。了解更多信息 "AKS 集群的存储类"。了解更多信息 "GKE 集群的存储类"。

c. 选择 * 下一步 * 。

d. 审核和批准：审核配置详细信息。

e. 选择*Add*将集群添加到Astra Control Service。

更改默认存储类

您可以更改集群的默认存储类。

使用**Astra Control**更改默认存储类

您可以在Astra Control中更改集群的默认存储类。如果集群使用先前安装的存储后端服务、则可能无法使用此方法更改默认存储类(不能选择*设置为默认值*操作)。在这种情况下、您可以 [使用命令行更改默认存储类]。

步骤

1. 在 Astra 控制服务 UI 中，选择 * 集群 * 。
2. 在*集群*页面上、选择要更改的集群。
3. 选择 * 存储 * 选项卡。
4. 选择*存储类*类别。
5. 选择要设置为默认值的存储类的*操作*菜单。
6. 选择*设置为默认值*。

使用命令行更改默认存储类

您可以使用Kubernetes命令更改集群的默认存储类。无论集群的配置如何、此方法都有效。

步骤

1. 登录到Kubernetes集群。
2. 列出集群中的存储类：

```
kubectl get storageclass
```

3. 从默认存储类中删除默认指定。将<SC_NAME> 替换为存储类的名称：

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

4. 将其他存储类标记为默认值。将<SC_NAME> 替换为存储类的名称：

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

5. 确认新的默认存储类：

```
kubectl get storageclass
```

检查Asta三端安装版本

要添加使用Asta Control置备程序或Asta三端对存储服务使用的自管理集群、请确保已安装的Asta三端对等版本为23.10或最新版本。

步骤

1. 确定您正在运行的Astra三项目标版本：

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

如果安装了Astra Trident、则会显示类似于以下内容的输出：

```
NAME       VERSION
trident    24.02.0
```

如果未安装Astra Trident、您将看到类似于以下内容的输出：

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

2. 执行以下操作之一：
   - 如果您运行的是Asta三端凹凸版23.01或更早版本、请使用这些版本 "说明" 在升级到Asta Control配置程序之前、升级到Asta三端到最新版本。您可以 "执行直接升级" 如果您的Astra三端存储在版本24.02的四个版本的窗口中、则将Astra Control配置程序更新为24.02。例如、您可以直接从Asta三端23.04升级到Asta Control配置程序24.02。
   - 如果您运行的是Astra Trdent 23.10或更高版本、请验证Astra Control配置程序是否已启用 "enabled" 。Asta Control配置程序不能用于23.10之前的Asta Control Center版本。 "升级Astra Control配置程序" 以便它与您要升级的Astra Control Center版本相同、以访问最新功能。
3. 确保Pod正在运行：

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

4. 检查存储类是否正在使用受支持的Astra Trident驱动程序。配置程序名称应为 `csi.trident.netapp.io`。请参见以下示例：

```
kubectl get sc
```

响应示例：

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

创建kubeconfig文件

您可以使用kubeconfig"文件将集群添加到Astra Control Service。根据要添加的集群类型、您可能需要使用特定步骤为集群手动创建kubeconfigfile文件。

- 为Amazon EKS集群创建kubeconfig.文件
- 为AWS (ROSA)集群上的Red Hat OpenShift Service创建一个kubeconfigfile文件
- [为其他类型的集群创建kubeconfig.文件]

为Amazon EKS集群创建kubeconfig.文件

按照以下说明为Amazon EKS集群创建kubeconfigfile文件和永久令牌密钥。EKS中托管的集群需要永久令牌密钥。

步骤

1. 按照亚马逊文档中的说明生成kubeconfig:

   "为Amazon EKS集群创建或更新kubeconconfig文件"

2. 按如下所示创建服务帐户：

   a. 创建名为的服务帐户文件 `astracontrol-service-account.yaml`。

      根据需要调整服务帐户名称。命名空间 `kube-system` 这些步骤需要。如果您在此处更改了服务帐户名称、则应在以下步骤中应用相同的更改。

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

+

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

3. 应用服务帐户：

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

4. 创建 `ClusterRoleBinding` 文件已调用 `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. 应用集群角色绑定：

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

6. 创建名为的服务帐户令牌机密文件 `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. 应用令牌密钥：

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

8. 检索令牌密钥：

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

9. 更换 `user` 部分的AWS EKS kubeconfigconfig文件以及令牌、如以下示例所示：

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

为AWS (ROSA)集群上的Red Hat OpenShift Service创建一个kubeconfigfile文件

按照以下说明为Red Hat OpenShift Service on AWS (ROSA)集群创建kubeconfigTM文件。

步骤

1. 登录到ROSA集群。
2. 创建服务帐户：

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

3. 添加集群角色：

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

4. 使用以下示例、创建一个服务帐户机密配置文件：

```
<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. 创建密钥：

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

6. 编辑您创建的服务帐户、并将Astra Control服务帐户机密名称添加到中 `secrets` 部分。

```
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. 列出服务帐户密码、替换 `<CONTEXT>` 使用适用于您的安装的正确环境：

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

输出的结尾应类似于以下内容：

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

中每个元素的索引 `secrets` 阵列以0开头。在上面的示例中、是的索引 `astracontrol-service-account-dockercfg-dvfcd` 将为0、并为创建索引 `secret-astracontrol-service-account` 将为1。在输出中、记下服务帐户密钥的索引编号。在下一步中、您将需要此索引编号。

8. 按如下所示生成 kubeconfig：
   a. 创建 `create-kubeconfig.sh` 文件替换 `TOKEN_INDEX` 在以下脚本的开头、使用正确的值。

```
<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. 获取用于将其应用于 Kubernetes 集群的命令。

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

9. (可选)将kubeconfig重命名为集群的有意义名称。

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

为其他类型的集群创建kubeconfig.文件

按照以下说明为然彻集群、上游Kubernetes集群和Red Hat OpenShift集群创建有限或扩展的角色kubeconconfig文件。

对于使用kubeconfig"管理的集群、您可以选择为Astra Control Service创建有限权限或扩展权限管理员角色。

如果您适用场景的环境发生以下任一情况、则此操作步骤可帮助您创建一个单独的kubeconfig:

- 您希望限制Astra Control对其管理的集群的权限
- 您使用多个环境、并且不能使用在安装期间配置的默认Asta Control kubeconfig,否则在您的环境中使用单一环境的有限角色将不起作用

开始之前

在完成操作步骤 步骤之前、请确保您对要管理的集群具有以下信息：

- 答 "支持的版本" 已安装kubeck.
- 对要使用Astra Control Service添加和管理的集群的kubect访问权限

> 对于此操作步骤、您不需要对运行Astra控制服务的集群进行kubect访问。

- 要使用活动环境的集群管理员权限管理的集群的活动kubeconfig

步骤

1. 创建服务帐户：
   a. 创建名为的服务帐户文件 `astracontrol-service-account.yaml`。

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

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

   b. 应用服务帐户：

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

2. 创建以下具有足够权限的集群角色之一、以使集群由Astra Control管理：

集群角色受限

此角色包含由Asta Control管理集群所需的最低权限：

a. 创建 `ClusterRole` 文件、例如、 `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. (仅适用于OpenShift集群)在末尾附加以下内容 `astra-admin-account.yaml` 文件：

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

c. 应用集群角色：

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

已扩展集群角色

此角色包含要由Asta Control管理的集群的扩展权限。如果您使用多个环境，并且无法使用在安装期间配置的默认Asta Control kubeconfig,则可以使用此角色，否则在您的环境中，只使用一个环境的有限角色将不起作用：

> 以下内容 `ClusterRole` 步骤是一个常规Kubbernetes示例。有关特定于您的环境的说明、请参见Kubennetes分发版的文档。

a. 创建 `ClusterRole` 文件、例如、 `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. 应用集群角色：

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

3. 为集群角色创建与服务帐户的集群角色绑定：

a. 创建 `ClusterRoleBinding` 文件已调用 `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. 应用集群角色绑定：

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

4. 创建并应用令牌密钥：
   a. 创建名为的令牌机密文件 `secret-astracontrol-service-account.yaml`。

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

```yaml
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. 应用令牌密钥：

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

5. 通过将令牌密钥名称添加到、将其添加到服务帐户 `secrets` 数组(以下示例中的最后一行)：

```
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. 列出服务帐户密码、替换 `<context>` 使用适用于您的安装的正确环境：

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

输出的结尾应类似于以下内容：

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

中每个元素的索引 `secrets` 阵列以0开头。在上面的示例中、是的索引 `astracontrol-service-account-dockercfg-48xhx` 将为0、并为创建索引 `secret-astracontrol-service-account` 将为1。在输出中、记下服务帐户密钥的索引编号。在下一步中、您将需要此索引编号。

7. 按如下所示生成 kubeconfig：
   a. 创建 `create-kubeconfig.sh` 文件
   b. 替换 `TOKEN_INDEX` 在以下脚本的开头、使用正确的值。

```
<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. 获取用于将其应用于 Kubernetes 集群的命令。

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

8. (可选)将kubeconfig重命名为集群的有意义名称。

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

版权信息

版权所有 © 2024 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本文档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。 美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 http://www.netapp.com/TM 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。