



新增自我管理的叢集 Astra Control Service

NetApp
April 24, 2024

目錄

新增自我管理的叢集	1
將公用自我管理叢集新增至 Astra Control Service	1
將私有自我管理叢集新增至 Astra Control Service	5
檢查 Astra Trident 版本	10
建立Kbeconfig檔案	11

新增自我管理的叢集

將公用自我管理叢集新增至 **Astra Control Service**

設定環境之後、您就可以建立Kubernetes叢集、然後將其新增至Astra Control Service。

自我管理的叢集是您直接進行資源配置與管理的叢集。Astra Control Service支援在公有雲環境中執行的自我管理叢集。您可以上傳、將自我管理的叢集新增至Astra Control Service `kubeconfig.yaml` 檔案：您必須確保叢集符合此處列出的需求。

支援的 **Kubernetes** 發佈

您可以使用 Astra Control Service 來管理下列類型的公用、自我管理叢集：

Kubernetes 配送	支援的版本
Kubernetes （上游）	1.27 至 1.29
Rancher Kubernetes引擎（RKE）	RKE 1：採用 Rancher Manager 2.7.9 的版本 1.24.17、1.25.13、1.26.8 RKE 2：版本 1.23.16 和 1.24.13 搭配 Rancher Manager 2.6.13 RKE 2：採用 Rancher Manager 2.7.9 的版本 1.24.17、1.25.14、1.26.9
Red Hat OpenShift Container Platform	4.12 至 4.14

這些指示假設您已建立自我管理的叢集。

- [將叢集新增至 Astra Control Service](#)
- [\[變更預設儲存類別\]](#)

將叢集新增至 **Astra Control Service**

登入Astra Control Service之後、您的第一步就是開始管理叢集。將叢集新增至Astra Control Service之前、您必須執行特定工作、並確保叢集符合特定需求。

自我管理的叢集是您直接進行資源配置與管理的叢集。Astra Control Service支援在公有雲環境中執行的自我管理叢集。您的自我管理叢集可以使用 Astra Control Provisioner 與 NetApp 儲存服務進行介面、或是使用 Container Storage Interface (CSI) 驅動程式與 Amazon Elastic Block Store (EBS)、Azure Managed Disks 及 Google Persistent Disk 進行介面。

Astra Control Service支援使用下列Kubernetes發佈版本的自我管理叢集：

- Red Hat OpenShift Container Platform
- Rancher Kubernetes引擎
- 上游Kubernetes

您的自我管理叢集必須符合下列需求：

- 叢集必須可透過網際網路存取。
- 如果您使用或打算使用已啟用SCSI驅動程式的儲存設備、則叢集上必須安裝適當的SCSI驅動程式。如需使用SCSI驅動程式來整合儲存設備的詳細資訊、請參閱儲存服務的說明文件。
- 您可以存取僅包含一個內容元素的叢集 kubeconfig 檔案。追蹤 [這些指示](#) 以產生 kubeconfig 檔案。
- 如果您要使用參考私有憑證授權單位 (CA) 的 kubeconfig 檔案來新增叢集、請將下列行新增至 cluster kubeconfig 檔案的一節。這可讓 Astra Control 新增叢集：

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

- *** 僅限Rancher ***：在Rancher環境中管理應用程式叢集時、請在Rancher提供的Kubeconfig檔案中修改應用程式叢集的預設內容、以使用控制面內容而非Rancher API伺服器內容。如此可減少Rancher API伺服器的負載、並改善效能。
- *** Astra Control Provisioner 需求 ***：您應該擁有正確設定的 Astra Control 資源配置程式、包括其 Astra Trident 元件、來管理叢集。
 - *** 檢閱 Astra Trident 環境需求 ***：在安裝或升級 Astra Control Provisioner 之前、請檢閱 [支援的前端、後端及主機組態](#)。
 - *** 啟用 Astra Control Provisioner 功能 ***：強烈建議您安裝 Astra Trident 23.10 或更新版本並啟用 ["Astra Control Provisioner 進階儲存功能"](#)。在後續版本中、如果 Astra Control 資源配置程式未啟用、Astra Control 將不支援 Astra Trident。
 - *** 設定儲存後端 ***：至少必須有一個儲存後端 ["在 Astra Trident 中設定"](#) 在叢集上。
 - *** 設定儲存類別 ***：至少必須有一個儲存類別 ["在 Astra Trident 中設定"](#) 在叢集上。如果已設定預設儲存類別、請確定它是具有預設註釋的 *** 僅 * 儲存類別**。
 - *** 設定 Volume Snapshot 控制器並安裝 Volume Snapshot 類別 ***：["安裝 Volume Snapshot 控制器"](#) 以便在Astra Control中建立快照。 ["建立"](#) 至少一個 VolumeSnapshotClass 使用 Astra Trident。

步驟

1. 在儀表板上、選取***管理Kubernetes叢集***。

依照提示新增叢集。

2. * 提供者 * : 選取 * 其他 * 索引標籤以新增自我管理叢集的詳細資料。
 - a. 其他: 上傳以提供自我管理叢集的詳細資料 `kubeconfig.yaml` 檔案或貼上的內容 `kubeconfig.yaml` 檔案。



如果您自行建立 `kubeconfig` 檔案中、您應該只定義*一個*內容元素。請參閱 "[Kubernetes文件](#)" 以取得有關建立的資訊 `kubeconfig` 檔案：

3. * 認證名稱 * : 為您上傳至 Astra Control 的自我管理叢集認證提供名稱。根據預設、認證名稱會自動填入為叢集名稱。
4. * 私有路由識別碼 * : 此欄位僅適用於私有叢集。
5. 選擇*下一步*。
6. (可選) * Storage* : (可選) 選擇您希望 Kubernetes 應用程式部署到此叢集的儲存類別、以供預設使用。
 - a. 若要為叢集選取新的預設儲存類別、請啟用 * 指派新的預設儲存類別 * 核取方塊。
 - b. 從清單中選取新的預設儲存類別。



每個雲端供應商的儲存服務都會顯示下列價格、效能和恢復能力資訊：

- 適用於Google Cloud的解決方案：價格、效能和恢復能力資訊Cloud Volumes Service
- Google持續磁碟：沒有可用的價格、效能或恢復能力資訊
- 支援：效能與恢復能力資訊Azure NetApp Files
- Azure託管磁碟：不提供價格、效能或恢復能力資訊
- Amazon Elastic Block Store：沒有可用的價格、效能或恢復能力資訊
- Amazon FSX for NetApp ONTAP 不提供價格、效能或恢復能力資訊
- NetApp Cloud Volumes ONTAP 產品：不提供價格、效能或恢復能力資訊

每個儲存類別都可以使用下列其中一項服務：

- "[適用於 Google Cloud Cloud Volumes Service](#)"
- "[Google持續磁碟](#)"
 - "[Azure NetApp Files](#)"
 - "[Azure託管磁碟](#)"
 - "[Amazon彈性區塊存放區](#)"
 - "[Amazon FSX for NetApp ONTAP 產品](#)"
 - "[NetApp Cloud Volumes ONTAP](#)"

深入瞭解 "[Amazon Web Services叢集的儲存類別](#)"。深入瞭解 "[適用於高效能叢集的儲存類別](#)"。
深入瞭解 "[GKE叢集的儲存類別](#)"。

- c. 選擇*下一步*。

- d. * 審查與核准 * : 檢閱組態詳細資料。
- e. 選取 * 新增 * 將叢集新增至 Astra Control Service 。

變更預設儲存類別

您可以變更叢集的預設儲存類別。

使用Astra Control變更預設儲存類別

您可以從Astra Control中變更叢集的預設儲存類別。如果叢集使用先前安裝的儲存後端服務、您可能無法使用此方法來變更預設儲存類別 (*設為預設*動作無法選取)。在這種情況下、您可以 [\[使用命令列變更預設儲存類別\]](#)。

步驟

1. 在Astra Control Service UI中、選取* Clusters*。
2. 在「叢集」頁面上、選取您要變更的叢集。
3. 選擇* Storage* (儲存設備) 選項卡。
4. 選擇*儲存類別*類別。
5. 針對您要設為預設的儲存類別、選取「動作」功能表。
6. 選擇*設為預設*。

使用命令列變更預設儲存類別

您可以使用Kubernetes命令變更叢集的預設儲存類別。無論叢集的組態為何、此方法都能正常運作。

步驟

1. 登入Kubernetes叢集。
2. 列出叢集中的儲存類別：

```
kubectl get storageclass
```

3. 從預設儲存類別中移除預設指定。以<SC_NAME> 儲存類別的名稱取代支援：

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

4. 將不同的儲存類別標示為預設。以<SC_NAME> 儲存類別的名稱取代支援：

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

5. 確認新的預設儲存類別：

```
kubectl get storageclass
```

將私有自我管理叢集新增至 Astra Control Service

設定環境之後、您就可以建立Kubernetes叢集、然後將其新增至Astra Control Service。

自我管理的叢集是您直接進行資源配置與管理的叢集。Astra Control Service支援在公有雲環境中執行的自我管理叢集。您可以上傳、將自我管理的叢集新增至Astra Control Service kubeconfig.yaml 檔案：您必須確保叢集符合此處列出的需求。

支援的 Kubernetes 發佈

您可以使用 Astra Control Service 來管理下列類型的私有、自我管理叢集：

Kubernetes 配送	支援的版本
Kubernetes （上游）	1.27 至 1.29
Rancher Kubernetes引擎（RKE）	RKE 1：採用 Rancher Manager 2.7.9 的版本 1.24.17、1.25.13、1.26.8 RKE 2：版本 1.23.16 和 1.24.13 搭配 Rancher Manager 2.6.13 RKE 2：採用 Rancher Manager 2.7.9 的版本 1.24.17、1.25.14、1.26.9
Red Hat OpenShift Container Platform	4.12 至 4.14

這些指示假設您已經建立了私有叢集、並準備了安全的方法來遠端存取它。

您必須執行下列工作、才能將您的私有叢集新增至 Astra Control Service：

1. [安裝 Astra Connector](#)
2. [\[設定持續儲存\]](#)
3. [將私有自我管理叢集新增至 Astra Control Service](#)

安裝 Astra Connector

在新增私有叢集之前、您需要在叢集上安裝 Astra Connector、以便 Astra Control 能夠與其通訊。請參閱 ["安裝以非 Kubernetes 原生工作流程管理的舊版 Astra Connector for Private 叢集"](#) 以取得相關指示。

設定持續儲存

設定叢集的持續儲存設備。如需設定持續儲存設備的詳細資訊、請參閱入門文件：

- ["使用Azure NetApp Files 更新功能來設定Microsoft Azure"](#)
- ["使用Azure託管磁碟來設定Microsoft Azure"](#)

- ["設定Amazon Web Services"](#)
- ["設定Google Cloud"](#)

將私有自我管理叢集新增至 **Astra Control Service**

您現在可以將私有叢集新增至 Astra Control Service 。

自我管理的叢集是您直接進行資源配置與管理的叢集。Astra Control Service支援在公有雲環境中執行的自我管理叢集。您的自我管理叢集可以使用 Astra Control Provisioner 與 NetApp 儲存服務進行介面、或是使用 Container Storage Interface (CSI) 驅動程式與 Amazon Elastic Block Store (EBS)、Azure Managed Disks 及 Google Persistent Disk 進行介面。

Astra Control Service支援使用下列Kubernetes發佈版本的自我管理叢集：

- Red Hat OpenShift Container Platform
- Rancher Kubernetes引擎
- 上游Kubernetes

您的自我管理叢集必須符合下列需求：

- 叢集必須可透過網際網路存取。
- 如果您使用或打算使用已啟用SCSI驅動程式的儲存設備、則叢集上必須安裝適當的SCSI驅動程式。如需使用SCSI驅動程式來整合儲存設備的詳細資訊、請參閱儲存服務的說明文件。
- 您可以存取僅包含一個內容元素的叢集 kubeconfig 檔案。追蹤 [這些指示](#) 以產生 kubeconfig 檔案。
- 如果您要使用參考私有憑證授權單位 (CA) 的 kubeconfig 檔案來新增叢集、請將下列行新增至 cluster kubeconfig 檔案的一節。這可讓 Astra Control 新增叢集：

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

- *** 僅限Rancher ***：在Rancher環境中管理應用程式叢集時、請在Rancher提供的Kubeconfig檔案中修改應用程式叢集的預設內容、以使用控制面內容而非Rancher API伺服器內容。如此可減少Rancher API伺服器的負載、並改善效能。
- *** Astra Control Provisioner 需求 ***：您應該擁有正確設定的 Astra Control 資源配置程式、包括其 Astra Trident 元件、來管理叢集。
 - *** 檢閱 Astra Trident 環境需求 ***：在安裝或升級 Astra Control Provisioner 之前、請檢閱 [支援的前端、後端及主機組態](#)。
 - *** 啟用 Astra Control Provisioner 功能 ***：強烈建議您安裝 Astra Trident 23.10 或更新版本並啟用 ["Astra Control Provisioner 進階儲存功能"](#)。在後續版本中、如果 Astra Control 資源配置程式未啟用、Astra Control 將不支援 Astra Trident。
 - *** 設定儲存後端 ***：至少必須有一個儲存後端 ["在 Astra Trident 中設定"](#) 在叢集上。
 - *** 設定儲存類別 ***：至少必須有一個儲存類別 ["在 Astra Trident 中設定"](#) 在叢集上。如果已設定預設儲存類別、請確定它是具有預設註釋的 *** 僅 * 儲存類別**。
 - *** 設定 Volume Snapshot 控制器並安裝 Volume Snapshot 類別 ***：["安裝 Volume Snapshot 控制器"](#) 以便在Astra Control中建立快照。 ["建立"](#) 至少一個 VolumeSnapshotClass 使用 Astra Trident。

步驟

1. 在儀表板上、選取***管理Kubernetes叢集***。

依照提示新增叢集。

2. * 提供者 *：選取 * 其他 * 索引標籤以新增自我管理叢集的詳細資料。
3. 其他：上傳以提供自我管理叢集的詳細資料 `kubeconfig.yaml` 檔案或貼上的內容 `kubeconfig.yaml` 檔案。



如果您自行建立 `kubeconfig` 檔案中、您應該只定義*一個*內容元素。請參閱 ["這些指示"](#) 以取得有關建立的資訊 `kubeconfig` 檔案：

4. * 認證名稱 *：為您上傳至 Astra Control 的自我管理叢集認證提供名稱。根據預設、認證名稱會自動填入為叢集名稱。
5. * 私有路由識別碼 *：輸入可從 Astra Connector 取得的私有路由識別碼。如果您透過查詢 Astra Connector `kubectl get astraconnector -n astra-connector` 命令時、私有路由識別碼稱為 `ASTRACONNECTORID`。



私有路由識別碼是與 Astra Connector 相關的名稱、可讓 Astra 管理私有 Kubernetes 叢集。在這種情況下、私有叢集是 Kubernetes 叢集、不會將其 API 伺服器暴露於網際網路上。

6. 選擇*下一步*。
7. (可選) * Storage*：(可選) 選擇您希望 Kubernetes 應用程式部署到此叢集的儲存類別、以供預設使用。
 - a. 若要為叢集選取新的預設儲存類別、請啟用 * 指派新的預設儲存類別 * 核取方塊。
 - b. 從清單中選取新的預設儲存類別。

每個雲端供應商的儲存服務都會顯示下列價格、效能和恢復能力資訊：



- 適用於Google Cloud的解決方案：價格、效能和恢復能力資訊Cloud Volumes Service
- Google持續磁碟：沒有可用的價格、效能或恢復能力資訊
- 支援：效能與恢復能力資訊Azure NetApp Files
- Azure託管磁碟：不提供價格、效能或恢復能力資訊
- Amazon Elastic Block Store：沒有可用的價格、效能或恢復能力資訊
- Amazon FSX for NetApp ONTAP 不提供價格、效能或恢復能力資訊
- NetApp Cloud Volumes ONTAP 產品：不提供價格、效能或恢復能力資訊

每個儲存類別都可以使用下列其中一項服務：

- ["適用於 Google Cloud Cloud Volumes Service"](#)
- ["Google持續磁碟"](#)
- ["Azure NetApp Files"](#)
- ["Azure託管磁碟"](#)
- ["Amazon彈性區塊存放區"](#)
- ["Amazon FSX for NetApp ONTAP 產品"](#)

- ["NetApp Cloud Volumes ONTAP"](#)

深入瞭解 ["Amazon Web Services叢集的儲存類別"](#)。深入瞭解 ["適用於高效能叢集的儲存類別"](#)。深入瞭解 ["GKE叢集的儲存類別"](#)。

- c. 選擇*下一步*。
- d. * 審查與核准 *：檢閱組態詳細資料。
- e. 選取 * 新增 * 將叢集新增至 Astra Control Service。

變更預設儲存類別

您可以變更叢集的預設儲存類別。

使用Astra Control變更預設儲存類別

您可以從Astra Control中變更叢集的預設儲存類別。如果叢集使用先前安裝的儲存後端服務、您可能無法使用此方法來變更預設儲存類別（*設為預設*動作無法選取）。在這種情況下、您可以 [\[使用命令列變更預設儲存類別\]](#)。

步驟

1. 在Astra Control Service UI中、選取* Clusters*。
2. 在「叢集」頁面上、選取您要變更的叢集。
3. 選擇* Storage*（儲存設備）選項卡。
4. 選擇*儲存類別*類別。
5. 針對您要設為預設的儲存類別、選取「動作」功能表。
6. 選擇*設為預設*。

使用命令列變更預設儲存類別

您可以使用Kubernetes命令變更叢集的預設儲存類別。無論叢集的組態為何、此方法都能正常運作。

步驟

1. 登入Kubernetes叢集。
2. 列出叢集中的儲存類別：

```
kubectl get storageclass
```

3. 從預設儲存類別中移除預設指定。以<SC_NAME> 儲存類別的名稱取代支援：

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

4. 將不同的儲存類別標示為預設。以<SC_NAME> 儲存類別的名稱取代支援：

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

5. 確認新的預設儲存類別：

```
kubectl get storageclass
```

檢查 Astra Trident 版本

若要新增使用 Astra Control Provisioner 或 Astra Trident 進行儲存服務的自我管理叢集、請確定 Astra Trident 的安裝版本為 23.10 或最新版本。

步驟

1. 判斷您正在執行的 Astra Trident 版本：

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

如果安裝了Astra Trident、您會看到類似下列的輸出：

NAME	VERSION
trident	24.02.0

如果未安裝Astra Trident、您會看到類似下列的輸出：

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

2. 執行下列其中一項：

- 如果您執行的是 Astra Trident 23.01 或更早版本、請使用這些項目 ["說明"](#) 在升級至 Astra Control Provisioner 之前、先升級至較新版本的 Astra Trident。您可以 ["執行直接升級"](#) 如果您的 Astra Trident 位於 24.02 版的四個版本視窗內、則為 Astra Control Provisioner 24.02。例如、您可以直接從 Astra Trident 23.04 升級至 Astra Control Provisioner 24.02。
- 如果您執行的是 Astra Trident 23.10 或更新版本、請確認 Astra Control Provisioner 已完成 ["已啟用"](#)
 - Astra Control Provisioner 無法搭配 23.10 之前的 Astra Control Center 版本使用。 ["升級 Astra Control Provisioner"](#) 因此它與您要升級以存取最新功能的 Astra Control Center 版本相同。

3. 確保Pod正在執行：

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

4. 檢查儲存類別是否使用支援的Astra Trident驅動程式。置備程式名稱應為 `csi.trident.netapp.io`。請參閱下列範例：

```
kubectl get sc
```

回應範例：

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

建立Kubeconfig檔案

您可以使用 kubeconfig 檔案將叢集新增至 Astra Control Service。視您要新增的叢集類型而定、您可能需要使用特定步驟手動建立叢集的 kudeconfig 檔案。

- 為 Amazon EKS 叢集建立一個 kubeconfig 檔案
- 為 AWS (ROSA) 叢集上的 Red Hat OpenShift Service 建立一個 KRBconfig 檔案
- 為其他類型的叢集建立一個 kubeconfig 檔案

為 Amazon EKS 叢集建立一個 kubeconfig 檔案

請依照下列指示、為 Amazon EKS 叢集建立 Kribeconfig 檔案和永久性權杖機密。在 EKS 中託管的叢集需要永久權杖密碼。

步驟

1. 請依照 Amazon 文件中的指示來產生一個 kubeconfig 檔案：

"為 Amazon EKS 叢集建立或更新 Kribeconfig 檔案"

2. 建立服務帳戶、如下所示：

- a. 建立名為的服務帳戶檔案 `astracontrol-service-account.yaml`。

視需要調整服務帳戶名稱。命名空間 `kube-system` 為這些步驟所需。如果您在此處變更服務帳戶名稱、您應該在下列步驟中套用相同的變更。

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

+

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

3. 套用服務帳戶：

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

4. 建立 ClusterRoleBinding 檔案已呼叫 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. 套用叢集角色繫結：

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

6. 建立名為的服務帳戶權杖秘密檔案 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. 套用權杖密碼：

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

8. 擷取權杖密碼：

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

9. 更換 `user` AWS EKS Kubeconfig 檔案的區段與 Token、如下列範例所示：

```
user: token: k8s-aws-  
v1.aHR0cHM6Ly9zdHMudXMtd2VzdC0yLmFtYXpvbmF3cy5jb20vP0FjdGlvbj1HZXRDYWxsZ  
XJJZGVudG10eSZWZXJzaW9uPTIwMTETetMDYtMTUmWC1BbXotQWxnbn3JpdGhtPUFXUzQtSE1BQ  
y1TSEEyNTYmWC1BbXotQ3JlZGVudGllhbD1BS01BM1JEWddkU0haWU9LSEQ2SyUyRjIwMjMwN  
DAzJTJGdXMtd2VzdC0yJTJGc3RzJTJGYXdzNF9yZXFlZS01ZlgtZW16LURhdGU9MjAyMzA0M  
DNUMjA0MzQwWiZYLUFteilFeHBpcmVzPTYwJlgtZW16LVNpZ25lZEhlYWRLcnM9aG9zdCUzQ  
ngtazhzLWF3cy1pZCZYLUFteilTaWduYXRlcuU9YjU4ZWM0NzdiM2NkZGYxNGRhNzU4MGII2Z  
WQ2zy2NzI2YWIwM2UyNThjMjRhNTJjNmVhNjc4MTRlNjJkOTg2Mg
```

為 **AWS (ROSA)** 叢集上的 **Red Hat OpenShift Service** 建立一個 **KRBeconfig** 檔案

請依照下列指示、為 AWS (ROSA) 叢集上的 Red Hat OpenShift Service 建立一個 kubeconfig 檔案。

步驟

1. 登入 ROSA 叢集。
2. 建立服務帳戶：

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

3. 新增叢集角色：

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

4. 使用下列範例建立服務帳戶秘密組態檔案：

```
<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. 建立秘密：

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

6. 編輯您建立的服務帳戶、並將 Astra Control 服務帳戶密碼名稱新增至 secrets 區段：

```
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. 列出取代的服務帳戶機密 <CONTEXT> 正確的安裝環境：


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

輸出的結尾應類似於下列內容：

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

中每個元素的索引 secrets 陣列開頭為0。在上述範例中、索引為 astracontrol-service-account-dockercfg-dvfcd 將為0、索引則為 secret-astracontrol-service-account 應該是1。在輸出中、記下服務帳戶密碼的索引編號。您在下一個步驟中需要此索引編號。

8. 產生以下的Kbeconfig：

- a. 建立 create-kubeconfig.sh 檔案：更換 TOKEN_INDEX 在下列指令碼開頭、使用正確的值。

```
<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. 請輸入命令以將其套用至Kubernetes叢集。

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

9. (選用) 將Kbeconfig重新命名為有意義的叢集名稱。

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

為其他類型的叢集建立一個 **kubeconfig** 檔案

請依照下列指示、為 Rancher、上游 Kubernetes 和 Red Hat OpenShift 叢集建立有限或擴充的角色 kubeconfig 檔案。

對於使用 kubeconfig 管理的叢集、您可以選擇性地為 Astra Control Service 建立有限權限或擴充權限管理員角色。

如果下列任一情況適用於您的環境、本程序可協助您建立個別的 Kubeconfig：

- 您想要限制其管理叢集的 Astra Control 權限
- 您使用多個內容範圍、無法使用安裝期間設定的預設 Astra Control Kubeconfig、或是具有單一內容的受限角色、都無法在您的環境中運作

開始之前

在完成程序步驟之前、請確定您要管理的叢集具備下列項目：

- 答 ["支援的版本"](#) 已安裝 kubectl。
- 使用 Astra Control Service 存取您想要新增及管理的叢集



在本程序中、您不需要對執行 Astra Control Service 的叢集進行 kubectl 存取。

- 使用叢集管理權限來管理作用中內容的叢集的作用中KECBEConfig

步驟

1. 建立服務帳戶：

- a. 建立名為的服務帳戶檔案 `astracontrol-service-account.yaml`。

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

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

- b. 套用服務帳戶：

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

2. 為要由 Astra Control 管理的叢集建立具有足夠權限的下列叢集角色之一：

有限的叢集角色

此角色包含 Astra Control 管理叢集所需的最低權限：

- a. 建立 ClusterRole 例如、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
- replicaset
- replicationcontrollers
- replicationcontrollers/scale
- rolebindings
- roles
- secrets
- serviceaccounts
- services
- statefulsets
- tridentmirrorrelationships
- tridentnapshotinfos
- 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. (僅限 OpenShift 叢集) 在的結尾處附加下列項目 `astra-admin-account.yaml` 檔案：

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

- c. 套用叢集角色：

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

擴充叢集角色

此角色包含將由 Astra Control 管理之叢集的擴充權限。如果您使用多個內容範圍、且無法使用安裝期間設定的預設 Astra Control Kbeconfig、或是具有單一內容的有限角色無法在您的環境中運作、則可以使用此角色：



以下內容 ClusterRole 步驟是 Kubernetes 的一般範例。請參閱 Kubernetes 散佈文件、以取得特定於您環境的指示。

- a. 建立 ClusterRole 例如、`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. 套用叢集角色：

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

3. 建立叢集角色與服務帳戶的叢集角色繫結：

a. 建立 ClusterRoleBinding 檔案已呼叫 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. 套用叢集角色繫結：


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

4. 建立並套用權杖密碼：

- a. 建立一個稱為的權杖秘密檔案 `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. 套用權杖密碼：

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

5. 將權杖密碼新增至服務帳戶、將其名稱新增至 `secrets Array`（以下範例中的最後一行）：

```
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. 列出取代的服務帳戶機密 <context> 正確的安裝環境：

```

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

```

輸出的結尾應類似於下列內容：

```

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

```

中每個元素的索引 secrets 陣列開頭為0。在上述範例中、索引為 astracontrol-service-account-dockercfg-48xhx 將為0、索引則為 secret-astracontrol-service-account 應該是1。在輸出中、記下服務帳戶密碼的索引編號。您在下一個步驟中需要此索引編號。

7. 產生以下的Kbeconfig：

- 建立 create-kubeconfig.sh 檔案：
- 更換 TOKEN_INDEX 在下列指令碼開頭、使用正確的值。

```
<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=astraccontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astraccontrol
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. 請輸入命令以將其套用至Kubernetes叢集。

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

8. (選用) 將Kbeconfig重新命名為有意義的叢集名稱。

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

版權資訊

Copyright © 2024 NetApp, Inc. 版權所有。台灣印製。非經版權所有人事先書面同意，不得將本受版權保護文件的任何部分以任何形式或任何方法（圖形、電子或機械）重製，包括影印、錄影、錄音或儲存至電子檢索系統中。

由 NetApp 版權資料衍伸之軟體必須遵守下列授權和免責聲明：

此軟體以 NETAPP「原樣」提供，不含任何明示或暗示的擔保，包括但不限於有關適售性或特定目的適用性之擔保，特此聲明。於任何情況下，就任何已造成或基於任何理論上責任之直接性、間接性、附隨性、特殊性、懲罰性或衍生性損害（包括但不限於替代商品或服務之採購；使用、資料或利潤上的損失；或企業營運中斷），無論是在使用此軟體時以任何方式所產生的契約、嚴格責任或侵權行為（包括疏忽或其他）等方面，NetApp 概不負責，即使已被告知有前述損害存在之可能性亦然。

NetApp 保留隨時變本文所述之任何產品的權利，恕不另行通知。NetApp 不承擔因使用本文所述之產品而產生的責任或義務，除非明確經過 NetApp 書面同意。使用或購買此產品並不會在依據任何專利權、商標權或任何其他 NetApp 智慧財產權的情況下轉讓授權。

本手冊所述之產品受到一項（含）以上的美國專利、國外專利或申請中專利所保障。

有限權利說明：政府機關的使用、複製或公開揭露須受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中的「技術資料權利 - 非商業項目」條款 (b)(3) 小段所述之限制。

此處所含屬於商業產品和 / 或商業服務（如 FAR 2.101 所定義）的資料均為 NetApp, Inc. 所有。根據本協議提供的所有 NetApp 技術資料和電腦軟體皆屬於商業性質，並且完全由私人出資開發。美國政府對於該資料具有非專屬、非轉讓、非轉授權、全球性、有限且不可撤銷的使用權限，僅限於美國政府為傳輸此資料所訂合約所允許之範圍，並基於履行該合約之目的方可使用。除非本文另有規定，否則未經 NetApp Inc. 事前書面許可，不得逕行使用、揭露、重製、修改、履行或展示該資料。美國政府授予國防部之許可權利，僅適用於 DFARS 條款 252.227-7015(b)（2014 年 2 月）所述權利。

商標資訊

NETAPP、NETAPP 標誌及 <http://www.netapp.com/TM> 所列之標章均為 NetApp, Inc. 的商標。文中所涉及的所有其他公司或產品名稱，均為其各自所有者的商標，不得侵犯。