新增叢集

Astra Control Service

NetApp
April 24, 2024

目錄

將叢集新增至 Astra Control Service 1
安裝 Astra Connector 以管理叢集 1
新增由供應商管理的叢集 7
新增自我管理的叢集 16

將叢集新增至 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。這可讓您使用 Astra Control Service 來保護叢集上的應用程式。

視您需要新增至 Astra Control Service 的叢集類型而定、您需要使用不同的步驟來新增叢集。

- "將公有供應商託管的叢集新增至 Astra Control Service"：請使用以下步驟來新增具有公有 IP 位址且由雲端供應商管理的叢集。您需要雲端供應商的服務主體帳戶、服務帳戶或使用者帳戶。
- "將私有提供者管理的叢集新增至 Astra Control Service"：請使用以下步驟來新增具有私有 IP 位址且由雲端供應商管理的叢集。您需要雲端供應商的服務主體帳戶、服務帳戶或使用者帳戶。
- "將公用自我管理叢集新增至 Astra Control Service"：請使用以下步驟來新增具有公用 IP 位址且由組織管理的叢集。您需要為想要新增的叢集建立一個 kubeconfig 檔案。
- "將私有自我管理叢集新增至 Astra Control Service"：請使用以下步驟來新增具有私有 IP 位址且由組織管理的叢集。您需要為想要新增的叢集建立一個 kubeconfig 檔案。

安裝 Astra Connector 以管理叢集

Astra Connector 是一種軟體、位於託管叢集上、可促進託管叢集與 Astra Control 之間的通訊。對於使用 Astra Control Service 管理的叢集、有兩個可用版本的 Astra Connector ：

- * 舊版 Astra Connector* ： "安裝舊版 Astra Connector" 如果您計畫使用非 Kubernetes 原生工作流程來管理叢集、請在叢集上進行。
- [技術預覽] * 聲明性 Kubernetes Astra Connector* ： "安裝 Astra Connector for 叢集、以宣告式 Kubernetes 工作流程進行管理" 如果您打算使用宣告式 Kubernetes 工作流程來管理叢集、請在叢集上執行。在叢集上安裝 Astra Connector 之後、叢集會自動新增至 Astra Control 。

> 聲明性 Kubernetes Astra Connector 僅可作為 Astra Control Early Adopter Program （ EAP ）的一部分使用。請聯絡您的 NetApp 銷售代表、以取得加入 EAP 的相關資訊。

安裝舊版 Astra Connector

Astra Control Service 使用舊版 Astra Connector 、在 Astra Control Service 和使用非 Kubernetes 原生工作流程管理的私有叢集之間進行通訊。您需要在想要使用非 Kubernetes 原生工作流程管理的私有叢集上安裝 Astra Connector 。

舊版 Astra Connector 支援下列類型的私有叢集、這些叢集是以非 Kubernetes 原生工作流程管理：

- Amazon Elastic Kubernetes Service （ EKS ）
- Azure Kubernetes服務（KS）
- Google Kubernetes Engine（GKE）
- AWS 上的 Red Hat OpenShift 服務（ ROSA ）
- ROSA 搭配 AWS Private Link

- 內部部署 Red Hat OpenShift Container Platform

關於這項工作

- 當您執行這些步驟時、請針對您要使用 Astra Control Service 管理的私有叢集執行這些命令。
- 如果您使用的是堡壘主機、請從堡壘主機的命令列發出這些命令。

開始之前

- 您需要存取想要使用 Astra Control Service 管理的私有叢集。
- 您需要 Kubernetes 管理員權限、才能在叢集上安裝 Astra Connector 運算子。

步驟

1. 使用非 Kubernetes 原生工作流程、在您要管理的私有叢集上安裝先前的 Astra Connector 運算子。當您執行此命令時、命名空間 `astra-connector-operator` 已建立並套用組態至命名空間：

```
kubectl apply -f https://github.com/NetApp/astra-connector-operator/releases/download/23.07.0-202310251519/astraconnector_operator.yaml
```

2. 確認操作員已安裝就緒：

```
kubectl get all -n astra-connector-operator
```

3. 從 Astra Control 取得 API 權杖。請參閱 "Astra Automation文件" 以取得相關指示。
4. 建立 Astra 連接器命名空間：

```
kubectl create ns astra-connector
```

5. 建立 Astra Connector CR 檔案並命名 `astra-connector-cr.yaml`。更新括弧 <> 中的值以符合 Astra Control 環境和叢集組態：
   - * <ASTRA_CONTROL_SERVICE_URL> * ： Astra 控制服務的網路 UI URL 。例如：

     ```
     https://astra.netapp.io
     ```

   - * <ASTRA_CONTROL_SERVICE_API_TOKEN> * ：您在上述步驟中取得的 Astra 控制 API 權杖。
   - * <PRIVATE_AKS_CLUSTER_NAME> * ：（僅限 AKS 叢集） - 私有 Azure Kubernetes 服務叢集的叢集名稱。只有在您要新增私有 aks 叢集時、才取消註解並填入此行。
   - * <ASTRA_CONTROL_ACCOUNT_ID> * ：從 Astra Control 網路 UI 取得。選取頁面右上角的圖示、然後選取 * API 存取 * 。

```yaml
apiVersion: netapp.astraconnector.com/v1
kind: AstraConnector
metadata:
  name: astra-connector
  namespace: astra-connector
spec:
  natssync-client:
    cloud-bridge-url: <ASTRA_CONTROL_SERVICE_URL>
  imageRegistry:
    name: theotw
    secret: ""
  astra:
    token: <ASTRA_CONTROL_SERVICE_API_TOKEN>
    #clusterName: <PRIVATE_AKS_CLUSTER_NAME>
    accountId: <ASTRA_CONTROL_ACCOUNT_ID>
    acceptEULA: yes
```

6. 填入之後 `astra-connector-cr.yaml` 使用正確值的檔案、請套用 CR ：

```
kubectl apply -f astra-connector-cr.yaml
```

7. 確認 Astra Connector 已完全部署：

```
kubectl get all -n astra-connector
```

8. 確認叢集已向 Astra Control 註冊：

```
kubectl get astraconnector -n astra-connector
```

您應該會看到類似下列的輸出：

```
NAME                REGISTERED   ASTRACONNECTORID
STATUS
astra-connector     true         be475ae5-1511-4eaa-9b9e-712f09b0d065
Registered with Astra
```

記下 ASTRACONECTORID ；將叢集新增至 Astra Control 時、您將需要它。

接下來呢？

安裝 Astra Connector 之後、您就可以將私有叢集新增至 Astra Control Service 。

- "將私有提供者管理的叢集新增至 Astra Control Service"：請使用以下步驟來新增具有私有 IP 位址且由雲端供應商管理的叢集。您需要雲端供應商的服務主體帳戶、服務帳戶或使用者帳戶。
- "將私有自我管理叢集新增至 Astra Control Service"：請使用以下步驟來新增具有私有 IP 位址且由組織管理的叢集。您需要為想要新增的叢集建立一個 kubeconfig 檔案。

以取得更多資訊

- "新增叢集"

（技術預覽）安裝宣告性 **Kubernetes Astra Connector**

使用宣告式 Kubernetes 工作流程管理的叢集使用 Astra Connector 來啟用託管叢集與 Astra Control 之間的通訊。您需要在所有要使用宣告式 Kubernetes 工作流程管理的叢集上安裝 Astra Connector 。

您可以使用 Kubernetes 命令和自訂資源（ CR ）檔案來安裝宣告式 Kubernetes Astra Connector 。

關於這項工作

- 執行這些步驟時、請在您要使用 Astra Control 管理的叢集上執行這些命令。
- 如果您使用的是堡壘主機、請從堡壘主機的命令列發出這些命令。

開始之前

- 您需要存取想要使用 Astra Control 管理的叢集。
- 您需要 Kubernetes 管理員權限、才能在叢集上安裝 Astra Connector 運算子。

> 如果叢集已設定 Pod 安全許可強制（ Kubernetes 1.25 及更新叢集的預設值）、則您需要針對適當的命名空間啟用 PSA 限制。請參閱 "使用Astra Control為環境做好叢集管理準備" 以取得相關指示。

步驟

1. 使用宣告式 Kubernetes 工作流程、在您要管理的叢集上安裝 Astra Connector 運算子。當您執行此命令時、命名空間 `astra-connector-operator` 已建立並套用組態至命名空間：

```
kubectl apply -f https://github.com/NetApp/astra-connector-
operator/releases/download/24.02.0-
202403151353/astraconnector_operator.yaml
```

2. 確認操作員已安裝就緒：

```
kubectl get all -n astra-connector-operator
```

3. 從 Astra Control 取得 API 權杖。請參閱 "Astra Automation文件" 以取得相關指示。
4. 使用權杖建立秘密。將 <API_TOKEN> 取代為您從 Astra Control 收到的權杖：

```
kubectl create secret generic astra-token \
--from-literal=apiToken=<API_TOKEN> \
-n astra-connector
```

5. 建立 Docker 秘密以用於拉出 Astra Connector 映像。以您環境的資訊取代括弧 <> 中的值：

> 您可以在 Astra 控制網路 UI 中找到 <ASTRA_CONTROL_ACCOUNT_ID> 。在 Web UI 中、選取頁面右上角的圖示、然後選取 * API 存取 * 。

```
kubectl create secret docker-registry regcred \
--docker-username=<ASTRA_CONTROL_ACCOUNT_ID> \
--docker-password=<API_TOKEN> \
-n astra-connector \
--docker-server=cr.astra.netapp.io
```

6. 建立 Astra Connector CR 檔案並命名 `astra-connector-cr.yaml`。更新括弧 <> 中的值以符合 Astra Control 環境和叢集組態：
   - <ASTRA_CONTROL_ACCOUNT_ID> ：在前一個步驟中從 Astra Control 網路 UI 取得。
   - <CLUSTER_NAME> ：應在 Astra Control 中指派此叢集的名稱。
   - <ASTRA_CONTROL_URL> ： Astra Control 的網路 UI URL 。例如：

     ```
     https://astra.control.url
     ```

```yaml
apiVersion: astra.netapp.io/v1
kind: AstraConnector
metadata:
  name: astra-connector
  namespace: astra-connector
spec:
  astra:
    accountId: <ASTRA_CONTROL_ACCOUNT_ID>
    clusterName: <CLUSTER_NAME>
    #Only set `skipTLSValidation` to `true` when using the default
self-signed
    #certificate in a proof-of-concept environment.
    skipTLSValidation: false #Should be set to false in production
environments
    tokenRef: astra-token
  natsSyncClient:
    cloudBridgeURL: <ASTRA_CONTROL_HOST_URL>
  imageRegistry:
    name: cr.astra.netapp.io
    secret: regcred
```

7. 填入之後 `astra-connector-cr.yaml` 使用正確值的檔案、請套用 CR ：

```
kubectl apply -n astra-connector -f astra-connector-cr.yaml
```

8. 確認 Astra Connector 已完全部署：

```
kubectl get all -n astra-connector
```

9. 確認叢集已向 Astra Control 註冊：

```
kubectl get astraconnectors.astra.netapp.io -A
```

您應該會看到類似下列的輸出：

```
NAMESPACE          NAME               REGISTERED   ASTRACONNECTORID
STATUS
astra-connector    astra-connector    true         00ac8-2cef-41ac-8777-
ed0583e   Registered with Astra
```

10. 驗證叢集是否出現在 Astra Control 網路 UI 的 * Clusters* 頁面上的受管理叢集清單中。

新增由供應商管理的叢集

將公有供應商託管的叢集新增至 **Astra Control Service**

在您設定雲端環境之後、您就可以建立 Kubernetes 叢集、然後將其新增至 Astra Control Service 。

- 建立Kubernetes叢集
- 將叢集新增至 Astra Control Service
- [變更預設儲存類別]

建立**Kubernetes**叢集

如果您還沒有叢集、可以建立符合的叢集 "Amazon Elastic Kubernetes服務（EKS）的Astra控制服務要求"。如果您還沒有叢集、可以建立符合的叢集 "Google Kubernetes Engine（GKE）的Astra控制服務要求"。如果您還沒有叢集、可以建立符合的叢集 "Azure Kubernetes Service（含Azure NetApp Files ）的Astra Control Service要求" 或 "Azure Kubernetes Service（Astra Control Service）與Azure託管磁碟的相關要求"。

> Astra Control Service支援使用Azure Active Directory（Azure AD）進行驗證與身分識別管理的高峰叢集。建立叢集時、請遵循中的指示 "正式文件" 設定叢集使用Azure AD。您必須確保叢集符合高峰管理Azure AD整合的要求。

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。

當您管理 Azure Kubernetes Service 和 Google Kubernetes Engine 叢集時、請注意 Astra Control Provisioner 的安裝和生命週期管理有兩個選項：

- 您可以使用 Astra Control Service 來自動管理 Astra Control Provisioner 的生命週期。為達成此目的、請確定未安裝 Astra Trident 、且 Astra Control Provisioner 未在您要使用 Astra Control Service 管理的叢集上啟用。在這種情況下、 Astra Control Service 會在您開始管理叢集時自動啟用 Astra Control Provisioner 、並自動處理 Astra Control Provisioner 升級。
- 您可以自行管理 Astra Control 資源配置程式的生命週期。若要執行此作業、請先在叢集上啟用 Astra Control Provisioner 、然後再使用 Astra Control Service 管理叢集。在這種情況下、 Astra Control Service 會偵測到 Astra Control Provisioner 已啟用、而且不會重新安裝或管理 Astra Control Provisioner 升級。請參閱 "啟用 Astra Control Provisioner" 針對步驟、請啟用 Astra Control Provisioner 。

當您使用 Astra Control Service 管理 Amazon Web Services 叢集時、如果您需要的儲存後端只能與 Astra Control Provisioner 搭配使用、則必須先在叢集上手動啟用 Astra Control Provisioner 、然後才能使用 Astra Control Service 進行管理。請參閱 "啟用 Astra Control Provisioner" 適用於啟用 Astra Control Provisioner 的步驟。

開始之前

Amazon Web Services

- 您應該擁有Json檔案、其中包含建立叢集的IAM使用者認證。 "瞭解如何建立IAM使用者"。
- Amazon FSX for NetApp ONTAP 需要 Astra Control Provisioner 。如果您打算將 Amazon FSX for NetApp ONTAP 作為 EKS 叢集的儲存後端、請參閱中的 Astra Control Provisioner 資訊 "EKS叢集需求"。
- （選用）如果您需要提供 `kubectl` 叢集的命令存取功能可讓其他不是叢集建立者的IAM使用者存取、請參閱中的指示 "如何在Amazon EKS中建立叢集後、提供其他IAM使用者和角色的存取權限？"。
- 如果您計畫將NetApp Cloud Volumes ONTAP 支援作為儲存後端、則需要設定Cloud Volumes ONTAP 支援以搭配Amazon Web Services使用的功能。請參閱Cloud Volumes ONTAP 《The》 "設定文件"。

Microsoft Azure

- 建立服務主體時、您應該擁有包含Azure CLI輸出的Json檔案。 "瞭解如何設定服務主體"。

  如果您未將Azure訂閱ID新增至Json檔案、您也需要Azure訂閱ID。

- 如果您計畫將NetApp Cloud Volumes ONTAP 支援作為儲存後端、則需要設定Cloud Volumes ONTAP 支援功能以搭配Microsoft Azure使用。請參閱Cloud Volumes ONTAP 《The》 "設定文件"。

Google Cloud

- 您應該擁有具有所需權限之服務帳戶的服務帳戶金鑰檔。 "瞭解如何設定服務帳戶"。
- 如果您打算將NetApp Cloud Volumes ONTAP 支援作為儲存後端、則需要設定Cloud Volumes ONTAP 支援功能以搭配Google Cloud使用。請參閱Cloud Volumes ONTAP 《The》 "設定文件"。

步驟

1. （選用）如果您要新增 Amazon EKS 叢集、或想要自行管理 Astra Control Provisioner 的安裝與升級、請在叢集上啟用 Astra Control Provisioner 。請參閱 "啟用 Astra Control Provisioner" 以瞭解啟用步驟。
2. 在瀏覽器中開啟 Astra Control Service Web UI 。
3. 在儀表板上、選取*管理Kubernetes叢集*。

   依照提示新增叢集。

4. 供應商：選取您的雲端供應商、然後提供必要的認證資料以建立新的雲端執行個體、或選取要使用的現有雲端執行個體。
5. * Amazon Web Services *：上傳Json檔案或從剪貼簿貼上Json檔案的內容、以提供Amazon Web Services IAM使用者帳戶的詳細資料。

   Json檔案應包含建立叢集的IAM使用者認證。

6. * Microsoft Azure *：上傳Json檔案或從剪貼簿貼上Json檔案的內容、以提供Azure服務主體的詳細資料。

   當您建立服務主體時、Json檔案應包含Azure CLI的輸出。它也可以包含您的訂閱ID、以便自動新增至Astra。否則、您必須在提供Json之後手動輸入ID。

7. * Google Cloud Platform *：上傳檔案或從剪貼簿貼上內容、以提供服務帳戶金鑰檔案。

Astra Control Service使用服務帳戶來探索在Google Kubernetes Engine中執行的叢集。

8. * 其他 * ：此索引標籤僅適用於自我管理叢集。
   a. * Cloud 執行個體名稱 * ：為新增叢集時將建立的新雲端執行個體提供名稱。深入瞭解 "雲端執行個體"。
   b. 選擇*下一步*。

      Astra Control Service 會顯示您可以選擇的叢集清單。

   c. * 叢集 * ：從清單中選取要新增至 Astra Control Service 的叢集。

      > 當您從叢集清單中選取時、請注意 **Eligiblity** 欄。如果叢集為「不合格」或「部分合格」、請將游標移至狀態上方、以判斷叢集是否有問題。例如、它可能會識別叢集沒有工作節點。

   d. 選擇*下一步*。
   e. （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。
9. 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。
10. 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"。

a. 選擇*下一步*。

b. * 審查與核准 * ：檢閱組態詳細資料。

c. 選取 * 新增 * 將叢集新增至 Astra Control Service 。

結果

如果這是您為此雲端供應商新增的第一個叢集、 Astra Control Service 會為雲端供應商建立物件儲存區、以便備份在合格叢集上執行的應用程式。（當您新增此雲端供應商的後續叢集時、不會再建立其他物件存放區。）如果您指定預設儲存類別、Astra Control Service會設定您指定的預設儲存類別。對於在Amazon Web Services或Google Cloud Platform中管理的叢集、Astra Control Service也會在叢集上建立管理員帳戶。這些動作可能需要幾分鐘的時間。

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
```

將私有提供者管理的叢集新增至 **Astra Control Service**

您可以使用 Astra Control Service 來管理私有 Google Kubernetes Engine （GKE）叢集。這些指示假設您已建立私有 aks 或 OpenShift 叢集、並準備了安全的方法來遠端存取；如需建立及存取私有 aks 或 OpenShift 叢集的詳細資訊、請參閱下列文件：

- "適用於私有 aks 叢集的 Azure 文件"
- "適用於私有 OpenShift 叢集的 Azure 文件"

您可以使用 Astra Control Service 來管理私有 Azure Kubernetes Service （Astra Control Service）叢集、以及在 AKS 中管理私有 Red Hat OpenShift 叢集。這些指示假設您已建立私有 aks 或 OpenShift 叢集、並準備了安全的方法來遠端存取；如需建立及存取私有 aks 或 OpenShift 叢集的詳細資訊、請參閱下列文件：

- "適用於私有 aks 叢集的 Azure 文件"
- "適用於私有 OpenShift 叢集的 Azure 文件"

您可以使用 Astra Control Service 來管理私有 Amazon Elastic Kubernetes Service （EKS）叢集。這些指示假設您已建立私有 EKS 叢集、並準備了安全的方法來遠端存取；如需建立及存取私有 EKS 叢集的詳細資訊、請參閱 "Amazon EKS 文件"。

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. 將私有提供者管理的叢集新增至 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"

將私有提供者管理的叢集新增至 **Astra Control Service**

您現在可以將私有叢集新增至 Astra Control Service 。

當您管理 Azure Kubernetes Service 和 Google Kubernetes Engine 叢集時、請注意 Astra Control Provisioner 的安裝和生命週期管理有兩個選項：

- 您可以使用 Astra Control Service 來自動管理 Astra Control Provisioner 的生命週期。為達成此目的、請確定未安裝 Astra Trident 、且 Astra Control Provisioner 未在您要使用 Astra Control Service 管理的叢集上啟用。在這種情況下、 Astra Control Service 會在您開始管理叢集時自動啟用 Astra Control Provisioner 、並自動處理 Astra Control Provisioner 升級。
- 您可以自行管理 Astra Control 資源配置程式的生命週期。若要執行此作業、請先在叢集上啟用 Astra Control Provisioner 、然後再使用 Astra Control Service 管理叢集。在這種情況下、 Astra Control Service 會偵測到 Astra Control Provisioner 已啟用、而且不會重新安裝或管理 Astra Control Provisioner 升級。請參閱 "啟用 Astra Control Provisioner" 針對步驟、請啟用 Astra Control Provisioner 。

當您使用 Astra Control Service 管理 Amazon Web Services 叢集時、如果您需要的儲存後端只能與 Astra Control Provisioner 搭配使用、則必須先在叢集上手動啟用 Astra Control Provisioner 、然後才能使用 Astra Control Service 進行管理。請參閱 "啟用 Astra Control Provisioner" 適用於啟用 Astra Control Provisioner 的步驟。

開始之前

Amazon Web Services

- 您應該擁有Json檔案、其中包含建立叢集的IAM使用者認證。 "瞭解如何建立IAM使用者"。
- Amazon FSX for NetApp ONTAP 需要 Astra Control Provisioner 。如果您打算將 Amazon FSX for NetApp ONTAP 作為 EKS 叢集的儲存後端、請參閱中的 Astra Control Provisioner 資訊 "EKS叢集需求"。
- （選用）如果您需要提供 `kubectl` 叢集的命令存取功能可讓其他不是叢集建立者的IAM使用者存取、請參閱中的指示 "如何在Amazon EKS中建立叢集後、提供其他IAM使用者和角色的存取權限？"。
- 如果您計畫將NetApp Cloud Volumes ONTAP 支援作為儲存後端、則需要設定Cloud Volumes ONTAP 支援以搭配Amazon Web Services使用的功能。請參閱Cloud Volumes ONTAP 《The》 "設定文件"。

Microsoft Azure

- 建立服務主體時、您應該擁有包含Azure CLI輸出的Json檔案。 "瞭解如何設定服務主體"。

  如果您未將Azure訂閱ID新增至Json檔案、您也需要Azure訂閱ID。

- 如果您計畫將NetApp Cloud Volumes ONTAP 支援作為儲存後端、則需要設定Cloud Volumes ONTAP 支援功能以搭配Microsoft Azure使用。請參閱Cloud Volumes ONTAP 《The》 "設定文件"。

Google Cloud

- 您應該擁有具有所需權限之服務帳戶的服務帳戶金鑰檔。 "瞭解如何設定服務帳戶"。
- 如果叢集為私有、則為 "授權網路" 必須允許Astra控制服務IP位址：

  52.188.218.166/32

- 如果您打算將NetApp Cloud Volumes ONTAP 支援作為儲存後端、則需要設定Cloud Volumes ONTAP 支援功能以搭配Google Cloud使用。請參閱Cloud Volumes ONTAP 《The》 "設定文件"。

步驟

1. （選用）如果您要新增 Amazon EKS 叢集、或想要自行管理 Astra Control Provisioner 的安裝與升級、請在叢集上啟用 Astra Control Provisioner 。請參閱 "啟用 Astra Control Provisioner" 以瞭解啟用步驟。
2. 在瀏覽器中開啟 Astra Control Service Web UI 。
3. 在儀表板上、選取*管理Kubernetes叢集*。

   依照提示新增叢集。

4. 供應商：選取您的雲端供應商、然後提供必要的認證資料以建立新的雲端執行個體、或選取要使用的現有雲端執行個體。
5. * Amazon Web Services *：上傳Json檔案或從剪貼簿貼上Json檔案的內容、以提供Amazon Web Services IAM使用者帳戶的詳細資料。

   Json檔案應包含建立叢集的IAM使用者認證。

6. * Microsoft Azure *：上傳Json檔案或從剪貼簿貼上Json檔案的內容、以提供Azure服務主體的詳細資料。

   當您建立服務主體時、Json檔案應包含Azure CLI的輸出。它也可以包含您的訂閱ID、以便自動新增至

Astra。否則、您必須在提供Json之後手動輸入ID。

7. * Google Cloud Platform *：上傳檔案或從剪貼簿貼上內容、以提供服務帳戶金鑰檔案。

   Astra Control Service使用服務帳戶來探索在Google Kubernetes Engine中執行的叢集。

8. * 其他 * ：此索引標籤僅適用於自我管理叢集。
   a. * Cloud 執行個體名稱 * ：為新增叢集時將建立的新雲端執行個體提供名稱。深入瞭解 "雲端執行個體"。
   b. 選擇*下一步*。

      Astra Control Service 會顯示您可以選擇的叢集清單。

   c. * 叢集 * ：從清單中選取要新增至 Astra Control Service 的叢集。

      > 當您從叢集清單中選取時、請注意 **Eligiblity** 欄。如果叢集為「不合格」或「部分合格」、請將游標移至狀態上方、以判斷叢集是否有問題。例如、它可能會識別叢集沒有工作節點。

9. 選擇*下一步*。
10. （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 。

結果

如果這是您為此雲端供應商新增的第一個叢集、 Astra Control Service 會為雲端供應商建立物件儲存區、以便備份在合格叢集上執行的應用程式。（當您新增此雲端供應商的後續叢集時、不會再建立其他物件存放區。）如果您指定預設儲存類別、Astra Control Service會設定您指定的預設儲存類別。對於在Amazon Web Services或Google Cloud Platform中管理的叢集、Astra Control Service也會在叢集上建立管理員帳戶。這些動作可能需要幾分鐘的時間。

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
```

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。

開始之前

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Kusbeconfig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 。

開始之前

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Kusbeconfig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Kbeconfig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 KRBeconfig 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>
```

+

```yaml
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>
```

```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. 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
XJJZGVudGl0eSZWZXJzaW9uPTIwMTEtMDYtMTUmWC1BbXotQWxnb3JpdGhtPUFXUzQtSE1BQ
y1TSEEyNTYmWC1BbXotQ3JlZGVudGlhbD1BS0lBM1JEWDdKU0haWU9LSEQ2SyUyRjIwMjMwN
DAzJTJGdXMtd2VzdC0yJTJGc3RzJTJGYXdzNF9yZXF1ZXN0JlgtQW16LURhdGU9MjAyMzA0M
DNUMjA0MzQwWiZYLUFtei1FeHBpcmVzPTYwJlgtQW16LVNpZ25lZEhlYWRlcnM9aG9zdCUzQ
ngtazhzLWF3cy1pZCZYLUFtei1TaWduYXR1cmU9YjU4ZWM0NzdiM2NkZGYxNGRhNzU4MGI2Z
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>
```

```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. 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
```

```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. 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 Kubleconfig ：

- 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 Kbeconfig 、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>
```

```yaml
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>
```

```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. （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>
```

```yaml
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>
```

```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. 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>
```

```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. 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：
   a. 建立 `create-kubeconfig.sh` 檔案：
   b. 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
```

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
```