安裝總覽

Astra Control Center

NetApp
August 11, 2025

目錄

安裝總覽 1
使用標準程序安裝Astra Control Center 1
下載並擷取Astra Control Center 2
安裝NetApp Astra kubecl外掛程式 3
將映像新增至本機登錄 3
設定具有驗證需求之登錄的命名空間和機密 5
安裝Astra Control Center操作員 7
設定Astra控制中心 10
完整的Astra控制中心和操作員安裝 19
驗證系統狀態 20
設定入口以進行負載平衡 25
登入Astra Control Center UI 29
疑難排解安裝 29
下一步 30
設定外部憑證管理程式 30
使用OpenShift作業系統集線器安裝Astra Control Center 31
下載並擷取Astra Control Center 32
安裝NetApp Astra kubecl外掛程式 33
將映像新增至本機登錄 33
尋找操作員安裝頁面 35
安裝操作員 36
安裝Astra Control Center 37
建立登錄機密 38
下一步 39
安裝Astra Control Center搭配Cloud Volumes ONTAP 一套功能性儲存後端 39
在Amazon Web Services中部署Astra Control Center 39
在Google Cloud Platform中部署Astra Control Center 44
在Microsoft Azure中部署Astra Control Center 48
安裝後設定Astra Control Center 53
移除資源限制 53
啟用命名空間之間的網路通訊 55
新增自訂TLS憑證 56

安裝總覽

選擇並完成下列其中一個Astra Control Center安裝程序：

- "使用標準程序安裝Astra Control Center"
- "（如果您使用Red Hat OpenShift）使用OpenShift作業系統集線器安裝Astra Control Center"
- "安裝Astra Control Center搭配Cloud Volumes ONTAP 一套功能性儲存後端"

視您的環境而定、安裝Astra Control Center之後可能需要額外的組態：

- "安裝後設定Astra Control Center"

使用標準程序安裝Astra Control Center

若要安裝Astra Control Center、請從NetApp 支援網站 下列網址下載安裝套件、並執行下列步驟。您可以使用此程序、在連線網際網路或無線環境中安裝Astra Control Center。

其他安裝程序

- *使用RedHat OpenShift操作員中樞*安裝：請使用此功能 "替代程序" 使用作業系統集線器在OpenShift上安裝Astra Control Center。
- 以**Cloud Volumes ONTAP** 支援功能的方式在公有雲上安裝：使用 "這些程序" 若要在Amazon Web Services（AWS）、Google Cloud Platform（GCP）或Microsoft Azure中安裝Astra Control Center、並提供Cloud Volumes ONTAP 一套支援整合式儲存後端的功能。

如需 Astra Control Center 安裝程序的示範，請參閱 "這段影片"。

開始之前

- "開始安裝之前、請先準備好環境以進行Astra Control Center部署"。
- 如果您已設定或想要在環境中設定Pod安全性原則、請熟悉Pod安全性原則、以及這些原則如何影響Astra Control Center安裝。請參閱 "瞭解Pod安全性原則限制"。
- 確保所有API服務均處於健全狀態且可供使用：

```
kubectl get apiservices
```

- 確保您打算使用的Astra FQDN可路由傳送至此叢集。這表示您在內部DNS伺服器中有DNS項目、或是使用已註冊的核心URL路由。
- 如果叢集中已存在憑證管理程式、您需要執行某些作業 "必要步驟" 因此Astra Control Center不會嘗試安裝自己的憑證管理程式。依預設、Astra Control Center會在安裝期間安裝自己的憑證管理程式。

在第三個故障網域或次要站台中部署 Astra Control Center 。這是應用程式複寫和無縫災難恢復的建議。

關於這項工作

Astra Control Center安裝程序可協助您執行下列作業：

- 將Astra元件安裝至 `netapp-acc` （或自訂命名）命名空間。
- 建立預設的Astra Control擁有者管理帳戶。
- 建立管理使用者電子郵件地址和預設初始設定密碼。此使用者會被指派第一次登入UI時所需的擁有者角色。
- 確定所有Astra Control Center Pod都在執行中。
- 安裝Astra Control Center UI。

> ⓘ 請勿刪除Astra Control Center運算子（例如、 `kubectl delete -f astra_control_center_operator_deploy.yaml`）在Astra Control Center安裝或操作期間、隨時避免刪除Pod。

步驟

若要安裝Astra Control Center、請執行下列步驟：

- 下載並擷取Astra Control Center
- 安裝NetApp Astra kubecl外掛程式
- [將映像新增至本機登錄]
- [設定具有驗證需求之登錄的命名空間和機密]
- 安裝Astra Control Center操作員
- 設定Astra控制中心
- 完整的Astra控制中心和操作員安裝
- [驗證系統狀態]
- [設定入口以進行負載平衡]
- 登入Astra Control Center UI

下載並擷取Astra Control Center

1. 前往 "Astra Control Center 下載頁面" 於 NetApp 支援網站。
2. 下載包含Astra Control Center的套裝組合 (`astra-control-center-[version].tar.gz`) 。
3. （建議但可選）下載Astra Control Center的憑證與簽名套件 (`astra-control-center-certs-[version].tar.gz`）若要驗證套件的簽名：

```
tar -vxzf astra-control-center-certs-[version].tar.gz
```

```
openssl dgst -sha256 -verify certs/AstraControlCenter-public.pub
-signature certs/astra-control-center-[version].tar.gz.sig astra-
control-center-[version].tar.gz
```

隨即顯示輸出 `Verified OK` 驗證成功之後。

4. 從Astra Control Center套裝組合擷取映像：

```
tar -vxzf astra-control-center-[version].tar.gz
```

安裝NetApp Astra kubecl外掛程式

您可以使用 NetApp Astra kubectl 命令列外掛程式、將影像推送至本機 Docker 儲存庫。

開始之前

NetApp為不同的CPU架構和作業系統提供外掛程式二進位檔。執行此工作之前、您必須先瞭解您的CPU和作業系統。

如果您已從先前的安裝中安裝外掛程式、 "請確定您擁有最新版本" 完成這些步驟之前。

步驟

1. 列出可用的NetApp Astra kubectl外掛程式二進位檔、並記下作業系統和CPU架構所需的檔案名稱：

   KECBECTl外掛程式庫是tar套件的一部分、會擷取到資料夾中 `kubectl-astra`。

```
ls kubectl-astra/
```

2. 將正確的二進位檔移至目前路徑、並將其重新命名為 `kubectl-astra`：

```
cp kubectl-astra/<binary-name> /usr/local/bin/kubectl-astra
```

將映像新增至本機登錄

1. 為您的Container引擎完成適當的步驟順序：

Docker

1. 切換到tar檔案的根目錄。您應該會看到這個檔案和目錄：

   ```
   acc.manifest.bundle.yaml
   acc/
   ```

2. 將Astra Control Center映像目錄中的套件映像推送到本機登錄。執行之前、請先進行下列替換 `push-images` 命令：

   - 以<BUNDLE_FILE> Astra Control套裝組合檔案的名稱取代 (`acc.manifest.bundle.yaml`) 。
   - 以<MY_FULL_REGISTRY_PATH> Docker儲存庫的URL取代支援；例如 "<a href="https://<docker-registry>"" class="bare">https://<docker-registry>"。
   - 以<MY_REGISTRY_USER> 使用者名稱取代。
   - 以<MY_REGISTRY_TOKEN> 登錄的授權權杖取代。

     ```
     kubectl astra packages push-images -m <BUNDLE_FILE> -r
     <MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
     <MY_REGISTRY_TOKEN>
     ```

Podman

1. 切換到tar檔案的根目錄。您應該會看到這個檔案和目錄：

   ```
   acc.manifest.bundle.yaml
   acc/
   ```

2. 登入您的登錄：

   ```
   podman login <YOUR_REGISTRY>
   ```

3. 針對您使用的Podman版本、準備並執行下列其中一個自訂指令碼。以包含任何子目錄的儲存庫URL取代<MY_FULL_REGISTRY_PATH> 。

   ```
   <strong>Podman 4</strong>
   ```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.04.2-7
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/${
PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.04.2-7
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/${
PACKAGEVERSION}/${astraImageNoPath}
done
```

指令碼所建立的映像路徑應如下所示、視登錄組態而定：

```
https://netappdownloads.jfrog.io/docker-astra-control-
prod/netapp/astra/acc/23.04.2-7/image:version
```

設定具有驗證需求之登錄的命名空間和機密

1. 匯出Astra Control Center主機叢集的KUBECCONFIG：

```
export KUBECONFIG=[file path]
```

完成安裝之前、請確定KUBECCONFIG指向您要安裝Astra Control Center的叢集。KUBECCONFIG只能包含一個內容。

2. 如果您使用需要驗證的登錄、則需要執行下列動作：

   a. 建立 `netapp-acc-operator` 命名空間：

   ```
   kubectl create ns netapp-acc-operator
   ```

   回應：

   ```
   namespace/netapp-acc-operator created
   ```

   b. 為建立秘密 `netapp-acc-operator` 命名空間。新增Docker資訊並執行下列命令：

   預留位置 `your_registry_path` 應與您先前上傳的影像位置相符（例如、`[Registry_URL]/netapp/astra/astracc/23.04.2-7`）。

   ```
   kubectl create secret docker-registry astra-registry-cred -n netapp-
   acc-operator --docker-server=[your_registry_path] --docker
   -username=[username] --docker-password=[token]
   ```

   回應範例：

   ```
   secret/astra-registry-cred created
   ```

   如果在產生機密之後刪除命名空間、請重新建立命名空間、然後重新產生命名空間的機密。

   c. 建立 `netapp-acc` （或自訂命名）命名空間。

   ```
   kubectl create ns [netapp-acc or custom namespace]
   ```

   回應範例：

   ```
   namespace/netapp-acc created
   ```

d. 為建立秘密 `netapp-acc` （或自訂命名）命名空間。新增Docker資訊並執行下列命令：

```
kubectl create secret docker-registry astra-registry-cred -n [netapp-acc or custom namespace] --docker-server=[your_registry_path] --docker-username=[username] --docker-password=[token]
```

回應

```
secret/astra-registry-cred created
```

安裝Astra Control Center操作員

1. 變更目錄：

```
cd manifests
```

2. 編輯Astra Control Center營運者部署Yaml (`astra_control_center_operator_deploy.yaml`）以參考您的本機登錄和機密。

```
vim astra_control_center_operator_deploy.yaml
```

附註的Y反 洗錢範例遵循下列步驟。

a. 如果您使用需要驗證的登錄、請取代的預設行 `imagePullSecrets: []` 提供下列功能：

```
imagePullSecrets: [{name: astra-registry-cred}]
```

b. 變更 `[your_registry_path]` 適用於 `kube-rbac-proxy` 映像到您在中推入映像的登錄路徑 上一步。

c. 變更 `[your_registry_path]` 適用於 `acc-operator-controller-manager` 映像到您在中推入映像的登錄路徑 上一步。

```
<strong>astra_control_center_operator_deploy.yaml</strong>
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    control-plane: controller-manager
```

```
  name: acc-operator-controller-manager
  namespace: netapp-acc-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      control-plane: controller-manager
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: [your_registry_path]/kube-rbac-proxy:v4.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        - name: ACCOP_HELM_INSTALLTIMEOUT
          value: 5m
        image: [your_registry_path]/acc-operator:23.04.36
        imagePullPolicy: IfNotPresent
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8081
          initialDelaySeconds: 15
          periodSeconds: 20
        name: manager
        readinessProbe:
          httpGet:
            path: /readyz
```

```
            port: 8081
          initialDelaySeconds: 5
          periodSeconds: 10
        resources:
          limits:
            cpu: 300m
            memory: 750Mi
          requests:
            cpu: 100m
            memory: 75Mi
        securityContext:
          allowPrivilegeEscalation: false
      imagePullSecrets: []
      securityContext:
        runAsUser: 65532
      terminationGracePeriodSeconds: 10
```

3. 安裝Astra Control Center操作員：

```
kubectl apply -f astra_control_center_operator_deploy.yaml
```

回應範例：

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.astra.
netapp.io created
role.rbac.authorization.k8s.io/acc-operator-leader-election-role created
clusterrole.rbac.authorization.k8s.io/acc-operator-manager-role created
clusterrole.rbac.authorization.k8s.io/acc-operator-metrics-reader
created
clusterrole.rbac.authorization.k8s.io/acc-operator-proxy-role created
rolebinding.rbac.authorization.k8s.io/acc-operator-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-manager-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-proxy-
rolebinding created
configmap/acc-operator-manager-config created
service/acc-operator-controller-manager-metrics-service created
deployment.apps/acc-operator-controller-manager created
```

4. 確認Pod正在執行：

```
kubectl get pods -n netapp-acc-operator
```

設定Astra控制中心

1. 編輯Astra Control Center自訂資源（CR）檔案 (`astra_control_center.yaml`）進行帳戶、支援、登錄及其他必要設定：

```
vim astra_control_center.yaml
```

附註的Y反 洗錢範例遵循下列步驟。

2. 修改或確認下列設定：

<code>accountName</code>

設定	指導	類型	範例
`accountName`	變更 `accountName` 字串至您要與Astra Control Center帳戶建立關聯的名稱。只能有一個帳戶名稱。	字串	`Example`

<code>astraVersion</code>

設定	指導	類型	範例
`astraVersion`	要部署的Astra Control Center版本。此設定不需要任何動作、因為此值將預先填入。	字串	`23.04.2-7`

<code>astraAddress</code>

設定	指導	類型	範例
`astraAddress`	變更 `astraAddress` 字串至您要在瀏覽器中使用的FQDN（建議）或IP位址、以存取Astra Control Center。此位址定義Astra Control Center在資料中心的找到方式、以及當您完成配置時、從負載平衡器配置的相同FQDN或IP位址 "Astra Control Center需求"。 附註：請勿使用 `http://` 或 `https://` 地址中。複製此FQDN以供在中使用 後續步驟。	字串	`astra.example.com`

<code>autoSupport</code>

您在本節中的選擇決定您是否會參與NetApp主動式支援應用程式NetApp Active IQ 功能、以及資料的傳送位置。需要網際網路連線（連接埠4442）、所有支援資料都會匿名。

設定	使用	指導	類型	範例
`autoSupport.enrolled`	也可以 `enrolled` 或 `url` 必須選取欄位	變更 `enrolled` for 解決方案AutoSupport `false` 適用於沒有網際網路連線或無法保留的網站 `true` 適用於連線站台。的設定 `true` 可將匿名資料傳送至 NetApp 以供支援之用。預設選項為 `false` 並表示不會將任何支援資料傳送給NetApp。	布林值	`false` （此值為預設值）
`autoSupport.url`	也可以 `enrolled` 或 `url` 必須選取欄位	此URL決定匿名資料的傳送位置。	字串	`https://support.netapp.com/asupprod/post/1.0/postAsup`

<code>email</code>

設定	指導	類型	範例
`email`	變更 `email` 字串至預設的初始系統管理員位址。複製此電子郵件地址以供在中使用 後續步驟。此電子郵件地址將作為初始帳戶登入UI的使用者名稱、並會收到Astra Control中事件的通知。	字串	`admin@example.com`

<code>firstName</code>

設定	指導	類型	範例
`firstName`	與Astra帳戶相關聯的預設初始系統管理員的名字。第一次登入後、此處使用的名稱會顯示在UI的標題中。	字串	`SRE`

<code>LastName</code>

設定	指導	類型	範例
`lastName`	與Astra帳戶相關聯的預設初始管理員姓氏。第一次登入後、此處使用的名稱會顯示在UI的標題中。	字串	`Admin`

<code>imageRegistry</code>

您在本節中的選擇定義了裝載Astra應用程式映像、Astra Control Center運算子和Astra Control Center Helm儲存庫的容器映像登錄。

設定	使用	指導	類型	範例
`imageRegistry.name`	必要	您在中推入映像的映像登錄名稱 上一步。請勿使用 `http://` 或 `https://` 在登錄名稱中。	字串	`example.registry.com/astra`
`imageRegistry.secret`	如果您輸入的字串則為必要 `imageRegistry.name' requires a secret.` `IMPORTANT: If you are using a registry that does not require authorization, you must delete this` `secret` 行內 `imageRegistry` 否則安裝將會失敗。	用來驗證映像登錄的Kubernetes機密名稱。	字串	`astra-registry-cred`

<code>storageClass</code>

設定	指導	類型	範例
`storageClass`	變更 `storageClass` 價值來源 `ontap-gold` 至安裝所需的另一個 Astra Trident storageClass 資源。執行命令 `kubectl get sc` 以判斷您現有的已設定儲存類別。必須在資訊清單檔案中輸入其中一個 Astra Trident 型儲存類別 (`astra-control-center-<version>.manifest`）、並將用於Astra PV。如果未設定、則會使用預設的儲存類別。 附註：如果已設定預設儲存類別、請確定它是唯一具有預設附註的儲存類別。	字串	`ontap-gold`

<code>volumeReclaimPolicy</code>

設定	指導	類型	選項
`volumeReclaimPolicy`	這為Astra的PV設定回收原則。將此原則設定為 `Retain` 刪除Astra後保留持續磁碟區。將此原則設定為 `Delete` 刪除Astra後刪除持續磁碟區。如果未設定此值、則會保留PV。	字串	• `Retain` （這是預設值） • `Delete`

<code>ingressType</code>

設定	指導	類型	選項
`ingressType`	使用下列其中一種入口類型： **`Generic`** (`ingressType: "Generic"`)（預設） 如果您使用另一個入口控制器、或偏好使用自己的入口控制器、請使用此選項。部署Astra Control Center之後、您需要設定 "入口控制器" 使用URL公開Astra Control Center。 **`AccTraefik`** (`ingressType: "AccTraefik"`) 如果您不想設定入口控制器、請使用此選項。這會部署Astra控制中心 `traefik` 作為Kubernetes負載平衡器類型服務的閘道。 Astra Control Center使用「負載平衡器」類型的服務 (`svc/traefik`（在Astra Control Center命名空間中）、並要求指派可存取的外部IP位址。如果您的環境允許負載平衡器、但您尚未設定負載平衡器、則可以使用MetalLB或其他外部服務負載平衡器、將外部IP位址指派給服務。在內部DNS伺服器組態中、您應該將Astra Control Center所選的DNS名稱指向負載平衡的IP位址。 附註：如需「負載平衡器」和入口服務類型的詳細資訊、請參閱 "需求"。	字串	• `Generic`（這是預設值） • `AccTraefik`

<code>scaleSize</code>

設定	指導	類型	選項
`scaleSize`	Astra 預設會使用高可用度（ HA ） `scaleSize` 的 `Medium` ，用於在 HA 中部署大多數服務並部署多個複本以實現冗餘。與 `scaleSize` 做為 `Small` 、 Astra 將減少所有服務的複本數量、但基本服務除外、以減少使用量。 秘訣： `Medium` 部署包含約 100 個 Pod （不包括暫時性工作負載）。100 個 Pod 以三個主節點和三個工作節點組態為基礎）。請注意、在您的環境中、每個 Pod 的網路限制可能是個問題、特別是在考慮災難恢復案例時。	字串	• `Small` • `Medium` （這是預設值）

<code>astraResourcesScaler</code>

設定	指導	類型	選項
`astraResourcesScaler`	適用的擴充選項適用於適用的適用範圍。依預設、Astra Control Center會針對Astra內的大部分元件設定資源要求來進行部署。此組態可讓Astra Control Center軟體堆疊在應用程式負載和擴充性增加的環境中、發揮更佳效能。 不過、在使用較小開發或測試叢集的案例中、則是使用「CR」欄位 `astraResourcesScalar` 可能設為 `Off` 。這會停用資源要求、並允許在較小的叢集上部署。	字串	• `Default` （這是預設值） • `Off`

<code>additionalValues</code>

- 對於 Astral Control Center 和 Cloud Insights 通訊、依預設會停用 TLS 憑證驗證。您可以在中新增下一節、以啟用 Cloud Insights 與 Astra 控制中心主機叢集和託管叢集之間通訊的 TLS 憑證驗證 `additionalValues`。

```
additionalValues:
  netapp-monitoring-operator:
    config:
      ciSkipTlsVerify: false
  cloud-insights-service:
    config:
      ciSkipTlsVerify: false
  telemetry-service:
    config:
      ciSkipTlsVerify: false
```

<code>crds</code>

您在本節中的選擇決定Astra Control Center應如何處理客戶需求日。

設定	指導	類型	範例
`crds.externalCertManager`	如果您使用外部憑證管理程式、請變更 `externalCertManager` 至 `true`。預設值 `false` 讓Astra Control Center在安裝期間安裝自己的憑證管理程式客戶檔案。 CRD是整個叢集的物件、安裝這些物件可能會影響叢集的其他部分。您可以使用此旗標向Astra控制中心發出訊號、表示這些客戶需求日將由Astra控制中心外部的叢集管理員安裝及管理。	布林值	`False` （此值為預設值）
`crds.externalTraefik`	依預設、Astra Control Center會安裝必要的Traefik客戶需求日。CRD是整個叢集的物件、安裝這些物件可能會影響叢集的其他部分。您可以使用此旗標向Astra控制中心發出訊號、表示這些客戶需求日將由Astra控制中心外部的叢集管理員安裝及管理。	布林值	`False` （此值為預設值）

在完成安裝之前、請務必為您的組態選擇正確的儲存類別和入口類型。

```
<strong>astra_control_center.yaml</strong>
```

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
  volumeReclaimPolicy: "Retain"
  ingressType: "Generic"
  scaleSize: "Medium"
  astraResourcesScaler: "Default"
  additionalValues: {}
  crds:
    externalTraefik: false
    externalCertManager: false
```

完整的**Astra**控制中心和操作員安裝

1. 如果您尚未在上一步中執行此動作、請建立 `netapp-acc` （或自訂）命名空間：

   ```
   kubectl create ns [netapp-acc or custom namespace]
   ```

   回應範例：

   ```
   namespace/netapp-acc created
   ```

2. 在中安裝Astra Control Center `netapp-acc` （或自訂）命名空間：

   ```
   kubectl apply -f astra_control_center.yaml -n [netapp-acc or custom
   namespace]
   ```

   回應範例：

```
astracontrolcenter.astra.netapp.io/astra created
```

Astra Control Center 駕駛員將自動檢查環境需求。遺失 "需求" 可能導致安裝失敗、或 Astra Control Center 無法正常運作。請參閱 下一節 檢查與自動系統檢查相關的警告訊息。

驗證系統狀態

您可以使用kubectl命令來驗證系統狀態。如果您偏好使用OpenShift、您可以使用相似的相關命令來進行驗證步驟。

步驟

1. 確認安裝程序未產生與驗證檢查相關的警告訊息：

```
kubectl get acc [astra or custom Astra Control Center CR name] -n
[netapp-acc or custom namespace] -o yaml
```

Astra Control Center 操作者記錄中也會報告其他警告訊息。

2. 修正自動化需求檢查所回報的環境問題。

您可以確保環境符合、以修正問題 "需求" 適用於 Astra Control Center 。

3. 驗證是否已成功安裝所有系統元件。

```
kubectl get pods -n [netapp-acc or custom namespace]
```

每個Pod的狀態應為 `Running`。部署系統Pod可能需要幾分鐘的時間。

回應範例

```
NAME                                          READY   STATUS
RESTARTS     AGE
acc-helm-repo-6cc7696d8f-pmhm8                1/1     Running     0
9h
activity-597fb656dc-5rd4l                     1/1     Running     0
9h
activity-597fb656dc-mqmcw                     1/1     Running     0
9h
api-token-authentication-62f84                1/1     Running     0
9h
api-token-authentication-68nlf                1/1     Running     0
9h
api-token-authentication-ztgrm                1/1     Running     0
9h
asup-669d4ddbc4-fnmwp                         1/1     Running     1
(9h ago)   9h
authentication-78789d7549-lk686               1/1     Running     0
9h
bucketservice-65c7d95496-24x7l                1/1     Running     3
(9h ago)   9h
cert-manager-c9f9fbf9f-k8zq2                  1/1     Running     0
9h
cert-manager-c9f9fbf9f-qjlzm                  1/1     Running     0
9h
cert-manager-cainjector-dbbbd8447-b5qll       1/1     Running     0
9h
cert-manager-cainjector-dbbbd8447-p5whs       1/1     Running     0
9h
cert-manager-webhook-6f97bb7d84-4722b         1/1     Running     0
9h
cert-manager-webhook-6f97bb7d84-86kv5         1/1     Running     0
9h
certificates-59d9f6f4bd-2j899                 1/1     Running     0
9h
certificates-59d9f6f4bd-9d9k6                 1/1     Running     0
9h
certificates-expiry-check-28011180--1-8lkxz   0/1     Completed   0
9h
cloud-extension-5c9c9958f8-jdhrp              1/1     Running     0
9h
cloud-insights-service-5cdd5f7f-pp8r5         1/1     Running     0
9h
composite-compute-66585789f4-hxn5w            1/1     Running     0
```

```
9h
composite-volume-68649f68fd-tb7p4             1/1     Running     0
9h
credentials-dfc844c57-jsx92                   1/1     Running     0
9h
credentials-dfc844c57-xw26s                   1/1     Running     0
9h
entitlement-7b47769b87-4jb6c                  1/1     Running     0
9h
features-854d8444cc-c24b7                     1/1     Running     0
9h
features-854d8444cc-dv6sm                     1/1     Running     0
9h
fluent-bit-ds-9tlv4                           1/1     Running     0
9h
fluent-bit-ds-bpkcb                           1/1     Running     0
9h
fluent-bit-ds-cxmwx                           1/1     Running     0
9h
fluent-bit-ds-jgnhc                           1/1     Running     0
9h
fluent-bit-ds-vtr6k                           1/1     Running     0
9h
fluent-bit-ds-vxqd5                           1/1     Running     0
9h
graphql-server-7d4b9d44d5-zdbf5               1/1     Running     0
9h
identity-6655c48769-4pwk8                     1/1     Running     0
9h
influxdb2-0                                   1/1     Running     0
9h
keycloak-operator-55479d6fc6-slvmt            1/1     Running     0
9h
krakend-f487cb465-78679                       1/1     Running     0
9h
krakend-f487cb465-rjsxx                       1/1     Running     0
9h
license-64cbc7cd9c-qxsr8                      1/1     Running     0
9h
login-ui-5db89b5589-ndb96                     1/1     Running     0
9h
loki-0                                        1/1     Running     0
9h
metrics-facade-8446f64c94-x8h7b               1/1     Running     0
9h
monitoring-operator-6b44586965-pvcl4          2/2     Running     0
```

```
9h
nats-0                                            1/1     Running     0
9h
nats-1                                            1/1     Running     0
9h
nats-2                                            1/1     Running     0
9h
nautilus-85754d87d7-756qb                         1/1     Running     0
9h
nautilus-85754d87d7-q8j7d                         1/1     Running     0
9h
openapi-5f9cc76544-7fnjm                          1/1     Running     0
9h
openapi-5f9cc76544-vzr7b                          1/1     Running     0
9h
packages-5db49f8b5-lrzhd                          1/1     Running     0
9h
polaris-consul-consul-server-0                    1/1     Running     0
9h
polaris-consul-consul-server-1                    1/1     Running     0
9h
polaris-consul-consul-server-2                    1/1     Running     0
9h
polaris-keycloak-0                                1/1     Running     2
(9h ago)   9h
polaris-keycloak-1                                1/1     Running     0
9h
polaris-keycloak-2                                1/1     Running     0
9h
polaris-keycloak-db-0                             1/1     Running     0
9h
polaris-keycloak-db-1                             1/1     Running     0
9h
polaris-keycloak-db-2                             1/1     Running     0
9h
polaris-mongodb-0                                 1/1     Running     0
9h
polaris-mongodb-1                                 1/1     Running     0
9h
polaris-mongodb-2                                 1/1     Running     0
9h
polaris-ui-66fb99479-qp9gq                        1/1     Running     0
9h
polaris-vault-0                                   1/1     Running     0
9h
polaris-vault-1                                   1/1     Running     0
```

```
9h
polaris-vault-2                                1/1     Running     0
9h
public-metrics-76fbf9594d-zmxzw                1/1     Running     0
9h
storage-backend-metrics-7d7fbc9cb9-lmd25       1/1     Running     0
9h
storage-provider-5bdd456c4b-2fftc              1/1     Running     0
9h
task-service-87575df85-dnn2q                   1/1     Running     3
(9h ago)   9h
task-service-task-purge-28011720--1-q6w4r      0/1     Completed   0
28m
task-service-task-purge-28011735--1-vk6pd      1/1     Running     0
13m
telegraf-ds-2r2kw                              1/1     Running     0
9h
telegraf-ds-6s9d5                              1/1     Running     0
9h
telegraf-ds-96jl7                              1/1     Running     0
9h
telegraf-ds-hbp84                              1/1     Running     0
9h
telegraf-ds-plwzv                              1/1     Running     0
9h
telegraf-ds-sr22c                              1/1     Running     0
9h
telegraf-rs-4sbg8                              1/1     Running     0
9h
telemetry-service-fb9559f7b-mk9l7              1/1     Running     3
(9h ago)   9h
tenancy-559bbc6b48-5msgg                       1/1     Running     0
9h
traefik-d997b8877-7xpf4                        1/1     Running     0
9h
traefik-d997b8877-9xv96                        1/1     Running     0
9h
trident-svc-585c97548c-d25z5                   1/1     Running     0
9h
vault-controller-88484b454-2d6sr               1/1     Running     0
9h
vault-controller-88484b454-fc5cz               1/1     Running     0
9h
vault-controller-88484b454-jktld               1/1     Running     0
9h
```

4. （選用）若要確保安裝完成、您可以觀看 `acc-operator` 使用下列命令記錄。

```
kubectl logs deploy/acc-operator-controller-manager -n netapp-acc-
operator -c manager -f
```

> `accHost` 叢集登錄是最後一項作業、如果失敗、也不會導致部署失敗。如果記錄中指出叢集登錄失敗、您可以透過再次嘗試登錄 "在UI中新增叢集工作流程" 或API。

5. 當所有Pod都在執行時、請確認安裝成功 (`READY` 是 `True`）並取得您登入Astra Control Center時所使用的初始設定密碼：

```
kubectl get AstraControlCenter -n [netapp-acc or custom namespace]
```

回應：

```
NAME    UUID                                   VERSION     ADDRESS
READY
astra   9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f  23.04.2-7   10.111.111.111
True
```

> 複製UUID值。密碼是 `ACC-` 接著是UUID值 (`ACC-[UUID]` 或者、在此範例中、 `ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f`）。

設定入口以進行負載平衡

您可以設定Kubernetes入口控制器來管理外部服務存取。如果您使用的預設值、這些程序會提供入口控制器的設定範例 `ingressType: "Generic"` Astra Control Center自訂資源 (`astra_control_center.yaml`）。如果您指定、則不需要使用此程序 `ingressType: "AccTraefik"` Astra Control Center自訂資源 (`astra_control_center.yaml`）。

部署Astra Control Center之後、您需要設定入口控制器、以URL顯示Astra Control Center。

設定步驟視您使用的入口控制器類型而有所不同。Astra Control Center支援多種入站控制器類型。這些設定程序提供下列入口控制器類型的範例步驟：

- Istio入口
- Nginx入口控制器
- OpenShift入口控制器

開始之前

- 必要的 "入口控制器" 應已部署。
- 。 "入口等級" 應已建立對應於入口控制器的。

Istio入侵步驟

1. 設定Istio入口。

   此程序假設使用「預設」組態設定檔來部署Istio。

2. 收集或建立Ingress閘道所需的憑證和私密金鑰檔案。

   您可以使用CA簽署或自我簽署的憑證。一般名稱必須是Astra位址（FQDN）。

   命令範例：

   ```
   openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out
   tls.crt
   ```

3. 建立秘密 `tls secret name` 類型 `kubernetes.io/tls` 中的TLS私密金鑰和憑證 `istio-system namespace` 如TLS機密所述。

   命令範例：

   ```
   kubectl create secret tls [tls secret name] --key="tls.key"
   --cert="tls.crt" -n istio-system
   ```

   機密名稱應與相符 `spec.tls.secretName` 提供於 `istio-ingress.yaml` 檔案：

4. 在中部署入口資源 `netapp-acc` （或自訂命名）命名空間、使用v1資源類型作為架構 (`istio-Ingress.yaml` 在本例中使用）：

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: istio
spec:
  controller: istio.io/ingress-controller
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: traefik
            port:
              number: 80
```

5. 套用變更：

```
kubectl apply -f istio-Ingress.yaml
```

6. 檢查入侵狀態：

```
kubectl get ingress -n [netapp-acc or custom namespace]
```

回應：

```
NAME    CLASS HOSTS              ADDRESS         PORTS   AGE
ingress istio astra.example.com 172.16.103.248  80, 443 1h
```

7. 完成Astra Control Center安裝。

適用於**Nginvin**像 控制器的步驟

1. 建立類型的秘密 `kubernetes.io/tls` 中的TLS私密金鑰和憑證 `netapp-acc` （或自訂命名）命名空間、如所述 "TLS機密"。
2. 在中部署入口資源 `netapp-acc` （或自訂命名）命名空間、使用v1資源類型作為架構 (`nginx-Ingress.yaml` 在本例中使用）：

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netapp-acc-ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: [class name for nginx controller]
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: <ACC address>
    http:
      paths:
        - path:
          backend:
            service:
              name: traefik
              port:
                number: 80
          pathType: ImplementationSpecific
```

3. 套用變更：

```
kubectl apply -f nginx-Ingress.yaml
```

NetApp建議將Ngin像 控制器安裝為部署、而非 `daemonSet`。

OpenShift入口控制器的步驟

1. 取得您的憑證、取得可供OpenShift路由使用的金鑰、憑證和CA檔案。
2. 建立OpenShift路由：

```
oc create route edge --service=traefik --port=web -n [netapp-acc or
custom namespace] --insecure-policy=Redirect --hostname=<ACC address>
--cert=cert.pem --key=key.pem
```

登入Astra Control Center UI

安裝Astra Control Center之後、您將變更預設管理員的密碼、並登入Astra Control Center UI儀表板。

步驟

1. 在瀏覽器中、輸入 FQDN （包括 `https://` 字首） `astraAddress` 在中 `astra_control_center.yaml` 請於何時進行 您安裝了Astra Control Center。
2. 收到提示時、請接受自我簽署的憑證。

   您可以在登入後建立自訂憑證。

3. 在Astra Control Center登入頁面、輸入您使用的值 `email` 在中 `astra_control_center.yaml` 請於何時進行 您安裝了Astra Control Center，然後輸入初始設定密碼 (`ACC-[UUID]`) 。

   如果您輸入錯誤密碼三次、系統將鎖定管理員帳戶15分鐘。

4. 選擇*登入*。
5. 出現提示時變更密碼。

   如果這是您第一次登入、但您忘記密碼、而且尚未建立其他管理使用者帳戶、請聯絡 "NetApp支援" 以取得密碼恢復協助。

6. （選用）移除現有的自我簽署TLS憑證、並以取代 "由憑證授權單位（CA）簽署的自訂TLS憑證"。

疑難排解安裝

如果有任何服務存在 `Error` 狀態、您可以檢查記錄。尋找400到500範圍內的API回應代碼。這些都表示發生故障的地點。

選項

- 若要檢查Astra控制中心的操作員記錄、請輸入下列內容：

```
kubectl logs deploy/acc-operator-controller-manager -n netapp-acc-
operator -c manager -f
```

- 若要檢查 Astra Control Center CR 的輸出：

```
kubectl get acc -n [netapp-acc or custom namespace] -o yaml
```

下一步

- （選用）視您的環境而定、請在安裝後完成 "組態步驟"。
- 執行以完成部署 "設定工作"。

設定外部憑證管理程式

如果Kubernetes叢集中已存在憑證管理程式、您需要執行一些必要步驟、使Astra Control Center不會安裝自己的憑證管理程式。

步驟

1. 確認您已安裝憑證管理程式：

```
kubectl get pods -A | grep 'cert-manager'
```

回應範例：

```
cert-manager   essential-cert-manager-84446f49d5-sf2zd              1/1
Running    0      6d5h
cert-manager   essential-cert-manager-cainjector-66dc99cc56-9ldmt   1/1
Running    0      6d5h
cert-manager   essential-cert-manager-webhook-56b76db9cc-fjqrq      1/1
Running    0      6d5h
```

2. 為建立憑證/金鑰配對 `astraAddress` FQDN：

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out
tls.crt
```

回應範例：

```
Generating a 2048 bit RSA private key
...................+++
.......................................+++
writing new private key to 'tls.key'
```

3. 使用先前產生的檔案建立秘密：

```
kubectl create secret tls selfsigned-tls --key tls.key --cert tls.crt -n
<cert-manager-namespace>
```

回應範例：

```
secret/selfsigned-tls created
```

4. 建立 `ClusterIssuer` 以下*確切*的檔案包含您的命名空間位置 `cert-manager` 已安裝Pod：

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: astra-ca-clusterissuer
  namespace: <cert-manager-namespace>
spec:
  ca:
    secretName: selfsigned-tls
```

```
kubectl apply -f ClusterIssuer.yaml
```

回應範例：

```
clusterissuer.cert-manager.io/astra-ca-clusterissuer created
```

5. 確認 `ClusterIssuer` 已正確啟動。 `Ready` 必須是 `True` 在繼續之前：

```
kubectl get ClusterIssuer
```

回應範例：

```
NAME                      READY   AGE
astra-ca-clusterissuer    True    9s
```

6. 完成 "Astra Control Center安裝程序"。有 "Astra Control Center叢集Yaml所需的組態步驟" 您可在其中變更CRD值、以表示外部安裝了憑證管理程式。您必須在安裝期間完成此步驟、Astra Control Center才能辨識外部憑證管理程式。

使用OpenShift作業系統集線器安裝Astra Control Center

如果您使用Red Hat OpenShift、可以使用Red Hat認證的操作員來安裝Astra Control Center。請使用此程序從安裝Astra Control Center "Red Hat生態系統目錄" 或使用Red Hat OpenShift Container Platform。

完成此程序之後、您必須返回安裝程序、才能完成 "剩餘步驟" 以驗證安裝是否成功並登入。

開始之前

- 符合環境先決條件： "開始安裝之前、請先準備好環境以進行Astra Control Center部署"。
- 健全的叢集運算子與**API**服務：
  - 從OpenShift叢集確保所有叢集操作員都處於健全狀態：

```
oc get clusteroperators
```

  - 從OpenShift叢集、確保所有API服務都處於健全狀態：

```
oc get apiservices
```

- * FQDN位址*：取得資料中心Astra Control Center的FQDN位址。
- * OpenShift權限*：取得必要的權限並存取Red Hat OpenShift Container Platform、以執行所述的安裝步驟。
- 已設定的憑證管理程式：如果叢集中已存在憑證管理程式、您需要執行某些作業 "必要步驟" 因此Astra Control Center不會安裝自己的憑證管理程式。依預設、Astra Control Center會在安裝期間安裝自己的憑證管理程式。
- * Kubernetes入口控制器*：如果您有一個Kubernetes入口控制器來管理外部服務存取、例如叢集中的負載平衡、您就需要將其設定為與Astra Control Center搭配使用：
  a. 建立運算子命名空間：

```
oc create namespace netapp-acc-operator
```

  b. "完成設定" 適用於您的入口控制器類型。

步驟

- 下載並擷取Astra Control Center
- 安裝NetApp Astra kubecl外掛程式
- [將映像新增至本機登錄]
- [尋找操作員安裝頁面]
- [安裝操作員]
- 安裝Astra Control Center

下載並擷取**Astra Control Center**

1. 前往 "Astra Control Center 下載頁面" 於 NetApp 支援網站。
2. 下載包含Astra Control Center的套裝組合 (`astra-control-center-[version].tar.gz`) 。
3. （建議但可選）下載Astra Control Center的憑證與簽名套件 (`astra-control-center-certs-`

[version].tar.gz）若要驗證套件的簽名：

```
tar -vxzf astra-control-center-certs-[version].tar.gz
```

```
openssl dgst -sha256 -verify certs/AstraControlCenter-public.pub
-signature certs/astra-control-center-[version].tar.gz.sig astra-
control-center-[version].tar.gz
```

隨即顯示輸出 `Verified OK` 驗證成功之後。

4. 從Astra Control Center套裝組合擷取映像：

```
tar -vxzf astra-control-center-[version].tar.gz
```

安裝NetApp Astra kubecl外掛程式

您可以使用 NetApp Astra kubectl 命令列外掛程式、將影像推送至本機 Docker 儲存庫。

開始之前

NetApp為不同的CPU架構和作業系統提供外掛程式二進位檔。執行此工作之前、您必須先瞭解您的CPU和作業系統。

步驟

1. 列出可用的NetApp Astra kubectl外掛程式二進位檔、並記下作業系統和CPU架構所需的檔案名稱：

   KECBECTI外掛程式庫是tar套件的一部分、會擷取到資料夾中 `kubectl-astra`。

```
ls kubectl-astra/
```

2. 將正確的二進位檔移至目前路徑、並將其重新命名為 `kubectl-astra`：

```
cp kubectl-astra/<binary-name> /usr/local/bin/kubectl-astra
```

將映像新增至本機登錄

1. 為您的Container引擎完成適當的步驟順序：

Docker

1. 切換到tar檔案的根目錄。您應該會看到這個檔案和目錄：

   ```
   acc.manifest.bundle.yaml
   acc/
   ```

2. 將Astra Control Center映像目錄中的套件映像推送到本機登錄。執行之前、請先進行下列替換 `push-images` 命令：

   - 以<BUNDLE_FILE> Astra Control套裝組合檔案的名稱取代 (`acc.manifest.bundle.yaml`) 。
   - 以<MY_FULL_REGISTRY_PATH> Docker儲存庫的URL取代支援；例如 "<a href="https://<docker-registry>"" class="bare">https://<docker-registry>"。
   - 以<MY_REGISTRY_USER> 使用者名稱取代。
   - 以<MY_REGISTRY_TOKEN> 登錄的授權權杖取代。

     ```
     kubectl astra packages push-images -m <BUNDLE_FILE> -r
     <MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
     <MY_REGISTRY_TOKEN>
     ```

Podman

1. 切換到tar檔案的根目錄。您應該會看到這個檔案和目錄：

   ```
   acc.manifest.bundle.yaml
   acc/
   ```

2. 登入您的登錄：

   ```
   podman login <YOUR_REGISTRY>
   ```

3. 針對您使用的Podman版本、準備並執行下列其中一個自訂指令碼。以包含任何子目錄的儲存庫URL取代<MY_FULL_REGISTRY_PATH> 。

   ```
   <strong>Podman 4</strong>
   ```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.04.2-7
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/${
PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.04.2-7
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/${
PACKAGEVERSION}/${astraImageNoPath}
done
```

指令碼所建立的映像路徑應如下所示、視登錄組態而定：

```
https://netappdownloads.jfrog.io/docker-astra-control-
prod/netapp/astra/acc/23.04.2-7/image:version
```

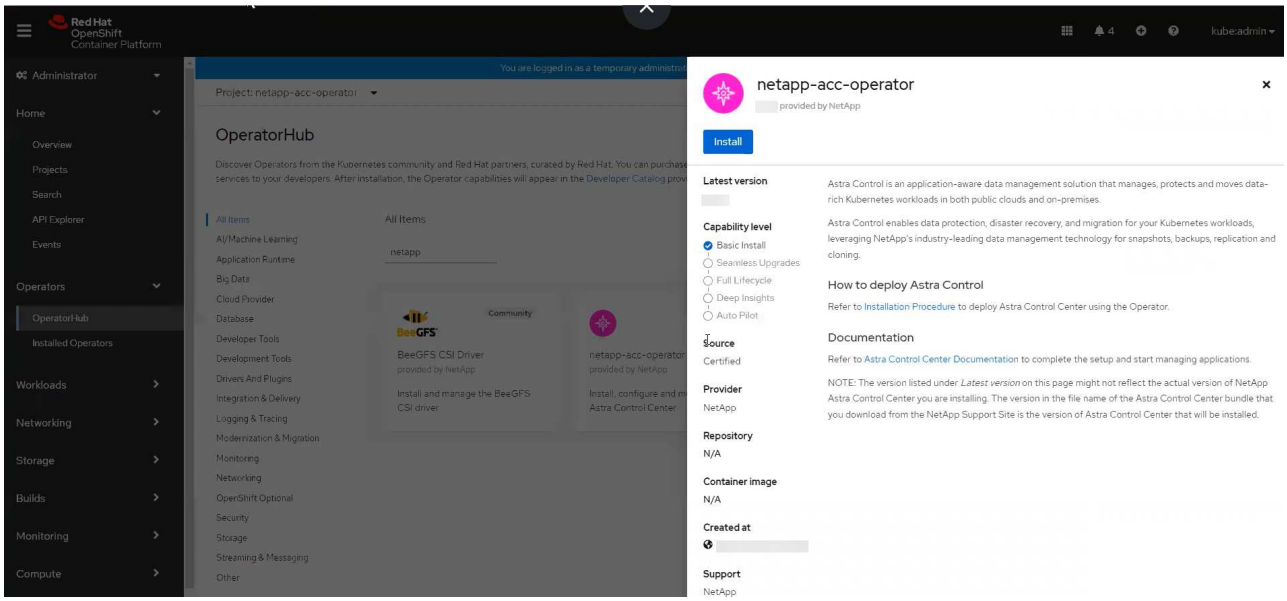
尋找操作員安裝頁面

1. 請完成下列其中一個程序、以存取操作員安裝頁面：
   - 從Red Hat Openshift Web主控台：

i. 登入OpenShift Container Platform UI。

ii. 從側功能表中、選取*運算子>運算子中樞*。

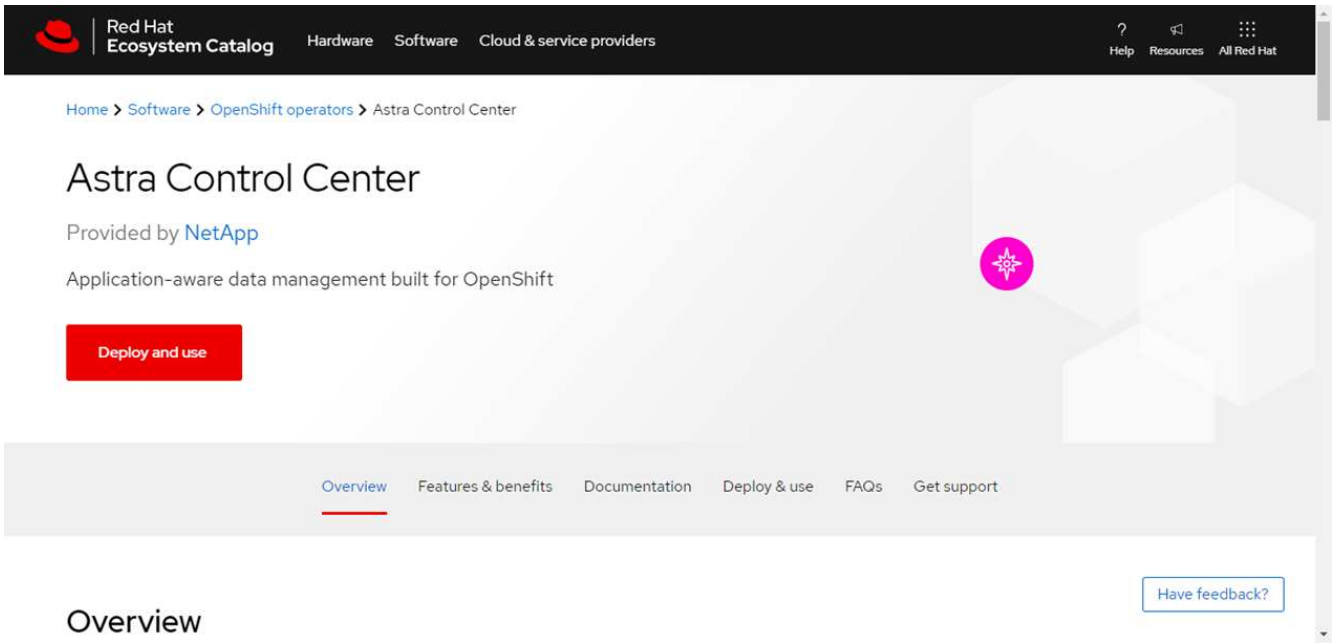
iii. 搜尋並選擇NetApp Astra Control Center營運者。



◦ 從Red Hat生態系統目錄：

i. 選擇NetApp Astra Control Center "營運者"。

ii. 選擇*部署和使用*。



安裝操作員

1. 完成*安裝操作員*頁面並安裝操作員：

   此運算子可用於所有叢集命名空間。

a. 選取運算子命名空間或 `netapp-acc-operator` 命名空間將會自動建立、做為操作員安裝的一部分。

b. 選取手動或自動核准策略。

> 建議手動核准。每個叢集只能執行單一運算子執行個體。

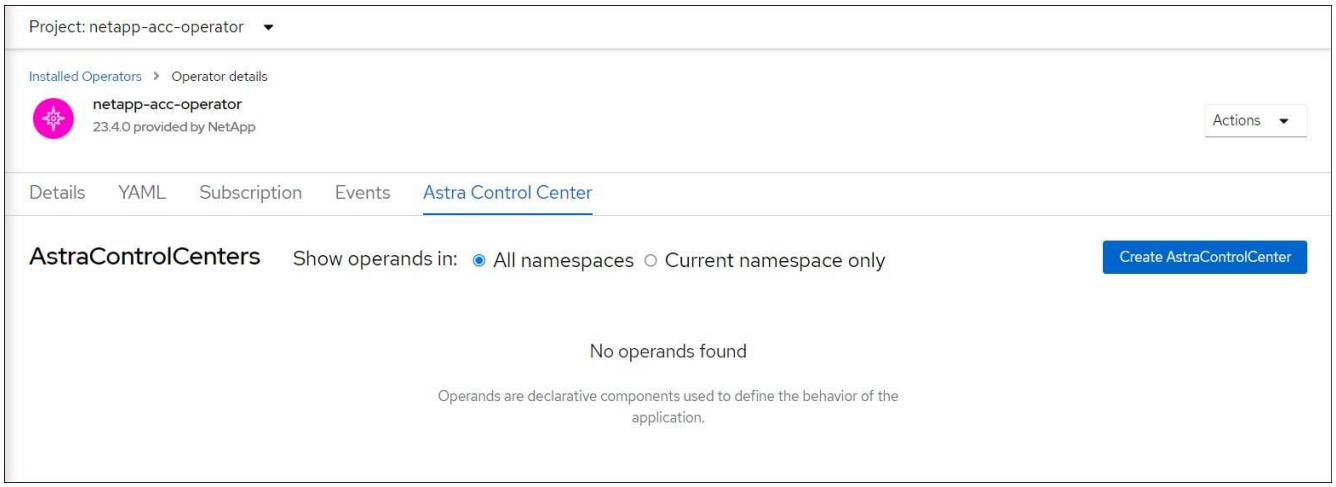
c. 選擇*安裝*。

> 如果您選擇手動核准策略、系統會提示您核准此操作員的手動安裝計畫。

2. 從主控台移至「作業系統集線器」功能表、確認操作員已成功安裝。

安裝Astra Control Center

1. 從Astra控制中心操作員* Astra控制中心*索引標籤內的主控台、選取*建立適用的*。



2. 完成 `Create AstraControlCenter` 表單欄位：

a. 保留或調整Astra Control Center名稱。

b. 新增Astra Control Center的標籤。

c. 啟用或停用自動支援。建議保留「自動支援」功能。

d. 輸入Astra Control Center FQDN或IP位址。請勿進入 `http://` 或 `https://` 在「地址」欄位中。

e. 輸入 Astra Control Center 版本、例如 23.04.2-7 。

f. 輸入帳戶名稱、電子郵件地址和管理員姓氏。

g. 選擇的Volume回收原則 `Retain`、 `Recycle`或 `Delete`。預設值為 `Retain`。

h. 選取安裝的 scaleSize 。

> Astra 預設會使用高可用度（ HA） `scaleSize` 的 `Medium`，用於在 HA 中部署大多數服務並部署多個複本以實現冗餘。與 `scaleSize` 做為 `Small`、 Astra 將減少所有服務的複本數量、但基本服務除外、以減少使用量。

i. 選取入口類型：

- **`Generic`**(`ingressType: "Generic"`) （預設）

如果您使用另一個入口控制器、或偏好使用自己的入口控制器、請使用此選項。部署Astra Control Center之後、您需要設定 "入口控制器" 使用URL公開Astra Control Center。

- **AccTraefik** (`ingressType: "AccTraefik"`)

  如果您不想設定入口控制器、請使用此選項。這會部署Astra控制中心 `traefik` 閘道即Kubernetes「負載平衡器」類型服務。

Astra Control Center使用「負載平衡器」類型的服務 (`svc/traefik` （在Astra Control Center命名空間中）、並要求指派可存取的外部IP位址。如果您的環境允許負載平衡器、但您尚未設定負載平衡器、則可以使用MetalLB或其他外部服務負載平衡器、將外部IP位址指派給服務。在內部DNS伺服器組態中、您應該將Astra Control Center所選的DNS名稱指向負載平衡的IP位址。

如需「負載平衡器」和入口服務類型的詳細資訊、請參閱 "需求"。

a. 在*映像登錄*中、輸入您的本機容器映像登錄路徑。請勿進入 `http://` 或 `https://` 在「地址」欄位中。
b. 如果您使用需要驗證的映像登錄、請輸入映像秘密。

   如果您使用需要驗證的登錄、 在叢集上建立秘密。

c. 輸入管理員名字。
d. 設定資源擴充。
e. 提供預設的儲存類別。

   如果已設定預設儲存類別、請確定它是唯一具有預設註釋的儲存類別。

f. 定義客戶需求日處理偏好設定。

3. 選取「Yaml」檢視以檢閱您所選的設定。
4. 選取 `Create`。

建立登錄機密

如果您使用需要驗證的登錄、請在Openshift叢集上建立密碼、然後在中輸入密碼名稱 `Create AstraControlCenter` 表單欄位。

1. 為Astra Control Center運算子建立命名空間：

```
oc create ns [netapp-acc-operator or custom namespace]
```

2. 在此命名空間中建立秘密：

```
oc create secret docker-registry astra-registry-cred n [netapp-acc-
operator or custom namespace] --docker-server=[your_registry_path]
--docker username=[username] --docker-password=[token]
```

Astra Control僅支援Docker登錄機密。

3. 填寫中的其餘欄位 「Create」（建立）「吧！Control Center」表單欄位。

下一步

完成 "剩餘步驟" 若要驗證Astra Control Center是否安裝成功、請設定入口控制器（選用）、然後登入UI。此外、您還需要執行 "設定工作" 安裝完成後。

安裝Astra Control Center搭配Cloud Volumes ONTAP 一套功能性儲存後端

有了Astra Control Center、您就能在混合雲環境中使用自我管理的Kubernetes叢集和Cloud Volumes ONTAP 實例來管理應用程式。您可以在內部部署的Kubernetes叢集或雲端環境中的其中一個自我管理Kubernetes叢集上部署Astra Control Center。

有了其中一項部署、您就能使用Cloud Volumes ONTAP 下列其中一項部署、以下列方式執行應用程式資料管理作業：將NetApp當成儲存後端。您也可以將S3儲存區設定為備份目標。

若要在Amazon Web Services（AWS）、Google Cloud Platform（GCP）和Microsoft Azure中安裝Astra Control Center、並搭配Cloud Volumes ONTAP 使用整套儲存後端、請視您的雲端環境而定、執行下列步驟。

- 在Amazon Web Services中部署Astra Control Center
- 在Google Cloud Platform中部署Astra Control Center
- 在Microsoft Azure中部署Astra Control Center

您可以使用自我管理的Kubernetes叢集、例如OpenShift Container Platform（OCP）、在發佈版本中管理應用程式。只有自我管理的OCP叢集已通過驗證、可用於部署Astra Control Center。

在Amazon Web Services中部署Astra Control Center

您可以在Amazon Web Services（AWS）公有雲上的自我管理Kubernetes叢集上部署Astra Control Center。

AWS所需的功能

在AWS中部署Astra Control Center之前、您需要下列項目：

- Astra Control Center授權。請參閱 "Astra Control Center授權要求"。
- "符合Astra Control Center的要求"。
- NetApp Cloud Central帳戶
- 如果使用OCP、則Red Hat OpenShift Container Platform（OCP）權限（位於命名空間層級以建立Pod）

- AWS認證資料、存取ID和秘密金鑰、具備可讓您建立儲存區和連接器的權限
- AWS帳戶彈性容器登錄（ECR）存取與登入
- 存取Astra Control UI所需的AWS託管區域和Route 53項目

AWS的作業環境需求

Astra Control Center需要下列AWS作業環境：

- Red Hat OpenShift Container Platform 4.8.

確保您選擇裝載Astra Control Center的作業環境符合環境正式文件中所述的基本資源需求。

除了環境的資源需求之外、Astra Control Center還需要下列資源：

元件	需求
後端**NetApp Cloud Volumes ONTAP** 功能儲存容量	至少提供300 GB
工作者節點（**AWS EC2**需求）	總共至少3個工作節點、每個節點有4個vCPU核心和12GB RAM
負載平衡器	服務類型「負載平衡器」可用於將入口流量傳送至作業環境叢集中的服務
FQDN	將Astra Control Center的FQDN指向負載平衡IP位址的方法
Astra Trident（安裝於**NetApp BlueXP**（前身為**Cloud Manager**）的**Kubernetes**叢集探索中）	Astra Trident 21.004或更新版本已安裝並設定、且NetApp ONTAP 的版本9.5或更新為儲存後端
映像登錄	您必須擁有現有的私有登錄、例如AWS Elastic Container登錄、才能將Astra Control Center建置映像推入其中。您需要提供映像登錄的URL、以便上傳映像。 Astra Control Center託管叢集和託管叢集必須能夠存取相同的映像登錄、才能使用還原型映像來備份和還原應用程式。

元件	需求
Astra Trident / ONTAP Estra組態	Astra Control Center需要建立儲存類別、並將其設為預設儲存類別。Astra Control Center支援下列ONTAP 將Kubernetes叢集匯入NetApp BlueXP（前身為Cloud Manager）時所建立的支援功能。這些資料由Astra Trident提供： • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

這些需求假設Astra Control Center是營運環境中唯一執行的應用程式。如果環境正在執行其他應用程式、請相應調整這些最低需求。

AWS登錄權杖會在12小時內過期、之後您必須更新Docker映像登錄機密。

AWS部署總覽

以下是安裝Astra Control Center for AWS的程序總覽、Cloud Volumes ONTAP 其中包含以作為儲存後端的功能。

以下將詳細說明每個步驟。

1. 確保您擁有足夠的IAM權限。
2. 在AWS上安裝RedHat OpenShift叢集。
3. 設定 AWS。
4. 設定適用於AWS的NetApp BlueXP。
5. 安裝AWS的Astra Control Center。

確保您擁有足夠的**IAM**權限

確保您擁有足夠的IAM角色和權限、可讓您安裝RedHat OpenShift叢集和NetApp BlueXP（前身為Cloud Manager）Connector。

請參閱 "初始 AWS 認證資料"。

在**AWS**上安裝**RedHat OpenShift**叢集

在AWS上安裝RedHat OpenShift Container Platform叢集。

如需安裝指示、請參閱 "在OpenShift Container Platform的AWS上安裝叢集"。

設定 **AWS**

接下來、設定AWS以建立虛擬網路、設定EC2運算執行個體、建立AWS S3儲存區、建立彈性容器登錄（ECR）以裝載Astra Control Center映像、然後將映像推送至此登錄。

請遵循AWS文件完成下列步驟。請參閱 "AWS安裝文件"。

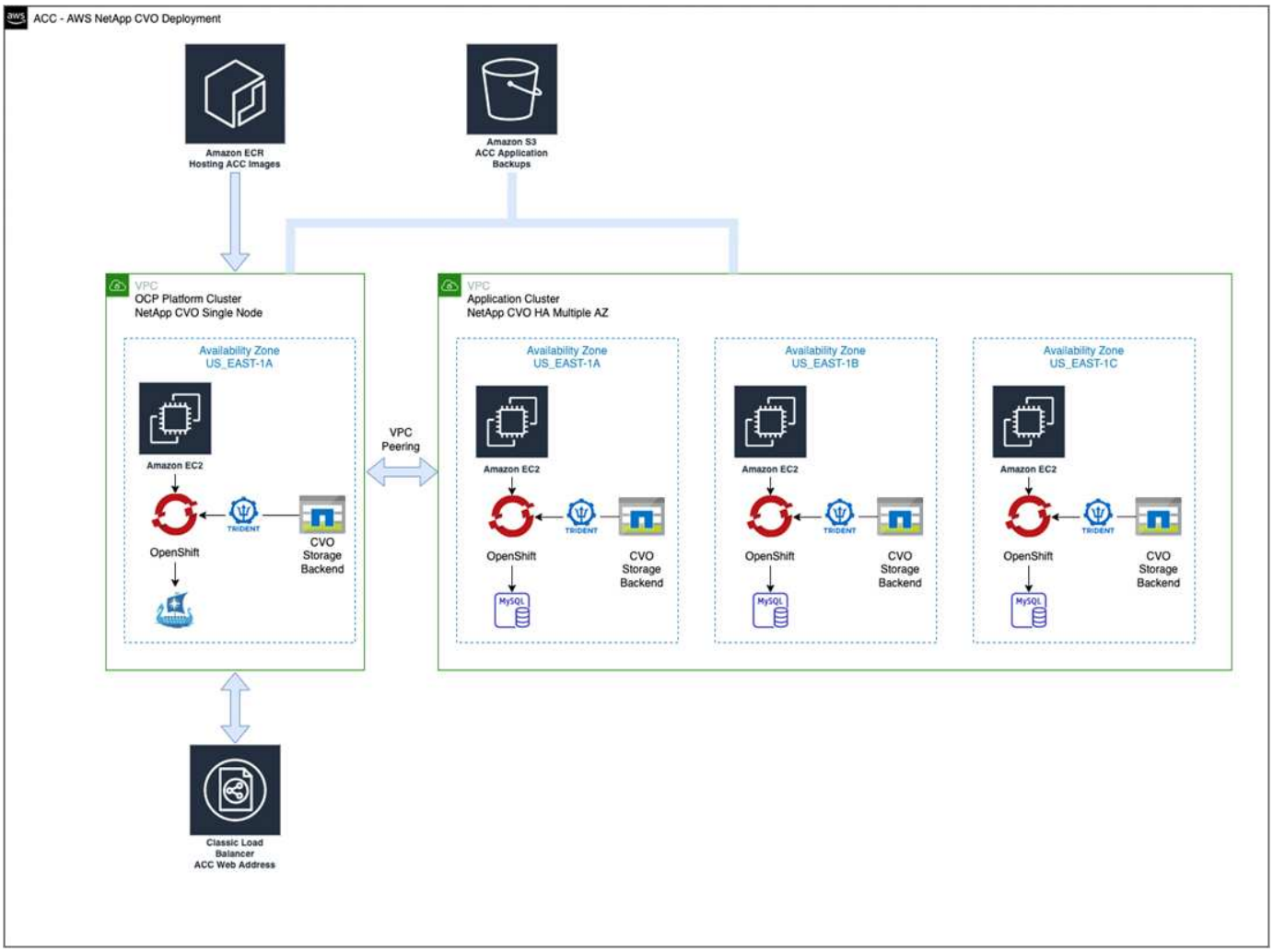
1. 建立AWS虛擬網路。
2. 檢閱EC2運算執行個體。這可以是AWS中的裸機伺服器或VM。
3. 如果執行個體類型尚未符合主節點和工作節點的Astra最低資源需求、請在AWS中變更執行個體類型以符合Astra需求。 請參閱 "Astra Control Center需求"。
4. 建立至少一個AWS S3儲存區來儲存備份。
5. 建立AWS彈性Container登錄（ECR）、以裝載所有的主動定速控制系統映像。

> 如果您未建立ECR、Astra Control Center將無法從含有Cloud Volumes ONTAP AWS後端的支援的叢集存取監控資料。此問題是因為您嘗試使用Astra Control Center探索及管理的叢集無法存取AWS ECR。

6. 將Acc映像推送到您定義的登錄。

> AWS Elastic Container登錄（ECR）權杖會在12小時後過期、導致跨叢集複製作業失敗。從Cloud Volumes ONTAP 針對AWS設定的功能進行的功能區管理儲存後端時、就會發生此問題。若要修正此問題、請再次向ECR驗證、並產生新的秘密、讓複製作業順利恢復。

以下是AWS部署範例：



設定適用於**AWS**的**NetApp BlueXP**

使用NetApp BlueXP（前身為Cloud Manager）建立工作區、新增AWS連接器、建立工作環境、以及匯入叢集。

請遵循BlueXP文件完成下列步驟。請參閱下列內容：

- "開始使用Cloud Volumes ONTAP AWS的功能"。
- "使用BlueXP在AWS中建立連接器"

步驟

1. 將您的認證資料新增至BlueXP。
2. 建立工作區。
3. 新增AWS的連接器。選擇AWS做為供應商。
4. 為您的雲端環境建立工作環境。
   a. 位置：「Amazon Web Services （AWS）」
   b. 類型：Cloud Volumes ONTAP 「EHA」
5. 匯入OpenShift叢集。叢集將連線至您剛建立的工作環境。
   a. 選擇* K8s*>*叢集清單*>*叢集詳細資料*、即可檢視NetApp叢集詳細資料。

b. 請注意右上角的 Astra Trident 版本。

c. 請注意Cloud Volumes ONTAP 、顯示NetApp為資源配置程式的叢集儲存類別。

這會匯入您的Red Hat OpenShift叢集、並將其指派為預設儲存類別。您可以選取儲存類別。Astra Trident 會在匯入和探索程序中自動安裝。

6. 請注意此Cloud Volumes ONTAP 功能部署中的所有持續磁碟區和磁碟區。

可作為單一節點或高可用度運作。Cloud Volumes ONTAP如果已啟用HA、請記下在AWS中執行的HA狀態和節點部署狀態。

安裝**AWS**的**Astra Control Center**

遵循標準 "Astra Control Center安裝說明"。

AWS使用一般S3儲存區類型。

在**Google Cloud Platform**中部署**Astra Control Center**

您可以在Google Cloud Platform（GCP）公有雲上的自我管理Kubernetes叢集上部署Astra Control Center。

GCP的必備功能

在 GCP 中部署 Astra Control Center 之前、您需要下列項目：

- Astra Control Center授權。請參閱 "Astra Control Center授權要求"。
- "符合Astra Control Center的要求"。
- NetApp Cloud Central帳戶
- 如果使用 OCP 、請選擇 Red Hat OpenShift Container Platform （ OCP ） 4.10
- 如果使用OCP、則Red Hat OpenShift Container Platform（OCP）權限（位於命名空間層級以建立Pod）
- GCP服務帳戶具備權限、可讓您建立貯體和連接器

GCP 的營運環境需求

確保您選擇裝載Astra Control Center的作業環境符合環境正式文件中所述的基本資源需求。

除了環境的資源需求之外、Astra Control Center還需要下列資源：

元件	需求
後端**NetApp Cloud Volumes ONTAP** 功能儲存容量	至少提供300 GB
工作者節點（**GCP**運算需求）	總共至少3個工作節點、每個節點有4個vCPU核心和12GB RAM
負載平衡器	服務類型「負載平衡器」可用於將入口流量傳送至作業環境叢集中的服務

元件	需求
FQDN（GCP DNS區域）	將Astra Control Center的FQDN指向負載平衡IP位址的方法
Astra Trident（安裝於NetApp BlueXP（前身為Cloud Manager）的Kubernetes叢集探索中）	Astra Trident 21.004或更新版本已安裝並設定、且NetApp ONTAP 的版本9.5或更新為儲存後端
映像登錄	您必須擁有現有的私有登錄、例如Google Container登錄、才能將Astra Control Center建置映像推送至該登錄。您需要提供映像登錄的URL、以便上傳映像。 您必須啟用匿名存取、才能拉出還原映像進行備份。
Astra Trident / ONTAP Estra組態	Astra Control Center需要建立儲存類別、並將其設為預設儲存類別。Astra Control Center支援下列ONTAP 將Kubernetes叢集匯入NetApp BlueXP時所建立的物件庫伯內特儲存類別。這些資料由Astra Trident提供： • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

這些需求假設Astra Control Center是營運環境中唯一執行的應用程式。如果環境正在執行其他應用程式、請相應調整這些最低需求。

GCP 部署總覽

以下是將Astra Control Center安裝在GCP的自我管理OCP叢集上的程序總覽、Cloud Volumes ONTAP 其中包含以作儲存後端的功能。

以下將詳細說明每個步驟。

1. 在 GCP 上安裝 RedHat OpenShift 叢集。
2. 建立GCP專案和虛擬私有雲端。
3. 確保您擁有足夠的IAM權限。
4. 設定 GCP。
5. 為 GCP 設定 NetApp BlueXP。
6. 安裝Astra Control Center for GCP。

在 GCP 上安裝 RedHat OpenShift 叢集

第一步是在GCP上安裝RedHat OpenShift叢集。

如需安裝指示、請參閱下列內容：

- "在GCP中安裝OpenShift叢集"
- "建立GCP服務帳戶"

建立GCP專案和虛擬私有雲端

建立至少一個GCP專案和虛擬私有雲端（VPC）。

OpenShift可能會建立自己的資源群組。此外、您也應該定義GCP VPC。請參閱OpenShift文件。

您可能想要建立平台叢集資源群組和目標應用程式OpenShift叢集資源群組。

確保您擁有足夠的IAM權限

確保您擁有足夠的IAM角色和權限、可讓您安裝RedHat OpenShift叢集和NetApp BlueXP（前身為Cloud Manager）Connector。

請參閱 "初始GCP認證與權限"。

設定 GCP

接下來、設定GCP以建立VPC、設定運算執行個體、建立Google Cloud Object Storage、建立Google Container Register以裝載Astra Control Center映像、然後將映像推送至此登錄。

請依照 GCP 文件完成下列步驟。請參閱在GCP中安裝OpenShift叢集。

1. 在您計畫用於具有CVO後端的OCP叢集的GCP中建立GCP專案和VPC。
2. 檢閱運算執行個體。這可以是 GCP 中的裸機伺服器或 VM 。
3. 如果執行個體類型尚未符合主要節點和工作節點的 Astra 最低資源需求、請在 GCP 中變更執行個體類型、以符合 Astra 需求。請參閱 "Astra Control Center需求"。
4. 建立至少一個GCP雲端儲存庫來儲存備份。
5. 建立儲存貯體存取所需的機密。
6. 建立Google Container登錄、以裝載所有Astra Control Center映像。
7. 設定所有Astra Control Center映像的Google Container登錄存取權、以供Docker推/拉。

   範例：輸入下列指令碼、即可將Acc映像推送至此登錄：

   ```
   gcloud auth activate-service-account <service account email address>
   --key-file=<GCP Service Account JSON file>
   ```

   此指令碼需要Astra Control Center資訊清單檔案和Google Image登錄位置。

範例：

```
manifestfile=astra-control-center-<version>.manifest
GCP_CR_REGISTRY=<target image repository>
ASTRA_REGISTRY=<source ACC image repository>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image
    docker push $GCP_CR_REGISTRY/$image
done < astra-control-center-22.04.41.manifest
```

8. 設定DNS區域。

為 GCP 設定 NetApp BlueXP

使用NetApp BlueXP（前身為Cloud Manager）建立工作區、將連接器新增至GCP、建立工作環境、以及匯入叢集。

請遵循BlueXP文件完成下列步驟。請參閱 "從GCP開始使用Cloud Volumes ONTAP"。

開始之前

- 以所需的IAM權限和角色存取GCP服務帳戶

步驟

1. 將您的認證資料新增至BlueXP。請參閱 "新增GCP帳戶"。
2. 新增 GCP 連接器。
   a. 選擇「GCP」作為供應商。
   b. 輸入GCP認證。請參閱 "從BlueXP在GCP中建立連接器"。
   c. 確認連接器正在執行、並切換至該連接器。
3. 為您的雲端環境建立工作環境。
   a. 地點：「 GCP 」
   b. 類型：Cloud Volumes ONTAP 「EHA」
4. 匯入OpenShift叢集。叢集將連線至您剛建立的工作環境。
   a. 選擇* K8s*>*叢集清單*>*叢集詳細資料*、即可檢視NetApp叢集詳細資料。
   b. 請注意右上角的Trident版本。
   c. 請注意Cloud Volumes ONTAP 、顯示「NetApp」為資源配置程式的叢集儲存類別。

      這會匯入您的Red Hat OpenShift叢集、並將其指派為預設儲存類別。您可以選取儲存類別。Astra Trident 會在匯入和探索程序中自動安裝。

5. 請注意此Cloud Volumes ONTAP 功能部署中的所有持續磁碟區和磁碟區。

可作為單一節點或高可用度（HA）運作。Cloud Volumes ONTAP如果 HA 已啟用、請注意 GCP 中執行的 HA 狀態和節點部署狀態。

安裝Astra Control Center for GCP

遵循標準 "Astra Control Center安裝說明"。

GCP使用通用S3儲存區類型。

1. 產生Docker祕密以擷取Astra Control Center安裝的映像：

```
kubectl create secret docker-registry <secret name> --docker
-server=<Registry location> --docker-username=_json_key --docker
-password="$(cat <GCP Service Account JSON file>)" --namespace=pcloud
```

在Microsoft Azure中部署Astra Control Center

您可以將Astra Control Center部署在Microsoft Azure公有雲上的自我管理Kubernetes叢集上。

Azure的必備功能

在Azure中部署Astra Control Center之前、您需要下列項目：

- Astra Control Center授權。請參閱 "Astra Control Center授權要求"。
- "符合Astra Control Center的要求"。
- NetApp Cloud Central帳戶
- 如果使用OCP、Red Hat OpenShift Container Platform（OCP）4.8
- 如果使用OCP、則Red Hat OpenShift Container Platform（OCP）權限（位於命名空間層級以建立Pod）
- Azure認證、具備可讓您建立儲存區和連接器的權限

Azure的營運環境需求

確保您選擇裝載Astra Control Center的作業環境符合環境正式文件中所述的基本資源需求。

除了環境的資源需求之外、Astra Control Center還需要下列資源：

請參閱 "Astra Control Center營運環境需求"。

元件	需求
後端**NetApp Cloud Volumes ONTAP** 功能儲存容量	至少提供300 GB

元件	需求
工作者節點（**Azure**運算需求）	總共至少3個工作節點、每個節點有4個vCPU核心和12GB RAM
負載平衡器	服務類型「負載平衡器」可用於將入口流量傳送至作業環境叢集中的服務
FQDN（**Azure DNS**區域）	將Astra Control Center的FQDN指向負載平衡IP位址的方法
Astra Trident（安裝於**NetApp BlueXP**的**Kubernetes**叢集探索中）	Astra Trident 21.004或更新版本已安裝並設定、NetApp ONTAP 版本9.5或更新版本將作為儲存後端使用
映像登錄	您必須擁有現有的私有登錄、例如Azure Container登錄（ACR）、才能將Astra Control Center建置映像推送至該登錄。您需要提供映像登錄的URL、以便上傳映像。 您必須啟用匿名存取、才能拉出還原映像進行備份。
Astra Trident / ONTAP Estra組態	Astra Control Center需要建立儲存類別、並將其設為預設儲存類別。Astra Control Center支援下列ONTAP 將Kubernetes叢集匯入NetApp BlueXP時所建立的物件庫伯內特儲存類別。這些資料由Astra Trident提供： • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

這些需求假設Astra Control Center是營運環境中唯一執行的應用程式。如果環境正在執行其他應用程式、請相應調整這些最低需求。

Azure部署總覽

以下是安裝Astra Control Center for Azure的程序總覽。

以下將詳細說明每個步驟。

1. 在Azure上安裝RedHat OpenShift叢集。
2. 建立Azure資源群組。
3. 確保您擁有足夠的IAM權限。
4. 設定Azure。

5. 設定適用於Azure的NetApp BlueXP（前身為Cloud Manager）。
6. 安裝及設定Azure的Astra Control Center。

在Azure上安裝RedHat OpenShift叢集

第一步是在Azure上安裝RedHat OpenShift叢集。

如需安裝指示、請參閱下列內容：

- "在Azure上安裝OpenShift叢集"。
- "安裝Azure帳戶"。

建立Azure資源群組

建立至少一個Azure資源群組。

> OpenShift可能會建立自己的資源群組。此外、您也應該定義Azure資源群組。 請參閱OpenShift文件。

您可能想要建立平台叢集資源群組和目標應用程式OpenShift叢集資源群組。

確保您擁有足夠的IAM權限

確保您擁有足夠的IAM角色和權限、可讓您安裝RedHat OpenShift叢集和NetApp BlueXP Connector。

請參閱 "Azure 認證與權限"。

設定Azure

接下來、設定Azure以建立虛擬網路、設定運算執行個體、建立Azure Blob容器、建立Azure Container Register（ACR）來裝載Astra Control Center映像、然後將映像推送至此登錄。

請依照Azure文件完成下列步驟。請參閱 "在Azure上安裝OpenShift叢集"。

1. 建立Azure虛擬網路。
2. 檢閱運算執行個體。這可以是Azure中的裸機伺服器或VM。
3. 如果執行個體類型尚未符合主節點和工作節點的Astra最低資源需求、請變更Azure中的執行個體類型以符合Astra要求。 請參閱 "Astra Control Center需求"。
4. 建立至少一個Azure Blob容器來儲存備份。
5. 建立儲存帳戶。您需要儲存帳戶來建立容器、以便在Astra Control Center中作為儲存庫。
6. 建立儲存貯體存取所需的機密。
7. 建立Azure Container登錄（ACR）、以裝載所有Astra Control Center映像。
8. 設定Docker推/拉所有Astra Control Center影像的ACR存取。
9. 輸入下列指令碼、將Acc映像推入此登錄：

```
az acr login -n <AZ ACR URL/Location>
This script requires ACC manifest file and your Azure ACR location.
```

範例：

```
manifestfile=astra-control-center-<version>.manifest
AZ_ACR_REGISTRY=<target image repository>
ASTRA_REGISTRY=<source ACC image repository>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRYY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < astra-control-center-22.04.41.manifest
```

10. 設定DNS區域。

設定適用於**Azure**的**NetApp BlueXP**（前身為**Cloud Manager**）

使用BlueXP（前身為Cloud Manager）建立工作區、將連接器新增至Azure、建立工作環境、以及匯入叢集。

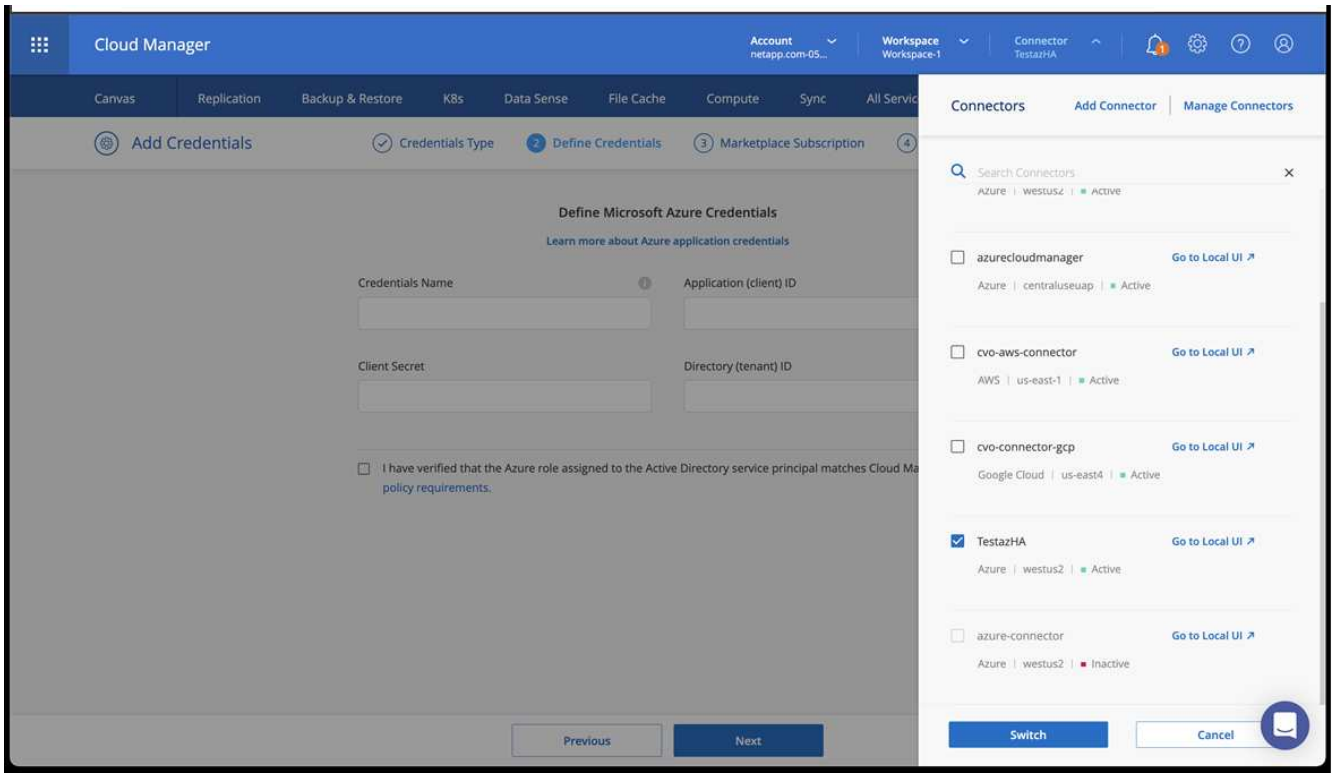
請遵循BlueXP文件完成下列步驟。請參閱 "Azure中的BlueXP入門指南"。

開始之前
以所需的IAM權限和角色存取Azure帳戶

步驟

1. 將您的認證資料新增至BlueXP。
2. 新增Azure連接器。請參閱 "BlueXP原則"。
   a. 選擇* Azure *作為供應商。
   b. 輸入Azure認證資料、包括應用程式ID、用戶端機密和目錄（租戶）ID。

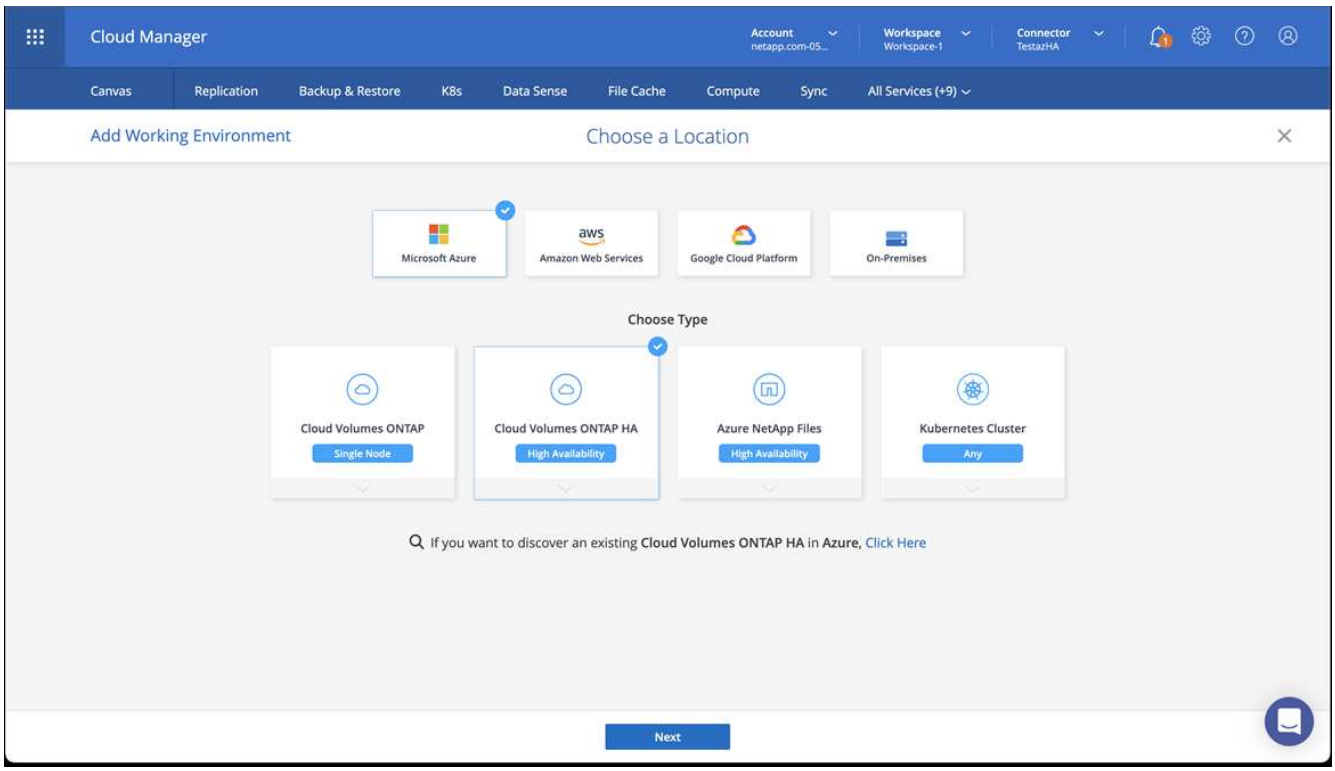
      請參閱 "從BlueXPr在Azure中建立連接器"。
3. 確認連接器正在執行、並切換至該連接器。



4. 為您的雲端環境建立工作環境。

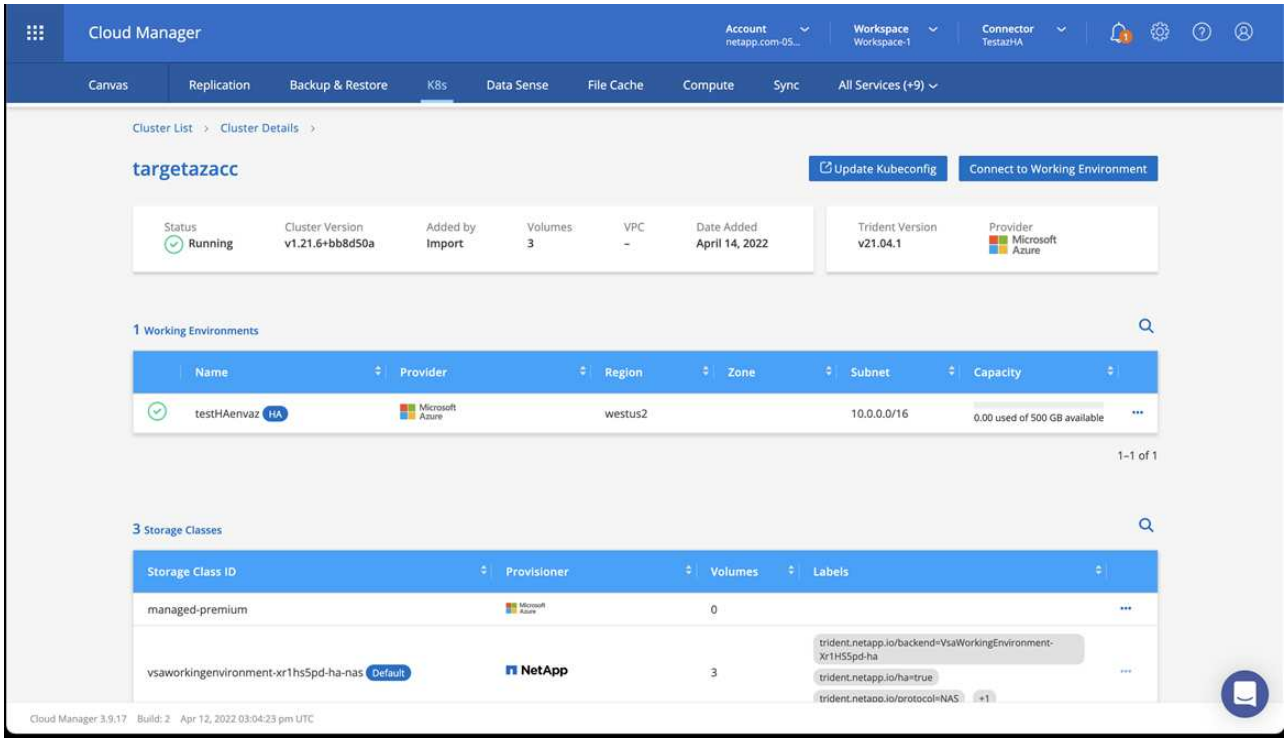
   a. 位置：「Microsoft Azure」。

   b. 輸入：Cloud Volumes ONTAP 「EHA」。



5. 匯入OpenShift叢集。叢集將連線至您剛建立的工作環境。

a. 選擇* K8s*>*叢集清單*>*叢集詳細資料*、即可檢視NetApp叢集詳細資料。



b. 請注意右上角的 Astra Trident 版本。

c. 請注意Cloud Volumes ONTAP 、顯示NetApp為資源配置程式的叢集儲存類別。

這會匯入您的Red Hat OpenShift叢集、並指派預設的儲存類別。您可以選取儲存類別。Astra Trident 會在匯入和探索程序中自動安裝。

6. 請注意此Cloud Volumes ONTAP 功能部署中的所有持續磁碟區和磁碟區。
7. 可作為單一節點或高可用度運作。Cloud Volumes ONTAP如果已啟用HA、請記下Azure中執行的HA狀態和節點部署狀態。

安裝及設定**Azure**的**Astra Control Center**

使用標準安裝Astra Control Center "安裝說明"。

使用Astra Control Center新增Azure儲存庫。請參閱 "設定Astra Control Center並新增鏟斗"。

安裝後設定Astra Control Center

視您的環境而定、安裝Astra Control Center之後可能需要額外的組態。

移除資源限制

某些環境使用資源配額和限制範圍物件、以防止命名空間中的資源消耗叢集上的所有可用CPU和記憶體。Astra Control Center並未設定上限、因此不符合這些資源。如果您的環境是以這種方式設定、則需要從您打算安裝Astra Control Center的命名空間中移除這些資源。

您可以使用下列步驟擷取及移除這些配額和限制。在這些範例中、命令輸出會在命令之後立即顯示。

步驟

1. 取得中的資源配額 `netapp-acc` （或自訂命名）命名空間：

```
kubectl get quota -n [netapp-acc or custom namespace]
```

回應：

```
NAME          AGE   REQUEST                                              LIMIT
pods-high     16s   requests.cpu: 0/20, requests.memory: 0/100Gi
limits.cpu: 0/200, limits.memory: 0/1000Gi
pods-low      15s   requests.cpu: 0/1, requests.memory: 0/1Gi
limits.cpu: 0/2, limits.memory: 0/2Gi
pods-medium   16s   requests.cpu: 0/10, requests.memory: 0/20Gi
limits.cpu: 0/20, limits.memory: 0/200Gi
```

2. 依名稱刪除所有資源配額：

```
kubectl delete resourcequota pods-high -n [netapp-acc or custom
namespace]
```

```
kubectl delete resourcequota pods-low -n [netapp-acc or custom
namespace]
```

```
kubectl delete resourcequota pods-medium -n [netapp-acc or custom
namespace]
```

3. 取得中的限制範圍 `netapp-acc` （或自訂命名）命名空間：

```
kubectl get limits -n [netapp-acc or custom namespace]
```

回應：

```
NAME              CREATED AT
cpu-limit-range   2022-06-27T19:01:23Z
```

4. 依名稱刪除限制範圍：

```
kubectl delete limitrange cpu-limit-range -n [netapp-acc or custom
namespace]
```

啟用命名空間之間的網路通訊

有些環境使用網路原則架構來限制命名空間之間的流量。Astra Control Center運算子和Astra Control Center位於不同的命名空間中。這些不同命名空間中的服務必須能夠彼此通訊。若要啟用此通訊、請遵循下列步驟。

步驟

1. 刪除Astra Control Center命名空間中的任何網路原則資源：

```
kubectl get networkpolicy -n [netapp-acc or custom namespace]
```

2. 對於上述命令傳回的每個網路原則物件、請使用下列命令加以刪除。以傳回物件的名稱取代 [object_name ：

```
kubectl delete networkpolicy [OBJECT_NAME] -n [netapp-acc or custom
namespace]
```

3. 套用下列資源檔案來設定 `acc-avp-network-policy` 允許 Astra 外掛程式服務向 Astra Control Center 服務提出要求的物件。將方括弧<>中的資訊取代為您環境中的資訊：

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: acc-avp-network-policy
  namespace: <ACC_NAMESPACE_NAME> # REPLACE THIS WITH THE ASTRA CONTROL
CENTER NAMESPACE NAME
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: <PLUGIN_NAMESPACE_NAME> #
REPLACE THIS WITH THE ASTRA PLUGIN NAMESPACE NAME
```

4. 套用下列資源檔案來設定 `acc-operator-network-policy` 此物件可讓Astra Control Center營運者與Astra Control Center服務進行通訊。將方括弧<>中的資訊取代為您環境中的資訊：

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: acc-operator-network-policy
  namespace: <ACC_NAMESPACE_NAME> # REPLACE THIS WITH THE ASTRA CONTROL
CENTER NAMESPACE NAME
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: <NETAPP-ACC-OPERATOR> #
REPLACE THIS WITH THE OPERATOR NAMESPACE NAME
```

新增自訂TLS憑證

Astra Control Center預設使用自我簽署的TLS憑證來處理入站控制器流量（僅適用於特定組態）、以及使用網頁瀏覽器進行Web UI驗證。您可以移除現有的自我簽署TLS憑證、並以由憑證授權單位（CA）簽署的TLS憑證取代。

> 預設的自我簽署憑證可用於兩種類型的連線：
>
> - HTTPS連線至Astra Control Center網路UI
> - 入口控制器流量（僅當 `ingressType: "AccTraefik"` 內容已在中設定 `astra_control_center.yaml` 安裝Astra Control Center期間的檔案）
>
> 取代預設TLS憑證會取代用於驗證這些連線的憑證。

開始之前

- Kubernetes叢集已安裝Astra Control Center
- 管理存取叢集上要執行的命令Shell `kubectl` 命令
- 來自CA的私密金鑰和憑證檔案

移除自我簽署的憑證

移除現有的自我簽署TLS憑證。

1. 使用SSH、以管理使用者身分登入裝載Astra Control Center的Kubernetes叢集。
2. 使用下列取代命令尋找與目前憑證相關的TLS密碼 `<ACC-deployment-namespace>` 使用Astra Control Center部署命名空間：

```
kubectl get certificate -n <ACC-deployment-namespace>
```

3. 使用下列命令刪除目前安裝的機密與憑證：

```
kubectl delete cert cert-manager-certificates -n <ACC-deployment-
namespace>
kubectl delete secret secure-testing-cert -n <ACC-deployment-namespace>
```

使用命令列新增憑證

新增由CA簽署的TLS憑證。

1. 使用下列命令以CA的私密金鑰和憑證檔案建立新的TLS秘密，並以適當的資訊取代括弧<>中的引數：

```
kubectl create secret tls <secret-name> --key <private-key-filename>
--cert <certificate-filename> -n <ACC-deployment-namespace>
```

2. 使用下列命令和範例編輯叢集自訂資源定義（CRD）檔案、然後變更 `spec.selfSigned` 價值 `spec.ca.secretName` 若要參考您先前建立的TLS秘密：

```
kubectl edit clusterissuers.cert-manager.io/cert-manager-certificates -n
<ACC-deployment-namespace>
....

#spec:
#  selfSigned: {}

spec:
  ca:
    secretName: <secret-name>
```

3. 使用下列命令和輸出範例來驗證變更是否正確、以及叢集是否已準備好驗證憑證、取代 `<ACC-deployment-namespace>` 使用Astra Control Center部署命名空間：

```
kubectl describe clusterissuers.cert-manager.io/cert-manager-
certificates -n <ACC-deployment-namespace>
....

Status:
  Conditions:
    Last Transition Time:  2021-07-01T23:50:27Z
    Message:               Signing CA verified
    Reason:                KeyPairVerified
    Status:                True
    Type:                  Ready
Events:                    <none>
```

4. 建立 `certificate.yaml` 使用下列範例將方括弧<>中的預留位置值取代為適當資訊的檔案：

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <certificate-name>
  namespace: <ACC-deployment-namespace>
spec:
  secretName: <certificate-secret-name>
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  dnsNames:
  - <astra.dnsname.example.com> #Replace with the correct Astra Control
Center DNS address
  issuerRef:
    kind: ClusterIssuer
    name: cert-manager-certificates
```

5. 使用下列命令建立憑證：

```
kubectl apply -f certificate.yaml
```

6. 使用下列命令和範例輸出來驗證憑證是否已正確建立、以及是否已使用您在建立期間所指定的引數（例如名稱、持續時間、續約期限及DNS名稱）。

```
kubectl describe certificate -n <ACC-deployment-namespace>
....

Spec:
  Dns Names:
    astra.example.com
  Duration:  125h0m0s
  Issuer Ref:
    Kind:        ClusterIssuer
    Name:        cert-manager-certificates
  Renew Before:  61h0m0s
  Secret Name:   <certificate-secret-name>
Status:
  Conditions:
    Last Transition Time:  2021-07-02T00:45:41Z
    Message:               Certificate is up to date and has not expired
    Reason:                Ready
    Status:                True
    Type:                  Ready
  Not After:               2021-07-07T05:45:41Z
  Not Before:              2021-07-02T00:45:41Z
  Renewal Time:            2021-07-04T16:45:41Z
  Revision:                1
Events:                    <none>
```

7. 編輯「入口CRD TLS」選項、使用下列命令和範例指向新的憑證密碼、並以適當的資訊取代方括弧<>中的預留位置值：

```
kubectl edit ingressroutes.traefik.containo.us -n <ACC-deployment-
namespace>
....

# tls:
#    options:
#      name: default
#    secretName: secure-testing-cert
#    store:
#      name: default

 tls:
    options:
      name: default
    secretName: <certificate-secret-name>
    store:
      name: default
```

8. 使用網頁瀏覽器瀏覽至Astra Control Center的部署IP位址。
9. 確認憑證詳細資料與您安裝的憑證詳細資料相符。
10. 匯出憑證並將結果匯入網頁瀏覽器中的憑證管理程式。