インストールの概要

Astra Control Center

NetApp
November 21, 2023

目次

インストールの概要 1
標準の手順で Astra Control Center をインストールします 1
OpenShift OperatorHub を使用して Astra Control Center をインストールします 23
Cloud Volumes ONTAP ストレージバックエンドに Astra Control Center をインストールします 29

インストールの概要

次の Astra Control Center のインストール手順のいずれかを選択して実行します。

- "標準の手順で Astra Control Center をインストールします"
- "（ Red Hat OpenShift を使用する場合） OpenShift OperatorHub を使用して Astra Control Center をインストールします"
- "Cloud Volumes ONTAP ストレージバックエンドに Astra Control Center をインストールします"

標準の手順で **Astra Control Center** をインストールします

Astra Control Center をインストールするには、ネットアップサポートサイトからインストールバンドルをダウンロードし、次の手順を実行して、 Astra Control Center Operator と Astra Control Center を環境にインストールします。この手順を使用して、インターネット接続環境またはエアギャップ環境に Astra コントロールセンターをインストールできます。

Red Hat OpenShift 環境では、を使用することもできます "代替手順" OpenShift OperatorHub を使用して Astra Control Center をインストールします。

必要なもの

- "インストールを開始する前に、 Astra Control Center の導入環境を準備します"。
- すべてのクラスタオペレータが正常な状態であり、使用可能であることを確認します。

  OpenShift の例：

  ```
  oc get clusteroperators
  ```

- すべての API サービスが正常な状態であり、使用可能であることを確認します。

  OpenShift の例：

  ```
  oc get apiservices
  ```

- 使用するネットアップ FQDN が、このクラスタにルーティング可能である必要があります。つまり、内部 DNS サーバに DNS エントリがあるか、すでに登録されているコア URL ルートを使用しています。

このタスクについて

Astra Control Center のインストールプロセスでは、次のことが実行されます。

- Astra コンポーネントを NetApp-acc' （またはカスタム名前の）名前空間にインストールします
- デフォルトアカウントを作成します。
- Astra Control Center のこのインスタンスに対して、デフォルトの管理ユーザの電子メールアドレスとデフォルトのワンタイムパスワード「 ACC-<UUID_OF_INSTALLIVE> 」を設定します。このユーザーには、システムのオーナーロールが割り当てられ、 UI への初回ログイン時に必要になります。

- Astra Control Center のすべてのポッドが実行されていることを確認するのに役立ちます。
- Astra の UI をインストールします。

> （環境 the Astra Data Store Early Access Program（EAP）リリースのみ）Astra Control Centerを使用してAstraデータストアを管理し、VMwareワークフローを有効にする場合 Astra Control Centerは'pcloud'ネームスペースにのみ導入し'この手順 の手順で説明したNetApp-acc'ネームスペースまたはカスタムネームスペースには導入しないでください

> インストールプロセス全体で次のコマンドを実行しないでください。 'kubectl delete -f Astra_control_center_deployment.yaml'

> Docker Engine の代わりに Red Hat の Podman を使用している場合は、 Docker コマンドの代わりに Podman コマンドを使用できます。

手順

Astra Control Center をインストールするには、次の手順に従います。

- Astra Control Centerバンドルをダウンロードして開梱します
- ネットアップAstra kubectlプラグインをインストール
- [イメージをローカルレジストリに追加します]
- [認証要件を持つレジストリのネームスペースとシークレットを設定します]
- Astra Control Center オペレータを設置します
- Astra Control Center を設定します
- Astra Control Center とオペレータのインストールを完了します
- [システムステータスを確認します]
- [ロードバランシング用の入力を設定します]
- Astra Control Center UI にログインします

Astra Control Centerバンドルをダウンロードして開梱します

1. から Astra Control Center バンドル（「 Astra - control-ccenter-[version].tar.gz` ）をダウンロードします "ネットアップサポートサイト"。
2. から Astra Control Center 証明書とキーの zip をダウンロードします "ネットアップサポートサイト"。
3. （任意）次のコマンドを使用して、バンドルのシグニチャを確認します。

```
openssl dgst -sha256 -verify astra-control-center[version].pub
-signature <astra-control-center[version].sig astra-control-
center[version].tar.gz
```

4. 画像を抽出します。

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

ネットアップ**Astra kubectl**プラグインをインストール

NetApp Astra 'kubectl'コマンド・ライン・プラグインは'Astra Control Centerの導入とアップグレードに関連する一般的なタスクを実行する際に時間を節約します

必要なもの

ネットアップでは、プラグイン用のバイナリを提供しており、CPUアーキテクチャやオペレーティングシステムが異なる場合はそのプラグインをこのタスクを実行する前に、使用しているCPUとオペレーティングシステムを把握しておく必要があります。LinuxおよびMacオペレーティングシステムでは、「uname -a」コマンドを使用してこの情報を収集できます。

手順

1. 使用可能なNetApp Astra 'kubectl'プラグイン・バイナリを列挙し'オペレーティング・システムとCPUアーキテクチャに必要なファイル名を書き留めます

```
ls kubectl-astra/
```

2. ファイルを標準のkubectl'ユーティリティと同じ場所にコピーしますこの例では'kubectl'ユーティリティは'/usr/local/bin'ディレクトリにあります「<binary-name>」を必要なファイル名に置き換えます。

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

イメージをローカルレジストリに追加します

1. Astraディレクトリに移動します。

```
cd acc
```

2. Astra Control Center イメージディレクトリ内のファイルをローカルレジストリに追加します。

> 以下の画像の自動ロードについては、サンプルスクリプトを参照してください。

   a. レジストリにログインします。

      Docker ：

```
docker login [your_registry_path]
```

      Podman ：

```
podman login [your_registry_path]
```

b. 適切なスクリプトを使用して、イメージのロード、イメージのタグ付け、[[[[[</Z1>[</Z1>[</Z1>_image_local_registry_push]] ローカルレジストリにイメージをプッシュします。 </Z2>

Docker ：

```
export REGISTRY=[Docker_registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image
trimming the 'Loaded images: '
  astraImage=$(docker load --input ${astraImageFile} | sed 's/Loaded
image: //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  docker tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  docker push ${REGISTRY}/${astraImage}
done
```

Podman ：

```
export REGISTRY=[Registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image trimming
the 'Loaded images: '
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  podman push ${REGISTRY}/${astraImage}
done
```

認証要件を持つレジストリのネームスペースとシークレットを設定します

1. 認証が必要なレジストリを使用する場合は、次の手順を実行する必要があります。

   a. NetApp-acc-operator という名前空間を作成します。

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

対応：

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

b. NetApp-acc-operator ネームスペースのシークレットを作成します。Docker 情報を追加して次のコマンドを実行します。

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

回答例：

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

c. NetApp-acc`（またはカスタムの名前を付けた）ネームスペースを作成します。

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

回答例：

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

d. NetApp-acc`（またはカスタムの名前を付けた）ネームスペースのシークレットを作成します。Docker 情報を追加して次のコマンドを実行します。

```
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]
```

応答

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

a. [[[[sup_kubeconfig _secret]] （オプション）インストール後に Astra Control Center でクラスタを自動的に管理する場合は、このコマンドを使用して展開する Astra Control Center ネームスペース内のシークレットとして kubeconfig を指定する必要があります。

```
kubectl create secret generic [acc-kubeconfig-cred or custom secret
name] --from-file=<path-to-your-kubeconfig> -n [netapp-acc or custom
namespace]
```

Astra Control Center オペレータを設置します

1. Astra Control Center オペレータの配備 YAML (`Astra_control_center_deployment.yaml ') を編集して、ローカルのレジストリと秘密を参照します。

```
vim astra_control_center_operator_deploy.yaml
```

   a. 認証が必要なレジストリを使用する場合は、デフォルト行の「 imagePullSecret:[] 」を次のように置き換えます。

```
imagePullSecrets:
- name: <name_of_secret_with_creds_to_local_registry>
```

   b. 「 kube-rbac プロキシ」イメージの「 [Your_registry_path] 」を、でイメージをプッシュしたレジストリパスに変更します 前の手順。

   c. 「 acc-operator-controller-manager 」イメージの「 [Your_registry_path] 」を、でイメージをプッシュしたレジストリパスに変更します 前の手順。

   d. （ Astra データストアプレビューを使用するインストールの場合）この問題に関する既知の情報を参照してください "ストレージクラスのプロビジョニングと YAML に対する追加の変更"。

```
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
  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
        command:
        - /manager
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        image: [your_registry_path]/acc-operator:[version x.y.z]
        imagePullPolicy: IfNotPresent
      imagePullSecrets: []
```

2. Astra Control Center オペレータをインストールします。

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

回答例：

```
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
```

Astra Control Center を設定します

1. Astra Control Center カスタムリソース（ CR ）ファイル（「 Astra _control_center_min YAML 」）を編集して、アカウント、 AutoSupport 、レジストリ、およびその他の必要な構成を作成します。

> ⓘ 環境に追加のカスタマイズが必要な場合は 'Astra_control_center.yaml を代替 CR として使用できます「 Astra_control_center_min YAML 」はデフォルトの CR で、ほとんどのインストールに適しています。

```
vim astra_control_center_min.yaml
```

> ⓘ CR によって設定されたプロパティは、最初の Astra Control Center の導入後に変更することはできません。

> ⓘ 認証を必要としないレジストリを使用している場合は、「 imageRegistry 」内の「秘密」行を削除する必要があります。削除しないとインストールが失敗します。

a. 前の手順でイメージをプッシュしたレジストリパスに '[Your_registry_path]' を変更します

b. 「 accountName 」文字列を、アカウントに関連付ける名前に変更します。

c. 「 astraトラ アドレス」文字列をブラウザで使用する FQDN に変更して、 Astra にアクセスします。アドレスには 'http://' または https:// を使用しないでくださいこの FQDN をコピーしてで使用します 後の手順。

d. 「 email 」の文字列をデフォルトの初期管理者アドレスに変更します。この E メールアドレスをコピ

ーしてで使用します 後の手順。

e. インターネットに接続されていないサイトの場合は AutoSupport の「 enrolled 」を「 false 」に変更し、接続されているサイトの場合は「 true 」を保持します。

f. （オプション）アカウントに関連付けられたユーザの姓「 firstName 」と名「 lastName 」を追加します。この手順は、 UI ですぐに実行することもあとで実行することもできます。

g. （任意）インストールで必要に応じて、「 torageClass 」の値を別の Trident ストレージクラスリソースに変更します。

h. （オプション）インストール後に Astra Control Center でクラスタを自動的に管理する場合は このクラスタの kubeconfig を含むシークレットを作成しましたを使用して、シークレットの名前を指定します。この YAML ファイルに「 astraeKubeConfigSecret ： "acc-kubeconfig -cred or custom secret name" 」という名前の新しいフィールドを追加します

i. 次のいずれかの手順を実行します。

- * その他の入力コントローラ（ ingressType: Generic ） * ：これはアストラコントロールセンターでのデフォルトのアクションです。Astra Control Center を展開したら、 Astra Control Center を URL で公開するように入力コントローラを設定する必要があります。

  デフォルトの Astra Control Center インストールでは ' ゲートウェイ (service/traefik`) が ClusterIP` タイプに設定されますこのデフォルトのインストールでは、トラフィックをルーティングするために Kubernetes IngressController/Ingress を追加で設定する必要があります。入力を使用する場合は、を参照してください "ロードバランシング用の入力を設定します"。

- * サービスロードバランサ (ingressType: AccTraefik) *: IngressController をインストールしない場合、または入力リソースを作成しない場合は、「 ingressType 」を「 AccTraefik 」に設定します。

  これにより 'Astra Control Center traefik' ゲートウェイが Kubernetes LoadBalancer タイプのサービスとして導入されます

  Astra Control Center は、 Astra Control Center ネームスペースの "LoadBalancer (svc/traefik`) タイプのサービスを使用し、アクセス可能な外部 IP アドレスが割り当てられている必要があります。お使いの環境でロードバランサが許可されていて、設定されていない場合は、 MetalLB または別の外部サービスロードバランサを使用して、外部 IP アドレスをサービスに割り当てることができます。内部 DNS サーバ構成では、 Astra Control Center に選択した DNS 名を、負荷分散 IP アドレスに指定する必要があります。

> サービスタイプ「 LoadBalancer 」および入力の詳細については、を参照してください "要件"。

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"
  ingressType: "Generic"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
```

Astra Control Center とオペレータのインストールを完了します

1. 前の手順で NetApp-acc` （またはカスタム）ネームスペースを作成していない場合は、次のようにします。

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

回答例：

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

2. Astra Control Center を NetApp-acc' （またはカスタムの）名前空間にインストールします

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

回答例：

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

システムステータスを確認します

OpenShift を使用する場合は、同等の OC コマンドを検証手順に使用できます。

1. すべてのシステムコンポーネントが正常にインストールされたことを確認します。

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

各ポッドのステータスは「 Running 」になります。システムポッドが展開されるまでに数分かかることがあります。

回答例：

```
NAME                                        READY   STATUS    RESTARTS
AGE
acc-helm-repo-5f75c5f564-bzqmt              1/1     Running   0
11m
activity-6b8f7cccb9-mlrn4                   1/1     Running   0
9m2s
api-token-authentication-6hznt              1/1     Running   0
8m50s
api-token-authentication-qpfgb              1/1     Running   0
8m50s
api-token-authentication-sqnb7              1/1     Running   0
8m50s
asup-5578bbdd57-dxkbp                       1/1     Running   0
9m3s
authentication-56bff4f95d-mspmq             1/1     Running   0
7m31s
bucketservice-6f7968b95d-9rrrl              1/1     Running   0
8m36s
cert-manager-5f6cf4bc4b-82khn               1/1     Running   0
6m19s
cert-manager-cainjector-76cf976458-sdrbc    1/1     Running   0
6m19s
cert-manager-webhook-5b7896bfd8-2n45j       1/1     Running   0
6m19s
cloud-extension-749d9f684c-8bdhq            1/1     Running   0
9m6s
cloud-insights-service-7d58687d9-h5tzw      1/1     Running   2
8m56s
composite-compute-968c79cb5-nv7l4           1/1     Running   0
9m11s
composite-volume-7687569985-jg9gg           1/1     Running   0
8m33s
```

```
credentials-5c9b75f4d6-nx9cz                1/1     Running   0
8m42s
entitlement-6c96fd8b78-zt7f8                1/1     Running   0
8m28s
features-5f7bfc9f68-gsjnl                   1/1     Running   0
8m57s
fluent-bit-ds-h88p7                         1/1     Running   0
7m22s
fluent-bit-ds-krhnj                         1/1     Running   0
7m23s
fluent-bit-ds-l5bjj                         1/1     Running   0
7m22s
fluent-bit-ds-lrclb                         1/1     Running   0
7m23s
fluent-bit-ds-s5t4n                         1/1     Running   0
7m23s
fluent-bit-ds-zpr6v                         1/1     Running   0
7m22s
graphql-server-5f5976f4bd-vbb4z             1/1     Running   0
7m13s
identity-56f78b8f9f-8h9p9                   1/1     Running   0
8m29s
influxdb2-0                                 1/1     Running   0
11m
krakend-6f8d995b4d-5khkl                    1/1     Running   0
7m7s
license-5b5db87c97-jmxzc                    1/1     Running   0
9m
login-ui-57b57c74b8-6xtv7                   1/1     Running   0
7m10s
loki-0                                      1/1     Running   0
11m
monitoring-operator-9dbc9c76d-8znck         2/2     Running   0
7m33s
nats-0                                      1/1     Running   0
11m
nats-1                                      1/1     Running   0
10m
nats-2                                      1/1     Running   0
10m
nautilus-6b9d88bc86-h8kfb                   1/1     Running   0
8m6s
nautilus-6b9d88bc86-vn68r                   1/1     Running   0
8m35s
openapi-b87d77dd8-5dz9h                     1/1     Running   0
9m7s
```

```
polaris-consul-consul-5ljfb                1/1     Running   0
11m
polaris-consul-consul-s5d5z                1/1     Running   0
11m
polaris-consul-consul-server-0             1/1     Running   0
11m
polaris-consul-consul-server-1             1/1     Running   0
11m
polaris-consul-consul-server-2             1/1     Running   0
11m
polaris-consul-consul-twmpq                1/1     Running   0
11m
polaris-mongodb-0                          2/2     Running   0
11m
polaris-mongodb-1                          2/2     Running   0
10m
polaris-mongodb-2                          2/2     Running   0
10m
polaris-ui-84dc87847f-zrg8w                1/1     Running   0
7m12s
polaris-vault-0                            1/1     Running   0
11m
polaris-vault-1                            1/1     Running   0
11m
polaris-vault-2                            1/1     Running   0
11m
public-metrics-657698b66f-67pgt            1/1     Running   0
8m47s
storage-backend-metrics-6848b9fd87-w7x8r   1/1     Running   0
8m39s
storage-provider-5ff5868cd5-r9hj7          1/1     Running   0
8m45s
telegraf-ds-dw4hg                          1/1     Running   0
7m23s
telegraf-ds-k92gn                          1/1     Running   0
7m23s
telegraf-ds-mmxjl                          1/1     Running   0
7m23s
telegraf-ds-nhs8s                          1/1     Running   0
7m23s
telegraf-ds-rj7lw                          1/1     Running   0
7m23s
telegraf-ds-tqrkb                          1/1     Running   0
7m23s
telegraf-rs-9mwgj                          1/1     Running   0
7m23s
```

```
telemetry-service-56c49d689b-ffrzx         1/1     Running   0
8m42s
tenancy-767c77fb9d-g9ctv                   1/1     Running   0
8m52s
traefik-5857d87f85-7pmx8                   1/1     Running   0
6m49s
traefik-5857d87f85-cpxgv                   1/1     Running   0
5m34s
traefik-5857d87f85-lvmlb                   1/1     Running   0
4m33s
traefik-5857d87f85-t2xlk                   1/1     Running   0
4m33s
traefik-5857d87f85-v9wpf                   1/1     Running   0
7m3s
trident-svc-595f84dd78-zb816               1/1     Running   0
8m54s
vault-controller-86c94fbf4f-krttq          1/1     Running   0
9m24s
```

2. （オプション）インストールが完了したことを確認するには、次のコマンドを使用して「 acc-operator' 」ログを監視します。

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

> 「 accHost 」クラスタの登録は最後の操作の 1 つであり、失敗した場合、原因 の配備に失敗することはありません。ログにクラスタ登録エラーが示された場合は、クラスタ追加ワークフローを通じて再度登録を試行できます "UI で" または API 。

3. すべてのポッドが動作している場合は、 Astra Control Center Operator によってインストールされた「 Astrad ControlCenter 」インスタンスを取得して、インストールが正常に完了したことを確認します。

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

4. YAML で、「 Deployed 」値に対する応答の「 status.deploymentState` 」フィールドを確認します。導入に失敗した場合は、代わりにエラーメッセージが表示されます。
5. Astra Control Center にログインするときに使用するワンタイムパスワードを取得するには、「 status.uuid 」値をコピーします。パスワードは「 ACC-` 」の後に UUID 値（「 ACC-[UUID] 」）、またはこの例では「 ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f 」）です。

YAML の詳細のサンプル

```
name: astra
   namespace: netapp-acc
   resourceVersion: "104424560"
   selfLink: /apis/astra.netapp.io/v1/namespaces/netapp-
acc/astracontrolcenters/astra
   uid: 9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f
 spec:
   accountName: Example
   astraAddress: astra.example.com
   astraVersion: 21.12.60
   autoSupport:
     enrolled: true
     url: https://support.netapp.com/asupprod/post/1.0/postAsup
   crds: {}
   email: admin@example.com
   firstName: SRE
   imageRegistry:
     name: registry_name/astra
     secret: astra-registry-cred
   lastName: Admin
 status:
   accConditionHistory:
     items:
     - astraVersion: 21.12.60
       condition:
         lastTransitionTime: "2021-11-23T02:23:59Z"
         message: Deploying is currently in progress.
         reason: InProgress
         status: "False"
         type: Ready
       generation: 2
       observedSpec:
         accountName: Example
         astraAddress: astra.example.com
         astraVersion: 21.12.60
         autoSupport:
           enrolled: true
           url: https://support.netapp.com/asupprod/post/1.0/postAsup
         crds: {}
         email: admin@example.com
         firstName: SRE
         imageRegistry:
           name: registry_name/astra
```

```
          secret: astra-registry-cred
        lastName: Admin
      timestamp: "2021-11-23T02:23:59Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:23:59Z"
        message: Deploying is currently in progress.
        reason: InProgress
        status: "True"
        type: Deploying
      generation: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      timestamp: "2021-11-23T02:23:59Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Post Install was successful
        observedGeneration: 2
        reason: Complete
        status: "True"
        type: PostInstallComplete
      generation: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
```

```
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      timestamp: "2021-11-23T02:29:41Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Deploying succeeded.
        reason: Complete
        status: "False"
        type: Deploying
      generation: 2
      observedGeneration: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      observedVersion: 21.12.60
      timestamp: "2021-11-23T02:29:41Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Astra is deployed
        reason: Complete
        status: "True"
        type: Deployed
      generation: 2
      observedGeneration: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
```

```
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      observedVersion: 21.12.60
      timestamp: "2021-11-23T02:29:41Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Astra is deployed
        reason: Complete
        status: "True"
        type: Ready
      generation: 2
      observedGeneration: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      observedVersion: 21.12.60
      timestamp: "2021-11-23T02:29:41Z"
  certManager: deploy
  cluster:
    type: OCP
    vendorVersion: 4.7.5
    version: v1.20.0+bafe72f
  conditions:
  - lastTransitionTime: "2021-12-08T16:19:55Z"
    message: Astra is deployed
    reason: Complete
    status: "True"
    type: Ready
  - lastTransitionTime: "2021-12-08T16:19:55Z"
    message: Deploying succeeded.
```

```
      reason: Complete
      status: "False"
      type: Deploying
    - lastTransitionTime: "2021-12-08T16:19:53Z"
      message: Post Install was successful
      observedGeneration: 2
      reason: Complete
      status: "True"
      type: PostInstallComplete
    - lastTransitionTime: "2021-12-08T16:19:55Z"
      message: Astra is deployed
      reason: Complete
      status: "True"
      type: Deployed
    deploymentState: Deployed
    observedGeneration: 2
    observedSpec:
      accountName: Example
      astraAddress: astra.example.com
      astraVersion: 21.12.60
      autoSupport:
        enrolled: true
        url: https://support.netapp.com/asupprod/post/1.0/postAsup
      crds: {}
      email: admin@example.com
      firstName: SRE
      imageRegistry:
        name: registry_name/astra
        secret: astra-registry-cred
      lastName: Admin
    observedVersion: 21.12.60
    postInstall: Complete
    uuid: 9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f
  kind: List
  metadata:
   resourceVersion: ""
   selfLink: ""
```

ロードバランシング用の入力を設定します

Kubernetes 入力コントローラをセットアップして、クラスタのロードバランシングなどのサービスへの外部アクセスを管理できます。

この手順 では、入力コントローラ（「 ingressType: Generic 」）の設定方法について説明します。これは、Astra Control Center でのデフォルトのアクションです。Astra Control Center を展開したら、 Astra Control Center を URL で公開するように入力コントローラを設定する必要があります。

> 入力コントローラを設定しない場合は、「 ingressType: AccTraefik 」を設定できます。Astra Control Center は、 Astra Control Center ネームスペースの "LoadBalancer (svc/traefik`) タイプのサービスを使用し、アクセス可能な外部 IP アドレスが割り当てられている必要があります。お使いの環境でロードバランサが許可されていて、設定されていない場合は、 MetalLB または別の外部サービスロードバランサを使用して、外部 IP アドレスをサービスに割り当てることができます。内部 DNS サーバ構成では、 Astra Control Center に選択した DNS 名を、負荷分散 IP アドレスに指定する必要があります。サービスタイプ「 LoadBalancer 」および入力の詳細については、を参照してください "要件"。

この手順は、使用する入力コントローラのタイプによって異なります。

- nginx 入力コントローラ
- OpenShift 入力コントローラ

必要なもの

- が必要です "入力コントローラ" すでに導入されている必要があります。
- 。 "入力クラス" 入力コントローラに対応するものがすでに作成されている必要があります。
- V1.19 と v1.22 の間で Kubernetes のバージョンを使用している。

Nginx Ingress Controller の手順

1. タイプのシークレットを作成します "8a637503539b25b68130b6e8003579d9" に示すように 'NetApp-acc'（またはカスタム名前の）名前空間内の TLS 秘密鍵と証明書の場合 "TLS シークレット"。
2. 非推奨または新しいスキーマのいずれかのv1beta1'（Kubernetesバージョン1.22で非推奨）またはv1'リソースタイプを使用して'NetApp-acc'（またはカスタムネームド）ネームスペースに入力リソースを導入します
   a. v1beta' 非推奨スキーマについては ' 次の例を参照してください

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-acc
  namespace: [netapp-acc or custom namespace]
  annotations:
    kubernetes.io/ingress.class: [class name for nginx controller]
spec:
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    http:
      paths:
      - backend:
        serviceName: traefik
        servicePort: 80
        pathType: ImplementationSpecific
```

b. 「 v1 」の新しいスキーマについては、次の例を参照してください。

```
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 addess>
    http:
      paths:
        - path:
          backend:
            service:
              name: traefik
              port:
                number: 80
          pathType: ImplementationSpecific
```

OpenShift 入力コントローラの手順

1. 証明書を調達し、 OpenShift ルートで使用できるようにキー、証明書、および CA ファイルを取得します。
2. OpenShift ルートを作成します。

```
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
```

Astra Control Center UI にログインします

Astra Control Center をインストールした後、デフォルトの管理者のパスワードを変更し、 Astra Control Center UI ダッシュボードにログインします。

手順

1. ブラウザで 'Astra_control_center_min YAML の 'astraitAddress' で使用した FQDN を入力します Astra Control Center をインストールした。
2. プロンプトが表示されたら、自己署名証明書を受け入れます。

カスタム証明書はログイン後に作成できます。

3. Astra Control Center のログインページで 'eMA_control_center_min YAML で 'email' に使用した値を次のように入力します Astra Control Center をインストールしたに続き、ワンタイム・パスワード（「ACC-[UUID]」）を入力します。

   誤ったパスワードを 3 回入力すると、管理者アカウントは 15 分間ロックされます。

4. **[Login]** を選択します。
5. プロンプトが表示されたら、パスワードを変更します。

   初めてログインする際にパスワードを忘れた場合、他の管理ユーザアカウントがまだ作成されていないときは、ネットアップのサポートに問い合わせて、パスワードのリカバリに関するサポートを依頼してください。

6. （オプション）既存の自己署名 TLS 証明書を削除して、に置き換えます "認証局（CA）が署名したカスタム TLS 証明書"。

インストールのトラブルシューティングを行います

いずれかのサービスのステータスが「Error」の場合は、ログを確認できます。400 ~ 500 の範囲の API 応答コードを検索します。これらは障害が発生した場所を示します。

手順

1. Astra Control Center のオペレータログを調べるには、次のように入力します。

```
kubectl logs --follow -n netapp-acc-operator $(kubectl get pods -n
netapp-acc-operator -o name)  -c manager
```

次のステップ

を実行して導入を完了します "セットアップのタスク"。

OpenShift OperatorHub を使用して Astra Control Center をインストールします

Red Hat OpenShift を使用する場合は、 Red Hat 認定オペレータを使用して Astra Control Center をインストールできます。この手順を使用して、から Astra Control Center をインストールします "Red Hat エコシステムカタログ" または、 Red Hat OpenShift Container Platform を使用します。

この手順を完了したら、インストール手順に戻ってを実行する必要があります "残りのステップ" インストールが成功したかどうかを確認し、ログオンします。

必要なもの

- "インストールを開始する前に、 Astra Control Center の導入環境を準備します"。

- OpenShift クラスタから、すべてのクラスタオペレータが正常な状態にあることを確認します（「 available 」は「 true 」）。

```
oc get clusteroperators
```

- OpenShift クラスタから、すべての API サービスが正常な状態（「 available 」は「 true 」）になっていることを確認します。

```
oc get apiservices
```

- データセンターに Astra Control Center の FQDN アドレスを作成しておきます。
- 説明したインストール手順を実行するために必要な権限と Red Hat OpenShift Container Platform へのアクセスが必要です。

手順

- Astra Control Centerバンドルをダウンロードして開梱します
- ネットアップAstra kubectlプラグインをインストール
- [イメージをローカルレジストリに追加します]
- [オペレータインストールページを検索します]
- [オペレータをインストールします]
- Astra Control Center をインストールします

Astra Control Centerバンドルをダウンロードして開梱します

1. から Astra Control Center バンドル（「 Astra - control-ccenter-[version].tar.gz` ）をダウンロードします "ネットアップサポートサイト"。
2. から Astra Control Center 証明書とキーの zip をダウンロードします "ネットアップサポートサイト"。
3. （任意）次のコマンドを使用して、バンドルのシグニチャを確認します。

```
openssl dgst -sha256 -verify astra-control-center[version].pub
-signature <astra-control-center[version].sig astra-control-
center[version].tar.gz
```

4. 画像を抽出します。

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

ネットアップAstra kubectlプラグインをインストール

NetApp Astra 'kubectl'コマンド・ライン・プラグインは'Astra Control Centerの導入とアップグレードに関連

する一般的なタスクを実行する際に時間を節約します

必要なもの

ネットアップでは、プラグイン用のバイナリを提供しており、CPUアーキテクチャやオペレーティングシステムが異なる場合はそのプラグインをこのタスクを実行する前に、使用しているCPUとオペレーティングシステムを把握しておく必要があります。LinuxおよびMacオペレーティングシステムでは、「uname -a」コマンドを使用してこの情報を収集できます。

手順

1. 使用可能なNetApp Astra 'kubectl'プラグイン・バイナリを列挙し'オペレーティング・システムとCPUアーキテクチャに必要なファイル名を書き留めます

```
ls kubectl-astra/
```

2. ファイルを標準のkubectl'ユーティリティと同じ場所にコピーしますこの例では'kubectl'ユーティリティは'/usr/local/bin'ディレクトリにあります「<binary-name>」を必要なファイル名に置き換えます。

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

イメージをローカルレジストリに追加します

1. Astraディレクトリに移動します。

```
cd acc
```

2. Astra Control Center イメージディレクトリ内のファイルをローカルレジストリに追加します。

> 以下の画像の自動ロードについては、サンプルスクリプトを参照してください。

   a. レジストリにログインします。

      Docker ：

```
docker login [your_registry_path]
```

      Podman ：

```
podman login [your_registry_path]
```

   b. 適切なスクリプトを使用して、イメージのロード、イメージのタグ付け、[[[[[</Z1>[</Z1>[</Z1>_image_local_registry_push]] ローカルレジストリにイメージをプッシュします。 </Z2>

Docker ：

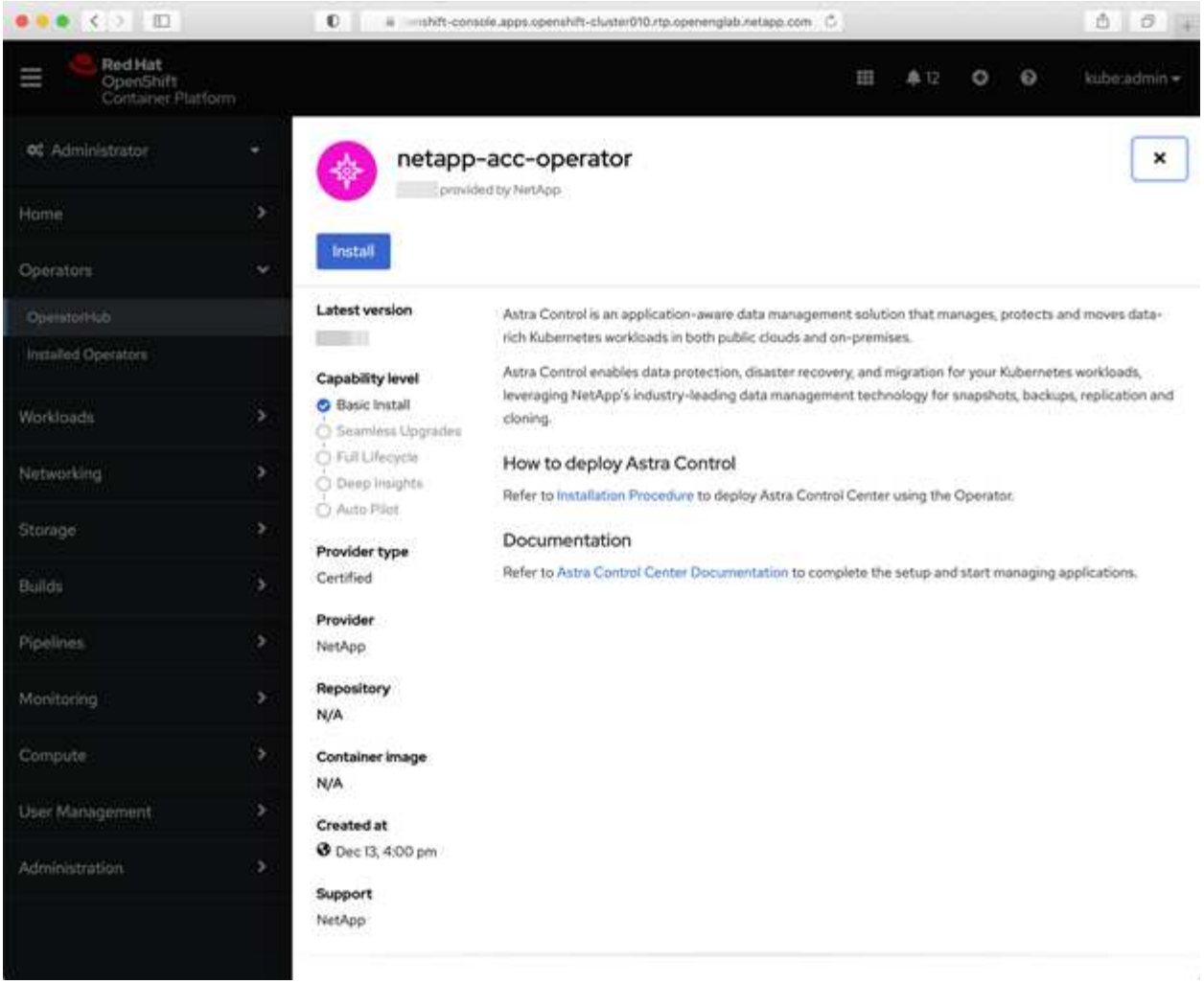
```
export REGISTRY=[Docker_registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image
trimming the 'Loaded images: '
  astraImage=$(docker load --input ${astraImageFile} | sed 's/Loaded
image: //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  docker tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  docker push ${REGISTRY}/${astraImage}
done
```

Podman ：

```
export REGISTRY=[Registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image trimming
the 'Loaded images: '
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  podman push ${REGISTRY}/${astraImage}
done
```

オペレータインストールページを検索します

1. 次のいずれかの手順を実行して、オペレータインストールページにアクセスします。
   - Red Hat OpenShift の Web コンソールから



:

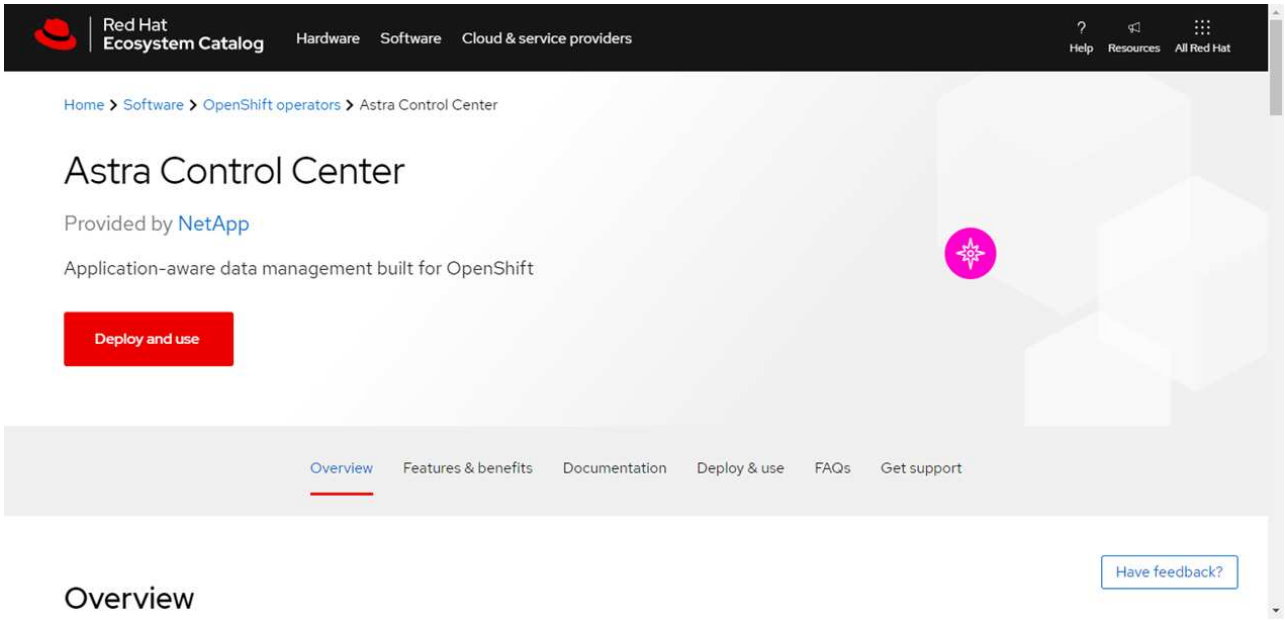
i. OpenShift Container Platform UI にログインします。

ii. サイドメニューから、 * 演算子 > OperatorHub * を選択します。

iii. NetApp Astra Control Center オペレータを選択します。

iv. 「 * Install * 」を選択します。

- Red Hat エコシステムカタログから
:



i. NetApp Astra Control Center を選択します "演算子"。

ii. [Deploy and Use] を選択します。

オペレータをインストールします

1. 「 * インストールオペレータ * 」ページに必要事項を入力し、オペレータをインストールします。

   オペレータはすべてのクラスタネームスペースで使用できます。

   a. operator 名前空間または NetApp-acc-operator' 名前空間を選択すると、オペレータのインストール時に自動的に作成されます。

   b. 手動または自動の承認方法を選択します。

      手動による承認が推奨されます。1 つのクラスタで実行する演算子インスタンスは 1 つだけです。

   c. 「 * Install * 」を選択します。

      手動承認方式を選択した場合は、このオペレータの手動インストール計画を承認するように求められます。

2. コンソールで、 OperatorHub メニューに移動して、オペレータが正常にインストールされたことを確認します。

Astra Control Center をインストールします

1. Astra Control Center オペレータの詳細ビュー内のコンソールから、 [Provided API] セクションの [Create instance] を選択します。

2. Create AstraControlCenter フォーム・フィールドに次のように入力します

   a. Astra Control Center の名前を保持または調整します。

b. （オプション） AutoSupport を有効または無効にします。Auto Support 機能の保持を推奨します。

c. Astra Control Center のアドレスを入力します。アドレスには 'http://' または https:// を入力しないでください

d. Astra Control Center のバージョンを入力します。たとえば、 21.12.60 と入力します。

e. アカウント名、 E メールアドレス、および管理者の姓を入力します。

f. デフォルトのボリューム再利用ポリシーをそのまま使用します。

g. * Image Registry * に、ローカルコンテナイメージのレジストリパスを入力します。アドレスには 'http://' または https:// を入力しないでください

h. 認証が必要なレジストリを使用する場合は、シークレットを入力します。

i. 管理者の名を入力します。

j. リソースの拡張を構成する。

k. デフォルトのストレージクラスは保持します。

l. CRD 処理の環境設定を定義します。

3. 「 Create 」を選択します。

次のステップ

Astra Control Center が正しくインストールされたことを確認し、を完了します "残りのステップ" ログインしてください。さらに、の導入も完了します "セットアップのタスク"。

Cloud Volumes ONTAP ストレージバックエンドに Astra Control Center をインストールします

Astra Control Center を使用すると、 Kubernetes クラスタと Cloud Volumes ONTAP インスタンスを自己管理することで、ハイブリッドクラウド環境でアプリケーションを管理できます。Astra Control Center は、オンプレミスの Kubernetes クラスタ、またはクラウド環境内の自己管理型 Kubernetes クラスタのいずれかに導入できます。

これらのいずれかの環境では、 Cloud Volumes ONTAP をストレージバックエンドとして使用して、アプリケーションデータの管理処理を実行できます。バックアップターゲットとして S3 バケットを設定することもできます。

Amazon Web Services （ AWS ）および Cloud Volumes ONTAP ストレージバックエンドを使用する Microsoft Azure に Astra Control Center をインストールするには、クラウド環境に応じて次の手順を実行します。

- Amazon Web Services に Astra Control Center を導入
- Microsoft Azure に Astra Control Center を導入

Amazon Web Services に Astra Control Center を導入

Amazon Web Services （ AWS ）パブリッククラウドでホストされる自己管理型の Kubernetes クラスタに Astra Control Center を導入できます。

Astra Control Center の導入でサポートされるのは、自己管理 OpenShift Container Platform （ OCP ）クラス

タのみです。

AWSに必要なもの

AWS に Astra Control Center を導入する前に、次のものが必要です。

- Astra Control Center ライセンス。を参照してください "Astra Control Center のライセンス要件"。
- "Astra Control Center の要件を満たす"。
- NetApp Cloud Central アカウント
- Red Hat OpenShift Container Platform （ OCP ）権限（ポッドを作成するためのネームスペースレベル）
- バケットとコネクタを作成するための権限を持つ AWS クレデンシャル、アクセス ID 、シークレットキー
- AWS アカウント Elastic Container Registry （ ECR ）アクセスおよびログイン
- AWS がホストするゾーンと Route 53 エントリは、 Astra Control UI にアクセスするために必要です

AWS の運用環境の要件

Astra Control Center を使用するには、 AWS 向けに次の運用環境が必要です。

- Red Hat OpenShift Container Platform 4.8 の場合

> Astra Control Center をホストするオペレーティングシステムが、環境の公式ドキュメントに記載されている基本的なリソース要件を満たしていることを確認します。

Astra Control Center では、環境のリソース要件に加え、次のリソースが必要です。

コンポーネント	要件
バックエンドの **NetApp Cloud Volumes ONTAP** ストレージ容量	300GB 以上のデータがあります
ワーカーノード（ **AWS EC2** の要件）	少なくとも 3 つのワーカーノードが必要です。 vCPU コア 4 基、 RAM はそれぞれ 12GB です
ロードバランサ	動作環境クラスタ内のサービスに送信される入力トラフィックに使用できるサービスタイプ「 LoadBalancer 」
FQDN	Astra Control Center の FQDN をロードバランシング IP アドレスに指定する方法
Astra Trident （ **NetApp Cloud Manager** の **Kubernetes** クラスタ検出の一部としてインストール）	Trident 21.04 以降がインストールおよび設定され、 NetApp ONTAP バージョン 9.5 以降がストレージバックエンドとしてインストールされている必要があります

コンポーネント	要件
イメージレジストリ	Astra Control Center のビルドイメージをプッシュできる、 AWS Elastic Container Registry などの既存のプライベートレジストリが必要です。イメージをアップロードするイメージレジストリの URL を指定する必要があります。 Restic ベースのイメージを使用してアプリケーションをバックアップおよび復元するには、 Astra Control Center ホストクラスタと管理対象クラスタが同じイメージレジストリにアクセスできる必要があります。
Astra Trident / ONTAP 構成	Astra Control Center を使用するには、ストレージクラスを作成してデフォルトのストレージクラスとして設定する必要があります。Astra Control Center では、 Kubernetes クラスタを NetApp Cloud Manager にインポートする際に作成される次の ONTAP Kubernetes ストレージクラスがサポートされます。Astra Trident によって提供される機能は次のとおりです。 • 「 vsaworkingenvironment 」 -<> -ha - NAS csi.trident.netapp.io` • 「 vsaworkingenvironment 」 -<> -ha -san csi.trident.netapp.io` • 「 vsaworkingenvironment 」 -<>-singsingle-nas csi.trident.netapp.io` • 「 vsaworkingenvironment 」 -<>-singsingle-san csi.trident.netapp.io`

これらの要件は、運用環境で実行されている唯一のアプリケーションが Astra Control Center であることを前提としています。環境で追加のアプリケーションを実行している場合は、それに応じてこれらの最小要件を調整します。

AWS レジストリトークンは 12 時間で期限切れになり、その後 Docker イメージのレジストリシークレットを更新する必要があります。

AWS の導入の概要を参照してください

Cloud Volumes ONTAP をストレージバックエンドとして使用して Astra Control Center for AWS をインストールするプロセスの概要を以下に示します。

これらの各手順については、以下で詳しく説明します。

1. 十分な IAM 権限があることを確認します。
2. AWS に Red Hat OpenShift クラスタをインストールします。
3. AWS を設定します。
4. NetApp Cloud Manager を設定します。
5. Astra Control Center をインストールします。

十分な **IAM** 権限があることを確認します

Red Hat OpenShiftクラスタとNetApp Cloud Manager Connectorをインストールできる十分な数のIAMロールと権限があることを確認します。

を参照してください "AWS の初期クレデンシャル"。

AWS に **Red Hat OpenShift** クラスタをインストールします

AWS に Red Hat OpenShift Container Platform クラスタをインストールします。

インストール手順については、を参照してください "AWS で OpenShift Container Platform にクラスタをインストールします"。

AWS を設定します

次に、仮想ネットワークの作成、EC2コンピューティングインスタンスのセットアップ、AWS S3バケットの作成、Astra Control Centerイメージをホストする Elastic Container Register（ECR）の作成、このレジストリへのイメージのプッシュを行うようにAWSを設定します。

AWS のドキュメントに従って次の手順を実行します。を参照してください "AWS インストールドキュメント"。

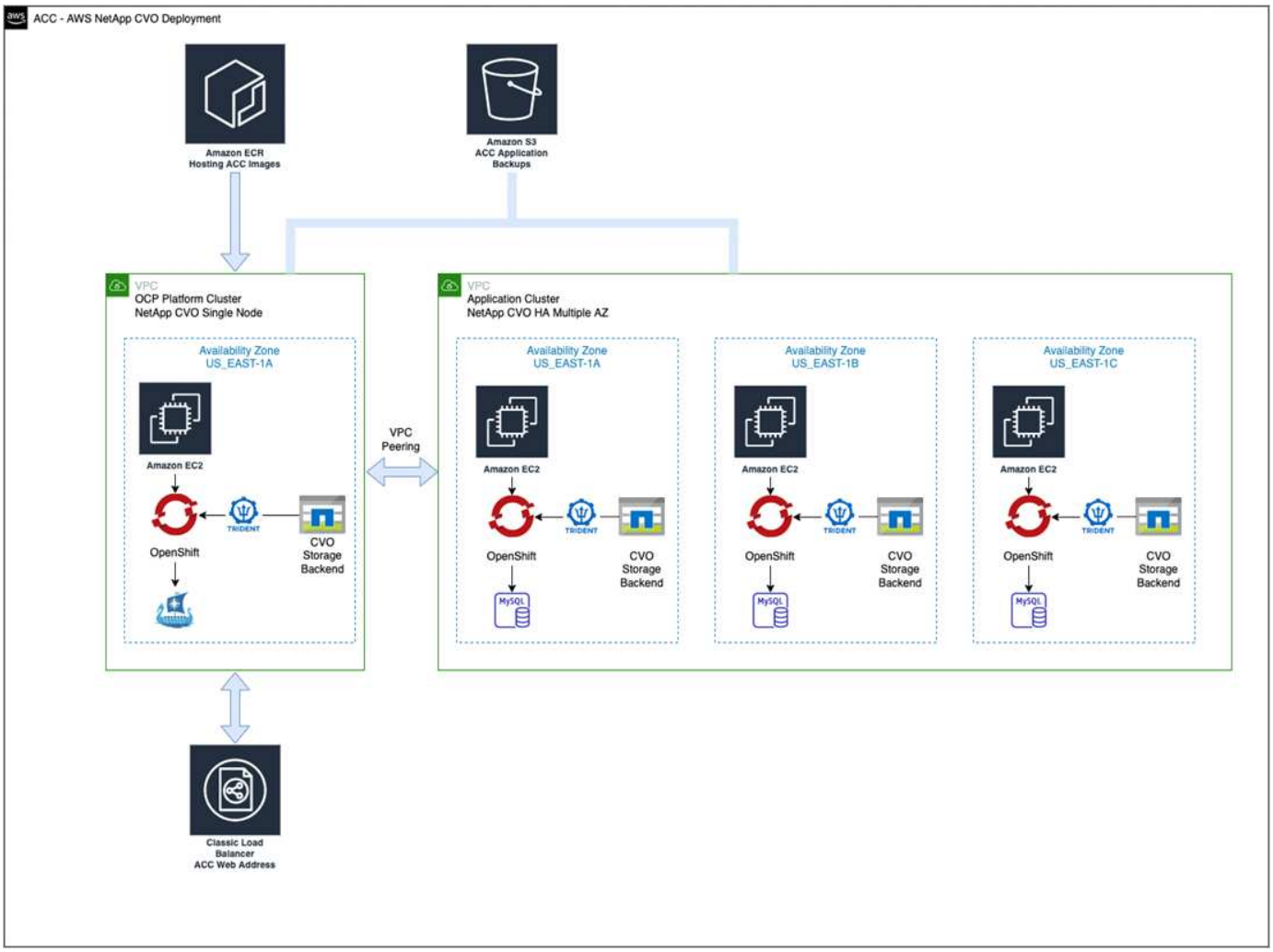
1. AWS仮想ネットワークを作成します。
2. EC2 コンピューティングインスタンスを確認します。AWS ではベアメタルサーバまたは VM を使用できます。
3. インスタンスタイプが、マスターノードとワーカーノードの Astra の最小リソース要件に一致していない場合は、 Astra の要件に合わせて AWS でインスタンスタイプを変更します。 を参照してください "Astra Control Center の要件"。
4. バックアップを格納する AWS S3 バケットを少なくとも 1 つ作成します。
5. すべての ACC イメージをホストする AWS Elastic Container Registry （ ECR ）を作成します。

> ECRを作成しないと、Astra Control Centerは、AWSバックエンドを持つCloud Volumes ONTAP を含むクラスタからモニタリングデータにアクセスできません。問題 は、 Astra Control Center を使用して検出および管理しようとしたクラスタに AWS ECR アクセスがない場合に発生します。

6. ACC イメージを定義済みのレジストリにプッシュします。

> AWS Elastic Container Registry （ ECR ）トークンの有効期限は 12 時間です。有効期限が切れたため、クラスタ間のクローニング処理が失敗します。この問題 は、AWS用に設定されたCloud Volumes ONTAP からストレージバックエンドを管理する場合に発生します。この問題 を修正するには、 ECR で再度認証を行い、クローン操作を再開するための新しいシークレットを生成します。

AWS 環境の例を次に示します。



NetApp Cloud Manager を設定します

Cloud Manager を使用して、ワークスペースの作成、 AWS へのコネクタの追加、作業環境の作成、クラスタのインポートを行います。

Cloud Manager のドキュメントに従って、次の手順を実行します。以下を参照してください。

- "AWS で Cloud Volumes ONTAP を使用するための準備"。
- "Cloud Manager を使用して AWS でコネクタを作成します"

手順

1. Cloud Manager にクレデンシャルを追加します。
2. ワークスペースを作成します。
3. AWS 用のコネクタを追加します。プロバイダとして AWS を選択します。
4. クラウド環境の作業環境を構築
    a. 場所：「 Amazon Web Services （ AWS ）」
    b. 「 Cloud Volumes ONTAP HA 」と入力します。
5. OpenShift クラスタをインポートします。作成した作業環境にクラスタが接続されます。

a. ネットアップクラスタの詳細を表示するには、 * K8s * > * Cluster list * > * Cluster Details * を選択します。

b. 右上隅に Trident のバージョンが表示されていることを確認します。

c. Cloud Volumes ONTAP クラスタのストレージクラスは、プロビジョニングツールとしてネットアップを使用していることに注目してください。

   これにより、 Red Hat OpenShift クラスタがインポートされ、デフォルトのストレージクラスに割り当てられます。ストレージクラスを選択します。Trident は、インポートと検出のプロセスの一環として自動的にインストールされます。

6. このCloud Volumes ONTAP 環境内のすべての永続ボリュームとボリュームをメモします。

> Cloud Volumes ONTAP は、シングルノードまたはハイアベイラビリティとして動作できます。HA が有効になっている場合は、 AWS で実行されている HA ステータスとノード導入ステータスを確認します。

Astra Control Center をインストールします

標準に従ってください "Astra Control Center のインストール手順"。

Microsoft Azure に Astra Control Center を導入

Microsoft Azure パブリッククラウドでホストされる自己管理型の Kubernetes クラスタに Astra Control Center を導入できます。

Azureに必要なもの

Azure に Astra Control Center を導入する前に、次のものが必要です。

- Astra Control Center ライセンス。を参照してください "Astra Control Center のライセンス要件"。
- "Astra Control Center の要件を満たす"。
- NetApp Cloud Central アカウント
- Red Hat OpenShift Container Platform （ OCP ） 4.8 の場合
- Red Hat OpenShift Container Platform （ OCP ）権限（ポッドを作成するためのネームスペースレベル）
- バケットとコネクタの作成を可能にする権限を持つ Azure クレデンシャル

Azure の運用環境の要件

Astra Control Center をホストするオペレーティングシステムが、環境の公式ドキュメントに記載されている基本的なリソース要件を満たしていることを確認します。

Astra Control Center では、環境のリソース要件に加え、次のリソースが必要です。

を参照してください "Astra Control Center の運用環境要件"。

コンポーネント	要件
バックエンドの **NetApp Cloud Volumes ONTAP** ストレージ容量	300GB 以上のデータがあります
ワーカーノード（**Azure** コンピューティング要件）	少なくとも 3 つのワーカーノードが必要です。 vCPU コア 4 基、 RAM はそれぞれ 12GB です
ロードバランサ	動作環境クラスタ内のサービスに送信される入力トラフィックに使用できるサービスタイプ「 LoadBalancer 」
FQDN （**Azure DNS** ゾーン）	Astra Control Center の FQDN をロードバランシング IP アドレスに指定する方法
Astra Trident （**NetApp Cloud Manager** の **Kubernetes** クラスタ検出の一部としてインストール）	Trident 21.04 以降がインストールおよび設定され、 NetApp ONTAP バージョン 9.5 以降がストレージバックエンドとして使用されます
イメージレジストリ	Astra Control Center ビルドイメージをプッシュできる、 Azure Container Registry （ACR ）などの既存のプライベートレジストリが必要です。イメージをアップロードするイメージレジストリの URL を指定する必要があります。 バックアップ用にリストアイメージを取得するには、匿名アクセスを有効にする必要があります。
Astra Trident / ONTAP 構成	Astra Control Center を使用するには、ストレージクラスを作成してデフォルトのストレージクラスとして設定する必要があります。Astra Control Center では、 Kubernetes クラスタを NetApp Cloud Manager にインポートする際に作成される次の ONTAP Kubernetes ストレージクラスがサポートされます。Astra Trident によって提供される機能は次のとおりです。 • 「 vsaworkingenvironment 」 -<> -ha - NAS csi.trident.netapp.io` • 「 vsaworkingenvironment 」 -<> -ha -san csi.trident.netapp.io` • 「 vsaworkingenvironment 」 -<>-singsingle-nas csi.trident.netapp.io` • 「 vsaworkingenvironment 」 -<>-singsingle-san csi.trident.netapp.io`

これらの要件は、運用環境で実行されている唯一のアプリケーションが Astra Control Center であることを前提としています。環境で追加のアプリケーションを実行している場合は、それに応じてこれらの最小要件を調整します。

Azure の導入の概要

ここでは、 Astra Control Center for Azure のインストールプロセスの概要を示します。

これらの各手順については、以下で詳しく説明します。

1. Azure に Red Hat OpenShift クラスタをインストールします。
2. Azure リソースグループを作成する。
3. 十分な IAM 権限があることを確認します。
4. Azure を設定。
5. NetApp Cloud Manager を設定します。
6. Astra Control Center をインストールして設定します。

Azure に Red Hat OpenShift クラスタをインストールします

まず、 Azure に Red Hat OpenShift クラスタをインストールします。

インストール手順については、のRedHatのマニュアルを参照してください "AzureにOpenShiftクラスタをインストールしています" および "Azureアカウントをインストールしています"。

Azure リソースグループを作成する

Azure リソースグループを少なくとも 1 つ作成します。

> OpenShift では、独自のリソースグループを作成できます。さらに、 Azure リソースグループも定義する必要があります。OpenShift のドキュメントを参照してください。

プラットフォームクラスタリソースグループおよびターゲットアプリケーション OpenShift クラスタリソースグループを作成できます。

十分な IAM 権限があることを確認します

Red Hat OpenShiftクラスタとNetApp Cloud Manager Connectorをインストールできる十分な数のIAMロールと権限があることを確認します。

を参照してください "Azure のクレデンシャルと権限"。

Azure を設定

次に、仮想ネットワークの作成、コンピューティングインスタンスのセットアップ、Azure Blobコンテナの作成、Astra Control Centerイメージをホストする Azure Container Register（ACR）の作成、このレジストリへのイメージのプッシュを行うようにAzureを設定します。

Azure のドキュメントに従って、次の手順を実行します。を参照してください "Azure への OpenShift クラスタのインストール"。

1. Azure Virtual Networkの作成
2. コンピューティングインスタンスを確認します。Azure の場合、ベアメタルサーバまたは VM を使用できます。
3. インスタンスタイプがまだマスターノードとワーカーノードの Astra 最小リソース要件に一致していない場合は、 Azure でインスタンスタイプを変更して Astra の要件を満たします。 を参照してください "Astra Control Center の要件"。
4. バックアップを格納するAzure BLOBコンテナを少なくとも1つ作成します。

5. ストレージアカウントを作成します。Astra Control Center でバケットとして使用するコンテナを作成するには、ストレージアカウントが必要です。
6. バケットへのアクセスに必要なシークレットを作成します。
7. Azure Container Registry （ACR ）を作成して、すべての Astra Control Center イメージをホストします。
8. ACR アクセスを設定して Docker プッシュ / プルをすべての Astra Control Center イメージに適用します。
9. 次のスクリプトを入力して、 ACC イメージをこのレジストリにプッシュします。

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

   - 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. DNS ゾーンを設定します。

NetApp Cloud Manager を設定します

Cloud Manager を使用して、ワークスペースの作成、 Azure へのコネクタの追加、作業環境の作成、クラスタのインポートを行います。

Cloud Manager のドキュメントに従って、次の手順を実行します。を参照してください "Azure で Cloud Manager を使用する準備をしています"。

必要なもの

必要な IAM 権限とロールを持つ Azure アカウントにアクセスします

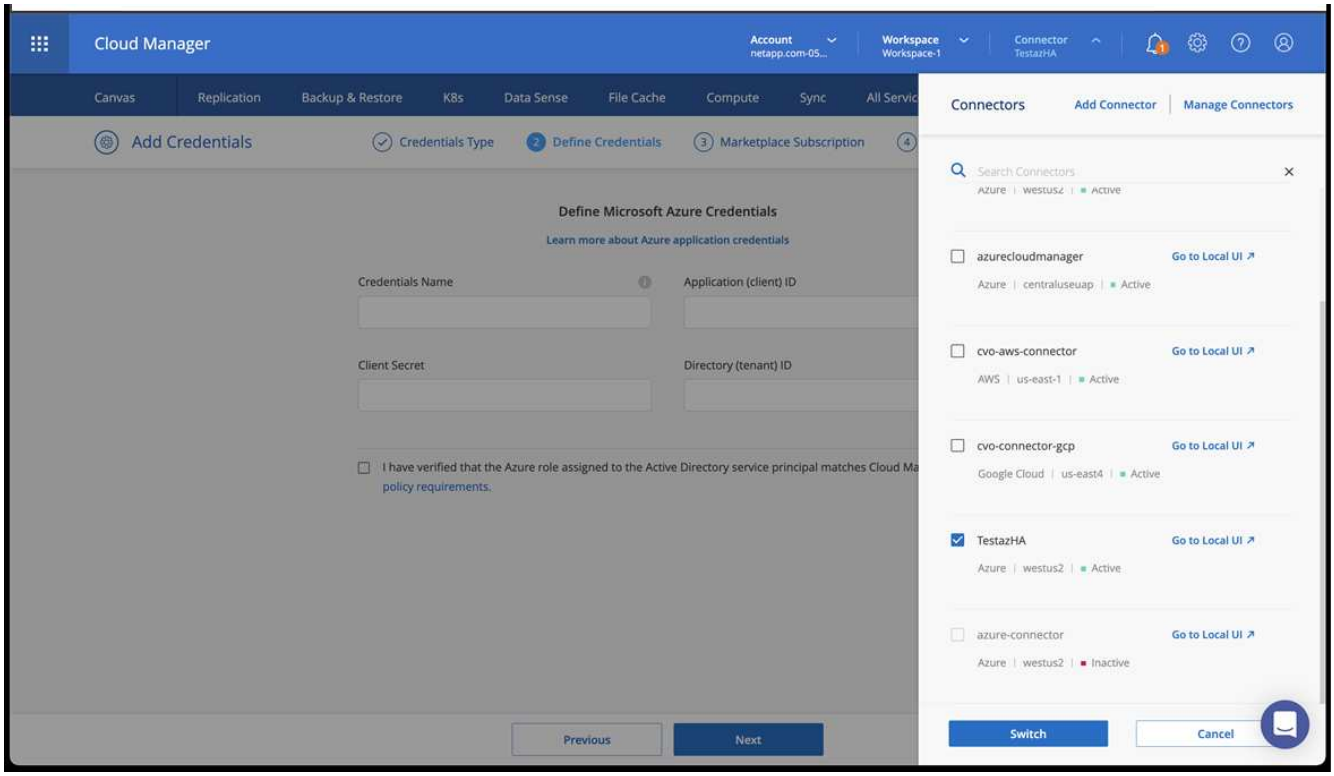
手順

1. Cloud Manager にクレデンシャルを追加します。
2. Azure 用のコネクタを追加します。を参照してください "Cloud Manager のポリシー"。
   a. プロバイダとして「 * Azure * 」を選択します。

b. アプリケーション ID 、クライアントシークレット、ディレクトリ（テナント） ID など、 Azure クレデンシャルを入力します。

を参照してください "Cloud Manager から Azure にコネクタを作成する"。

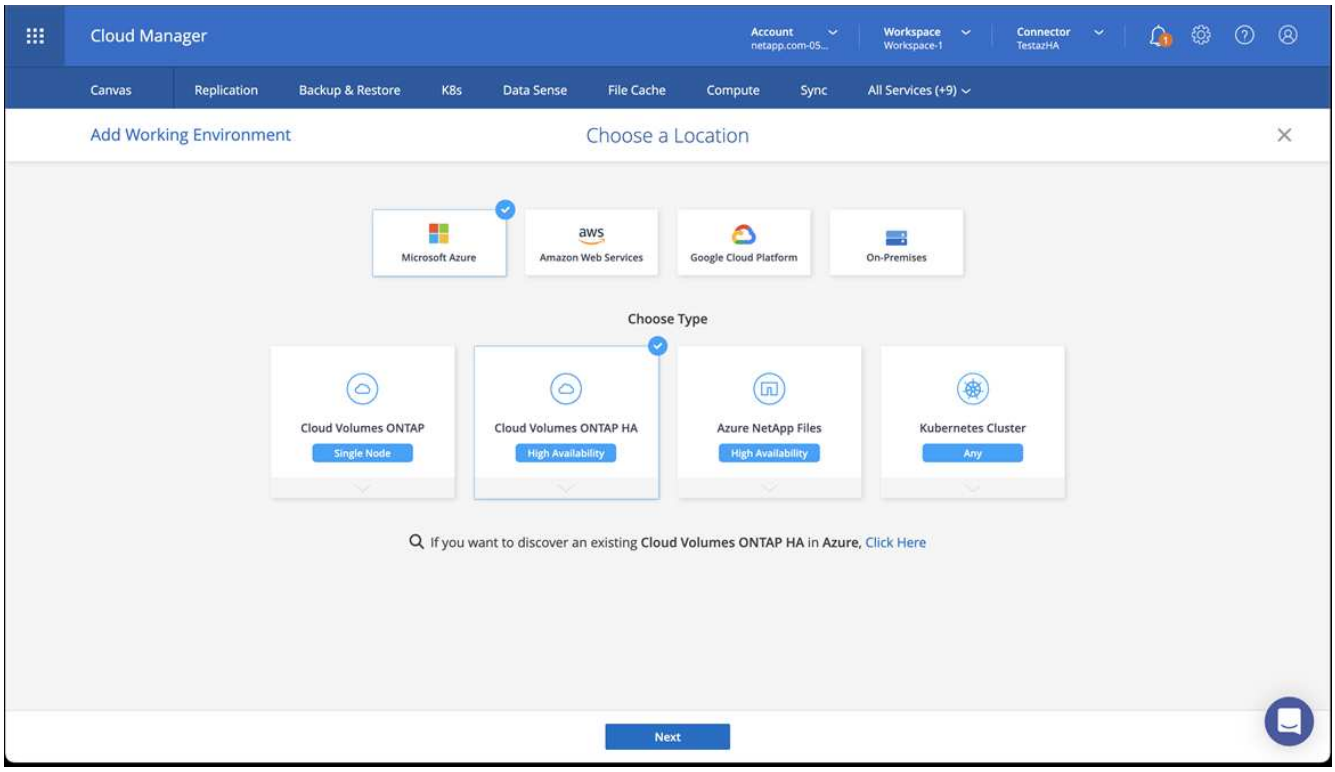
3. コネクタが動作していることを確認し、コネクタに切り替えます。



4. クラウド環境の作業環境を構築

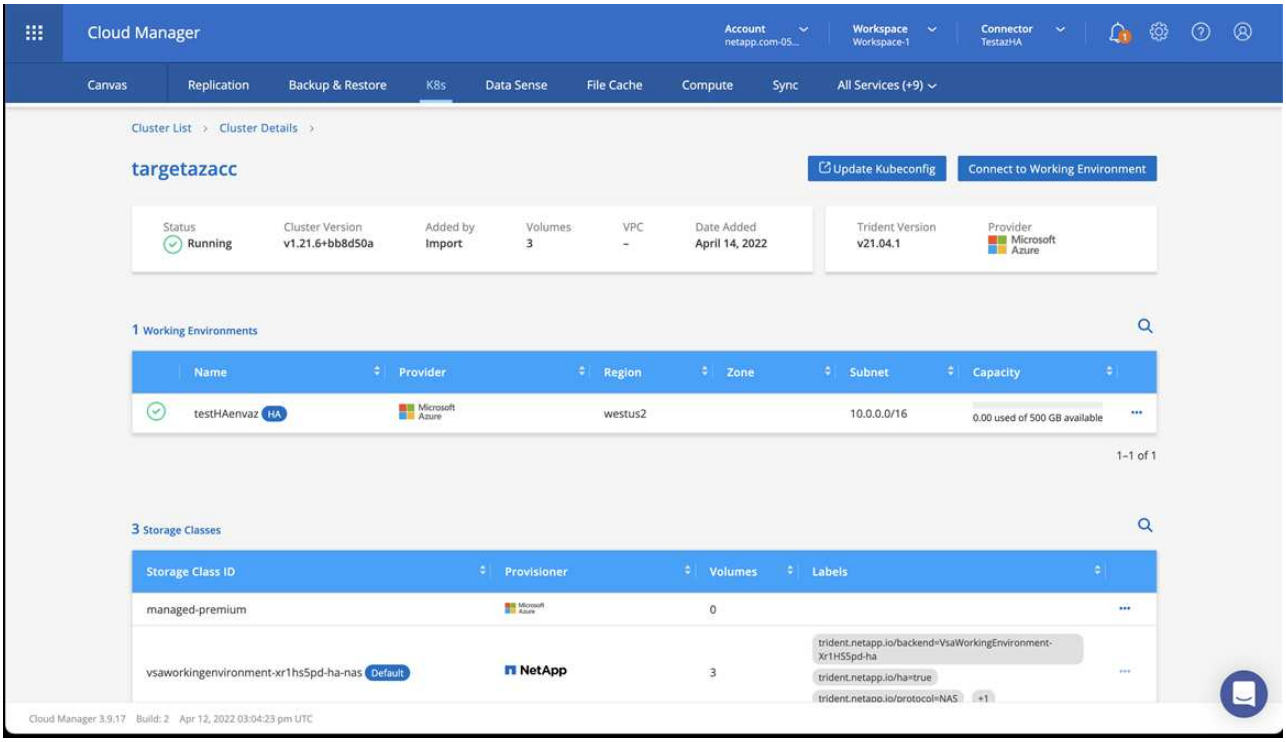
a. 場所：「 Microsoft Azure 」。

b. 「 Cloud Volumes ONTAP HA 」と入力します。



5. OpenShift クラスタをインポートします。作成した作業環境にクラスタが接続されます。

   a. ネットアップクラスタの詳細を表示するには、 * K8s * > * Cluster list * > * Cluster Details * を選択します。



   b. 右上隅に Trident のバージョンが表示されていることを確認します。

   c. Cloud Volumes ONTAP クラスタのストレージクラスは、プロビジョニングツールとしてネットアップを使用していることに注目してください。

これにより、 Red Hat OpenShift クラスタがインポートされ、デフォルトのストレージクラスが割り当てられます。ストレージクラスを選択します。Trident は、インポートと検出のプロセスの一環として自動的にインストールされます。

6. このCloud Volumes ONTAP 環境内のすべての永続ボリュームとボリュームをメモします。
7. Cloud Volumes ONTAP は、シングルノードまたはハイアベイラビリティとして動作できます。HA が有効になっている場合は、 Azure で実行されている HA ステータスとノード導入ステータスを確認します。

Astra Control Center をインストールして設定します

Astra Control Center を標準でインストールします "インストール手順"。

Astra Control Center を使用して、 Azure バケットを追加する。を参照してください "Astra Control Center をセットアップし、バケットを追加する"。