



# インストールの概要

## Astra Control Center

NetApp  
August 11, 2025

# 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
Astra Control Centerをダウンロードして展開します	3
ローカルレジストリを使用する場合は、追加の手順を実行します。	4
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	8
Astra Control Center オペレータを設置します	9
Astra Control Center を設定します	12
Astra Control Center とオペレータのインストールを完了します	26
システムステータスを確認します	27
ロードバランシング用の入力を設定します	34
Astra Control Center UI にログインします	38
インストールのトラブルシューティングを行います	38
別のインストール手順	39
次のステップ	39
外部証明書マネージャを設定します	39
OpenShift OperatorHub を使用して Astra Control Center をインストールします	41
Astra Control Centerをダウンロードして展開します	43
ローカルレジストリを使用する場合は、追加の手順を実行します。	44
オペレータインストールページを検索します	48
オペレータをインストールします	50
Astra Control Center をインストールします	50
レジストリシークレットを作成します	52
次のステップ	52
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	52
Amazon Web Services に Astra Control Center を導入	53
Astra Control CenterをGoogle Cloud Platformに導入	56
Microsoft Azure に Astra Control Center を導入	60
インストール後にAstra Control Centerを設定します	64
リソースの制限を解除します	65
カスタム TLS 証明書を追加します	66

# 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"

環境によっては、Astra Control Centerのインストール後に追加の設定が必要になる場合があります。

- "インストール後にAstra Control Centerを設定します"

## 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、インストールイメージをダウンロードして次の手順を実行します。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。

Astra Control Centerのインストールプロセスのデモについては、を参照してください "[このビデオでは](#)"。

作業を開始する前に

- 環境条件を満たしている： "[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](#)"。



3つ目の障害ドメインまたはセカンダリサイトにAstra Control Centerを導入これは、アプリケーションのレプリケーションとシームレスなディザスタリカバリに推奨されます。

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

```
kubectl get apiservices
```

- \*ルーティング可能なFQDN\*：使用するAstra FQDNをクラスタにルーティングできることを確認します。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。
- 証明書マネージャの設定:クラスタに証明書マネージャがすでに存在する場合は、一部の証明書マネージャを実行する必要があります。 "[事前に必要な手順](#)" そのため、Astra Control Centerは独自の証明書マネージャのインストールを試みません。デフォルトでは、Astra Control Centerはインストール時に独自の証明書マネージャをインストールします。
- (ONTAP SANドライバのみ) マルチパスの有効化：ONTAP SANドライバを使用している場合は、すべてのKubernetesクラスタでマルチパスが有効になっていることを確認してください。

また、次の点も考慮する必要があります。

- \* NetApp Astra Controlイメージレジストリへのアクセス\*：

Astra Control Provisionerなど、Astra Controlのインストールイメージや機能強化された機能をNetAppイメージレジストリから取得することができます。

a. レジストリへのログインに必要なAstra ControlアカウントIDを記録します。

アカウントIDはAstra Control Service Web UIで確認できます。ページ右上の図アイコンを選択し、\*APIアクセス\*を選択して、アカウントIDを書き留めます。

b. 同じページから\*APIトークンの生成\*を選択し、APIトークン文字列をクリップボードにコピーしてエディターに保存します。

c. Astra Controlレジストリにログインします。

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

- セキュアな通信のためのサービスメッシュをインストール：Astra Controlホストクラスタの通信チャンネルは、["サポートされるサービスメッシュ"](#)。



Astra Control Centerとサービスメッシュの統合は、Astra Control Centerでのみ実行可能 ["インストール"](#)。そしてこのプロセスから独立していません。メッシュ環境から非メッシュ環境に戻すことはサポートされていません。

Istioサービスメッシュを使用するには、次の手順を実行する必要があります。

- を追加します。 `istio-injection:enabled` [ラベル](#) にアクセスしてからAstra Control Centerを導入する必要があります。
- を使用します `Generic` [入力設定](#) 別のインGRESSを提供します。 [外部ロードバランシング](#)。
- Red Hat OpenShiftクラスタの場合は、 `NetworkAttachmentDefinition` 関連付けられているすべてのAstra Control Center名前空間 (`netapp-acc-operator`、`netapp-acc`、`netapp-monitoring` アプリケーションクラスタの場合、または置換されたカスタム名前空間の場合)。

```
cat <<EOF | oc -n netapp-acc-operator create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-acc create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-monitoring create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF
```

## 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](#)をダウンロードして展開します
- [ローカルレジストリを使用する場合は、追加の手順を実行します。]
- [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オペレータ (たとえば、`kubectl delete -f astra_control_center_operator_deploy.yaml`) Astra Control Centerのインストール中または操作中はいつでも、ポッドを削除しないようにします。

## Astra Control Centerをダウンロードして展開します

次のいずれかの場所からAstra Control Centerのイメージをダウンロードします。

- \* Astra Controlサービスのイメージレジストリ\*：Astra Control Centerのイメージでローカルレジストリを使用しない場合や、NetApp Support Siteからバンドルをダウンロードするよりもこの方法を使用する場合は、このオプションを使用します。
- \* NetApp Support Site \*：このオプションは、Astra Control Centerのイメージでローカルレジストリを使用する場合に使用します。

### Astra Controlイメージレジストリ

1. Astra Control Serviceにログインします。
2. ダッシュボードで、\*[Deploy a self-managed instance of Astra Control]\*を選択します。
3. 手順に従ってAstra Controlイメージのレジストリにログインし、Astra Control Centerのインストールイメージを取得してイメージを展開します。

### NetApp Support Site

1. Astra Control Centerを含むバンドルをダウンロードします (astra-control-center-[version].tar.gz) をクリックします "[Astra Control Centerのダウンロードページ](#)"。
2. (推奨ですがオプション) Astra Control Centerの証明書と署名のバンドルをダウンロードします (astra-control-center-certs-[version].tar.gz) をクリックして、バンドルのシグネチャを確認します。

```
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
```

出力にはと表示されます Verified OK 検証が成功したあとに、

3. Astra Control Centerバンドルからイメージを抽出します。

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

ローカルレジストリを使用する場合は、追加の手順を実行します。

Astra Control Centerバンドルをローカルのレジストリにプッシュする場合は、NetApp Astra kubectlコマンドラインプラグインを使用する必要があります。

ネットアップAstra kubectlプラグインをインストール

次の手順を実行して、最新のNetApp Astra kubectlコマンドラインプラグインをインストールします。

作業を開始する前に

ネットアップでは、CPUアーキテクチャやオペレーティングシステム別にプラグインのバイナリを提供して

います。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。

以前のインストールからプラグインがインストールされている場合は、"[最新バージョンがインストールされていることを確認してください](#)" これらの手順を実行する前に。

#### 手順

1. 使用可能なNetApp Astra kubectlプラグインのバイナリを一覧表示します。



kubectlプラグインライブラリはtarバンドルの一部であり、フォルダに解凍されます kubectl-astra。

```
ls kubectl-astra/
```

2. 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を現在のパスに移動し、次の名前に変更します。 kubectl-astra :

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

#### イメージをレジストリに追加する

1. Astra Control Centerバンドルをローカルのレジストリにプッシュする場合は、コンテナエンジンに応じた手順を実行します。

## Docker です

- a. tarballのルートディレクトリに移動します。次のように表示されます。  
acc.manifest.bundle.yaml ファイルと次のディレクトリ：

```
acc/  
kubect1-astra/  
acc.manifest.bundle.yaml
```

- b. Astra Control Centerのイメージディレクトリにあるパッケージイメージをローカルレジストリにプッシュします。を実行する前に、次の置換を行ってください push-images コマンドを実行します

- <BUNDLE\_FILE> をAstra Controlバンドルファイルの名前に置き換えます (acc.manifest.bundle.yaml)。
- &lt;MY\_FULL\_REGISTRY\_PATH&gt; をDockerリポジトリのURLに置き換えます。次に例を示します。 "<a href="https://&lt;docker-registry&gt;" class="bare">https://&lt;docker-registry&gt;"</a>。
- <MY\_REGISTRY\_USER> をユーザ名に置き換えます。
- <MY\_REGISTRY\_TOKEN> をレジストリの認証済みトークンに置き換えます。

```
kubect1 astra packages push-images -m <BUNDLE_FILE> -r  
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p  
<MY_REGISTRY_TOKEN>
```

## ポドマン

- a. tarballのルートディレクトリに移動します。次のファイルとディレクトリが表示されます。

```
acc/  
kubect1-astra/  
acc.manifest.bundle.yaml
```

- b. レジストリにログインします。

```
podman login <YOUR_REGISTRY>
```

- c. 使用するPodmanのバージョンに合わせてカスタマイズされた次のいずれかのスクリプトを準備して実行します。<MY\_FULL\_REGISTRY\_PATH> を'サブディレクトリを含むリポジトリのURLに置き換えます

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

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
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
```

**Podman 3**

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
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
```



レジストリ設定に応じて、スクリプトが作成するイメージパスは次のようになります。

```
https://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

## 2. ディレクトリを変更します。

```
cd manifests
```

## 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. Astra Control Centerホストクラスタのkubeconfigをエクスポートします。

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



インストールを完了する前に、Astra Control Centerをインストールするクラスタをkubeconfigで指定していることを確認してください。

2. 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. を作成します netapp-acc-operator ネームスペース：

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

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



プレースホルダ `your_registry_path` 以前にアップロードした画像の場所と一致する必要があります（例： `[Registry_URL]/netapp/astra/astracc/24.02.0-69`）。

```
kubectl create secret docker-registry astra-registry-cred -n netapp-acc-operator --docker-server=cr.astra.netapp.io --docker-username=[astra_account_id] --docker-password=[astra_api_token]
```

+

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

+



シークレットの生成後にネームスペースを削除した場合は、ネームスペースを再作成し、ネームスペースのシークレットを再生成します。

- a. を作成します netapp-acc（またはカスタム名）ネームスペース。

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

- b. のシークレットを作成します netapp-acc（またはカスタム名）ネームスペース。Docker情報を追

加し、レジストリの設定に応じて適切なコマンドのいずれかを実行します。

```
kubectl create secret docker-registry astra-registry-cred -n [netapp-acc or custom namespace] --docker-server=cr.astra.netapp.io --docker-username=[astra_account_id] --docker-password=[astra_api_token]
```

```
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]
```

## Astra Control Center オペレータを設置します

1. (ローカルレジストリのみ) ローカルレジストリを使用している場合は、次の手順を実行します。
  - a. Astra Control Centerオペレータによる導入YAMLを開きます。

```
vim astra_control_center_operator_deploy.yaml
```



注釈付きサンプルYAMLは以下の手順に従います。

- b. 認証が必要なレジストリを使用する場合は、のデフォルト行を置き換えます imagePullSecrets:  
[] 次の条件を満たす場合：

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

- c. 変更 ASTRA\_IMAGE\_REGISTRY をクリックします kube-rbac-proxy でイメージをプッシュしたレジストリパスへのイメージ [前の手順](#)。
- d. 変更 ASTRA\_IMAGE\_REGISTRY をクリックします acc-operator-controller-manager でイメージをプッシュしたレジストリパスへのイメージ [前の手順](#)。

```
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: ASTRA_IMAGE_REGISTRY/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: ASTRA_IMAGE_REGISTRY/acc-operator:24.02.68
        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
```

## 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
```

回答例を表示するには展開します。

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.as
tra.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
```

## 3. ポッドが実行中であることを確認します

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

## Astra Control Center を設定します

1. Astra Control Centerカスタムリソース（CR）ファイルを編集します (astra\_control\_center.yaml) アカウント、サポート、レジストリ、およびその他の必要な設定を行うには、次の手順を実行します。

```
vim astra_control_center.yaml
```



注釈付きサンプルYAMLは以下の手順に従います。

2. 次の設定を変更または確認します。

### アカウント名

設定	ガイダンス（Guidance）	を入力します	例
accountName	を変更します accountName stringには、Astra Control Centerアカウントに関連付ける名前を指定します。アカウント名は1つだけです。	文字列	Example

### astraVersion

設定	ガイダンス（Guidance）	を入力します	例
astraVersion	導入するAstra Control Centerのバージョン。この設定には値があらかじめ入力されているため、対処は不要です。	文字列	24.02.0-69

## astraitAddress

設定	ガイダンス (Guidance)	を入力します	例
astraAddress	<p>を変更します</p> <p>astraAddress ブラウザで使用するFQDN (推奨) またはIPアドレスを指定して、Astra Control Centerにアクセスします。このアドレスは、データセンターでAstra Control Centerがどのように検出されるかを定義します。このアドレスは、完了時にロードバランサからプロビジョニングしたFQDNまたはIPアドレスと同じです "<a href="#">Astra Control Center の要件</a>"。</p> <p>注：は使用しないでください <a href="#">http://</a> または <a href="#">https://</a> をクリックします。この FQDN をコピーしてで使用します <a href="#">後の手順</a>。</p>	文字列	astra.example.com

## AutoSupport

このセクションで選択した内容によって、NetAppのプロアクティブサポートアプリケーション、デジタルアドバイザー、およびデータの送信先が決まります。インターネット接続が必要です（ポート442）。サポートデータはすべて匿名化されます。

設定	使用	ガイダンス (Guidance)	を入力します	例
<code>autoSupport.enrolled</code>	または <code>enrolled</code> または <code>url</code> フィールドを選択する必要があります	変更 <code>enrolled</code> を選択します AutoSupport <code>false</code> インターネットに接続されていないか、または保持されているサイト <code>true</code> 接続されているサイト 用の設定 <code>true</code> 匿名データをネットアップに送信し、サポートを目的として使用できるようにします。 デフォルトの選択は <code>false</code> およびは、サポートデータがネットアップに送信されないことを示します。	ブール値	<code>false</code> (デフォルト値)
<code>autoSupport.url</code>	または <code>enrolled</code> または <code>url</code> フィールドを選択する必要があります	このURLは匿名データの送信先を決定します。	文字列	<a href="https://support.netapp.com/asupprod/post/1.0/postAsup">https://support.netapp.com/asupprod/post/1.0/postAsup</a>

## E メール

設定	ガイダンス (Guidance)	を入力します	例
email	を変更します email デフォルトの初期管理者アドレスを表す文字列。この E メールアドレスをコピーしてで使用します <a href="#">後の手順</a> 。この E メールアドレスは、最初のアカウントが UI にログインする際のユーザー名として使用され、Astra Control のイベントが通知されます。	文字列	admin@example.com

## FirstName

設定	ガイダンス (Guidance)	を入力します	例
firstName	アストラアカウントに関連付けられている初期管理者の名前。ここで使用した名前は、初回ログイン後に UI の見出しに表示されます。	文字列	SRE

## 姓

設定	ガイダンス (Guidance)	を入力します	例
lastName	アストラアカウントに関連付けられている初期管理者の姓です。ここで使用した名前は、初回ログイン後に UI の見出しに表示されません。	文字列	Admin

## imageRegistryのこと

このセクションで選択すると、Astraアプリケーションイメージ、Astra Control Center Operator、Astra Control Center Helmリポジトリをホストするコンテナイメージレジストリが定義されます。

設定	使用	ガイダンス (Guidance)	を入力します	例
imageRegistry.name	必須	Astra Control Centerの導入に必要なすべてのイメージをホストするAstra Controlイメージレジストリの名前。値は事前に入力されます。ローカルレジストリを設定しないかぎり、アクションは必要ありません。ローカルレジストリの場合は、この既存の値を、 <a href="#">前の手順</a> 。使用しないでください http:// または https:// をレジストリ名に追加します。	文字列	cr.astra.netapp.io (デフォルト) example.registry.com/astra (ローカルレジストリの例)

設定	使用	ガイダンス (Guidance)	を入力します	例
imageRegistry. secret	任意。	<p>イメージレジストリでの認証に使用するKubernetesシークレットの名前。値は事前に入力されます。ローカルレジストリを設定し、でそのレジストリに入力した文字列を設定しない限り、アクションは必要ありません。</p> <p>imageRegistry.name シークレットが必要です。</p> <p><b>重要：</b> 認証を必要としないローカルレジストリを使用している場合は、これを削除する必要があります。</p> <p>secret ラインの内側</p> <p>imageRegistry または、インストールが失敗します。</p>	文字列	astra-registry-cred

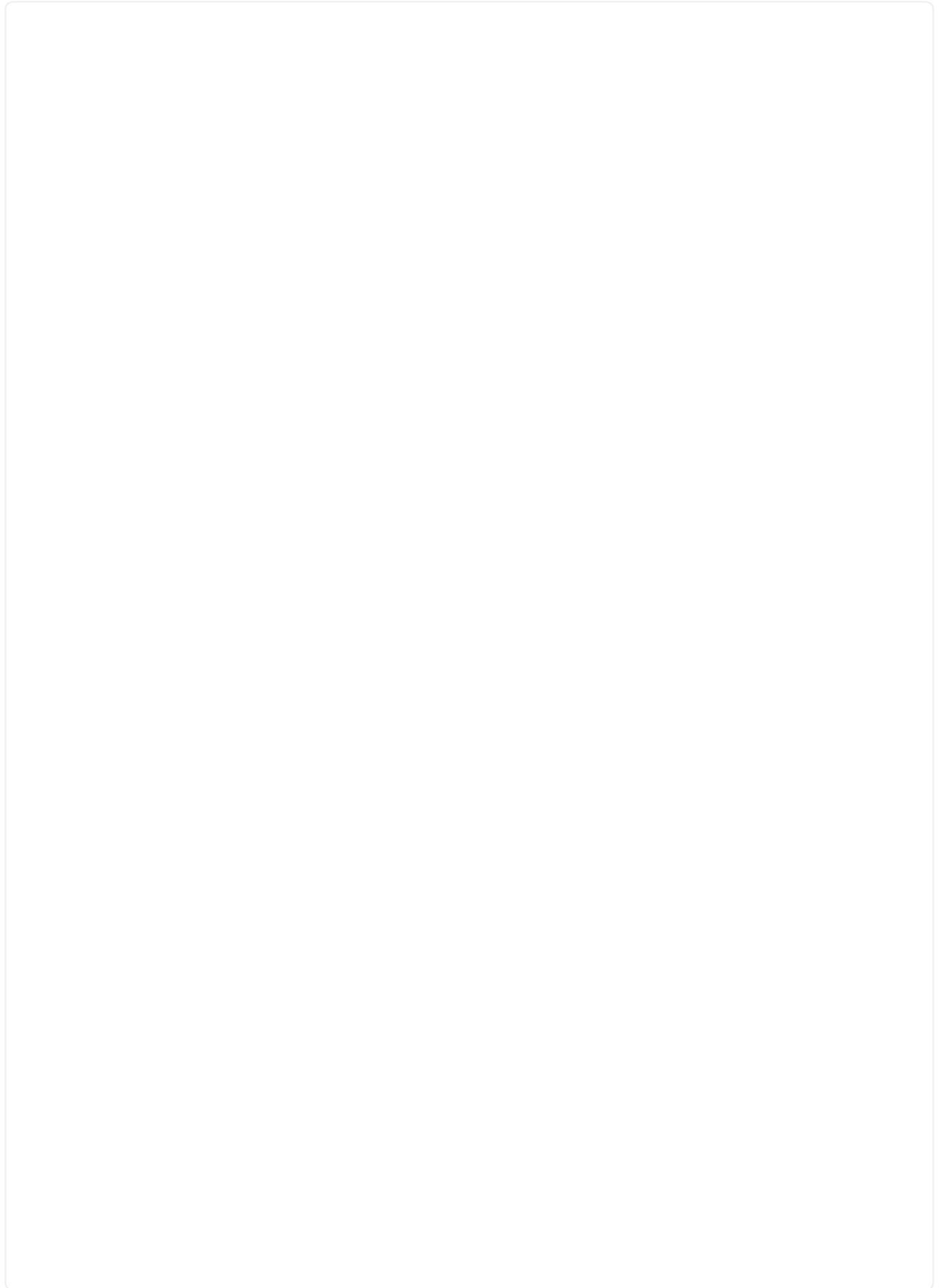
## ストレージクラス

設定	ガイダンス (Guidance)	を入力します	例
storageClass	<p>を変更します</p> <p>storageClass からの値 ontap-gold インストールで必要な別の storageClass リソースに移動します。コマンドを実行します</p> <p>kubect1 get sc をクリックして、設定済みの既存のストレージクラスを確認します。Astra Control Provisioner で設定されたストレージクラスのいずれかをマニフェストファイルに入力する必要があります。</p> <p>(astra-control-center- &lt;version&gt;.manifest) とを Astra PVS に使用します。設定されていない場合は、デフォルトのストレージクラスが使用されます。</p> <p>メモ：デフォルトのストレージクラスが設定されている場合は、デフォルトのアノテーションが設定されている唯一のストレージクラスであることを確認してください。</p>	文字列	ontap-gold

volumeReclaimPolicyのように指定します

設定	ガイダンス (Guidance)	を入力します	オプション (Options)
volumeReclaimPolicy	これにより、AstraのPVSの再利用ポリシーが設定されます。このポリシーをに設定しています Retain Astraが削除されたあとに永続的なボリュームを保持このポリシーをに設定しています Delete Astraが削除されたあとに永続的ボリュームを削除する。この値が設定されていない場合、PVSは保持されま	文字列	<ul style="list-style-type: none"><li>• Retain (デフォルト値)</li><li>• Delete</li></ul>

**ingressType**





設定	ガイダンス (Guidance)	を入力します	オプション (Options)
ingressType	<p>次の入力タイプのいずれかを使用します。</p> <p>汎用 (ingressType: "Generic") (デフォルト)</p> <p>このオプションは、別の入力コントローラを使用している場合、または独自の入力コントローラを使用する場合に使用します。Astra Control Centerを導入したら、<a href="#">"入力コントローラ"</a> URLを使用してAstra Control Centerを公開します。</p> <p>重要：Astra Control Centerでサービスメッシュを使用する場合は、Generic 入力タイプとして入力し、独自の設定を行います。<a href="#">"入力コントローラ"</a>。</p> <p>* AccTraefik * (ingressType: "AccTraefik") 入力コントローラを設定しない場合は、このオプションを使用します。これにより、Astra Control Centerが導入されます traefik Gateway as a Kubernetes LoadBalancer type serviceの略。</p> <p>Astra Control Centerは、タイプ「LoadBalancer」のサービスを使用します。(svc/traefik Astra Control Centerの名前空間) で、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。お使用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</p>	文字列	<ul style="list-style-type: none"> <li>• Generic (デフォルト値)</li> <li>• AccTraefik</li> </ul>

## スケールサイズ

設定	ガイダンス (Guidance)	を入力します	オプション (Options)
scaleSize	<p>デフォルトでは、Astraで高可用性 (HA) が使用されます。</p> <p>scaleSize の Medium`ほとんどのサービスをHAに導入し、冗長性を確保するために複数のレプリカを導入します。を使用`scaleSize として Small`Astraは、消費量を削減するための必須サービスを除き、すべてのサービスのレプリカ数を削減します。</p> <p>ヒント： `Medium 環境は約100個のポッドで構成されています (一時的なワークロードは含まれません)。100個のポッドは、3つのマスターノードと3つのワーカーノード構成に基づいています)。特にディザスタリカバリのシナリオを検討する場合は、環境で問題となる可能性があるポッド単位のネットワーク制限に注意してください。</p>	文字列	<ul style="list-style-type: none"> <li>• Small</li> <li>• Medium (デフォルト値)</li> </ul>

## astraitcesScaler

設定	ガイダンス (Guidance)	を入力します	オプション (Options)
astraResourcesScaler	<p>AstraeControlCenterリソース制限のスケールリングオプションデフォルトでは、Astra Control CenterはAstra内のほとんどのコンポーネントに対してリソース要求を設定して展開します。この構成により、アプリケーションの負荷と拡張性が高い環境では、Astra Control Centerソフトウェアスタックのパフォーマンスが向上します。ただし、小規模な開発またはテストクラスタを使用するシナリオでは、CRフィールドを使用します</p> <p>astraResourcesScaler に設定できます Off。これにより、リソース要求が無効になり、小規模なクラスタへの導入が可能になります。</p>	文字列	<ul style="list-style-type: none"><li>• Default (デフォルト値)</li><li>• Off</li></ul>

### 追加値



インストール時に既知の問題が表示されないように、Astra Control CenterのCRに次の値を追加します。

```
additionalValues:  
  keycloak-operator:  
    livenessProbe:  
      initialDelaySeconds: 180  
    readinessProbe:  
      initialDelaySeconds: 180
```

## CRD

このセクションで選択した内容によって、Astra Control CenterでのCRDの処理方法が決まります。

設定	ガイダンス (Guidance)	を入力します	例
<code>crds.externalCertManager</code>	外部証明書マネージャを使用する場合は、変更します externalCertManager 終了: true。デフォルト false Astra Control Centerが、インストール時に独自の証明書マネージャCRDをインストールするようにします。SSDはクラスタ全体のオブジェクトであり、クラスタの他の部分に影響を及ぼす可能性があります。このフラグを使用すると、これらのCRDがAstra Control Centerの外部にあるクラスタ管理者によってインストールおよび管理されることをAstra Control Centerに伝えることができます。	ブール値	False (デフォルト値)
<code>crds.externalTraefik</code>	デフォルトでは、Astra Control Centerは必要なTraefik CRDをインストールします。SSDはクラスタ全体のオブジェクトであり、クラスタの他の部分に影響を及ぼす可能性があります。このフラグを使用すると、これらのCRDがAstra Control Centerの外部にあるクラスタ管理者によってインストールおよび管理されることをAstra Control Centerに伝えることができます。	ブール値	False (デフォルト値)



インストールを完了する前に、構成に適したストレージクラスと入力タイプを選択していることを確認してください。

## astra\_control\_center.yamlの例

```
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: "[cr.astra.netapp.io or your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
  volumeReclaimPolicy: "Retain"
  ingressType: "Generic"
  scaleSize: "Medium"
  astraResourcesScaler: "Default"
  additionalValues:
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
  crds:
    externalTraefik: false
    externalCertManager: false
```

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

1. 前の手順でまだ行っていない場合は、を作成します netapp-acc (またはカスタム) ネームスペース:

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

2. Astra Control Centerでサービスマッシュを使用している場合は、 netapp-acc またはカスタムネームスペース:



入力タイプ (ingressType) をに設定する必要があります。Generic このコマンドを実行する前に、Astra Control Center CRで確認する必要があります。

```
kubectl label ns [netapp-acc or custom namespace] istio-  
injection:enabled
```

### 3. (推奨) "厳密なMTLを有効にする" Istioサービスマッシュの場合：

```
kubectl apply -n istio-system -f - <<EOF  
apiVersion: security.istio.io/v1beta1  
kind: PeerAuthentication  
metadata:  
  name: default  
spec:  
  mtls:  
    mode: STRICT  
EOF
```

### 4. にAstra Control Centerをインストールします netapp-acc (またはカスタムの) ネームスペース：

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



Astra Control Centerのオペレータが環境要件の自動チェックを実行ありません "要件" 原因でインストールが失敗するか、Astra Control Centerが正常に動作しない可能性があります。を参照してください [次のセクション](#) 自動システムチェックに関連する警告メッセージをチェックします。

## システムステータスを確認します

kubectlコマンドを使用すると、システムステータスを確認できます。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。

### 手順

1. インストールプロセスで検証チェックに関連する警告メッセージが生成されなかったことを確認します。

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



その他の警告メッセージは、Astra Control Centerのオペレータログでも報告されます。

2. 自動化された要件チェックによって報告された環境の問題を修正します。



問題を解決するには、環境が満たしていることを確認します "要件" (Astra Control Center向け)。

3. 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がになっている必要があります Running。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。

サンプル応答のために展開

acc-helm-repo-5bd77c9ddd-8wxm2 1h	1/1	Running	0
activity-5bb474dc67-819ss 1h	1/1	Running	0
activity-5bb474dc67-qbrtq 1h	1/1	Running	0
api-token-authentication-6wbj2 1h	1/1	Running	0
api-token-authentication-9pgw6 1h	1/1	Running	0
api-token-authentication-tqf6d 1h	1/1	Running	0
asup-5495f44dbd-z4kft 1h	1/1	Running	0
authentication-6fdd899858-5x45s 1h	1/1	Running	0
bucket-service-84d47487d-n9xgp 1h	1/1	Running	0
bucket-service-84d47487d-t5jhm 1h	1/1	Running	0
cert-manager-5dcb7648c4-hbldc 1h	1/1	Running	0
cert-manager-5dcb7648c4-nr9qf 1h	1/1	Running	0
cert-manager-cainjector-59b666fb75-bk2tf 1h	1/1	Running	0
cert-manager-cainjector-59b666fb75-pfnck 1h	1/1	Running	0
cert-manager-webhook-c6f9b6796-ngz2x 1h	1/1	Running	0
cert-manager-webhook-c6f9b6796-rwtbn 1h	1/1	Running	0
certificates-5f5b7b4dd-52tnj 1h	1/1	Running	0
certificates-5f5b7b4dd-gtjbx 1h	1/1	Running	0
certificates-expiry-check-28477260-dz5vw 1h	0/1	Completed	0
cloud-extension-6f58cc579c-lzfmv 1h	1/1	Running	0
cloud-extension-6f58cc579c-zw2km 1h	1/1	Running	0
cluster-orchestrator-79dd5c8d95-qjg92	1/1	Running	0

1h			
composite-compute-85dc84579c-nz82f	1/1	Running	0
1h			
composite-compute-85dc84579c-wx2z2	1/1	Running	0
1h			
composite-volume-bff6f4f76-789nj	1/1	Running	0
1h			
composite-volume-bff6f4f76-kwnd4	1/1	Running	0
1h			
credentials-79fd64f788-m7m8f	1/1	Running	0
1h			
credentials-79fd64f788-qnc6c	1/1	Running	0
1h			
entitlement-f69cdbc77-4p2kn	1/1	Running	0
1h			
entitlement-f69cdbc77-hswm6	1/1	Running	0
1h			
features-7b9585444c-7xd7m	1/1	Running	0
1h			
features-7b9585444c-dcqwc	1/1	Running	0
1h			
fluent-bit-ds-crq8m	1/1	Running	0
1h			
fluent-bit-ds-gmgq8	1/1	Running	0
1h			
fluent-bit-ds-gzr4f	1/1	Running	0
1h			
fluent-bit-ds-j6sf6	1/1	Running	0
1h			
fluent-bit-ds-v4t9f	1/1	Running	0
1h			
fluent-bit-ds-x7j59	1/1	Running	0
1h			
graphql-server-6cc684fb46-2x8lr	1/1	Running	0
1h			
graphql-server-6cc684fb46-bshbd	1/1	Running	0
1h			
hybridauth-84599f79fd-fjc7k	1/1	Running	0
1h			
hybridauth-84599f79fd-s9pmn	1/1	Running	0
1h			
identity-95df98cb5-dvlmz	1/1	Running	0
1h			
identity-95df98cb5-krf59	1/1	Running	0
1h			
influxdb2-0	1/1	Running	0

1h			
keycloak-operator-6d4d688697-cfq8b	1/1	Running	0
1h			
krakend-5d5c8f4668-7bq8g	1/1	Running	0
1h			
krakend-5d5c8f4668-t8hbn	1/1	Running	0
1h			
license-689cdd4595-2gsc8	1/1	Running	0
1h			
license-689cdd4595-g6vwk	1/1	Running	0
1h			
login-ui-57bb599956-4fwgz	1/1	Running	0
1h			
login-ui-57bb599956-rhztb	1/1	Running	0
1h			
loki-0	1/1	Running	0
1h			
metrics-facade-846999bdd4-f7jdm	1/1	Running	0
1h			
metrics-facade-846999bdd4-lnsxl	1/1	Running	0
1h			
monitoring-operator-6c9d6c4b8c-ggkrl	2/2	Running	0
1h			
nats-0	1/1	Running	0
1h			
nats-1	1/1	Running	0
1h			
nats-2	1/1	Running	0
1h			
natssync-server-6df7d6cc68-9v2gd	1/1	Running	0
1h			
nautilus-64b7fbbd98-bsgwb	1/1	Running	0
1h			
nautilus-64b7fbbd98-djllhw	1/1	Running	0
1h			
openapi-864584bccc-75nlv	1/1	Running	0
1h			
openapi-864584bccc-zh6bx	1/1	Running	0
1h			
polaris-consul-consul-server-0	1/1	Running	0
1h			
polaris-consul-consul-server-1	1/1	Running	0
1h			
polaris-consul-consul-server-2	1/1	Running	0
1h			
polaris-keycloak-0	1/1	Running	2 (1h)

```

ago)      1h
polaris-keycloak-1      1/1      Running      0
1h
polaris-keycloak-db-0   1/1      Running      0
1h
polaris-keycloak-db-1   1/1      Running      0
1h
polaris-keycloak-db-2   1/1      Running      0
1h
polaris-mongodb-0       1/1      Running      0
1h
polaris-mongodb-1       1/1      Running      0
1h
polaris-mongodb-2       1/1      Running      0
1h
polaris-ui-66476dcf87-f6s8j  1/1      Running      0
1h
polaris-ui-66476dcf87-ztjk7  1/1      Running      0
1h
polaris-vault-0         1/1      Running      0
1h
polaris-vault-1         1/1      Running      0
1h
polaris-vault-2         1/1      Running      0
1h
public-metrics-bfc4fc964-x4m79  1/1      Running      0
1h
storage-backend-metrics-7dbb88d4bc-g78cj  1/1      Running      0
1h
storage-provider-5969b5df5-hjvcm  1/1      Running      0
1h
storage-provider-5969b5df5-r79ld  1/1      Running      0
1h
task-service-5fc9dc8d99-4q4f4  1/1      Running      0
1h
task-service-5fc9dc8d99-8l5zl  1/1      Running      0
1h
task-service-task-purge-28485735-fdzkd  1/1      Running      0
12m
telegraf-ds-2rgm4       1/1      Running      0
1h
telegraf-ds-4qp6r       1/1      Running      0
1h
telegraf-ds-77frs       1/1      Running      0
1h
telegraf-ds-bc725       1/1      Running      0

```

```

1h
telegraf-ds-cvmxf          1/1    Running    0
1h
telegraf-ds-tqzgj          1/1    Running    0
1h
telegraf-rs-5wtd8          1/1    Running    0
1h
telemetry-service-6747866474-5djnc 1/1    Running    0
1h
telemetry-service-6747866474-thb7r 1/1    Running    1 (1h
ago)
tenancy-5669854fb6-gzdzf  1/1    Running    0
1h
tenancy-5669854fb6-xvsm2  1/1    Running    0
1h
traefik-8f55f7d5d-4lgfw   1/1    Running    0
1h
traefik-8f55f7d5d-j4wt6   1/1    Running    0
1h
traefik-8f55f7d5d-p6gcq   1/1    Running    0
1h
trident-svc-7cb5bb4685-54cnq 1/1    Running    0
1h
trident-svc-7cb5bb4685-b28xh 1/1    Running    0
1h
vault-controller-777b9bbf88-b5bqt 1/1    Running    0
1h
vault-controller-777b9bbf88-fdfd8 1/1    Running    0
1h

```

#### 4. (オプション) acc-operator 進捗状況を監視するログ：

```
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処理の1つです。登録に失敗しても原因の導入は失敗しません。ログにクラスタ登録エラーが記録されている場合は、を使用して再度登録を試行できます ["UIでクラスタワークフローを追加します"](#) または API。

- すべてのポッドが実行中の場合は、インストールが正常に完了したことを確認します (READY はです True) にアクセスし、Astra Control Centerにログインするときに使用する初期セットアップパスワードを取得します。

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

対応：

NAME	UUID	VERSION	ADDRESS
READY			
astra	9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f	24.02.0-69	
10.111.111.111	True		



UUIDの値をコピーします。パスワードはです ACC- 続けてUUIDの値を指定します (ACC-[UUID] または、この例では、ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f)。

## ロードバランシング用の入力を設定します

サービスへの外部アクセスを管理するKubernetes入力コントローラを設定できます。これらの手順では、デフォルトの使用した場合の入力コントローラの設定例を示します `ingressType: "Generic" Astra Control Center`のカスタムリソース (`astra_control_center.yaml`)。を指定した場合、この手順を使用する必要はありません `ingressType: "AccTraefik" Astra Control Center`のカスタムリソース (`astra_control_center.yaml`)。

Astra Control Centerを導入したら、URLを使用してAstra Control Centerを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。

セットアップ手順は、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。Astra Control Centerは、多くの入力コントローラタイプをサポートしています。ここでは、一部の一般的な入力コントローラタイプの設定手順の例を示します。

作業を開始する前に

- 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。

## Istio Ingressの手順

1. Istio Ingressを設定します。



この手順では、「デフォルト」の構成プロファイルを使用してIstioが導入されていることを前提としています。

2. 入力ゲートウェイに必要な証明書と秘密鍵ファイルを収集または作成します。

CA署名証明書または自己署名証明書を使用できます。共通名はAstraアドレス (FQDN) である必要があります。

コマンド例：

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

3. シークレットを作成します `tls secret name` を入力します `kubernetes.io/tls` でTLS秘密鍵と証明書を使用する場合 `istio-system namespace TLSシークレット` で説明されているように、

コマンド例：

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



シークレットの名前はと一致する必要があります `spec.tls.secretName` で提供されま  
す `istio-ingress.yaml` ファイル。

4. に入力リソースを配置します `netapp-acc` (またはカスタムネームスペース)。スキーマにはv1リソースタイプを使用します (`istio-Ingress.yaml` は次の例で使用されています)。

```

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. 変更を適用します。

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

6. 入力ステータスを確認します。

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

対応:

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

## 7. Astra Control Centerのインストールを完了します。

### Ngix Ingress Controller の手順

1. タイプのシークレットを作成します `kubernetes.io/tls` でTLSの秘密鍵と証明書を使用する場合 `netapp-acc` (またはカスタム名前付き) ネームスペース。を参照してください "[TLS シークレット](#)"。
2. 入力リソースをに配置します `netapp-acc` (またはカスタムネームスペース)。スキーマにはv1リソースタイプを使用します (`nginx-Ingress.yaml` は次の例で使用されています)。

```
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. 変更を適用します。

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



ネットアップでは、nginxコントローラをではなく導入環境としてインストールすることを推奨します `daemonSet`。

### 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ら、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、（を含む）FQDNを入力します `https://` プレフィックス）を使用します `astraAddress` を参照してください `astra_control_center.yaml` CR When（時間） [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、自己署名証明書を承認します。



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、に使用した値を入力します `email` インチ `astra_control_center.yaml` CR When（時間） [Astra Control Center をインストールした](#) をクリックし、次に初期セットアップパスワードを入力します (`ACC-[UUID]`) 。



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。



初めてログインしたときにパスワードを忘れ、他の管理ユーザアカウントがまだ作成されていない場合は、にお問い合わせください "[ネットアップサポート](#)" パスワード回復のサポートを受けるには、

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がにある場合 `Error` ステータスを確認すると、ログを調べることができます。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。

オプション（Options）

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

- Astra Control Center CRの出力を確認するには、次の手順を実行します。

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

## 別のインストール手順

- \* Red Hat OpenShift OperatorHubでインストール\*：これを使用 ["代替手順"](#) OperatorHubを使用してOpenShiftにAstra Control Centerをインストールするには、次の手順を実行します。
- \* Cloud Volumes ONTAP バックエンドを使用してパブリッククラウドにインストール\*：ユース ["これらの手順に従います"](#) Amazon Web Services (AWS)、Google Cloud Platform (GCP)、または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、次の手順を実行します。

## 次のステップ

- (オプション) お使いの環境に応じて、インストール後に実行します ["設定手順"](#)。
- ["Astra Control Centerをインストールし、UIにログインしてパスワードを変更したら、ライセンスのセットアップ、クラスタの追加、認証の有効化、ストレージの管理、バケットの追加を行うことができます。"](#)

## 外部証明書マネージャを設定します

Kubernetesクラスタに証明書マネージャがすでに存在する場合は、Astra Control Centerで独自の証明書マネージャがインストールされないように、いくつかの前提条件となる手順を実行する必要があります。

### 手順

1. 証明書マネージャがインストールされていることを確認します。

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

### 回答例：

```
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. の証明書とキーのペアを作成します astraAddress FQDN：

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

回答例：

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

3. 以前に生成したファイルを使用してシークレットを作成します。

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

回答例：

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

4. を作成します ClusterIssuer \*とまったく同じ\*のファイル。ただし、の名前空間の場所が含まれます cert-manager ポッドがインストールされます。

```
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
```

回答例：

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

5. を確認します ClusterIssuer が正常に起動しました。Ready はである必要があります True 次の手順に進む前に、次の手順

```
kubectl get ClusterIssuer
```

回答例：

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

6. を実行します ["Astra Control Center のインストールプロセス"](#)。があります ["Astra Control Center クラスターYAMLの必須の設定手順"](#) CRD値を変更して、証明書マネージャが外部にインストールされていることを示します。Astra Control Centerが外部証明書マネージャを認識するように、インストール時にこの手順を完了する必要があります。

## 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。

作業を開始する前に

- 環境条件を満たしている：["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"](#)。



3つ目の障害ドメインまたはセカンダリサイトにAstra Control Centerを導入これは、アプリケーションのレプリケーションとシームレスなディザスタリカバリに推奨されます。

- 正常なクラスタオペレータとAPIサービスを確保：
  - 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。

```
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であることを確認します。

```
oc get apiservices
```

- **\*ルーティング可能なFQDN\***：使用するAstra FQDNをクラスタにルーティングできることを確認します。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。

- \* OpenShiftの権限を取得する\*:説明されているインストール手順を実行するには、Red Hat OpenShift Container Platformに必要なすべての権限とアクセス権が必要です。
- 証明書マネージャの設定:クラスタに証明書マネージャがすでに存在する場合は、一部の証明書マネージャを実行する必要があります。"事前に必要な手順" そのため、Astra Control Centerは独自の証明書管理ツールをインストールしません。デフォルトでは、Astra Control Centerはインストール時に独自の証明書マネージャをインストールします。
- \* Kubernetes Ingressコントローラのセットアップ\*: クラスタ内のロードバランシングなど、サービスへの外部アクセスを管理するKubernetes Ingressコントローラがある場合は、Astra Control Centerで使用するようにセットアップする必要があります。

- a. operatorネームスペースを作成します。

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

- b. "セットアップを完了" 入力コントローラのタイプ。

- (ONTAP SANドライバのみ) マルチパスの有効化: ONTAP SANドライバを使用している場合は、すべてのKubernetesクラスタでマルチパスが有効になっていることを確認してください。

また、次の点も考慮する必要があります。

- \* NetApp Astra Controlイメージレジストリへのアクセス\* :

Astra Control Provisionerなど、Astra Controlのインストールイメージや機能強化された機能をNetAppイメージレジストリから取得することができます。

- a. レジストリへのログインに必要なAstra ControlアカウントIDを記録します。

アカウントIDはAstra Control Service Web UIで確認できます。ページ右上の図アイコンを選択し、\* APIアクセス\*を選択して、アカウントIDを書き留めます。

- b. 同じページから\* APIトークンの生成\*を選択し、APIトークン文字列をクリップボードにコピーしてエディターに保存します。

- c. Astra Controlレジストリにログインします。

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

- セキュアな通信のためのサービスメッシュをインストール: Astra Controlホストクラスタの通信チャンネルは、"サポートされるサービスメッシュ"。



Astra Control Centerとサービスメッシュの統合は、Astra Control Centerでのみ実行可能 "インストール" そしてこのプロセスから独立していません。メッシュ環境から非メッシュ環境に戻すことはサポートされていません。

Istioサービスメッシュを使用するには、次の手順を実行する必要があります。

- を追加します。istio-injection:enabled Astra Control Centerを導入する前に、Astraネームスペースにラベルを付けます。

- を使用します Generic [入力設定](#) 別のインGRESSを提供します。"外部ロードバランシング"。
- Red Hat OpenShiftクラスタの場合は、NetworkAttachmentDefinition 関連付けられているすべてのAstra Control Center名前空間 (netapp-acc-operator、netapp-acc、netapp-monitoring アプリケーションクラスタの場合、または置換されたカスタム名前空間の場合)。

```
cat <<EOF | oc -n netapp-acc-operator create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-acc create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-monitoring create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF
```

## 手順

- [Astra Control Center](#)をダウンロードして展開します
- [ローカルレジストリを使用する場合は、追加の手順を実行します。]
- [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オペレータ (たとえば、`kubectl delete -f astra_control_center_operator_deploy.yaml`) Astra Control Centerのインストール中または操作中はいつでも、ポッドを削除しないようにします。

## Astra Control Centerをダウンロードして展開します

次のいずれかの場所からAstra Control Centerのイメージをダウンロードします。

- \* Astra Controlサービスのイメージレジストリ\* : Astra Control Centerのイメージでローカルレジストリを使用しない場合や、NetApp Support Siteからバンドルをダウンロードするよりもこの方法を使用する場合

は、このオプションを使用します。

- **\* NetApp Support Site \***：このオプションは、Astra Control Centerのイメージでローカルレジストリを使用する場合に使用します。

#### Astra Controlイメージレジストリ

1. Astra Control Serviceにログインします。
2. ダッシュボードで、\*[Deploy a self-managed instance of Astra Control]\*を選択します。
3. 手順に従ってAstra Controlイメージのレジストリにログインし、Astra Control Centerのインストールイメージを取得してイメージを展開します。

#### NetApp Support Site

1. Astra Control Centerを含むバンドルをダウンロードします (astra-control-center-[version].tar.gz) をクリックします "[Astra Control Centerのダウンロードページ](#)"。
2. (推奨ですがオプション) Astra Control Centerの証明書と署名のバンドルをダウンロードします (astra-control-center-certs-[version].tar.gz) をクリックして、バンドルのシグネチャを確認します。

```
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
```

出力にはと表示されます Verified OK 検証が成功したあとに、

3. Astra Control Centerバンドルからイメージを抽出します。

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

ローカルレジストリを使用する場合は、追加の手順を実行します。

Astra Control Centerバンドルをローカルのレジストリにプッシュする場合は、NetApp Astra kubectlコマンドラインプラグインを使用する必要があります。

#### ネットアップAstra kubectlプラグインをインストール

次の手順を実行して、最新のNetApp Astra kubectlコマンドラインプラグインをインストールします。

作業を開始する前に

ネットアップでは、CPUアーキテクチャやオペレーティングシステム別にプラグインのバイナリを提供しています。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。

以前のインストールからプラグインがインストールされている場合は、"[最新バージョンがインストールされていることを確認してください](#)" これらの手順を実行する前に。

#### 手順

1. 使用可能なNetApp Astra kubectlプラグインのバイナリを表示し、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の名前をメモします。



kubectlプラグインライブラリはtarバンドルの一部であり、フォルダに解凍されます kubectl-astra。

```
ls kubectl-astra/
```

2. 正しいバイナリを現在のパスに移動し、名前をに変更します kubectl-astra :

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

#### イメージをレジストリに追加する

1. Astra Control Centerバンドルをローカルのレジストリにプッシュする場合は、コンテナエンジンに応じた手順を実行します。

## Docker です

- a. tarballのルートディレクトリに移動します。次のように表示されます。  
acc.manifest.bundle.yaml ファイルと次のディレクトリ：

```
acc/  
kubect1-astra/  
acc.manifest.bundle.yaml
```

- b. Astra Control Centerのイメージディレクトリにあるパッケージイメージをローカルレジストリにプッシュします。を実行する前に、次の置換を行ってください push-images コマンドを実行します

- <BUNDLE\_FILE> をAstra Controlバンドルファイルの名前に置き換えます (acc.manifest.bundle.yaml)。
- &lt;MY\_FULL\_REGISTRY\_PATH&gt; をDockerリポジトリのURLに置き換えます。次に例を示します。 "<a href="https://&lt;docker-registry&gt;"" class="bare">https://&lt;docker-registry&gt;""</a>。
- <MY\_REGISTRY\_USER> をユーザ名に置き換えます。
- <MY\_REGISTRY\_TOKEN> をレジストリの認証済みトークンに置き換えます。

```
kubect1 astra packages push-images -m <BUNDLE_FILE> -r  
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p  
<MY_REGISTRY_TOKEN>
```

## ポドマン

- a. tarballのルートディレクトリに移動します。次のファイルとディレクトリが表示されます。

```
acc/  
kubect1-astra/  
acc.manifest.bundle.yaml
```

- b. レジストリにログインします。

```
podman login <YOUR_REGISTRY>
```

- c. 使用するPodmanのバージョンに合わせてカスタマイズされた次のいずれかのスクリプトを準備して実行します。<MY\_FULL\_REGISTRY\_PATH> を'サブディレクトリを含むリポジトリのURLに置き換えます

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

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
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
```

**Podman 3**

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
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
```



レジストリ設定に応じて、スクリプトが作成するイメージパスは次のようになります。

```
https://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

## 2. ディレクトリを変更します。

```
cd manifests
```

## 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。

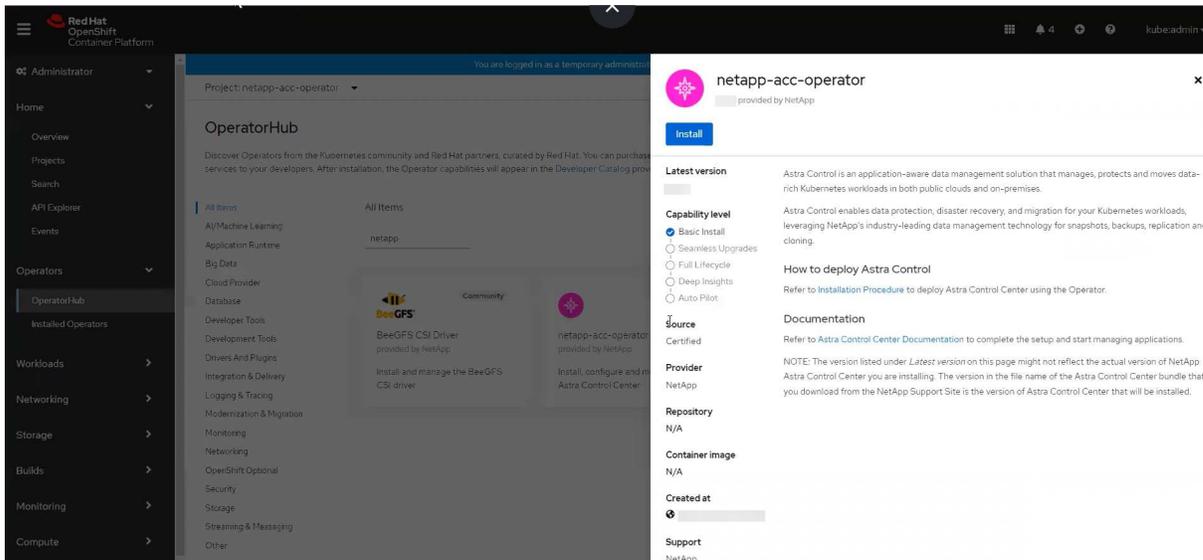
## Red Hat OpenShift Webコンソール

- OpenShift Container Platform UI にログインします。
- サイドメニューから、\* 演算子 > OperatorHub \* を選択します。



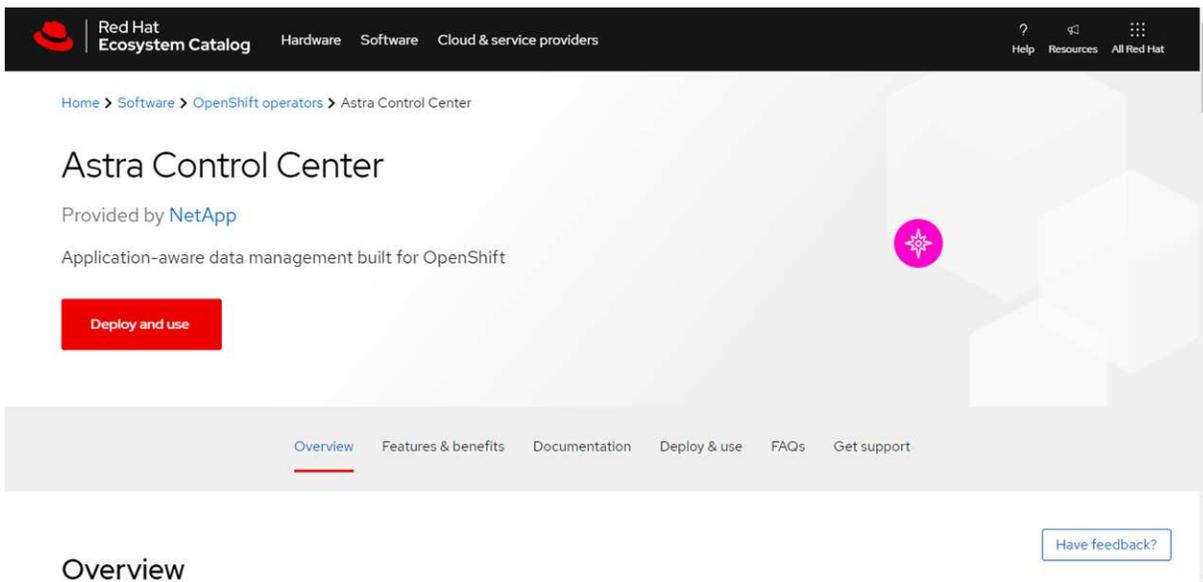
このオペレータを使用している場合は、Astra Control Centerの最新バージョンにのみアップグレードできます。

- を検索します netapp-acc にアクセスし、NetApp Astra Control Centerオペレータを選択します。



## Red Hat エコシステムカタログ

- NetApp Astra Control Center を選択します "演算子".
- [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名前空間で使用できます。

- a. operator名前空間またはを選択します netapp-acc-operator オペレータのインストールの一環として、名前空間が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。

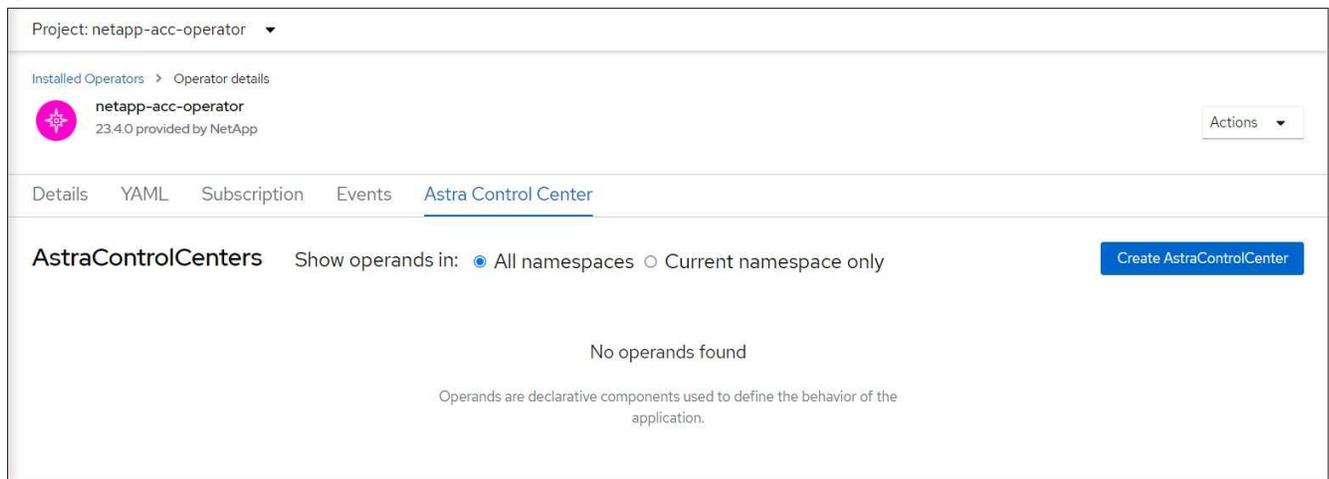


手動承認ストラテジーを選択した場合は、このオペレータの手動インストール計画を承認するように求められます。

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[Astra Control Center]タブ内のコンソールから[\*Create AstraControlCenter \*]を選択します



2. を実行します Create AstraControlCenter フォームフィールド：
  - a. Astra Control Center の名前を保持または調整します。
  - b. Astra Control Centerのラベルを追加します。
  - c. 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。
  - d. Astra Control CenterのFQDNまたはIPアドレスを入力します。入らないでください http:// または https:// をクリックします。
  - e. Astra Control Centerのバージョン（例：24.02.0-69）を入力します。

- f. 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。
- g. ボリューム再利用ポリシーを選択してください Retain、Recycle`または `Delete。デフォルト値はです Retain。
- h. インストールのスケールサイズを選択します。



デフォルトでは、Astraで高可用性 (HA) が使用されます。scaleSize の Medium`ほとんどのサービスをHAに導入し、冗長性を確保するために複数のレプリカを導入します。を使用 `scaleSize として `Small` Astraは、消費量を削減するための必須サービスを除き、すべてのサービスのレプリカ数を削減します。

- i. 入力タイプを選択します。

- 汎用 (ingressType: "Generic") (デフォルト)

このオプションは、別の入力コントローラを使用している場合、または独自の入力コントローラを使用する場合に使用します。Astra Control Centerを導入したら、**"入力コントローラ"** URLを使用してAstra Control Centerを公開します。

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

入力コントローラを設定しない場合は、このオプションを使用します。これにより、Astra Control Centerが導入されます traefik ゲートウェイをKubernetesの「LoadBalancer」タイプのサービスとして使用します。

Astra Control Centerは、タイプ「LoadBalancer」のサービスを使用します。(svc/traefik Astra Control Centerの名前空間)で、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。



「LoadBalancer」およびIngressのサービスタイプの詳細については、を参照してください **"要件"**。

- a. **\* Image Registry\***では、ローカルレジストリを構成していない限り、デフォルト値を使用します。ローカルレジストリの場合は、この値を、前の手順でイメージをプッシュしたローカルイメージレジストリパスに置き換えます。入らないでください http:// または https:// をクリックします。
- b. 認証が必要なイメージレジストリを使用する場合は、イメージシークレットを入力します。



認証が必要なレジストリを使用する場合は、**クラスタでシークレットを作成します**。

- c. 管理者の名を入力します。
- d. リソースの拡張を構成する。
- e. デフォルトのストレージクラスを指定します。



デフォルトのストレージクラスが設定されている場合は、そのストレージクラスがデフォルトのアノテーションを持つ唯一のストレージクラスであることを確認します。

- f. CRD 処理の環境設定を定義します。
3. YAMLビューを選択して、選択した設定を確認します。
4. 選択するオプション `Create`。

## レジストリシークレットを作成します

認証が必要なレジストリを使用する場合は、OpenShiftクラスタでシークレットを作成し、シークレット名を `Create AstraControlCenter` フォームフィールド。

1. Astra Control Centerオペレータの名前空間を作成します。

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

2. この名前空間にシークレットを作成します。

```
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は、Dockerレジストリシークレットのみをサポートします。

3. の残りのフィールドに値を入力します [Create AstraControlCenterフォーム・フィールド](#)。

## 次のステップ

を実行します ["残りのステップ"](#) Astra Control Centerが正常にインストールされたことを確認するには、入力コントローラ（オプション）をセットアップし、UIにログインします。さらに、["セットアップのタスク"](#) インストールが完了したら、

# 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)、Google Cloud Platform (GCP)、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](#)
- [Astra Control CenterをGoogle Cloud Platformに導入](#)
- [Microsoft Azure に Astra Control Center を導入](#)

OpenShift Container Platform (OCP) などの自己管理型Kubernetesクラスタを使用して、ディストリビューション内のアプリケーションを管理できます。Astra Control Centerを導入するために検証されるのは、自己管理型のOCPクラスタのみです。

## 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。

### AWSに必要なもの

AWSにAstra Control Centerを導入する前に、次の項目が必要になります。

- 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
- OCPを使用する場合は、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でホストされるゾーンとAmazon Route 53のエントリがAstra Control UIにアクセスするために必要

### 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 4.11~4.13

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て、特定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](#)"。



AWSレジストリトークンは12時間で期限切れになります。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 BlueXP for AWSを構成します。](#)
5. [Astra Control Center for AWSをインストール。](#)

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 BlueXP（旧Cloud Manager）コネクタをインストールできる十分な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、仮想ネットワークを作成するようにAWSを設定し、EC2コンピューティングインスタンスをセットアップし、AWS S3バケットを作成します。NetApp Astra Control Centerのイメージレジストリにアクセスできない場合は、Astra Control CenterのイメージをホストするElastic Container Registry（ECR）を作成し、イメージをこのレジストリにプッシュする必要があります。

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. (オプション) NetAppイメージレジストリにアクセスできない場合は、次の手順を実行します。
  - a. AWS Elastic Container Registry（ECR）を作成して、Astra Control Centerのすべてのイメージをホストします。



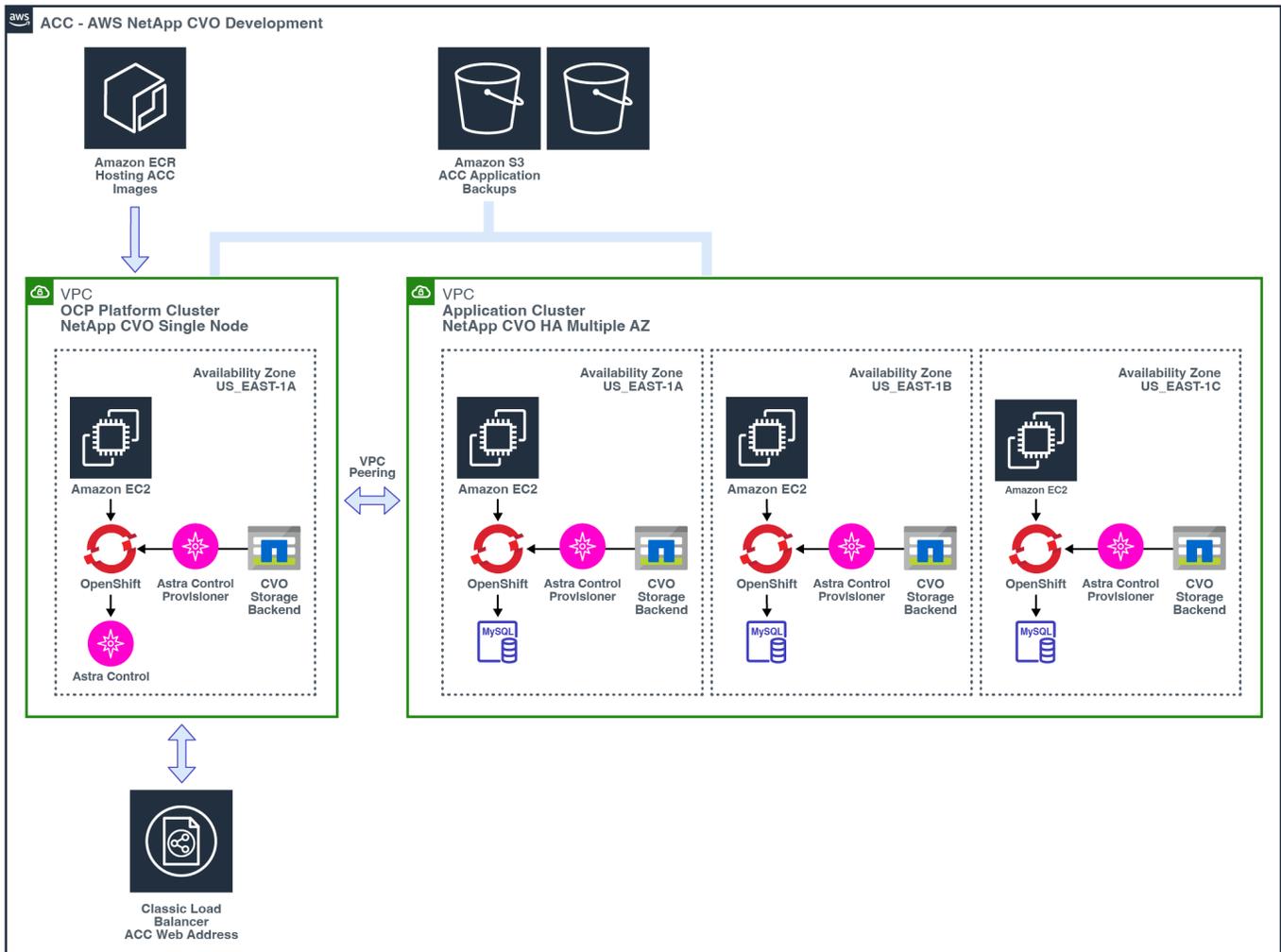
ECRを作成しないと、Astra Control Centerは、AWSバックエンドを持つCloud Volumes ONTAP を含むクラスタからモニタリングデータにアクセスできません。問題は、Astra Control Center を使用して検出および管理しようとしたクラスタに AWS ECR アクセスがない場合に発生します。

- b. Astra Control Centerのイメージを定義済みのレジストリにプッシュ



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 BlueXP for AWSを構成します

NetApp BlueXP (旧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。

BlueXPのマニュアル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"](#)。
- ["BlueXPを使用してAWSでコネクタを作成します"](#)

### 手順

1. 資格情報をBlueXPに追加します。

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. 右上にあるAstra Control Provisionerのバージョンを確認します。
  - 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。

Astra Control Provisionerは、インポートと検出のプロセスで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せん。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 for AWSをインストール

標準に従ってください "[Astra Control Center のインストール手順](#)".



AWSでは汎用のS3バケットタイプが使用されます。

## Astra Control CenterをGoogle Cloud Platformに導入

Astra Control Centerは、Google Cloud Platform (GCP) パブリッククラウドでホストされる自己管理型のKubernetesクラスタに導入できます。

### GCPに必要なもの

GCPにAstra Control Centerを導入する前に、次の項目が必要です。

- 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
- OCPを使用している場合、Red Hat OpenShift Container Platform (OCP) 4.11~4.13
- OCPを使用する場合は、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GCPサービスアカウント

## GCP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て、特定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"](#)。

## GCPの導入の概要

ここでは、Cloud Volumes ONTAP をストレージバックエンドとして使用して、GCP内の自己管理型OCPクラスタにAstra Control Centerをインストールする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. [GCPにRed Hat OpenShiftクラスタをインストールします。](#)
2. [GCPプロジェクトとVirtual Private Cloudを作成します。](#)
3. 十分な IAM 権限があることを確認します。
4. [GCPを設定します。](#)
5. [NetApp BlueXP for GCPを構成します。](#)
6. [Astra Control Center for GCPをインストールします。](#)

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

まず、GCPにRedHat 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、次を参照してください。

- ["GCPにOpenShiftクラスタをインストールする"](#)
- ["GCPサービスアカウントの作成"](#)

## GCPプロジェクトとVirtual Private Cloudを作成します

少なくとも1つのGCPプロジェクトとVirtual Private Cloud (VPC) を作成します。



OpenShift では、独自のリソースグループを作成できます。さらに、GCP VPC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 BlueXP (旧Cloud Manager) コネクタをインストールできる十分な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 ["GCPの初期資格情報と権限"](#)。

## GCPを設定します

次に、GCPを設定してVPCを作成し、コンピューティングインスタンスをセットアップし、Google Cloud Object Storageを作成します。NetApp Astra Control Centerのイメージレジストリにアクセスできない場合は、Astra Control CenterのイメージをホストするGoogleコンテナレジストリを作成し、このレジストリにイメージをプッシュする必要もあります。

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

1. GCPでGCPプロジェクトとVPCを作成します。GCPでは、CVOバックエンドでOCPクラスタ用にを使用する予定です。
2. コンピューティングインスタンスを確認します。GCP内のベアメタルサーバまたはVMです。
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最小リソース要件と一致していない場合は、GCP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. バックアップを保存するGCP Cloud Storageバケットを少なくとも1つ作成します。
5. 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。
6. (オプション) NetAppイメージレジストリにアクセスできない場合は、次の手順を実行します。
  - a. Astra Control CenterのイメージをホストするGoogle Container Registryを作成します。
  - b. すべてのAstra Control Centerイメージに対して、Dockerプッシュ/プル用のGoogle Container Registryアクセスを設定します。

例：次のスクリプトを入力して、Astra Control Centerのイメージをこのレジストリにプッシュできます。

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

このスクリプトには、Astra Control CenterマニフェストファイルとGoogle Image Registryの場所が必要が必要です。例

```
manifestfile=acc.manifest.bundle.yaml
GCP_CR_REGISTRY=<target GCP image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

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 < acc.manifest.bundle.yaml
```

7. DNS ゾーンを設定します。

### NetApp BlueXP for GCPを構成します

NetApp BlueXP (旧Cloud Manager) を使用して、ワークスペースの作成、GCP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。

BlueXPのマニュアルに従って'次の手順を実行しますを参照してください "[GCPでCloud Volumes ONTAP の使用を開始する](#)".

作業を開始する前に

- 必要なIAM権限と役割を持つGCPサービスアカウントにアクセスします

手順

1. 資格情報をBlueXPに追加します。を参照してください "[GCP アカウントの追加](#)".
2. GCPのコネクターを追加します。
  - a. プロバイダーとして[GCP]を選択します。
  - b. GCP資格情報を入力します。を参照してください "[BlueXPからGCPでコネクタを作成する](#)".
  - c. 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。
3. クラウド環境の作業環境を構築
  - a. 場所: "GCP"
  - b. 「Cloud Volumes ONTAP HA」と入力します。
4. 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. 右上にあるAstra Control Provisionerのバージョンを確認します。
  - 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。

Astra Control Provisionerは、インポートと検出のプロセスで自動的にインストールされます。

5. 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 (HA) で動作します。HAが有効になっている場合は、GCP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 for GCP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](#)".



GCPでは汎用S3バケットタイプが使用されます。

1. Astra Control Centerインストール用のイメージをプルするDocker Secretを生成します。

```
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
```

## 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、次の項目が必要になります。

- 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
- OCPを使用している場合、Red Hat OpenShift Container Platform (OCP) 4.11~4.13
- OCPを使用する場合は、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て、特定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](#)"。

### 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 BlueXP \(旧Cloud Manager\) をAzure向けに設定します。](#)
6. [Azure向けAstra Control Centerのインストールと設定。](#)

## 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、次を参照してください。

- ["Azure への OpenShift クラスタのインストール"](#)。
- ["Azure アカウントをインストールする"](#)。

## 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 BlueXP Connector をインストールできる十分な 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、仮想ネットワークを作成し、コンピューティングインスタンスをセットアップし、Azure Blob コンテナを作成するように Azure を設定します。NetApp Astra Control Center のイメージレジストリにアクセスできない場合は、Astra Control Center のイメージをホストする Azure Container Registry (ACR) を作成し、イメージをこのレジストリにプッシュする必要もあります。

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. (オプション) NetApp イメージレジストリにアクセスできない場合は、次の手順を実行します。

- a. Astra Control CenterのイメージをホストするAzure Container Registry (ACR) を作成します。
- b. Astra Control Centerのすべてのイメージに対して、Dockerによるプッシュ/プルACRアクセスをセットアップします。
- c. 次のスクリプトを使用して、Astra Control Centerの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 the Astra Control Center manifest file and your
Azure ACR location.
```

▪ 例 \* :

```
manifestfile=acc.manifest.bundle.yaml
AZ_ACR_REGISTRY=<target Azure ACR image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

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 $AZ_ACR_REGISTRY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < acc.manifest.bundle.yaml
```

8. DNS ゾーンを設定します。

### NetApp BlueXP (旧Cloud Manager) をAzure向けに設定します

BlueXP (旧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、クラスターのインポートを行います。

BlueXPのマニュアルに従って'次の手順を実行しますを参照してください "[BlueXPの使用を開始しました](#)".

作業を開始する前に

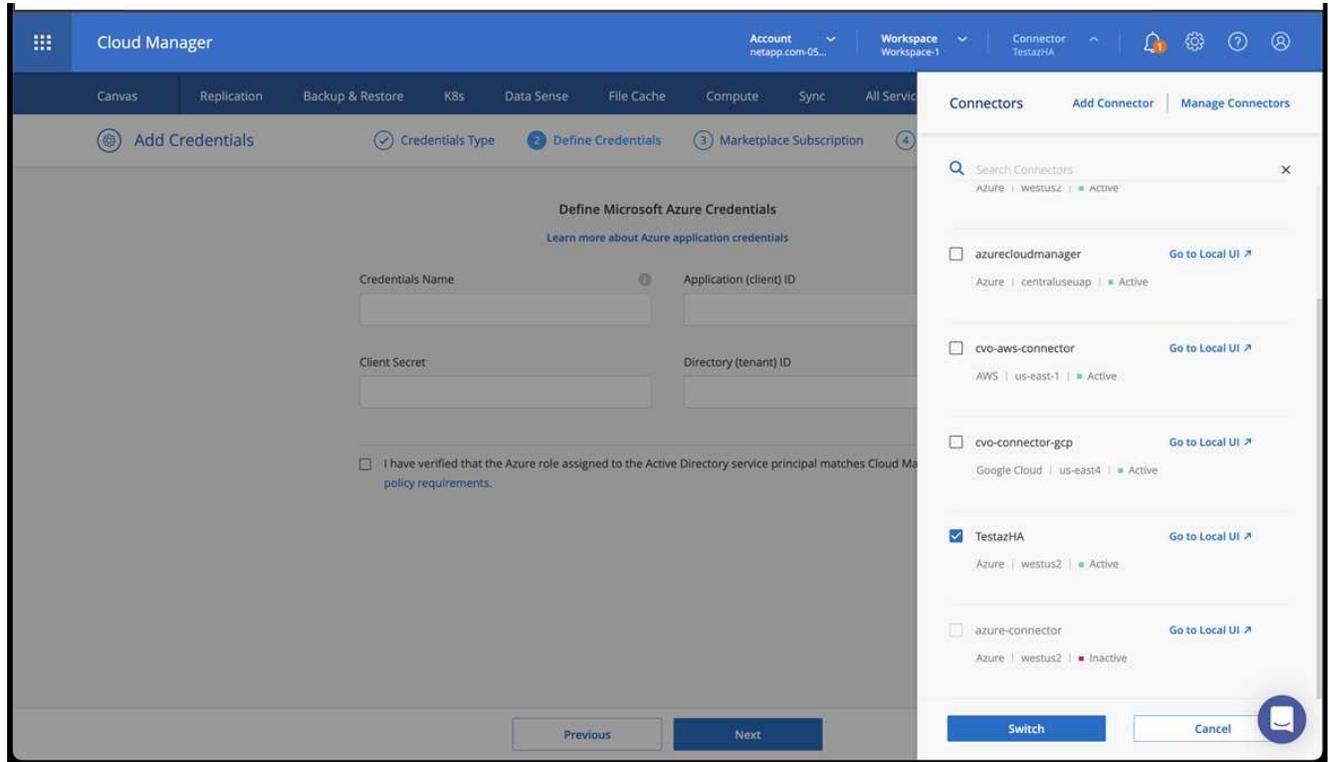
必要な IAM 権限とロールを持つ Azure アカウントにアクセスします

手順

1. 資格情報をBlueXPに追加します。
2. Azure 用のコネクタを追加します。を参照してください "[BlueXP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 "[BlueXPからAzureでコネクタを作成しています](#)".

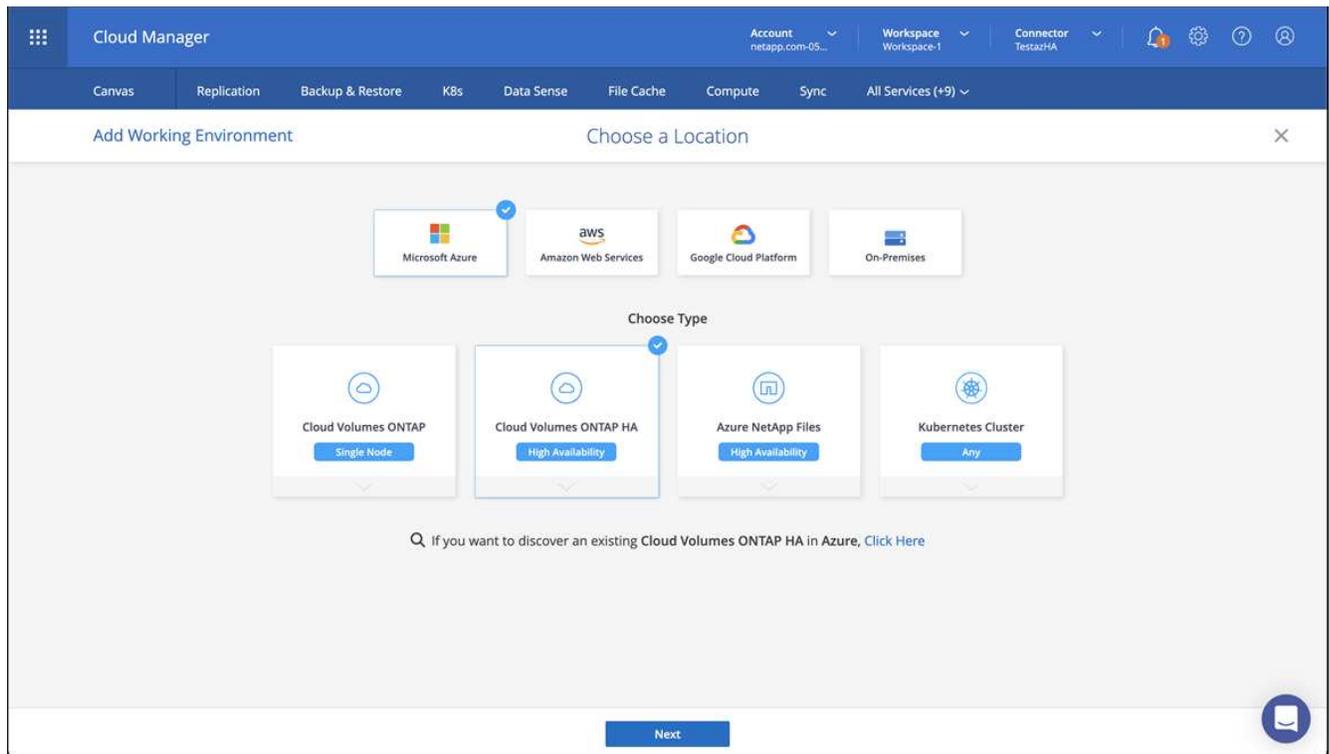
3. コネクタが動作していることを確認し、コネクタに切り替えます。



4. クラウド環境の作業環境を構築

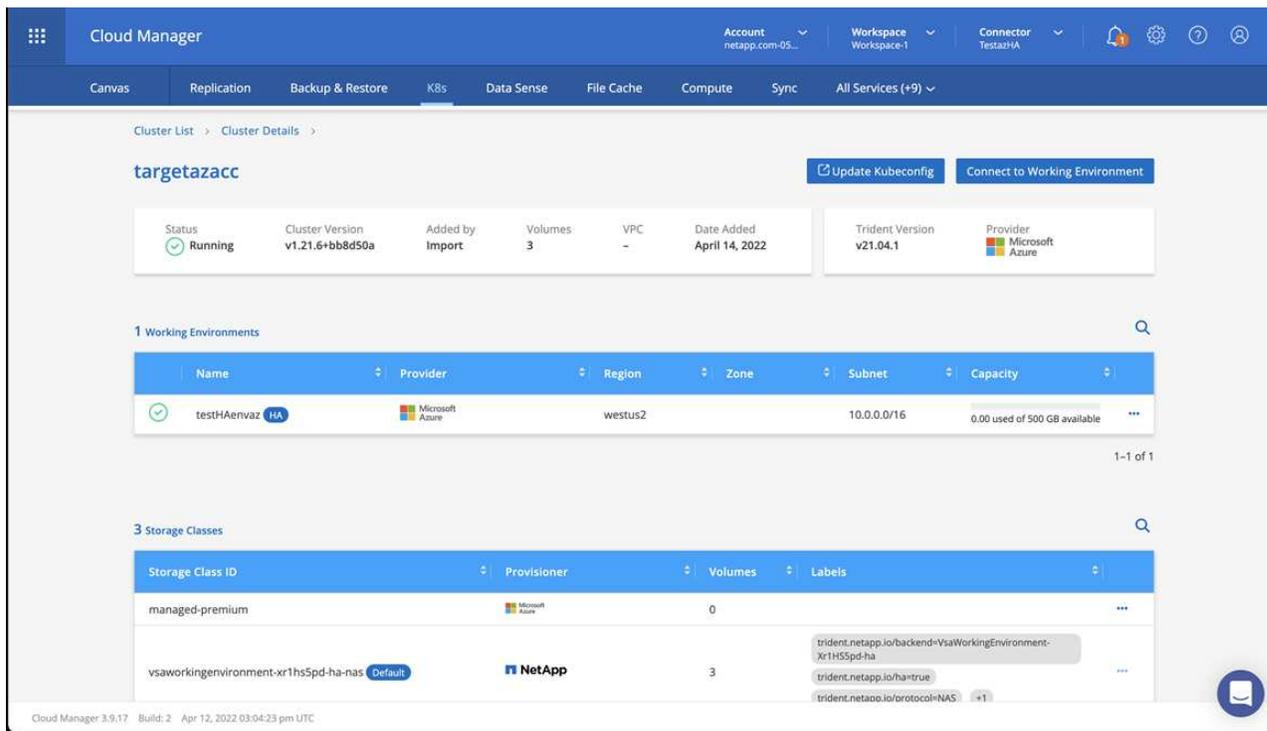
a. 場所：「Microsoft Azure」。

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. 右上にあるAstra Control Provisionerのバージョンを確認します。

- 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。

Astra Control Provisionerは、インポートと検出のプロセスで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。

## Azure向けAstra Control Centerのインストールと設定

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"](#)。

## インストール後にAstra Control Centerを設定します

環境によっては、Astra Control Centerのインストール後に追加の設定が必要になる場合があります。

## リソースの制限を解除します

一部の環境では、ResourceQuotasオブジェクトとLimitRangesオブジェクトを使用して、名前空間内のリソースがクラスター上の使用可能なCPUとメモリをすべて消費しないようにします。Astra Control Centerでは上限が設定されていないため、これらのリソースに準拠していません。この方法で環境を構成している場合は、Astra Control Centerをインストールする名前空間からリソースを削除する必要があります。

これらのクォータと制限を取得および削除するには、次の手順を実行します。これらの例では、コマンド出力はコマンド出力の直後に表示されます。

### 手順

1. でリソースクォータを取得します netapp-acc（またはカスタム名）名前空間：

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

対応：

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. 名前別にすべてのリソースクォータを削除します。

```
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. で制限範囲を取得します netapp-acc（またはカスタム名）名前空間：

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

対応：

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

#### 4. 制限範囲を名前で削除します。

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

## カスタム TLS 証明書を追加します

Astra Control Centerは、入力コントローラトラフィック（一部の設定のみ）およびWebブラウザでのWeb UI 認証に、デフォルトで自己署名TLS証明書を使用します。本番環境で使用する場合は、既存の自己署名TLS証明書を削除して、認証局（CA）によって署名されたTLS証明書に置き換える必要があります。

デフォルトの自己署名証明書は、次の2種類の接続に使用されます。



- Astra Control Center Web UIへのHTTPS接続
- 入力コントローラトラフィック（がの場合のみ） ingressType: "AccTraefik" プロパティはで設定されました astra\_control\_center.yaml Astra Control Centerのインストール中にファイルを作成)

これらの接続の認証に使用される証明書は、デフォルトのTLS証明書に置き換えられます。

作業を開始する前に

- Astra Control Center をインストールした Kubernetes クラスタ
- 実行するクラスタ上のコマンドシェルへの管理アクセス kubectl コマンド
- CA の秘密鍵ファイルと証明書ファイル

自己署名証明書を削除します

既存の自己署名 TLS 証明書を削除します。

1. SSH を使用して、Astra Control Center をホストする Kubernetes クラスタに管理ユーザとしてログインします。
2. 次のコマンドを使用して、現在の証明書に関連付けられているTLSシークレットを検索します <ACC-deployment-namespace> Astra Control Center導入ネームスペースを使用して、次の作業を行います。

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

3. 次のコマンドを使用して、現在インストールされているシークレットと証明書を削除します。

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

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

コマンドラインを使用して新しい証明書を追加します

CAによって署名された新しい TLS 証明書を追加します。

1. 次のコマンドを使用して、CAの秘密鍵ファイルと証明書ファイルを使用して新しい TLS シークレットを作成し、括弧 <> の引数を適切な情報に置き換えます。

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

2. 次のコマンドと例を使用して、クラスタカスタムリソース定義 (CRD) ファイルを編集し、を変更します。spec.selfSigned の値 spec.ca.secretName 以前に作成した TLS シークレットを参照するには、次の手順を実行します

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

CRD :

```
#spec:  
# selfSigned: {}  
  
spec:  
  ca:  
    secretName: <secret-name>
```

3. 次のコマンドと出力例を使用して、変更が正しいこと、および交換する証明書をクラスタで検証する準備ができていることを確認します <ACC-deployment-namespace> Astra Control Center 導入ネームスペースを使用して、次の作業を行います。

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

対応 :

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

4. を作成します certificate.yaml 次の例を使用してファイルを作成し、括弧<>のプレースホルダ値を適切な情報に置き換えます。



この例では、dnsNames Astra Control CenterのDNSアドレスを指定するプロパティ。Astra Control Centerでは、Common Name (CN) プロパティを使用したDNSアドレスの指定はサポートされていません。

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

5. 次のコマンドを使用して証明書を作成します。

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

6. 次のコマンドと出力例を使用して、証明書が正しく作成されていること、および作成時に指定した引数（名前、期間、更新期限、DNS名など）を使用していることを確認します。

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

対応：

```

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. 次のコマンドと例を使用してTLS Stores CRDを編集し、括弧<>のプレースホルダ値を適切な情報に置き換えます。

```
kubectl edit tlsstores.traefik.io -n <ACC-deployment-namespace>
```

CRD :

```

...
spec:
  defaultCertificate:
    secretName: <certificate-secret-name>

```

8. 次のコマンドおよび例を使用して、入力 CRD TLS オプションを編集し、新しい証明書シークレットを指定します。括弧 <> のプレースホルダ値を適切な情報に置き換えます。

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

CRD :

```
...  
  tls:  
    secretName: <certificate-secret-name>
```

9. Web ブラウザを使用して、Astra Control Center の導入 IP アドレスにアクセスします。
10. 証明書の詳細がインストールした証明書の詳細と一致していることを確認します。
11. 証明書をエクスポートし、結果を Web ブラウザの証明書マネージャにインポートします。

## 著作権に関する情報

Copyright © 2025 NetApp, Inc. All Rights Reserved. Printed in the U.S.このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および/または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

## 商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。