



クラスタを追加 Astra Control Service

NetApp
April 24, 2024

目次

Astra Control Serviceにクラスタを追加	1
Astra Connectorをインストールしてクラスタを管理	1
プロバイダ管理クラスタを追加します	7
自己管理型クラスタを追加します	18

Astra Control Serviceにクラスタを追加

環境のセットアップが完了したら、Kubernetes クラスタを作成し、Astra Control Service に追加することができます。これにより、Astra Control Serviceを使用してクラスタ上のアプリケーションを保護できます。

Astra Control Serviceに追加する必要があるクラスタのタイプに応じて、異なる手順でクラスタを追加する必要があります。

- **"パブリックプロバイダが管理するクラスタをAstra Control Serviceに追加します"**：パブリックIPアドレスが割り当てられ、クラウドプロバイダが管理するクラスタを追加するには、次の手順を実行します。クラウドプロバイダのサービスプリンシパルアカウント、サービスアカウント、またはユーザアカウントが必要です。
- **"プロバイダが管理するプライベートクラスタをAstra Control Serviceに追加します"**：プライベートIPアドレスを持ち、クラウドプロバイダによって管理されるクラスタを追加するには、次の手順を実行します。クラウドプロバイダのサービスプリンシパルアカウント、サービスアカウント、またはユーザアカウントが必要です。
- **"Astra Control Serviceにパブリック自己管理型クラスタを追加"**：パブリックIPアドレスを持ち、組織で管理されているクラスタを追加するには、次の手順を実行します。追加するクラスタのkubeconfigファイルを作成する必要があります。
- **"Astra Control Serviceに自己管理型プライベートクラスタを追加"**：プライベートIPアドレスを持ち、組織で管理されているクラスタを追加するには、次の手順を実行します。追加するクラスタのkubeconfigファイルを作成する必要があります。

Astra Connectorをインストールしてクラスタを管理

Astra Connectorは、管理対象クラスタに実装されるソフトウェアで、管理対象クラスタとAstra Controlの間の通信を促進します。Astra Control Serviceを使用して管理されるクラスタには、次の2つのバージョンのAstra Connectorがあります。

- ***以前のバージョンのAstra Connector***：**"以前のバージョンのAstra Connectorをインストール"**
Kubernetesネイティブでないワークフローでクラスタを管理する場合は、クラスタ。
- **[技術プレビュー]*宣言型Kubernetes Astra Connector***：**"Kubernetesの宣言ワークフローで管理されるクラスタ用にAstra Connectorをインストール"**
宣言型Kubernetesワークフローを使用してクラスタを管理する場合は、クラスタ上。クラスタにAstra Connectorをインストールすると、クラスタが自動的にAstra Controlに追加されます。



宣言型Kubernetes Astra Connectorは、Astra Control Early Adopter Program（EAP；Astra Control早期導入プログラム）の一部としてのみ利用できます。EAPへの参加については、NetAppの営業担当者にお問い合わせください。

以前のバージョンのAstra Connectorをインストール

Astra Control Serviceでは、以前のバージョンのAstra Connectorを使用して、Astra Control Serviceと非Kubernetesネイティブのワークフローで管理されるプライベートクラスタの間の通信を可能にします。Kubernetesネイティブでないワークフローで管理す

るプライベートクラスタにAstra Connectorをインストールする必要があります。

以前のバージョンのAstra Connectorでは、非Kubernetesネイティブのワークフローで管理される次のタイプのプライベートクラスタがサポートされていました。

- Amazon Elastic Kubernetes Service (EKS)
- Azure Kubernetes Service (AKS)
- Google Kubernetes Engine (GKE)
- Red Hat OpenShift Service on AWS (ROSA)
- RosaとAWS PrivateLink
- Red Hat OpenShift Container Platform (オンプレミス)

このタスクについて

- これらの手順を実行する場合は、Astra Control Serviceで管理するプライベートクラスタに対してこれらのコマンドを実行します。
- 要塞ホストを使用している場合は、要塞ホストのコマンドラインからこれらのコマンドを問題 で実行します。

作業を開始する前に

- Astra Control Serviceで管理するプライベートクラスタにアクセスする必要があります。
- クラスタにAstra Connectorオペレータをインストールするには、Kubernetes管理者の権限が必要です。

手順

1. Kubernetesネイティブでないワークフローで管理するプライベートクラスタに、以前のAstra Connectorオペレータをインストールします。このコマンドを実行すると、ネームスペース `astra-connector-operator` が作成され、設定がネームスペースに適用されます。

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

2. オペレータが設置され、準備ができていることを確認します。

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

3. Astra ControlからAPIトークンを取得を参照してください ["Astra Automationのドキュメント"](#) 手順については、を参照し
4. Astra Connectorネームスペースを作成します。

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

5. Astra Connector CRファイルを作成してという名前を付ける `astra-connector-cr.yaml`。カッコ内の

値を、Astra Controlの環境とクラスタの構成に合わせて更新します。

- **<ASTRA_CONTROL_SERVICE_URL>** : Astra ControlサービスのWeb UI URL。例 :

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

- **<ASTRA_CONTROL_SERVICE_API_TOKEN>** : 前の手順で取得したAstra Control APIトークン。
- ****** : (AKSクラスタのみ) -プライベート<PRIVATE_AKS_CLUSTER_NAME> Kubernetes Serviceクラスタのクラスタ名。プライベートAKSクラスタを追加する場合にのみ、この行のコメントを解除して入力します。
- **<ASTRA_CONTROL_ACCOUNT_ID>** : Astra ControlのWeb UIから取得。ページ右上の図アイコンを選択し、* API access *を選択します。

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

6. データを入力した後、astra-connector-cr.yaml 正しい値を持つファイルを作成し、CRを適用します。

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

7. Astra Connectorが完全に導入されたことを確認します。

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

8. クラスタがAstra Controlに登録されたことを確認します。

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

次のような出力が表示されます。

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



ASTRACONNECTORIDはメモしておきます。ASTRACONNECTORIDは、クラスタをAstra Controlに追加するときに必要になります。

次の手順

Astra Connectorのインストールが完了したら、Astra Control Serviceにプライベートクラスタを追加する準備が整いました。

- ["プロバイダが管理するプライベートクラスタをAstra Control Serviceに追加します"](#)：プライベートIPアドレスを持ち、クラウドプロバイダによって管理されるクラスタを追加するには、次の手順を実行します。クラウドプロバイダのサービスプリンシパルアカウント、サービスアカウント、またはユーザアカウントが必要です。
- ["Astra Control Serviceに自己管理型プライベートクラスタを追加"](#)：プライベートIPアドレスを持ち、組織で管理されているクラスタを追加するには、次の手順を実行します。追加するクラスタのkubeconfigファイルを作成する必要があります。

を参照してください。

- ["クラスタを追加"](#)

(テクニカルプレビュー) 宣言型Kubernetes Astra Connectorのインストール

宣言型Kubernetesワークフローで管理されるクラスタは、Astra Connectorを使用して、管理対象クラスタとAstra Controlの間の通信を有効にします。Kubernetesの宣言ワークフローで管理するすべてのクラスタにAstra Connectorをインストールする必要があります。

宣言型Kubernetes Astra Connectorは、KubernetesコマンドとCustom Resource (CR) ファイルを使用してインストールします。

このタスクについて

- これらの手順を実行する場合は、Astra Controlで管理するクラスタでこれらのコマンドを実行します。
- 要塞ホストを使用している場合は、要塞ホストのコマンドラインからこれらのコマンドを問題 で実行します。

作業を開始する前に

- Astra Controlで管理するクラスタへのアクセスが必要です。
- クラスタにAstra Connectorオペレータをインストールするには、Kubernetes管理者の権限が必要です。



Kubernetes 1.25以降のクラスタのデフォルトであるポッドセキュリティアドミッション適用がクラスタに設定されている場合は、適切なネームスペースに対してPSA制限を有効にする必要があります。を参照してください ["Astra Controlを使用して、クラスタ管理のための環境を準備する"](#) 手順については、を参照し

手順

1. Kubernetesの宣言ワークフローを使用して管理するクラスタにAstra Connectorオペレータをインストールします。このコマンドを実行すると、ネームスペース `astra-connector-operator` が作成され、設定がネームスペースに適用されます。

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

2. オペレータが設置され、準備ができていることを確認します。

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

3. Astra ControlからAPIトークンを取得を参照してください ["Astra Automationのドキュメント"](#) 手順については、を参照し
4. トークンを使用してシークレットを作成します。<API_TOKEN>を、Astra Controlから受け取ったトークンに置き換えます。

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

5. Astra Connectorイメージの取得に使用するDockerシークレットを作成します。括弧<>の値は、環境の情報で置き換えます。



<ASTRA_CONTROL_ACCOUNT_ID>はAstra Control Web UIで確認できます。Web UIで、ページの右上にある図アイコンを選択し、*[API access]*を選択します。

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

6. Astra Connector CRファイルを作成してという名前を付ける `astra-connector-cr.yaml`。カッコ内の値を、Astra Controlの環境とクラスタの構成に合わせて更新します。
 - <ASTRA_CONTROL_ACCOUNT_ID>：前の手順でAstra Control Web UIから取得。

- <CLUSTER_NAME>：このクラスタをAstra Controlで割り当てる名前。
- <ASTRA_CONTROL_URL>：Astra ControlのWeb UI URL。例：

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

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

7. データを入力した後、astra-connector-cr.yaml 正しい値を持つファイルを作成し、CRを適用します。

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

8. Astra Connectorが完全に導入されたことを確認します。

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

9. クラスタがAstra Controlに登録されたことを確認します。

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

次のような出力が表示されます。

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

10. Astra Control Web UIの*[Clusters]*ページで、管理対象クラスタのリストにクラスタが表示されることを確認します。

プロバイダ管理クラスタを追加します

パブリックプロバイダが管理するクラスタを**Astra Control Service**に追加します

クラウド環境のセットアップが完了したら、Kubernetesクラスタを作成してAstra Control Serviceに追加できます。

- [Kubernetes クラスタを作成](#)
- [Astra Control Serviceにクラスタを追加](#)
- [\[デフォルトのストレージクラスを変更する\]](#)

Kubernetes クラスタを作成

まだクラスタがない場合は、を満たすクラスタを作成できます "[Amazon Elastic Kubernetes Service \(EKS\) 用のAstra制御サービスの要件](#)". まだクラスタがない場合は、を満たすクラスタを作成できます "[Google Kubernetes Engine \(GKE\) 用 Astra Control Service の要件](#)". まだクラスタがない場合は、を満たすクラスタを作成できます "[Azure NetApp Files を備えた Azure Kubernetes Service \(AKS\) の Astra 制御サービスの要件](#)" または "[Azure 管理ディスクを使用する Azure Kubernetes Service \(AKS\) の Astra 制御サービスの要件](#)".



Astra Control Serviceは、Azure Active Directory (Azure AD) を使用して認証とID管理を行うAKSクラスタをサポートします。クラスタを作成するときは、の手順に従ってください "[公式ドキュメント](#)" をクリックして、Azure ADを使用するようにクラスタを設定してください。AKSで管理されるAzure AD統合の要件をクラスタが満たしていることを確認する必要があります。

Astra Control Serviceにクラスタを追加

Astra Control Service にログインしたら、最初にクラスタの管理を開始します。Astra Control Serviceにクラスタを追加する前に、特定のタスクを実行し、クラスタが一定の要件を満たしていることを確認する必要があります。

Azure Kubernetes ServiceクラスタとGoogle Kubernetes Engineクラスタを管理する場合、Astra Control Provisionerのインストールとライフサイクル管理には次の2つのオプションがあることに注意してください。

- Astra Control Serviceを使用すると、Astra Control Provisionerのライフサイクルを自動的に管理できます。そのためには、Astra Control Serviceで管理するクラスタでAstra Tridentがインストールされておらず、Astra Control Provisionerが有効になっていないことを確認してください。この場合、クラスタの管理を開始すると、Astra Control ServiceによってAstra Control Provisionerが自動的に有効になり、Astra

Control Provisionerのアップグレードは自動的に処理されます。

- Astra Control Provisionerのライフサイクルは自分で管理できます。これを行うには、Astra Control Serviceでクラスタを管理する前に、クラスタでAstra Control Provisionerを有効にします。この場合、Astra Control ServiceはAstra Control Provisionerがすでに有効になっていることを検出し、再インストールやAstra Control Provisionerのアップグレードの管理は行いません。を参照してください ["Astra Control Provisionerを有効にする"](#) では、Astra Control Provisionerを有効にします。

Astra Control ServiceでAmazon Web Servicesクラスタを管理する場合、Astra Control Provisionerでのみ使用できるストレージバックエンドが必要な場合は、Astra Control Serviceで管理する前に、クラスタでAstra Control Provisionerを手動で有効にする必要があります。を参照してください ["Astra Control Provisionerを有効にする"](#) をクリックし、Astra Control Provisionerを有効にします。

作業を開始する前に

Amazon Web Services の

- クラスタを作成したIAMユーザのクレデンシャルを含むJSONファイルが必要です。 ["IAMユーザを作成する方法を説明します"](#)。
- Amazon FSx for NetApp ONTAPにはAstra Controlプロビジョニングツールが必要です。EKSクラスタのストレージバックエンドとしてAmazon FSx for NetApp ONTAPを使用する予定の場合は、 ["EKSクラスタ要件"](#)。
- （オプション）を指定する必要がある場合 `kubectl` クラスタの作成者ではない他のIAMユーザにクラスタにアクセスするためのコマンドアクセスについては、の手順を参照してください ["Amazon EKSでクラスタを作成したあと、他のIAMユーザとロールにアクセスを提供するにはどうすればよいですか？"](#)。
- NetApp Cloud Volumes ONTAP をストレージバックエンドとして使用する場合は、Amazon Web Servicesと連携するようにCloud Volumes ONTAP を設定する必要があります。Cloud Volumes ONTAP を参照してください ["セットアップのドキュメント"](#)。

Microsoft Azure

- サービスプリンシパルの作成時に、Azure CLIからの出力を含むJSONファイルが必要です。 ["サービスプリンシパルの設定方法について説明します"](#)。

JSON ファイルに追加していない場合は、Azure サブスクリプション ID も必要になります。

- NetApp Cloud Volumes ONTAP をストレージバックエンドとして使用する場合は、Microsoft Azure と連携するようにCloud Volumes ONTAP を設定する必要があります。Cloud Volumes ONTAP を参照してください ["セットアップのドキュメント"](#)。

Google Cloud

- 必要な権限を持つサービスアカウントのサービスアカウントキーファイルが必要です。 ["サービスアカウントの設定方法について説明します"](#)。
- NetApp Cloud Volumes ONTAP をストレージバックエンドとして使用する場合は、Google Cloudと連携するようにCloud Volumes ONTAP を設定する必要があります。Cloud Volumes ONTAP を参照してください ["セットアップのドキュメント"](#)。

手順

1. （オプション）Amazon EKSクラスタを追加する場合、またはAstra Control Provisionerのインストールとアップグレードを自分で管理する場合は、クラスタでAstra Control Provisionerを有効にします。を参照してください ["Astra Control Provisionerを有効にする"](#) を参照してください。

2. ブラウザでAstra Control Service Web UIを開きます。
3. ダッシュボードで、* Kubernetes クラスタの管理 * を選択します。

プロンプトに従ってクラスタを追加します。

4. プロバイダ：クラウドプロバイダを選択し、新しいクラウドインスタンスを作成するために必要なクレデンシャルを入力するか、使用する既存のクラウドインスタンスを選択します。
5. * Amazon Web Services *：JSONファイルをアップロードするか、クリップボードからそのJSONファイルの内容を貼り付けて、Amazon Web Services IAMユーザアカウントの詳細を指定します。

JSONファイルに、クラスタを作成したIAMユーザのクレデンシャルを含める必要があります。

6. * Microsoft Azure *：JSON ファイルをアップロードするか、クリップボードからその JSON ファイルの内容を貼り付けて、Azure サービスプリンシパルの詳細を指定します。

JSON ファイルに、サービスプリンシパルの作成時に Azure CLI からの出力が含まれている必要があります。サブスクリプション ID を含めることで、Astra に自動的に追加されるようにすることもできます。ID が必要な場合は、JSON の指定後に手動で ID を入力する必要があります。

7. * Google Cloud Platform *：ファイルをアップロードするか、クリップボードからコンテンツを貼り付けて、サービスアカウントのキーファイルを提供します。

Astra Control Service は、サービスアカウントを使用して、Google Kubernetes Engine で実行されているクラスタを検出します。

8. その他：このタブは自己管理クラスタでのみ使用できます。
 - a. クラウドインスタンス名：このクラスタを追加するときに作成される新しいクラウドインスタンスの名前を指定します。の詳細を確認してください "[クラウドインスタンス](#)"。
 - b. 「* 次へ *」を選択します。

Astra Control Serviceに、クラスタのリストが表示されます。

- c. クラスタ：Astra Control Serviceに追加するクラスタをリストから選択します。



クラスタのリストから選択する場合は、* Eligibility *列に注意してください。クラスタが「ineligible」または「partially eligible」の場合は、ステータスにカーソルを合わせて、そのクラスタに問題があるかどうかを確認します。たとえば、クラスタにワーカーノードがないことがわかります。

- d. 「* 次へ *」を選択します。
 - e. (オプション) * Storage *：必要に応じて、このクラスタに導入されたKubernetesアプリケーションでデフォルトで使用するストレージクラスを選択します。
9. クラスタの新しいデフォルトのストレージクラスを選択するには、*[新しいデフォルトのストレージクラスを割り当てる]*チェックボックスを有効にします。
 10. 新しいデフォルトのストレージクラスをリストから選択します。

各クラウドプロバイダのストレージサービスには、コスト、パフォーマンス、耐障害性に関する次の情報が表示されます。



- Cloud Volumes Service for Google Cloud：価格、パフォーマンス、耐障害性に関する情報
- Google Persistent Disk：コスト、パフォーマンス、耐障害性に関する情報は提供されません
- Azure NetApp Files：パフォーマンスと耐障害性に関する情報
- Azure Managed Disks：価格、パフォーマンス、耐障害性に関する情報は提供されません
- Amazon Elastic Block Store：価格、パフォーマンス、耐障害性に関する情報がない
- Amazon FSX for NetApp ONTAP：価格、パフォーマンス、耐障害性に関する情報は提供されません
- NetApp Cloud Volumes ONTAP：価格、パフォーマンス、耐障害性に関する情報は提供されません

ストレージクラスごとに、次のいずれかのサービスを利用できます。

- ["Cloud Volumes Service for Google Cloud"](#)
- ["Google Persistent Disk のことです"](#)
 - ["Azure NetApp Files の特長"](#)
 - ["Azure で管理されるディスク"](#)
 - ["Amazon Elastic Block Store"](#)
 - ["NetApp ONTAP 対応の Amazon FSX"](#)
 - ["NetApp Cloud Volumes ONTAP の略"](#)

の詳細を確認してください ["Amazon Web Services クラスタのストレージクラス"](#)。の詳細を確認してください ["AKS クラスタのストレージクラス"](#)。の詳細を確認してください ["GKE クラスタのストレージクラス"](#)。

- 「* 次へ *」を選択します。
- 確認と承認：構成の詳細を確認します。
- [Add]*を選択して、Astra Control Serviceにクラスタを追加します。

結果

このクラウドプロバイダ用に最初に追加したクラスタである場合、Astra Control Serviceは、対象となるクラスタで実行されているアプリケーションのバックアップ用に、クラウドプロバイダ用のオブジェクトストアを作成します。（このクラウドプロバイダに対して後続のクラスタを追加した場合、オブジェクトストアは作成されません）。デフォルトのストレージクラスを指定した場合は、指定したデフォルトのストレージクラスがAstra Control Serviceによって設定されます。Amazon Web ServicesまたはGoogle Cloud Platformで管理されるクラスタの場合、Astra Control Serviceはクラスタにも管理者アカウントを作成します。この処理には数分かかることがあります。

デフォルトのストレージクラスを変更する

クラスタのデフォルトのストレージクラスは変更できます。

Astra Controlを使用してデフォルトのストレージクラスを変更する

クラスタのデフォルトのストレージクラスは、Astra Control内から変更できます。以前にインストールしたストレージバックエンドサービスをクラスタで使用している場合は、このメソッドを使用してデフォルトのストレージクラスを変更できない可能性があります（*デフォルトに設定*アクションは選択できません）。この場合は、を実行できます [\[コマンドラインを使用してデフォルトのストレージクラスを変更します\]](#)。

手順

1. Astra Control Service UI で、[* Clusters] を選択します。
2. [* Clusters]ページで、変更するクラスタを選択します。
3. [* ストレージ *] タブを選択します。
4. 「ストレージクラス」カテゴリを選択します。
5. デフォルトとして設定するストレージクラスの* Actions *メニューを選択します。
6. 「デフォルトに設定」を選択します。

コマンドラインを使用してデフォルトのストレージクラスを変更します

Kubernetesコマンドを使用してクラスタのデフォルトのストレージクラスを変更することができます。この方法は、クラスタの構成に関係なく機能します。

手順

1. Kubernetesクラスタにログインします。
2. クラスタ内のストレージクラスを表示します。

```
kubectl get storageclass
```

3. デフォルトのストレージクラスからデフォルトの指定を削除する。<SC_NAME> をストレージクラスの名前に置き換えます。

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

4. 別のストレージクラスをデフォルトとしてマークします。<SC_NAME> をストレージクラスの名前に置き換えます。

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

5. 新しいデフォルトストレージクラスを確認します。

```
kubectl get storageclass
```

プロバイダが管理するプライベートクラスタを**Astra Control Service**に追加します

Astra Control Serviceを使用して、Google Kubernetes Engine (GKE) のプライベートクラスタを管理できます。ここでは、プライベートAKSまたはOpenShiftクラスタがすでに作成されており、リモートからアクセスするための安全な方法が用意されていることを前提としています。プライベートAKSまたはOpenShiftクラスタの作成とアクセスの詳細については、次のドキュメントを参照してください。

- ["プライベートAKSクラスタに関するAzureのドキュメント"](#)
- ["プライベートOpenShiftクラスタに関するAzureのドキュメント"](#)

Astra Control Serviceを使用して、AKSでプライベートAzure Kubernetes Service (AKS) クラスタとプライベートRed Hat OpenShiftクラスタを管理できます。ここでは、プライベートAKSまたはOpenShiftクラスタがすでに作成されており、リモートからアクセスするための安全な方法が用意されていることを前提としています。プライベートAKSまたはOpenShiftクラスタの作成とアクセスの詳細については、次のドキュメントを参照してください。

- ["プライベートAKSクラスタに関するAzureのドキュメント"](#)
- ["プライベートOpenShiftクラスタに関するAzureのドキュメント"](#)

Astra Control Serviceを使用して、プライベートAmazon Elastic Kubernetes Service (EKS) クラスタを管理できます。ここで説明する手順では、プライベートEKSクラスタがすでに作成されており、リモートからアクセスするためのセキュアな方法が用意されていることを前提としています。プライベートEKSクラスタの作成とアクセスの詳細については、["Amazon EKSのドキュメント"](#)を参照してください。

Astra Control Serviceにプライベートクラスタを追加するには、次のタスクを実行する必要があります。

1. [Astra Connectorのインストール](#)
2. [\[永続的ストレージをセットアップする\]](#)
3. [プロバイダが管理するプライベートクラスタをAstra Control Serviceに追加](#)

Astra Connectorのインストール

プライベートクラスタを追加する前に、Astra Controlがクラスタと通信できるように、クラスタにAstra Connectorをインストールする必要があります。を参照してください ["Kubernetesネイティブではないワークフローで管理されるプライベートクラスタ用に、以前のバージョンのAstra Connectorをインストール"](#) 手順については、[を参照し](#)

永続的ストレージをセットアップする

クラスタに永続的ストレージを設定してください。永続的ストレージの設定の詳細については、『[Get Started](#)』ドキュメントを参照してください。

- ["Azure NetApp Files を使用して Microsoft Azure をセットアップする"](#)
- ["Azure で管理されているディスクを使用して Microsoft Azure をセットアップする"](#)
- ["Amazon Web Servicesをセットアップする"](#)
- ["Google Cloud をセットアップします"](#)

プロバイダが管理するプライベートクラスタを**Astra Control Service**に追加

プライベートクラスタをAstra Control Serviceに追加できるようになりました。

Azure Kubernetes ServiceクラスタとGoogle Kubernetes Engineクラスタを管理する場合、Astra Control Provisionerのインストールとライフサイクル管理には次の2つのオプションがあることに注意してください。

- Astra Control Serviceを使用すると、Astra Control Provisionerのライフサイクルを自動的に管理できます。そのためには、Astra Control Serviceで管理するクラスタでAstra Tridentがインストールされておらず、Astra Control Provisionerが有効になっていないことを確認してください。この場合、クラスタの管理を開始すると、Astra Control ServiceによってAstra Control Provisionerが自動的に有効になり、Astra Control Provisionerのアップグレードは自動的に処理されます。
- Astra Control Provisionerのライフサイクルは自分で管理できます。これを行うには、Astra Control Serviceでクラスタを管理する前に、クラスタでAstra Control Provisionerを有効にします。この場合、Astra Control ServiceはAstra Control Provisionerがすでに有効になっていることを検出し、再インストールやAstra Control Provisionerのアップグレードの管理は行いません。を参照してください ["Astra Control Provisionerを有効にする"](#) では、Astra Control Provisionerを有効にします。

Astra Control ServiceでAmazon Web Servicesクラスタを管理する場合、Astra Control Provisionerでのみ使用できるストレージバックエンドが必要な場合は、Astra Control Serviceで管理する前に、クラスタでAstra Control Provisionerを手動で有効にする必要があります。を参照してください ["Astra Control Provisionerを有効にする"](#) をクリックし、Astra Control Provisionerを有効にします。

Amazon Web Services の

- クラスタを作成したIAMユーザのクレデンシャルを含むJSONファイルが必要です。"[IAMユーザを作成する方法を説明します](#)"。
- Amazon FSx for NetApp ONTAPにはAstra Controlプロビジョニングツールが必要です。EKSクラスタのストレージバックエンドとしてAmazon FSx for NetApp ONTAPを使用する予定の場合は、"[EKSクラスタ要件](#)"。
- (オプション) を指定する必要がある場合 `kubectl` クラスタの作成者ではない他のIAMユーザにクラスタにアクセスするためのコマンドアクセスについては、[手順を参照してください](#) "[Amazon EKSでクラスタを作成したあと、他のIAMユーザとロールにアクセスを提供するにはどうすればよいですか?](#)"。
- NetApp Cloud Volumes ONTAP をストレージバックエンドとして使用する場合は、Amazon Web Servicesと連携するようにCloud Volumes ONTAP を設定する必要があります。Cloud Volumes ONTAP を参照してください "[セットアップのドキュメント](#)"。

Microsoft Azure

- サービスプリンシパルの作成時に、Azure CLIからの出力を含むJSONファイルが必要です。"[サービスプリンシパルの設定方法について説明します](#)"。

JSON ファイルに追加していない場合は、Azure サブスクリプション ID も必要になります。

- NetApp Cloud Volumes ONTAP をストレージバックエンドとして使用する場合は、Microsoft Azure と連携するようにCloud Volumes ONTAP を設定する必要があります。Cloud Volumes ONTAP を参照してください "[セットアップのドキュメント](#)"。

Google Cloud

- 必要な権限を持つサービスアカウントのサービスアカウントキーファイルが必要です。"[サービスアカウントの設定方法について説明します](#)"。
- クラスタがプライベートの場合は、[を参照してください](#) "[許可されたネットワーク](#)" Astra Control Service の IP アドレスを許可する必要があります。

52.188.218.166-32

- NetApp Cloud Volumes ONTAP をストレージバックエンドとして使用する場合は、Google Cloudと連携するようにCloud Volumes ONTAP を設定する必要があります。Cloud Volumes ONTAP を参照してください "[セットアップのドキュメント](#)"。

手順

1. (オプション) Amazon EKSクラスタを追加する場合、またはAstra Control Provisionerのインストールとアップグレードを自分で管理する場合は、クラスタでAstra Control Provisionerを有効にします。[を参照してください](#) "[Astra Control Provisionerを有効にする](#)" を参照してください。

2. ブラウザでAstra Control Service Web UIを開きます。

3. ダッシュボードで、* Kubernetes クラスタの管理 * を選択します。

プロンプトに従ってクラスタを追加します。

4. プロバイダ：クラウドプロバイダを選択し、新しいクラウドインスタンスを作成するために必要なクレデ

ンシャルを入力するか、使用する既存のクラウドインスタンスを選択します。

5. * Amazon Web Services * : JSON ファイルをアップロードするか、クリップボードからその JSON ファイルの内容を貼り付けて、Amazon Web Services IAM ユーザアカウントの詳細を指定します。

JSON ファイルに、クラスタを作成した IAM ユーザのクレデンシャルを含める必要があります。

6. * Microsoft Azure * : JSON ファイルをアップロードするか、クリップボードからその JSON ファイルの内容を貼り付けて、Azure サービスプリンシパルの詳細を指定します。

JSON ファイルに、サービスプリンシパルの作成時に Azure CLI からの出力が含まれている必要があります。サブスクリプション ID を含めることで、Astra に自動的に追加されるようにすることもできます。ID が必要な場合は、JSON の指定後に手動で ID を入力する必要があります。

7. * Google Cloud Platform * : ファイルをアップロードするか、クリップボードからコンテンツを貼り付けて、サービスアカウントのキーファイルを提供します。

Astra Control Service は、サービスアカウントを使用して、Google Kubernetes Engine で実行されているクラスタを検出します。

8. その他 : このタブは自己管理クラスタでのみ使用できます。

- a. クラウドインスタンス名 : このクラスタを追加するときに作成される新しいクラウドインスタンスの名前を指定します。の詳細を確認してください ["クラウドインスタンス"](#)。
- b. 「* 次へ *」を選択します。

Astra Control Service に、クラスタのリストが表示されます。

- c. クラスタ : Astra Control Service に追加するクラスタをリストから選択します。



クラスタのリストから選択する場合は、* Eligibility * 列に注意してください。クラスタが「ineligible」または「partially eligible」の場合は、ステータスにカーソルを合わせて、そのクラスタに問題があるかどうかを確認します。たとえば、クラスタにワーカーノードがないことがわかります。

9. 「* 次へ *」を選択します。
10. (オプション) * Storage * : 必要に応じて、このクラスタに導入された Kubernetes アプリケーションでデフォルトで使用するストレージクラスを選択します。
 - a. クラスタの新しいデフォルトのストレージクラスを選択するには、*[新しいデフォルトのストレージクラスを割り当てる]*チェックボックスを有効にします。
 - b. 新しいデフォルトのストレージクラスをリストから選択します。

各クラウドプロバイダのストレージサービスには、コスト、パフォーマンス、耐障害性に関する次の情報が表示されます。



- Cloud Volumes Service for Google Cloud：価格、パフォーマンス、耐障害性に関する情報
- Google Persistent Disk：コスト、パフォーマンス、耐障害性に関する情報は提供されません
- Azure NetApp Files：パフォーマンスと耐障害性に関する情報
- Azure Managed Disks：価格、パフォーマンス、耐障害性に関する情報は提供されません
- Amazon Elastic Block Store：価格、パフォーマンス、耐障害性に関する情報がない
- Amazon FSX for NetApp ONTAP：価格、パフォーマンス、耐障害性に関する情報は提供されません
- NetApp Cloud Volumes ONTAP：価格、パフォーマンス、耐障害性に関する情報は提供されません

ストレージクラスごとに、次のいずれかのサービスを利用できます。

- ["Cloud Volumes Service for Google Cloud"](#)
- ["Google Persistent Disk のことです"](#)
- ["Azure NetApp Files の特長"](#)
- ["Azure で管理されるディスク"](#)
- ["Amazon Elastic Block Store"](#)
- ["NetApp ONTAP 対応の Amazon FSX"](#)
- ["NetApp Cloud Volumes ONTAP の略"](#)

の詳細を確認してください ["Amazon Web Services クラスタのストレージクラス"](#)。の詳細を確認してください ["AKS クラスタのストレージクラス"](#)。の詳細を確認してください ["GKE クラスタのストレージクラス"](#)。

- c. 「* 次へ *」を選択します。
- d. 確認と承認：構成の詳細を確認します。
- e. [Add]*を選択して、Astra Control Serviceにクラスタを追加します。

結果

このクラウドプロバイダ用に最初に追加したクラスタである場合、Astra Control Serviceは、対象となるクラスタで実行されているアプリケーションのバックアップ用に、クラウドプロバイダ用のオブジェクトストアを作成します。（このクラウドプロバイダに対して後続のクラスタを追加した場合、オブジェクトストアは作成されません）。デフォルトのストレージクラスを指定した場合は、指定したデフォルトのストレージクラスがAstra Control Serviceによって設定されます。Amazon Web ServicesまたはGoogle Cloud Platformで管理されるクラスタの場合、Astra Control Serviceはクラスタにも管理者アカウントを作成します。この処理には数分かかることがあります。

デフォルトのストレージクラスを変更する

クラスタのデフォルトのストレージクラスは変更できます。

Astra Controlを使用してデフォルトのストレージクラスを変更する

クラスタのデフォルトのストレージクラスは、Astra Control内から変更できます。以前にインストールしたストレージバックエンドサービスをクラスタで使用している場合は、このメソッドを使用してデフォルトのストレージクラスを変更できない可能性があります（*デフォルトに設定*アクションは選択できません）。この場合は、を実行できます [\[コマンドラインを使用してデフォルトのストレージクラスを変更します\]](#)。

手順

1. Astra Control Service UI で、[* Clusters] を選択します。
2. [* Clusters]ページで、変更するクラスタを選択します。
3. [* ストレージ*] タブを選択します。
4. 「ストレージクラス」カテゴリを選択します。
5. デフォルトとして設定するストレージクラスの* Actions *メニューを選択します。
6. 「デフォルトに設定」を選択します。

コマンドラインを使用してデフォルトのストレージクラスを変更します

Kubernetesコマンドを使用してクラスタのデフォルトのストレージクラスを変更することができます。この方法は、クラスタの構成に関係なく機能します。

手順

1. Kubernetesクラスタにログインします。
2. クラスタ内のストレージクラスを表示します。

```
kubectl get storageclass
```

3. デフォルトのストレージクラスからデフォルトの指定を削除する。<SC_NAME> をストレージクラスの名前に置き換えます。

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

4. 別のストレージクラスをデフォルトとしてマークします。<SC_NAME> をストレージクラスの名前に置き換えます。

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

5. 新しいデフォルトストレージクラスを確認します。

```
kubectl get storageclass
```

自己管理型クラスタを追加します

Astra Control Serviceにパブリック自己管理型クラスタを追加

環境のセットアップが完了したら、Kubernetes クラスタを作成し、Astra Control Service に追加することができます。

自己管理型クラスタは、ユーザが直接プロビジョニングおよび管理するクラスタです。Astra Control Service は、パブリッククラウド環境で実行される自己管理型クラスタをサポートします。をアップロードすることで、Astra Control Serviceに自己管理型クラスタを追加できます kubeconfig.yaml ファイル。クラスタがここで説明する要件を満たしていることを確認する必要があります。

サポートされている**Kubernetes**ディストリビューション

Astra Control Serviceを使用して、次のタイプのパブリック自己管理クラスタを管理できます。

Kubernetesディストリビューション	サポートされるバージョン
Kubernetes（アップストリーム）	1.27～1.29
Rancher Kubernetes Engine（RKE）	RKE 1：バージョン1.24.17、1.25.13、1.26.8 （Rancher Manager 2.7.9を使用） RKE 2：Rancher Manager 2.6.13を使用したバージョン1.23.16および1.24.13 RKE 2：バージョン1.24.17、1.25.14、1.26.9 （Rancher Manager 2.7.9を使用）
Red Hat OpenShift Container Platform	4.12～4.14

ここで説明する手順は、自己管理型クラスタがすでに作成されていることを前提としています。

- [Astra Control Serviceにクラスタを追加](#)
- [\[デフォルトのストレージクラスを変更する\]](#)

Astra Control Serviceにクラスタを追加

Astra Control Service にログインしたら、最初にクラスタの管理を開始します。Astra Control Serviceにクラスタを追加する前に、特定のタスクを実行し、クラスタが一定の要件を満たしていることを確認する必要があります。

自己管理型クラスタは、ユーザが直接プロビジョニングおよび管理するクラスタです。Astra Control Serviceは、パブリッククラウド環境で実行される自己管理型クラスタをサポートします。自己管理型クラスタでは、Astra Control Provisionerを使用してNetAppストレージサービス进行操作したり、Container Storage Interface (CSI) ドライバを使用してAmazon Elastic Block Store (EBS)、Azure Managed Disks、Google Persistent Disk进行操作したりできます。

Astra Control Serviceは、次のKubernetesディストリビューションを使用する自己管理クラスタをサポートします。

- Red Hat OpenShift Container Platform
- Rancher Kubernetes Engineの略
- アップストリームKubernetes

自己管理型クラスタは、次の要件を満たしている必要があります。

- クラスタにインターネット経由でアクセスできる必要があります。
- CSIドライバで有効にしたストレージを使用または使用する予定の場合は、適切なCSIドライバをクラスタにインストールする必要があります。CSIドライバを使用してストレージを統合する方法の詳細については、ご使用のストレージサービスのマニュアルを参照してください。
- context要素を1つだけ含むcluster kubeconfigファイルにアクセスできる必要があります。をクリックします ["以下の手順を参照して"](#) kubeconfigファイルを生成します。
- プライベート認証局 (CA) を参照するkubeconfigファイルを使用してクラスタを追加する場合は、cluster kubeconfigファイルのセクションを参照してください。これにより、Astra Controlでクラスタを追加できます。

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

- **rancherのみ:** Rancher環境でアプリケーションクラスタを管理する場合、rancherから提供されたkubeconfigファイルでアプリケーションクラスタのデフォルトコンテキストを変更して、rancher APIサーバコンテキストではなくコントロールプレーンコンテキストを使用します。これにより、Rancher API サーバの負荷が軽減され、パフォーマンスが向上します。
- *** Astra Control Provisionerの要件***：クラスタを管理するには、Astra Tridentコンポーネントを含むAstra Control Provisionerを適切に設定する必要があります。
 - *** Astra Trident環境要件の確認***：Astra Control Provisionerをインストールまたはアップグレードする前に、["サポートされるフロントエンド、バックエンド、およびホスト構成"](#)。
 - *** Astra Control Provisioner機能を有効にする***：Astra Trident 23.10以降をインストールして有効にすることを強く推奨します。["Astra Control Provisionerの高度なストレージ機能"](#)。今後のリリースでは、Astra Control Provisionerが有効になっていない場合、Astra ControlはAstra Tridentをサポートしません。
 - **ストレージバックエンドの構成**:少なくとも1つのストレージバックエンドが ["Astra Tridentで設定"](#) クラスタのポリシーを確認してください。
 - **ストレージクラスの設定**：少なくとも1つのストレージクラスが ["Astra Tridentで設定"](#) クラスタのポリシーを確認してください。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ます。

- 。ボリュームスナップショットコントローラを設定し、ボリュームスナップショットクラスをインストールする： ["ボリュームSnapshotコントローラのインストール"](#) Astra Controlでスナップショットを作成できるようにします。 ["作成"](#) 1つ以上 VolumeSnapshotClass Astra Tridentを使用

手順

1. ダッシュボードで、 [* Kubernetes クラスタの管理 *](#) を選択します。

プロンプトに従ってクラスタを追加します。

2. プロバイダ： [*\[その他\]*](#) タブを選択して、自己管理クラスタに関する詳細を追加します。
 - a. その他：をアップロードして、自己管理クラスタに関する詳細を指定します kubeconfig.yaml ファイルまたはの内容を貼り付けます kubeconfig.yaml クリップボードからファイル。



自分で作成する場合は kubeconfig ファイルには、 [* 1つの*コンテキストエレメント](#) のみを定義する必要があります。を参照してください ["Kubernetes のドキュメント"](#) を参照してください kubeconfig ファイル。

3. クレデンシャル名：Astra Controlにアップロードする自己管理型クラスタのクレデンシャルの名前を指定します。デフォルトでは、クレデンシャル名がクラスタの名前として自動的に入力されます。
4. プライベートルート識別子：このフィールドはプライベートクラスタでのみ使用できます。
5. [「* 次へ *](#)」を選択します。
6. (オプション) [* Storage *](#)：必要に応じて、このクラスタに導入されたKubernetesアプリケーションでデフォルトで使用するストレージクラスを選択します。
 - a. クラスタの新しいデフォルトのストレージクラスを選択するには、 [*\[新しいデフォルトのストレージクラスを割り当てる\]*](#) チェックボックスを有効にします。
 - b. 新しいデフォルトのストレージクラスをリストから選択します。

各クラウドプロバイダのストレージサービスには、コスト、パフォーマンス、耐障害性に関する次の情報が表示されます。



- Cloud Volumes Service for Google Cloud：価格、パフォーマンス、耐障害性に関する情報
- Google Persistent Disk：コスト、パフォーマンス、耐障害性に関する情報は提供されません
- Azure NetApp Files：パフォーマンスと耐障害性に関する情報
- Azure Managed Disks：価格、パフォーマンス、耐障害性に関する情報は提供されません
- Amazon Elastic Block Store：価格、パフォーマンス、耐障害性に関する情報がない
- Amazon FSX for NetApp ONTAP：価格、パフォーマンス、耐障害性に関する情報は提供されません
- NetApp Cloud Volumes ONTAP：価格、パフォーマンス、耐障害性に関する情報は提供されません

ストレージクラスごとに、次のいずれかのサービスを利用できます。

- ["Cloud Volumes Service for Google Cloud"](#)
- ["Google Persistent Disk のことです"](#)
 - ["Azure NetApp Files の特長"](#)
 - ["Azure で管理されるディスク"](#)
 - ["Amazon Elastic Block Store"](#)
 - ["NetApp ONTAP 対応の Amazon FSX"](#)
 - ["NetApp Cloud Volumes ONTAP の略"](#)

の詳細を確認してください ["Amazon Web Services クラスタのストレージクラス"](#)。の詳細を確認してください ["AKS クラスタのストレージクラス"](#)。の詳細を確認してください ["GKE クラスタのストレージクラス"](#)。

- c. 「* 次へ *」を選択します。
- d. 確認と承認：構成の詳細を確認します。
- e. [Add]*を選択して、Astra Control Serviceにクラスタを追加します。

デフォルトのストレージクラスを変更する

クラスタのデフォルトのストレージクラスは変更できます。

Astra Controlを使用してデフォルトのストレージクラスを変更する

クラスタのデフォルトのストレージクラスは、Astra Control内から変更できます。以前にインストールしたストレージバックエンドサービスをクラスタで使用している場合は、このメソッドを使用してデフォルトのストレージクラスを変更できない可能性があります（*デフォルトに設定*アクションは選択できません）。この場合は、を実行できます [\[コマンドラインを使用してデフォルトのストレージクラスを変更します\]](#)。

手順

1. Astra Control Service UI で、[* Clusters] を選択します。
2. [* Clusters]ページで、変更するクラスタを選択します。
3. [* ストレージ *] タブを選択します。
4. 「ストレージクラス」カテゴリを選択します。
5. デフォルトとして設定するストレージクラスの* Actions *メニューを選択します。
6. 「デフォルトに設定」を選択します。

コマンドラインを使用してデフォルトのストレージクラスを変更します

Kubernetesコマンドを使用してクラスタのデフォルトのストレージクラスを変更することができます。この方法は、クラスタの構成に関係なく機能します。

手順

1. Kubernetesクラスタにログインします。
2. クラスタ内のストレージクラスを表示します。


```
kubectl get storageclass
```

3. デフォルトのストレージクラスからデフォルトの指定を削除する。<SC_NAME> をストレージクラスの名前に置き換えます。

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

4. 別のストレージクラスをデフォルトとしてマークします。<SC_NAME> をストレージクラスの名前に置き換えます。

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

5. 新しいデフォルトストレージクラスを確認します。

```
kubectl get storageclass
```

Astra Control Serviceに自己管理型プライベートクラスタを追加

環境のセットアップが完了したら、Kubernetes クラスタを作成し、Astra Control Service に追加することができます。

自己管理型クラスタは、ユーザが直接プロビジョニングおよび管理するクラスタです。Astra Control Service は、パブリッククラウド環境で実行される自己管理型クラスタをサポートします。をアップロードすることで、Astra Control Serviceに自己管理型クラスタを追加できます kubeconfig.yaml ファイル。クラスタがここで説明する要件を満たしていることを確認する必要があります。

サポートされている**Kubernetes**ディストリビューション

Astra Control Serviceを使用して、次のタイプの自己管理型プライベートクラスタを管理できます。

Kubernetesディストリビューション	サポートされるバージョン
Kubernetes（アップストリーム）	1.27～1.29
Rancher Kubernetes Engine（RKE）	RKE 1：バージョン1.24.17、1.25.13、1.26.8 （Rancher Manager 2.7.9を使用） RKE 2：Rancher Manager 2.6.13を使用したバージョン1.23.16および1.24.13 RKE 2：バージョン1.24.17、1.25.14、1.26.9 （Rancher Manager 2.7.9を使用）
Red Hat OpenShift Container Platform	4.12～4.14

ここで説明する手順は、すでにプライベートクラスタを作成し、リモートからアクセスするためのセキュアな方法を準備していることを前提としています。

Astra Control Serviceにプライベートクラスタを追加するには、次のタスクを実行する必要があります。

1. [Astra Connectorのインストール](#)
2. [\[永続的ストレージをセットアップする\]](#)
3. [Astra Control Serviceに自己管理型プライベートクラスタを追加](#)

Astra Connectorのインストール

プライベートクラスタを追加する前に、Astra Controlがクラスタと通信できるように、クラスタにAstra Connectorをインストールする必要があります。を参照してください ["Kubernetesネイティブではないワークフローで管理されるプライベートクラスタ用に、以前のバージョンのAstra Connectorをインストール"](#) 手順については、を参照し

永続的ストレージをセットアップする

クラスタに永続的ストレージを設定してください。永続的ストレージの設定の詳細については、『[Get Started](#)』ドキュメントを参照してください。

- ["Azure NetApp Files を使用して Microsoft Azure をセットアップする"](#)
- ["Azure で管理されているディスクを使用して Microsoft Azure をセットアップする"](#)
- ["Amazon Web Servicesをセットアップする"](#)
- ["Google Cloud をセットアップします"](#)

Astra Control Serviceに自己管理型プライベートクラスタを追加

プライベートクラスタをAstra Control Serviceに追加できるようになりました。

自己管理型クラスタは、ユーザが直接プロビジョニングおよび管理するクラスタです。Astra Control Serviceは、パブリッククラウド環境で実行される自己管理型クラスタをサポートします。自己管理型クラスタでは、Astra Control Provisionerを使用してNetAppストレージサービス进行操作したり、Container Storage Interface (CSI) ドライバを使用してAmazon Elastic Block Store (EBS)、Azure Managed Disks、Google Persistent Disk进行操作したりできます。

Astra Control Serviceは、次のKubernetesディストリビューションを使用する自己管理クラスタをサポートします。

- Red Hat OpenShift Container Platform
- Rancher Kubernetes Engineの略
- アップストリームKubernetes

自己管理型クラスタは、次の要件を満たしている必要があります。

- クラスタにインターネット経由でアクセスできる必要があります。
- CSIドライバで有効にしたストレージを使用または使用する予定の場合は、適切なCSIドライバをクラスタにインストールする必要があります。CSIドライバを使用してストレージを統合する方法の詳細については、ご使用のストレージサービスのマニュアルを参照してください。
- context要素を1つだけ含むcluster kubeconfigファイルにアクセスできる必要があります。をクリックします ["以下の手順を参照して"](#) kubeconfigファイルを生成します。
- プライベート認証局 (CA) を参照するkubeconfigファイルを使用してクラスタを追加する場合は、cluster kubeconfigファイルのセクションを参照してください。これにより、Astra Controlでクラスタを追加できます。

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

- **rancherのみ:** Rancher環境でアプリケーションクラスタを管理する場合、rancherから提供されたkubeconfigファイルでアプリケーションクラスタのデフォルトコンテキストを変更して、rancher APIサーバコンテキストではなくコントロールプレーンコンテキストを使用します。これにより、Rancher API サーバの負荷が軽減され、パフォーマンスが向上します。
- *** Astra Control Provisionerの要件***：クラスタを管理するには、Astra Tridentコンポーネントを含むAstra Control Provisionerを適切に設定する必要があります。
 - *** Astra Trident環境要件の確認***：Astra Control Provisionerをインストールまたはアップグレードする前に、["サポートされるフロントエンド、バックエンド、およびホスト構成"](#)。
 - *** Astra Control Provisioner機能を有効にする***：Astra Trident 23.10以降をインストールして有効にすることを強く推奨します。["Astra Control Provisionerの高度なストレージ機能"](#)。今後のリリースでは、Astra Control Provisionerが有効になっていない場合、Astra ControlはAstra Tridentをサポートしません。
 - **ストレージバックエンドの構成**:少なくとも1つのストレージバックエンドが ["Astra Tridentで設定"](#) クラスタのポリシーを確認してください。
 - **ストレージクラスの設定**：少なくとも1つのストレージクラスが ["Astra Tridentで設定"](#) クラスタのポリシーを確認してください。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ます。

- 。ボリュームスナップショットコントローラを設定し、ボリュームスナップショットクラスをインストールする： ["ボリュームSnapshotコントローラのインストール"](#) Astra Controlでスナップショットを作成できるようにします。 ["作成"](#) 1つ以上 VolumeSnapshotClass Astra Tridentを使用：

手順

1. ダッシュボードで、 [* Kubernetes クラスタの管理 *](#) を選択します。

プロンプトに従ってクラスタを追加します。

2. プロバイダ： [*\[その他\]*](#) タブを選択して、自己管理クラスタに関する詳細を追加します。
3. その他： をアップロードして、自己管理クラスタに関する詳細を指定します kubeconfig.yaml ファイルまたはの内容を貼り付けます kubeconfig.yaml クリップボードからファイル。



自分で作成する場合は kubeconfig ファイルには、 [* 1つの*コンテキストエレメントのみ](#) を定義する必要があります。を参照してください ["以下の手順を参照して"](#) を参照してください kubeconfig ファイル。

4. クレデンシャル名： Astra Controlにアップロードする自己管理型クラスタのクレデンシャルの名前を指定します。デフォルトでは、クレデンシャル名がクラスタの名前として自動的に入力されます。
5. プライベートルート識別子： Astra Connectorから取得できるプライベートルート識別子を入力します。を使用してAstra Connectorを照会した場合 `kubectl get astraconnector -n astra-connector` プライベートルート識別子と呼ばれます `ASTRACONNECTORID`。



プライベートルート識別子は、AstraでプライベートKubernetesクラスタを管理できるようにするAstra Connectorに関連付けられた名前です。この場合、プライベートクラスタは、APIサーバをインターネットに公開しないKubernetesクラスタです。

6. [「* 次へ *](#)」を選択します。
7. (オプション) [* Storage *](#)：必要に応じて、このクラスタに導入されたKubernetesアプリケーションでデフォルトで使用するストレージクラスを選択します。
 - a. クラスタの新しいデフォルトのストレージクラスを選択するには、 [*\[新しいデフォルトのストレージクラスを割り当てる\]*](#) チェックボックスを有効にします。
 - b. 新しいデフォルトのストレージクラスをリストから選択します。

各クラウドプロバイダのストレージサービスには、コスト、パフォーマンス、耐障害性に関する次の情報が表示されます。



- Cloud Volumes Service for Google Cloud：価格、パフォーマンス、耐障害性に関する情報
- Google Persistent Disk：コスト、パフォーマンス、耐障害性に関する情報は提供されません
- Azure NetApp Files：パフォーマンスと耐障害性に関する情報
- Azure Managed Disks：価格、パフォーマンス、耐障害性に関する情報は提供されません
- Amazon Elastic Block Store：価格、パフォーマンス、耐障害性に関する情報がない
- Amazon FSX for NetApp ONTAP：価格、パフォーマンス、耐障害性に関する情報は提供されません
- NetApp Cloud Volumes ONTAP：価格、パフォーマンス、耐障害性に関する情報は提供されません

ストレージクラスごとに、次のいずれかのサービスを利用できます。

- ["Cloud Volumes Service for Google Cloud"](#)
- ["Google Persistent Disk のことです"](#)
- ["Azure NetApp Files の特長"](#)
- ["Azure で管理されるディスク"](#)
- ["Amazon Elastic Block Store"](#)
- ["NetApp ONTAP 対応の Amazon FSX"](#)
- ["NetApp Cloud Volumes ONTAP の略"](#)

の詳細を確認してください ["Amazon Web Services クラスタのストレージクラス"](#)。の詳細を確認してください ["AKS クラスタのストレージクラス"](#)。の詳細を確認してください ["GKE クラスタのストレージクラス"](#)。

- c. 「* 次へ *」を選択します。
- d. 確認と承認：構成の詳細を確認します。
- e. [Add]*を選択して、Astra Control Serviceにクラスタを追加します。

デフォルトのストレージクラスを変更する

クラスタのデフォルトのストレージクラスは変更できます。

Astra Controlを使用してデフォルトのストレージクラスを変更する

クラスタのデフォルトのストレージクラスは、Astra Control内から変更できます。以前にインストールしたストレージバックエンドサービスをクラスタで使用している場合は、このメソッドを使用してデフォルトのストレージクラスを変更できない可能性があります（*デフォルトに設定*アクションは選択できません）。この場合は、を実行できます [\[コマンドラインを使用してデフォルトのストレージクラスを変更します\]](#)。

手順

1. Astra Control Service UI で、[* Clusters] を選択します。
2. [* Clusters] ページで、変更するクラスタを選択します。
3. [* ストレージ*] タブを選択します。
4. 「ストレージクラス」カテゴリを選択します。
5. デフォルトとして設定するストレージクラスの* Actions *メニューを選択します。
6. 「デフォルトに設定」を選択します。

コマンドラインを使用してデフォルトのストレージクラスを変更します

Kubernetes コマンドを使用してクラスタのデフォルトのストレージクラスを変更することができます。この方法は、クラスタの構成に関係なく機能します。

手順

1. Kubernetes クラスタにログインします。
2. クラスタ内のストレージクラスを表示します。

```
kubectl get storageclass
```

3. デフォルトのストレージクラスからデフォルトの指定を削除する。<SC_NAME> をストレージクラスの名前に置き換えます。

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

4. 別のストレージクラスをデフォルトとしてマークします。<SC_NAME> をストレージクラスの名前に置き換えます。

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

5. 新しいデフォルトストレージクラスを確認します。

```
kubectl get storageclass
```

Astra Tridentのバージョンを確認

ストレージサービスにAstra Control ProvisionerまたはAstra Tridentを使用する自己管理型クラスタを追加するには、Astra Tridentのバージョンが23.10以降であることを確認し

てください。

手順

1. 実行しているAstra Tridentのバージョンを確認します。

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

Astra Tridentがインストールされている場合は、次のような出力が表示されます。

NAME	VERSION
trident	24.02.0

Astra Tridentがインストールされていない場合は、次のような出力が表示されます。

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

2. 次のいずれかを実行します。

- Astra Trident 23.01以前を実行している場合は、以下を使用 ["手順"](#) Astra Control Provisionerにアップグレードする前に、Astra Tridentの最新バージョンにアップグレードすること。可能です ["直接アップグレードを実行する"](#) Astra Tridentがバージョン24.02の4リリース期間内にある場合は、Astra Control Provisioner 24.02をダウンロードします。たとえば、Astra Trident 23.04からAstra Control Provisioner 24.02に直接アップグレードできます。
- Astra Trident 23.10以降を実行している場合は、Astra Control Provisionerが ["有効"](#)。Astra Control Provisionerは、23.10より前のリリースのAstra Control Centerでは機能しません。 ["Astra Control Provisionerのアップグレード"](#) 最新の機能にアクセスするには、アップグレードするAstra Control Centerと同じバージョンを使用する必要があります。

3. ポッドが実行されていることを確認します。

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

4. サポートされているAstra Tridentドライバをストレージクラスで使用しているかどうかを確認します。プロビジョニング担当者の名前はとします `csi.trident.netapp.io`。次の例を参照してください。

```
kubectl get sc
```

回答例：

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

kubeconfigファイルを作成します

kubeconfigファイルを使用して、Astra Control Serviceにクラスタを追加できます。追加するクラスタのタイプによっては、特定の手順を使用してクラスタ用のkubeconfigファイルを手動で作成しなければならない場合があります。

- [Amazon EKSクラスタ用のkubeconfigファイルを作成します](#)
- [Red Hat OpenShift Service on AWS \(ROSA\) クラスタ用のkubeconfigファイルを作成する](#)
- [\[他のタイプのクラスタ用にkubeconfigファイルを作成します\]](#)

Amazon EKSクラスタ用のkubeconfigファイルを作成します

以下の手順に従って、Amazon EKSクラスタ用のkubeconfigファイルと永続的トークンシークレットを作成します。EKSでホストされるクラスタには、永続的なトークンシークレットが必要です。

手順

1. Amazonのドキュメントの手順に従って、kubeconfigファイルを生成します。

["Amazon EKSクラスタ用のkubeconfigファイルを作成または更新します"](#)

2. 次の手順でサービスアカウントを作成します。

- a. という名前のサービスアカウントファイルを作成します `astracontrol-service-account.yaml`。

必要に応じてサービスアカウント名を調整します。ネームスペース `kube-system` これらの手順では必須です。ここでサービスアカウント名を変更する場合は、次の手順で同じ変更を適用する必要があります。

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

+

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

3. サービスアカウントを適用します。

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

4. を作成します ClusterRoleBinding という名前のファイルです astracontrol-clusterrolebinding.yaml。

astracontrol-clusterrolebinding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astra-admin-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: astra-admin-account
  namespace: kube-system
```

5. クラスターロールバインドを適用します。

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

6. という名前のサービスアカウントトークンシークレットファイルを作成します astracontrol-secret.yaml。

astracontrol-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: astra-admin-account
  name: astra-admin-account
  namespace: kube-system
type: kubernetes.io/service-account-token
```


7. トークンシークレットを適用します。

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

8. トークンシークレットを取得します。

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

9. を交換します `user` 次の例に示すように、AWS EKS kubeconfigファイルのセクションでトークンを指定します。

```
user:  
token: k8s-aws-
```

v1.aHR0cHM6Ly9zdHMudXMtd2Vzdc0yLmFtYXpvbmF3cy5jb20vP0FjdGlrbjIHZXRdYWxsZ
XJJZGVudGll0eSZWZXJzaW9uPTIwMTetMDYTMTUmWC1BbXotQWxnbn3JpdGhtPUFXUzQtSElBQ
ylTSEEyNTYmWC1BbXotQ3JlZGVudGllhbDlBS0lBM1JEWDdkU0haWU9LSEQ2SyUYrJIWMjMWN
DAzJTJGdXMtd2Vzdc0yJTJGc3RzJTJGYXdzNF9yZXF1ZnN0JlgtQW16LURhdGU9MjAyMzM0M
DNUMjaA0MzQwWiZYLUFTeilFeHBpcmVzPTYwJlgtQW16LVNPZ25lZEhlYWRLcnM9aG9zdCUzQ
ngtazhzLWF3cy1pZCZYLUFTeilTaWduYXRlcMU9YjU4ZWMOZndiM2NmZGRxNGRhbnZU4MGII2Z
WQ2zy2NZIZWIwm2UYNThjMJRhNTJjNmVhbjc4MTRlnJkOTg2MQ

Red Hat OpenShift Service on AWS (ROSA) クラスタ用のkubecconfigファイルを作成する

次の手順に従って、Red Hat OpenShift Service on AWS (ROSA) クラスタ用のkubeconfigファイルを作成します。

手順

1. ROSAクラスタにログインします。
2. サービスアカウントを作成します。

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

3. クラスターロールを追加します。

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

4. 次の例を使用して、サービスアカウントシークレットコンフィギュレーションファイルを作成します。

```
<strong>secret-astra-sa.yaml</strong>
```

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-astracontrol-service-account
  annotations:
    kubernetes.io/service-account.name: "astracontrol-service-account"
type: kubernetes.io/service-account-token
```

5. シークレットを作成します。

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

6. 作成したサービスアカウントを編集し、Astra Controlサービスアカウントのシークレット名を secrets セクション。

```
oc edit sa astracontrol-service-account
```

```
apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-dvfcd
kind: ServiceAccount
metadata:
  creationTimestamp: "2023-08-04T04:18:30Z"
  name: astracontrol-service-account
  namespace: default
  resourceVersion: "169770"
  uid: 965fa151-923f-4fbd-9289-30cad15998ac
secrets:
- name: astracontrol-service-account-dockercfg-dvfcd
- name: secret-astracontrol-service-account ####ADD THIS ONLY####
```

7. サービスアカウントのシークレットを一覧表示します（置き換えます） <CONTEXT> インストールに適したコンテキストを使用して、次の操作を行います。

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

出力の末尾は次のようになります。

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

内の各要素のインデックス `secrets` アレイは0から始まります。上記の例では、のインデックスです `astracontrol-service-account-dockercfg-dvfcfcd` は0、のインデックスです `secret-astracontrol-service-account` は1です。出力で、サービスアカウントシークレットのインデックス番号をメモします。このインデックス番号は次の手順で必要になります。

8. 次のように `kubeconfig` を生成します。

- a. を作成します `create-kubeconfig.sh` ファイル。交換してください `TOKEN_INDEX` 次のスクリプトの先頭に正しい値を入力します。

```
<strong>create-kubeconfig.sh</strong>
```

```
# Update these to match your environment.  
# Replace TOKEN_INDEX with the correct value  
# from the output in the previous step. If you  
# didn't change anything else above, don't change  
# anything else here.  
  
SERVICE_ACCOUNT_NAME=astracontrol-service-account  
NAMESPACE=default  
NEW_CONTEXT=astracontrol  
KUBECONFIG_FILE='kubeconfig-sa'  
  
CONTEXT=$(kubectl config current-context)  
  
SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \  
  --context ${CONTEXT} \  
  --namespace ${NAMESPACE} \  
  -o jsonpath='{.secrets[TOKEN_INDEX].name}')TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \  
  --context ${CONTEXT} \  
  --namespace ${NAMESPACE} \  
  -o jsonpath='{.data.token}')  
TOKEN=$(echo ${TOKEN_DATA} | base64 -d)  
  
# Create dedicated kubeconfig  
# Create a full copy  
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp
```

```

# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp

# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token
-user

# Set context to correct namespace
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}

# Flatten/minify kubeconfig
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  view --flatten --minify > ${KUBECONFIG_FILE}

# Remove tmp
rm ${KUBECONFIG_FILE}.full.tmp
rm ${KUBECONFIG_FILE}.tmp

```

- b. コマンドをソースにし、Kubernetes クラスタに適用します。

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

9. (オプション) クラスタにわかりやすい名前にコバーベキューの名前を変更します。

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

他のタイプのクラスタ用に**kubeconfig**ファイルを作成します

以下の手順に従って、Rancher、Upstream Kubernetes、およびRed Hat OpenShiftクラスタ用に、制限付きまたは拡張されたロールkubeconfigファイルを作成します。

kubeconfigを使用して管理されるクラスタについては、必要に応じて、Astra Control Service用の制限された権限または拡張された権限管理者ロールを作成できます。

この手順を使用すると、次のいずれかのシナリオで環境を環境化する場合に、別のkubeconfigを作成できます。

- 管理対象のクラスタに対するAstra Controlの権限を制限する
- 複数のコンテキストを使用し、インストール時に設定されたデフォルトのAstra Control kubeconfigは使用できません。また、単一のコンテキストを持つ限定されたロールは環境では機能しません。

作業を開始する前に

手順 の手順を実行する前に、管理するクラスタに次の情報があることを確認してください。

- A "[サポートされているバージョン](#)" のkubectlがインストールされています。
- Astra Control Serviceを使用して追加および管理するクラスタへのkubectlアクセス



この手順では、Astra Controlサービスを実行しているクラスタへのkubectlアクセスは必要ありません。

- アクティブなコンテキストのクラスタ管理者の権限で管理するクラスタのアクティブなkubeconfigです

手順

1. サービスアカウントを作成します。

- a. という名前のサービスアカウントファイルを作成します `astracontrol-service-account.yaml`。

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

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

- b. サービスアカウントを適用します。

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

2. 次のいずれかのクラスタロールを作成し、Astra Controlで管理するクラスタに必要な権限を割り当てます。

クラスターロールの制限

このロールには、Astra Controlでクラスタを管理するために必要な最小限の権限が含まれています。

- a. を作成します ClusterRole という名前のファイル。例：astra-admin-account.yaml。

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:

# Get, List, Create, and Update all resources
# Necessary to backup and restore all resources in an app
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - get
  - list
  - create
  - patch

# Delete Resources
# Necessary for in-place restore and AppMirror failover
- apiGroups:
  - ""
  - apps
  - autoscaling
  - batch
  - crd.projectcalico.org
  - extensions
  - networking.k8s.io
  - policy
  - rbac.authorization.k8s.io
  - snapshot.storage.k8s.io
  - trident.netapp.io
  resources:
  - configmaps
  - cronjobs
  - daemonsets
  - deployments
```

```

- horizontalpodautoscalers
- ingresses
- jobs
- namespaces
- networkpolicies
- persistentvolumeclaims
- poddisruptionbudgets
- pods
- podtemplates
- replicaset
- replicationcontrollers
- replicationcontrollers/scale
- rolebindings
- roles
- secrets
- serviceaccounts
- services
- statefulsets
- tridentmirrorrelationships
- tridentnapshotinfos
- volumesnapshots
- volumesnapshotcontents
verbs:
- delete

# Watch resources
# Necessary to monitor progress
- apiGroups:
  - ""
  resources:
  - pods
  - replicationcontrollers
  - replicationcontrollers/scale
  verbs:
  - watch

# Update resources
- apiGroups:
  - ""
  - build.openshift.io
  - image.openshift.io
  resources:
  - builds/details
  - replicationcontrollers
  - replicationcontrollers/scale
  - imagestreams/layers

```

```
- imagestreamtags
- imagetags
verbs:
- update
```

b. (OpenShiftクラスタの場合のみ) `astra-admin-account.yaml` ファイル：

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

c. クラスタロールを適用します。

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

クラスタロールの拡張

このロールには、Astra Controlで管理するクラスタに対する権限が拡張されています。このロールは、複数のコンテキストを使用し、インストール時に設定されたデフォルトのAstra Control kubeconfigを使用できない場合や、単一のコンテキストを持つ限定されたロールが環境で機能しない場合に使用できます。



次のようになります ClusterRole 手順はKubernetesの一般的な例です。ご使用の環境に固有の手順については、ご使用のKubernetesディストリビューションのドキュメントを参照してください。

a. を作成します ClusterRole という名前のファイル。例： `astra-admin-account.yaml`。

```
<strong>astra-admin-account.yaml</strong>
```



```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

b. クラスターロールを適用します。

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

3. サービスアカウントへのクラスターロールバインド用に、クラスターロールを作成します。

a. を作成します ClusterRoleBinding という名前のファイルです astracontrol-clusterrolebinding.yaml。

```
<strong>astracontrol-clusterrolebinding.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astracontrol-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: astra-admin-account
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

b. クラスターロールバインドを適用します。

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

4. トークンシークレットを作成して適用します。

- a. という名前のトークンシークレットファイルを作成します。 secret-astracontrol-service-account.yaml。

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

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-astracontrol-service-account
  namespace: default
  annotations:
    kubernetes.io/service-account.name: "astracontrol-service-
account"
type: kubernetes.io/service-account-token
```

- b. トークンシークレットを適用します。

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

5. トークンシークレットの名前を secrets Array（次の例の最後の行）：

```
kubectl edit sa astracontrol-service-account
```

```

apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-48xhx
kind: ServiceAccount
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},"name":"astracontrol-service-account","namespace":"default"},"creationTimestamp":"2023-06-14T15:25:45Z","name":"astracontrol-service-account","namespace":"default","resourceVersion":"2767069","uid":"2ce068c4-810e-4a96-ada3-49cbf9ec3f89"}
secrets:
- name: astracontrol-service-account-dockercfg-48xhx
<strong>- name: secret-astracontrol-service-account</strong>

```

6. サービスアカウントのシークレットを一覧表示します（置き換えます） <context> インストールに適したコンテキストを使用して、次の操作を行います。

```

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

```

出力の末尾は次のようになります。

```

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

```

内の各要素のインデックス secrets アレイは0から始まります。上記の例では、のインデックスです astracontrol-service-account-dockercfg-48xhx は0、のインデックスです secret-astracontrol-service-account は1です。出力で、サービスアカウントシークレットのインデックス番号をメモします。このインデックス番号は次の手順で必要になります。

7. 次のように kubeconfig を生成します。
 - a. を作成します create-kubeconfig.sh ファイル。
 - b. 交換してください TOKEN_INDEX 次のスクリプトの先頭に正しい値を入力します。

```

<strong>create-kubeconfig.sh</strong>

```

```

# Update these to match your environment.
# Replace TOKEN_INDEX with the correct value
# from the output in the previous step. If you
# didn't change anything else above, don't change
# anything else here.

SERVICE_ACCOUNT_NAME=astraccontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astraccontrol
KUBECONFIG_FILE='kubeconfig-sa'

CONTEXT=$(kubectl config current-context)

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  *-o jsonpath='{.secrets[TOKEN_INDEX].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.data.token}')

TOKEN=$(echo ${TOKEN_DATA} | base64 -d)

# Create dedicated kubeconfig
# Create a full copy
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp

# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp

# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user

```

```
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \  
    set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token-  
user  
  
# Set context to correct namespace  
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \  
    set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}  
  
# Flatten/minify kubeconfig  
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \  
    view --flatten --minify > ${KUBECONFIG_FILE}  
  
# Remove tmp  
rm ${KUBECONFIG_FILE}.full.tmp  
rm ${KUBECONFIG_FILE}.tmp
```

c. コマンドをソースにし、Kubernetes クラスタに適用します。

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

8. (オプション) クラスタにわかりやすい名前にコバーベキューの名前を変更します。

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

著作権に関する情報

Copyright © 2024 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。