# Astra Control Center をセットアップします

Astra Control Center

NetApp
November 21, 2023

# 目次

# Astra Control Center をセットアップします

Astra Control Center は、 ONTAP と Astra データストアをストレージバックエンドとしてサポートおよび監視します。Astra Control Center をインストールして UI にログインし、パスワードを変更したら、ライセンスの設定、クラスタの追加、ストレージの管理、バケットの追加を行います。

タスク

- Astra Control Center のライセンスを追加します
- [クラスタを追加]
- [ストレージバックエンドを追加します]
- [バケットを追加します]

## Astra Control Center のライセンスを追加します

UI またはを使用して新しいライセンスを追加できます "API" Astra Control Center の全機能を利用できます。ライセンスがないと、 Astra Control Center の使用は、ユーザの管理と新しいクラスタの追加に限定されます。

ライセンスの計算方法の詳細については、を参照してください "ライセンス"。

> 既存の評価版またはフルライセンスを更新するには、を参照してください "既存のライセンスを更新する"。

Astra Control Center ライセンスは、 Kubernetes CPU ユニットを使用して CPU リソースを測定します。ライセンスには、管理対象のすべての Kubernetes クラスタのワーカーノードに割り当てられた CPU リソースが含まれている必要があります。ライセンスを追加する前に、からライセンスファイル（ NLF ）を取得する必要があります "NetApp Support Site"。

また、 Astra Control Center に評価ライセンスをお試しいただくこともできます。このライセンスは、 Astra Control Center をダウンロードした日から 90 日間使用できます。登録すると、無償トライアルに登録できます "こちらをご覧ください"。

> インストールがライセンス数を超えると、 Astra Control Center は新しいアプリケーションを管理できなくなります。容量を超えるとアラートが表示されます。

必要なもの

から Astra Control Center をダウンロードした場合 "NetApp Support Site"には、 NetApp License File （ NLF ）もダウンロードします。このライセンスファイルにアクセスできることを確認してください。

手順

1. Astra Control Center UI にログインします。
2. 「 * アカウント * > * ライセンス * 」を選択します。
3. 「 * ライセンスの追加 * 」を選択します。
4. ダウンロードしたライセンスファイル（ NLF ）を参照します。
5. 「 * ライセンスの追加 * 」を選択します。

**Account**>*License* ページには、ライセンス情報、有効期限、ライセンスシリアル番号、アカウント ID 、および使用されている CPU ユニットが表示されます。

> 評価用ライセンスをお持ちの場合は、 Astra Control Center に障害が発生したときに ASUP を送信していないときにデータが失われないように、アカウント ID を必ず保存してください。

# クラスタを追加

アプリケーションの管理を開始するには、 Kubernetes クラスタを追加し、コンピューティングリソースとして管理します。Kubernetes アプリケーションを検出するには、 Astra Control Center のクラスタを追加する必要があります。Astra データストアの場合は、 Astra データストアによってプロビジョニングされたボリュームを使用するアプリケーションを含む Kubernetes アプリケーションクラスタを追加します。

> 他のクラスタを Astra Control Center に追加して管理する前に、 Astra Control Center が最初に導入したクラスタを管理することをお勧めします。指標およびトラブルシューティング用の Kubemetrics データとクラスタ関連データを送信するには、最初のクラスタを管理下に配置する必要があります。Add Cluster * 機能を使用して、 Astra Control Center でクラスタを管理できます。

> Astra Controlは、クラスタを管理する際に、クラスタのデフォルトストレージクラスを追跡します。を使用してストレージクラスを変更する場合は、を使用します `kubectl` コマンドを実行すると、Astra Controlによって変更が元に戻されます。Astra Controlで管理されるクラスタのデフォルトのストレージクラスを変更するには、次のいずれかの方法を使用します。
>
> - Astra Control API を使用 `PUT /managedClusters` エンドポイントを追加し、で別のデフォルトストレージクラスを割り当てます `DefaultStorageClass` パラメータ
> - Astra Control Web UIを使用して、別のデフォルトストレージクラスを割り当てます。を参照してください [デフォルトのストレージクラスを変更する]。

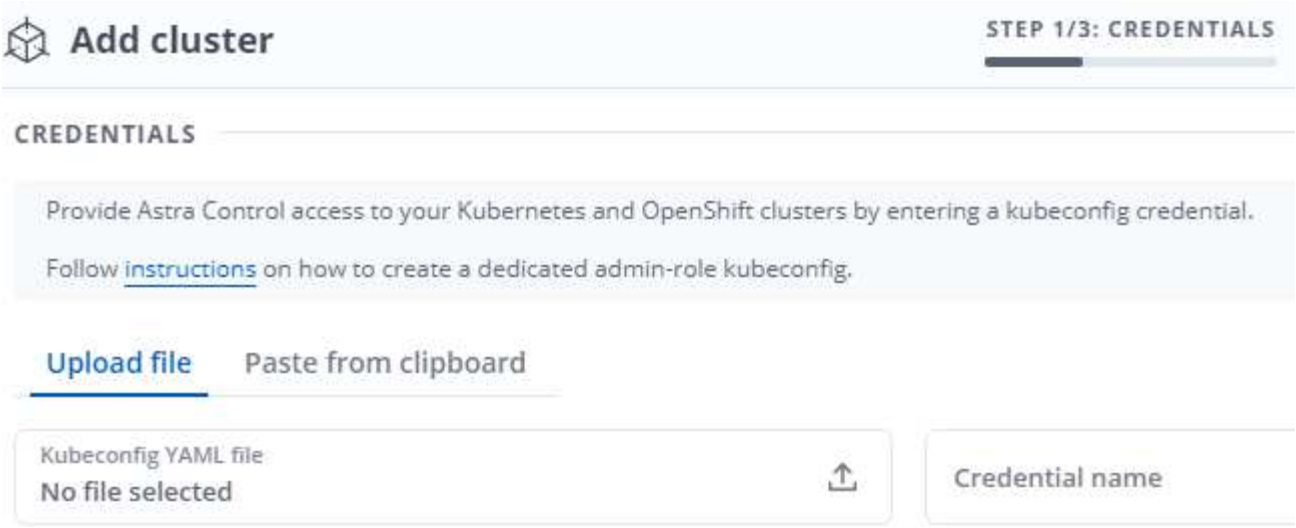
必要なもの

- クラスタを追加する前に、必要なを確認し、実行しておきます "前提条件となるタスク"。

手順

1. Astra Control Center UI の * ダッシュボード * から、クラスタセクションで * 追加 * を選択します。
2. 表示された*クラスタの追加*ウィンドウで、をアップロードします `kubeconfig.yaml` の内容をファイルまたは貼り付けます `kubeconfig.yaml` ファイル。

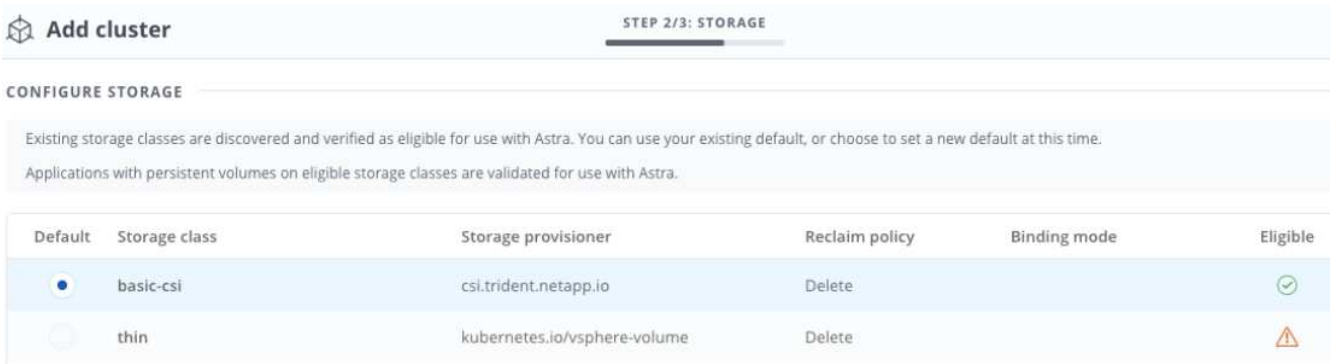
> 。 `kubeconfig.yaml` ファイルには、1つのクラスタのクラスタクレデンシャルのみを含める必要があります*。




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. 「ストレージの設定」を選択します。
5. この Kubernetes クラスタに使用するストレージクラスを選択し、 * Review * を選択します。

ONTAP ストレージまたは Astra データストアからバックアップされた Trident ストレージクラスを選択する必要があります。




6. 情報を確認し、問題がない場合は「 * クラスタの追加 * 」を選択します。

結果

クラスタが「 Discovering * 」ステータスになり、「 Running 」に変わります。Kubernetes クラスタが正常に追加され、 Astra Control Center で管理できるようになりました。

> Astra Control Center で管理するクラスタを追加したあと、監視オペレータの配置に数分かかる場合があります。それまでは、通知アイコンが赤に変わり、 * モニタリングエージェントステータスチェック失敗 * イベントが記録されます。この問題は無視してかまいません。問題は、Astra Control Center が正しいステータスを取得したときに解決します。数分経っても問題 が解決しない場合は、クラスタに移動してを実行します `oc get pods -n netapp-monitoring` を開始点として指定します。問題をデバッグするには、監視オペレータのログを調べる必要があります。

# ストレージバックエンドを追加します

ストレージバックエンドを追加して、 Astra Control がリソースを管理できるようにすることができます。管理対象クラスタにストレージバックエンドを導入するか、既存のストレージバックエンドを使用できます。

ストレージバックエンドとして Astra Control のストレージクラスタを管理することで、永続ボリューム（ PVS ）とストレージバックエンドの間のリンケージを取得できるだけでなく、追加のストレージ指標も取得できます。

既存の**Astra**データストアの導入に必要なもの

- Kubernetesアプリケーションクラスタと基盤となるコンピューティングクラスタを追加しておきます。

> Astraデータストア用のKubernetesアプリケーションクラスタを追加し、Astra Controlで管理したあと、クラスタはのように表示されます `unmanaged` 検出されたバックエンドのリスト。次に、 Astra データストアを含むコンピューティングクラスタを追加し、Kubernetes アプリケーションクラスタの基盤を構築する必要があります。これは、 UI の * Backends * から実行できます。クラスタのActions（操作）メニューを選択し、を選択します `Manage`および "クラスタを追加"。をクラスタの状態のあとに続けて追加します `unmanaged` Kubernetesクラスタの名前を変更した場合は、バックエンドの追加に進むことができます。

新しい**Astra**データストアの導入に必要なもの

- これで完了です "導入するインストールバンドルのバージョンをアップロードしました" Astra Controlからアクセス可能な場所への移動。
- 導入に使用するKubernetesクラスタを追加しておきます。
- をアップロードしました Astraデータストアライセンス Astra Controlからアクセス可能な場所への導入をサポートします。

オプション（ **Options** ）

- [ストレージリソースを導入]
- [既存のストレージバックエンドを使用する]

## ストレージリソースを導入

新しいAstraデータストアを導入して、関連するストレージバックエンドを管理できます。

手順

1. ダッシュボードまたはバックエンドメニューから移動します。
    - ダッシュボードから*：リソースサマリからストレージバックエンドペインからリンクを選択し、バッ

クエンドセクションから*追加*を選択します。

- バックエンドから * ：
  - i. 左側のナビゲーション領域で、 * Backends * を選択します。
  - ii. 「 * 追加」を選択します。

2. Deploy タブで Astra Data Store *導入オプションを選択します。
3. 導入するAstraデータストアパッケージを選択：
   a. Astraデータストアアプリケーションの名前を入力します。
   b. 導入するAstraデータストアのバージョンを選択します。

> 展開するバージョンをまだアップロードしていない場合は、*パッケージの追加*オプションを使用するか、ウィザードを終了してを使用できます "パッケージ管理" インストールバンドルをアップロードします。

4. 以前にアップロードしたAstraデータストアライセンスを選択するか、*ライセンスの追加*オプションを使用して、アプリケーションで使用するライセンスをアップロードします。

> 完全な権限を持つAstra Data StoreライセンスはKubernetesクラスタに関連付けられており、この関連クラスタは自動的に表示されるはずです。管理対象クラスタがない場合は、*クラスタの追加*オプションを選択してAstra Control管理に追加できます。Astra Data Storeライセンスの場合、ライセンスとクラスタの間に関連付けが行われていない場合は、ウィザードの次のページでこの関連付けを定義できます。

5. KubernetesクラスタをAstra Control管理に追加していない場合は、* Kubernetes cluster *ページから追加する必要があります。リストから既存のクラスタを選択するか、「*基盤となるクラスタを追加」を選択してAstra Control管理用にクラスタを追加します。
6. Astraデータストアにリソースを提供するKubernetesクラスタのテンプレートサイズを選択します。次のいずれかを選択できます。
   - をクリックします `Recommended Kubernetes worker node requirements`をクリックし、ライセンスで許可されている内容に基づいて、テンプレートを大規模から小に選択します。
   - をクリックします `Custom Kubernetes worker node requirements`をクリックし、各クラスタノードに必要なコア数と総メモリを選択します。また、コアとメモリの選択基準を満たす、クラスタ内の対応するノード数も表示できます。

> テンプレートを選択する際は、大規模なワークロードにはメモリとコアが多く、小規模なワークロードにはノード数が多い大規模なノードを選択します。ライセンスで許可されている内容に基づいてテンプレートを選択する必要があります。推奨されるテンプレートオプションごとに、各ノードのメモリとコアおよび容量のテンプレートパターンを満たす、適格なノードの数が提示されます。

7. ノードを設定します。
   a. ノードラベルを追加して、このAstraデータストアクラスタをサポートするワーカーノードのプールを特定します。

> このラベルは、Astraデータストアの導入に使用するクラスタ内の各ノードに追加してからでないと、導入や導入が失敗します。

b. ノードあたりの容量（GiB）を手動で設定するか、許容される最大ノード容量を選択します。

c. クラスタで許可される最大ノード数を設定するか、クラスタで許容される最大ノード数を設定します。

8. （Astraデータストアフルライセンスのみ）保護ドメインに使用するラベルのキーを入力します。

> 各ノードのキーに対して、少なくとも3つの一意のラベルを作成します。たとえば、キーがの場合などです `astra.datastore.protection.domain`次のラベルを作成できます。`astra.datastore.protection.domain=domain1,astra.datastore.protection.domain=domain2`および `astra.datastore.protection.domain=domain3`。

9. 管理ネットワークを設定します。

   a. Astraデータストアの内部管理用の管理IPアドレスを入力します。このIPアドレスは、ワーカーノードのIPアドレスと同じサブネットにあります。

   b. 管理ネットワークとデータネットワークで同じNICを使用するか、または個別に設定します。

   c. データネットワークのIPアドレスプール、サブネットマスク、ストレージアクセス用のゲートウェイを入力してください。

10. 設定を確認し、「* Deploy *」を選択してインストールを開始します。

結果

インストールが正常に完了すると、バックエンドがに表示されます `available` バックエンドにアクティブなパフォーマンス情報とともに表示

> バックエンドが表示されるようにページを更新する必要がある場合があります。

## 既存のストレージバックエンドを使用する

検出されたONTAP またはAstraデータストアのストレージバックエンドをAstra Control Center管理に組み込むことができます。

手順

1. ダッシュボードまたはバックエンドメニューから移動します。
   - ダッシュボードから*：リソースサマリからストレージバックエンドペインからリンクを選択し、バックエンドセクションから*追加*を選択します。
   - バックエンドから * ：
     i. 左側のナビゲーション領域で、 * Backends * を選択します。
     ii. 管理対象クラスタから検出されたバックエンドで* Manage を選択するか、 Add *を選択して追加の既存バックエンドを管理します。
2. [ 既存の使用（ Use Existing ） ] * タブを選択します。
3. バックエンドの種類に応じて、次のいずれかの操作を行います。
   - * Astra データストア * ：
     i. 「* Astra Data Store *」を選択します。
     ii. 管理対象のコンピューティングクラスタを選択し、 * Next * を選択します。

iii. バックエンドの詳細を確認し、「Add storage backend *」を選択します。

- * ONTAP * ：
  i. 「* ONTAP 」を選択し、「 Next *」を選択します。
  ii. ONTAP クラスタ管理IPアドレスと管理者クレデンシャルを入力します。

> ここで入力するクレデンシャルのユーザは、を持っている必要があります `ontapi` ONTAP クラスタのONTAP System Managerで有効になっているユーザログインアクセス方法。SnapMirrorレプリケーションを使用する場合は、アクセス方法を有効にします `ontapi` および `http` 両方のONTAP クラスタ上のユーザに対して設定します。を参照してください "ユーザアカウントを管理する" を参照してください。

  iii. [* Review （レビュー） ] を選択します
  iv. バックエンドの詳細を確認し、「Add storage backend *」を選択します。

結果

バックエンドがに表示されます `available` リストに概要情報を表示します。

> バックエンドが表示されるようにページを更新する必要がある場合があります。

# バケットを追加します

アプリケーションと永続的ストレージをバックアップする場合や、クラスタ間でアプリケーションのクローニングを行う場合は、オブジェクトストアバケットプロバイダの追加が不可欠です。Astra Control は、これらのバックアップまたはクローンを、定義したオブジェクトストアバケットに格納します。

バケットを追加すると、 Astra Control によって、 1 つのバケットがデフォルトのバケットインジケータとしてマークされます。最初に作成したバケットがデフォルトバケットになります。

アプリケーション構成と永続的ストレージを同じクラスタにクローニングする場合、バケットは必要ありません。

次のいずれかのバケットタイプを使用します。

- NetApp ONTAP S3
- NetApp StorageGRID S3 の略
- 汎用 S3

> Amazon Web Services（AWS）とGoogle Cloud Platform（GCP）では、汎用のS3バケットタイプを使用します。

- Microsoft Azure

> Astra Control Center は Amazon S3 を汎用 S3 バケットプロバイダとしてサポートしていますが、 Astra Control Center は Amazon の S3 サポートを要求するすべてのオブジェクトストアベンダーをサポートしているわけではありません。

- Microsoft Azure

Astra Control API を使用してバケットを追加する手順については、を参照してください "Astra の自動化と API に関する情報"。

手順

1. 左側のナビゲーション領域で、 * バケット * を選択します。
   a. 「 * 追加」を選択します。
   b. バケットタイプを選択します。

      バケットを追加するときは、正しいバケットプロバイダを選択し、そのプロバイダに適したクレデンシャルを指定します。たとえば、タイプとして NetApp ONTAP S3 が許可され、 StorageGRID クレデンシャルが受け入れられますが、このバケットを使用して原因の以降のアプリケーションのバックアップとリストアはすべて失敗します。

   c. 新しいバケット名を作成するか、既存のバケット名とオプションの概要を入力します。

      バケット名と概要は、バックアップを作成するときに後で選択できるバックアップの場所として表示されます。この名前は、保護ポリシーの設定時にも表示されます。

   d. S3 エンドポイントの名前または IP アドレスを入力します。
   e. このバケットをすべてのバックアップのデフォルトバケットにする場合は、を確認します `Make this bucket the default bucket for this private cloud` オプション

      このオプションは、最初に作成したバケットに対しては表示されません。

   f. 追加して続行します クレデンシャル情報。

### S3 アクセスクレデンシャルを追加します

S3 アクセスクレデンシャルはいつでも追加できます。

手順

1. バケット（ Buckets ）ダイアログで、 * 追加（ Add ） * または * 既存の * を使用（ Use Existing * ）タブのいずれかを選択します。
   a. Astra Control の他のクレデンシャルと区別するクレデンシャルの名前を入力します。
   b. クリップボードからコンテンツを貼り付けて、アクセス ID とシークレットキーを入力します。

## 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。

手順

1. Astra Control Center Web 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。

# 次の手順

Astra Control Center にログインしてクラスタを追加したので、 Astra Control Center のアプリケーションデータ管理機能を使い始めることができます。

- "ユーザを管理します"
- "アプリの管理を開始します"
- "アプリを保護します"
- "アプリケーションをクローニング"
- "通知を管理します"
- "Cloud Insights に接続します"
- "カスタム TLS 証明書を追加します"

# 詳細については、こちらをご覧ください

- "Astra Control API を使用"
- "既知の問題"

# クラスタを追加するための前提条件

クラスタを追加する前に、前提条件が満たされていることを確認する必要があります。また、資格チェックを実行して、アストラコントロールセンターにクラスタを追加する準備ができていることを確認する必要があります。

## クラスタを追加する前に必要な作業

クラスタがに記載された要件を満たしていることを確認します "アプリケーションクラスタの要件"。

> 管理対象のコンピューティングリソースとして 2 つ目の OpenShift 4.6 、 4.7 、または 4.8 クラスタを追加する場合は、 Astra Trident ボリュームスナップショット機能が有効になっていることを確認する必要があります。ネットアップの公式 Astra Trident をご覧ください "手順 " Trident を使用して、ボリューム Snapshot を有効にしてテストしてください。

- で構成されたAstra Trident StorageClasses "サポートされるストレージバックエンド" （すべてのタイプのクラスタに必要）
- Astra Control Center を使用してアプリケーションをバックアップおよび復元するためにバックアップ ONTAP システムで設定されたスーパーユーザーおよびユーザー ID 。ONTAP コマンドラインで次のコマンドを実行します。

```
export-policy rule modify -vserver <storage virtual machine name> -policyname
<policy name> -ruleindex 1 -superuser sysm --anon 65534
```

- Astra Trident `volumesnapshotclass` 管理者によって定義されたオブジェクト。Astra Trident の詳細をご確認ください "手順" Trident を使用して、ボリューム Snapshot を有効にしてテストしてください。
- Kubernetes クラスタにデフォルトのストレージクラスが 1 つだけ定義されていることを確認します。

## 資格チェックを実行します

次の資格チェックを実行して、 Astra Control Center にクラスタを追加する準備ができていることを確認します。

手順

1. Trident のバージョンを確認

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

Trident が存在する場合は、次のような出力が表示されます。

```
NAME      VERSION
trident   21.04.0
```

Trident が存在しない場合は、次のような出力が表示されます。

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

> Trident がインストールされていない場合や、インストールされているバージョンが最新でない場合は、次に進む前に最新バージョンの Trident 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 "Trident のドキュメント" 手順については、を参照し

2. サポートされている Trident ドライバをストレージクラスが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
NAME                    PROVISIONER                    RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)    csi.trident.netapp.io          Delete
Immediate           true                   5d23h
thin                    kubernetes.io/vsphere-volume   Delete
Immediate           false                  6d
```

## admin ロールの kubeconfig を作成します

手順を実行する前に、マシンに次のものがあることを確認してください。

- `kubectl` V1.19以降がインストールされている
- アクティブなコンテキストのクラスタ管理者権限があるアクティブな kubeconfig です

手順

1. 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`。

      名前と名前空間を必要に応じて調整します。ここで変更を行った場合は、以降の手順でも同じ変更を適用する必要があります。

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

+

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

   a. サービスアカウントを適用します。

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

2. （オプション）権限付きポッドの作成を許可しない制限付きポッドセキュリティポリシーをクラスタで使用している場合、またはポッドコンテナ内のプロセスをルートユーザとして実行できるようにしていない場合は、 Astra Control でポッドを作成および管理できるように、クラスタ用のカスタムポッドセキュリティポリシーを作成します。手順については、を参照してください "カスタムのポッドセキュリティポリシーを作成します"。

3. 次のようにクラスタ管理者権限を付与します。

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

      必要に応じて、サービスアカウントの作成時に変更した名前と名前空間を調整します。

```
<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: cluster-admin
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

a. クラスタロールバインドを適用します。

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

4. 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-vhz87"},
{ "name": "astracontrol-service-account-token-r59kr"}
]
```

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-vhz87` は0、のインデックスです `astracontrol-service-account-token-r59kr` は1です。出力で、 "token" という単語が含まれるサービスアカウント名のインデックスをメモしてください。

5. 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
```

6. （* オプション *）クラスタにわかりやすい名前にコベ econfig の名前を変更します。クラスタのクレデンシャルを保護します。

```
chmod 700 create-kubeconfig.sh
mv kubeconfig-sa.txt YOUR_CLUSTER_NAME_kubeconfig
```

### 次の手順

前提条件が満たされていることを確認したら、次は準備ができています "クラスタを追加"。

### 詳細については、こちらをご覧ください

- "Trident のドキュメント"
- "Astra Control API を使用"

## カスタム TLS 証明書を追加します

既存の自己署名 TLS 証明書を削除して、認証局（ CA ）が署名した TLS 証明書に置き換えることができます。

必要なもの

- 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>
```

## 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>
....

#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>
....

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。

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

5. 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>
....

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.containo.us -n <ACC-deployment-
namespace>
....

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

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

8. 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。
9. 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。
10. 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。

# カスタムのポッドセキュリティポリシーを作成します

Astra Control では、管理対象のクラスタに Kubernetes ポッドを作成して管理する必要があります。クラスタで、特権ポッドの作成を許可しない制限付きポッドセキュリティポリシーを使用している場合、またはポッドコンテナ内のプロセスをルートユーザとして実行できるように許可していない場合は、制限の少ないポッドセキュリティポリシーを作成して、 Astra Control がこれらのポッドを作成および管理できるようにする必要があります。

手順

1. デフォルトよりも制限の緩いクラスタの PoD セキュリティポリシーを作成してファイルに保存します。
   例：

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: astracontrol
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  volumes:
  - '*'
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
```

2. ポッドセキュリティポリシーの新しいロールを作成します。

```
kubectl-admin create role psp:astracontrol \
    --verb=use \
    --resource=podsecuritypolicy \
    --resource-name=astracontrol
```

3. 新しいロールをサービスアカウントにバインドします。

```
kubectl-admin create rolebinding default:psp:astracontrol \
    --role=psp:astracontrol \
    --serviceaccount=astracontrol-service-account:default
```