インストールの概要

Astra Data Store

NetApp
January 12, 2022

目次

Astra データストアのインストールの概要

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

- "標準のプロセスを使用して Astra データストアをインストールします"。
- "Red Hat OpenShift を使用する場合は、 OpenShift を使用して Astra データストアをインストールします"。

Astra データストアプレビューをインストールします

Astra データストアプレビューをインストールするには、からインストールバンドルをダウンロードします "ネットアップサポートサイト" およびこの手順に記載されているインストール手順を実行します。

または、を使用することもできます "Red Hat OpenShift Container Platform （ OCP ）への Astra データストアプレビューのインストール"。

必要なもの

- "インストールを開始する前に、 Astra データストアプレビュー環境を準備します"。
- にアクセスします "ネットアップサポートサイト"。 フルアクセスのネットアップサポートサイトアカウントをまだお持ちでない場合は、プレビュー版をお持ちください。
- A "ネットアップライセンスファイル（ NLF ）" Astra データストアプレビュー版ライセンスのダウンロード手順がお客様に送信されます "サインアップ"。
- 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。
- の理解 "ロールと権限" Astra データストアプレビュー版で使用
- インターネット接続：Astra データストアプレビューは、エアギャップのある環境には対応していません。直接またはプロキシ経由で support.netapp.com にアクセスするには、インターネット接続が必要です。

Astra Data Store プレビューインストールプロセスでは、次の手順を実行できます。

- [Download the Astra Data Store preview bundle and extract the images]
- [Copy the binary and push images to your local registry]
- [Install the Astra Data Store preview operator]
- [Deploy the Astra Data Store preview version YAML]
- [Apply the Astra Data Store preview license]
- [Install the Astra Data Store preview cluster]
- [Understand deployment-related events]
- [Configure Astra Data Store preview monitoring]

Astra Data Store プレビューバンドルをダウンロードして、イメージを展開

1. にログインします "ネットアップサポートサイト" Astra データストアプレビューバンドル (`2021.12_Astrundsstore.tar') をダウンロードします
2. （任意）次のコマンドを使用して、バンドルのシグニチャを確認します。

```
openssl dgst -sha256 -verify 2021.12_astradatastore.pub -signature
2021.12_astradatastore.sig 2021.12_astradatastore.tar
```

3. 画像を抽出します。

```
tar -xvf 2021.12_astradatastore.tar
```

バイナリをコピーし、ローカルレジストリにイメージをプッシュします

1. kubectl-astras バイナリを ' イメージを抽出するために使用したディレクトリから 'k8s kubectl バイナリがインストールされている標準パス (/usr/bin/ など) にコピーしますkubectl-astras は、 Astra データストアプレビュークラスタをインストールおよび管理するカスタムの kubectl 拡張機能です。

```
cp -p ./bin/kubectl-astrads /usr/bin/.
```

2. Astra Data Store プレビューイメージディレクトリ内のファイルをローカルレジストリに追加します。

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

   a. レジストリにログインします。

   ```
   docker login [your_registry_path]
   ```

   b. 環境変数を 'Astra Data Store プレビューイメージをプッシュするレジストリパスに設定しますたとえば 'repo.company.com` です

   ```
   export REGISTRY=repo.company.com/astrads
   ```

   c. スクリプトを実行して Docker にイメージをロードし、イメージにタグを付けます。[[[</Z1>[</Z1>[</Z1>_image_local_registry_push]] ローカルレジストリにイメージをプッシュします。</Z2>

```
for astraImageFile in $(ls images/*.tar) ; do
   astraImage=$(docker load --input ${astraImageFile} | sed 's~Loaded
image: ~~')
   astraImageShort=`echo $astraImage | sed 's~.*/~~'`
   docker tag ${astraImage} ${REGISTRY}/${astraImageShort}
   docker push ${REGISTRY}/${astraImageShort}
done
sed -i 's~\[YOUR REGISTRY\]~'${REGISTRY}'~' ./manifests/*.yaml
```

Astra Data Store プレビューオペレータをインストール

1. Astra データストアのプレビューマニフェストを表示する：

```
ls manifests/*yaml
```

対応：

```
manifests/astradscluster.yaml
manifests/astradsoperator.yaml
manifests/astradsversion.yaml
manifests/monitoring_operator.yaml
```

2. kubectl apply を使用してオペレータを配備します。

```
kubectl apply -f ./manifests/astradsoperator.yaml
```

対応：

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsautosupports.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslicenses.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnfsoptions.astrads.
netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnodeinfoes.astrads.
netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsqospolicies.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumefiles.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumes.astrads.net
```

```
app.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.ast
rads.netapp.io created
role.rbac.authorization.k8s.io/astrads-leader-election-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-metrics-reader created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-proxy-role created
rolebinding.rbac.authorization.k8s.io/astrads-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
```

```
created
clusterrolebinding.rbac.authorization.k8s.io/astrads-proxy-rolebinding
created
configmap/astrads-autosupport-cm created
configmap/astrads-firetap-cm created
configmap/astrads-fluent-bit-cm created
configmap/astrads-kevents-asup created
configmap/astrads-metrics-cm created
service/astrads-operator-metrics-service created
deployment.apps/astrads-operator created
```

3. Astra データストアオペレータポッドが起動し、実行中であることを確認します。

```
kubectl get pods -n astrads-system
```

対応：

```
NAME                                  READY   STATUS    RESTARTS   AGE
astrads-operator-5ffb94fbf-7ln4h      1/1     Running   0          17m
```

Astra Data Store プレビュー版 YAML を導入します

1. kubectl apply を使用した導入：

```
kubectl apply -f ./manifests/astradsversion.yaml
```

2. ポッドが実行されていることを確認します。

```
kubectl get pods -n astrads-system
```

対応：

```
NAME                                              READY   STATUS    RESTARTS
AGE
astrads-cluster-controller-7f6f884645-xxf2n   1/1     Running   0
117s
astrads-ds-nodeinfo-astradsversion-2jqnk      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-dbk7v      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-rn9tt      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-vsmhv      1/1     Running   0
2m7s
astrads-license-controller-fb8fd56bc-bxq7j    1/1     Running   0
2m2s
astrads-operator-5ffb94fbf-7ln4h              1/1     Running   0
2m10s
```

Astra データストアプレビューライセンスを適用

1. プレビュー版への登録時に入手したネットアップライセンスファイル（ NLF ）を適用します。コマンドを実行する前に、使用しているクラスタの名前（「 <AstrA-Data-Store-cluster-name>` ）を入力します 導入に進みます または ' すでに配備されているか ' ライセンス・ファイルへのパス (<file_path/file.txt>) があります

```
kubectl astrads license add --license-file-path <file_path/file.txt>
--ads-cluster-name <Astra-Data-Store-cluster-name> -n astrads-system
```

2. ライセンスが追加されたことを確認します。

```
kubectl astrads license list
```

対応：

```
NAME        ADSCLUSTER                  VALID   PRODUCT
EVALUATION  ENDDATE     VALIDATED
p100000006  astrads-example-cluster    true    Astra Data Store Preview
true        2022-01-23  2021-11-04T14:38:54Z
```

Astra データストアプレビュークラスタをインストール

1. YAML ファイルを開きます。

```
vim ./manifests/astradscluster.yaml
```

2. YAML ファイルで次の値を編集します。

   YAML ファイルの簡単な例は、次の手順を実行します。

   a. （必須） * Metadata* ：「 metadata 」で、「 name 」の文字列をクラスタの名前に変更します。このクラスタ名は、ときと同じである必要があります ライセンスを適用します。

   b. (必須)**Spec**:'spec' の次の必須値を変更します

      - 「 mvip 」文字列を、クラスタ内の任意のワーカーノードからルーティング可能なフローティング管理 IP の IP アドレスに変更します。
      - 「 adsDataNetworks 」に、 NetApp ボリュームをマウントするホストからルーティング可能なフローティング IP アドレス（「アドレス」）をカンマで区切って追加します。ノードごとに 1 つのフローティング IP アドレスを使用します。データネットワーク IP アドレスは、 Astra Data Store のプレビューノードと同じ数以上必要です。Astra データストアプレビューの場合、少なくとも 4 つのアドレスを意味します。あとで 5 つのノードにクラスタを拡張する予定の場合は、 5 つのアドレスを意味します。
      - 「 adsDataNetworks 」で、データネットワークが使用するネットマスクを指定します。
      - 「 adsNetworkInterfaces 」で、「 mgmt 」と「 data 」の値を、管理、クラスタ、およびストレージに使用するネットワークインターフェイス名に置き換えます。名前を指定しない場合、ノードのプライマリインターフェイスが管理、クラスタ、ストレージのネットワークに使用されます。

        クラスタとストレージのネットワークのインターフェイスが同じである必要があります。Astra Data Store プレビュー管理インターフェイスは、 Kubernetes ノードの管理インターフェイスと同じである必要があります。

   c. （任意） * monitoringConfig* ：を設定する場合 監視オペレータ （監視に Astra Control Center を使用していない場合はオプション）、セクションからコメントを削除し、エージェント CR （監視用オペレータリソース）が適用されるネームスペース（デフォルトは「 NetApp-monitoring 」）を追加し、前の手順で使用したレジストリ（「 Your_registry_path 」）のリポジトリパスを追加します。

   d. （任意） * autoSupportConfig * ：を保持します "AutoSupport" プロキシを設定する必要がない場合のデフォルト値は次のとおりです。

      - 「 ProxyURL 」の場合は、 AutoSupport バンドルの転送に使用するポートにプロキシの URL を設定します。

        ほとんどのコメントは YAML サンプルから削除されています。

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
```

```
    cpu: 9
    memory: 34
  adsNodeCount: 4
  mvip: ""
  adsDataNetworks:
    - addresses: ""
      netmask:
  # Specify the network interface names to use for management, cluster
and storage networks.
  # If none are specified, the node's primary interface will be used for
management, cluster and storage networking.
  # To move the cluster and storage networks a different interface than
management, specify all three interfaces to use here.
  # The Astra Data Store management interface should be same as the
Kubernetes node’s management interface.
  # NOTE: The cluster and storage networks need to be on the same
interface.
  adsNetworkInterfaces:
    managementInterface: "mgmt"
    clusterInterface: "data"
    storageInterface: "data"
  # [Optional] Provide a monitoring config to be used to setup/configure
a monitoring agent.
 # monitoringConfig:
   # namespace: "netapp-monitoring"
   # repo: "[YOUR REGISTRY]"
  autoSupportConfig:
    autoUpload: true
    enabled: true
    coredumpUpload: false
    historyRetentionCount: 25
    destinationURL: "https://support.netapp.com/put/AsupPut"
    # ProxyURL defines the URL of the proxy with port to be used for
AutoSupport bundle transfer
    # proxyURL:
    periodic:
      - schedule: "0 0 * * *"
        periodicconfig:
        - component:
            name: storage
            event: dailyMonitoring
          userMessage: Daily Monitoring Storage AutoSupport bundle
          nodes: all
        - component:
            name: controlplane
            event: daily
```

```
          userMessage: Daily Control Plane AutoSupport bundle
```

3. kubectl apply を使用してクラスタを導入します

```
kubectl apply -f ./manifests/astradscluster.yaml
```

4. クラスタ作成処理が完了するまで数分待ってから、ポッドが実行されていることを確認します。

```
kubectl get pods -n astrads-system
```

回答例：

```
NAME READY STATUS RESTARTS AGE
astrads-cluster-controller-7c67cc7f7b-2jww2 1/1 Running 0 7h31m
astrads-deployment-support-788b859c65-2qjkn 3/3 Running 19 12d
astrads-ds-astrads-cluster-1ab0dbc-j9jzc 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-1ab0dbc-k9wp8 1/1 Running 0 5d1h
astrads-ds-astrads-cluster-1ab0dbc-pwk42 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-1ab0dbc-qhvc6 1/1 Running 0 8h
astrads-ds-nodeinfo-astradsversion-gcmj8 1/1 Running 1 12d
astrads-ds-nodeinfo-astradsversion-j826x 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-vdthh 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-xwgsf 1/1 Running 0 12d
astrads-ds-support-828vw 2/2 Running 2 5d2h
astrads-ds-support-cfzts 2/2 Running 0 8h
astrads-ds-support-nzkkr 2/2 Running 15 7h49m
astrads-ds-support-xxbnp 2/2 Running 1 5d2h
astrads-license-controller-86c69f76bb-s6fb7 1/1 Running 0 8h
astrads-operator-79ff8fbb6d-vpz9m 1/1 Running 0 8h
```

5. クラスタの導入の進捗を確認します。

```
kubectl get astradscluster -n astrads-system
```

回答例：

```
NAME                        STATUS    VERSION    SERIAL NUMBER    MVIP
AGE

astrads-example-cluster   created   2021.10.0   p100000006
10.x.x.x   10m
```

導入に関連するイベントを把握

クラスタの導入中に ' オペレーション・ステータスは ' ブランクから ' 進行中 ' から作成済みに変更する必要がありますクラスタの導入には約 8~10 分かかります。導入中にクラスタイベントを監視するには、次のいずれかのコマンドを実行します。

```
kubectl get events --field-selector involvedObject.kind=AstraDSCluster -n
astrads-system
```

```
kubectl describe astradscluster <cluster name> -n astrads-system
```

導入時の主なイベントを次に示します。

イベントメッセージ	意味
ADS クラスタに参加するコントロールプレーンノードを 4 つ選択しました	Astra Data Store プレビューオペレータは、 Astra データストアプレビュークラスタを構築するために、 CPU 、メモリ、ストレージ、ネットワークを備えた十分なノードを特定しました。
ADS クラスタが作成中です	Astra データストアプレビュークラスタコントローラが、クラスタ作成処理を開始しました。
ADS クラスタが作成されました	クラスタが作成されました。

クラスタのステータスが「 in progress 」に変わらない場合は、オペレータログでノード選択の詳細を確認します。

```
kubectl logs -n astrads-system <astrads operator pod name>
```

クラスタのステータスが「処理中」のままである場合は、クラスタコントローラのログを確認します。

```
kubectl logs -n astrads-system <astrads cluster controller pod name>
```

Astra データストアのプレビュー監視を設定

Astra データストアプレビューは、 Astra Control Center の監視用、または別のテレメトリサービスによる監視用に設定できます。

Astra Control Center プレビューの監視を設定します

次の手順は、 Astra データストアのプレビューが Astra Control Center のバックエンドとして管理された後にのみ実行します。

1. Astra Control Center によるモニタリングのための Astra データストアプレビューの構成：

```
kubectl astrads monitoring -m netapp-monitoring -r [YOUR REGISTRY] setup
```

監視オペレータをインストールします

（オプション） Astra データストアのプレビューが Astra Control Center にインポートされない場合にのみ、監視オペレータをお勧めします。モニタリングオペレータは、アストラデータストアプレビューインスタンスがスタンドアロン環境である場合、 Cloud Insights を使用してテレメトリを監視する場合、または Elastic などのサードパーティのエンドポイントにログをストリーミングする場合にインストールできます。

1. 次のインストールコマンドを実行します。

```
kubectl apply -f ./manifests/monitoring_operator.yaml
```

2. Astra データストアプレビューで監視を設定：

```
kubectl astrads monitoring -m netapp-monitoring -r [YOUR REGISTRY] setup
```

次の手順

を実行して導入を完了します "セットアップのタスク"。

Red Hat OpenShift Container Platform に Astra データストアプレビューをインストールします

Red Hat OpenShift Container Platform （ OCP ）に Astra データストアプレビューをインストールするには、からインストールバンドルをダウンロードします "ネットアップサポートサイト" およびこの手順に記載されているインストール手順を実行します。

必要なもの

- インストールを開始する前に、 "Astra データストア環境の準備"。
- にアクセスします "ネットアップサポートサイト"。 フルアクセスのネットアップサポートサイトアカウントをまだお持ちでない場合は、プレビュー版をお持ちください。
- A "ネットアップライセンスファイル" （ NLF ） for Astra Data Store のプレビュー。ライセンスのダウンロード手順は、サインアップ後に送信されます。
- 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。
- の理解 "ロールと権限" Astra データストアプレビュー版で使用
- インターネット接続：Astra データストアプレビューは、エアギャップ環境には対応していません。直接またはプロキシ経由で support.netapp.com にアクセスするには、インターネット接続が必要です。

このタスクについて

Astra Data Store プレビューインストールプロセスでは、次の手順を実行できます。

- [Download the Astra Data Store preview bundle and extract the images]
- [Copy the binary and push images to your local registry]
- [Create a namespace to deploy Astra Data Store preview]
- [Create a custom SCC]
- [Create the roles and role bindings]
- [Install the Astra Data Store preview operator]
- [Deploy the Astra Data Store preview version YAML]
- [Apply the Astra Data Store preview license]
- [Install the Astra Data Store preview cluster]
- [Understand deployment-related events]
- [Configure Astra Data Store preview monitoring]
- [Install the monitoring operator]
- [Set up iptables]

Astra Data Store プレビューバンドルをダウンロードして、イメージを展開

1. にログインします "ネットアップサポートサイト" Astra データストアプレビューバンドル (`2021.12_Astrundsstore.tar') をダウンロードします
2. （任意）バンドルのシグニチャを確認します。

```
openssl dgst -sha256 -verify 2021.12_astradatastore.pub -signature
2021.12_astradatastore.sig 2021.12_astradatastore.tar
```

3. 画像を抽出します。

```
tar -xvf 2021.12_astradatastore.tar
```

バイナリをコピーし、ローカルレジストリにイメージをプッシュします

1. kubectl-astras バイナリを ' イメージを抽出するために使用したディレクトリから 'k8s kubectl バイナリがインストールされている標準パス (/usr/bin/ など) にコピーしますkubectl-astras は、 Astra データストアプレビュークラスタをインストールおよび管理するカスタムの kubectl 拡張機能です。

```
cp -p ./bin/kubectl-astrads /usr/bin/.
```

2. Astra Data Store プレビューイメージディレクトリ内のファイルをローカルレジストリに追加します。

以下の画像の自動ロードについては、サンプルスクリプトを参照してください。

a. レジストリにログインします。

```
docker login [your_registry_path]
```

b. 環境変数を 'Astra Data Store プレビューイメージをプッシュするレジストリパスに設定しますたとえば 'repo.company.com` です

```
export REGISTRY=repo.company.com/astrads
```

c. スクリプトを実行して Docker にイメージをロードし、イメージにタグを付けます。[[[</Z1>[</Z1>[</Z1>_image_local_registry_push]] ローカルレジストリにイメージをプッシュします。</Z2>

```
for astraImageFile in $(ls images/*.tar) ; do
   astraImage=$(docker load --input ${astraImageFile} | sed 's~Loaded
image: ~~')
   astraImageShort=`echo $astraImage | sed 's~.*/~~'`
   docker tag ${astraImage} ${REGISTRY}/${astraImageShort}
   docker push ${REGISTRY}/${astraImageShort}
done
sed -i 's~\[YOUR REGISTRY\]~'${REGISTRY}'~' ./manifests/*.yaml
```

Astra データストアプレビューを導入するためのネームスペースを作成します

すべての Astra Data Store プレビューコンポーネントをインストールする名前空間「 astras -system 」を作成します。

1. ネームスペースを作成します。

```
kubectl create -f ads_namespace.yaml
```

例： ads_namespac.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: operator
  name: astrads-system
```

カスタム **SCC** を作成します

OpenShift では、セキュリティコンテキスト制約（ SCC ）を使用して、ポッドで実行できるアクションを制御します。デフォルトでは、任意のコンテナの実行には制限付き SCC が付与され、その SCC で定義された機能のみが付与されます。

制限付き SCC では、 Astra Data Store プレビュークラスタポッドで必要な権限が提供されません。この手順を使用して、 Astra データストアのプレビュー版に必要な権限（サンプルに記載）を付与します。

カスタム SCC を Astra Data Store Preview ネームスペースのデフォルトのサービスアカウントに割り当てます。

手順

1. カスタム SCC を作成します。

```
kubectl create -f ads_privileged_scc.yaml
```

サンプル： ads_privileged_ssc.yaml

```
allowHostDirVolumePlugin: true
allowHostIPC: true
allowHostNetwork: true
allowHostPID: true
allowHostPorts: true
allowPrivilegeEscalation: true
allowPrivilegedContainer: true
allowedCapabilities:
- '*'
allowedUnsafeSysctls:
- '*'
apiVersion: security.openshift.io/v1
defaultAddCapabilities: null
fsGroup:
  type: RunAsAny
groups: []
kind: SecurityContextConstraints
metadata:
  annotations:
    kubernetes.io/description: 'ADS privileged. Grant with caution.'
    release.openshift.io/create-only: "true"
  name: ads-privileged
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities: null
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: RunAsAny
seccompProfiles:
- '*'
supplementalGroups:
  type: RunAsAny
users:
- system:serviceaccount:astrads-system:default
volumes:
- '*'
```

2. 「 OC get SCC 」コマンドを使用して、新たに追加された SCC を表示します。

```
# oc get scc/ads-privileged
NAME              PRIV   CAPS    SELINUX    RUNASUSER   FSGROUP
SUPGROUP   PRIORITY     READONLYROOTFS   VOLUMES
ads-privileged   true   ["*"]   RunAsAny   RunAsAny    RunAsAny
RunAsAny   <no value>   false            ["*"]
#
```

ロールとロールのバインドを作成します

Astra Data Store プレビューのデフォルトのサービスアカウントで使用する必要なロールとロールのバインドを作成します。

次の YAML 定義は ``astrads.netapp.io` API グループの Astra Data Store プレビューリソースで必要なさまざまな役割 (役割のバインドを使用) を割り当てます

1. 定義されたロールとロールのバインドを作成します。

```
kubectl create -f oc_role_bindings.yaml
```

例： OC_ROLE_bindings. yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: privcrole
rules:
- apiGroups:
  - security.openshift.io
  resourceNames:
  - ads-privileged
  resources:
  - securitycontextconstraints
  verbs:
  - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: default-scc-rolebinding
  namespace: astrads-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: privcrole
```

```
subjects:
- kind: ServiceAccount
  name: default
  namespace: astrads-system
---

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ownerref
  namespace: astrads-system
rules:
- apiGroups:
  - astrads.netapp.io
  resources:
  - '*/finalizers'
  verbs:
  - update
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: or-rb
  namespace: astrads-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: ownerref
subjects:
- kind: ServiceAccount
  name: default
  namespace: astrads-system
```

ワーカーノードを準備します

Astra データストアのワーカーノードでクラスタのプレビュー導入を準備この手順は、 Astra データストアプレビュークラスタで使用されているすべてのワーカーノードで実行します。

OpenShift では、 kubelet 構成ファイル（ /var/lib/kubebelet/config.json ）に JSON 形式を使用します。Astra Data Store プレビュークラスタは 'kubelet config' ファイルの YAML 形式を検索します

手順

1. クラスタのインストールを開始する前に ' 各ワーカー・ノードに /var/lib/kubelet/config.yaml ファイルを作成します

```
sudo cp /var/lib/kubelet/config.json /var/lib/kubelet/config.yaml`
```

2. クラスタ YAML が適用される前に、すべての Kubernetes ノードでこの手順を完了します。

この操作を行わないと、 Astra データストアプレビュークラスタのインストールが失敗します。

Astra Data Store プレビューオペレータをインストール

1. Astra データストアのプレビューマニフェストを表示する：

```
ls manifests/*yaml
```

対応：

```
manifests/astradscluster.yaml
manifests/astradsoperator.yaml
manifests/astradsversion.yaml
manifests/monitoring_operator.yaml
```

2. kubectl apply コマンドを使用して ' オペレータを配備します

```
kubectl apply -f ./manifests/astradsoperator.yaml
```

対応：

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsautosupports.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslicenses.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnfsoptions.astrads.
```

```
netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnodeinfoes.astrads.
netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsqospolicies.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumefiles.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumes.astrads.net
app.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.ast
rads.netapp.io created
role.rbac.authorization.k8s.io/astrads-leader-election-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-metrics-reader created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-viewer-
role created
```

```
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-proxy-role created
rolebinding.rbac.authorization.k8s.io/astrads-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
clusterrolebinding.rbac.authorization.k8s.io/astrads-proxy-rolebinding
created
configmap/astrads-autosupport-cm created
configmap/astrads-firetap-cm created
configmap/astrads-fluent-bit-cm created
configmap/astrads-kevents-asup created
configmap/astrads-metrics-cm created
service/astrads-operator-metrics-service created
deployment.apps/astrads-operator created
```

3. Astra データストアオペレータポッドが起動し、実行中であることを確認します。

```
kubectl get pods -n astrads-system
```

対応：

```
NAME                                  READY   STATUS    RESTARTS   AGE
astrads-operator-5ffb94fbf-7ln4h      1/1     Running   0          17m
```

Astra Data Store プレビュー版 YAML を導入します

1. kubectl apply コマンドを使用して配備します

```
kubectl apply -f ./manifests/astradsversion.yaml
```

2. ポッドが実行されていることを確認します。

```
kubectl get pods -n astrads-system
```

対応：

```
NAME                                              READY   STATUS    RESTARTS
AGE
astrads-cluster-controller-7f6f884645-xxf2n   1/1     Running   0
117s
astrads-ds-nodeinfo-astradsversion-2jqnk      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-dbk7v      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-rn9tt      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-vsmhv      1/1     Running   0
2m7s
astrads-license-controller-fb8fd56bc-bxq7j    1/1     Running   0
2m2s
astrads-operator-5ffb94fbf-7ln4h              1/1     Running   0
2m10s
```

Astra データストアプレビューライセンスを適用

1. プレビュー版への登録時に入手したネットアップライセンスファイル（ NLF ）を適用します。コマンドを実行する前に、使用しているクラスタの名前（「 <AstrA-Data-Store-cluster-name>` ）を入力します 導入に進みます または ' すでに配備されているか ' ライセンス・ファイルへのパス (<file_path/file.txt>) があります

```
kubectl astrads license add --license-file-path <file_path/file.txt>
--ads-cluster-name <Astra-Data-Store-cluster-name> -n astrads-system
```

2. ライセンスが追加されたことを確認します。

```
kubectl astrads license list
```

対応：

```
NAME         ADSCLUSTER                   VALID   PRODUCT
EVALUATION   ENDDATE     VALIDATED
p100000006   astrads-example-cluster    true    Astra Data Store Preview
true         2022-01-23   2021-11-04T14:38:54Z
```

Astra データストアプレビュークラスタをインストール

1. YAML ファイルを開きます。

```
vim ./manifests/astradscluster.yaml
```

2. YAML ファイルで次の値を編集します。

   YAML ファイルの簡単な例は、次の手順を実行します。

   a. （必須） * Metadata* ：「 metadata 」で、「 name 」の文字列をクラスタの名前に変更します。このクラスタ名は、ときと同じである必要があります ライセンスを適用します。

   b. (必須)**Spec**:'spec' の次の必須値を変更します

      - 「 mvip 」文字列を、クラスタ内の任意のワーカーノードからルーティング可能なフローティング管理 IP の IP アドレスに変更します。
      - 「 adsDataNetworks 」に、 NetApp ボリュームをマウントするホストからルーティング可能なフローティング IP アドレス（「アドレス」）をカンマで区切って追加します。ノードごとに 1 つのフローティング IP アドレスを使用します。データネットワーク IP アドレスは、 Astra Data Store のプレビューノードと同じ数以上必要です。Astra データストアプレビューの場合、少なくとも 4 つのアドレスを意味します。あとで 5 つのノードにクラスタを拡張する予定の場合は、 5 つのアドレスを意味します。
      - 「 adsDataNetworks 」で、データネットワークが使用するネットマスクを指定します。
      - 「 adsNetworkInterfaces 」で、「 mgmt 」と「 data 」の値を、管理、クラスタ、およびストレージに使用するネットワークインターフェイス名に置き換えます。名前を指定しない場合、ノードのプライマリインターフェイスが管理、クラスタ、ストレージのネットワークに使用されます。

        クラスタとストレージのネットワークのインターフェイスが同じである必要があります。Astra Data Store プレビュー管理インターフェイスは、 Kubernetes ノードの管理インターフェイスと同じである必要があります。

   c. （任意） * monitoringConfig* ：を設定する場合 監視オペレータ （監視に Astra Control Center を使用していない場合はオプション）、セクションからコメントを削除し、エージェント CR （監視用オペレータリソース）が適用されるネームスペース（デフォルトは「 NetApp-monitoring 」）を追加し、前の手順で使用したレジストリ（「 Your_registry_path 」）のリポジトリパスを追加します。

   d. （任意） * autoSupportConfig * ：を保持します "AutoSupport" プロキシを設定する必要がない場合のデフォルト値は次のとおりです。

      - 「 ProxyURL 」の場合は、 AutoSupport バンドルの転送に使用するポートにプロキシの URL を設定します。

        ほとんどのコメントは YAML サンプルから削除されています。

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
```

```
    cpu: 9
    memory: 34
  adsNodeCount: 4
  mvip: ""
  adsDataNetworks:
    - addresses: ""
      netmask:
  # Specify the network interface names to use for management, cluster
and storage networks.
  # If none are specified, the node's primary interface will be used for
management, cluster and storage networking.
  # To move the cluster and storage networks a different interface than
management, specify all three interfaces to use here.
  # The Astra Data Store management interface should be same as the
Kubernetes node’s management interface.
  # NOTE: The cluster and storage networks need to be on the same
interface.
  adsNetworkInterfaces:
    managementInterface: "mgmt"
    clusterInterface: "data"
    storageInterface: "data"
  # [Optional] Provide a monitoring config to be used to setup/configure
a monitoring agent.
 # monitoringConfig:
   # namespace: "netapp-monitoring"
   # repo: "[YOUR REGISTRY]"
  autoSupportConfig:
    autoUpload: true
    enabled: true
    coredumpUpload: false
    historyRetentionCount: 25
    destinationURL: "https://support.netapp.com/put/AsupPut"
    # ProxyURL defines the URL of the proxy with port to be used for
AutoSupport bundle transfer
    # proxyURL:
    periodic:
      - schedule: "0 0 * * *"
        periodicconfig:
        - component:
            name: storage
            event: dailyMonitoring
          userMessage: Daily Monitoring Storage AutoSupport bundle
          nodes: all
        - component:
            name: controlplane
            event: daily
```

```
            userMessage: Daily Control Plane AutoSupport bundle
```

3. kubectl apply を使用してクラスタを導入します

```
kubectl apply -f ./manifests/astradscluster.yaml
```

4. SELinux が有効になっている場合は、 Astra Data Store プレビュークラスタ内のノードで、次のディレクトリの「 SELinux 」コンテキストにラベルを付け直します。

```
sudo chcon -R -t container_file_t
/var/opt/netapp/firetap/rootfs/var/asup/notification/firetap/
```

```
sudo chcon -R -t container_file_t /var/netapp/firetap/firegen/persist/
```

> これは 'SELinux がこれらのディレクトリの書き込みを禁止し ' サポートポッドが CrashLoopBackoff 状態になるためですこの手順は、 Astra データストアプレビュークラスタ内のすべてのノードで実行する必要があります。

5. クラスタ作成処理が完了するまで数分待ってから、ポッドが実行されていることを確認します。

```
kubectl get pods -n astrads-system
```

回答例：

```
NAME READY STATUS RESTARTS AGE
astrads-cluster-controller-7c67cc7f7b-2jww2 1/1 Running 0 7h31m
astrads-deployment-support-788b859c65-2qjkn 3/3 Running 19 12d
astrads-ds-astrads-cluster-1ab0dbc-j9jzc 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-1ab0dbc-k9wp8 1/1 Running 0 5d1h
astrads-ds-astrads-cluster-1ab0dbc-pwk42 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-1ab0dbc-qhvc6 1/1 Running 0 8h
astrads-ds-nodeinfo-astradsversion-gcmj8 1/1 Running 1 12d
astrads-ds-nodeinfo-astradsversion-j826x 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-vdthh 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-xwgsf 1/1 Running 0 12d
astrads-ds-support-828vw 2/2 Running 2 5d2h
astrads-ds-support-cfzts 2/2 Running 0 8h
astrads-ds-support-nzkkr 2/2 Running 15 7h49m
astrads-ds-support-xxbnp 2/2 Running 1 5d2h
astrads-license-controller-86c69f76bb-s6fb7 1/1 Running 0 8h
astrads-operator-79ff8fbb6d-vpz9m 1/1 Running 0 8h
```

6. クラスタの導入の進捗を確認します。

```
kubectl get astradscluster -n astrads-system
```

回答例：

```
NAME                         STATUS    VERSION    SERIAL NUMBER    MVIP
AGE

astrads-example-cluster   created   2021.10.0   p100000006
10.x.x.x   10m
```

導入に関連するイベントを把握

クラスタの導入中に ' オペレーション・ステータスは ' ブランクから ' 進行中 ' から作成済みに変更する必要がありますクラスタの導入には約 8~10 分かかります。導入中にクラスタイベントを監視するには、次のいずれかのコマンドを実行します。

```
kubectl get events --field-selector involvedObject.kind=AstraDSCluster -n
astrads-system
```

```
kubectl describe astradscluster <cluster name> -n astrads-system
```

導入時の主なイベントを次に示します。

イベントメッセージ	意味
ADS クラスタに参加するコントロールプレーンノードを 4 つ選択しました	Astra Data Store プレビューオペレータは、 Astra データストアプレビュークラスタを構築するために、 CPU 、メモリ、ストレージ、ネットワークを備えた十分なノードを特定しました。
ADS クラスタが作成中です	Astra データストアプレビュークラスタコントローラが、クラスタ作成処理を開始しました。
ADS クラスタが作成されました	クラスタが作成されました。

クラスタのステータスが「 in progress 」に変わらない場合は、オペレータログでノード選択の詳細を確認します。

```
kubectl logs -n astrads-system <astrads operator pod name>
```

クラスタのステータスが「処理中」のままである場合は、クラスタコントローラのログを確認します。

```
kubectl logs -n astrads-system <astrads cluster controller pod name>
```

Astra データストアのプレビュー監視を設定

Astra データストアプレビューは、 Astra Control Center の監視用、または別のテレメトリサービスによる監視用に設定できます。

Astra Control Center プレビューの監視を設定します

次の手順は、 Astra データストアのプレビューが Astra Control Center のバックエンドとして管理された後にのみ実行します。

1. Astra Control Center によるモニタリングのための Astra データストアプレビューの構成：

   ```
   kubectl astrads monitoring -m netapp-monitoring -r [YOUR REGISTRY] setup
   ```

監視オペレータをインストールします

（オプション） Astra データストアのプレビューが Astra Control Center にインポートされない場合にのみ、監視オペレータをお勧めします。モニタリングオペレータは、アストラデータストアプレビューインスタンスがスタンドアロン環境である場合、 Cloud Insights を使用してテレメトリを監視する場合、または Elastic などのサードパーティのエンドポイントにログをストリーミングする場合にインストールできます。

1. 次のインストールコマンドを実行します。

```
kubectl apply -f ./manifests/monitoring_operator.yaml
```

2. Astra データストアプレビューで監視を設定：

```
kubectl astrads monitoring -m netapp-monitoring -r [YOUR REGISTRY] setup
```

iptables をセットアップします

OpenShift SDN （ OpenShift 用のデフォルト CNI プラグイン）は、ホストループバックインターフェイス（ 127.0.0.1 / localhost ）からホストポートへのトラフィックを偽装しません。

Astra Data Store プレビュークラスタでは、 localhost からクラスタサービスポート（ 9920 ）にトラフィックを転送するための NAT ルールが追加されている必要があります。

手順

1. NAT テーブルの出力チェーンの現在の「 KUBE-HOSTPORTS 」ルールの行番号をメモします。

   次の例では 'KUBE-HOSTPORTS' は 4 番目の位置にあります

```
$ sudo iptables -t nat -L OUTPUT --line-numbers
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 KUBE-SERVICES all -- anywhere anywhere /* kubernetes service portals
*/
2 KUBE-PORTALS-HOST all -- anywhere anywhere /* handle ClusterIPs; NOTE:
this must be before the NodePort rules */
3 KUBE-NODEPORT-HOST all -- anywhere anywhere ADDRTYPE match dst-type
LOCAL /* handle service NodePorts; NOTE: this must be the last rule in
the chain */
4 KUBE-HOSTPORTS all -- anywhere anywhere /* kube hostport portals */
ADDRTYPE match dst-type LOCAL
```

2. 新しいルールを NAT テーブルに追加します。この場合、「 KUBE-HOSTPORTS 」エントリのすぐ上に追加します。

```
$ sudo iptables -t nat -I OUTPUT 4 -s 127.0.0.1 -j KUBE-MARK-MASQ -p tcp
--dport 9920
```

3. 新しく追加されたルールが 'KUBE-HOSTPORTS' ルールのすぐ上の NAT テーブルに追加されていることを確認します

```
$ sudo iptables -t nat -L OUTPUT --line-numbers
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 KUBE-SERVICES all -- anywhere anywhere /* kubernetes service portals
*/
2 KUBE-PORTALS-HOST all -- anywhere anywhere /* handle ClusterIPs; NOTE:
this must be before the NodePort rules */
3 KUBE-NODEPORT-HOST all -- anywhere anywhere ADDRTYPE match dst-type
LOCAL /* handle service NodePorts; NOTE: this must be the last rule in
the chain */
4 KUBE-MARK-MASQ tcp -- localhost anywhere tcp dpt:9920
5 KUBE-HOSTPORTS all -- anywhere anywhere /* kube hostport portals */
ADDRTYPE match dst-type LOCAL
```

Astra Data Store プレビュー版クラスタ内のすべてのノードで、次の手順を実行します。

Astra データストアプレビュークラスタノードが再起動した場合は、この手順を繰り返します。

次の手順

を実行して導入を完了します "セットアップのタスク"。