



# はじめに Astra Data Store

NetApp  
February 09, 2022

# 目次

はじめに .....	1
Astra データストアのプレビュー要件 .....	1
Astra データストアプレビューのクイックスタート .....	5
Astra データストアのインストールの概要 .....	6
Astra Data Store のプレビューコンポーネントをセットアップする .....	32
Astra データストアのプレビュー制限 .....	42
Astra データストアのプレビューに関する FAQ .....	42

# はじめに

## Astra データストアのプレビュー要件

まずは、Astra データストアのプレビュー要件を満たす環境であることを確認してください。

Astra データストアプレビュー：ベアメタル環境と VM ベース環境の両方をサポートAstra データストアプレビュークラスタは、4 つ以上のワーカーノードを含む Kubernetes クラスタで実行できます。Astra データストアプレビューソフトウェアは、同じ Kubernetes クラスタで実行されている他のアプリケーションと共存できます。

Astra Data Store のプレビューでは、Astra Trident CSI ドライバを使用して Kubernetes ワークロード用の永続的ボリュームのみをプロビジョニングできます。VM ワークロードは、Astra データストアの今後のリリースでサポートされる予定です。



Astra Control Center から Astra データストアプレビュークラスタを管理する場合は、Astra データストアプレビュークラスタが満たしていることを確認します ["Astra Control Center で管理するクラスタの要件"](#) ここに記載されている要件に加えて、

### Kubernetes ワーカーノードのリソース要件

Kubernetes クラスタ内の各ワーカーノード上の Astra Data Store プレビューソフトウェアに割り当てるために必要なリソース要件は次のとおりです。

リソース	最小（ Minimum ）	最大
データドライブの数	<ul style="list-style-type: none"><li>• 3（独立したキャッシュデバイスを使用）</li><li>• 4（キャッシュデバイスがない場合）</li></ul>	14
データドライブのサイズ	100GiB	4TiB 未満
オプションのキャッシュデバイスの数	1（8GiB 以上）	該当なし
vCPU の数	10.	10.
RAM	35GiB	35GiB



書き込みパフォーマンスを最大限に高めるには、専用の耐久性、低レイテンシ、低容量キャッシュデバイスを構成する必要があります。

各ワーカーノードには、次の追加要件があります。

- 100GiB 以上の空き容量がホストディスク（ブート）にあり、Astra データストアプレビューログファイルを保存します。
- クラスタトラフィック、データトラフィック、および管理トラフィック用に、10GbE 以上のネットワークインターフェイスが少なくとも 1 つ必要です。必要に応じて、1GbE 以上のインターフェイスを追加して管理トラフィックを分離できます。

## ハードウェアとソフトウェアの要件

Astra Data Store プレビューソフトウェアは、次のハードウェアプラットフォーム、ソフトウェア、ストレージ構成で検証済みです。にアクセスします ["ネットアップコミュニティによるサポート"](#) Kubernetes クラスタの構成が異なる場合。

### ハードウェアプラットフォーム

- HPE DL360
- HPE DL380
- Dell R640
- Dell R740

Astra データストアプレビューは、次のドライブタイプで検証済みです。

- \* ベアメタル環境 \* : ハイパーバイザーを使用せずに Linux クラスタ上の Kubernetes クラスタに Astra データストアプレビューを直接インストール
  - NVMe TLC SSD
- \* VM ベースの導入 \* : ESXi クラスタでホストされた Linux VM 上の Kubernetes クラスタに Astra データストアプレビューをインストール
  - SAS または NVMe TLC SSD ベースのデータストア
  - 仮想ディスクまたはパススルードライブとして提供されるドライブ



ホストがハードウェア RAID コントローラの背後で SSD を使用している場合は、「パススルー」モードを使用するようにハードウェア RAID コントローラを設定します。



各ドライブのシリアル番号は一意である必要があります。VM 作成中に仮想マシンの詳細設定に 'disk.enableuuid=true' 属性を追加します

### ソフトウェア

- ハイパーバイザー: Astra データストアプレビュー版は、ESXi 7.0 を使用する VMware ベースの VM 環境で検証済みです。KVM ベースの導入は、Astra データストアプレビューではサポートされていません。
- Astra データストアプレビューは、次のホストオペレーティングシステムで検証済みです。
  - Red Hat Enterprise Linux 8.4
  - Red Hat Enterprise Linux 8.2 の場合
  - Red Hat Enterprise Linux 7.9
  - Red Hat Enterprise Linux CoreOS (RHCS)
  - CentOS 8
  - Ubuntu 20.04
- Astra Data Store プレビュー版は、以下の Kubernetes ディストリビューションで検証済みです。
  - Red Hat OpenShift 4.7
  - Google Anthos 1.7
  - Kubernetes 1.21



Astra Data Store のプレビュー版では、ストレージのプロビジョニングとオーケストレーションに Trident バージョン 21.10.1 が必要です。を参照してください ["Astra Trident のインストール手順"](#)。

## ネットワーク要件

Astra データストアプレビューを実行するには、クラスタごとに MVIP 用の IP アドレスが 1 つ必要です。MIP と同じサブネット内の未使用の IP アドレスまたは未設定の IP アドレスを指定する必要があります。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。

また、次の表に示すように各ノードを設定することもできます。



この表では、MIP : 管理 IP アドレス CIP : クラスタ IP アドレス MVIP : 管理仮想 IP アドレスの略語を使用しています

設定	IP アドレスが必要です
ノードごとに 1 つのネットワークインターフェイス	<ul style="list-style-type: none"> <li>ノードごとに 2 つ : <ul style="list-style-type: none"> <li>MIP/CIP : ノードごとに管理インターフェイスに設定済みの IP アドレスが 1 つあります</li> <li>データ IP : MIP と同じサブネットに含まれる、ノードごとに未使用の IP アドレスまたは未設定の IP アドレスの 1 つ</li> </ul> </li> </ul>
ノードごとに 2 つのネットワークインターフェイス	<ul style="list-style-type: none"> <li>ノードあたり 3 本 : <ul style="list-style-type: none"> <li>mip : ノードごとに管理インターフェイスで事前に設定された IP アドレスを 1 つ</li> <li>cip : MIP とは異なるサブネット内のノードごとに、データインターフェイスに事前に設定された IP アドレスを 1 つだけ指定します</li> <li>データ IP : CIP と同じサブネット内の各ノードに未使用または未設定の IP アドレスが 1 つあります</li> </ul> </li> </ul>



これらの構成では 'クラスタカスタムリソース (CR) ファイルのデータネットワークゲートウェイフィールド 'astraadscluster.yaml' は省略してください各ノードの既存のルーティング設定には、すべてのアドレスを指定できます。



これらの構成では VLAN タグは使用されません。

## Astra Trident

Astra データストアプレビューを実行するには、Kubernetes クラスタが Astra Trident 21.10.1 を実行してい

する必要があります。Astra データストアプレビューは、として構成できます ["ストレージバックエンド"](#) ネットアップの Trident で永続的ボリュームをプロビジョニング

## CNI 構成

Astra Data Store のプレビューは、次の NNI で検証済みです。

- バニラ Kubernetes クラスタ用 Calico および Weave Net CNI
- Red Hat OpenShift Container Platform （ OCP ） 向け OpenShift SDN
- Google Anthos 向け Cilium

これらの NNI では、ホストファイアウォール（ firewalld ）を無効にする必要があります。

## 永続的ボリュームの共有に関する要件

各アストラデータストアプレビュークラスタでは、永続ボリュームを使用して、そのクラスタにインストールされているアプリケーションのストレージニーズに対応できます。Astra Data Store プレビュー版の永続的ボリュームについては、次の要件を考慮してください。

### 要件

- NFSv4.1 クライアント / サーバを Kubernetes クラスタにインストールする必要があります。
- nfs-utils パッケージはワーカーノードにインストールする必要があります。
- Kubernetes アプリケーションは、NFSv4.1 で共有されている永続的ボリュームを使用してファイルにアクセスします。NFSv4.1 では、AUTH\_SYS の認証方法が必要です。

## ライセンス

Astra データストアのプレビュー機能をフル活用するには、Astra データストアのプレビューライセンスが必要です。 ["こちらから登録してください"](#) から Astra データストアプレビューライセンスを取得できます。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。

## AutoSupport の設定

Astra データストアプレビューを利用するには、AutoSupport を有効にし、AutoSupport バックエンドに接続する必要があります。これは、直接インターネットアクセスまたはプロキシ設定を経由する可能性があります。

。 ["必須のテレメトリ AutoSupport バンドルの送信に使用される定期設定"](#) 変更しないでください。定期的な AutoSupport バンドルの送信を無効にすると、クラスタがロックダウンされ、定期的な設定が再度有効になるまで新しいボリュームを作成できなくなります。

## 次の手順

を表示します ["クイックスタート"](#) 概要（ Overview ）：

を参照してください。

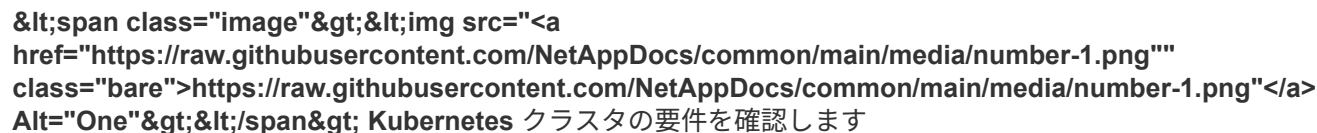
["Astra データストアのプレビュー制限"](#)

# Astra データストアプレビューのクイックスタート

このページでは、Astra データストアのプレビューを開始するために必要な手順の概要を説明します。各ステップ内のリンクから、詳細が記載されたページに移動できます。

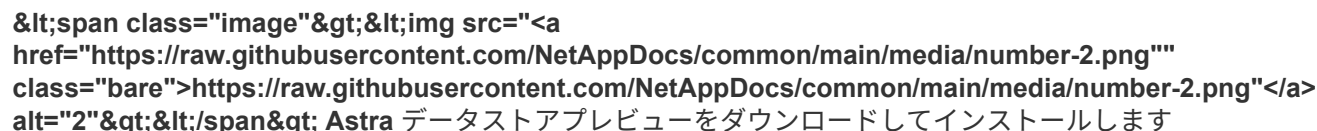
ぜひお試しください。Astra Data Store のプレビューを試す場合は、90 日間のプレビューライセンスを使用できます。

["こちらから登録してください"](#) から Astra データストアプレビューライセンスを取得できます。

 **Kubernetes クラスタの要件を確認します**

- クラスタが正常な状態で稼働し、少なくとも 4 つのワーカーノードがある必要があります。
- Astra データストアプレビュー環境を構成する各 Kubernetes ワーカーノードには、同じインターフェイスタイプ（SATA、SAS、NVMe）の SSD と、Astra データストアプレビュークラスタに割り当てられているドライブの数が同じである必要があります。
- SSD のシリアル番号はそれぞれ一意である必要があります。

の詳細を確認してください ["Astra データストアのプレビュー要件"](#)。

 **Astra データストアプレビューをダウンロードしてインストールします**

- から Astra データストアプレビューをダウンロードします ["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をローカル環境にインストールします。
- Astra データストアプレビューライセンスを適用
- Astra データストアプレビュークラスタをインストール
- Astra データストアのプレビュー監視を設定
- Red Hat OpenShift を使用する場合は、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します。

の詳細を確認してください ["Astra データストアプレビューをインストールしています"](#)。

 **初期設定タスクをいくつか実行します**

- Astra Trident をインストール
- Kubernetes スナップショットカスタムリソース定義（CRD）とコントローラをインストールする。
- Astra データストアプレビューをストレージバックエンドとしてセットアップする。
- デフォルトの Astra データストアプレビューストレージクラスを作成

の詳細については、を参照してください ["初期セットアッププロセス"](#)。

Astra データストアプレビューのセットアップが完了したら、次の手順を実行します。

- kubectl コマンドと kubectl ファイルシステム拡張機能を使用して、ノードの保守モードへの切り替え、ドライブの交換、ノードの交換などのタスクを含む、クラスタを管理します。の詳細を確認してください ["kubectl コマンドを Astra データストアプレビューで使用方法"](#)。
- 監視エンドポイントを設定する。の詳細を確認してください ["監視エンドポイントの設定"](#)。

["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"](#)。

## Astra Data Store deployment



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\_Astrandsstore.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-astras /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.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslicenses.astrads.netapp.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.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.astrads.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	ADSCUSTER	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。

- (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](#)。
- (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\_interface\_name>」および「<cluster\_and\_storage\_interface\_name>」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、前の手順で使ったレジストリ (「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 to a different interface
        # than management, specify all three interfaces to use here.
        # NOTE: The cluster and storage networks need to be on the same
        # interface.
  adsNetworkInterfaces:
    managementInterface: "<mgmt_interface_name>"
    clusterInterface: "<cluster_and_storage_interface_name>"
    storageInterface: "<cluster_and_storage_interface_name>"
    # [Optional] Provide a k8s label key that defines which protection
    # domain a node belongs to.
    # adsProtectionDomainKey: ""
    # [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-lab0dbc-j9jzc	1/1	Running	0	5d2h
astrads-ds-astrads-cluster-lab0dbc-k9wp8	1/1	Running	0	5d1h
astrads-ds-astrads-cluster-lab0dbc-pwk42	1/1	Running	0	5d2h
astrads-ds-astrads-cluster-lab0dbc-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-nzkkkr	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
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行われます。

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。

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

## 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\_Astradsstore.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-astras /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に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に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/kubelet/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.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslicenses.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnfsoptions.astrads.netapp.io created
```

```
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	ADSCUSTER	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\_interface\_name>」および「<cluster\_and\_storage\_interface\_name>」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。
- 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。



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。



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 to a different interface
      than management, specify all three interfaces to use here.
      # NOTE: The cluster and storage networks need to be on the same
      interface.
adsNetworkInterfaces:
  managementInterface: "<mgmt_interface_name>"
  clusterInterface: "<cluster_and_storage_interface_name>"
  storageInterface: "<cluster_and_storage_interface_name>"
  # [Optional] Provide a k8s label key that defines which protection
  domain a node belongs to.
  # adsProtectionDomainKey: ""
  # [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-lab0dbc-j9jzc 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-lab0dbc-k9wp8 1/1 Running 0 5d1h
astrads-ds-astrads-cluster-lab0dbc-pwk42 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-lab0dbc-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-nzkkkr 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
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行われます。

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"](#)。

## Astra Data Store のプレビューコンポーネントをセットアップする

Astra Data Store のプレビューをインストールし、環境に関するいくつかの前提条件に対処したら、Astra Trident をインストールし、Kubernetes のスナップショット機能を設定し、ストレージバックエンドをセットアップして、デフォルトのストレージクラスを作成します。

- [\[Install Astra Trident\]](#)
- [\[Install Kubernetes snapshot CRDs and Controller\]](#)
- [\[Set up Astra Data Store as storage backend\]](#)
- [\[Create a default Astra Data Store storage class\]](#)

### Astra Trident をインストール

Astra データストアプレビュー版を利用するには、Astra Trident 21.10.1 をインストールする必要があります。Trident は次のいずれかの方法でインストールできます。

- ["tridentctl を使用して Astra Trident をインストールします"](#)。
- ["Trident オペレータを使用して Astra Trident をインストール"](#)。



Trident オペレータは、手動または Helm を使用して導入できます。

### Kubernetes スナップショットの CRD とコントローラをインストールします

永続的ボリューム要求（PVC）の Snapshot を作成するには、Kubernetes の Snapshot SSD とコントローラが必要です。環境に CRD とコントローラがインストールされていない場合は、次のコマンドを実行してインストールします。



次のコマンド例では 'ディレクトリとして /trident' を想定していますが '使用するディレクトリは 'YAML ファイルのダウンロードに使用した任意のディレクトリにすることができます

必要なもの

- ["インストールを開始する前に、Astra データストアプレビュー環境を準備します"](#)。
- をダウンロードします ["Kubernetes snapshot controller yaml ファイル"](#)：
  - setup-snapshot-controller.yaml

- rbac -snapshot-controller.yaml
- をダウンロードします "YAML CRD" :
  - snapshot.storage.k8s.io\_volumesnapshotclasses.yaml
  - snapshot.storage.k8s.io\_volumesnapshotcontents.yaml
  - snapshot.storage.k8s.io\_volumesnapshots.yaml

## 手順

1. snapshot.storage.k8s.io\_volumesnapshotclasses.yaml を適用します。

```
kubectl apply -f
trident/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
```

対応:

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshotclasses.snapshot.storage.k8s.io created
```

2. snapshot.storage.k8s.io\_volumesnapshotcontents.yaml を適用します。

```
kubectl apply -f
trident/snapshot.storage.k8s.io_volumesnapshotcontents.yaml
```

対応:

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshotcontents.snapshot.storage.k8s.io created
```

3. snapshot.storage.k8s.io\_volumesnapshots.yaml を適用します。

```
kubectl apply -f trident/snapshot.storage.k8s.io_volumesnapshots.yaml
```

対応:

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshots.snapshot.storage.k8s.io created
```

4. setup-snapshot-controller.yaml を適用します。

```
kubectl apply -f trident/setup-snapshot-controller.yaml
```

対応：

```
deployment.apps/snapshot-controller created
```

##### 5. RBAC の -snapshot-controller.yaml を適用します。

```
kubectl apply -f trident/rbac-snapshot-controller.yaml
```

対応：

```
serviceaccount/snapshot-controller created
clusterrole.rbac.authorization.k8s.io/snapshot-controller-runner created
clusterrolebinding.rbac.authorization.k8s.io/snapshot-controller-role
created
role.rbac.authorization.k8s.io/snapshot-controller-leaderelection
created
rolebinding.rbac.authorization.k8s.io/snapshot-controller-leaderelection
created
```

##### 6. CRD YAML ファイルが適用されていることを確認します。

```
kubectl get crd | grep volumesnapshot
```

回答例：

```
astradsvolumesnapshots.astrads.netapp.io          2021-08-
04T17:48:21Z
volumesnapshotclasses.snapshot.storage.k8s.io      2021-08-
04T22:05:49Z
volumesnapshotcontents.snapshot.storage.k8s.io      2021-08-
04T22:05:59Z
volumesnapshots.snapshot.storage.k8s.io            2021-08-
04T22:06:17Z
```

##### 7. Snapshot コントローラファイルが適用されたことを確認します。

```
kubectl get pods -n kube-system | grep snapshot
```

回答例：

```
snapshot-controller-7f58886ff4-cdh78
1/1      Running    0          13s
snapshot-controller-7f58886ff4-tmrd9
1/1      Running    0          32s
```

## Astra データストアをストレージバックエンドとしてセットアップする

ads\_backend.json ファイルにストレージバックエンドパラメータを設定し、Astra データストアストレージバックエンドを作成する。

手順

1. 安全な端末を使用して「ads\_backend.json」を作成します。

```
vi ads_backend.json
```

2. JSON ファイルを設定します。

- a. 「cluster」の値を Astra Data Store クラスタのクラスタ名に変更します。
- b. 「namespace」の値を、ボリュームの作成に使用するネームスペースに変更します。
- c. バックエンドではなく 'exportpolicy-CR を設定している場合を除き 'autoExportPolicy' の値を true に変更します
- d. 「autoExportCIDRs」リストに、アクセスを許可する IP アドレスを入力します。すべてを許可するには '0.0.0.0/0' を使用します
- e. 「kubeconfig」の値については、次の手順を実行します。
  - i. .kube/config YAML ファイルをスペースなしの JSON 形式に変換して最小化します。

変換例：

```
python3 -c 'import sys, yaml, json;
json.dump(yaml.load(sys.stdin), sys.stdout, indent=None)' <
~/.kube/config > kubeconfig.json
```

- ii. base64 としてエンコードし、base64 出力を「kubeconfig」値に使用します。

エンコーディングの例：

```
cat kubeconfig.json | base64 | tr -d '\n'
```

```

{
  "version": 1,
  "storageDriverName": "astrads-nas",
  "storagePrefix": "",
  "cluster": "example-1234584",
  "namespace": "astrads-system",
  "autoExportPolicy": true,
  "autoExportCIDRs": ["0.0.0.0/0"],
  "kubeconfig": "<base64_output_of_kubeconf_json>",
  "debugTraceFlags": {"method": true, "api": true},
  "labels": {"cloud": "on-prem", "creator": "trident-dev"},
  "defaults": {
    "qosPolicy": "bronze"
  },
  "storage": [
    {
      "labels": {
        "performance": "extreme"
      },
      "defaults": {
        "qosPolicy": "bronze"
      }
    },
    {
      "labels": {
        "performance": "premium"
      },
      "defaults": {
        "qosPolicy": "bronze"
      }
    },
    {
      "labels": {
        "performance": "standard"
      },
      "defaults": {
        "qosPolicy": "bronze"
      }
    }
  ]
}

```

3. Trident インストーラをダウンロードしたディレクトリに移動します。

```
cd <trident-installer or path to folder containing tridentctl>
```

#### 4. ストレージバックエンドを作成します。

```
./tridentctl create backend -f ads_backend.json -n trident
```

回答例：

```
+-----+-----+
+-----+-----+-----+
|      NAME      | STORAGE DRIVER |                      UUID
| STATE  | VOLUMES |
+-----+-----+
+-----+-----+-----+
| example-1234584 | astrads-nas    | 2125fa7a-730e-43c8-873b-
6012fcc3b527 | online |          0 |
+-----+-----+
+-----+-----+-----+
```

## Default Astra Data Store ストレージクラスを作成

Astra Trident のデフォルトのストレージクラスを作成し、ストレージバックエンドに適用

手順

#### 1. trident-csi ストレージクラスを作成します。

##### a. ads\_sc\_example.yaml を作成します：

```
vi ads_sc_example.yaml
```

対応：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: trident-csi
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
mountOptions:
  - vers=4.1
```

b. trident-csi の作成：

```
kubectl create -f ads_sc_example.yaml
```

対応：

```
storageclass.storage.k8s.io/trident-csi created
```

2. ストレージクラスが追加されたことを確認します。

```
kubectl get storageclass -A
```

対応：

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION	AGE		
trident-csi	csi.trident.netapp.io	Delete	Immediate
true	6h29m		

3. Trident インストーラをダウンロードしたディレクトリに移動します。

```
cd <trident-installer or path to folder containing tridentctl>
```

4. Astra Trident バックエンドがデフォルトのストレージクラスパラメータで更新されたことを確認します。

```
./tridentctl get backend -n trident -o yaml
```

回答例：



```

items:
- backendUUID: 2125fa7a-730e-43c8-873b-6012fcc3b527
  config:
    autoExportCIDRs:
    - 0.0.0.0/0
    autoExportPolicy: true
    backendName: ""
    cluster: example-1234584
    credentials: null
    debug: false
    debugTraceFlags:
      api: true
      method: true
    defaults:
      exportPolicy: default
      qosPolicy: bronze
      size: 1G
      snapshotDir: "false"
      snapshotPolicy: none
    disableDelete: false
    kubeconfig: <ID>
    labels:
      cloud: on-prem
      creator: trident-dev
    limitVolumeSize: ""
    namespace: astrads-system
    nfsMountOptions: ""
    region: ""
    serialNumbers: null
    storage:
      - defaults:
          exportPolicy: ""
          qosPolicy: bronze
          size: ""
          snapshotDir: ""
          snapshotPolicy: ""
        labels:
          performance: extreme
          region: ""
          supportedTopologies: null
          zone: ""
      - defaults:
          exportPolicy: ""
          qosPolicy: bronze
          size: ""

```

```

    snapshotDir: ""
    snapshotPolicy: ""
  labels:
    performance: premium
    region: ""
    supportedTopologies: null
    zone: ""
  - defaults:
    exportPolicy: ""
    qosPolicy: bronze
    size: ""
    snapshotDir: ""
    snapshotPolicy: ""
  labels:
    performance: standard
    region: ""
    supportedTopologies: null
    zone: ""
  storageDriverName: astrads-nas
  storagePrefix: ""
  supportedTopologies: null
  version: 1
  zone: ""
configRef: ""
name: example-1234584
online: true
protocol: file
state: online
storage:
  example-1234584_pool_0:
    name: example-1234584_pool_0
    storageAttributes:
      backendType:
        offer:
          - astrads-nas
      clones:
        offer: true
      encryption:
        offer: false
      labels:
        offer:
          cloud: on-prem
          creator: trident-dev
          performance: extreme
    snapshots:
      offer: true

```

```

storageClasses:
- trident-csi
supportedTopologies: null
example-1234584_pool_1:
name: example-1234584_pool_1
storageAttributes:
  backendType:
    offer:
      - astrads-nas
  clones:
    offer: true
  encryption:
    offer: false
  labels:
    offer:
      cloud: on-prem
      creator: trident-dev
      performance: premium
  snapshots:
    offer: true
storageClasses:
- trident-csi
supportedTopologies: null
example-1234584_pool_2:
name: example-1234584_pool_2
storageAttributes:
  backendType:
    offer:
      - astrads-nas
  clones:
    offer: true
  encryption:
    offer: false
  labels:
    offer:
      cloud: on-prem
      creator: trident-dev
      performance: standard
  snapshots:
    offer: true
storageClasses:
- trident-csi
supportedTopologies: null
volumes: []

```

# Astra データストアのプレビュー制限

Astra データストアは、クラウドネイティブアプリケーションの管理を支援する、Kubernetes ネイティブのオンプレミスデータセンター向け共有ファイル Software-Defined Storage（SDS）解決策です。

Astra Data Store プレビューリリースには、次のリソース制限があります。

リソース	最小（Minimum）	最大
Astra データストアプレビュークラスタに含まれるノード数	4.	5.
ノードあたりの永続ボリュームの数	該当なし	10.
ノードあたりの永続ボリュームのプロビジョニング済み容量の合計	該当なし	1TiB
ボリュームサイズ	20MiB	1TiB
ボリュームあたりの Snapshot 数	0	256
ボリュームあたりのクローン数	0	9.



Astra データストアプレビュー版は、VM ワークロードをサポートしていない。VMware VVOL ワークロードは今後のリリースでサポートする予定です。



Astra データストアのプレビューはパフォーマンスを調整したものであり、パフォーマンスの特性評価には使用しないでください。

## Astra データストアのプレビューに関する FAQ

Astra Data Store のプレビュー版のインストール、設定、アップグレード、トラブルシューティングに関する FAQ を掲載しています。

### 一般的な質問

- 本番環境で Astra データストアプレビューを使用できますか？ \* いいえ Astra データストアはエンタープライズクラスの耐障害性を実現するように設計、開発されていますが、Astra データストアプレビュー版は本番環境のワークロードには適していません。
- 仮想マシンのワークロードに対して Astra Data Store プレビューを使用できますか？ \* Astra Data Store プレビューリリースは、ベアメタルマシンでも仮想マシンでも Kubernetes で実行されているアプリケーションに限定されます。今後のリリースでは、Kubernetes と ESXi 仮想マシンの両方でアプリケーションがサポートされる予定です。を参照してください ["Astra データストアの要件"](#)。
- Astra Data Store のプレビュー版には、他のネットアップ製品との依存関係はありますか？ \*

はい。Astra Data Store のプレビューを利用するには、NetApp CSI ドライバ Astra Trident バージョン 21.10.1 以降をワークロードの Kubernetes クラスタに導入する必要があります。詳細はこちら ["Astra データストアの要件"](#)。

Astra Data Store プレビュー版クラスタをストレージバックエンドとして使用するアプリケーションであれば ["Astra Control Center の略"](#) バージョン 21.12 データ保護、ディザスタリカバリ、Kubernetes ワークロードの

移行など、アプリケーション対応のデータ管理機能を活用するために必要です。

- Astra Data Store Preview Cluster の管理方法 \* Astra Data Store のプレビュー資産の管理には、kubectl コマンドと Kubernetes API 拡張機能を使用できます。

'kubectl astras コマンド'には '-h' スイッチが含まれており ' 便利な使用法とフラグ・ドキュメントが提供されています

- アストラデータストアのプレビュークラスター指標はどのように監視できますか？ \* Cloud Insights を使用して、アストラデータストアのプレビュー指標を監視できます。を参照してください ["Cloud Insights で指標を監視"](#)。

ログを監視することもできます。を参照してください ["イベントログを設定して監視する"](#)。

- Kubernetes クラスターで ONTAP や他のストレージプロバイダとともに Astra データストアプレビューを使用できますか？ \* はい。Astra データストアプレビューは、アプリケーションクラスター内の他のストレージプロバイダとともに使用できます。
- Astra Trident は、Astra Data Store プレビューから Kubernetes クラスターを削除した場合にアンインストールされますか？ \* Astra Trident は、Astra Data Store プレビューをアンインストールしてもクラスターからアンインストールされません。Astra Trident のアンインストールが必要な場合は、別途アンインストールする必要があります。

## ライセンス

- Astra Data Store プレビューにはライセンスが必要ですか？ \* はい、Astra Data Store プレビューにはネットアップライセンスファイル（NLF）が必要です。

を参照してください ["Astra データストアの要件"](#)。

- Astra データストアプレビューライセンスの有効期間はどのくらいですか？ \* Astra データストアプレビューライセンスのデフォルト期間は、ダウンロード日から 90 日間です。

## Kubernetes クラスターに Astra データストアプレビューをインストールして使用

- ベアメタルまたは仮想マシンで実行されている Kubernetes クラスターに Astra Data Store プレビューをインストールできますか？ \* はい。Astra データストアプレビューは、ベアメタルまたは ESXi VM で実行されている Kubernetes クラスターにインストールできます。を参照してください ["Astra データストアのプレビュー要件"](#)。
- Astra Data Store プレビュー版でサポートされている Kubernetes のバージョンは何ですか。 \*

Astra Data Store プレビューは、v1.20 以降と互換性のある Kubernetes ディストリビューションで機能します。ただし、現時点では、Kubernetes のすべてのディストリビューションで検証されているわけではありません。詳細はこちら ["Astra データストアのプレビュー要件"](#)。

- My Kubernetes クラスターは 5 つ以上のワーカーノードで構成されています。Astra データストアプレビューを IT にインストールできますか。 \* はい。Astra データストアプレビュークラスターは、Kubernetes クラスター内の 4 つのワーカーノードに導入できます。導入後、クラスターを 5 つのワーカーノードに拡張できます。
- Astra データストアプレビューは、プライベートレジストリからのオフラインインストールをサポートしていますか。 \* はい。Astra データストアプレビューは、ローカルレジストリからオフラインでインストールできます。を参照してください ["Astra データストアプレビューをインストールします"](#)。ただし、

Astra データストアプレビューを継続的に利用するには、NetApp AutoSupport バックエンド（[support.netapp.com](https://support.netapp.com)）に（直接またはプロキシ経由で）接続する必要があります。

- Astra データストアプレビューを使用するにはインターネットに接続する必要がありますか？ \* Astra データストアプレビューを利用するには、必須の AutoSupport バンドルを定期的送信するために、ネットアップ AutoSupport バックエンドに接続する必要があります。直接接続かプロキシ経由で接続できます。この接続がないか AutoSupport が無効な場合、クラスタがロックダウンされ、定期的なバンドルアップロードが再開されるまで新しいボリュームの作成が無効になります。
- Astra Data Store プレビューで使用する役割と権限は何ですか？ \* Astra Data Store プレビューオペレータを配備するには、`kubeadmin` である必要があります。

Astra Data Store のプレビューには、ノードの選択に使用されるノードリソースを検出するための「`astra -ds -nodeinfo -astradsversion`」という特権的なデミスがあります。

さらに、管理者は、権限付き Kubernetes ジョブを使用して、選択したワーカーノードにストレージクラスタのコンテナをインストールし、Astra Data Store プレビューストレージクラスタを構築します。

- Astra Data Store プレビューインストール用に更新する必要があるマニフェストファイルは何ですか？ \* からダウンロードした Astra Data Store プレビューバンドルから ["ネットアップサポートサイト"](#)では、次のマニフェストが表示されます。
- `Astradscluster.yaml`
- `Astradsoperator.yaml`
- `astadsversion.yaml`
- `Monitoring_operator.yaml`

配備固有の設定で `'astraadscluster.yaml'` マニフェストを更新する必要がありますを参照してください ["Astra データストアプレビューをインストールします"](#)。

## トラブルシューティングとサポート

ネットアップのコンテナ向け Slack チャンネルを使用して、ネットアップの Astra データストアプレビューでコミュニティサポートにアクセスできます。このチャンネルの監視は、ネットアップのサポートエンジニアとテクニカルマーケティングエンジニアが行います。

### ["ネットアップコンテナ向け Slack チャンネル"](#)

プレビューリリースでは、システムがクラウドに接続され、NetApp Active IQ ツールと AutoSupport ツールに統合されている必要があります。

を参照してください ["Astra データストアサポート業務"](#)。

- サポートケースを作成する方法、または簡単な質問を明確にする方法を教えてください。 \* サポートケースを作成する方法、または簡単な質問について説明する方法については、問題またはの質問を参照してください ["ネットアップコンテナ向け Slack チャンネル"](#)。ネットアップサポートがご連絡し、ベストエフォートベースでサポートを提供します。
- 新機能のリクエストをどのようにして提出しますか？ \* サポートされている構成や機能について質問がある場合は、[astra.feedback@netapp.com](mailto:astra.feedback@netapp.com) までお問い合わせください。
- サポートログバンドルの生成方法については、を参照してください ["サポートバンドルの生成"](#) Astra Data Store プレビュー版のサポートログバンドルをセットアップおよびダウンロードする手順については、こちらを参照してください。

- Astra データストアプレビューで Kubernetes ノードが見つかりません。どうすれば修正できますか？ \* を参照してください "[Astra データストアプレビューをインストールします](#)"。
- IPv6 アドレスは管理ネットワーク、データネットワーク、クラスタネットワークに使用できますか？ \* いいえ、Astra データストアプレビューでサポートされているのは IPv4 アドレスのみです。IPv6 のサポートは、Astra データストアプレビューの今後のリリースで追加される予定です。
- Astra Data Store プレビューでボリュームをプロビジョニングする際に使用される NFS のバージョンは何ですか？ \* デフォルトでは、Kubernetes アプリケーション用にプロビジョニングされたすべてのボリュームに対して、Astra Data Store プレビューで NFS v4.1 がサポートされています。
- 大容量ドライブで Astra データストアプレビューを構成しても、大容量の永続ボリュームを取得できないのはなぜですか？ \* Astra データストアプレビューにより、Astra データセンターのすべてのノードでプロビジョニングされる最大容量が 1TiB に、すべてのノードで最大 5TiB に制限されます クラスタのプレビューを保存します。

を参照してください "[Astra データストアのプレビュー要件](#)" および。

## Astra データストアプレビューのアップグレード

- Astra Data Store プレビューリリースからアップグレードできますか。 \* いいえAstra データストアプレビューは本番環境のワークロードには適用されず、Astra データストアプレビューソフトウェアの新しいリリースには新規インストールが必要になります。

## Copyright Information

Copyright © 2022 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means-graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system-without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

## Trademark Information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.