



はじめに Astra Data Store

NetApp
June 17, 2022

目次

はじめに	1
Astra データストアの要件	1
Astra データストアのクイックスタート	6
Astra データストアをインストール	7
Astra データストアのコンポーネントをセットアップする	25
Astra Data Store Early Access Program (EAP ; 早期アクセスプログラム) の制限	36
Astra データストアに関するFAQ	37

はじめに

Astra データストアの要件

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

Astraデータストアは、ベアメタル環境とVMベース環境の両方をサポートしています。Astraデータストアクラスタは、4つ以上のワーカーノードで構成されるKubernetesクラスタで実行できます。Astra Data Store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。

- [\[Kubernetes worker node resources\]](#)
- [\[Hardware and software\]](#)
- [\[Networking\]](#)
- [Astra Trident](#)
- [\[Container Network Interfaces\]](#)
- [\[Licensing\]](#)



Astra Control CenterからAstraデータストアクラスタを管理する場合は、Astraデータストア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

Kubernetesクラスタ内の各ワーカーノード上のAstra Data Storeソフトウェアに割り当てる必要があるリソース要件は次のとおりです。

リソース	最小（ Minimum ）	最大
Astraデータストアクラスタのノード数	4.	16
データドライブの数	<ul style="list-style-type: none">• 3（独立したキャッシュデバイスを使用）• 4（キャッシュデバイスがない場合）	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サイズ）	該当なし

Astraデータストアでは、Kubernetesワーカーノードごとに次のvCPUとRAMの組み合わせをサポートしています。

- vCPU×9、38GiBのRAM
- vCPU×23、9GiBのRAM



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データストアのログファイルを格納できます。
- 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で使用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：

Astra Data Store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 クラスターの構成が異なる場合。

ハードウェアプラットフォーム

- Dell R640
- Dell R740
- HPE DL360
- HPE DL380
- Lenovo SR630
- Lenovo SR650



VMベースの環境では、に互換性があると表示されているサーバを使用することもできます ["VMware Compatibility Guide"』を参照してください](#)。

ストレージ

Astraデータストアは、SATA、SAS、NVMe TLC SSDで検証済みです。

VMベースの環境では、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を使用できます。



- 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データストアは、vSphere 7.0を使用したVMwareベースのVM環境で検証済みです。KVMベースの導入はAstraデータストアではサポートされていません。
- Astraデータストアは、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データストアは、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 4.6～4.9
 - Google Anthos 1.8～1.10
 - Kubernetes 1.20～1.23
 - Rancher RKE 1.3.3



Astraデータストアには、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を行うAstra Tridentが必要Tridentバージョン21.10.1～22.04がサポートされています。を参照してください
"[Astra Trident のインストール手順](#)"。

ネットワーキング

AstraデータストアのMVIP用にクラスタごとに1つのIPアドレスが必要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データストア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"> • ノードごとに 2 つ： <ul style="list-style-type: none"> ◦ 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 ◦ 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

設定	IP アドレスが必要です
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"> ノードあたり 3 本： <ul style="list-style-type: none"> 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 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 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



これらの構成では VLAN タグは使用されません。

ファイアウォールに関する考慮事項

ネットワークファイアウォールトラフィックフィルタリングを実行する環境では、次の表に示すポートおよびプロトコルへの着信トラフィックを許可するようにファイアウォールを設定する必要があります。[IP Address]列には、次の省略形が使用されます。

- mip : 各ノードの管理インターフェイスのプライマリIPアドレス
- cip : 各ノードのクラスタインターフェイスのプライマリIPアドレス
- DIP : ノードに設定された1つ以上のデータIPアドレス
- MVIP : 1つのクラスタノードに設定された管理仮想IPアドレス

ポート/プロトコル	IP アドレス	目的	注：
111 TCP	ディップ	NFS	データIPは実行時にクラスタノード間で移動されます。各ノードで、すべてのデータIP（またはサブネット全体）に対してこのトラフィックを許可する必要があります。
442/TCP	MIP	API	
635 TCP	ディップ	NFS	データIPは実行時にクラスタノード間で移動されます。各ノードで、すべてのデータIP（またはサブネット全体）に対してこのトラフィックを許可する必要があります。

ポート/プロトコル	IP アドレス	目的	注：
2010/UDP	CIP (CIP)	クラスタ/ノード検出	UDPポート2010との間で送受信されるユニキャストトラフィックとブロードキャストトラフィックの両方が含まれます（応答も含まれます）。
2049 TCP	ディップ	NFS	データIPは実行時にクラスタノード間で移動されます。各ノードで、すべてのデータIP（またはサブネット全体）に対してこのトラフィックを許可する必要があります。
2181-2183/TCP	CIP (CIP)	分散通信	
2443/TCPの場合	MIP	API	
2443/TCPの場合	MVIPの数	API	MVIPアドレスは任意のクラスタノードでホストでき、必要に応じて実行時に再配置されます。
4000-4006 /TCP	CIP (CIP)	クラスタ内RPC	
4045-4046/TCP	ディップ	NFS	データIPは実行時にクラスタノード間で移動されます。各ノードで、すべてのデータIP（またはサブネット全体）に対してこのトラフィックを許可する必要があります。
7700/TCP	CIP (CIP)	セッションマネージャ	
9919/TCP	MIP	DMS API	
9920/TCP	ディップ	DMS RESTサーバ	
ICMP	CIP + DIP	ノード内通信、ヘルスチェック	データIPは実行時にクラスタノード間で移動されます。各ノードで、すべてのデータIP（またはサブネット全体）に対してこのトラフィックを許可する必要があります。

Astra Trident

Astraデータストアを利用するには、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を行うために、アプリケーションのKubernetesクラスタがAstra Tridentを実行する必要があります。Tridentバージョン21.10.1～22.04がサポートされています。Astraデータストアは、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

コンテナネットワークインターフェイス

Astraデータストアは、次のContainer Network Interface（CNI;コンテナネットワークインターフェイス）で検証済みです。

- RKEクラスタの場合はCalico
- 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



- Cilium CNIを使用して導入されたAstraデータストアでは、HostPortサポート用のportmapプラグインが必要です。CNIチェイニングモードをイネーブルにするには、cilium-config configMapに「ctie-mode:portmap」を追加し、Ciliumポッドを再起動します。
- ファイアウォール対応の構成は、CalicoおよびOpenShiftのSDN NNIでのみサポートされません。

ライセンス

Astraデータストアのすべての機能を有効にするには、有効な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。

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データストアの制限"](#)

Astra データストアのクイックスタート

このページでは、Astraデータストアの導入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。

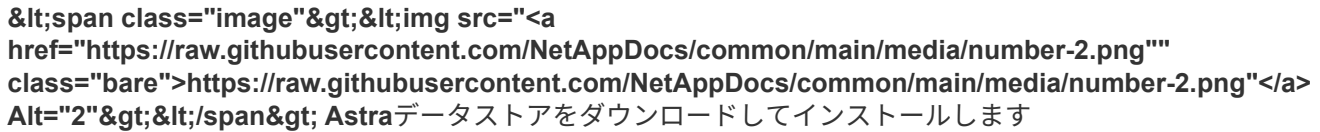
ぜひお試しください。Astraデータストアを試用する場合は、90日間の評価版ライセンスを使用できます。 ["こちらから登録してください"](#) Astraデータストアライセンスを取得する方法

 <https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-1.png>
 <https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-1.png>
Alt="One"  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データストア環境の一部であるKubernetesワーカーノードには、いずれも同じインターフェイスタイプ（SATA、SAS、NVMe）のSSDと、Astraデータストアクラスタに割り当てられているドライブ数が必要です。


- 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 データストアの要件"](#)。

 <https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-2.png>
Alt="2" [Astraデータストア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データストアをローカル環境にインストール
- Astraデータストアライセンスを適用します。
- Astraデータストアクラスタをインストール

の詳細を確認してください ["Astraデータストアのインストール"](#)。

 <https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-3.png>
Alt="3" [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データストアをストレージバックエンドとしてセットアップする。
- Default Astra Data Storeストレージクラスを作成
- 遠隔測定サービス用のAstraデータストアを構成する。

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データストアのセットアップが完了したら、次のことを行うことができます。

- 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"](#)。
- ["VMwareでAstraデータストアを使用"](#)。

["Astra データストアをインストール"](#)。

Astra データストアをインストール

Kubernetesネイティブコマンドを使用するか、Astra Control CenterのUIを使用して、Astraデータストアをインストールできます。

インストールオプション

- * Kubernetesネイティブコマンド*：Kubernetesネイティブコマンドを使用してAstraデータストアをインストールするには、で説明するインストール手順を実行します [この手順](#)。

- * Astra Control Centerを使用している場合*：Astra Control Centerを使用してAstraデータストアをインストールするには、に記載されているインストール手順を実行します [この手順](#)。

必要なもの

- "インストールを開始する前に、Astra Data Store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"。 "登録" 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がない場合は、評価版をダウンロードしてください。
- A "ネットアップライセンスファイル（NLF）" Astraデータストア向け。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データストアで使用。

Astra データストアをインストール

標準のKubernetesクラスタを使用するAstraデータストアのインストールプロセスでは、以下の手順を大まかに実行できます。また、Red Hat OpenShift Container Platform（OCP）環境用の追加のインストール手順についても説明します。

- [\[Download the Astra Data Store bundle and extract the images\]](#)
- [\[Copy the binary and push images to your local registry\]](#)
- [\[OpenShift procedure\]](#)
- [\[Install the Astra Data Store operator\]](#)
- [\[Deploy the Astra Data Store version YAML\]](#)
- [\[Apply the Astra Data Store license\]](#)
- [\[Install the Astra Data Store cluster\]](#)
- [\[Understand deployment-related events\]](#)



Google AnthosでAstraデータストアを使用するには、以下のインストール手順を実行し、AstraデータストアクラスタをAnthos環境に追加します。



Rancher RKE環境でAstra Data Storeをインストールするには、次のインストール手順を実行し、kubectlコマンドのrancherコマンドを置き換えます。

Astra Data Store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 Data Storeバンドルをダウンロードします('Astra_Data-Store_22.05.tar')



以前のバージョンのバンドルに関する手順については、を参照してください ["該当バージョンのドキュメントを参照してください"](#)。

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 Astra_Data_Store_2022.05.pub -signature  
Astra_Data_Store_2022.05.sig 2022.12.01_ads.tar
```

3. ディレクトリを作成します。

```
mkdir Astra_Data_Store_2022.05  
cd Astra_Data_Store_2022.05
```

4. 画像を抽出します。

```
tar -xvf <path to tar file>/Astra_Data_Store_2022.05.tar
```



画像は、作業ディレクトリ内に作成された「astrs/images」ディレクトリに抽出されます。

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. k8s kubectlバイナリがインストールされている標準パスに'イメージを抽出するために使用したディレクトリからkubectl-astrasバイナリをコピーします(以下の例では'/usr/bin/'がパスとして使用されます)kubectl-astrasは、Astraデータストアクラスター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。



kubectlバイナリがインストールされているパスを検索するには'which kubectl'コマンドを使用します

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

2. Astra Data Store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データストアイメージをプッシュするレジストリパス（例：repo.company.com`）に設定します。

```
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 $(ls astrads/images/*.tar) ; do
    astraImage=$(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}'~'
./astrads/manifests/*.yaml
```

OpenShift 手順 の略

以下の手順 は、Red Hat OpenShift Container Platform（OCP）への導入にのみ使用してください。OCP以外のKubernetesクラスタへの導入については、この手順 をスキップしてください。

- Astraデータストアを導入するためのネームスペースを作成
- カスタム SCC を作成します
- ロールとロールのバインドを作成します

例 1. 詳細

すべてのAstraデータストアコンポーネントをインストールする名前空間「astrs-system」を作成します。

次の手順は、Red Hat OpenShift Container Platform（OCP）に導入する場合にのみ必要です。

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
```

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 クラスタポッドに必要な権限は提供されません。この手順を使用して、Astra データストアに対して必要な権限（サンプルに記載）を付与します。

カスタム SCC を Astra Data Store 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。

以下の手順は、Red Hat OpenShift Container Platform（OCP）に導入する場合にのみ必要です。

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_scc.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          ["*"]
#
```

Astraデータストア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定義では'Astraデータストアのリソースがastrads.netapp.io` APIグループに必要とするさまざまな役割を割り当てます(役割のバインドを使用)

以下の手順は、Red Hat OpenShift Container Platform（ OCP ）に導入する場合にのみ必要です。

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

```

プライベートイメージレジストリを設定します

環境によっては、シークレットを使用するプライベートレジストリからイメージをプルするように設定を変更することもできます。

1. 前の手順に進んでいない限り 'astrs-system' 名前空間を作成します

```
kubectl create namespace astrads-system
```

2. シークレットを作成します。


```
kubectl create secret docker-registry <secret-name> -n astrads-system
--docker-server=<registry name> --docker-username= <registry username>
--docker-password=<registry user password>
```

3. サービスアカウントにシークレット設定情報を追加します。

```
kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name":
"<secret-name>"}]}' -n astrads-system
```



これらの変更は、のときに適用されます [Astra Data Storeオペレータをインストール](#)。

Astra Data Storeオペレータをインストール

1. Astraデータストアマニフェストの一覧を表示します。

```
ls astrads/manifests/*.yaml
```

対応：

```
astrads/manifests/monitoring_operator.yaml
astrads/manifests/astradscluster.yaml
astrads/manifests/astradsversion.yaml
astrads/manifests/astradsoperator.yaml
astrads/manifests/vasa_asup_certs.yaml
astrads/manifests/manifest.yaml
astrads/manifests/configuration.yaml
```

2. 「kubectl apply」を使用してオペレータを配備します。

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

対応：



ネームスペースの応答は、標準インストールとのどちらを実行したかによって異なる場合があります ["OpenShift Container Platformのインストール"](#)。

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsaddrives.astrads.n
etapp.io 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/astradsexportpolicies.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradskeyproviders.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/astradsnodemANagements.ast
rads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsqospolicies.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradssearkeyrotatereques
ts.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsversions.astrads.ne
tapp.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-astrads-system-admin-role created
role.rbac.authorization.k8s.io/astrads-astrads-system-reader-role
created
role.rbac.authorization.k8s.io/astrads-astrads-system-writer-role
created
role.rbac.authorization.k8s.io/astrads-leader-election-role created
role.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-admin-clusterrole
created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-reader-clusterrole
created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-writer-clusterrole
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsautosupport-editor-
role created
```

```
clusterrole.rbac.authorization.k8s.io/astrads-astradsautosupport-viewer-  
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-astradsexportpolicy-  
editor-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsexportpolicy-  
viewer-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsfaileddrive-editor-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsfaileddrive-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-astradsnfsoption-editor-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnfsoption-viewer-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodeinfo-editor-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodeinfo-viewer-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodemanagement-  
editor-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodemanagement-  
viewer-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsqospolicy-viewer-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsversion-editor-role  
created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsversion-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-astradsvolumeeditor-  
role created
```

```

clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumeviewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumesnapshot-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumesnapshot-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-admin-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-reader-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-writer-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-leader-election-
rolebinding created
rolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-admin-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-reader-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-writer-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
configmap/astrads-autosupport-cm created
configmap/astrads-firetap-cm created
configmap/astrads-kevents-asup created
configmap/astrads-metrics-cm created
secret/astrads-autosupport-certs created
secret/astrads-webhook-server-cert created
service/astrads-webhook-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バージョンYAMLを導入します

1. kubectl applyを使用した配備:

```
kubectl apply -f ./astrads/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-2jqnk	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-dbk7v	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-rn9tt	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-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データストアライセンスを適用します

1. ネットアップから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	
e100000006-ads-capacity	astrads-example-cluster	true	Astra Data Store
true	2023-01-23	2022-04-04T14:38:54Z	

Astraデータストアクラスタをインストール

1. YAML ファイルを開きます。

```
vim ./astrads/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
 - ライセンスとAstra Data Storeのインストール・サイズに応じて'adsNodeConfig'の値をインストールに必要な値に変更します
 - 小：CPU 9個とメモリ38個
 - 中：CPU 23基、メモリ94基
 - (オプション) 「adsNodeSelector」セクションに関するコメントを削除します。選択したワーカーノードのプールにのみインストールするようにAstraデータストアを制限する場合は、この設定を行います。
 - (オプション) Astra Data Storeクラスタで使用するノードの数を4-16の範囲で指定します。
 - 「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データストアノードと同じ数以上にする必要があります。Astraデータストアの場合、この方法は、あとでクラスタを拡張する場合は少なくとも4つのアドレス、または最大16個を意味します。
 - 「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。「adsNetworkInterfaces」セクションのコメントも削除してください。



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データストア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。



簡潔にするために、以下のYAMLサンプルからコメントが削除されています。

```
apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
    cpu: 9
    memory: 38
    # [Optional] Specify node selector labels to select the nodes for
    # creating ADS cluster
    # adsNodeSelector:
    #   matchLabels:
    #     customLabelKey: customLabelValue
    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 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 ./astrads/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 7h31m	1/1	Running	0
astrads-deployment-support-788b859c65-2qjkn 12d	3/3	Running	19
astrads-ds-astrads-cluster-1ab0dbc-j9jzc 5d2h	1/1	Running	0
astrads-ds-astrads-cluster-1ab0dbc-k9wp8 5d1h	1/1	Running	0
astrads-ds-astrads-cluster-1ab0dbc-pwk42 5d2h	1/1	Running	0
astrads-ds-astrads-cluster-1ab0dbc-qhvc6 8h	1/1	Running	0
astrads-ds-nodeinfo-gcmj8 12d	1/1	Running	1
astrads-ds-nodeinfo-j826x 12d	1/1	Running	3
astrads-ds-nodeinfo-vdthh 12d	1/1	Running	3
astrads-ds-nodeinfo-xwgsf 12d	1/1	Running	0
astrads-ds-support-828vw 5d2h	2/2	Running	2
astrads-ds-support-astrads-example-cluster-cfzts 8h	2/2	Running	0
astrads-ds-support-astrads-example-cluster-nzkkr 7h49m	2/2	Running	15
astrads-ds-support-astrads-example-cluster-xxbnp 5d2h	2/2	Running	1
astrads-license-controller-86c69f76bb-s6fb7 8h	1/1	Running	0
astrads-operator-79ff8fbb6d-vpz9m 8h	1/1	Running	0

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	2022.05.0-X	e100000006	
10.x.x.x	13m			

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	メッセージと重要性
ControlPlaneNoDesSelected」を選択します	ADSクラスタに参加するための[number]コントロールプレーンノードが正常に選択されました。Astra Data Storeオペレータは、Astraデータストア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。
ADSClusterCreateInProgress	Astraデータストア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。
ADSClusterCreateSuccess	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「in progress」のままになっている場合は、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 Control Centerを使用してAstraデータストアをインストールします

Astra Control Centerを使用してAstraデータストアを導入および使用するには、次の手順を実行します。

必要なもの

- 確認が完了しました [一般的なAstraデータストアの前提条件](#)。
- Astra Control Centerをインストールしておきます。

手順

1. ["Astra Control Centerを使用してAstraデータストアを導入"](#)。

次の手順

- * Kubernetesネイティブな導入環境とサードパーティのディストリビューション*：Astra Data Storeの導入を完了するには、追加の作業を実行します ["セットアップのタスク"](#)。
- * Astra Control Center *：Astra Control Centerを使用してAstraデータストアを導入したことがある場合、これらの手順に従う必要はありません ["セットアップのタスク"](#) その他の監視オプションを設定する場合を除きます。Astraデータストアの導入後、Astra Control Center UIを使用して次の作業を実行できます。
 - ["Astraデータストア資産の健全性を監視"](#)。
 - ["Astraデータストアのバックエンドストレージを管理"](#)。
 - ["ノード、ディスク、永続的ボリューム要求（PVC）を監視"](#)。

Astraデータストアのコンポーネントをセットアップする

お先にどうぞ ["スタンドアロンのAstraデータストアをインストール"](#) とで、いくつかの対処方法を ["環境に関する前提条件"](#) 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。



Astra Control Centerを使用してAstraデータストアを導入している場合は、セットアップする予定がない限り、これらの手順に従う必要はありません [その他の監視オプション](#)。

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

Astra Trident をインストール

Astraデータストアの場合、Astra Trident 21.10.1以降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 Data Store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.k8es.io_volumesnapshotclasses.yaml
 - snapshot.storage.k8es.io_volumesnapshotcontentes.yaml
 - snapshot.storage.k8es.io_volumesnapshots.yaml

手順

1. snapshot.storage.k8es.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 configured
```

2. snapshot.storage.k8es.io_volumesnapshotcontentes.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 configured
```

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 configured
```

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 configured
```

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

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。



JSONの例を次に示します。

- 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": "silver"
  },
  "storage": [
    {
      "labels": {
        "performance": "extreme"
      },
      "defaults": {
        "qosPolicy": "gold"
      }
    },
    {
      "labels": {
        "performance": "premium"
      },
      "defaults": {
        "qosPolicy": "silver"
      }
    },
    {
      "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
```

例

```

allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  creationTimestamp: "2022-05-09T18:05:21Z"
  name: ads-silver
  resourceVersion: "3361772"
  uid: 1o023456-da4b-51e3-b430-3aa1e3bg111a
mountOptions:
- vers=4
parameters:
  backendType: astrads-nas
  selector: performance=premium
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate

```

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
```

対応：

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE
ads-silver	csi.trident.netapp.io	Delete	Immediate
allowVolumeExpansion	age		
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: gold
        size: ""
        snapshotDir: ""
        snapshotPolicy: ""
      labels:
        performance: extreme
      region: ""
```

```

supportedTopologies: null
zone: ""
- defaults:
  exportPolicy: ""
  qosPolicy: silver
  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:
  - ads-silver
supportedTopologies: null
volumes: []
```

Astraデータストアの監視を設定

（オプション）別のテレメトリサービスによる監視用にAstraデータストアを設定できます。この手順は、Astraデータストア監視用のAstra Control Centerを使用していない場合や、監視を追加のエンドポイントに拡張する場合にお勧めします。

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。



Astra Control Centerの導入では、モニタリングオペレータが自動的に設定されます。次の手順の最初の2つのコマンドはスキップできます。

監視を設定する前に'astras-system'名前空間にアクティブなAstraデータストアクラスタが必要です

手順

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データストアを監視用に設定します。

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

3. ElasticエンドポイントにEMSログをストリーミングするようにAstraデータストアを設定します。

```
kubectl astrads monitoring es --port <portname> --host <hostname>
```

Astra Data Store Early Access Program（EAP；早期アクセスプログラム）の制限

Astraデータストアには、Early Access Programにおける次のリソース制限があります。

リソース	最小（ Minimum ）	最大
Astraデータストアクラスタのノード数	4.	16

リソース	最小（ Minimum ）	最大
ノードあたりの永続ボリュームの数	該当なし	50
ボリュームサイズ	20MiB	2TiB
ボリュームあたりの Snapshot 数	0	1、023
ボリュームあたりのクローン数	0	9.
ノードあたりのVM数	0	50

Astraデータストアに関するFAQ

Astra Data Store Early Access Programリリースのインストール、設定、アップグレード、および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

*製造にはAstra Data Store Early Access Programリリースを使用できますか？*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データストアのEarly Access Programリリースは本番環境のワークロードをターゲットとしたものではありません。

*仮想マシンのワークロードにAstraデータストアを使用できますか？*はい。Astraデータストアは、KubernetesとVMwareのVVOLワークロードの両方をサポートします。

を参照してください ["VMwareを使用したAstraデータストアの詳細をご確認ください"](#)。

- VMware vSphereを使用してAstraデータストアを管理できますか？*はい、ネットアップのAstraプラグインを使用して、ネットアップのAstraデータストアをvCenterでネイティブに管理できます。を参照してください ["VMware環境のAstra Data Storeコンポーネントを管理します"](#)。
- Astraデータストアには、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では、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"](#)。

VMwareワークフローとNetApp Astra Plugin for VMware vSphereを有効にするには、Astra Control Centerが必要です。

Astraデータストア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は、を使用できます ["Astra Control Center の略"](#) 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。

- Astra Data Storeクラスタの管理方法* kubectlコマンドを使用し、Kubernetes API拡張機能を使用してAstraデータストアのリソースを管理できます。

'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

- Astraデータストアクラスタの指標を監視するにはどうすればよいですか？* Cloud Insights を使用して、Astraデータストアの指標を監視できます（を参照） ["Cloud Insights で指標を監視"](#) またはKubernetesのネイティブ監視（を参照 ["PrometheusとGrafanaで監視します"](#)）。

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クラスタでONTAP やその他のストレージプロバイダとともにAstraデータストアを使用できますか？*はい。Astraデータストアは、アプリケーションクラスタ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データストアからKubernetesクラスタを削除してもアンインストールされますか？* Astra Tridentは、Astraデータストアをアンインストールしてもクラスタ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データストアにはライセンスが必要ですか？*はい、Astraデータストアには、Early Access Program（EAP；早期アクセスプログラム）用の評価用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データストア評価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クラスタへのAstraデータストアのインストールと使用

*ベアメタルまたは仮想マシンで稼働するKubernetesクラスタにAstraデータストアをインストールできますか？*はい。Astraデータストアは、ベアメタルまたはvSphere VMで実行されているKubernetesクラスタにインストールできます。を参照してください ["Astra データストアの要件"](#)。

- Astra Data StoreでサポートされているKubernetesのバージョンは何ですか。*

Astraデータストアは、v1.20以降と互換性のあるKubernetesディストリビューションで動作します。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 データストアの要件"](#)。

- My Kubernetesクラスタは4つ以上のワーカーノードで構成されています。AstraデータストアをITにインストールできますか。*はい。Astraデータストアクラスタは、最初にKubernetesクラスタ内の4つのワーカーノードに導入する必要があります。導入後、クラスタを最大16個のワーカーノードまで拡張できます。
- Astraデータストア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 データストアをインストール"](#)。

*アストラデータストアを使用するにはインターネット接続が必要ですか？*いいえ、アストラデータストア早期アクセスプログラムではインターネット接続は必要ありません。ただし、テレメトリバンドルを定期的送信するために、NetApp AutoSupport バックエンドに接続することを推奨します。直接接続かプロキシ経由で接続できます。

- Astra Data Storeで使用する役割と権限は何ですか。* Astra Data Store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「astrs-ds-nodeinfo」という特権的なデーモンがあります。

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ストレージクラスタを構築します。

- Astra Data Store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を教えてください。*からダウンロードしたAstra Data Store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 'astradscluster.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 データストアをインストール"](#)。

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チャネルを使用して、ネットアップのAstraデータストアからコミュニティサポートにアクセスできます。このチャネルの監視は、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 チャネル"](#)

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 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データストア用のサポートログバンドルのセットアップとダウンロードの手順については、を参照してください。
- AstraデータストアがKubernetesノードを見つけられない。どうすれば修正できますか？* を参照してください ["Astra データストアをインストール"](#)。
- IPv6アドレスを管理、データ、およびクラスタネットワークに使用できますか？* いいえ、AstraデータストアはIPv4アドレスのみをサポートしています。IPv6のサポートは、Astraデータストアの今後のリリースで追加される予定です。
- Astra Data Store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？* Astra Data Store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NFS v4.1、VMwareワークロード用にプロビジョニングされたすべてのボリュームにNFSv3バージョン4.1がサポートされています。

を参照してください ["Astra データストアの要件"](#) および ["Astraデータストアの制限"](#)。

Astraデータストアのアップグレード

- 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*はい。Astra Data Store Early Access Programリリースから将来のリリースにアップグレードできます。

著作権情報

Copyright © 2022 NetApp, Inc. All rights reserved. 米国で印刷されていますこのドキュメントは著作権によって保護されています。画像媒体、電子媒体、および写真複写、記録媒体などの機械媒体など、いかなる形式および方法による複製も禁止します。テープ媒体、または電子検索システムへの保管-著作権所有者の書面による事前承諾なし。

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

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、いかなる場合でも、間接的、偶発的、特別、懲罰的、またはまたは結果的損害（代替品または代替サービスの調達、使用の損失、データ、利益、またはこれらに限定されないものを含みますが、これらに限定されません。）ただし、契約、厳格責任、または本ソフトウェアの使用に起因する不法行為（過失やその他を含む）のいずれであっても、かかる損害の可能性について知らされていた場合でも、責任の理論に基づいて発生します。

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

このマニュアルに記載されている製品は、1 つ以上の米国特許、その他の国の特許、および出願中の特許により特許、その他の国の特許、および出願中の特許。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7103（1988 年 10 月）および FAR 52-227-19（1987 年 6 月）の Rights in Technical Data and Computer Software（技術データおよびコンピュータソフトウェアに関する諸権利）条項の（c）（1）（ii）項、に規定された制限が適用されます。

商標情報

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