



Astra データストアのドキュメント

Astra Data Store

NetApp
June 01, 2022

目次

Astra データストアのドキュメント	1
リリースノート	2
このリリースの Astra データストアの新機能	2
既知の問題	3
既知の制限	4
概念	5
Astra データストアの詳細	5
Astra データストア導入モデル	6
クラスタの拡張	7
Astra データストアのストレージ効率	8
Astra データストアのセキュリティ	9
はじめに	10
Astra データストアの要件	10
Astra データストアのクイックスタート	14
Astra データストアをインストール	15
Astra データストアのコンポーネントをセットアップする	32
Astra Data Store Early Access Program (EAP；早期アクセスプログラム) の制限	43
Astra データストアに関するFAQ	44
Astra データストアを使用	47
kubectl コマンドを使用して Astra データストアのリソースを管理	47
テストアプリケーションを展開します	53
Astra データストアクラスタを管理	57
Astra データストアを監視	67
Secure Astra データストア	82
Astra データストアライセンスを更新	87
Astra データストアをアップグレード	88
Astra データストアを自動化されたスクリプトでアンインストールします	90
VMware で Astra データストアを使用	105
VMware を使用した Astra データストアの詳細をご確認ください	105
VMware の要件を満たす Astra データストア	105
VMware を使用して Astra データストアをセットアップする	106
VMware 環境のコンポーネントを監視する	114
VMware 環境の Astra Data Store コンポーネントを管理します	115
VMware 統合環境から Astra データストアをアンインストールします	119
知識とサポート	120
トラブルシューティング	120
ヘルプを表示します	120
サポートの自動監視	121
旧バージョンの Astra データストア	126

法的通知	127
著作権	127
商標	127
特許	127
プライバシーポリシー	127
オープンソース	127

Astra データストアのドキュメント

リリースノート

この度、アストラデータストアの2022.05 Early Access Program (EAP)リリースを発表いたしました。

- ["このリリースの Astra データストアの新機能"](#)
- ["既知の問題"](#)
- ["既知の制限"](#)

Twitter で @NetAppDoc をフォローしてください。を作成し、ドキュメントに関するフィードバックを送信します ["GitHub の貢献者"](#) または、 doccomments@netapp.com に電子メールを送信します。

このリリースの **Astra** データストアの新機能

この度、アストラデータストアの2022.05 Early Access Program (EAP)リリースを発表いたしました。

2022年5月31日（2022.05 EAP Release）

Astra Data Store 2022.05リリースには次の機能が含まれています。

- VMware環境の統合：
 - NFS VVOLデータストアのワークフローとストレージポリシーベースの管理ストレージのプロビジョニング
 - vCenterからのネイティブのストレージ管理と監視に使用するAstra Plugin for VMware vSphere
- を含む、クラスタ管理の強化 ["Astra Control CenterからのGUIベースのクラスタの導入"](#)。
- のサポート ["クラスタを拡張"](#)（ノード数の増加、CPUと容量の増加）
- セキュリティの機能拡張（キーと証明書、および外部キー管理（KMIP）のサポート）

2022 年 2 月 8 日（2021.12.1）

Astra Data Store プレビュー (21.12) 用のパッチリリース (2021.12.1)。

- 今回のリリースでは、Astra データストアのプレビューで、VXLAN 構成を Calico CNI でサポートしています。
- Calico WireGuard 対応のネットワーク構成が Astra データストアプレビューでサポートされるようになりました。
- 更新された AstraDSCluster.YAML 「adsNetworkInterfaces」セクションには、わかりやすいコメントが含まれています。
- kubectl コマンドアンインストールプロセスの代わりに、新しい Astra データストアアンインストールスクリプトが提供されるようになりました。

2021 年 12 月 21 日（21.12）

Astra データストアプレビューの初回リリース。

- ["それは何であるか"](#)
- ["導入モデルとコンポーネント"](#)
- ["開始には何が必要ですか"](#)
- ["をインストールします" および "セットアップ \(Setup \) "](#)
- ["管理" および "モニタ" パフォーマンス](#)
- ["Cloud Insights で監視"](#)
- ["ヘルプを表示します" および "自動サポート監視を使用"](#)

詳細については、こちらをご覧ください

- ["既知の問題"](#)
- ["既知の制限"](#)
- ["ネットアップの技術情報アーティクル"](#)

既知の問題

既知の問題は、このリリースの製品を正常に使用できない可能性のある問題を特定します。

Astra Data Store 2202.05 Early Access Program (EAP) リリース

このリリースには、これらの既知の問題が含まれていてこれらの既知の問題をよく確認してください。

ストレージバックエンドのタスクがvSphereに表示されません

既存のストレージバックエンドを使用するアクションやストレージバックエンドを削除するアクションが送信された場合、そのタスクはvSphere Clientの最近のタスクパネルに表示されません。

VMの作成時に追加のフォルダが作成されます

vSphereでVMを作成すると、データストア内のVMごとに1つのフォルダではなく、追加のフォルダが作成されることがあります。

Astra Plugin for VMware vSphereでフィルタリングされたビューに、断続的にデータが表示されないことがあります

仮想マシンのテーブル列フィルタでは、フィルタリングされたデータが除外されることがあります。フィルタ処理されたビューには、代わりにロードアイコンが表示されます。この問題を解決するには、別のビューに移動し、フィルタをリセットしてデータをもう一度表示します。

Kubernetes ワーカーノードのIPアドレスの制限

次のシナリオで、Astraデータストアクラスタの作成が失敗する。

1. Kubernetes ワーカーノードには、Astraデータストアに割り当てられていないインターフェイスでホストされているIPアドレスがあります。
2. このIPアドレスは、Astraデータストアクラスタに割り当てられたクラスタ/データIPアドレスと同じサブ

ネット内にあります。

詳細については、こちらをご覧ください

- ["既知の制限"](#)

既知の制限

既知の制限事項は、製品のこのEarly Access Program（EAP；早期アクセスプログラム）リリースでサポートされていないプラットフォーム、デバイス、または機能、またはそれと正常に相互運用できない機能を示しています。

Astra Data Store 2022.05 Early Access Program (EAP)リリース

このリリースには、次の既知の制限事項があります。これらの既知の制限事項をよく確認してください。

VMwareとの統合に関する制限事項

次の機能はサポートされません。

- VMワークフロー：
 - テンプレートからのクローン作成を含めて、VMをクローニングします
 - 以前にVMディスクを取り外したあとでVMディスクを接続する
- VVolレプリケーション
- VVolデータストアを削除します
- ファーストクラスディスク（FC）
- VMware vMotion
- リンクモードのvCenter Server
- コンプライアンスのアラートと通知
- vCenterとVASA Providerが複数ある構成
 - vCenter構成ごとに1つのVASA Providerを使用できます
- ストレージポリシーベースの管理はVMレベルでのみ実行でき、VMの作成後は変更できません。

複数の**VM**ディスクを同時にサイズ変更することはできません

このリリースでは、一度に1つのディスクのサイズ変更のみが可能です。一度に複数のディスクのサイズを変更しようとする、サイズ変更処理は最初のディスクに対してのみ実行されます。

詳細については、こちらをご覧ください

- ["既知の問題"](#)

概念

Astra データストアの詳細

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 データストアは、コンテナと VM の両方のワークロードに対して、ネットアップのエンタープライズデータ管理とネイティブの共有ファイルサービスを提供します。

Astra データストアでは、次のことが可能です。

- * Kubernetes のコンテナ化されたワークロード * をサポート：使用するエンタープライズデータ管理サービスとツールを利用できます。
- * DevOps 向けの Kubernetes 「アプリケーションサービス」プラットフォームを使用 *：柔軟性に優れたソフトウェア定義のセルフサービス型プラットフォームを構築して、自動化された反復可能なサービスを提供し、開発者が必要とする複雑さを解消します。

Astra データストアは、Astra 製品ファミリーの一部です。の詳細を確認してください ["アストラファミリー"](#)。

Astra データストアの機能

Astra データストアは、Kubernetes ネイティブのエンドツーエンドのストレージとデータ管理機能をクラウドネイティブアプリケーションに提供します。以下の機能が必要です。

- * Kubernetes ネイティブの共有ファイルサービス *：標準の NFS クライアントをコンテナと VM の統合データストアとして使用し、Kubernetes にネイティブな共有ファイルサービスを提供します。
.esk.esub.us
- * クラウド規模 *：Kubernetes ネイティブの複数の並列ファイルシステムを同じリソースプール上に提供し、クラウドレベルの拡張性と利用率を実現します。これにより、ストレージをクラスタとは別に管理する必要がなくなります。
- * API ファーストアプローチ *：クラスタとワークロードの管理が自動化された状態になるコードとしてインフラを提供
- * エンタープライズクラスの詳細なデータ管理 *：アプリケーション対応の自動データ保護とディザスタリカバリを提供：
 - * ネットアップのテクノロジー *：Snapshot、バックアップ、レプリケーション、クローニングにネットアップのデータ管理テクノロジーを活用することで、ユーザは Kubernetes でエンタープライズアプリケーションを構築、導入できます。とは
 - * 耐障害性 *：Kubernetes ネイティブのワークロードにレプリケーションテクノロジーとイレイジャーコーディングテクノロジーを使用して耐障害性を向上
 - * データ効率 *：インラインの重複排除機能と圧縮機能により、拡張に合わせてコストを抑制します。
- * 既存の環境に適合 *：マイクロサービスベースの従来のワークロードをサポートし、主要な Kubernetes ディストリビューションに対応し、ファイルストレージを提供し、任意のハードウェア・ベース・ワークロードで動作します。
- * NetApp Cloud Insights との統合 *：継続的な最適化処理を実行するためのオブザーバビリティ、分析、監視を提供します。とは

Astra データストアの利用を開始しましょう

最初に、["Astra データストアの要件をご確認ください"](#)。

次に、["始めましょう"](#)。

を参照してください。

- ["Astra ファミリーの紹介"](#)
- ["Astra Control Service のマニュアル"](#)
- ["Astra Control Center のドキュメント"](#)
- ["Astra Trident のドキュメント"](#)
- ["Astra Control API を使用"](#)
- ["Cloud Insights のドキュメント"](#)
- ["ONTAP のドキュメント"](#)

Astra データストア導入モデル

Astraデータストアは、Kubernetesと連携して導入およびオーケストレーションされたアプリケーションを使用して、ホスト上でストレージドライブを直接管理します。

次のいずれかの方法で、ベアメタルサーバまたは仮想サーバにAstraデータストアをインストールできます。

- スタンドアロンの専用 Kubernetes クラスタに導入して、別のクラスタ（スタンドアロンクラスタ）で実行されている Kubernetes アプリケーションに永続ボリュームを提供します。
- Kubernetes クラスタに導入し、同じノードプール（コンバージドクラスタ）で他のワークロードアプリケーションをホストすることもできます。
- Kubernetes クラスタに導入すると、他のワークロードアプリケーションも別のノードプール（分離型クラスタ）にホストできます。

["Astra Data Storeのハードウェア要件の詳細"](#)。

Astraデータストアは、Astra製品ファミリーの一部です。アストラファミリー全体の展望については、[を参照してください "Astra ファミリーの紹介"](#)。

Astra データストアエコシステム

Astraデータストアは次の機能を備えています。

- *** Astra Control Center ***：オンプレミス環境で Kubernetes クラスタをアプリケーション対応で管理するために、Astra Control Center ソフトウェアを使用。Kubernetes アプリケーションを簡単にバックアップし、データを別のクラスタに移行して、作業用アプリケーションのクローンを瞬時に作成できます。

Astra Control Centerは、ONTAP またはAstraデータストアストレージバックエンドを備えたAstra Trident ストレージバックエンドでKubernetesクラスタをサポートします。

- *** Trident ***：ネットアップが管理する、完全にサポートされているオープンソースのストレージプロビジ

ョニングおよびオーケストレーションツールである Astra Trident を使用すると、 Docker と Kubernetes で管理されるコンテナ化アプリケーション用のストレージボリュームを作成できます。

Astra Tridentを使用して、Astraデータストアにボリュームを作成する。

- *** Cloud Insights *** : ネットアップのクラウドインフラ監視ツールである Cloud Insights を使用すると、Astra Control で管理された Kubernetes クラスタのパフォーマンスと利用率を監視できます。Cloud Insights : ストレージ使用率とワークロードの相関関係を示します。

Cloud Insights 接続を Astra Control で有効にすると、Astra Control の UI ページにテレメトリ情報が表示されます。Cloud Insights は、Astraデータストアで管理されているリソースに関する情報を表示します。

Astra データストアインターフェイス

さまざまなインターフェイスを使用してタスクを完了できます。

- *** Web ユーザーインターフェイス (UI) *** : Astra Control Service と Astra Control Center は、同じ Web ベースの UI を使用して、アプリケーションの管理、移行、保護を行うことができます。また、Astraデータストアボリュームに関する情報も表示されます。
- *** API *** : Astra Control Service と Astra Control Center は、同じ Astra Control API を使用します。API を使用するタスクは、UI を使用するタスクと同じです。Astra Control APIを使用して、Astraデータストアに関する情報を取得することもできます。
- **kubectl** コマンド: Astraデータストアを操作するには'kubectl'コマンドを直接使用できます
- *** Kubernetes拡張機能*** : さらに、Astra Data Store Kubernetes API拡張機能を使用できます。

カスタムリソース定義 (CRD) はKubernetes REST APIの拡張機能で、Astra Data Storeオペレータが導入されたときに作成されます。外部エンティティは、Kubernetes API サーバを呼び出して SSD とやり取りします。Astraデータストアは、特定のSSDに対する更新を監視してから、内部のREST APIを呼び出します。

を参照してください。

- ["Astra ファミリーの紹介"](#)
- ["Astra Control Service のマニュアル"](#)
- ["Astra Control Center のドキュメント"](#)
- ["Astra Trident のドキュメント"](#)
- ["Astra Control API を使用"](#)
- ["Cloud Insights のドキュメント"](#)
- ["ONTAP のドキュメント"](#)

クラスタの拡張

Astraデータストアは、クラスタ内でさまざまなタイプや機能のノードをサポートします。クラスタを拡張する場合、Astraデータストアでは、パフォーマンス機能を持つノードの追加をサポートしています。ただし、クラスタ内のパフォーマンスが最も低いノードよりも少ないノードだけでは可能です。新しいノードのストレージ容量は既存のノードと同じである必要があります。拡張時の新しいノードを含むすべてのノードは、に示

した最小要件を満たしている必要があります ["Astra データストアの要件"](#)。

を参照してください。

- ["Astra ファミリーの紹介"](#)
- ["Astra Control Service のマニュアル"](#)
- ["Astra Control Center のドキュメント"](#)
- ["Astra Trident のドキュメント"](#)
- ["Astra API を使用"](#)
- ["Cloud Insights のドキュメント"](#)
- ["ONTAP のドキュメント"](#)

Astra データストアのストレージ効率

Astraデータストアでは、ネットアップのONTAP テクノロジとSolidFire テクノロジを基盤に、次のようなStorage Efficiencyテクノロジを使用しています。

- *** シンプロビジョニング ***：シンプロビジョニングされたボリュームは、ストレージが事前に予約されていないボリュームです。代わりに、ストレージは必要に応じて動的に割り当てられます。ボリュームまたは LUN 内のデータが削除されると、空きスペースはストレージシステムに戻されます
- **ゼロブロックの検出と排除**：シンプロビジョニングを使用したAstra Data Storeストレージシステムは、ボリュームが初期化されている領域を検出して、その領域を再利用して他の場所で使用することができます。
- *** 圧縮 ***：圧縮では、データブロックを圧縮グループに集約し、各データブロックを 1 つのブロックとして格納することで、ボリュームに必要な物理ストレージの量を削減します。Astra Data Storeでは、ファイル全体ではなく、要求されたデータを含む圧縮グループのみを解凍するため、従来の圧縮手法よりも圧縮されたデータの読み取りが高速になります。
- *** 重複排除 ***：重複するブロックを破棄して単一の共有ブロックへの参照に置き換えることで、ボリューム（またはAFF アグリゲート内のすべてのボリューム）に必要なストレージ容量を削減します。通常、重複排除されたデータの読み取りがパフォーマンスに影響することはありません。ノードに負荷が集中している場合を除き、書き込みによる影響もほとんどありません。

これらの機能はすべてデフォルトで有効になっています。

を参照してください ["Storage Efficiency の詳細"](#)。

を参照してください。

- ["Astra ファミリーの紹介"](#)
- ["Astra Control Service のマニュアル"](#)
- ["Astra Control Center のドキュメント"](#)
- ["Astra Trident のドキュメント"](#)
- ["Astra Control API を使用"](#)
- ["ONTAP のドキュメント"](#)

Astra データストアのセキュリティ

Astra データストアでは、クライアントと管理者がストレージにアクセスするのを保護し、通信とデータを保護し、ウイルスから保護するために、いくつかの方法を使用します。

Astra データストアでは、次のセキュリティ方法を使用します。

- MTLS（相互トランスポート層セキュリティ）を使用した通信暗号化
- 機能へのアクセスを制御するロールベースアクセス制御
- 展開のセキュリティ
- 証明書管理
- 内部および外部キー管理を含む保存データのソフトウェア暗号化

を参照してください。

- ["Astra ファミリーの紹介"](#)
- ["Astra Control Service のマニュアル"](#)
- ["Astra Control Center のドキュメント"](#)
- ["Astra Trident のドキュメント"](#)
- ["Astra Control API を使用"](#)
- ["Cloud Insights のドキュメント"](#)
- ["ONTAP のドキュメント"](#)

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。

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 を実行している必要があります。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 CNII
- 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す。

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）](#)：

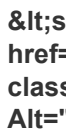
を参照してください。

["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> **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> **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> **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 ["ルールと権限"](#) 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ーを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 ["該当するバージョンのドキュメント"](#)。

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。

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。

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. 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

特定の環境では、オプションとして設定を変更して、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 ["OCPのインストール"](#)。

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsadddrives.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 を使用した導入：

```
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 AGE	STATUS	VERSION	SERIAL NUMBER	MVIP
astrads-example-cluster 10.x.x.x 13m	created	2022.05.0-X	e100000006	

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。

Astra データストアを使用

kubectlコマンドを使用してAstraデータストアのリソースを管理

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。

サンプルアプリの展開方法については、を参照してください ["テストアプリケーションを展開します"](#)。

クラスタのメンテナンスについては、を参照してください ["クラスタを管理"](#)。

必要なもの

- にインストールしたAstra Data Store kubectlプラグイン ["Astra データストアをインストール"](#)

Astraデータストア用のKubernetesカスタムAPIリソースを列挙

Kubernetes内でkubectlコマンドを使用して、Astraデータストアクラスタとやり取りし、状態を確認することができます。

「api-resources」コマンドにリストされた各項目は、Astraデータストアがクラスタの管理に内部的に使用するKubernetesカスタムリソース定義（CRD）を表します。

このリストは、後で示すように、各Astra Data Storeオブジェクトの簡単な名前を取得して入力を減らすのに特に役立ちます。

1. Astraデータストア用のKubernetesカスタムAPIリソースのリストを表示します。

```
kubectl api-resources --api-group astrads.netapp.io
```

対応：

NAME	SHORTNAMES	APIGROUP	NAMESPACED	KIND
astradsautosupports	adsas	astrads.netapp.io/v1alpha1	true	
AstraDSAutoSupport				
astradscLOUDsnapshots	adscs	astrads.netapp.io/v1alpha1	true	
AstraDSCloudSnapshot				
astradsclusters	adscl	astrads.netapp.io/v1alpha1	true	
AstraDSCluster				
astradsexportpolicies	adsep	astrads.netapp.io/v1alpha1	true	
AstraDSEExportPolicy				
astradsfaileddrives	adsfd	astrads.netapp.io/v1alpha1	true	
AstraDSFailedDrive				
astradslicenses	adsli	astrads.netapp.io/v1alpha1	true	
AstraDSLICENSE				
astradsnfsoptions	adsnf	astrads.netapp.io/v1alpha1	true	
AstraDSNfsOption				
astradsnodeinfoes	adsni	astrads.netapp.io/v1alpha1	true	
AstraDSNodeInfo				
astradsnodemanagements	adsnm	astrads.netapp.io/v1alpha1	true	
AstraDSNodeManagement				
astradsqospolicies	adsqp	astrads.netapp.io/v1alpha1	true	
AstraDSQosPolicy				
astradsversions	adsve	astrads.netapp.io/v1alpha1	true	
AstraDSVersion				
astradsvolumeFiles	adsvf	astrads.netapp.io/v1alpha1	true	
AstraDSVolumeFiles				
astradsvolumes	adsvo	astrads.netapp.io/v1alpha1	true	
AstraDSVolume				
astradsvolumesnapshots	adsvs	astrads.netapp.io/v1alpha1	true	
AstraDSVolumeSnapshot				

2. Kubernetesクラスタ内の現在のAstraデータストアオブジェクトをすべて取得するには、「`kubect1 get ads-a`」 コマンドを使用します。

```
kubect1 get ads -A
```

対応：

NAMESPACE	NAME	AGE
astrads-system	astradsqospolicy.astrads.netapp.io/bronze	45h
astrads-system	astradsqospolicy.astrads.netapp.io/gold	45h
astrads-system	astradsqospolicy.astrads.netapp.io/silver	45h

NAMESPACE	NAME	STATUS	VERSION	SERIAL NUMBER	MVIP	AGE
-----------	------	--------	---------	---------------	------	-----

```
astrads-system    astradscluster.astrads.netapp.io/astrads-cluster-9f1
created    arda-9.11.1    e000000009    10.224.8.146    46h
```

```
NAMESPACE        NAME
AGE
astrads-system    astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
astrads-system    astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
astrads-system    astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
astrads-system    astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
```

```
NAMESPACE        NAME                                AGE
astrads-system    astradsversion.astrads.netapp.io/astradsversion    46h
```

```
NAMESPACE        NAME                                AGE
astrads-system    astradsvolumefiles.astrads.netapp.io/test23        27h
astrads-system    astradsvolumefiles.astrads.netapp.io/test234        27h
astrads-system    astradsvolumefiles.astrads.netapp.io/test2345        4h22m
```

```
NAMESPACE        NAME                                SIZE    IP
CLUSTER          CREATED
astrads-system    astradsvolume.astrads.netapp.io/test234    21Gi
172.25.123.123    astrads-cluster-9f1    true
astrads-system    astradsvolume.astrads.netapp.io/test2345    21Gi
172.25.123.123    astrads-cluster-9f1    true
```

```
NAMESPACE        NAME
SEQUENCE COMPONENT          EVENT          TRIGGER    PRIORITY    SIZE
STATE
astrads-system    astradsautosupport.astrads.netapp.io/controlplane-
adsclustercreatesuccess-20211214t 9          controlplane
adsclustercreatesuccess k8sEvent    notice    0          uploaded
astrads-system    astradsautosupport.astrads.netapp.io/controlplane-
daily-20211215t0          15          controlplane    daily
periodic    notice    0          uploaded
astrads-system    astradsautosupport.astrads.netapp.io/controlplane-
daily-20211216t0          20          controlplane    daily
periodic    notice    0          uploaded
astrads-system    astradsautosupport.astrads.netapp.io/storage-
callhome.dbs.cluster.cannot.sync.blocks 10          storage
callhome.dbs.cluster.cannot.sync.blocks    firetapEvent    emergency    0
uploaded
```

NAMESPACE	NAME	ADSCluster
VALID PRODUCT	EVALUATION ENDDATE	VALIDATED
astrads-system	astradslicense.astrads.netapp.io/e0	astrads-cluster-
9f1 true Astra Data Store true	2022-02-07 2021-12-16T20:43:23Z	

- 短縮名の 1 つを使用して、クラスタ内のボリュームの現在の状態を表示します。

```
kubectl get adsvo -A
```

対応：

NAMESPACE	NAME	SIZE	IP	CLUSTER
CREATED				
astrads-system	test234	21Gi	172.25.138.109	astrads-cluster-
9f1c99f true				
astrads-system	test2345	21Gi	172.25.138.111	astrads-cluster-
9f1c99f true				

kubectl 拡張子の help オプションを使用します

'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 Data Store kubectl拡張機能のすべてのコマンドのヘルプを表示します。

```
kubectl astrads -h
```

対応：

```
A kubectl plugin for inspecting your AstraDS deployment

Usage:
  astrads [command]

Available Commands:
  asup          Manage AutoSupport
  clusters      Manage clusters
  drives        Manage drives in a cluster
  faileddrive   Manage drive replacement in a cluster
  help          Help about any command
  license       Manage license in the astrads cluster
  maintenance   Manage maintenance status of a node
```

monitoring	Manage Monitoring Output
nodes	Manage nodes in a cluster

Flags:

<code>--as</code> string	Username to impersonate for the operation
<code>--as-group</code> stringArray	Group to impersonate for the operation, this flag can be repeated to specify multiple groups.
<code>--cache-dir</code> string	Default HTTP cache directory (default "/u/arda/.kube/http-cache")
<code>--certificate-authority</code> string	Path to a cert file for the certificate authority
<code>--client-certificate</code> string	Path to a client certificate file for TLS
<code>--client-key</code> string	Path to a client key file for TLS
<code>--cluster</code> string	The name of the kubeconfig cluster to use
<code>--context</code> string	The name of the kubeconfig context to use
<code>-h, --help</code>	help for astrads
<code>--insecure-skip-tls-verify</code>	If true, the server's certificate will not be checked for validity. This will make your HTTPS connections insecure
<code>--kubeconfig</code> string	Path to the kubeconfig file to use for CLI requests.
<code>-n, --namespace</code> string	If present, the namespace scope for this CLI request
<code>--request-timeout</code> string	The length of time to wait for a single server request. Non-zero values should contain a corresponding time unit (e.g. 1s, 2m, 3h).
<code>-s, --server</code> string	The address and port of the Kubernetes API server
<code>--token</code> string	Bearer token for authentication to the API server
<code>--user</code> string	The name of the kubeconfig user to use

2. コマンドの詳細については 'astrads [command]--help' を使用してください

```
kubectl astrads asup collect --help
```

対応:

Collect the autosupport bundle by specifying the component to collect. It will default to manual event.

Usage:

```
astrads asup collect [flags]
```

Examples:

```
# Control plane collection
```

```
kubectl astrads collect --component controlplane example1
```

```
# Storage collection for single node
```

```
kubectl astrads collect --component storage --nodes node1 example2
```

```
# Storage collection for all nodes
```

```
kubectl astrads collect --component storage --nodes all example3
```

```
# Collect but don't upload to support
```

```
kubectl astrads collect --component controlplane --local example4
```

NOTE:

```
--component storage and --nodes <name> are mutually inclusive.
```

```
--component controlplane and --nodes <name> are mutually exclusive.
```

Flags:

```
-c, --component string      Specify the component to collect:  
[storage , controlplane , vasaprovider, all]
```

```
-d, --duration int          Duration is the duration in hours from  
the startTime for collection  
of AutoSupport.
```

```
-e, --event string          Specify the callhome event to trigger.  
(default "manual")
```

```
-f, --forceUpload           Configure an AutoSupport to upload if  
it is in the compressed state  
and not  
uploading because it was created with  
the 'local' option or if  
automatic uploads of AutoSupports is
```

```

disabled
                                at the cluster level.
    -h, --help                    help for collect
    -l, --local                    Only collect and compress the
autosupport bundle. Do not upload
                                to support.
                                Use 'download' to copy the collected
bundle after it is in
                                the 'compressed' state
                                Specify nodes to collect for storage
                                --nodes string
                                component. (default "all")
                                -t, --startTime string    StartTime is the starting time for
collection of AutoSupport.
                                This should be in the ISO 8601 date
time format.
                                Example format accepted:
                                2021-01-01T15:20:25Z, 2021-01-
01T15:20:25-05:00
                                -u, --usermessage string  UserMessage is the additional message
to include in the
                                AutoSupport subject.
                                (default "Manual event trigger from
CLI")

```

テストアプリケーションを展開します

Astra データストアで利用できるテストアプリケーションを導入する手順は次のとおりです。

この例では、Helm リポジトリを使用して Bitnami から MongoDB チャートを導入します。

必要なもの

- Astra データストア クラスターの導入と構成
- Trident のインストールが完了しました

手順

1. Bitnami から Helm repo を追加します。

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. MongoDB の導入

```
helm install mongohelm4 --set persistence.storageClass=trident-csi
bitnami/mongodb --namespace=ns-mongodb --create-namespace
```


3. MongoDB ポッドのステータスを確認します。

```
~% kubectl get pods -n ns-mongodb
NAME                                READY   STATUS    RESTARTS   AGE
mongodb-9846ff8b7-rfr4r            1/1     Running   0           67s
```

4. MongoDB で使用した Persistent Volume Claim (PVC ; 永続ボリューム要求)を確認します。

```
~% kubectl get pvc -n ns-mongodb
NAME          STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS  AGE
mongodb       Bound    pvc-1133453a-e2f5-48a5                   8Gi        RWO
trident-csi    97s
```

5. kubectl コマンド 'get astradsvolume' を使用して 'ボリュームを一覧表示します

```
~% kubectl get astradsvolume pvc-1133453a-e2f5-48a5 -n astrads-system
NAME                                SIZE          IP              CLUSTER      CREATED
pvc-1133453a-e2f5-48a5            8830116Ki     10.192.2.192    jai-ads      true
```

6. kubectl コマンド 'describe astadsvolume' を使用して 'ボリュームを説明します

```
~% kubectl describe astradsvolume pvc-1133453a-e2f5-48a5 -n astrads-
system
Name:          pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Namespace:     astrads-system
Labels:        astrads.netapp.io/cluster=jai-ads
               astrads.netapp.io/mip=10.192.1.39
               astrads.netapp.io/volumeUUID=cf33fd38-a451-596c-b656-
61b8270d2b5e
               trident.netapp.io/cloud=on-prem
               trident.netapp.io/creator=trident-dev
               trident.netapp.io/performance=premium
Annotations:   provisioning: {"provisioning":{"cloud":"on-
prem","creator":"trident-dev","performance":"premium"}}
               trident:
                 {"trident":{"version":"21.10.0-test.jenkins-trident-
stable-v21.10-
2+e03219ce37294d9ba54ec476bbe788c1a7772548","backendUUID":"","platform":
...
API Version:   astrads.netapp.io/v1alpha1
Kind:          AstraDSVolume
Metadata:
```

```
Creation Timestamp: 2021-12-08T19:35:26Z
Finalizers:
  trident.netapp.io/astradsvolume-finalizer
  astrads.netapp.io/astradsvolume-finalizer
Generation: 1
Managed Fields:
  API Version: astrads.netapp.io/v1alpha1
  Fields Type: FieldsV1
  fieldsV1:
    f:metadata:
      f:labels:
        f:astrads.netapp.io/cluster:
        f:astrads.netapp.io/mip:
        f:astrads.netapp.io/volumeUUID:
    f:status:
      .:
      f:cluster:
      f:conditions:
      f:created:
      f:displayName:
      f:exportAddress:
      f:internalName:
      f:mip:
      f:permissions:
      f:qosPolicy:
      f:requestedSize:
      f:restoreCacheSize:
      f:size:
      f:snapshotReservePercent:
      f:state:
      f:volumePath:
      f:volumeUUID:
  Manager: cluster-controller
  Operation: Update
  Time: 2021-12-08T19:35:32Z
  API Version: astrads.netapp.io/v1alpha1
  Fields Type: FieldsV1
  fieldsV1:
    f:status:
      f:exportPolicy:
  Manager: dms-controller
  Operation: Update
  Subresource: status
  Time: 2021-12-08T19:35:32Z
  API Version: astrads.netapp.io/v1alpha1
  Fields Type: FieldsV1
```

```

fieldsV1:
  f:metadata:
    f:annotations:
      .:
    f:provisioning:
    f:trident:
  f:finalizers:
    v:"trident.netapp.io/astradsvolume-finalizer":
  f:labels:
    .:
    f:trident.netapp.io/cloud:
    f:trident.netapp.io/creator:
    f:trident.netapp.io/performance:
  f:spec:
    .:
    f:cluster:
    f:displayName:
    f:exportPolicy:
    f:noSnapDir:
    f:permissions:
    f:qosPolicy:
    f:size:
    f:snapshotReservePercent:
    f:type:
    f:volumePath:

Manager:      trident_orchestrator
Operation:    Update
Time:         2021-12-08T19:35:34Z
Resource Version: 12007115
UID:          d522ae4f-e793-49ed-bbe0-9112d7f9167b
Spec:
  Cluster:      jai-ads
  Display Name:  pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
  Export Policy: pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
  No Snap Dir:  true
  Permissions:  0777
  Qos Policy:    silver
  Size:          9042036412
  Snapshot Reserve Percent: 5
  Type:          ReadWrite
  Volume Path:   /pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Status:
  Cluster:      jai-ads
  Conditions:
    Last Transition Time: 2021-12-08T19:35:32Z
    Message:             Volume is online

```

```

Reason:                VolumeOnline
Status:                True
Type:                 AstraDSVolumeOnline
Last Transition Time:  2021-12-08T19:35:32Z
Message:              Volume creation request was successful
Reason:                VolumeCreated
Status:                True
Type:                 AstraDSVolumeCreated
Created:              true
Display Name:         pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Export Address:       10.192.2.192
Export Policy:        pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Internal Name:        pvc_1133453a_e2f5_48a5_a06c_d14b8aa7be07
Mip:                  10.192.1.192
Permissions:          777
Qos Policy:           silver
Requested Size:        9042036412
Restore Cache Size:    0
Size:                  8830116Ki
Snapshot Reserve Percent: 5
State:                 online
Volume Path:          /pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Volume UUID:          cf33fd38-a451-596c-b656-61b8270d2b5e
Events:
  Type      Reason          Age    From                      Message
  ----      -
  Normal    VolumeCreated    3m9s   ADSClusterController    Volume creation
request was successful

```

Astraデータストアクラスタを管理

クラスタを管理するには、Astraデータストアでkubectlコマンドを使用します。

- [\[Add a node\]](#)
- [\[Remove a node\]](#)
- [\[Place a node in maintenance mode\]](#)
- [\[Add drives to a node\]](#)
- [\[Replace a drive\]](#)

必要なもの

- kubectl および kubectl-astras プラグインがインストールされたシステム。を参照してください ["Astra データストアをインストール"](#)。

ノードを追加します

追加するノードは Kubernetes クラスタに含まれ、クラスタ内の他のノードと同様の設定である必要があります。



Astra Control Centerを使用してノードを追加するには、を参照してください ["ストレージバックエンドクラスタにノードを追加します"](#)。

手順

1. 新しいノードのdataIPがまだAstra Data StoreクラスタCRに含まれていない場合は、次の手順を実行します。
 - a. クラスタCRを編集し、「adsDataNetworks*Addresses *」フィールドにデータIPを追加します。大文字の情報を環境に適した値に置き換えます。

```
kubectl edit astradscluster CLUSTER_NAME -n astrads-system
```

対応：

```
adsDataNetworks:
  -addresses: dataIP1,dataIP2,dataIP3,dataIP4,NEW_DATA_IP
```

- a. CR を保存します。
- b. アストラデータストアクラスタにノードを追加します。大文字の情報を環境に適した値に置き換えます。

```
kubectl astrads nodes add --cluster CLUSTER_NAME
```

2. それ以外の場合はノードを追加します。大文字の情報を環境に適した値に置き換えます。

```
kubectl astrads nodes add --cluster CLUSTER_NAME
```

3. ノードが追加されたことを確認します。

```
kubectl astrads nodes list
```

ノードを削除します

クラスタ内のノードを削除するには、kubectlコマンドをAstraデータストアとともに使用します。

手順

1. すべてのノードを一覧表示します。

```
kubectl astrads nodes list
```

対応：

NODE NAME	NODE STATUS	CLUSTER NAME
sti-rx2540-534d...	Added	cluster-multinodes-21209
sti-rx2540-535d...	Added	cluster-multinodes-21209
...		

2. ノードを削除対象としてマークします。大文字の情報を環境に適した値に置き換えます。

```
kubectl astrads nodes remove NODE_NAME
```

対応：

```
Removal label set on node sti-rx2540-534d.lab.org  
Successfully updated ADS cluster cluster-multinodes-21209 desired node  
count from 4 to 3
```

ノードを削除対象としてマークするとノードのステータスはアクティブから現在の状態に変わります

3. 削除したノードの現在のステータスを確認します

```
kubectl get nodes --show-labels
```

対応：

NAME	STATUS	ROLES
sti-astramaster-050.lab.org v1.20.0 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-astramaster-050.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=	Ready	control-plane,master 3h39m
sti-rx2540-556a.lab.org v1.20.0 astrads.netapp.io/cluster=astrads-cluster-890c32c,astrads.netapp.io/storage-cluster-status=active,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-556a.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true	Ready	worker 3h38m
sti-rx2540-556b.lab.org v1.20.0 astrads.netapp.io/cluster=astrads-cluster-890c32c,astrads.netapp.io/storage-cluster-status=active,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-556b.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true	Ready	worker 3h38m
sti-rx2540-534d.lab.org v1.20.0 astrads.netapp.io/storage-cluster-status=present,astrads.netapp.io/storage-node-removal=,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-557a.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true	Ready	worker 3h38m
sti-rx2540-557b.lab.org v1.20.0 astrads.netapp.io/cluster=astrads-cluster-890c32c,astrads.netapp.io/storage-cluster-status=active,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-557b.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true	Ready	worker 3h38m

4. ノードからAstraデータストアをアンインストールします。大文字の情報を環境に適した値に置き換えます。

```
kubectl astrads nodes uninstall NODE_NAME
```

5. ノードがクラスタから削除されたことを確認します。

```
kubectl astrads nodes list
```

ノードがAstraデータストアから削除される。

ノードをメンテナンスモードにします

ホストのメンテナンスやパッケージのアップグレードが必要な場合は、ノードをメンテナンスモードにする必要があります。



ノードがAstraデータストアクラスタにすでに含まれている必要があります。

ノードが保守モードのときは、クラスタにノードを追加できません。この例では、ノード「nhcitj1525」をメンテナンスモードにします。

手順

1. ノードの詳細を表示します。

```
kubectl get nodes
```

対応：

NAME	STATUS	ROLES	AGE	VERSION
nhcitjj1525	Ready	<none>	3d18h	v1.20.0
nhcitjj1526	Ready	<none>	3d18h	v1.20.0
nhcitjj1527	Ready	<none>	3d18h	v1.20.0
nhcitjj1528	Ready	<none>	3d18h	v1.20.0
scs000039783-1	Ready	control-plane,master	3d18h	v1.20.0

2. ノードがまだメンテナンスモードになっていないことを確認します。

```
kubectl astrads maintenance list
```

応答（メンテナンスモードのノードがありません）：

NAME	NODE NAME	IN MAINTENANCE	MAINTENANCE STATE	MAINTENANCE VARIANT
------	-----------	----------------	-------------------	---------------------

3. メンテナンスモードを有効にします。大文字の情報を環境に適した値に置き換えます。

```
kubectl astrads maintenance create CR_NAME --node-name=NODE_NAME  
--variant=Node
```

例：


```
kubectl astrads maintenance create maint1 --node-name="nhcitjj1525"
--variant=Node
```

対応：

```
Maintenance mode astrads-system/maint1 created
```

4. ノードを一覧表示します。

```
kubectl astrads nodes list
```

対応：

NODE NAME	NODE STATUS	CLUSTER NAME
nhcitjj1525	Added	ftap-astra-012
...		

5. メンテナンスモードのステータスを確認します。

```
kubectl astrads maintenance list
```

対応：

NAME	NODE NAME	IN MAINTENANCE	MAINTENANCE STATE
MAINTENANCE VARIANT			
node4	nhcitjj1525	true	ReadyForMaintenance
			Node

保守モードの場合は 'false' として起動し 'true' に変更します「保守状態」が「準備期間の保守」から「ReadyforMaintenance」に変更されます。

6. ノードのメンテナンスが完了したら、メンテナンスモードを無効にします。

```
kubectl astrads maintenance update maint1 --node-name="nhcitjj1525"
--variant=None
```

7. ノードが保守モードでなくなったことを確認します。

```
kubectl astrads maintenance list
```

ノードにドライブを追加

Astraデータストアでkubectlコマンドを使用して、Astraデータストアクラスタ内のノードに物理ドライブまたは仮想ドライブを追加。

必要なもの

- 次の条件を満たす1つ以上のドライブ：
 - ノードに設置済み（物理ドライブ）またはノードVMに追加済み（仮想ドライブ）
 - ドライブにパーティションがありません
 - ドライブがクラスタで使用されていません
 - クラスタ内のドライブの物理容量がライセンスで許可されている物理容量を超えない（たとえば、CPUコアあたり2TBのストレージをライセンスで付与すると、10ノードのクラスタの最大物理ドライブ容量は20TBになります）
 - ドライブには、ノード内の他のアクティブドライブのサイズ以上が含まれています



Astraデータストアのノードあたりのドライブ数は16本以下17台目のドライブを追加しようとすると、ドライブ追加要求は拒否されます。

手順

1. クラスタについて説明します。

```
kubectl astrads clusters list
```

対応：

CLUSTER NAME	CLUSTER STATUS	NODE COUNT
cluster-multinodes-21209	created	4

2. クラスタ名をメモします。
3. クラスタ内のすべてのノードに追加できるドライブを表示します。cluster_nameをクラスタの名前に置き換えます。

```
kubectl astrads drives adddrive show-available --cluster=CLUSTER_NAME
```

対応：

```

Node: node1.name
Add drive maximum size: 100.0 GiB
Add drive minimum size: 100.0 GiB
NAME IDPATH SERIAL PARTITIONCOUNT SIZE ALREADYINCLUSTER
sdg /dev/disk/by-id/scsi-3c290e16d52479a9af5eac c290e16d52479a9af5eac 0
100 GiB false
sdh /dev/disk/by-id/scsi-3c2935798df68355dee0be c2935798df68355dee0be 0
100 GiB false

Node: node2.name
Add drive maximum size: 66.7 GiB
Add drive minimum size: 100.0 GiB
No suitable drives to add exist.

Node: node3.name
Add drive maximum size: 100.0 GiB
Add drive minimum size: 100.0 GiB
NAME IDPATH SERIAL PARTITIONCOUNT SIZE ALREADYINCLUSTER
sdg /dev/disk/by-id/scsi-3c29ee82992ed7a36fc942 c29ee82992ed7a36fc942 0
100 GiB false
sdh /dev/disk/by-id/scsi-3c29312aa362469fb3da9c c29312aa362469fb3da9c 0
100 GiB false

Node: node4.name
Add drive maximum size: 66.7 GiB
Add drive minimum size: 100.0 GiB
No suitable drives to add exist.

```

4. 次のいずれかを実行します。

- 使用可能なすべてのドライブの名前が同じ場合は、それらのドライブをそれぞれのノードに同時に追加できます。大文字の情報を環境に適した値に置き換えます。

```

kubect1 astrads drives adddrive create --cluster=CLUSTER_NAME --name
REQUEST_NAME --drivesbyname all=DRIVE_NAME

```

- ドライブの名前が異なる場合は、各ノードにドライブを1つずつ追加できます（追加する必要があるドライブごとにこの手順を繰り返す必要があります）。大文字の情報を環境に適した値に置き換えます。

```

kubect1 astrads drives adddrive create --cluster=CLUSTER_NAME --name
REQUEST_NAME --drivesbyname NODE_NAME=DRIVE_NAME

```

Astraデータストアがドライブの追加要求を作成し、要求の結果を含むメッセージが表示される。

ドライブを交換します

クラスタ内のドライブで障害が発生した場合は、データの整合性を確保するために、できるだけ早くドライブを交換する必要があります。ドライブで障害が発生した場合は、クラスタのCRノードステータス、クラスタの健全性状態、および指標エンドポイントにある、障害が発生したドライブに関する情報を確認できます。障害が発生したドライブの情報を表示するには、次のコマンドを使用します。

nodeStatus.driveStatuses で障害が発生したドライブを示すクラスタの例

```
kubectl get adsc1 -A -o yaml
```

対応：

```
...
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
...
nodeStatuses:
  - driveStatuses:
    - driveID: 31205e51-f592-59e3-b6ec-185fd25888fa
      driveName: scsi-36000c290ace209465271ed6b8589b494
      drivesStatus: Failed
    - driveID: 3b515b09-3e95-5d25-a583-bee531ff3f31
      driveName: scsi-36000c290ef2632627cb167a03b431a5f
      drivesStatus: Active
    - driveID: 0807fa06-35ce-5a46-9c25-f1669def8c8e
      driveName: scsi-36000c292c8fc037c9f7e97a49e3e2708
      drivesStatus: Active
  ...
```

障害が発生したドライブCRは、障害が発生したドライブのUUIDに対応する名前でクラスタ内に自動的に作成されます。

```
kubectl get adsfd -A -o yaml
```

対応：

```

...
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSFailedDrive
metadata:
  name: c290a-5000-4652c-9b494
  namespace: astrads-system
spec:
  executeReplace: false
  replaceWith: ""
status:
  cluster: arda-6e4b4af
  failedDriveInfo:
    failureReason: AdminFailed
    inUse: false
    name: scsi-36000c290ace209465271ed6b8589b494
    path: /dev/disk/by-id/scsi-36000c290ace209465271ed6b8589b494
    present: true
    serial: 6000c290ace209465271ed6b8589b494
    node: sti-rx2540-300b.lab.org
  state: ReadyToReplace

```

```
kubectl astrads faileddrive list --cluster arda-6e4b4af
```

対応：

NAME	NODE	CLUSTER	STATE
AGE			
6000c290	sti-rx2540-300b.lab.netapp.com	ard-6e4b4af	ReadyToReplace
13m			

手順

1. 交換可能なドライブを'kubectl strads faileddrive show -replacions'コマンドで一覧表示しますこのコマンドは'交換の制限に適合するドライブをフィルタリングします（クラスタでは未使用'マウントなし'パーティションなし'または障害が発生したドライブ以上）

可能な交換用ドライブをフィルタリングせずにすべてのドライブを一覧表示するには '--all' を 'show-replacements' コマンドに追加します

```
kubectl astrads faileddrive show-replacements --cluster ard-6e4b4af
--name 6000c290
```

対応：

NAME	IDPATH	SERIAL	PARTITIONCOUNT	MOUNTED	SIZE
sdh	/scsi-36000c29417	45000c	0	false	100GB

2. パスしたシリアル番号でドライブを交換するには 'replace' コマンドを使用します。コマンドは置換を完了するか '--wait' 時間が経過すると失敗します。

```
kubectl astrads faileddrive replace --cluster arda-6e4b4af --name
6000c290 --replaceWith 45000c --wait
Drive replacement completed successfully
```



kubectl の astrads faileddrive replace' が不適切なシリアル番号を使用して実行された場合、次のようなエラーが表示されます。

```
kubectl astrads replacedrive replace --cluster astrads-cluster-f51b10a
--name 6000c2927 --replaceWith BAD_SERIAL_NUMBER
Drive 6000c2927 replacement started
Failed drive 6000c2927 has been set to use BAD_SERIAL_NUMBER as a
replacement
...
Drive replacement didn't complete within 25 seconds
Current status: {FailedDriveInfo:{InUse:false Present:true Name:scsi-
36000c2 FiretapUUID:444a5468 Serial:6000c Path:/scsi-36000c
FailureReason:AdminFailed Node:sti-b200-0214a.lab.netapp.com}
Cluster:astrads-cluster-f51b10a State:ReadyToReplace
Conditions:[{Message: "Replacement drive serial specified doesn't
exist", Reason: "DriveSelectionFailed", Status: False, Type:' Done']]}
```

3. ドライブ交換を再実行するには '前のコマンドで --force' を使用します。

```
kubectl astrads replacedrive replace --cluster astrads-cluster-f51b10a
--name 6000c2927 --replaceWith VALID_SERIAL_NUMBER --force
```

を参照してください。

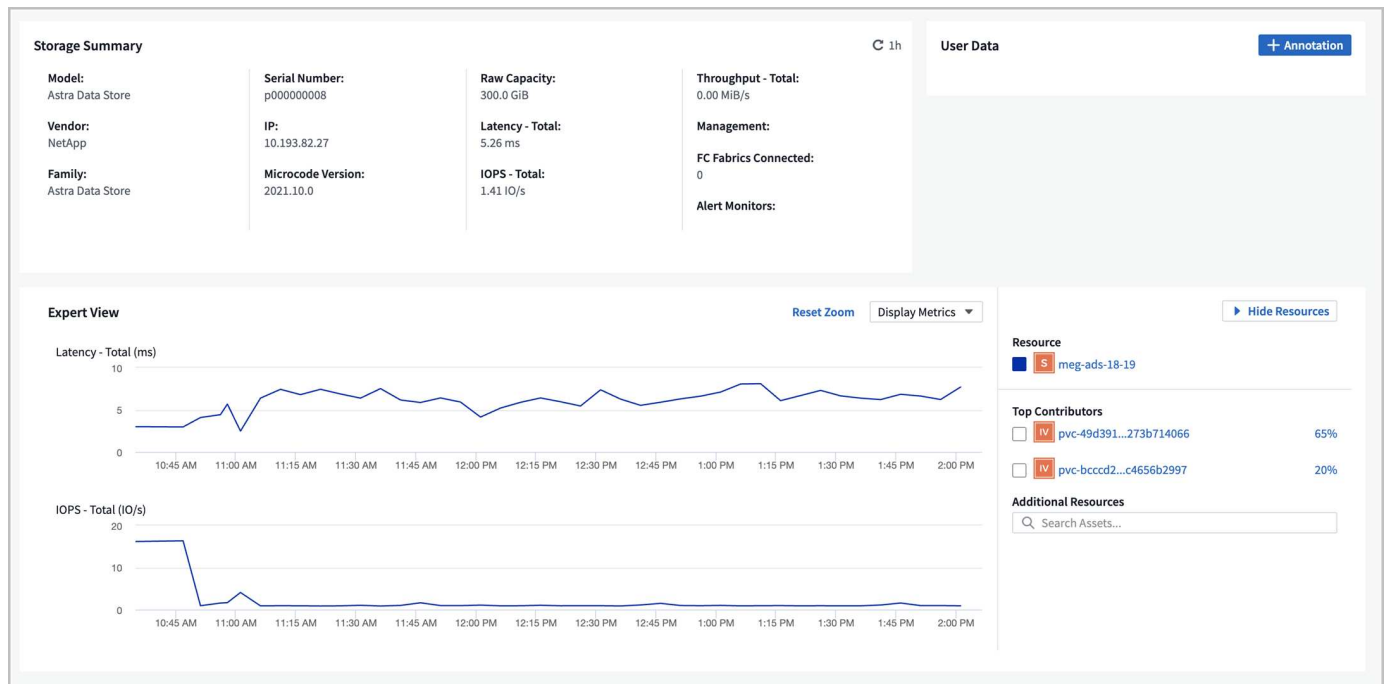
- ["kubectlコマンドを使用してAstraデータストアのリソースを管理"](#)

Astraデータストアを監視

Cloud Insights で指標を監視

Cloud Insights を使用して、Astraデータストアの指標を監視できます。

Cloud Insights に表示されるAstraデータストアの指標の例を次に示します。



を使用して、Astraデータストアで生成された指標のリストを表示することもできます [\[Open Metrics API help\]](#)。

次のタスクを実行できます。

- [\[Complete Cloud Insights connection prerequisite tasks\]](#)
- [\[Acquisition Unit storage\]](#)
- [\[Download and run the installation script\]](#)
- [\[Edit the Cloud Insights connection\]](#)
- [\[Disconnect from Cloud Insights\]](#)

Cloud Insights 接続の前提条件となる作業を完了する

Cloud Insights を使用して Astra データストアに接続する前に、次の作業を完了する必要があります。

- ["Astra Data Store Monitoring Operator をインストールします"](#) これは、Astra Data Storeのインストール手順の一部です。
- ["kubecti-astras バイナリをインストールします"](#) これは、Astra Data Storeのインストール手順の一部です。
- ["Cloud Insights アカウントを作成します"](#)。
- 'awk'、'curl'、'grep' および 'jq' の各コマンドが使用可能であることを確認します

次の情報を収集します。

- * Cloud Insights API アクセストークン *。各カテゴリに対する読み取り / 書き込み権限：Acquisition Unit、Data Collection、Data Ingestion、Log Ingestionこれは、読み取り / 書き込み処理、Acquisition Unit のセットアップ、およびデータの取り込みプロセスのセットアップに使用されます。

- * Kubernetes API サーバの IP アドレスとポート *。これは、Astra データストア クラスターを監視するために使用されます。
- * Kubernetes API トークン *。これは Kubernetes API を呼び出すために使用されます。
- * 永続ボリューム構成 *。永続ボリュームのプロビジョニング方法に関する情報。

Acquisition Unit のストレージ

Acquisition Unit には、インストールファイル、設定データ、およびログを格納するための永続ボリュームが 3 つ必要です。Monitoring Operator は、デフォルトのストレージクラスを使用して、永続的ボリューム要求を作成します。インストーラ・スクリプトの実行時に '-s' オプションを使用して '別のストレージ・クラス名を指定できます

Kubernetes クラスターにストレージプロビジョニングツール（NetApp Trident など）がない場合は、インストーラ スクリプトの実行時に '-r' オプションを使用してローカルファイルシステムパスを提供できます。'-r' オプションが設定されている場合 'インストーラ・スクリプトは '指定されたディレクトリ内に 3 つの永続ボリュームを作成しますこのディレクトリには、150GB 以上の空きスペースが必要です。

インストールスクリプトをダウンロードして実行します

Cloud Insights では、モニタリングオペレータによるアストラデータストアのモニタリングを有効にするための Bash スクリプトを提供しています。インストールスクリプトは、Astra Data Store コレクタと Fluent ビットエージェントを備えた Acquisition Unit をインストールします。

Cloud Insights テナントのドメイン名と選択した Cloud Insights API アクセストークンは、ダウンロード時にインストーラ スクリプトに組み込まれます。

その後、次のように指標が送信されます。

- Cloud Insights Acquisition Unit から Cloud Insights データレイクに指標が送信されます。
- FLUENT ビットは、ログ取り込みサービスにログを送信します。

インストーラ スクリプトのヘルプを表示します

インストーラ スクリプトの完全なヘルプテキストを次に示します。

インストーラ スクリプトのヘルプテキストを表示します。

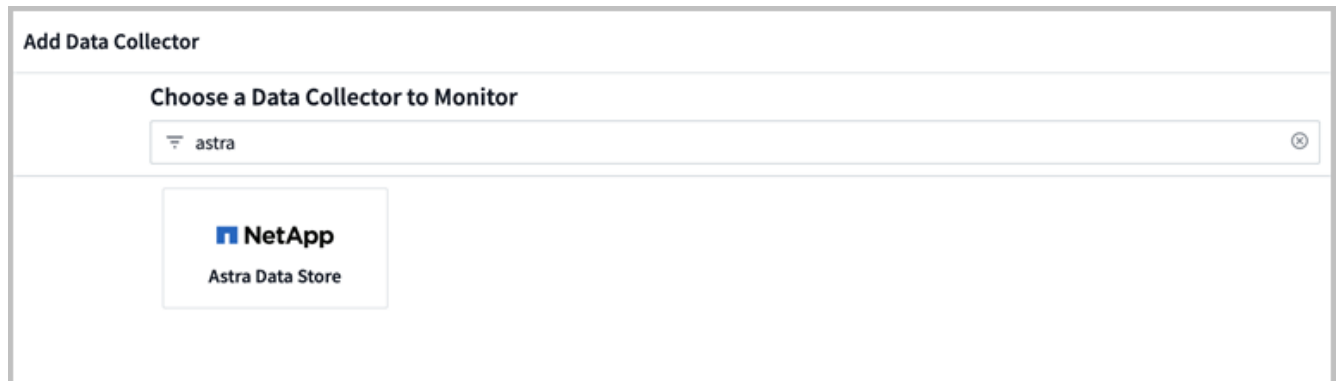
```
./cloudinsights-ads-monitoring.sh -h
```

対応：


```
USAGE: cloudinsights-ads-monitoring.sh [OPTIONS]
Configure monitoring of Astra Data Store by Cloud Insights.
OPTIONS:
  -h                                Display this help message.
  -d ci_domain_name                 Cloud Insights tenant domain name.
  -i kubernetes_ip                 Kubernetes API server IP address.
  -k ci_api_key                     Cloud Insights API Access Token.
  -n namespace                      Namespace for monitoring components. (default:
netapp-monitoring)
  -p kubernetes_port                Kubernetes API server port. (default: 6443)
  -r root_pv_dir                   Create 3 Persistent Volumes in this directory
for the Acquisition Unit.
                                   Only specify this option if there is no Storage
Provisioner installed and the PVs do not already exist.
  -s storage_class                 Storage Class name for provisioning Acquisition
Unit PVs. If not specified, the default storage class will be used.
  -t kubernetes_token              Kubernetes API server token.
```

インストールスクリプトを実行します

1. Cloud Insights アカウントがない場合は作成します。
2. Cloud Insights にログインします。
3. Cloud Insights メニューから、**Admin>*Data Collector*** をクリックします。
4. 「* + Data Collector *」 をクリックして、新しいコレクタを追加します。



5. 「* アストラデータストア *」 タイルをクリックします。
6. 正しい Cloud Insights API アクセストークンを選択するか、新しいトークンを作成します。
7. 指示に従って、インストーラスクリプトをダウンロードし、権限を更新し、スクリプトを実行します。

このスクリプトには、Cloud Insights テナントの URL と選択した Cloud Insights API アクセストークンが含まれています。



Select a Data Collector



Configure Collector




NetApp
Astra Data Store

Configure Collector

Configure Kubernetes Operator to monitor NetApp Astra Data Store (ADS).

What Operating System or Platform Are You Using?


Kubernetes

Select existing API Access Token or create a new one

default_ads_api_key1 (...d0gHof)

[+ API Access Token](#)

[Production Best Practices ?](#)

Configure Astra Data Store [Need Help?](#)

- 1

The commands *awk*, *curl*, *grep*, *jq*, *kubectl* and the *kubectl-astrads* plugin must be installed where the installer script is run. You will need the Kubernetes API server IP address and a Kubernetes API token to run this install script. See the documentation if you need help finding this information.
- 2

Copy Installer Script

☐ Reveal Installer Script

```
#!/usr/bin/env bash
SCRIPT='basename $0'

CI_DOMAIN_NAME="f49uaky.gstabler-ads.cloudinsights-dev.netapp.com"
CI_API_KEY="eyJraWQ1OiI5OTk5IiwidHlwIjoIc3R5bWVudF9hZHNfYXBPX2tleTEgKG9uIGJlaGFsZiBvZiBhZG1pbik"

```
- 3

Copy the above installer script and save it as *cloudinsights-ads-monitoring.sh*
- 4

Copy Permissions Command

☐ Reveal Permissions Command

```
chmod +x cloudinsights-ads-monitoring.sh

```
- 5

Paste the permissions command in a terminal to enable execute permissions on the installer script.
- 6

Copy Install Command

☐ Reveal Install Command

```
./cloudinsights-ads-monitoring.sh -i <KUBERNETES_IP> -t <KUBERNETES_TOKEN>

```
- 7

Paste the install command in a terminal, replace the placeholders with the correct values for your environment, and run the command. It will take several minutes to complete. The script will use the default namespace 'netapp-monitoring'. Additional options are available for customized environments. The default configuration for kubectl should point to the kubernetes cluster to be monitored.
- 8

Complete Setup

8. スクリプトが完了したら、[セットアップの完了]をクリックします。

インストールスクリプトが完了すると、「データソース」リストに「Astra Data Store」コレクタが表示されます。



エラーが原因でスクリプトが終了した場合は、エラーが解決してから再度実行できます。デフォルトの設定を使用しない環境では、Monitoring Operator 名前空間や Kubernetes API サーバポートなどの追加のパラメータがサポートされます。使い方とヘルプテキストを表示するには、cloudinsights-ads-monitoring.sh -h オプションを使用します。

設定が正常に完了すると、次のような出力が生成されます。

```
Configuring Cloud Insights monitoring for Astra Data Store . . .
Configuring monitoring namespace
...
Configuring output sink and Fluent Bit plugins
Configuring Acquisition Unit
...
Acquisition Unit has been installed successfully.
Configuring Astra Data Store data collector
Astra Data Store collector data '<CLUSTER_NAME>' created
Configuration done!
```

エージェント **CR** の例

以下に、インストーラスクリプトの実行後の「Monitoring - NetApp」エージェントの CR の例を示します。

```

spec:
  au:
    isEnabled: true
    storageClassName: auto-sc
  cluster-name: meg-ads-21-22-29-30
  docker-repo: docker.repo.eng.netapp.com/global/astra
  fluent-bit:
    - name: ads-tail
      outputs:
        - sink: ADS_STDOUT
      substitutions:
        - key: TAG
          value: firetapems
        - key: LOG_FILE
          values:
            - /var/log/firetap/*/ems/ems
            - /var/log/firetap/ems/*/ems/ems
        - key: ADS_CLUSTER_NAME
          value: meg-ads-21-22-28-29-30
    - name: agent
    - name: ads-tail-ci
      outputs:
        - sink: CI
      substitutions:
        - key: TAG
          value: netapp.ads
        - key: LOG_FILE
          values:
            - /var/log/firetap/*/ems/ems
            - /var/log/firetap/ems/*/ems/ems
        - key: ADS_CLUSTER_NAME
          value: meg-ads-21-22-28-29-30
  output-sink:
    - api-key: abcd
      domain-name: bz19ngz.gst-adsdemo.ci-dev.netapp.com
      name: CI
  serviceAccount: sa-netapp-monitoring
  status:
    au-pod-status: UP
    au-uuid: eddeccc6-3aa3-4dd2-a98c-220085fae6a9

```

Cloud Insights 接続を編集します

Kubernetes API トークンまたは Cloud Insights API アクセストークンはあとから編集できます。

- Kubernetes API トークンを更新する場合は、Cloud Insights UI から Astra データストアコレクタを編集する必要があります。
- テレメトリとログに使用される Cloud Insights API アクセストークンを更新する場合は、kubectl コマンドを使用して Monitoring Operator CR を編集する必要があります。

Kubernetes API トークンを更新します

1. Cloud Insights にログインします。
2. **[Admin>]** > **[* Data Collectors]** を選択して、**[Data Collectors]** ページにアクセスします。
3. Astra データストアクラスターのエントリを探します。
4. ページの右側にあるメニューをクリックし、「* 編集 *」を選択します。
5. Kubernetes API トークンフィールドを新しい値で更新します。
6. **[コレクタの保存 *]** を選択します

Cloud Insights API アクセストークンを更新します

1. Cloud Insights にログインします。
2. **[Admin>*API Access*]** を選択し、**[*+API アクセストークン *]** をクリックして、新しい Cloud Insights API アクセストークンを作成します。
3. エージェント CR を編集します。

```
kubectl --namespace netapp-monitoring edit agent agent-monitoring-netapp
```

4. 「output-sink」セクションを探し、「ci」という名前のエントリを見つけます。
5. ラベル「api-key」の場合は、現在の値を新しい Cloud Insights API アクセストークンに置き換えます。

セクションは次のようになります。

```
output-sink:
  - api-key: <api key value>
    domain-name: <tenant url>
    name: CI
```

6. エディタウィンドウを保存して終了します。

モニタリングオペレータは、新しいCloud Insights APIアクセストークンを使用するようにFluentビットを更新します。

Cloud Insights から切断します

Cloud Insights から切断するには、最初に Cloud Insights UI から Astra データストアコレクタを削除する必要があります。これが完了したら、モニタリングオペレータからAcquisition Unit、Telegraf（設定されている場合）、およびFluentビットの設定を削除できます。

Astra Data Store コレクタを削除

1. Cloud Insights にログインします。
2. [Admin>] > [* Data Collectors] を選択して、[Data Collectors] ページにアクセスします。
3. Astra データストアクラスターのエントリを探します。
4. 画面の右側のメニューを選択し、「* 削除 *」を選択します。
5. 確認ページで * Delete * をクリックします。

Acquisition Unit、Telegraf（設定されている場合）、およびFluentビットを取り外します

1. エージェント CR を編集します。

```
kubectl --namespace netapp-monitoring edit agent agent-monitoring-netapp
```

2. 「au」セクションを探し、「IsEnabled」を「false」に設定します
3. 「FLUENT ビット」セクションを探し、「ADS テール CI」という名前のプラグインを削除します。プラグインがない場合は、「FLUENT - BIT」セクションを削除できます。
4. Telegrafが設定されている場合は、「テレグラム」セクションを探し、「ADSオープンメトリック」という名前のプラグインを削除します。プラグインがない場合は、「テレグラム」セクションを削除できます。
5. 「output-sink」セクションを探し、「ci」という名前のシンクを取り外します。
6. エディタウィンドウを保存して終了します。

モニタリングオペレータがTelegraf（設定されている場合）およびFluentビット設定を更新し、Acquisition Unitポッドを削除します。

7. ストレージプロビジョニング担当者ではなく Acquisition Unit PVS にローカルディレクトリを使用した場合は、PVS を削除します。

```
kubectl delete pv au-lib au-log au-pv
```

次に、Acquisition Unit を実行していたノードの実際のディレクトリを削除します。

8. Acquisition Unit ポッドが削除されたら、Cloud Insights から Acquisition Unit を削除できます。
 - a. Cloud Insights メニューで、**Admin>*Data Collector*** を選択します。
 - b. [* Acquisition Units *（Acquisition Unit *）] タブをクリックします。
 - c. Acquisition Unit ポッドの横にあるメニューをクリックします。
 - d. 「* 削除」を選択します。

モニタリングオペレータは、Telegraf（設定されている場合）およびFluentビットの設定を更新し、Acquisition Unitを削除します。

Open Metrics API のヘルプを参照してください

Astraデータストアから指標を収集するために使用できるAPIのリストを次に示します。

- 「help」行は指標を表します。
- 「type」行は、メトリックがゲージかカウンタかを示します。

```
# HELP astrads_cluster_capacity_logical_percent Percentage cluster logical
capacity that is used (0-100)
# TYPE astrads_cluster_capacity_logical_percent gauge
# HELP astrads_cluster_capacity_max_logical Max Logical capacity of the
cluster in bytes
# TYPE astrads_cluster_capacity_max_logical gauge
# HELP astrads_cluster_capacity_max_physical The sum of the space in the
cluster in bytes for storing data after provisioning efficiencies, data
reduction algorithms and replication schemes are applied
# TYPE astrads_cluster_capacity_max_physical gauge
# HELP astrads_cluster_capacity_ops The IO operations capacity of the
cluster
# TYPE astrads_cluster_capacity_ops gauge
# HELP astrads_cluster_capacity_physical_percent The percentage of cluster
physical capacity that is used (0-100)
# TYPE astrads_cluster_capacity_physical_percent gauge
# HELP astrads_cluster_capacity_used_logical The sum of the bytes of data
in all volumes in the cluster before provisioning efficiencies, data
reduction algorithms and replication schemes are applied
# TYPE astrads_cluster_capacity_used_logical gauge
# HELP astrads_cluster_capacity_used_physical Used Physical capacity of a
cluster in bytes
# TYPE astrads_cluster_capacity_used_physical gauge
# HELP astrads_cluster_other_latency The sum of the accumulated latency in
seconds for other IO operations of all the volumes in a cluster. Divide by
astrads_cluster_other_ops to get the average latency per other operation
# TYPE astrads_cluster_other_latency counter
# HELP astrads_cluster_other_ops The sum of the other IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_other_ops counter
# HELP astrads_cluster_read_latency The sum of the accumulated latency in
seconds of read IO operations of all the volumes in a cluster. Divide by
astrads_cluster_read_ops to get the average latency per read operation
# TYPE astrads_cluster_read_latency counter
# HELP astrads_cluster_read_ops The sum of the read IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_read_ops counter
# HELP astrads_cluster_read_throughput The sum of the read throughput of
all the volumes in a cluster in bytes
```

```

# TYPE astrads_cluster_read_throughput counter
# HELP astrads_cluster_storage_efficiency Efficacy of data reduction
technologies. (logical used / physical used)
# TYPE astrads_cluster_storage_efficiency gauge
# HELP astrads_cluster_total_latency The sum of the accumulated latency in
seconds of all IO operations of all the volumes in a cluster. Divide by
astrads_cluster_total_ops to get average latency per operation
# TYPE astrads_cluster_total_latency counter
# HELP astrads_cluster_total_ops The sum of the IO operations of all the
volumes in a cluster
# TYPE astrads_cluster_total_ops counter
# HELP astrads_cluster_total_throughput The sum of the read and write
throughput of all the volumes in a cluster in bytes
# TYPE astrads_cluster_total_throughput counter
# HELP astrads_cluster_utilization_factor The ratio of the current cluster
IO operations based on recent IO sizes to the cluster iops capacity. (0.0
- 1.0)
# TYPE astrads_cluster_utilization_factor gauge
# HELP astrads_cluster_volume_used The sum of used capacity of all the
volumes in a cluster in bytes
# TYPE astrads_cluster_volume_used gauge
# HELP astrads_cluster_write_latency The sum of the accumulated latency in
seconds of write IO operations of all the volumes in a cluster. Divide by
astrads_cluster_write_ops to get the average latency per write operation
# TYPE astrads_cluster_write_latency counter
# HELP astrads_cluster_write_ops The sum of the write IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_write_ops counter
# HELP astrads_cluster_write_throughput The sum of the write throughput of
all the volumes in a cluster in bytes
# TYPE astrads_cluster_write_throughput counter
# HELP astrads_disk_base_seconds Base for busy, pending and queued.
Seconds since collection began
# TYPE astrads_disk_base_seconds counter
# HELP astrads_disk_busy Seconds the disk was busy. 100 *
(astrads_disk_busy / astrads_disk_base_seconds) = percent busy (0-100)
# TYPE astrads_disk_busy counter
# HELP astrads_disk_capacity Raw Capacity of a disk in bytes
# TYPE astrads_disk_capacity gauge
# HELP astrads_disk_io_pending Summation of the count of pending io
operations for a disk times time. Divide by astrads_disk_base_seconds to
get the average pending operation count
# TYPE astrads_disk_io_pending counter
# HELP astrads_disk_io_queued Summation of the count of queued io
operations for a disk times time. Divide by astrads_disk_base_seconds to
get the average queued operations count

```



```

# TYPE astrads_disk_io_queued counter
# HELP astrads_disk_read_latency Total accumulated latency in seconds for
disk reads. Divide by astrads_disk_read_ops to get the average latency per
read operation
# TYPE astrads_disk_read_latency counter
# HELP astrads_disk_read_ops Total number of read operations for a disk
# TYPE astrads_disk_read_ops counter
# HELP astrads_disk_read_throughput Total bytes read from a disk
# TYPE astrads_disk_read_throughput counter
# HELP astrads_disk_write_latency Total accumulated latency in seconds for
disk writes. Divide by astrads_disk_write_ops to get the average latency
per write operation
# TYPE astrads_disk_write_latency counter
# HELP astrads_disk_write_ops Total number of write operations for a disk
# TYPE astrads_disk_write_ops counter
# HELP astrads_disk_write_throughput Total bytes written to a disk
# TYPE astrads_disk_write_throughput counter
# HELP astrads_value_scrape_duration Duration to scrape values
# TYPE astrads_value_scrape_duration gauge
# HELP astrads_volume_capacity_available The minimum of the available
capacity of a volume and the available capacity of the cluster in bytes
# TYPE astrads_volume_capacity_available gauge
# HELP astrads_volume_capacity_available_logical Logical available
capacity of a volume in bytes
# TYPE astrads_volume_capacity_available_logical gauge
# HELP astrads_volume_capacity_percent Percentage of volume capacity
available (0-100). (capacity available / provisioned) * 100
# TYPE astrads_volume_capacity_percent gauge
# HELP astrads_volume_capacity_provisioned Provisioned capacity of a
volume in bytes after setting aside the snapshot reserve. (size - snapshot
reserve = provisioned)
# TYPE astrads_volume_capacity_provisioned gauge
# HELP astrads_volume_capacity_size Total capacity of a volume in bytes
# TYPE astrads_volume_capacity_size gauge
# HELP astrads_volume_capacity_snapshot_reserve_percent Snapshot reserve
percentage of a volume (0-100)
# TYPE astrads_volume_capacity_snapshot_reserve_percent gauge
# HELP astrads_volume_capacity_snapshot_used The amount of volume snapshot
data that is not in the active file system in bytes
# TYPE astrads_volume_capacity_snapshot_used gauge
# HELP astrads_volume_capacity_used Used capacity of a volume in bytes.
This is bytes in the active filesystem unless snapshots are consuming more
than the snapshot reserve. (bytes in the active file system + MAX(0,
snapshot_used-(snapshot_reserve_percent/100*size))
# TYPE astrads_volume_capacity_used gauge
# HELP astrads_volume_other_latency Total accumulated latency in seconds

```

```
for operations on a volume that are neither read or write. Divide by
astrads_volume_other_ops to get the average latency per other operation
# TYPE astrads_volume_other_latency counter
# HELP astrads_volume_other_ops Total number of operations for a volume
that are neither read or write
# TYPE astrads_volume_other_ops counter
# HELP astrads_volume_read_latency Total accumulated read latency in
seconds for a volume. Divide by astrads_volume_read_ops to get the average
latency per read operation
# TYPE astrads_volume_read_latency counter
# HELP astrads_volume_read_ops Total number of read operations for a
volume
# TYPE astrads_volume_read_ops counter
# HELP astrads_volume_read_throughput Total read throughput for a volume
in bytes
# TYPE astrads_volume_read_throughput counter
# HELP astrads_volume_total_latency Total accumulated latency in seconds
for all operations on a volume. Divide by astrads_volume_total_ops to get
the average latency per operation
# TYPE astrads_volume_total_latency counter
# HELP astrads_volume_total_ops Total number of operations for a volume
# TYPE astrads_volume_total_ops counter
# HELP astrads_volume_total_throughput Total throughput for a volume in
bytes
# TYPE astrads_volume_total_throughput counter
# HELP astrads_volume_write_latency Total accumulated write latency in
seconds for volume. Divide by astrads_volume_write_ops to get the average
latency per write operation
# TYPE astrads_volume_write_latency counter
# HELP astrads_volume_write_ops Total number of write operations for a
volume
# TYPE astrads_volume_write_ops counter
# HELP astrads_volume_write_throughput Total write throughput for a volume
in bytes
# TYPE astrads_volume_write_throughput counter
```

Prometheus と Grafana で指標を監視します

PrometheusとGrafanaを使用して、Astraデータストアの指標を監視できます。PrometheusでAstra Data Store Kubernetesクラスター指標エンドポイントから指標を収集するように設定でき、Grafanaを使用して指標データを表示できます。

必要なもの

- PrometheusパッケージとGrafanaパッケージをAstra Data Storeクラスター、またはAstra Data Storeクラスターと通信可能な別のクラスターでダウンロードしてインストールしておく必要があります。公式ドキュメントの指示に従って、各ツールをインストールします。

- ["Prometheus をインストールする"](#)
- ["Grafana をインストールします"](#)
- PrometheusとGrafanaがAstra Data Store Kubernetesクラスタと通信する必要があります。PrometheusとGrafanaがAstra Data Storeクラスタにインストールされていない場合は、Astra Data Storeクラスタで実行されている指標サービスと通信できることを確認する必要があります。

Prometheus を設定する

Astraデータストアは、KubernetesクラスタのTCPポート9341で指標サービスを公開します。このサービスから指標を収集するには、Prometheus を設定する必要があります。

手順

1. Prometheus インストール用の「prometheus.yml」構成ファイルを編集します。
2. Astra Data Storeサービス名とそのポートを指すサービスターゲットを追加します。例：

```
scrape_configs:
static_configs:
- targets: ['astrads-metrics-service.astrads-system:9341']
```

3. Prometheus サービスを開始します。

Grafana を設定します

Prometheus で収集された指標を表示するように Grafana を設定できます。

手順

1. Grafana インストールの「datasources.yaml」設定ファイルを編集します。
2. Prometheus をデータソースとして追加します。例：

```
apiVersion: 1

datasources:
- name: astradatastore-prometheus
  type: prometheus
  access: proxy
  url: http://localhost:9090
  jsonData:
    manageAlerts: false
```

3. Grafana サービスを開始します。
4. Grafana のマニュアルに記載されている手順に従ってください ["始めましょう"](#)。

Grafana ダッシュボードテンプレートをインポートします

Astra Data Storeをインストールするためにダウンロードしたバンドルファイルには、Grafana内からインポートできるGrafanaダッシュボードテンプレートファイルが含まれています。このダッシュボードテンプレートを使用すると、Astraデータストアから取得できる指標の種類とその表示方法を確認できます。

手順

1. Astra Data Storeの.tar.gzバンドルを開きます。
2. 'マニフェスト'ディレクトリを開きます
3. 「`graafana_cluster.json`」ファイルと「`graafana_volume.json`」ファイルを展開します。
4. Grafana Web UI の使用、["でのダッシュボードテンプレートファイルを Grafana にインポートします"](#)。

イベントログを設定して監視する

Event Management System（EMS；イベント管理システム）ログを監視するには、次の高度なタスクを実行します。

- [\[Configure monitoring in the Astra Data Store cluster custom resource \(CR\)\]](#)
- [\[Set up Cloud Insights\]](#)
- [\[Stream event logs to Elastic\]](#)。

Astra Data Storeクラスタカスタムリソース（CR）での監視の設定

Astra Data StoreクラスタCRでモニタリングオプションが設定されていない場合は、「astras」拡張機能を使用して設定できます。

入力するコマンド

```
kubectl astrads monitoring setup -n <NAMESPACE OF AGENT INSTALLED> -r  
<DOCKER REPO TO FIND FLUENT/TELEGRAF ETC IMAGES>
```

ここで、

- インストールされているエージェントのネームスペース：Monitoring Agent のネームスペースを入力します。この名前は、Monitoring Operator のデフォルトの名前である NetApp CR になります。
- -r は 'Fluent イメージまたは Telegraf イメージが置かれている Docker レジストリをセットアップするためのオプションですデフォルトでは 'パスは `docker.repo.eng.netapp.com/global/astra` ' に設定されていますこのパスは変更できます

Cloud Insights をセットアップする

ログを表示するには Cloud Insights の設定は任意ですが、Cloud Insights を使用してデータを表示すると便利です。を参照してください ["NetApp Cloud Insights のセットアップ方法"](#) Astraデータストアで使用。

イベントログを Elastic にストリーミングする

EMS イベントやその他のポッドログを Elastic などのサードパーティのエンドポイントにストリーミングする

には、「astrads」拡張機能を使用します。

入力するコマンド

```
kubectl astrads monitoring --host <ELASTIC HOST NAME> --port <ELASTIC HOST PORT> es
```



Elastic ホスト名は IP アドレスでもかまいません。

Secure Astra データストア

セキュリティ証明書を管理する

Astra データストアでは、クラスタのソフトウェアコンポーネント間でMTLS (Mutual Transport Layer Security) 暗号化を使用しています。各Astra Data Storeクラスタには、自己署名ルートCA証明書 (「astrs-cert-root」) と中間CA証明書 (「astrs-cert-<cluster_name>」) があります。この証明書はAstra Data Store オペレータによって管理され、有効期限の7日前にオペレータが証明書を自動的に更新します。証明書を手動で取り消すこともできます。

証明書を取り消します

Astra データストアコントローラ、ノード、またはCA証明書が侵害された場合、MTLSシークレットを削除することでCA証明書を取り消すことができます。これを行うと、Astra Data Storeオペレータは自動的に新しい証明書を発行します。Astra データストア証明書はいつでも取り消すことができます。



CA証明書を取り消すと、そのCAによって署名された証明書がすべて取り消されます。

手順

1. Astra Data Storeクラスタのコントローラノードにログインします。
2. システム上の既存の証明書の一覧を表示します。例：

```
kubectl get secrets -n astrads-system | grep astrads-cert
```

次のような出力が表示されます。

```

astrads-cert-astrads-cluster-controller
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-ds-dms-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-ds-support-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-support-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-root
kubernetes.io/tls      4      6d6h
astrads-cert-sti-net-com
kubernetes.io/tls      5      6d6h

```

- 出力には、取り消す必要がある証明書の名前が表示されます。
- 'kubectl'ユーティリティを使用して'証明書を取消しますこれは'certificate_name'を証明書の名前に置き換えます例：

```
kubectl delete secret CERTIFICATE_NAME -n astrads-system
```

既存の証明書が失効し、代わりに新しい証明書が自動的に生成されます。

外部キーを管理します

1 つ以上の外部キー管理サーバを使用して、暗号化されたデータにアクセスする際にクラスタで使用するキーを安全に保管できます。外部キー管理サーバはストレージ環境に配置されたサードパーティのシステムで、Key Management Interoperability Protocol (KMIP) を使用してノードにキーを提供します。



Astraデータストアでは、Astraデータストアクラスタを作成すると、デフォルトで内部キープロバイダを使用して保存データの暗号化 (sear) が有効になります。

キーの管理には、次のカスタムリソース定義 (CRD) が含まれます。

- * AstraDSKeyProvider*：外部KMIPサーバを設定します。このサーバはサーバのクラスタの場合があります。
- * AstraDSSEARKeyRotate*：キープロバイダから新しいキー暗号化キーを取得し、Astraデータストアに提供します。

外部キー管理に関連して次のタスクを実行できます。

- [\[Set up external key management\]](#)
- [\[Check the software encryption at rest status\]](#)
- [\[Change external to internal key management\]](#)

- [\[Rotate keys for security\]](#)

外部キー管理をセットアップする

Astra Data Storeで外部キー管理を設定するには'kubectl astrs'コマンドを使用します

クラスタまたはKMIPサーバにSSL証明書が必要です。これにより、OpenSSLなどを使用した外部キーの設定などが可能になります。

手順

1. キープロバイダクライアントの証明書を準備します。クライアント証明書、クライアント秘密鍵、および信頼CAバンドルが含まれます。



クラスタまたはKMIPサーバで、OpenSSLなどを使用した外部キーの設定を可能にするSSL証明書を準備します。

2. Astraデータストアクラスタのいずれかのノードにログインします。
3. 次のkubectl拡張コマンドを入力して、Astraデータストアクラスタのキープロバイダを設定します。

```
kubectl-astrads key-provider certs --key key.pem
--client-cert client_cert.pem --ca-cert server_ca.pem
--hostnames=<kmip_server_ip> <key_provider_cr_name>
--namespace astrads-system --cluster <ads_cluster_name>
```

次の例では、ADSクラスタ「astrs-cluster-f23d158」に対して「hashicorp」という名前の外部キープロバイダを設定します。

```
kubectl-astrads key-provider certs --key key.pem
--client-cert client_cert.pem --ca-cert server_ca.pem
--hostnames=10.235.nnn.nnn hashicorp
--namespace astrads-system --cluster astrads-cluster-f23d158
```

1. Astra Data StoreクラスタをAstraDSCluster CR経由で、外部キーマネージャを使用するように設定します。ヘルプを表示します。

```
kubectl-astrads clusters sears -h
```

対応：

Configure SEARS in AstraDS cluster

Usage:

```
astrads clusters sears [flags]
```

Flags:

```
-d, --duration string    Duration for key rotation (default "2160h")
-h, --help              help for sears
```

Global Flags:

```
--ads-cluster-name string      Name of the ADS Cluster
--ads-cluster-namespace string  Namespace of the ADS Cluster
...
```

次のコマンドは'Astra Data Store'クラスターを'AstraDSKeyProvider hashicorp'をsearのキー管理ツールとして使用するよう設定します。また、キーのローテーション時間も使用されます。この時間のデフォルト値は90日(2160時間)です。

```
kubectl-astrads clusters sears -d 500h hashicorp
--ads-cluster-name=astrads-cluster-f23d158
--ads-cluster-namespace=astrads-system
```

ソフトウェアの保存データの暗号化ステータスを確認します

保存データのソフトウェア暗号化の設定を確認できます。

ステップ

1. AstraDSCluster CRを確認します。


```

Name:          astrads-cluster-f23d158
Namespace:     astrads-system
Labels:        <none>
Annotations:   <none>
API Version:   astrads.netapp.io/v1beta1
Kind:          AstraDSCluster
...
Spec:
...
  Software Encryption At Rest:
    Ads Key Provider:      hashicorp
    Key Rotation Period:   500h0m0s
...
Status:
...
  Software Encryption At Rest Status:
    Key Active Time:       2022-05-16T15:53:47Z
    Key Provider Name:     hashicorp
    Key Provider UUID:     ccfc2b0b-dd98-5ca4-b778-99debef83550
    Key UUID:              nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn

```

外部キー管理を内部キー管理に変更します

現在外部キー管理ツールを使用している場合は、内部キー管理ツールに変更できます。

手順

1. SoftwareEncryptionAtRest設定を削除してAstraDSCluster CRを変更します。
2. (オプション) 前のAstraeDSKeyProviderと関連付けられている秘密を削除します。



以前のキープロバイダとシークレットは自動的に削除されません。

キーをローテーションしてセキュリティを確保します

キーのローテーションにより、セキュリティが向上します。デフォルトでは、Astraデータストアはキーを90日ごとに自動的にローテーションします。デフォルト設定を変更できます。また、必要に応じてキーをオンデマンドでローテーションすることもできます。

自動キーローテーションを設定する

1. CRDのAstraeSSEARKeyRotateパラメータを更新します。

```

kubectl patch astradscluster astrads-cluster-f23d158
-n astrads-system
--type=merge -p '{"spec": {"softwareEncryptionAtRest": {
"keyRotationPeriod": "3000h"}}}'

```

オンデマンドのキーローテーションを設定する

1. AstraatDSSEARKeyRotateRequest CRを作成してキーを回転します。

```
cat << EOF | kubectl apply -f -
apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSSEARKeyRotateRequest
metadata:
  name: manual
  namespace: astrads-system
spec:
  cluster: astrads-cluster-f23d158
EOF
```

Astraデータストアライセンスを更新

Astraデータストアにインストールされた評価用ライセンスを更新して、評価期間を延長できます。ライセンスは、次の3つの方法のいずれかを使用して更新できます。

- Astra Control Centerを使用してAstraデータストアライセンスを更新するには、を参照してください ["ストレーజバックエンドライセンスを更新する"](#)。
- Astra VMwareプラグインを使用してAstraデータストアライセンスを更新するには、を参照してください ["VMwareでAstraデータストアを管理"](#)。
- コマンドラインを使用してAstraデータストアライセンスを更新するには、を参照してください [\[Update the Astra Data Store license using the command line\]](#)。

コマンドラインを使用してAstraデータストアライセンスを更新

'kubectl'ユーティリティを使用して'Astra Data Storeライセンスを更新できます

手順

1. ネットアップから入手した交換用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、クラスタの名前（「<Astra-Data-Store-cluster-name>」）とライセンスファイルのパス（「<file_path/file.txt」）を入力します。

```
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
```

次のような応答が表示されます。

NAME	ADSCUSTER	VALID	PRODUCT
EVALUATION	ENDDATE	VALIDATED	
p100000006	astrads-example-cluster	true	Astra Data Store
2023-01-23	2022-04-04T14:38:54Z		true

Astraデータストアをアップグレード

Astraデータストアをアップグレードすると、最新の機能や修正を利用できます。Astra Data Storeの拡張機能を使用して'Astra Data Storeをアップグレードできます

kubectlを使用してAstraデータストアをアップグレード

Astra Data Storeの「kubectl」拡張機能を使用して、Astraデータストアをアップグレードできます。

Astra Data Storeバンドルをダウンロードしてイメージを展開

手順

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')
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 -vxzf <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

手順

1. 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Kubernetes kubectlバイナリがインストールされている標準パスにコピーします(/usr/bin/は下の例の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. 次の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、ローカルレジストリにイメージをプッシュします。

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

アップグレードを実行する

手順

1. 'astraadsoperator.yaml'ファイルをローカルディレクトリにコピーします

```
cp /PATH/TO/FILE/astradsoperator.yaml ./
```

2. オペレータをアップグレードします。大文字の引数を環境に適した情報に置き換えます。

```
kubectl-astrads upgrade ads-operator --repository-url REPOSITORY_URL
--operator-yaml astradsoperator.yaml
```

3. Astraデータストアのアップグレードを開始大文字の引数を環境に適した情報に置き換えます。

```
kubectl-astrads upgrade ads-version --repository-url REPOSITORY_URL
--ads-version-yaml ./astrads/manifests/astradsversion.yaml
```

アップグレードが開始されたことを示すメッセージが表示され、完了までに数分かかります。

Astraデータストアを自動化されたスクリプトでアンインストールします

Astraデータストアとコントロールプレーンをアンインストールするには、ワークロード、バインド、ボリューム、エクスポートポリシー、Astraデータストアクラスタ、ライセンス、導入環境、およびAstraデータストアネームスペースを削除します。

アンインストールにはさまざまな方法があります。

- [\[Uninstall Astra Data Store with an automated script\]](#)
- [\[Uninstall Astra Data Store manually without a script\]](#)
- [\[Troubleshoot the Astra Data Store uninstall process\]](#)

Astraデータストアを自動化されたスクリプトでアンインストールします

このプロセスでは、自動スクリプトを使用してAstraデータストアをアンインストールします。

必要なもの

- root 管理権限

Astra Data Storeのアンインストールプロセスでは、次の手順を実行できます。

- [\[Remove existing workloads and bindings\]](#)
- [\[Uninstall Astra Data Store cluster\]](#)
- [\[Validate the removal of the astrads-system namespace\]](#)
- [\[Ensure containers are not running on worker nodes\]](#)
- [\[Delete OpenShift Container Platform resources\]](#)

既存のワークロードとバインドを削除します

Astraデータストアをアンインストールする前に、次のものを削除する必要があります

- Astraデータストアをストレージバックエンドとして使用するすべてのアプリケーションワークロード

- バックエンドとしてAstraデータストアを使用するTridentバインディング

これにより、Kubernetes 環境をクリーンな状態のまま維持できます。これは、を再インストールする場合に重要です。

Astra データストアクラスタをアンインストールします

Astra Data Storeをアンインストールするには、ネットアップサポートサイトからダウンロードしたAstra Data Store tarファイルで「uninstall.sh」スクリプトを使用します。

1. 'マニフェスト'ディレクトリの'uninstall.sh'を探します
2. 次の「sed」コマンドを実行します。

```
sed -i -e 's~netappdsoperator.yaml~astradsoperator.yaml~' uninstall.sh
```

3. 次のスクリプトを実行して、アンインストールする項目を指定します。

```
./uninstall.sh
```

```
You must run this script with an argument specifying what should be
uninstalled
```

```
To uninstall the ADS cluster run ./uninstall.sh cluster
```

```
To uninstall everything run ./uninstall all
```

4. クラスタのみをアンインストールする場合は、「uninstall.sh <cluster>」と入力します

それ以外の場合は、すべてをアンインストールするには、「uninstall.sh」と入力します



ほとんどの場合、すべてをアンインストールします。その後クラスタを再導入する場合は、クラスタだけをアンインストールすることもできます。

5. プロンプトで続行することを確認し、「erasedata」と入力します

対応：

```
./uninstall.sh all
```

```
Enter 'erasedata' to confirm you want proceed with the uninstall:
```

```
erasedata
```

```
+-----+
```

```
| Wed Feb  2 10:14:01 EST 2022 |
```

```
| ADS cluster uninstall started |
```

```
+-----+
```

```
Deleting astradsvolumes
```

```
Deleted astradsvolumes
```

```

Deleting astradsexportpolicies
Deleted astradsexportpolicies
Deleting astradsvolumesnapshots
Deleted astradsvolumesnapshots
Deleting astradsclusters
Deleting astradsclusters
Deleting astradslicenses
Deleted astradslicenses

+-----+
| Wed Feb  2 10:15:18 EST 2022 |
| ADS cluster uninstall done   |
+-----+

+-----+
| Wed Feb  2 10:15:18 EST 2022 |
| ADS system uninstall started  |
+-----+

Removing astradsversion
astradsversion.astrads.netapp.io "astradsversion" deleted
Removed astradsversion
Removing daemonsets
daemonset.apps "astrads-ds-nodeinfo-astradsversion" deleted
Removed daemonsets
Removing deployments
deployment.apps "astrads-cluster-controller" deleted
deployment.apps "astrads-license-controller" deleted
deployment.apps "astrads-operator" deleted
Removed deployments
Removing all other AstraDS resources
namespace "astrads-system" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsautosupports.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradscloudsnapshots.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsclusters.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsexportpolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsfaileddrives.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradslicenses.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnfsoptions.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io

```

```
"astradsnodeinfoes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnodemanagements.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsqospolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsversions.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumefiles.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumesnapshots.astrads.netapp.io" deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-admin-role"
deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-reader-role"
deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-writer-role"
deleted
role.rbac.authorization.k8s.io "astrads-leader-election-role" deleted
role.rbac.authorization.k8s.io "astrads-manager-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-admin-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-reader-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-writer-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsautosupport-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsautosupport-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsexportpolicy-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsexportpolicy-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsfaileddrive-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsfaileddrive-
```



```
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnfsoption-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnfsoption-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodeinfo-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodeinfo-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodemanagement-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodemanagement-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsqospolicy-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsversion-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsversion-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumefile-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumefile-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumesnapshot-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumesnapshot-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-manager-role" deleted
rolebinding.rbac.authorization.k8s.io "astrads-astrads-admin-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-astrads-reader-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-astrads-writer-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-leader-election-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-manager-rolebinding"
deleted
```

```

clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-admin-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-reader-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-writer-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-manager-
rolebinding" deleted
configmap "astrads-autosupport-cm" deleted
configmap "astrads-firetap-cm" deleted
configmap "astrads-kevents-asup" deleted
configmap "astrads-metrics-cm" deleted
secret "astrads-autosupport-certs" deleted
+-----+
| Wed Feb  2 10:16:36 EST 2022 |
| ADS system uninstall done   |
+-----+

```

astrs-system 名前空間の削除を検証します

次のコマンドで結果が返されないことを確認します。

```
kubectl get ns | grep astrads-system
```

ワーカーノードでコンテナが実行されていないことを確認します

'FIRETAAP' や 'netwd' などのコンテナがワーカー・ノードで実行されていないことを確認します各ノードで次のコマンドを実行します。

```
ssh <mynode1>
# runc list
```

OpenShift Container Platform リソースを削除します

Red Hat OpenShift Container Platform (OCP) にAstraデータストアをインストールした場合は、OCPセキュリティコンテキスト制約 (SCC) と役割バインディングリソースをアンインストールできます。

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。

標準のアンインストールプロセスが完了したら、次の手順を実行します。

1. SCC リソースを削除します。

```
oc delete -f ads_privileged_scc.yaml
```

2. ロールバインドリソースを削除します

```
oc delete -f oc_role_bindings.yaml
```



これらの手順で「resources not found」エラーを無視します。

スクリプトを使用せずにAstraデータストアを手動でアンインストールする

このプロセスでは、スクリプトを使用せずにAstraデータストアを手動でアンインストールします。

自動スクリプトを使用せずにAstraデータストアを手動でアンインストールするには、ワークロード、バインド、ボリューム、エクスポートポリシー、クラスタ、ライセンス、導入環境、Astraデータストアネームスペース

必要なもの

- root 管理権限

Astra Data Storeのアンインストールプロセスでは、次の手順を実行できます。

- [\[Remove existing workloads and bindings\]](#)
- [\[Uninstall the Astra Data Store cluster and control plane\]](#)
- [\[Delete the license\]](#)
- [\[Delete the Astra Data Store installation\]](#)
- [\[Validate the removal of the astrads-system namespace\]](#)
- [\[Ensure containers are not running on worker nodes\]](#)
- [\[Delete OpenShift Container Platform resources\]](#)

既存のワークロードとバインドを削除します

Astraデータストアをアンインストールする前に、次のものを削除する必要があります

- Astraデータストアをストレージバックエンドとして使用するすべてのアプリケーションワークロード
- バックエンドとしてAstraデータストアを使用するTridentバインディング

これにより、Kubernetes 環境をクリーンな状態のまま維持できます。これは、を再インストールする場合に重要です。

Astraデータストアクラスタとコントロールプレーンをアンインストールします

Astraデータストアを手動でアンインストールするには、次の手順に従います。

ボリュームとエクスポートポリシーを削除します

クラスタを削除する前に、Astraデータストアボリュームとエクスポートポリシーを削除する必要があります。



ボリュームとエクスポートポリシーを最初に削除しないと、Astra Data Storeボリュームオブジェクトが削除されるまで、クラスタの削除プロセスは一時停止します。クラスタの削除を開始する前に、それらの項目を削除の方が効率的です。

手順

1. ボリュームを削除します。

```
~% kubectl delete astradsvolumes --all -A
~% kubectl get astradsvolumes -A
```

2. エクスポートポリシーを削除します。

```
~% kubectl delete astradsexportpolicies --all -A
~% kubectl get astradsexportpolicies -A
```

Astraデータストアクラスタを削除

クラスタを削除すると、Astra Data Storeクラスタオブジェクトのカスタムリソース（CR）とクラスタを対象としたリソースのみが削除される。



オペレータ、nodeinfo ポッド、およびクラスタコントローラ（Kubernetes を対象としたリソース）は、クラスタを削除しても削除されません。

クラスタを削除すると、基盤となるオペレーティング・システムもノードからアンインストールされ、'FIRETAAP' および 'netwd' サービスが停止します。

アンインストーラが完了するまでに約 1 分かかります。次に、Astraデータストアクラスタを対象としたリソースの削除が開始されます。

1. クラスタを削除します。

```
~% kubectl delete astradsclusters --all -A
~% kubectl get astradsclusters -A
```

ライセンスを削除します

1. クラスタ内の各ワーカー・ノードに SSH 接続し、ワーカー・ノードで 'FIRETAAP' または 'netwd' が実行されていないことを確認します。
2. Astraデータストアライセンスを削除します。

```
~% kubectl delete astradslicenses --all -A
~% kubectl get astradslicenses -A
```

Astraデータストアのインストールを削除

クラスタ内のコントローラ、演算子、ネームスペース、およびサポートポッドを削除します。

1. Astra Data Storeインストールオブジェクトを削除します。

```
~% kubectl delete astradsversion astradsversion -n astrads-system
~% kubectl get astradsversion -n astrads-system
```

2. データストアDemonSetsとすべてのAstraデータストアコントローラリソースを削除します。

```
~% kubectl delete ds --all -n astrads-system
~% kubectl get ds -n astrads-system

~% kubectl delete deployments --all -n astrads-system
~% kubectl get deployments -n astrads-system
```

3. 残りのアーティファクトと演算子 YAML ファイルを削除します。

```
~% kubectl delete -f ./manifests/astradsoperator.yaml
~% kubectl get pods -n astrads-system
```

astrs-system 名前空間の削除を検証します

次のコマンドで結果が返されないことを確認します。

```
~% kubectl get ns | grep astrads-system
```

ワーカーノードでコンテナが実行されていないことを確認します

'FIRETAAP' や 'netwd' などのコンテナがワーカー・ノードで実行されていないことを確認します各ノードで次のコマンドを実行します。

```
ssh <mynode1>
# runc list
```

OpenShift Container Platform リソースを削除します

Red Hat OpenShift Container Platform (OCP) にAstraデータストアをインストールした場合は、OCPセキュリティコンテキスト制約 (SCC) と役割バインディングリソースをアンインストールできます。

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。

標準のアンインストールプロセスが完了したら、次の手順を実行します。

1. SCC リソースを削除します。

```
oc delete -f ads_privileged_scc.yaml
```

2. ロールバインドリソースを削除します

```
oc delete -f oc_role_bindings.yaml
```



これらの手順で「resources not found errors」を無視します。

手動削除のサンプル

次に、手動アンインストールスクリプトの実行例を示します。

```
$ kubectl delete astradsvolumes --all -A
No resources found
$ kubectl delete astradsexportpolicies --all -A
No resources found
$ kubectl delete astradsclusters --all -A
astradscluster.astrads.netapp.io "astrads-sti-c6220-09-10-11-12" deleted

$ kubectl delete astradslicenses --all -A
astradslicense.astrads.netapp.io "e900000005" deleted

$ kubectl delete astradsdeployment astradsdeployment -n astrads-system
astradsdeployment.astrads.netapp.io "astradsdeployment" deleted

$ kubectl delete ds --all -n astrads-system
daemonset.apps "astrads-ds-astrads-sti-c6220-09-10-11-12" deleted
daemonset.apps "astrads-ds-nodeinfo-astradsdeployment" deleted
daemonset.apps "astrads-ds-support" deleted

$ kubectl delete deployments --all -n astrads-system
deployment.apps "astrads-cluster-controller" deleted
deployment.apps "astrads-deployment-support" deleted
```

```

deployment.apps "astrads-license-controller" deleted
deployment.apps "astrads-operator" deleted

$ kubectl delete -f /.../firetap/sds/manifests/netappsdsoperator.yaml
namespace "astrads-system" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsautosupports.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradscloudsnapshots.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsclusters.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsdeployments.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsexportpolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsfaileddrives.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradslicenses.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnfsoptions.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnodeinfoes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsgospolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumefiles.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumesnapshots.astrads.netapp.io" deleted
role.rbac.authorization.k8s.io "astrads-leader-election-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-editor-role"
deleted

```

```

clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-autosupport-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-autosupport-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-manager-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-metrics-reader" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappexportpolicy-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappexportpolicy-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsdeployment-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsdeployment-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnfsoption-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnfsoption-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnodeinfo-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnodeinfo-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-proxy-role" deleted
rolebinding.rbac.authorization.k8s.io "astrads-leader-election-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-manager-rolebinding"
deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-proxy-rolebinding"
deleted
configmap "astrads-autosupport-cm" deleted
configmap "astrads-firetap-cm" deleted
configmap "astrads-fluent-bit-cm" deleted
configmap "astrads-kevents-asup" deleted
configmap "astrads-metrics-cm" deleted
service "astrads-operator-metrics-service" deleted
Error from server (NotFound): error when deleting
"/.../export/firetap/sds/manifests/netappsdsoperator.yaml":
deployments.apps "astrads-operator" not found

$ kubectl get ns | grep astrads-system

```

```

[root@sti-rx2540-535c ~]# runc list

```

ID	PID	STATUS	BUNDLE	CREATED	OWNER
----	-----	--------	--------	---------	-------

Astraデータストアのアンインストールプロセスのトラブルシューティング

アンインストールプロセスのトラブルシューティングが必要な場合は、次の点を確認してください。

ポッドが終了状態です

原因 ポッドがKubernetesで終了状態のままになる場合、Astraデータストアのアンインストールプロセスが必要になることがあります。

この問題が発生した場合は、次のコマンドを実行して、'astrs-system' 名前空間内のすべてのポッドを強制的に削除します。

```
kubect1 delete pods --all -n astrads-system --force --grace-period 0
```

QoSポリシーは古いクラスタをポイントしています

Astraデータストアクラスタのみを削除して再導入した場合、サービス品質（QoS）ポリシーが古いクラスタをポイントしていて検出できないために、永続的ボリューム要求（PVC）またはボリュームを作成できないことがあります。

1. この問題を回避するには、Astraデータストアクラスタを削除したあとで、QoSポリシーを手動で削除します。

```
kubect1 delete AstraDSQosPolicy --all -A
```

2. （クラスタだけでなく）Astraデータストア環境全体を削除する。

```
uninstall.sh all
```

Astraデータストアを削除またはアンインストールしたあとに、キープロバイダ**CRS**が削除されない

削除またはアンインストールするAstraデータストアクラスタ用に外部キープロバイダが設定されている場合は、削除されていないキープロバイダCRを手動でクリーンアップしなければならないことがあります。

例 2. 詳細

次の回避策 手順を使用します。

手順

1. キープロバイダCRSが削除されていないことを確認します。

```
kubectl get astradskeyprovider --selector  
astrads.netapp.io/cluster=astrads-cluster-example -n astrads-system
```

対応：

NAME	AGE
externalkeyprovider1	94s

2. キープロバイダCRSを削除します。

- a. ファイナライザを取り外します。

```
kubectl edit astradskeyprovider -n astrads-system
```

- b. 以下のハイライトされたファイナライザラインを取り外します。

```
kubectl edit astradskeyprovider externalkeyprovider1 -n astrads-  
system
```

```

apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSKeyProvider
metadata:
  creationTimestamp: "2022-05-24T16:38:27Z"
  finalizers:
    - astrads.netapp.io/astradskeyprovider-finalizer
  generation: 1
  labels:
    astrads.netapp.io/cluster: astrads-cluster-example
    astrads.netapp.io/rsid: "1"
  name: externalkeyprovider1
  namespace: astrads-system
  resourceVersion: "1134699"
  uid: a11111b2-31c0-4575-b7f3-97f9ab1a1bla
spec:
  cluster: astrads-cluster-example
  kmipServer:
    hostnames:
      - 10.xxx.xxx.xxx
    port: 5696
    secretRef: externalkeyprovider1
status:
  keyProviderUUID: a1b2cd34-4fc6-5bae-9184-2288c673181d
  kmipServerStatus:
    capabilities: '{ KMIP_library_version()=17367809,
KMIP_library_version_str()="KMIP
  1.9.3a 8-Apr-2019", KMIP_library_version_tag()="KMIP part
of KMIP 1.9.3a 8-Apr-2019",
  KMIP_library_is_eval()=false,
KMIP_library_fips_capable()=true(FIPS140),
KMIP_SSL_provider_build_version()=268444095,
  KMIP_SSL_provider_version()=268444095,
KMIP_SSL_provider_version_str()="OpenSSL
  1.0.2zb-fips 23 Sep 2021" }'
  keyServerUUID: 8422bdd0-74ad-579d-81bd-6d544ac4224a

```

c. ファイナライザを削除したら、キープロバイダCRを削除します。

```

kubectl delete astradskeyprovider <key-provider-cr-name> -n
astrads-system

```

VMwareでAstraデータストアを使用

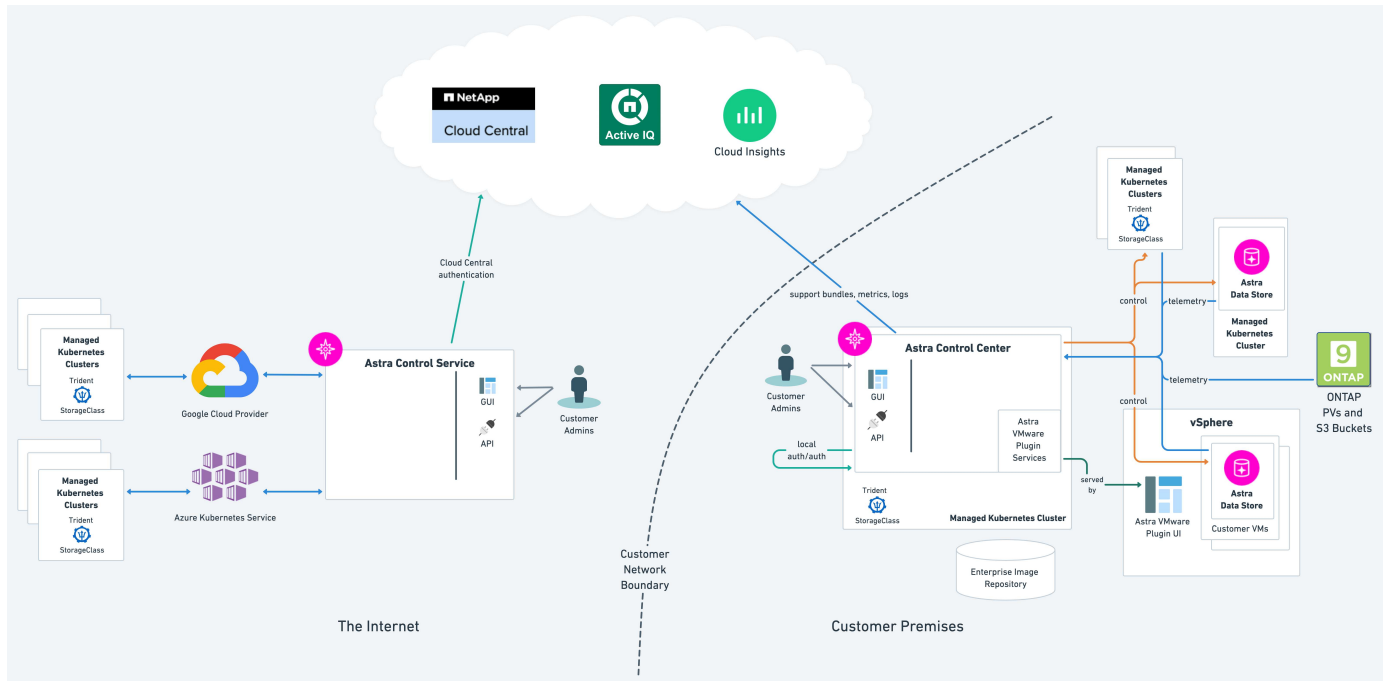
VMwareを使用したAstraデータストアの詳細をご確認ください

Astraデータストアは、コンテナ化ワークロードと仮想化ワークロードの両方をサポートします。VVolおよびストレージポリシーベースの管理と統合することで、vSphere管理者はストレージQoSを適用できます。NetApp Astra Plugin for VMware vSphereは、管理と監視に使い慣れたプラグインで、煩雑なストレージタスクを排除します。

Astra Plugin for VMware vSphereには、次のようなメリットがあります。

- VVOLとVASAの完全な統合による、VM単位のストレージプロビジョニング
- ストレージポリシーベースの管理統合
- vSphereネイティブ管理用のvCenterプラグイン

次の図は、VMwareを使用したAstraファミリーエコシステムを示しています。



を参照してください。

- ["Astra Control Center のドキュメント"](#)
- ["Astra ファミリーの紹介"](#)

VMwareの要件を満たすAstraデータストア

環境が全般的な要件を満たしていることを確認したら ["Astra データストアの要件"](#) を使用している環境が、VMware VASAプロバイダやNetApp Astra Plugin for VMware vSphereなどのVMwareコンポーネントの最小要件を満たしていることを確認してください。

VMware vSphereの要件

Astraデータストアは、VMware VASA Providerをストレージと通信するためのAPIインターフェイスとして使用します。環境が基本的な要件を満たしていることを確認します ["VASA Providerの要件"](#) さらに、次の追加要件があります。

- VMware vSphere 7.0（Update 1～Update 3をサポート）
- 入力トラフィックに割り当てられていないIPアドレスが1つあります



- NetApp Astra Plugin for VMware vSphereはvSphere 7.0 Update 3cをサポートしません。代わりにvSphere 7.0 Update 3Dを使用してください。
- NetApp Astra Plugin for VMware vSphereは、リンクモードのvCenter Serverをサポートしていません。

NetApp Astra Plugin for VMware vSphereの要件

NetApp Astra Plugin for VMware vSphereの要件は次のとおりです。

- Kubernetesクラスタで実行されるAstra Control Centerインスタンス
- Kubernetesクラスタ上で実行される、ライセンス供与されたAstraデータストアインスタンス
- AstraデータストアクラスタでNFSv3を有効化

サポートされている **Web** ブラウザ

NetApp Astra Plugin for VMware vSphereは、次のWebブラウザの最新バージョンをサポートします。

- Mozilla Firefox
- Microsoft Edge（クロムベース）
- Google Chrome

を参照してください。

- ["Astra Control Center のドキュメント"](#)
- ["Astra ファミリーの紹介"](#)
- ["Astra データストアの要件"](#)

VMwareを使用してAstraデータストアをセットアップする

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し、NetApp Astra Plugin for VMware vSphereを使用して管理できます。

VMwareでAstraデータストアをセットアップするには、次のタスクを実行します。

- [\[Activate VMware vSphere services using Astra Control Center\]](#)。
- [\[Add a vCenter using Astra Control Center\]](#)。

- [\[Create a custom SCC \(if using OpenShift\)\]](#)
- [\[Use an existing storage backend in the Astra Plugin for VMware vSphere\]](#)。
- [\[Create a datastore using the Astra Plugin for VMware vSphere\]](#)。
- [\[Generate VM storage policies\]](#)。

VMwareを使用してAstraデータストアをセットアップする前に、次のことを確認する必要があります。

- Astra Control Centerは **"インストール済み"** セットアップも完了しています。
- Astraデータストアのライセンスを取得し、導入している。を参照してください **"Astra データストアをインストール"**。



Astra Data Store Early Access Program (EAP) リリースの場合、Astra Control CenterとAstra Data Storeを2つの異なるKubernetesクラスタに導入する必要があります。

- Astra Control CenterとAstraデータストアの導入に使用したKubernetesクラスタは、Astra Control Centerで管理しておく必要があります。
- NFSv3が有効になります。
- vCenterを追加する前に、Astra Control CenterとVASA Providerのパッケージをアップロードしておきます。 **"ソフトウェアパッケージの管理を参照してください"**。

Astra Control Centerを使用してVMware vSphereサービスをアクティブ化します

Astra Control CenterでvSphereサービスをアクティブにして、VMwareでAstraデータストアのセットアップを開始します。



Astra Control CenterのVMware vSphereサービスは、デフォルトでは有効になっていません。

1. Astra Control Center にログインします。
2. 左側のナビゲーションから、***クラスタ***を選択します。

バナーには、VMware vSphereサービスがまだ有効になっていないことを示すメッセージが表示されます。

3. 「* VMware vSphereサービスを有効にする*」を選択します。

この処理には時間がかかることがあります。サービスを有効にすると、*** vCenterの追加***ボタンが有効になります。

Astra Control Centerを使用してvCenterを追加します

最初のvCenterを追加してAstra Plugin for VMware vSphereを登録します。

vCenterをAstra Control Centerに追加するには、管理者権限が必要です。



プラグインをVMware vSphereに登録すると、VMwareのショートカットページにAstra Plugin for VMware vSphereアイコンが表示されます。

1. Astra Control Center にログインします。
2. 左側のナビゲーションから、*クラスタ*を選択します。
3. 「* vCenterの追加*」を選択します。
4. vCenter Serverの詳細、vCenterポート、および管理ユーザ名とパスワードを入力してAstra Control Center に提供します。



これにより、このvCenter用のAstraプラグインをVMware vSphere Clientに導入できるようになります。

5. 「* 追加」を選択します。

vCenterがクラスタページに表示され、Astra Control Dashboardで管理対象vCenterの総数が更新されます。これにより、Astra Plugin for VMware vSphereの導入も開始されます。

vCenterの追加を確認

新しく追加したvCenterがクラスタページおよびダッシュボードに表示されます。



vCenterとKubernetesクラスタの両方がAstra Control Center Dashboardに表示される。

1. Astra Control Centerにアクセスします。
2. 左側のナビゲーションから、*クラスタ*を選択します。
3. 新しく管理されたvCenterがクラスタページに表示されていることを確認します。
4. 左側のナビゲーションから、* ダッシュボード * を選択します。
5. Astra Control Center Dashboardから、管理対象の*クラスタ数の一部として新しい管理対象vCenterクラスタをメモします。



管理対象クラスタの数には、vCenterとKubernetesクラスタの両方が含まれます。

6. クラスタの詳細を表示するには、* Managed * countをクリックします。

クラスタページが表示されます。

カスタムSCCを作成する（OpenShiftを使用する場合）

OpenShiftを使用している場合は、必要に応じてセキュリティコンテキスト制約（SCC）を割り当てて、ポッドが実行できるアクションを制御したり、ポッドがアクセスできるアクションを制御したりできます。

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、VASA Providerの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、VASA Providerの導入で使用するサービスアカウントに、必要な上位の権限（サンプルに記載）を付与します。

カスタムSCCを、特権およびノードエクスポートSCCのハイブリッドであるAstra Data Store 'NTV-system'ネームスペースのさまざまなデフォルト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. 「VP_backend_privileged_scc.yaml」というカスタムSCCを作成します。

```
kubectl create -f vp_backend_privileged_scc.yaml
```

例：VP_backend_Privileged_SCC.YAML

```
allowHostDirVolumePlugin: true
allowHostIPC: false
allowHostNetwork: true
allowHostPID: false
allowHostPorts: true
allowPrivilegeEscalation: true
allowPrivilegedContainer: true
allowedCapabilities:
  - '*'
allowedUnsafeSysctls:
  - '*'
apiVersion: security.openshift.io/v1
defaultAddCapabilities: null
fsGroup:
  type: RunAsAny
groups: []
kind: SecurityContextConstraints
metadata:
  name: vpbackend-privileged
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities: null
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: RunAsAny
seccompProfiles:
  - '*'
supplementalGroups:
  type: RunAsAny
users:
  - system:serviceaccount:ntv-system:default
  - system:serviceaccount:ntv-system:ntv-auth-svc
  - system:serviceaccount:ntv-system:ntv-autosupport
  - system:serviceaccount:ntv-system:ntv-compliance-svc
  - system:serviceaccount:ntv-system:ntv-datastore-svc
  - system:serviceaccount:ntv-system:ntv-metallb-controller
```



```

- system:serviceaccount:ntv-system:ntv-metallb-speaker
- system:serviceaccount:ntv-system:ntv-mongodb
- system:serviceaccount:ntv-system:ntv-nfs-svc
- system:serviceaccount:ntv-system:ntv-rabbitmq-svc
- system:serviceaccount:ntv-system:ntv-storage-svc
- system:serviceaccount:ntv-system:ntv-vault
- system:serviceaccount:ntv-system:ntv-vault-admin
- system:serviceaccount:ntv-system:ntv-vault-agent-injector
- system:serviceaccount:ntv-system:ntv-vault-controller
- system:serviceaccount:ntv-system:ntv-vault-initializer
- system:serviceaccount:ntv-system:ntv-vcenter-svc
- system:serviceaccount:ntv-system:ntv-vm-management-svc
- system:serviceaccount:ntv-system:ntv-watcher-svc
- system:serviceaccount:ntv-system:ntv-vault-sa-vault-tls
- system:serviceaccount:ntv-system:ntv-gateway-svc
- system:serviceaccount:ntv-system:ntv-jobmanager-svc
- system:serviceaccount:ntv-system:ntv-vasa-svc
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 vpbakend-privileged
```

対応：

NAME	PRIV	CAPS	SELINUX	RUNASUSER	FSGROUP	SUPGROUP
PRIORITY	READONLYROOTFS	VOLUMES				
vpbakend-privileged	true	["*"]	RunAsAny	RunAsAny	RunAsAny	RunAsAny
<no value>	false	["*"]				

Astra Plugin for VMware vSphereで既存のストレージバックエンドを使用します

Astra Control Center UIを使用してvCenterを追加したあと、Astra Data StoreストレージバックエンドをAstra Plugin for VMware vSphereを使用して追加します。

このプロセスで完了する操作は次のとおりです。

- 選択したvCenterに既存のストレージバックエンドを追加します。
- 選択したvCenterにVASA Providerを登録します。VASAプロバイダは、VMwareとAstraデータストアの間の通信を提供します。
- VASA Providerの自己署名証明書をストレージバックエンドに追加します。



追加したvCenterがストレージバックエンドウィザードに表示されるまでに10分かかることがあります。



Astraデータストアを複数のvCenterと共有しない。

手順

1. NetApp Astra Plugin for VMware vSphereにアクセスします。
2. 左側のナビゲーションから、「* Astra Plugin for VMware vSphere」を選択するか、ショートカットページから「Astra Plugin for VMware vSphere *」アイコンを選択します。
3. Astra Plugin for VMware vSphereの概要ページで、既存のストレージバックエンドを使用する*を選択します。または、左のナビゲーションから Storage Backends > Add を選択し、Use existing storage backend *を選択します。
4. ストレージバックエンドとして既存のAstraデータストアを選択し、「次へ」を選択します。
5. VASA Providerのページで、VASA Providerの名前、IPアドレス（ロードバランサを使用している場合）、ユーザ名、パスワードを入力します。



ユーザ名には、英数字とアンダースコアを使用できます。特殊文字は入力しないでください。ユーザ名の先頭の文字はアルファベットにする必要があります。

6. ロードバランサを導入してIPアドレスを入力するかどうかを指定します。このIPアドレスを使用してVASA Providerにアクセスします。ノードIPとは別の、ルーティング可能な追加のフリーIPであることが必要です。ロードバランサを有効にすると、KubernetesクラスターAstraにMetalLBが導入され、空いているIPを割り当てるように設定されます。



Google Anthosクラスターを導入する場合、Anthosではすでにメタリがロードバランサとして実行されているため、ロードバランサを導入しないように選択します。VASA Provider CRでmetalLB deployフラグをfalseに設定します（v1beta1_vasaprovider.yaml）。

ロードバランサを導入しない場合は、ロードバランサがすでに導入され、タイプ*ロードバランサ*のKubernetesサービスにIPを割り当てるように設定されているとみなされます。



この時点では、VASA Providerは導入されていません。

7. 「* 次へ *」を選択します。
8. [証明書]ページで、自己署名証明書の証明書情報を確認します。
9. 「* 次へ *」を選択します。
10. 概要情報を確認します。
11. 「* 追加」を選択します。

VASA Providerが導入されます。

Astra Plugin for VMware vSphereでストレージバックエンドを確認します

Astra Data Storeストレージバックエンドが登録されると、Astra Plugin for VMware vSphereストレージバックエンドのリストに表示されます。

ストレージのバックエンドステータスとVASA Providerのステータスを確認できます。各ストレージバックエンドの使用済み容量も確認できます。

ストレージバックエンドを選択すると、使用済み容量と使用可能容量、データ削減率、および内部ネットワーク管理IPアドレスも表示されます。

手順

1. NetApp Astra Plugin for VMware vSphereの左側のナビゲーションから、「* Storage Backends」を選択します。
2. Astra Data Storeストレージバックエンドを選択すると、[Summary]タブが表示されます。
3. VASA Providerの使用済み容量と使用可能容量、データ削減比率、およびステータスを確認します。
4. その他のタブを選択して、VM、データストア、ホスト、およびストレージノードに関する情報を表示します。

Astra Plugin for VMware vSphereを使用してデータストアを作成します

ストレージバックエンドを追加してAstra Plugin for VMware vSphereを登録したら、VMwareでデータストアを作成できます。

データストアは、データセンター、コンピューティング、またはホストクラスタに追加できます。



同じストレージバックエンドを使用して、同じデータセンターに複数のデータストアを作成することはできません。

NFSプロトコルを使用して、VVOLデータストアタイプを追加できます。

手順

1. Astra Plugin for VMware vSphereにアクセスします。
2. プラグインメニューから、*データストアの作成*を選択します。
3. 新しいデータストアの名前、タイプ (VVOL) 、プロトコル (NFS) を入力します。
4. 「* 次へ *」を選択します。
5. Storage (ストレージ) ページで、作成したAstra Data Storeストレージバックエンドを選択します。



既存のデータストアがあるストレージバックエンドは使用できません。

6. 「* 次へ *」を選択します。
7. 概要ページで、情報を確認します。
8. 「* Create *」を選択します。



スキャンの失敗または一般的なシステムエラーに関連するエラーが発生した場合は、["vCenterでストレージプロバイダを再スキャン/同期します"](#) 次に、データストアの作成をもう一度実行してください。

VMストレージポリシーを生成する

データストアを作成したあと、VMを作成する前に、REST API UIで「/virtualization/v1/v1/vCenters /vm-storage-policies」を使用して、事前設計済みのVMストレージポリシーを生成する必要があります。

手順

1. 「https://<ads_gateway_ip>:8443」にアクセスして、REST API UIページにアクセスします。
2. APIの「POST/virtualization/api/auth/login」に移動し、ユーザ名、パスワード、およびvCenterホスト名を入力します。

対応：

```
{
  "vmware-api-session-id": "212f4d6447b05586ab1509a76c6e7da56d29cc5b",
  "vcenter-guid": "8e475060-b3c8-4267-bf0f-9d472d592d39"
}
```

3. APIのget /virtualization/api/auth/validate-sessionに移動し、次の手順を実行します
 - a. 上記で生成された「vmware-api-session-id」と「vcenter-guid」をヘッダーとして使用します。
 - b. [今すぐ試す]を選択します。

応答：（以下の認証は省略されています）：

```
authorization: eyJhbGciOiJSUzI1NiIsInR...9h15DYYvClT3oA connection:
keep-alive content-type: application/json date: Wed,18 May 2022
13:31:18 GMT server: nginx transfer-encoding: chunked
```

4. 前の応答で生成されたベアラートークンを「authorization/api/v1/vCenters /vm-storage-policies」に移動して「authorization」として追加します。

「200」と表示され、3つのVMストレージポリシーが生成されます。

5. vCenter Storage Policyページで、新しいVMストレージポリシー（Bronze、Silver、Gold）を確認します。
6. VMを作成して続行します。

次の手順

次に、次のタスクを実行します。

- VMを作成する
- データストアをマウントを参照してください ["データストアをマウント"](#)。

を参照してください。

- ["Astra Control Center のドキュメント"](#)
- ["Astra ファミリーの紹介"](#)

VMware環境のコンポーネントを監視する

NetApp Astra Plugin for VMware vSphereを使用すると、Astraデータストアインストール環境のコンポーネントを監視できます。ストレージバックエンド、VASAプロバイダ、VM、VVolなど、システムの健全性を監視できます。容量やvCenterの情報を表示することもできます。

Astra Plugin for VMware vSphere Dashboardを使用して、システムのヘルスを監視します

VMware環境でAstraデータストアを管理するには、ストレージバックエンドとVASAプロバイダの全体的な健全性を監視する必要があります。

NetApp Astra Plugin for VMware vSphere Dashboardを使用すると、次の情報を確認できます。

- このvCenter内のすべてのストレージバックエンドの使用済み物理容量と使用可能容量。情報にカーソルを合わせると、詳細を確認できます。
- 健全で正常でないストレージバックエンドとVASAプロバイダ
- 上位10個のVMのレイテンシ、IOPS、スループット、容量利用率。

ダッシュボードでは、さらにいくつかのタスクを実行できます。

- 容量を監視
- 既存のストレージバックエンドを使用する。を参照してください ["ストレージバックエンドをセットアップ"](#)。
- 製品ドキュメントにアクセスする

ダッシュボードを確認する手順

1. Astra Plugin for VMware vSphereにアクセスします。
2. 概要ページで、次のセクションを確認します。
 - a. *ストレージバックエンド*セクション：ストレージバックエンドとVASAプロバイダの両方の状態をクリックすると、その状態の詳細を確認できます。をクリックして、すべてのストレージバックエンドを表示することもできます。
 - b. * Storage Backend Capacity *セクション：選択したvCenterのストレージバックエンドの使用済み物理容量と使用可能な容量の合計を確認します。vCenterサーバを変更するには、右上のvCenter Serverオプションをクリックします。
 - c. *仮想マシン*セクション：容量使用率が上位10個の仮想マシンを確認します。



その代わりに、テーブルの見出しをクリックすると、レイテンシが高い上位10個のVMなどの目的を表示できます。

Astraデータストアを他のビューで監視する手順

1. Astraデータストアのコンポーネントを監視するには、次のビューにアクセスします。
 - * Virtual Machines *タブ：Astraデータストアで管理されているすべてのVMをダッシュボードと比較して表示。このダッシュボードには、上位10個のVMのみが表示されます。
 - * Storage * drill down：ストレージ・システムに関連づけられているホスト、仮想マシン、データストアを表示します。
 - * VM Storage *ビュー：VASA Providerで作成されたVVOLの詳細情報が表示されます。

ストレージバックエンドのしきい値設定を確認します

ストレージバックエンドの容量のしきい値設定によって、ストレージバックエンド内のすべてのデータストアにアラート通知が表示されるタイミングが決まります。

Astra Plugin for VMware vSphereを使用してストレージバックエンドを導入または追加する場合、デフォルトのしきい値は次のとおりです。

- 90%使用されると、赤のアラートが生成されます
- 80%フルの場合、黄色のアラートが生成されます

VMwareでアラートが生成されるレベルを確認できます。



Astra Data Store Early Access Programでは、同じストレージコンテナが複数のデータセンターで使用されている場合、データストアに対して誤ったアラームが表示されることがあります。

手順

1. NetApp Astra Plugin for VMware vSphereにアクセスします。
2. 左側のナビゲーションから、「* Settings (*設定)」*を選択します。
3. 設定しきい値を確認します。

を参照してください。

- ["Astra Control Center のドキュメント"](#)
- ["Astra ファミリーの紹介"](#)

VMware環境のAstra Data Storeコンポーネントを管理します

次のAstra Data StoreコンポーネントをvSphere環境とAstra Control Centerから管理できます。

- [\[Work with managed vCenters\]](#)
- [\[Manage VMs from vSphere\]](#)
- [\[Manage the storage backend\]](#)
- [\[Manage datastores\]](#)

管理されたvCenterと連携する

管理対象のvCenterと次の方法で連携できます。

- [\[View vCenter details in Astra Control Center\]](#)
- [\[View vCenter details in Astra Control Center\]](#)
- [\[Unmanage a vCenter in Astra Control Center\]](#)

vCenterの詳細をAstra Control Centerで表示

クラスタに関連付けられているすべてのvCenterを確認できます。

手順

1. Astra Control Centerの左ナビゲーションから、[* Clusters]を選択します。
2. vCenterのリストを表示します。
3. [保存 (Save)]を選択します。

vCenterの詳細をAstra Control Centerで表示

システムやクラスタの健全性の確認が必要になることがあります。管理対象のクラスタ数は、Astra Control Center Dashboardで確認できます。

手順

1. Astra Control Centerの左ナビゲーションから、[* Clusters]を選択します。
2. vCenterを選択します。
3. 情報を表示します。

Astra Control CenterでvCenterの管理を解除します

Astra Control CenterでvCenterを管理する必要がなくなった場合は、管理を解除できます。これにより、vCenterがAstra Control Centerから削除され、登録解除されます。



最初に、このvCenterに関連付けられているクラスタ、ストレージバックエンド、およびVMをAstra Plugin for VMware vSphereから削除する必要があります。

手順

1. Astra Control Centerの左ナビゲーションから、[* Clusters]を選択します。
2. クラスタページでvCenterを選択します。



または、複数のvCenterを選択し、*すべてを管理解除*を選択します。

3. [アクション* (Actions)]メニューを右クリックし、[*管理解除 (Unmanage *)]を選択します。
4. vCenterの管理解除ページで「unmanage」と入力します。
5. 「* Yes, unmanage vCenter*」を選択します。

vSphereからVMを管理します

ネットアップのvSphere標準の処理を使用して、Astraデータストアに関連付けられたVMを管理できます。

- "VMを削除します"
- "VM名を変更します"
- "VMのサイズ変更"



このリリースでは、一度に1つのVMディスクのサイズのみを変更できます。複数のディスクのサイズを変更しようとしても失敗します。

- "VMの電源をオンまたはオフにします"
- "VMを一時停止"
- "VMをリセットします"

vCenterの標準の処理を使用して、Astraデータストアで次のスナップショットワークフローを利用できます。

- "Astraデータストアのスナップショットを作成"
- "Snapshotをリバートする"
- "Snapshot を削除します"



Snapshotの処理が失敗してVVolのランタイムエラーが表示されることがあります。この場合は、処理を再試行してください。

ストレージバックエンドを管理します

ストレージバックエンドを削除できます。ストレージバックエンドを削除しても破棄されるわけではなく、Astraデータストア製品自体は削除されません。VASA ProviderがVMwareから登録解除され、そのvCenterのストレージバックエンドのリンクが解除されるだけです。



VASA Providerが有効になっていてvCenterの外部に導入されている場合は、Astraデータストアのみを削除できます。ストレージバックエンドがデータストアのプロビジョニングプロセスの一環として使用されている場合、ストレージバックエンドを削除できません。

Astraデータストアが複数のvCenterにリンクされていない場合は、削除するとVASA Providerの登録が解除され、アンインストールされます。

手順

1. Astra Plugin for VMware vSphereにアクセスします。
2. 左側のナビゲーションから、「* Storage Backends *」を選択します。
3. Storage Backendsページで、Storage Backend Actionsメニューをクリックし、* Remove *を選択します。
4. VASA Providerのユーザ名とパスワードを入力します。
5. 「* 削除」を選択します。

データストアを管理します

vSphere環境でAstraデータストアを管理するには、vCenterの標準の処理を使用して、VMとAstraプラグインの拡張機能を管理し、データストアを管理します。

- ["データストアを作成"](#)
- [\[Mount a datastore\]](#)
- [\[Delete a datastore\]](#)

データストアをマウント

Astra Plugin for VMware vSphereを使用すると、データストアを追加の1つ以上のホストにマウントできます。

手順

1. vCenterのデータセンターインベントリから、Astraデータストア用のデータストアを選択します。
2. データストアを右クリックし、* Astra Plugin for VMware vSphere > Mount Datastore *を選択します。
3. ホストにデータストアをマウントページで、データストアをマウントするホストを選択します。



データストアをすべてのホストにマウントする場合は、すべてのホストに*マウントをオンにします。

4. [*Mount]を選択します。

操作を開始した後は、vSphere Clientの[最近のタスク]パネルで進行状況を確認できます。



スキンの失敗または一般的なシステムエラーに関連するエラーが発生した場合は、["vCenterでストレージプロバイダを再スキャン/同期します"](#)次に、データストアの作成をもう一度実行してください。

データストアを削除する

Astra Plugin for VMware vSphereを使用して、データストアを削除できます。



データストアを削除するには、データストア上のすべてのVMを削除しておく必要があります。

手順

1. vCenterで、データセンターのインベントリからデータストアを選択します。
2. データストアを右クリックし、* Astra Plugin > Delete Datastore *を選択します。
3. データストアの削除ページで、情報を確認するか、データストアを削除するための推奨される追加の操作を実行します。
4. 「*削除」を選択します。

を参照してください。

- ["Astra Control Center のドキュメント"](#)
- ["Astra ファミリーの紹介"](#)

VMware統合環境からAstraデータストアをアンインストールします

Astra Data Storeとその関連コンポーネントは、vSphere環境からアンインストールできます。

を参照してください ["以下の手順を参照して"](#) Astraデータストアのアンインストール時。

を参照してください。

- ["Astra Control Center のドキュメント"](#)
- ["Astra ファミリーの紹介"](#)

知識とサポート

トラブルシューティング

発生する可能性のある一般的な問題を回避する方法について説明します。

https://kb.netapp.com/Advice_and_Troubleshooting/Cloud_Services/Astra

ヘルプを表示します

ネットアップでは、Astra データストアのサポートをさまざまな方法で提供しています。無償のセルフサービスサポートオプションは、ナレッジベース（KB）記事や Slack チャンネルなど、24 時間 365 日ご利用いただけます。



Astra データストアのコミュニティテクニカルサポートを利用できます。を使用してケースを作成します "ネットアップサポートサイト（NSS）" は、このリリースでは使用できません。フィードバックオプションを通じてサポートに連絡するか、Slack チャンネルを利用してセルフサービスで連絡できます。

セルフサービスサポートオプション

次のオプションは、24 時間 365 日無料で利用できます。

- "ナレッジベース（ログインが必要）"

Astra データストアに関する記事、FAQ、またはトラブルシューティング情報を検索します。

- ドキュメント

現在表示しているドキュメントサイトです。

- "ネットアップの「コンテナ」 Slack チャンネル"

「コンテナ」チャンネルに移動して、同業者やエキスパートと交流しましょう。

- フィードバック用 E メール

astra.feedback@netapp.com に電子メールを送信して、あなたの考え、アイデア、懸念事項をお知らせください。

詳細については、こちらをご覧ください

- "ネットアップにファイルをアップロードする方法（ログインが必要）"
- "ネットアップの技術情報アーティクル"

サポートの自動監視

AutoSupport は、Astra データストアシステムの稼働時間と情報を監視し、ネットアップサポートにメッセージを送信構成に応じて、次のシステムコンポーネントを監視できます。

- コントロールプレーン
- ストレージ

AutoSupport は、の実行時にデフォルトで有効になります "[Astra データストアクラスタのインストール](#)" または、AutoSupport のカスタムリソース（CR）をクラスタに適用したあとに追加します。有効にすると、AutoSupport（ASUP）バンドルが自動的にアップロードされます "[ネットアップサポートサイト（NSS）](#)" または、手動でダウンロードすることもできます。

オプション（Options）

- [\[AutoSupport triggers and scenarios\]](#)
- [\[Configure custom control plane AutoSupport collection\]](#)
- [\[Configure custom storage AutoSupport collection\]](#)
- [\[List ASUPs in the system\]](#)
- [\[Download an ASUP Bundle\]](#)
- [\[Upload a core file\]](#)

AutoSupport のトリガーとシナリオ

AutoSupport バンドルは、次の方法でトリガーされます。

- * 定期的 * : CR で定義された間隔で ASUP バンドルが作成されます。
- * ユーザトリガー * : 自分の ASUP を手動で作成して、ログを参照できます。
- * コアダンプ * : ノードにコアダンプがある場合は ASUP が生成され、コアがネットアップに送信されて詳しい調査が行われます。
- * CallHome イベントベース * : オペレーティングシステムから特定の callhome イベントから ASUP が生成されます。
- * Kubernetes イベントベース * : コントロールプレーンの特定の Kubernetes イベントから ASUP が生成されます。

これらのトリガーシナリオでは、次のいずれかの AutoSupport タイプが生成されます。

- * ControlPlane AutoSupport * : Astra データストアのコントロールプレーンログと CRS のコレクション。
- * Storage AutoSupport * : ストレージ・レポートとパフォーマンス・データの集合。
- * コアダンプ AutoSupport * : システムコアダンプの集まり。

カスタムコントロールプレーンの AutoSupport 収集を設定します

コントロールプレーンイベントをレポートするカスタムの AutoSupport 収集設定を作成できます。ほとんどのインストールでは、定期的なイベントレポートがの間にデフォルトで有効になっています "[Astra データストアクラスタのインストール](#)"。この手順では、選択したパラメータに基づいてレポートする AutoSupport CR を設定する方法について説明します。

手順

1. コントロールプレーンコレクション CR を作成するには、次のコマンドをカスタマイズします。

```
kubectl astrasds asup collect -c controlplane --namespace=astrads-system
```

- a. カスタムパラメータを定義します。

- `<myASUPNAME>` : 生成する AutoSupport CR の名前。
- `-e <event name>` : コレクションをトリガーするイベント名。イベント名は `component.yaml` (コントローラをサポートするためにマウントされている) で事前に定義する必要があります

例

```
kubectl astrasds asup collect -c controlplane custom-asup-name -e debug --namespace=astrads-system
```

- a. 必要に応じて、システムに追加のパラメータを追加します。

- `--cluster` : このフラグはマルチクラスタ環境で必要です。
- `--localCollection` : ローカルコレクションをイネーブルにします。デフォルトは「false」です。
- `--forceUpload` : 強制アップロードを有効にします。デフォルトは「false」です。
- `--retry` : 再試行を有効にしますデフォルトは「false」です。

カスタムのストレージ **AutoSupport** による収集を設定する

ストレージコンポーネントのイベントをレポートするカスタムの AutoSupport 収集設定を作成できます。ほとんどのインストールでは、定期的なイベントレポートがの間にデフォルトで有効になっています "[Astra データストアクラスタのインストール](#)"。この手順では、選択したパラメータに基づいてレポートする AutoSupport CR を設定する方法について説明します。

手順

1. 次のコマンドをカスタマイズして、ストレージ収集 CR を作成します。

```
kubectl astrasds asup collect -c storage --namespace=astrads-system
```

- a. カスタムパラメータを定義します。

- `<myASUPNAME>` : 生成する AutoSupport CR の名前。
- `-e <event name>` : コレクションをトリガーするイベント名。イベント名は `component.yaml` (コントローラをサポートするためにマウントされている) で事前に定義する必要があります

パフォーマンスイベントを使用した例：

```
kubectl-astrads asup collect -c storage -e performance example-perf-storage-asup
```

- ° `-t <iso_format> -d <hours>` :すべてのノードのストレージ ASUP を指定した期間収集します。標準の ISO 日付時刻形式 (-t) を時間単位で継続時間 (d) で使用します例:

```
kubectl astrads asup collect -c storage -t 2021-01-01T15:00:00Z -d 24
```

- ° `--nodes <nodename>` : 指定したノードのストレージ ASUP を収集します。例:

```
kubectl astrads asup collect -c storage --nodes example1
```

- ° `--ノードノード nodename1 、サブグループ 2 、サブグループ 3` : 指定したノードのストレージ ASUP を収集します。

```
kubectl astrads asup collect -c storage --nodes example1,example2,example3
```

a. 必要に応じて、システムに追加のパラメータを追加します。

- ° `--cluster` : このフラグはマルチクラスタ環境で必要です。
- ° `--localCollection` : ローカルコレクションをイネーブルにします。デフォルトは「false」です。
- ° `--forceUpload` : 強制アップロードを有効にします。デフォルトは「false」です。
- ° `--retry` : 再試行を有効にしますデフォルトは「false」です。

システム内の **ASUP** をリストします

次のコマンドを使用して、システム内の ASUP を名前別に表示します。

```
kubectl astrads asup list --namespace=astrads-system
```

回答例:

NAMESPACE	NAME	SEQUENCE NUMBER	EVENT
SIZE	STATE	LOCAL COLLECTION	
astrads-system	storage-callhome.reboot.unknown-...	1	
callhome.reboot.unknown	0	uploaded	astrads-ds-support-tdl2h:
astrads-system	storage-callhome.reboot.unknown-...	2	
callhome.reboot.unknown	0	uploaded	astrads-ds-support-xx6n8:
astrads-system	storage-callhome.reboot.unknown-...	3	
callhome.reboot.unknown	0	uploaded	astrads-ds-support-qghnx:

ASUP バンドルをダウンロード

このコマンドを使用すると、ローカルで収集した ASUP バンドルをダウンロードできます。現在の作業ディレクトリ以外の場所を指定するには '-o <location>' を使用します

```
./kubectl-astrasds asup download <ASUP_bundle_name> -o <location>
```

コアファイルをアップロードします

サービスがクラッシュすると、クラッシュ（コアファイル）時に関連するメモリの内容を含むファイルとともに AutoSupport（ASUP）メッセージが作成されます。ASUP メッセージは Astra データストアからネットアップサポートに自動的にアップロードされますが、コアファイルを手動でアップロードして ASUP メッセージに関連付ける必要があります。

手順

1. 次の「kubectl」コマンドを使用して ASUP メッセージを表示します。

```
kubectl astrasds asup list --namespace=astrads-system
```

次のような出力が表示されます。

NAMESPACE	NAME	SEQUENCE NUMBER	EVENT
SIZE	STATE	LOCAL COLLECTION	
astrads-system	storage-coredump-2021...	1	coredump
197848373	compressed	astrads-ds-support-sxxn7:/var/...	

2. 次の「kubectl」コマンドを使用して、ASUP メッセージからコアファイルをダウンロードします。ダウンロードするファイルの保存先ディレクトリを指定するには '-o オプション' を使用します

```
kubectl astrads asup download storage-coredump-20211216t140851311961680  
-o <absolute_path_to_destination_directory>
```



まれに、他のコアファイルが適切に処理されていたために、コアファイルをダウンロードできない場合があります。この場合、コマンドは「Cannot stat : No such file or directory」というエラーを返します。このエラーが表示された場合は、を実行できます "[ヘルプを表示します](#)"。

3. Web ブラウザを開き、を参照します "[NetApp Authenticated File Upload ツール](#)"ログインしていない場合は、ネットアップサポートのクレデンシャルを入力します。
4. [ケース番号を持たない *] チェックボックスをオンにします。
5. [* Closest Region] * メニューで、最も近いリージョンを選択します。
6. [* Upload (アップロード)] ボタンを選択します。
7. 前の手順でダウンロードしたコアファイルを参照して選択します。

アップロードが開始されます。アップロードが完了すると、成功のメッセージが表示されます。

詳細については、こちらをご覧ください

- "[ネットアップにファイルをアップロードする方法（ログインが必要）](#)"

旧バージョンの**Astra**データストア

以前のリリースのドキュメントを参照できます。

- ["Astra Data Store 21.12ドキュメント"](#)

法的通知

著作権に関する声明、商標、特許などにアクセスできます。

著作権

<http://www.netapp.com/us/legal/copyright.aspx>

商標

NetApp、NetApp のロゴ、および NetApp の商標ページに記載されているマークは、NetApp, Inc. の商標です。その他の会社名および製品名は、それぞれの所有者の商標である場合があります。

<http://www.netapp.com/us/legal/netapptmlist.aspx>

特許

ネットアップが所有する特許の最新リストは、次のサイトで入手できます。

<https://www.netapp.com/us/media/patents-page.pdf>

プライバシーポリシー

<https://www.netapp.com/us/legal/privacypolicy/index.aspx>

オープンソース

通知ファイルには、ネットアップソフトウェアで使用されるサードパーティの著作権およびライセンスに関する情報が記載されています。

["Astra Data Store EAPのリリースに関する注意事項"](#)

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。